

Comparison of Electromagnetic Data With Irregular or Discontinuous Surfaces Using the Feature Selective Validation Method

Amruthavarshini Subburaya Bharathi

P16210886

Faculty of Computing, Engineering and Media

May 2022

*A thesis submitted in fulfilment of the University's requirements
for the Degree of Doctor of Philosophy*

Under the supervision of

Prof. Alistair Duffy



AUTHOR'S DECLARATION

To the best of my knowledge, I confirm that the work in this thesis is my original work undertaken for the degree of PhD in the Faculty of CEM, De Montfort University. I confirm that no material of this thesis has been submitted for any other degree or qualification at any other university.

Amruthavarshini Subburaya Bharathi

Leicester

May 2022

ABSTRACT

In the field of Computational Electromagnetics, validation is a formal process to ensure the expected behaviour of the model. The Feature Selective Validation (FSV) method was originally developed to aid the validation of computational electromagnetics, and particularly electromagnetic compatibility (EMC). Since then, it has been adopted by the IEEE Standard for Validation of Computational Electromagnetics Computer Modelling and Simulations 1597.1 and used in a variety of other applications. The FSV method quantifies the difference between two sets of original data using a reliability function based on the decomposition of the data into a number of component parts that are then combined using a weighted scheme, giving a number of presentations of the comparison data. From the literature review, it is identified that the FSV method has been applied for various structured data like the S-parameter, radiation pattern, efficiency, and gain of an antenna. Since the original development of the FSV in the field of computational electromagnetics, more complex data like surface current, electric, and magnetic field outputs from Ultra High Frequency (UHF) devices are represented in 2- (or higher) dimensional image formats with irregular shapes, which are non-rectangular structures including features such as spaces or gaps within the device structure. However, performing FSV on such image data is challenging, particularly to avoid non-contributing spaces or voids in the image structure dominating the comparison results. This thesis discusses in detail a methodology developed to perform 2-Dimensional FSV on 2-Dimensional regular (rectangular and square) shaped images by segmenting the image into multiple blocks. In each block, 2-Dimensional FSV is performed separately, and then the outputs from the segmented blocks are concatenated to form the original 2-Dimensional image. Finally, the FSV outputs from the segmented approach are compared with the FSV outputs obtained from the original (full) structure, and the results are analysed using a non-parametric statistical test, which shows that the approach leads to results that support the proposed solution of comparison by segmentation and recombination. The proposed method is demonstrated on regular and irregular images to allow a detailed analysis.

ACKNOWLEDGEMENTS

First and foremost, I want to thank my supervisor, Professor Alistair Duffy, for his immense knowledge, enduring opinions, encouragement and continuous support during the research study and writing of this thesis. His guidance was invaluable.

I'd also like to express my gratitude to Dr. Ammar Ghazal, my second supervisor, for his tremendous guidance, and support throughout my PhD programme.

I want to express my gratitude to my mother, father and sister for their support throughout this research. I can't thank my friends enough for their unwavering support throughout this intense writing year.

Last but not least, a special thanks to Nanda Sankar, my dearest friend and husband; I couldn't have done it without him. I am so grateful for all his help, patience and encouragement; without it, this thesis would have not been possible. Many thanks to him once again!

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION	01
1.1 Overview of The Thesis	03
1.2 Aim	04
1.3 Objectives	04
1.4 Organization of the Research Work	04
1.5 References	05
CHAPTER 2: LITERATURE REVIEW	06
2.1 Fundamentals of an Image	07
2.1.1 Pixel	09
2.1.2 Dimension	09
2.1.3 Bit Depth	09
2.1.4 Colour Model	10
2.1.4.1 Black and White Colour Model	10
2.1.4.2 Greyscale Colour Model	10
2.1.4.4 RGB Colour Model	11
2.1.4.4 CMYK Colour Model	12
2.1.5 Raster and Vector Images	12
2.1.5.1 Vector Images	13
2.1.5.2 Raster Images	13
2.1.6 File Formats	13
2.1.6.1 JPEG	14
2.1.6.2 BMP	14
2.1.6.3 GIF	14
2.1.6.4 PNG	15
2.1.6.5 TIFF	15
2.2 Image Processing	16
2.2.1 Enhancement	16
2.2.2 Segmentation	17
2.2.3 Analysis	18
2.3 Introduction to Feature Selective Validation	20
2.3.1 Correlation	22

2.3.2 Reliability Factor	22
2.4 What is FSV?	24
2.5 Multidimensional FSV	30
2.6 Probability Density Function (PDF)	32
2.7 References	33
CHAPTER 3 : FSV IMPLEMENTATION	40
3.1 Implementation of 1-Dimensional FSV	41
3.1.1 STEP 1: To Read Working Data Set	42
3.1.2 STEP 2: Fast Fourier Transform (FFT)	42
3.1.3 STEP 3: To Calculate the Filter Boundaries (The DC, 40% Cut Off and Break Point)	43
3.1.4 STEP 4: To Obtain Low and High Windows	45
3.1.5 STEP 5: Inverse Fast Fourier Transform (IFFT)	46
3.1.6 STEP 6: To Calculate Difference Measures	47
3.1.6.1 Amplitude Difference Measure (ADM)	47
3.1.6.2 Feature Difference Measure (FDM)	49
3.1.6.3 Global Difference Measure (GDM)	51
3.2 Multi-Dimensional FSV	52
3.3 Implementation of 2-Dimensional FSV	53
3.3.1 STEP 1: Read Input Data	53
3.3.2 STEP 2: To Separate Rows and Column Data	54
3.3.3 STEP 3: Perform 1-D FSV on Each Row and Column Separately	54
3.3.4 STEP 4: Recombine the Rows and Column to bring it back to 2-D	55
3.3.5 STEP 5: Perform 2-D FSV using Weighted Scheme	55
3.4 Probability Density Functions (Statistical Analysis of FSV)	57
3.4.1 PDF Calculations	57
3.4.2 Statistical Analysis	58
3.4.3 The Kolmogorov-Smirnov (KS) Test:	59
3.5 References	60
CHAPTER 4 : PROPOSED METHOD	62
4.1 Data Preparation	65
4.2 Image Conversion	66

4.2.1 Python for Image Conversion	68
4.2.1.1 Read Input Image	70
4.2.1.2 Convert RGB Image to Greyscale Image	70
4.2.1.3 Store Image in to NumPy Array	71
4.2.1.4 Writing NumPy Array to CSV	72
4.2.1.5 Verifying Conversion Method	73
4.3 Image Segmentation	74
4.4 Data Manipulation - Copy Rows & Columns Using Excel VBA Macro	78
4.4.1 Copy Each Rows and Columns from Source Array	82
4.4.2 Create a New Sheet	83
4.4.3 Paste the Copied Row/Column to the New Sheet	83
4.4.4 Save The Sheet to User Specified Filename and Filepath (.csv)	83
4.4.5 Iterate the above Steps for All Rows/Columns in the Source Array	84
4.5 Perform 1-D FSV on Each Row and Column Separated	84
4.6 Data Manipulation to Combine Rows & Columns Using Excel VBA Macro	85
4.6.1 Open Individual Source Sheets	87
4.6.2 Select and Copy Data from the Rows/Columns Outputs	87
4.6.3 Paste to the Destination Sheet	88
4.6.4 Iterate the Above Steps for all the Row/Column Source Sheets	88
4.7 Perform 2-D FSV using Weighted Scheme	88
4.8 References	89
CHAPTER 5 : EXPERIMENT RESULTS AND ANALYSIS	93
5.1 Input Datasets	94
5.2 Stage -1: Result Analysis of 2-D FSV on Regular Image Data Sets	97
5.2.1 FSV Result Analysis for E-Field Input Image Data Sets	97
5.2.2 FSV Results Of E2800_Field (0) vs E2800_Field (2)	99
5.2.3 Approach 1: 2-D FSV Results for Full Structure	99
5.2.4 Approach 2 - 2-D FSV Results for Segmented Structure	101
5.2.5 K-S Test Results for Approach 1 and Approach 2	103
5.2.6 FSV Results of Other E-Field Data Sets	104
5.2.7 FSV Result Analysis for Einstein Image Data Sets	110

5.3 Overall Inference for Stage -1 Results:	115
5.4 Stage - 2: Application and Result Analysis of 2-D FSV on Irregular Image Data Sets	117
5.4.1 FSV Result Analysis for Microstrip Patch Antenna's Surface Current Image Data Sets	117
5.4.2 Overall Inference for Stage -2 Results	121
5.5 References	121
CHAPTER 6 : CONCLUSION AND FUTURE WORK	122
6.1 Conclusion	123
6.2 Future Work	124
6.3 References	126
APPENDIX	127
Appendix -1 : 1- Dimensional FSV Implementation	128
Appendix -2 : Image Conversion Using Python	143
Appendix -3 : VBA Macro Function to Export Each Row from Source Sheet to Individual Sheets	145
Appendix -4 : VBA Macro Function to Export Each Column from Source Sheet to Individual Sheets	147
Appendix -5 : Repeated 1-D FSV (1-D FSV with a for Loop)	149
Appendix -6 : VBA Macro Function to Copy Each Row from Source Sheet to the Destination Sheet	158
Appendix -7 : VBA Macro Function to Copy Each Column from Source Sheet to the Destination Sheet	160
Appendix -8 : Program to Calculate 2-D FSV and Plot its Output in MATLAB	162
Appendix -9 : Program to Calculate PDF, CDF and Statistical Moments	166
Appendix -10 : Program to perform K-S Test	168

TABLE OF FIGURES

CHAPTER 1: INTRODUCTION

Figure 1: Simulated surface current distribution of On-skin passive UHF RFID tag02

CHAPTER 2: LITERATURE REVIEW

Figure 1: E-shaped and H-shaped microstrip patch antenna with spaces or gaps in the device structure..... **Error! Bookmark not defined.**

Figure 2: Demonstration of Real and Virtual images by a human eye08

Figure 3: Zoomed pixel representation of a digital image09

Figure 4: Pixel Dimensions.....09

Figure 5: Demonstrating the bit depth of a black and white image, a greyscale image, an RGB image and a CMYK image 10

Figure 6: Greyscale Colour Model 11

Figure 7: RGB colour model 11

Figure 8: CMYK colour model 12

Figure 9: Vector Image versus Raster Image..... 13

Figure 10: Demonstration of enhancements in the Eiffel Tower image 17

Figure 11: Demonstration of simple segmentation in the Eiffel Tower image by partitioning into three regions..... 18

Figure 12: Demonstrating features such as pattern and shape of the Eiffel Tower 19

Figure 13: An example of typical EMC data21

Figure 14: Design of two MIMO antenna placed vertically26

Figure 15: Simulation model of the dual-band MIMO antennas27

Figure 16: Top view of the aircraft with probes setup.....28

Figure 17: Brahmos missile's schematic showing observation angle29

Figure 18: Cross talk model29

Figure 19: Down sampling method30

CHAPTER 3: FSV IMPLEMENTATION

Figure 1: Steps to Implement 1- Dimensional FSV41

Figure 2: Input Data 1 and 2 for comparison.....42

Figure 3: Fourier transformed data 1 and 2	43
Figure 4: DC points for data 1 and 2 in the transformed domain	43
Figure 5: Low and High windows for data 1	45
Figure 6: Low and High windows for data 2	45
Figure 7: Low filter data 1 and 2	46
Figure 8: High filter data 1 and 2	46
Figure 9: Inverse transformed DC for data 1 and 2	46
Figure 10: Inverse transformed Low filter for data 1 and 2	46
Figure 11: Inverse transformed High filter for data 1 and 2	46
Figure 12: Point-by-point ADM	48
Figure 13: Histogram for ADM	49
Figure 14: Point-by-point FDM	50
Figure 15: Histogram for FDM	51
Figure 16: Point-by-point GDM	51
Figure 17: Histogram for GDM	52
Figure 18: Steps to Implement 2- Dimensional FSV	53
Figure 19: 2-Dimensional Input data	53
Figure 20: 2-D data separated vertical and horizontally	54
Figure 21: n-D FSV method on a 3D data	56

CHAPTER 4: PROPOSED METHOD

Figure 1: Illustration of the proposed methodology and research approaches	63
Figure 2: Illustration of steps involved in Approach 1	64
Figure 3: Illustration of steps involved in Approach 2	64
Figure 4: Image of Einstein	65
Figure 5: E-Field output of an EMC structure	66
Figure 6: RGB Channels of an E-field image	67
Figure 7: E-Field Image converted in to RGB array	67
Figure 8: Steps for Image conversion using Python	69
Figure 9: E-field image in RBG colour mode	70
Figure 10: E-field image converted into a greyscale image	71

Figure 11: A illustrative example of a Greyscale image converted into NumPy array and opened in Microsoft Excel.	73
Figure 12: Simulated surface current distribution of On-skin passive UHF RFID tag	75
Figure 13: Masks layered on gap or space.....	76
Figure 14: Segmented blocks of a RFID tag.....	77
Figure 15: Example of Pixel level segmentation	78
Figure 16: Example of 2-D full array copied to individual	79
Figure 17: Example of 2-D segmented array copied to	80
Figure 18: Steps to separate rows and columns from a 2-D array	81
Figure 19: Repeated 1-D FSV using For-Loop.....	85
Figure 20: Recombining row and column data into a single output file.....	86
Figure 21: Steps to recombine row and column data into a single output file.....	87

CHAPTER 5: EXPERIMENT RESULTS AND ANALYSIS

Figure 1: E-field image dataset used for Stage- 1 FSV processing	95
Figure 2: Einstein Image dataset used for Stage- 1 FSV processing	95
Figure 3: Microstrip patch antennas simulations operating at frequencies of 5.2 GHz, 5.3 GHz, and 5.6 GHz used for Stage-2 FSV processing	97
Figure 4: Different comparisons of E-field images used as input datasets for 2-D FSV processing	98
Figure 5: Full structures of E2800_field (0) vs E2800_field (2) images used in Approach 1	99
Figure 6: Confidence histograms for ADM, FDM and GDM	99
Figure 7: Linearized GDM, PDF and CDF Outputs	100
Figure 8: E2800_field (0) vs E2800_field (2) images segmented horizontally used in Approach 2.....	101
Figure 9: Confidence histograms for ADM, FDM and GDM	102
Figure 10: Linearized GDM, PDF and CDF Outputs	103
Figure 11: 2-D FSV Results of E2800_Field (2) vs E2800_Field (15) using Approach 1 ...	105
Figure 12: 2-D FSV Results of E2800_Field (2) vs E2800_Field (15) using Approach 2....	106
Figure 13: K-S Test results of E2800_Field (2) vs E2800_Field (15) comparing Approach 1 and Approach 2	106
Figure 14: 2-D FSV Results of E2800_Field (2) vs E2800_Field (25) using Approach 1 ...	107
Figure 15: 2-D FSV Results of E2800_Field (2) vs E2800_Field (25) using Approach 2....	108

Figure 16: K-S Test results of E2800_Field (2) vs E2800_Field (25) comparing Approach 1 and Approach 2 108

Figure 17: 2-D FSV Results of E2800_Field (2) vs E2800_Field (90) using Approach 1... 109

Figure 18: 2-D FSV Results of E2800_Field (2) vs E2800_Field (90) using Approach 2... 110

Figure 19: K-S Test results of E2800_Field (2) vs E2800_Field (90) comparing Approach 1 and Approach 2 110

Figure 20: Different comparisons of Einstein images used as input datasets for 2-D FSV processing 111

Figure 21: 2-D FSV Results of Einstein_Original vs Einstein_Blur using Approach 1 112

Figure 22: 2-D FSV Results of Einstein_Original vs Einstein_Blur using Approach 2 113

Figure 23: K-S Test results of Einstein_Original vs Einstein_Blur comparing Approach 1 and Approach 2 113

Figure 24: 2-D FSV Results of Einstein_Original vs Einstein_Impulse using Approach 1 ... 114

Figure 25: 2-D FSV Results of Einstein_Original vs Einstein_Impulse using Approach 2 ... 115

Figure 26: K-S Test results of Einstein_Original vs Einstein_Impluse comparing Approach 1 and Approach 2 115

Figure 27: Different runs of Microstrip patch antenna surface current input datasets for 2-D FSV processing 118

Figure 28: 2-D FSV Results of Patch_Antenna_5.2 GHz vs Patch_Antenna_5.3 GHz using Approach 2 119

Figure 29: 2-D FSV Results of Patch_Antenna_5.2 GHz vs Patch_Antenna_5.6 GHz using Approach 2 120

CHAPTER 6: CONCLUSION AND FUTURE WORK

Figure 1: Surface currents distribution on 5G mobile antenna 125

Figure 2: Surface currents plots on rectangular monopole antenna 125

TABLE OF TABLES

CHAPTER 2: INTRODUCTION

Table 1: Comparison of Image File formats	15
Table 2: FSV Interpretation xDM indicates ADM, FDM or GDM	25

CHAPTER 3: FSV IMPLEMENTATION

Table 1: Filter Values for Low windows	45
Table 2: Filter Values for High windows	45
Table 3: FSV Interpretation scale	49
Table 4: Piecewise linear FSV interpretation	57

CHAPTER 5: EXPERIMENT RESULTS AND ANALYSIS

Table 1: References of Input Image datasets used for Stage-1 FSV processing	95
Table 2: References of Input Image datasets used for Stage-2 FSV processing	96
Table 3: Comparison references and E-field images used as input datasets for 2-D FSV processing	98
Table 4: Total Value for ADM, FDM & GDM	100
Table 5: Statistical Moments	101
Table 6: Total Value for ADM, FDM & GDM	102
Table 7: Statistical Moments	103
Table 8: K-S Test Results for Approach 1 and Approach 2	104
Table 9: Comparison references of Einstein images used as input datasets for 2-D FSV processing	111
Table 10: Comparison references of surface current output images from a microstrip antenna used as input datasets for 2-D FSV	118

LIST OF ABBREVIATIONS

ADM	Amplitude Difference Measure
AHP	Analytic Hierarchy Process
AI	Adobe Illustrator
BMP	Bit Map Picture
CAD	Computer Aided Design
CDF	Continuous Distribution Function
CEM	Computational Electromagnetics
CMYK	Cyan, Magenta, Yellow, Key
CPW	Coplanar Waveguide
CST	Computer Simulation Technology
CSV	Comma-Separated Values
DIP	Digital Image Processing
DRSG - SCM	Dimension-Reduced Sparse Grids Scheme
DSP	Digital Signal Processing
ECDF	Empirical Cumulative Distribution Function
E-Field	Electric Field
EM	Electromagnetic
EMC	Electromagnetic Compatibility
EPS	Encapsulated PostScript
FDM	Feature Difference Measure
FEM	Finite Element method
FFT	Fast Fourier Transform

FIT	Finite Integration Technique
FSV	Feature Selective Validation
GDM	Global Difference Measure
GIF	Graphics Interchange Format
HIRF	High Intensity Radiated Field
IDE	Integrated Development Environment
IFFT	Inverse Fast Fourier Transform
IQA	Image Quality Assessment
JPEG	Joint Photographic Experts' Group
K-S	Kolmogorov-Smirnov
LEED	Low-Energy Electron Diffraction
MATLAB	Matrix Laboratory
MoM	Moment Method
NumPy	Numerical Python
PDF	Probability Density Function
PIL	Python Image Library
PIP	Preferred Installer Program
PNG	Portable Network Graphics
PO	Physical Optics
RCS	Radar Cross Section
R-Factor	Reliability Factor
RFID	Radio Frequency Identification
RGB	Red, Blue, Green
SAR	Synthetic Aperture Radar

SCM	Stochastic Collocation Method
S – Parameter	Scattering Parameter
SVG	Scalable Vector Graphic
TIFF	Tag Image File Format
UHF	Ultra High Frequency
UWB	Ultra-Wide band
1-D	One-Dimensional
2-D	Two-Dimensional
3-D	Three-Dimensional

CHAPTER - 1

INTRODUCTION

Validation in general is the process to ensure the expected behaviour of the modelling technique in the field of Computational electromagnetics (CEM). The Feature Selective Validation (FSV) technique was originally developed to aid the validation in Computational Electromagnetics (CEM). It majorly focuses on applications in Electromagnetic compatibility (EMC). From then, the application of FSV became the major element in EMC and it was considered as the recommended validation technique for CEM in IEEE Standard 1597.1 for Validation of Computational Electromagnetics Computer Modelling and Simulations [1] [2].

The FSV method was developed to aid visual evaluation technique in EMC. The FSV technique decomposes the original EMC data sets into two major components namely, Amplitude Difference Measure (ADM) and Feature Difference Measure (FDM). The ADM measures the slow-moving trend of the EMC data sets. On the other hand, the FDM measures the consistency of rapidly varying features of the EMC data sets. The Global Difference Measure (GDM) is determined by combining the ADM and FDM which measures the overall goodness of fit between the data sets. Therefore, The FSV technique is an experimental process which quantifies the difference between two set of EMC data into several components using weighted sum approach and then combining the data into several metrics. In FSV method, detailed analysis of the data is reviewed by point-by point basis on their individual feature or trend [2] [3].

Since the original development of the FSV method, more complex data in EMC such as surface current outputs from Ultra High Frequency (UHF) devices with space or gap within the device structure is available to be compared. Performing 2-Dimensional FSV on Electric and Magnetic field data represented in 2-Dimensional image format are challenging because the gaps could dominate the overall comparison if those ‘voids’ were included in the comparison itself. Similarly, some of these structures contain small, fine details that can be studied alone as well as part of the overall structure. Figure 1 represents one such example of an Simulated surface current distribution of On-skin passive UHF RFID tag [4].

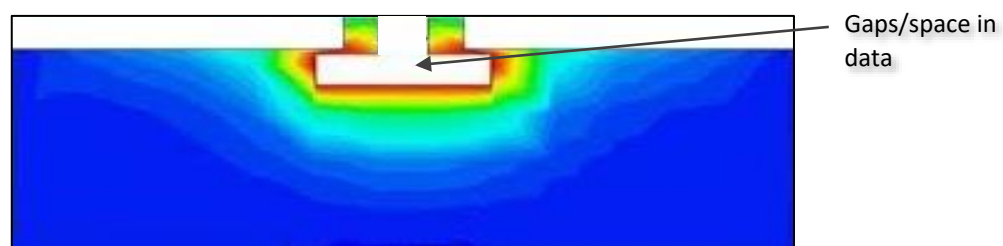


Figure 1: Simulated surface current distribution of On-skin passive UHF RFID tag [4]

For the reason that a gap or space in an image does not represent any data points of a device structure, a masking approach is chosen to overlay the gap with a binary mask image (containing zero and non-zero values) to perform FSV for the full device structure [6] [7]. Although the masking approach appears to be simple and straightforward, it fails from certain drawbacks which are discussed in Chapter 4, Section 4.3.

To overcome the drawback of the masking approach, a simple manual block-based segmentation method was developed to compare device structures with gaps or spaces [8]. In this process, an input image is segmented into non-overlapping blocks of pixels, and FSV for each segmented block is performed to compare the feature and trend of segmented blocks and subsequent recombination.

To ensure that the segmentation and subsequent recombination do not produce comparison data that has been affected by the process; two approaches are developed to perform 2-D FSV for such input images. The first approach performs 2-D FSV on the original full structure of the input images. While the second approach is accomplished by segmenting the 2-D image into multiple regular 2-D blocks and then performing 2-D FSV on each block individually. The original 2-D image is then created by concatenating the 2-D FSV outputs from the segmented blocks. Finally, approach 2 (segmented approach) FSV outputs are compared with approach 1 (full structure) FSV outputs, and the findings are analysed.

This 2-Dimensional FSV approach developed to process image with gaps or spaces within the image structure are discussed and explained in detailed in this thesis. Equally, the development, enhancement, criticism and challenges of the FSV method are also analysed in this thesis [5].

1.1 OVERVIEW OF THE THESIS

In this thesis, images in EMC are used as source input data to perform 2-Dimensional FSV on full and segmented image structures. As a result, the thesis begins with a thorough discussion of the fundamentals of an image, including image categories, image pixels, pixel attributes, file formats, and image colour modes. In addition, the evolution of FSV is covered in depth in the literature review.

Secondly, the detailed steps to implement 1-Dimensional, 2-Dimensional, and n-Dimensional FSV using IEEE STD 1597.1 and IEEE STD 1597.2 are described with an example [1] [9]. Moreover, to verify FSV performance; Probability Density Functions (PDF) and Statistical moments were calculated. These steps are implemented in MATLAB. Following that, two approaches for performing 2-D FSV on the original structure of the source image as well as

segmented blocks of the source image were developed. The acquired source images are transformed into a 2-D array, which is then segmented into regular blocks. Now, 2-D FSV performed on each block separately and then concatenated to the original image structure. Finally, the developed 2-D FSV approaches are used to process various real-time and computer-generated images, and the results were analysed.

1.2 AIM

To develop and analyse an approach to perform 2-Dimensional Feature Selective Validation on data that will ultimately allow application to geometrically irregular data with spaces or gaps within the image structure.

1.3 OBJECTIVES

The major objectives of the research study are as below,

- Develop a segmentation approach to perform 2-D FSV and demonstrate the general effectiveness of the approach.
- To analyse the approach using a Kolmogorov-Smirnov (K-S) Test to ensure that the segmentation and subsequent recombination does not result in comparison data that is unduly affected by the process itself.
- Propose an approach whereby the method can be applied to structures of irregular shape.

1.4 ORGANIZATION OF THE RESEARCH WORK

The thesis is structured into six chapters as follows,

- **Chapter 1:** Aim, objectives and a brief introduction of the research study is given here.
- **Chapter 2:** Detailed literature review on an image and its fundamentals. The evolution of FSV is also discussed in detail in this chapter.
- **Chapter 3:** Steps developed to implement 1-D, 2-D and n-D FSV including Density function, Statistical moments and KS test are given.
- **Chapter 4:** Develop proposed methodology to perform 2-D FSV approaches on the original structure and segmented blocks of the source images.
- **Chapter 5:** The proposed methodology is applied to input image datasets and the result are analysed. The FSV outputs of the original and segmentation approaches are compared and findings are discussed in this chapter.

- **Chapter 6:** The findings for the research study are concluded and the future work are discussed.

Therefore, the thesis focuses on aim and objectives to develop a methodology to perform 2-Dimensional Feature Selective Validation (FSV) on irregular dataset with spaces or gaps within the image structure. To accomplish this, it is important to understand the fundamentals of an image, characteristics of an image, image pixels, colour modes and formats. On the other hand, it is also important to study the evolution of Feature Selective Validation (FSV) and its applications in the field of Computational Electromagnetics (CEM). These are discussed in detail in the following Chapter 2.

1.5 REFERENCES

- [1] A. Duffy, A. Martin, A. Orlandi, G. Antonini, T. Benson and M. Woolfson, "Feature Selective Validation (FSV) for Validation of Computational Electromagnetics (CEM). Part I—The FSV Method", *IEEE Transactions on Electromagnetic Compatibility*, vol. 48, no. 3, pp. 449-459, 2006.
- [2] A. Martin, "QUANTITATIVE DATA VALIDATION (AUTOMATED VISUAL EVALUATIONS)", PhD Thesis, De Montfort University, 1999.
- [3] V. Rajamani, C. Bunting, A. Orlandi and A. Duffy, "Introduction to feature selective validation (FSV)", 2006 IEEE Antennas and Propagation Society International Symposium, 2006.
- [4] M. Rutschlin, "5G Antenna Design for Mobile Phones | The SIMULIA Blog", *The SIMULIA Blog*, 2020.
- [5] "IEEE Standard for Validation of Computational Electromagnetics Computer Modeling and Simulations".
- [6] U. Qidwai and C. Chen, *Digital image processing*. Boca Raton, Fla.: CRC/Chapman & Hall, 2010.
- [7] J. Enns and V. Di Lollo, "What's new in visual masking?", *Trends in Cognitive Sciences*, vol. 4, no. 9, pp. 345-352, 2000.
- [8] S. B and R. B, "COMPARATIVE STUDY OF BLOCK-BASED IMAGE SEGMENTATION TECHNIQUE", *International Research Journal of Mathematics, Engineering and IT*, vol. 3, no. 12, 2016.
- [9] "IEEE Recommended Practice for Validation of Computational Electromagnetics Computer Modeling and Simulations", 2011.

CHAPTER - 2

LITERATURE REVIEW

The modern world of science and technology are strongly reliant on electromagnetic (EM) systems such as radio, television, radar, microwaves, telephones, satellite communication systems, internet and computers. Computational electromagnetics (CEM) tools are useful in designing and analysing EM systems. CEM also aids in validating EM problems using computational methods [1]. One such method is known as Feature selective validation (FSV), which is a data comparison technique for Validation of Computational Electromagnetics Computer Modelling and Simulations in IEEE Standard 1597.1 [2].

Electromagnetic devices such as antennas and filters can be modelled using computers for analysis and optimisation. When such devices are simulated, output parameters including Scattering parameters (S- parameter), Electric field (E- field), Magnetic field (H- field) and surface currents are generated. These outputs are then exported as 2- or higher-dimensional image formats for further processing and comparison using the FSV technique. However, performing FSV on the microstrip antennas shown in Figure 1 is challenging due to the spaces and gaps within the device structure. Therefore, this research study focuses on developing a methodology to perform 2-D FSV on irregular data with spaces or gaps within the device structure.

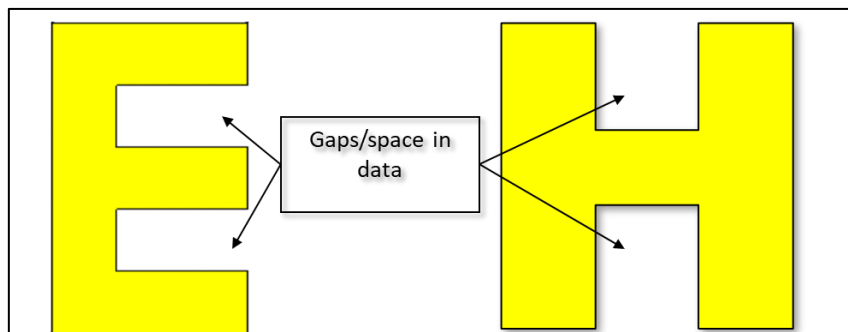


Figure 1: E-shaped and H-shaped microstrip patch antenna with spaces or gaps in the device structure [3][4]

Hence, it is important to understand an image, types of images, properties of an image and how these images could be further processed using FSV method. Therefore, this chapter explains the fundamentals of an image and processing FSV. On the other hand, understating a detailed literature on evaluation of FSV is studied in this chapter.

2.1 FUNDAMENTALS OF AN IMAGE

In the modern world, images are omnipresent. The imaging system has evolved into a vital component of human life. It is used for various applications such as communication, entertainment, military, marine, science and technology. An image is formed when a light ray

from an object intersects or appear to intersect after a reflection or refraction. Such images can be viewed by human eyes or can be captured using imaging systems such as cameras, microscopes, telescopes and lenses [5].

In other words, image is a visual representation of an object that emits light rays either directly or indirectly. An image is broadly classified as real and virtual images depending upon the reflection or refraction from the object. The Image formed when light rays from an object intersect with each other after reflection or refraction is known as real image. While the light rays from an object does not intersect after reflection or refraction is knows as virtual image [6]. This is shown in figure 2 below. However, as the light rays diverge in both real and virtual images, human eye is unable to differentiate it. For example, Images that appear a on computer screens or a movie screen are real and those images on a plane mirror are virtual.

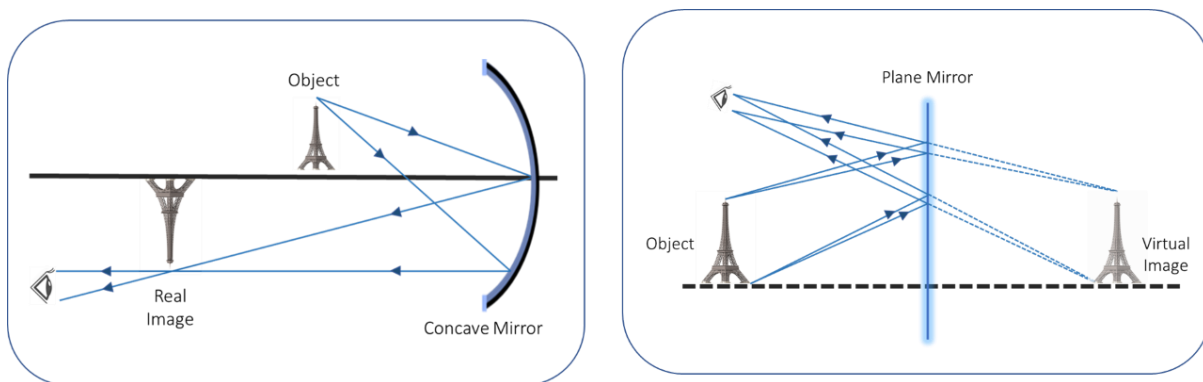


Figure 2: Demonstration of Real and Virtual images by a human eye

When an image is captured using digital equipment such as video recorders, photographic devices and scanners; images are digitized to form a digital image. Digitization of an image is a process of mapping continuous signal from an optical image to a discrete number of spatially organised points, (commonly known as pixels) with the amount of information stored in each pixel (known as pixel depth). These data are stored in the internal memory space of the image capturing devices which is known as digital image. The data stored in the memory can then be read into the computer across the communications interface and can be transferred to the computer memory. As computers excel in storing and manipulating numbers, it is used to examine and view the digital image [7] [10].

To understand the Digital imaging in detail, it is important to know about the four basic features of an image, which are the pixel, dimension, bit depth and colour models [8]. A brief on each feature is discussed below.

2.1.1 PIXEL

Comparison of Electromagnetic data with irregular or discontinuous surfaces using the Feature Selective Validation method

A pixel (an abbreviated form of the phrase ‘Picture Element’ [7]), represents the smallest unit in a digital image. A digital image is given by a two-dimensional grid of dots, which has unique position and colour. Each of these dots are known as pixel [8][9]. Figure 3 represents a zoomed pixel representation of a digital image.

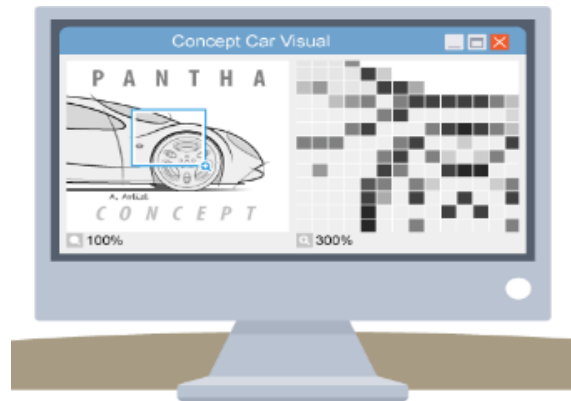


Figure 3: Zoomed pixel representation of a digital image [10]

2.1.2 DIMENSION

The number of pixels along the row and column of an image is known as pixel dimension [8].

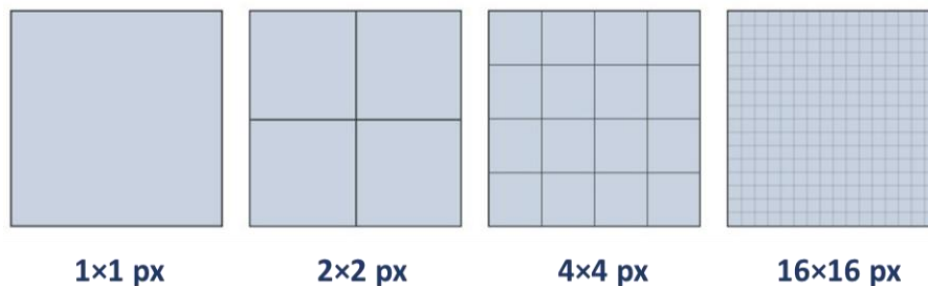


Figure 4: Pixel Dimensions

Images from left to right in Figure 4 are, 1×1 pixel: 1 dot high and 1 dot wide (1 pixel), 2×2 pixels: 2 dots high and 2 dots wide (4 pixels), 4×4 pixels: 4 dots high and 4 dots wide (16 pixels), and 16×16 pixels: 16 dots high and 16 dots wide (256 pixels).

2.1.3 BIT DEPTH

An image's bit depth determines how many colours it may include. The total number of colours that can be present in an image is specified by one or more numbers and the range within which those numbers can fall. The more colours an image can store, the higher its bit depth [9].

For example, each pixel in a basic 1-bit image may generally represent only two colours: black and white. This is because a 1-bit pixel can only hold one of two values: 0 (white) or 1 (black). In a 2-bit image, there are four possible pixel values (00, 01, 10, and 11). As a result, there are

four different colours or grey levels. On the other hand, an 8-bit image, can store 256 possible values for Red, Green and Blue. Similarly, a 24-bit image, may display nearly 16 million four-channel colours - Cyan, Magenta, Yellow, and Key-Black (CMYK) [11].

Figure 5 below illustrates the bit depths of various colour models. The file size of the image increases as the bit depth increases when more colour information is recorded for each pixel in the image.

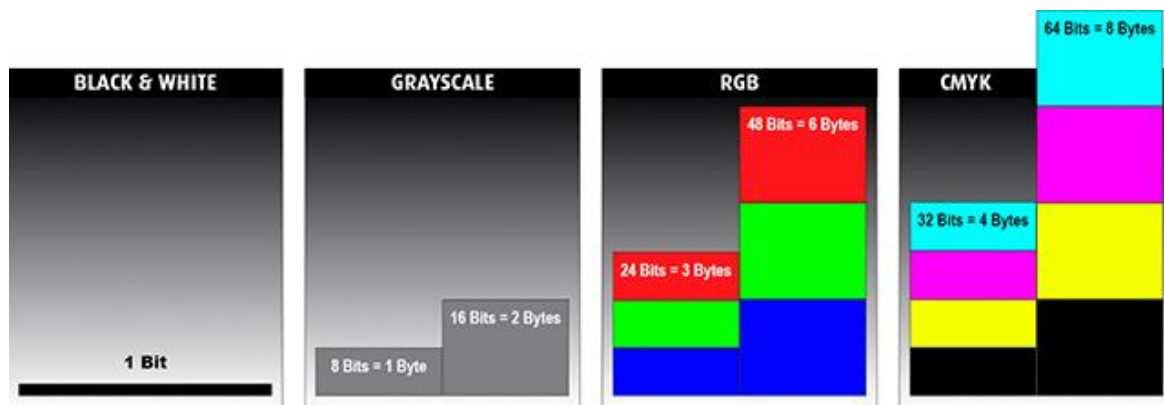


Figure 5: Demonstrating the bit depth of a black and white image, a greyscale image, an RGB image and a CMYK image [11]

2.1.4 COLOUR MODEL

Another essential property of an image which governs how colours may be quantitatively represented in an image is called the colour model. It is a mathematical model that specifies how the colour components are represented as a group of numbers. The most popular colour models used in image representation are Black and White, Greyscale, RGB and CMYK [8]. This plays a vital role in establishing communication between human and computers to understand the information of an image [12]. The various colour models are briefly explained below.

2.1.4.1 BLACK AND WHITE COLOUR MODEL

Black and white colour model, also known as binary images; hold either '0' or '1' representing white or black respectively. These images may have only 1 bit [13] [14].

2.1.4.2 GREYSCALE COLOUR MODEL

Here, the image contains the shades from black and white colour model. To form an even ground grey pixel, intensity of black and white are mixed [15]. The Greyscale images provide 8 bits of information, in which each pixel represents 256 colours of pure black, absolute white colours, and 254 shades of grey [14]. Figure 6 below represents a Greyscale colour model.

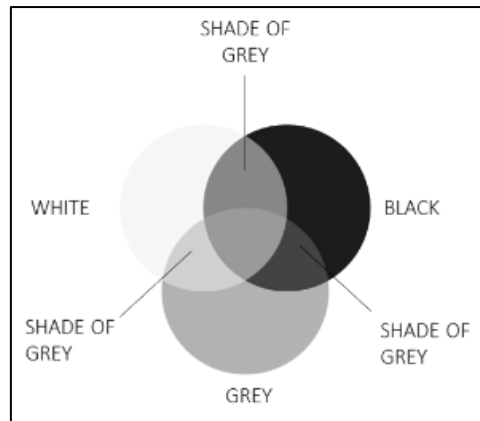


Figure 6: Greyscale Colour Model

2.1.4.4 RGB COLOUR MODEL

Red, Green and Blue (RGB) is an additive colour model used for displaying digital images on any sort of light transmitting medium such as computer displays. A white light source projected in a digital display performs additive mixing to generate a wide variety of additional apparent colours on the retina of the eye by combining red, green, and blue light and varying their intensities [12].

An RGB model's Red, Blue, and Green levels are represented by a decimal number ranging from 0 to 255, where 0 represent no representation of the colour and 255 represents the greatest possible. Moreover, this colour model can be represented in a 3-dimensional matrix, where each dimension holds red, green and blue respectively [16]. Figure 7 below represents an RGB colour model.

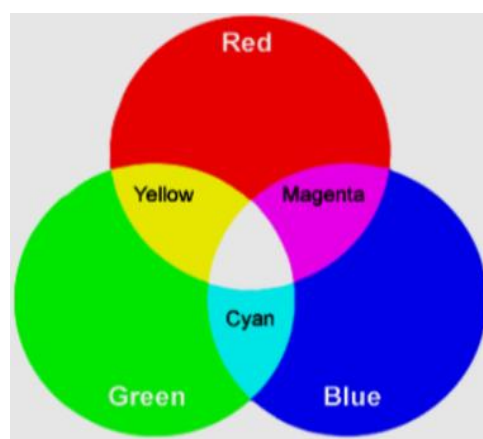


Figure 7: RGB colour model [16]

2.1.4.4 CMYK COLOUR MODEL

CMYK (Cyan, Magenta, Yellow, Key/Black) is a subtractive colour model. Adding colours in CMYK mode has the reverse results as adding colours in RGB mode. All colours begin as white, the more colour added, the darker the results. Therefore, colours are subtracted to create a colour in lighter intensities. The colour models are commonly used in printing products which will blend CMYK colours to print digital images [17]. Figure 8 below illustrates a CMYK colour model.

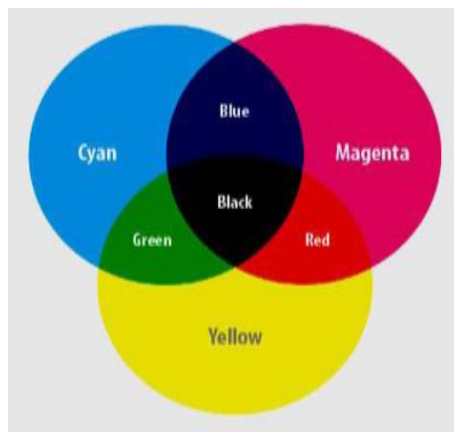


Figure 8: CMYK colour model [16]

In this thesis, the input test data for image comparison experiments uses monochrome images exhibiting a single grey colour or 256 shades of a grey colour. Instead of collecting descriptors and working on colour images directly, greyscale representations were chosen to simplify and minimise computational techniques. Moreover, this helps to prove the concept and approach for the work presented in this thesis.

2.1.5 RASTER AND VECTOR IMAGES

Digital images are the fundamental components in the word of Information Technology. These digital images are categorised into two types: Raster images and Vector images. When an image is compiled using regularly sampled values known as pixel, it is called a raster image. While, vector images are those which consists of lines and curves generated mathematically in a computer [13][18]. Understanding the difference between these image forms is essential for creating and working with digital images. Each image forms have its unique properties and applications which are further explained in the below sections. The raster and vector images created, replicated or manipulated using software are saves in electronic file formats, which are further explained in this chapter section 2.6.1. Figure 9 below represents an example of Vector and Raster image.

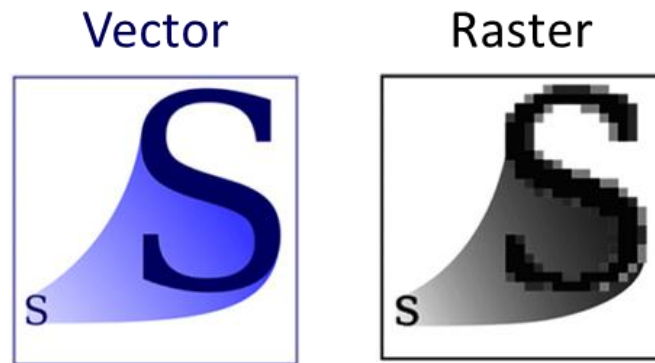


Figure 9: Vector Image versus Raster Image [19]

2.1.5.1 VECTOR IMAGES

Vector images are computer-generated images that adhere to a mathematical formula. These images have various attributes like thickness of a line, length and colour. For example, these images are used for creating fonts, logos, 2-D or 3-D computer animations. These are stored in a file formats such as Encapsulated PostScript (EPS), Scalable Vector Graphic (SVG), Adobe Illustrator (AI) [8].

2.1.5.2 RASTER IMAGES

Raster images, also known as bitmap, is commonly encountered form of representation. These are used to represent and manage real-world phenomena as they are resolution dependent, made up of fixed number of pixels. Here, each pixel is defined with its own spatial location and colour [8]. For example, these images are used for photography and print materials and are stored in a file formats such as BMP, GIF, JPG, PNG and TIFF [13].

2.1.6 FILE FORMATS

In the scientific field, an image can be made up of numbers in a 2-dimensional array. However, when it comes to real world applications, it is important to store images compactly to display, transmit and receive it between networks [15].

An established specification for encoding information about an image into bits of data for storage is known as a file format of an image. This plays a vital role, as the image saved to a well-known format recognises itself as an image and provides relevant information like bit depth and size, which in turn allows interaction with the file. Moreover, any program which can follow the file format could open the file and display the image. Therefore, the aim of any file format of an image is to store data efficiently [8].

To use the storage capacity efficiently, it is important to make sure that the data is compact, for this purpose image file formats use fundamental data compression techniques named lossy and lossless compression. Both techniques use the redundancy of the image to compress it. The Lossless compression is when the mathematical redundancy is reduced, while Lossy compression is when the perceptual redundancy is reduced [8].

Some common image formats are Joint Photographic Experts' Group (JPEG), Graphics Interchange Format (GIF), Portable Network Graphics (PNG), Bit Map Picture (BMP) and Tag Image File Format (TIFF). These formats of an image are mostly used for printing and scanning [15] [7].

2.1.6.1 JPEG

The Joint Photographic Experts Group (JPEG) is the international image compression standard developed for multilevel images in greyscale and colour [16]. Naturally, JPEG is able to compress large files. It can store up to 36 bits for applications in medical and scientific fields. For RGB colour images, JPEG is capable of storing up to 24-bits. It is used in other applications such as web browsing, printing, scanning and in digital cameras [15] [20]. These files are often stored with '*.jpg*' extension.

2.1.6.2 BMP

Bitmap (BMP) was originally developed by the Microsoft Windows operating system as a non-compressed file format which is device driven to convert and display images. This also uses 24-bit to display graphic images in colour and grayscale [15] [21]. Bit -mapped images can be used in digital document files and images in photographic [22].

2.1.6.3 GIF

Graphic Interchange Format (GIF) most widely used for online graphics that supports lossless compression. .GIF represents both animated and static images with 256 colours or shades of grey. The advantage of a GIF format is that it occupies relatively smaller file size which makes it easy to store and transmit the data rapidly across the network. GIF files are often saved with the '*.gif*' extension [21] [22].

2.1.6.4 PNG

Portable Network Graphics (PNG) is a computer file format which is widely used to store, transmit and display images. Moreover, this format supports lossless compression and can store large files which in turn provides data transparency. The advantage of a PNG is that it supports range of colour depths and simplifies the use of 16, 24 and 32-bit images. PNG file format is capable to store up to 48 bits per pixel of RGB colour images and up to 16 bits per pixel of grey scale images. PNG files are commonly saved with the ‘.png’ extension [21] [23] [24].

2.1.6.5 TIFF

Tagged Image File Format (TIFF) was originally developed to support devices in digital image processing such as printers, monitors and scanners. It supports lossless compression and capable to store large files without any loss in the detail. Over the time, TIFF evolved in to colour images from grey scale images. A TIFF has ‘.tiff’ (or) ‘.tif’ file extensions [21] [22].

The table 1 below represents overall comparison of various file format discussed in this section.

File Formats	Description	Advantages	Disadvantages	File Extensions	Application
Joint Photographic Experts Group (JPEG)	A commonly used lossy compression format.	Widely supported compression format for photos.	Lossy compression could result in image degradation.	.jpg & .jpeg	Web graphics and digital photography.
Bitmap (BMP)	A simple and widely supported uncompressed format.	Widely supported uncompressed format suitable for simple graphics.	It has large file size and limited colour depth.	.bmp	Window based applications and simple graphics.
Graphic Interchange Format (GIF)	A format that supports transparency and animations.	Suitable for simple graphics.	Limited colour depth and not suitable for photographs.	.gif	Web graphics and simple animations.
Portable Network Graphics (PNG)	A format that supports transparency and lossless compression.	Good for simple graphics and photos.	It has larger file size when compared to JPEG. It does not support older software.	.png	Web graphics, digital photography and graphic design.
Tagged Image File Format (TIFF)	A high quality format that supports multiple layers.	High quality and suitable for professional printing and archiving.	Large file size and not widely supported. It is not suitable for web.	.tif & .tiff	Graphic design, professional photography and printing .

Table 1: Comparison of Image File formats.

Following the study of the fundamentals of images, the next section discusses Image Processing and its techniques. In this research study, Image Processing involves evaluating the trend and features of an image obtained from real-time scenarios and EMC simulations.

2.2 IMAGE PROCESSING

An image is processed to improve graphic and human interpretation, for data storage, transmission and represent the outcomes from an independent machine perception [25]. Digital images are made up of 2- dimensional array of pixels which represents the physical quantity such as dimensions, colour and bit depth of an image, which when processed using a computer or visual recognition provides useful outcomes that can be to interpreted by human / computer.

Digital Image Processing (DIP) is an interdisciplinary system that combines principles from various fields such as engineering, mathematics, medical, satellite communications, optical devices and in military applications [26] [27]. Digital Images can be processed in many ways as per the requirement, some of the image processing techniques are restoration, enhancement, removing noise/blur or any feature extraction of an image, segmentation, compression, and transformation [28]. With the number of processing methods, most common image processing techniques used in various applications are Image enhancement, Image segmentation and Image recognition [29].

2.2.1 ENHANCEMENT

An image is enhanced to provide a better visual representation. Images can be enhanced by improving the visual quality of an image through manipulating its attributes such as brightness, sharpness, colour balance, contrast and noise removal. The technique used for image enhancement varies with respect to its applications and outcomes.

This can be broadly classified as spatial domain and frequency domain techniques. In spatial domain, pixels of an image are directly manipulated. This technique is simple to understand and implement. Some of the commonly used spatial domain techniques are histogram equalisation, sharpening and contrast stretching which are used in in real time applications such as robotics, surveillance systems and video processing. While the frequency domain technique is based on filtering the high frequency and low frequency components of an image using filtering methods such as Fourier, Wavelet, and Cosine transforms to enhance an image [28] [30]. This technique is used in medical imaging and remote sensing applications that require high quality image enhancements [28].

In this thesis, it is important to understand image segmentation, as it plays a vital role data acquisition for the research study. This is further discussed and applied in Chapter 4, Section 4.1. The Figure 10 below, shows enhanced images of an Eiffel tower. The original image,

blurred image, 40% contrast, and 40% brightness of the Eiffel Tower are shown from left to right.

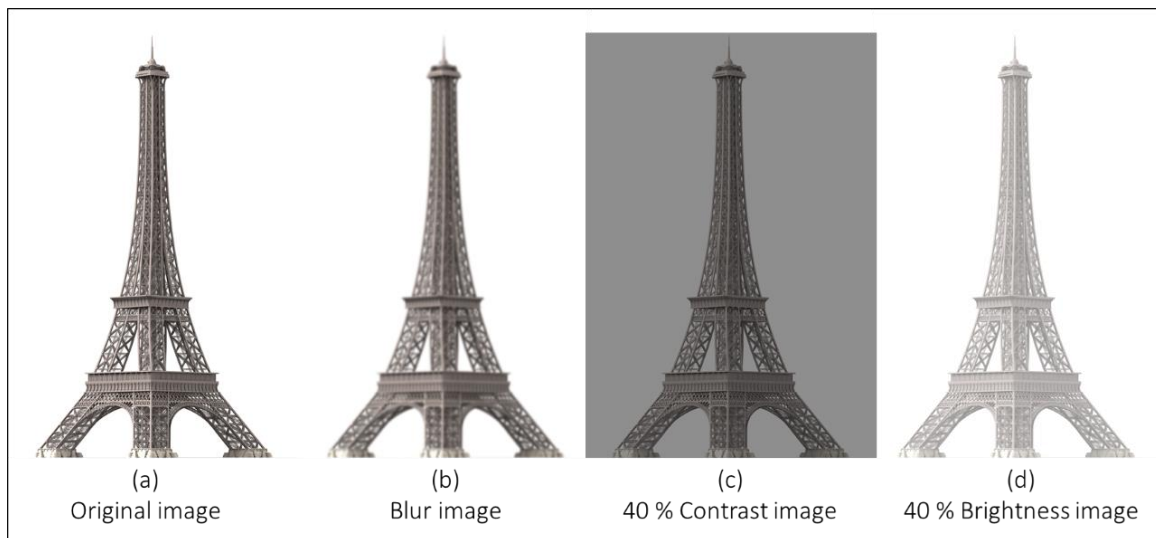


Figure 10: Demonstration of enhancements in the Eiffel Tower image.

2.2.2 SEGMENTATION

Image segmentation is the process of partitioning an image into multiple regions or segments [31]. Here, images are subdivided into a number of uniformly regular sections that are connected but do not intersect. Each segment contains a group of pixels that represent a distinct region of the image. Further, each segmented region could be characterised by texture, colour, and shape for the purpose of analysis [29] [31]. Segmentation is performed with the goal of identifying and extracting meaningful information from an image.

Therefore, segmenting an image into multiple sections aids in image analysis and processing for further applications such as medical imaging, object recognition, robotics, and autonomous vehicles. In this thesis, it is important to understand image segmentation, as it plays a vital role in segmenting input data for the research study. This is further discussed and applied in Chapter 4, Section 4.3. The below Figure 11 represents an example of image segmentation for the Eiffel Tower.

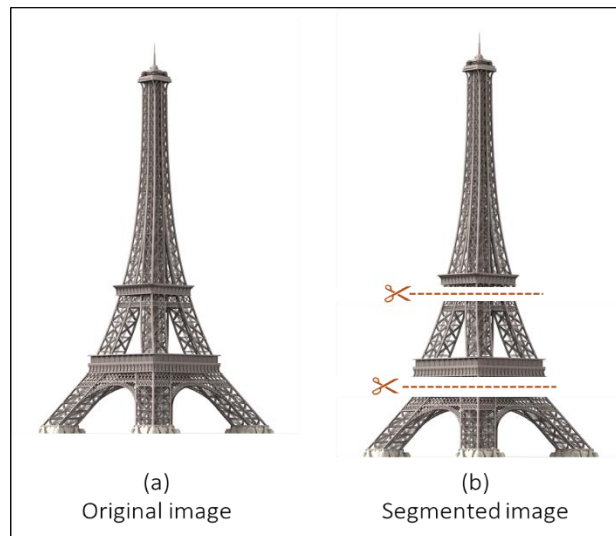


Figure 11: Demonstration of simple segmentation in the Eiffel Tower image by partitioning into three regions.

2.2.3 ANALYSIS

Image analysis is a technique that is used for the interpretation of visual information within an image, such as patterns, texture, shape, colour, and other features. The term “Patterns” refer to the regularity of the arrangement of objects or elements in an image. This can include features like lines, shapes, and repeating objects. The term "texture" refers to the surface characteristics of an object or area in an image, which can be described as smooth, rough, bumpy, or any other attribute. The term "shape" refers to the geometric form of an object in an image, which can be described by its boundaries or edges and could include elements like curves, angles, and corners. The term "colour" refers to the visual properties of an object in an image, which include hue, saturation, and brightness. Moreover, the other features of an image could include size, orientation, intensity, and depth.

These features can be extracted using various image analysis methods including segmentation, feature extraction and classification. The choice of these methods is applied depending upon its applications such as remote sensing, medicine, computer vision and more [32] [33]. In this thesis, it is important to understand image analysis, as it plays a vital role in developing the proposed methodology for the research study. This is further discussed and applied in Chapter 4, Section 4.2. The below Figure 12 represents an example showing features such as pattern and shape of the Eiffel Tower.

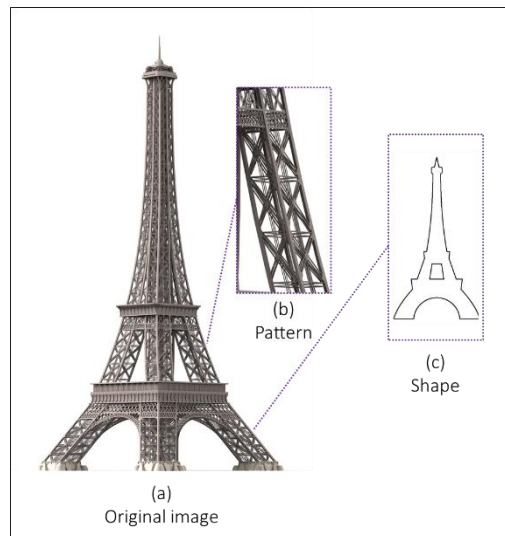


Figure 12: Demonstrating features such as pattern and shape of the Eiffel Tower

The advancement of technology has led to a significant dependency on electronic systems such as computers, mobile phones, and smart devices. These electronic systems play a vital role in image processing as they are more adaptable and affordable [27] [29]. This has created an opportunity for computer vision to be applied in various applications of image enhancement, segmentation, and analysis [34] [36].

The most significant aim of computer vision is to achieve human-like visual recognition techniques. Despite all the processing methods available, human perception plays a vital role in analysing an image because vision is the most advanced of our senses [35] [37]. Moreover, human brains are the most powerful tool that can recognise differences, features, colour, pattern, and imperfections in an image .

In the field of computational electromagnetics (CEM), image processing techniques are often used to validate the accuracy of computer-generated results. One such technique that mimics the visual interpretation is known as the Feature Selective Validation (FSV) method. The FSV method is a data comparison tool that decomposes the original pair of data sets into two components: low-pass components and high-pass components. The low-pass component contains the "trend" information in the data, which represents the overall pattern or shape of the data over time. The high-pass component contains the "feature" information in the data, which represents the specific oscillations or frequencies that are superimposed on the trend.

These low-pass and high-pass components are then compared using a combination of differences and derivatives of the filtered data. This comparison allows the approach to amplify the effects of resonant-like features in the data, which are frequencies that are particularly

strong or prominent when compared to the other frequencies present in the data. This method is important to aid the objective of the research, and thus it is important to understand the evolution and implementation of the FSV method. In the following section, the evolution of FSV is explained, and the procedure to implement the FSV method is given in Chapter 3.

2.3 INTRODUCTION TO FEATURE SELECTIVE VALIDATION

In the field of Computational Electromagnetics (CEM), validation is an essential process that is used to ensure the accuracy and reliability of modelling techniques. The models, such as mathematical and computer simulations, are developed in CEM to analyse and predict the behaviour of electromagnetic phenomena. These models are then used to develop and optimise a wide range of electromagnetic devices and systems, including antennas, filters, and communication systems. The validation procedure involves comparing the results from the simulation model to the results from real-world data sets in order to validate their accuracy and identify any differences or imperfections between the two. This is accomplished by comparing the expected outcomes from the models to the actual measurements from the real-world data [38].

Moreover, validation is required for combining similar data sets in EMC applications such as Digital Signal Processing (DSP), fingerprints, and retinal scanning. For example, in DSP, validation is used to ensure that the algorithm employed for signal processing appropriately replicates the real-world signal. Similarly, validation is used in biometric applications such as fingerprint scanning and retinal scanning to confirm the accuracy of the biometric data and to prevent errors or false positives [38] [39]. Therefore, validation plays a vital role in the field of CEM because it provides confirmation of the accuracy and reliability of the simulation models.

The initial method used to validate EMC (Electromagnetic Compatibility) data sets was the visual evaluation method. Engineers would compare the data sets and identify the differences between them. However, this approach had its own disadvantages due to subjectivity and the potential for human error. This led to the development of other data comparison models, such as Correlation and the Reliability factor. These techniques involved decomposing the original data sets, which further led to the development of the Feature Selective Validation (FSV) method.

Therefore, it is important to understand the visual evaluation method and other data comparison models to appreciate the need for FSV and how it is implemented. An example of typical EMC

data sets is shown in the figure 13 below. These data sets were obtained from a mode stirred reverberation chamber with different stirrer positions [39]. In general, these forms of EMC results, would be examined for validation techniques.

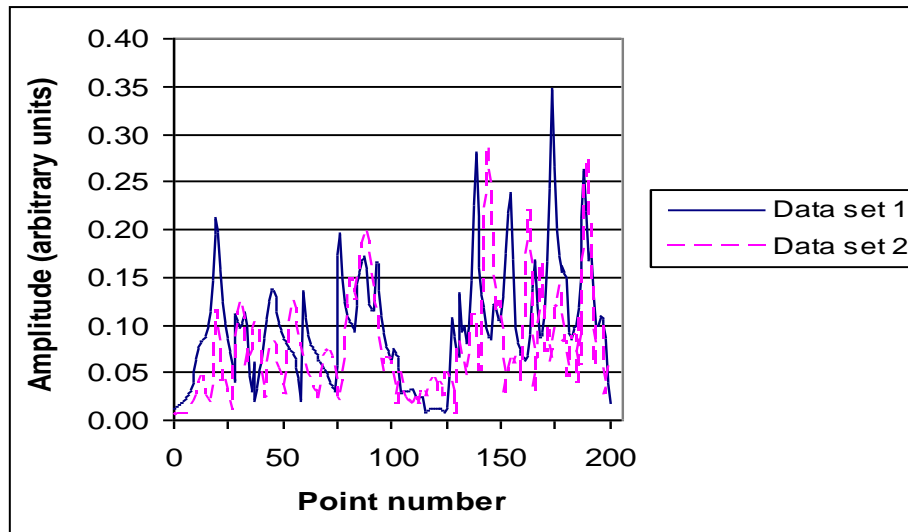


Figure 13: An example of typical EMC data [39]

The visual evaluation method was an early approach used to validate data sets in the field of Computational Electromagnetics (CEM). This method was based on the belief that the human brain is the most powerful device for recognising similarities and differences between data sets. This method can be determined in two stages: the construction of the imaginary model and the analysis stage. Here, the human brain acts as a stimulus to observe the data throughout its path.

In the first stage of the visual evaluation method, an imaginary model is built to begin the comparison of the data sets. This is done to establish a baseline against which the data sets can be compared. In the second stage, the data sets are validated by extracting them into three categories: atomic, relational, and positional. In the atomic extraction, complete data intensity is observed. In relational extraction, derivatives of the data are determined. Finally, in the positional extraction, the coordinate locations of the atomic and relational extractions are found [39].

However, this visual evaluation approach was also noted to have some drawbacks, when several complex data need to be validated, the level of accuracy and focus tend to decrease due to a multitude of reasons including fatigue, stress, etc. Moreover, individual perceptions vary from one person to another through differences including experience and education which will contribute to a spread of opinions [38] [39].

Therefore, this gave rise to an analytically based validation method, which was developed to mirror the visual evaluation technique [35]. The algorithmic approach for data comparison techniques gave rise to two major techniques namely, the correlation and reliability factor for the validation in CEM [38]. These techniques are applied on a typical EMC data as shown in figure 10 below.

2.3.1 CORRELATION

The correlation technique is the most commonly used method for comparing data sets by measuring their similarity. This technique requires minimal computing to provide the best fit between data points [38] [39]. The correlation for discrete data points is given in equation 1 below,

$$R(\tau) = \sum_{fmin}^{fmax} Iset1(x)Iset2(x + \tau) \quad (1)$$

Where,

$x \rightarrow$ Independent axis ranges from maximum to minimum

$Iset \rightarrow$ Intensity of the data sets

$\tau \rightarrow$ offset function

2.3.2 RELIABILITY FACTOR

To determine the consistency between the experimental and modelled results in the field of low-energy electron diffraction (LEED), the Reliability Factors (R-Factor) were developed. The aim was to view the data “globally” unlike correlation. Zanazzi, Jona and Van Hove in 1977 developed the first R-factor [38] [39]. They emphasized matching the peak positions rather than peak height by differencing the first derivatives of the data, which compares the full spectrum $F(f)$ as given in the equation 2.

$$F(f) = |I'set_1(f) - CI'set_2(f)| \quad (2)$$

Where,

$I'set \rightarrow$ First derivative of $Iset$

$f \rightarrow$ Maximum to minimum values of the independent axis

Second derivatives were obtained and used to compare the sharp features of the data through the weighting factor $W(f)$ as represented in equation 3.

$$w(f) = \frac{|I''set_1(f) - CI''set_2(f)|}{|I'set_1(f)| + \varepsilon} \quad (3)$$

Where, $\varepsilon = |I'set_1(f)|_{max}$
 $I''set \rightarrow$ Second derivative of $Iset$

The Normalized average intensity C allows the assessment based on the shapes and position of the features is given in equation 4.

$$C = \frac{\sum_{f_{min}}^{f_{max}} |Iset_1(f)|}{\sum_{f_{min}}^{f_{max}} Iset} \quad (4)$$

However, they did not consider the amplitudes. The global integral R_{zj} is determined by introducing reducing factor A to eliminate the dependencies of R-factor such as peaks, slopes, positions and feature with reference to intensity signal as shown in equation 5 and 6.

$$A = \frac{\delta_e}{\sum_{f_{min}}^{f_{max}} Iset_1(f)} \quad (5)$$

Where, $\delta_e = f_{max} - f_{min}$

$$R_{zj} = \frac{A}{\delta_e} \sum_{f_{min}}^{f_{max}} w(f) F(f) \quad (6)$$

Van Hove introduced another widely used R-Factor, proposing five formulae R-factor to measure of position and widths of peaks, the shape of the peaks, number of peaks, shoulders and valleys and their relative heights [39]. Equation 7 and 8 represents Van Hove's proposed R1 and R2.

$$R_1 = \frac{\sum_{f_{min}}^{f_{max}} |Iset_1(f) - CIset_2(f)|}{\sum_{f_{min}}^{f_{max}} |Iset_1(f)|} \quad (7)$$

$$R_2 = \frac{\sum_{f_{min}}^{f_{max}} (Iset_1(f) - CIset_2(f))^2}{\sum_{f_{min}}^{f_{max}} (Iset_1(f))^2} \quad (8)$$

R1 and R2 determined peaks, heights and widths. They did not measure shoulders and valleys. To accommodate this, Van Hove proposed R3, R4, and R5 as given in the equations 9,10 and 11.

$$R_3 = \frac{\text{No. of positive samples in } I'set_1(f)}{\text{No. of positive samples in } I'set_2(f)} \quad (9)$$

$$R_4 = \frac{\sum_{f_{min}}^{f_{max}} |I'_{set_1}(f) - CI'_{set_2}(f)|}{\sum_{f_{min}}^{f_{max}} |I'_{set_1}(f)|} \quad (10)$$

$$R_5 = \frac{\sum_{f_{min}}^{f_{max}} (I'_{set_1}(f) - CI'_{set_2}(f))^2}{\sum_{f_{min}}^{f_{max}} (I'_{set_1}(f))^2} \quad (11)$$

Finally, the Total discrepancy R_T for the signals compared is given by the vector addition of the five R-Factor defined earlier as shown in equation 12.

$$R_T = \sqrt{R_1^2 + R_2^2 + R_3^2 + R_4^2 + R_5^2} \quad (12)$$

These validation techniques involved decomposition of original data. This led to the development of different approaches for the automated validation considering the current techniques and enhancing them. This in turn led to development of the Feature selective validation (FSV) technique [39].

2.4 WHAT IS FSV?

The FSV method was developed by reiterating visual evaluation technique, with the aim of providing more accurate and reliable validation model. Primarily, typical EMC data sets as shown in Figure 13, are compared with the FSV technique. The general shape of this data has rapid variation features superposed onto a slow-moving trend line.

The FSV method decomposes the original data into two major components, namely, Amplitude Difference Measure (ADM) and the Feature Difference Measure (FDM). The ADM measures the difference in overall trend information and the consistency of amplitude of the data sets. While the FDM measures the consistency of rapidly varying features of the data sets. The Global Difference Measure (GDM) is determined by combining the ADM and FDM. The GDM provides an overall measure of the differences between the data sets [40] [41] [42].

The FSV method compares each point of the EMC data sets from origin to end on a point-by-point basis. Thus, providing comparison outputs of ADM_i, FDM_i and GDM_i. The suffix 'i' in the in the FSV outputs represents the point-by-point comparison. Finally, the ADM_i, FDM_i and GDM_i are considered to obtain the histogram for the above measure. These are known as the confidence histograms, represented as ADM_c, FDM_c and GDM_c [37][5]. The confidence histograms provide a visual representation of the distribution of differences between the data sets. Since FSV technique is an enhanced version of visual evaluation method, the result comparison involve interpretation such as “Excellent”, “Very Good”, “Good”, “Fair”, “Poor”,

“Very Poor” [39] [40]. These results are directly related to the visual rating scale as shown in Table 2. The detailed steps to perform FSV algorithm using the IEEE Standard (Std.) 1597.1, which is the IEEE Standard for Validation of Computational Electromagnetics (CEM) Computer Modelling and Simulations [2] is given in Chapter 3, Section 3.1.

FSV value (quantitative)	FSV interpretation (qualitative)	FSV Visual six point scale
Less than 0.1	Excellent	1
Between 0.1 and 0.2	Very good	2
Between 0.2 and 0.4	Good	3
Between 0.4 and 0.8	Fair	4
Between 0.8 and 1.6	Poor	5
Greater than 1.6	Very poor	6

Table 2: FSV Interpretation [40] xDM indicates ADM, FDM or GDM

Moreover, the quality of the results is determined by obtaining the total value for ADM, FDM and GDM which is given by single value and represented as ADM_{tot} , FDM_{tot} and GDM_{tot} [40][41].

The FSV method was broadly used for validation in CEM and EMC applications. However, several challenges were addressed to enhance and make the FSV method practical [42]. The FSV method was directly compared with the visual evaluation method. This was achieved by comparing the confidence histogram outputs from the visual evaluation method and the FSV method. The agreement of the compared outputs was then tested in both fuzzy and float logic. The fuzzy logic is developed to achieve better agreement with visual comparison outputs by overlapping the probability functions of two adjacent categories of the histograms. While the float logic is developed to improve the consistency of visual evaluation by changing the boundary location instead of the probability functions [43] [44]. Moreover, in [45], the FSV method was compared to the visual evaluation method to confirm that the FSV predictions provide a similar benchmark to the visual evaluation method. It also assures that this design technique is applicable to engineers. Other applications where FSV method was used are discussed below.

The FSV is performed to a pair of simple wave forms such as outputs from crossed lines pairs, Edge feature, Gaussian waveform, Antenna Radiation pattern. Here, the outputs from FSV technique seems to respond to variations in amplitude and feature associated aspects of simple

waveforms. As a result, FSV is an excellent tool for comparing data sets quantitatively [46]. The return loss from various twisted pair cables were observed using FSV method to obtain point by point comparisons [41]. S - Parameters of two horn antennas results were obtained from the reverberation chamber and the FSV method was applied on the results to compare them [47]. Real time emissions from electronic circuits were analysed using FSV tool [48].

Two Multiple Input Multiple Output (MIMO) antennas with FR-4 substrate is placed vertically as shown in Figure 14 below. This MIMO antennas was designed, simulated and optimised using Computer simulation technology (CST), which is a software used to design and analyse EM tools. Finally, using the FSV method, the S-parameter results of the MIMO antennas were quantified, and the reliability and confidence histograms were obtained [49].



Figure 14: Design of two MIMO antenna placed vertically [49].

Similarly, in [50] a dual-band MIMO antennas were designed with FR-4 substrate at operating bandwidth of 2.18GHz - 2.51 GHz and 3.25GHz - 3.96 GHz respectively. The S- parameter results and measurement at different bandwidth are compared using FSV method to identify the accuracy of antenna measurements. The Figure 15 below represents the simulation model of the dual-band MIMO antennas.

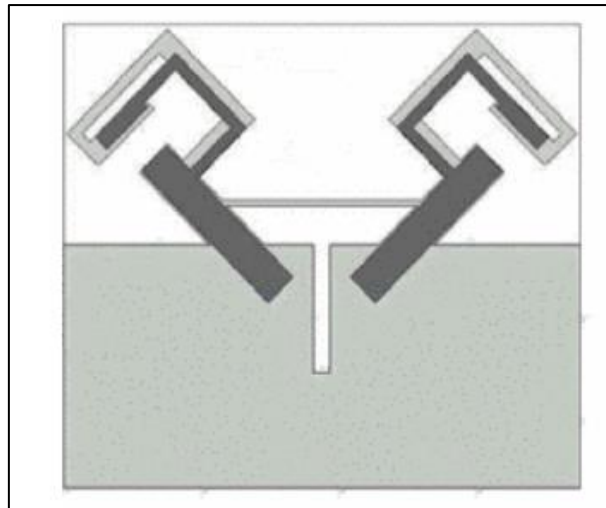


Figure 15: Simulation model of the dual-band MIMO antennas [50].

The FSV method for the transient phenomenon is obtained at three stages, starting from the initial quiescent phase ($t = 0$) to the transient event, then from the transient event to a predefined limit. Finally, there is the post-transient phase, where some energy is still present in the system but is dwindling back to the quiescent state. These stages are analysed by verifying offsets and weighted systems [51]. In [52], the averaging result of FSV was demonstrated by obtaining FSV for multiple data sets and then comparing the overall results. Three such examples are provided in [52]. A 1-dimensional dynamic sea surface is modelled using the Monte Carlo method in [53], and the influence of different wind speeds on the sea surface, gravity spectrum, and tensor spectrum is analysed using the FSV method.

However, the above applications of the FSV were simple examples. Some complex data sets, like aircraft, vehicle-mounted antennas, and others, are discussed below, which would provide a deep understanding for the research study. In [54], controlled radio frequency measures were analysed in U.S. air force aircraft to predict CEM quantities. The effects of the electromagnetic environment were evaluated using the FSV method. Figure 16 below shows the top view of the aircraft, where the highlighted parts are the probes labelled and configured.

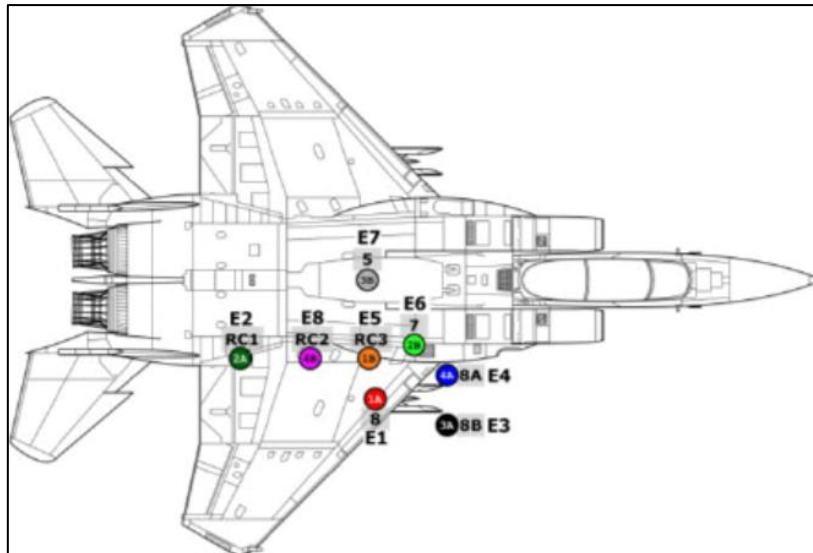


Figure 16: Top view of the aircraft with probes setup [54]

Here, the highlighted areas are where probes are set up and ensure that every wall of the anechoic chamber receives the RF energy as much as possible. Other parts of the aircraft are covered with foam and tested. There were seven different aircraft antennas tested here. These antennas included Very High Frequency (VHF) and Ultra High Frequency (UHF) communication antennas, aircraft navigation antennas, and three locally placed telemetry antennas. The Gaussian derivatives of the antennas were obtained using Computer Simulation Technology (CST). Computer Simulation Technology is a software for designing, analysing and optimising electromagnetic (EM) components and systems. Now the physical output and CST output were compared using FSV method [54]. This application tests only a part of an object and hence differs from other applications of FSV.

In [55], similar aircraft is exposed to High intensity Radiated Field (HIRF) to test their robustness against catastrophic effects. This is done using non-intrusive method, which means there is no disruption to the aircraft during the testing process. This ensures that the testing of the model is safe and accurate, while also providing useful results on the aircraft's robustness to HIRF effects. Finally, the FSV method is applied to the simulated and measured results of the HIRF effects to obtain the reliability and confidence histograms. Similarly, in [56], the shielding effectiveness of aircraft under HIRF conditions was tested. The results were obtained from two numerical tools and compared using the FSV method.

The Radar Cross Section (RCS) measurement of a Brahmos missile in [57] is positioned horizontally, and the incident wave is polarised vertically. The observation angle for pitch is 0

degrees, and for orientation, it is 0 -180 degrees, with a 1-degree interval. Figure 17 below represent the schematic of the Brahmos missile positioned at the observation angle.

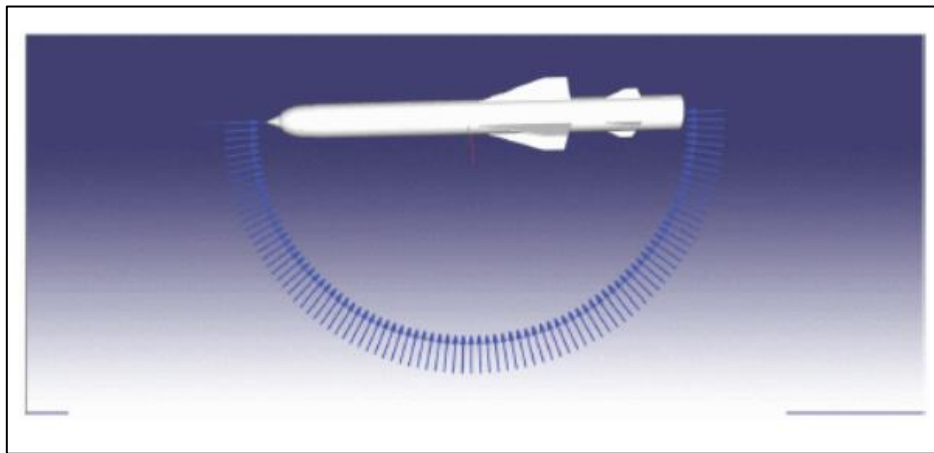


Figure 17: Brahmos missile's schematic showing observation angle [57].

Now, the RCS at frequencies 1 GHz and 6 GHz is calculated. This missile is tested across various electromagnetic measurements, such as the Physical Optics (PO) method, the Finite Element Method (FEM), and the Moment Method (MoM). These results are compared using the FSV method. Unlike the above applications, here three sets of data are compared. Therefore, this leads to the comparison of multiple data sets.

In [58], the FSV method is used to compare multiple data sets. Here, the Analytic Hierarchy Process (AHP) is used to observe multiple data sets. An example of the cross-talk model is shown in Figure 18 below. This AHP approach increases the number of data sets. Therefore, pairwise comparison of data sets is done layer by layer, and then a weighted FSV is obtained. This provides a clear output for multiple data sets.

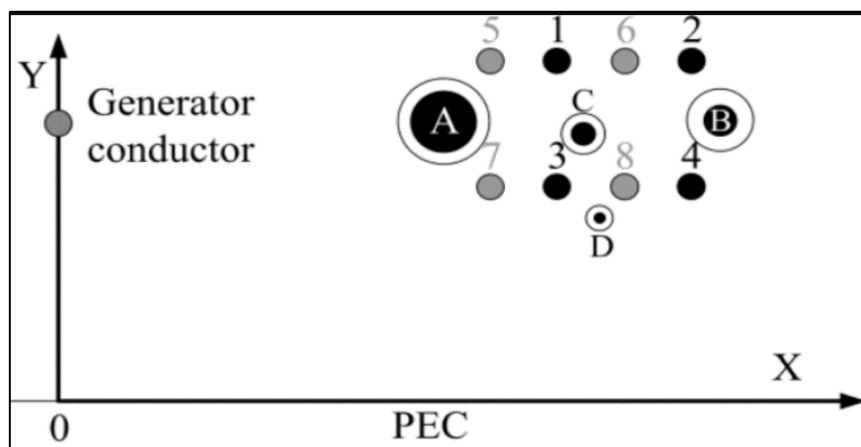


Figure 18: Cross talk model [58]

The quantification of FSV by removing or reducing data points is done by down-sampling and under-sampling [59]. The down-sampling is done by resampling the original data sets and creating a subset, as shown in Figure 19 below. These subsets are then compared using the FSV method. On the other hand, the under-sampling approach involves missing data randomly or continuously. The missed data points are predicted by the linear interpolation method, and then FSV is applied to them.

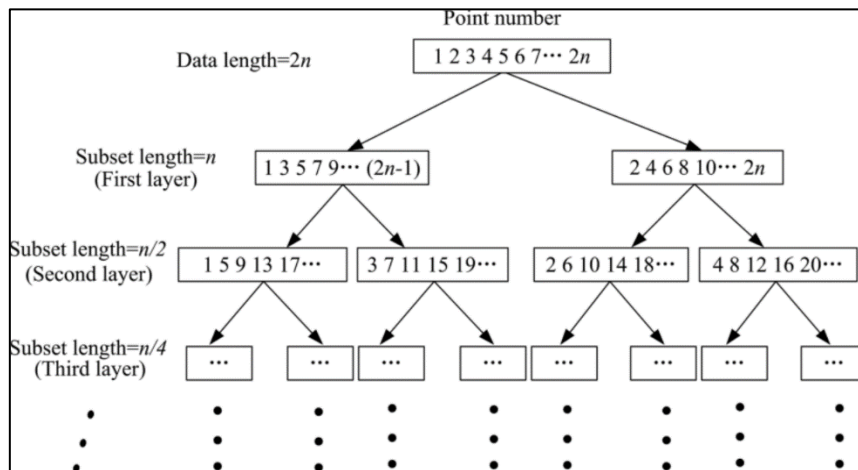


Figure 19: Down sampling method [59]

As seen above, with the developing applications of FSV within the EMC and CEM communities, more complex data such as images, 2-D or 3-D electric and magnetic fields, and surface currents on an antenna device or from an aircraft during lightning are inevitable. However, this is challenging when the number of parameters in data sets increases [60].

For this, the multi-dimensional FSV method was developed, which is explained in the below section.

2.5 MULTIDIMENSIONAL FSV

A data is called multidimensional when the number of degrees of freedom in the data sets develops. In the 2-D FSV method, the Amplitude Difference Measure (ADM), Feature Difference Measure (FDM) and the Global Difference Measure (GDM) are obtained for both X and Y direction which is represented (x,y). Unlike, 1-D FSV, a sub measure of the low frequency components are considered [61]. Initially, when the 2-D FSV method was developed and implemented, 2-D data were treated as folded 1-D data and thus only first order representation of 2-D filters were used to achieve this. Although this method was simple and easy to implement, It does not allow the nD data to differ or vary in it features in all direction. This is because the converted 1-D data were collected together in one direction. Therefore, this

method did not provide convincing results [63]. Thus, this approach was limited to 2-D data sets.

This in turn, led to the general approach followed in standard 1-D FSV, that was performed by applying 2-D Fourier transform for the data instead of 1-D Fourier transform [62]. This in turn provides xDM's which can be scaled to the standard FSV interpretation scale specified in table (1) above. However, this approach was challenging when the number of data points differ in x and y direction.

To overcome these limitations, an iterative process to study the data is given in [63]. Here, 1-D FSV is repeatedly used on every row and column of data. Now, the weighted root square at each point of intersection between row and column data are given to obtain the combined ADM, FDM and GDM. This in turn provides values that can be interpreted using normal 1-D FSV interpretation scale as in [2]. This is further explained in detail under chapter 3 section 3.1.6. Moreover, this approach provides xDMi's for data which has different data points in x and y direction and thus providing the number of degrees of freedom. On the other hand, the major advantage of this approach is that it can be used on nD data without any changes in its process, thus allowing improvements and enhancements in 1-D FSV [64] [65].

Extending the dimensionality of the data play a vital role in development and improvement in FSV method. When 2-D FSV is extended in 3-D FSV, it uses the generalised nD approach which is to repeatedly use 1-D FSV as given in [58]. In the 3-D FSV, the Amplitude Difference Measure (ADM), Feature Difference Measure (FDM) and the Global Difference Measure (GDM) are obtained for X, Y and Z direction which is represented (x,y,z) [64]. The most challenging aspects of nD FSV is to obtain visual assessment to verify its output as it is difficult to visually evaluate data in higher dimensions [64] [65].

Few applications to verify 2-D and 3-D FSV are given below:

In [61] simulations for the roof antennas mounted on vehicles are analysed using 2-D FSV method. This is done based on similarity degree, which is represented as ADM, FDM and GDM in 1-D FSV. The field distribution of vehicle mounted antennas are compared using 2-D FSV method in [66]. Here, two monopole antennas are placed in the rear roof and rear wing of the vehicle where it is surrounded by the electric field. .

The similarity between the measured, computed data and 2-D FSV method is analysed which separated the amplitude and feature of the data. On the other hand, tolerance value for the ADM, FDM and GDM are considered for the purpose of evaluation scales. Similar approach

of 2-D FSV is given in [62] a small difference here is that the DC components on the data is also considered to provide the xDMi's.

Now, both the 2-D and 3-D FSV algorithm were applied on a LIVE video and analysed in [63] and [64] respectively.

The field distribution of vehicle mounted antennas are compared using 2-D FSV method in [66] two monopole antennas are placed in the rear roof and rear wing of the vehicle where it is surrounded by the electric field. The similarity between the measured, computed data and 2-D FSV method is analysed which separated the amplitude and feature of the data.

In [67] the Stochastic Collocation Method (SCM) is used to measure uncertain data. In SCM, the collocation points increase exponentially with increase of dimensionality. Therefore, reducing the computational efficiency. To overcome this, a new SCM scheme called Dimension-Reduced Sparse Grids Scheme (DRSG-SCM) is developed where the number of collocation points is proportional to the dimensionality. Here, simulation of two shielding effectiveness is built using Computer Simulation Technology (CST) software, which include random size and placement of an aperture in a shielded box. Then the 2-D and 3-D FSV technique is applied to the shielding box to obtain the performance of the DRSG-SCM method.

The major strengths and drawbacks of 2-D FSV is elaborated in [68]. Here, an Image quality assessment (IQA) is performed. The IQA data base generally holds distortion free reference images. Now the 2-D FSV method is applied on these images. It is concluded by saying that the 2-D FSV method is more accurate compared to the LIVE data results mentioned in [61] [62].

In [69] the major challenges faced in building an n-dimensional FSV are discussed. This includes, a mathematical framework for FSV, mirroring with visual perception and positive and negative transient points.

2.6 PROBABILITY DENSITY FUNCTION (PDF)

Generally, point-by-point FSV provides qualitative and quantitative results that could directly link with visual evaluation. These outputs are then binned into a confidence histogram with six categories (Excellent, Very Good, Good, Fair, Poor, Very Poor) as shown in table 1 [70]. These confidence histograms are one of the simplest techniques used in FSV to interpret the quality of the data [70] [71].

However, the broad classification in histograms could mask some of the detail in the comparative result distribution [70]. Also, the number of bins opted, plays a vital role in interpreting histograms as in [71]. On the other hand, it is difficult to obtain a smooth density function from a histogram due to the number of bins used, it is either under- smoothed or over-smoothed histograms [72]. When FSV is applied in multiple comparisons such as model improvements or data optimisation, the histograms are not sufficiently qualified to interpret these results as it is difficult to overlay two or more histograms together for comparing the data [71].

To overcome the above discussed issues, and a smooth density function; continuous Probability Density Function (PDFs) is derived from the point-by-point FSV. The PDF can represent the distribution of quantitative results in simpler form when compared to the confidence histograms. Moreover, PDF's can provide significant comparisons between iterative measurements, between measured versus visual data and different computational models [60] [71]. The use of PDF's in multiple data comparisons is given in [71]. To further investigate the results, non-parametric test methods such as Kolmogorov-Smirnov (KS) test can be used in PDF's. This test is robust and does not make any assumption on the distribution of the data [70] [71].

Further, PDFs allow statistical moments such as variance, skewness and kurtosis to analyse FSV performance. It is a key advantage of using FSV technique. In datasets, variance provides the details on the dispersion, skewness provides measure of symmetry of the shape and kurtosis provides measure of flatness [70].

The steps followed to obtain PDF, KS-test and other statistical moments and further explained in detail in chapter 3.

2.7 REFERENCES

- [1] M. N. O. Sadiku, A. E. Shadare, and S. M. Musa, "Computational electromagnetics," *International Journal of Engineering Research*, vol. 6, no. 8, p. 414, 2017.
- [2] "IEEE Standard for Validation of Computational Electromagnetics Computer Modelling and Simulations".
- [3] Y. Yulindon, N. Samsudin, and R. Efendi, "Optimization of E-shaped microstrip patch antenna using particle swarm optimization (PSO) for Wideband Application,"

- International Journal of Intelligent Engineering and Systems*, vol. 13, no. 4, pp. 305–315, 2020.
- [4] R. D. Bhavani and D. J. L. Narayana, “Microstrip H-shape patch antenna’s analysis and layout,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 10, pp. 498–501, 2019.
- [5] H. Barrett and K. Myers, *Foundations of Image Science*. Wiley, 2013.
- [6] M. Wilkinson and A. Doan, *Essential optics review for the boards*. Corralville, Iowa: F.E.P. International, Inc., 2009.
- [7] C. Solomon and T. Breckon, *Fundamentals of digital image processing*. Hoboken, N.J.: Wiley, 2013.
- [8] L. Tan, "Image file formats", *Biomedical Imaging and Intervention Journal*, vol. 2, no. 1, 2006.
- [9] M. Terras, *Digital images for the information professional*. 2016.
- [10] BBC Bitesize. 2022. *Digital images - Encoding images - GCSE Computer Science Revision - BBC Bitesize*. [online] Available at: <<https://www.bbc.co.uk/bitesize/guides/zqyrq6f/revision/1>> [Accessed 08 March 2022].
- [11] B. Liu, R. Rajasekar, A. Shukla, A. Solomon, L. Xu, A. Saravanan and J. Kuleshov, "Integrating Natural Language Processing & Computer Vision into an Interactive Learning Platform", *2020 IEEE MIT Undergraduate Research Technology Conference (URTC)*, 2020
- [12] N. Ibraheem, M. Hasan, R. Zaman Khan and P. Mishra, "Understanding Colour Models: A Review", *ARPN Journal of Science and Technology*, vol. 2, no. 3, 2012
- [13] S. Jayaraman, S. Esakkirajan and T. Veerakumar, *Digital image processing*. New Delhi: Tata McGraw-Hill Education, 2009.
- [14] T. Kumar and K. Verma, "A Theory Based on Conversion of RGB image to Gray image", *International Journal of Computer Applications*, vol. 7, no. 2, 2010
- [15] R. Vidhya, P. Ashritha, M. Kumar and R. N., "Image Noise Declining Approaches by Adopting Effective Filters", *2019 International Conference on Emerging Trends in Science and Engineering (ICESE)*, 2019.
- [16] M. Dwairi, Z. Alqadi, A. A Abujazar and R. Abu Zneit, "Optimized True-Color Image Processing", *World Applied Sciences Journal*, vol. 8, no. 10, 2010
- [17] K. Lakhwani and P. Murarka, "Color Space Transformation for Visual Enhancement of noisy color Image", *International Journal of ICT and Management*, vol. 3, no. 2, 2015 Available:https://www.researchgate.net/publication/287994342_Color_Space_Transformation_for_Visual_Enhancement_of_noisy_color_Image. [Accessed: 15- Mar- 2022]
- [18] E. Fiume, *The mathematical structure of raster graphics*. Boston: Acad. Press, 1990.

- [19] R. Lloyd Cody, J. Cosmas and E. Tseklevs, "Open-standards rich media mobile platform & rapid deployment service creation tool", p. 2, 2008 Available: https://www.researchgate.net/publication/49401211_Open-standards_rich_media_mobile_platform_rapid_deployment_service_creation_tool. [Accessed: 15- Mar- 2022]
- [20] A. Skodras, C. Christopoulos and T. Ebrahimi, "The JPEG 2000 still image compression standard", *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36-58, 2001
- [21] J. Kabachinski, "TIFF, GIF, and PNG: Get the Picture?", *Biomedical Instrumentation & Technology*, vol. 41, no. 4, pp. 297-300, 2007.
- [22] R. Wiggins, H. Davidson, H. Harnsberger, J. Lauman and P. Goede, "Image File Formats: Past, Present, and Future", *RadioGraphics*, vol. 21, no. 3, pp. 789-798, 2001.
- [23] G. Roelofs, *PNG: The Definitive Guide*, 2nd ed. O'Reilly Media, 2003.
- [24] J. Murray and W. Ryper, *Encyclopedia of graphics file formats*, 2nd ed. Cambridge: O'Reilly, 1996.
- [25] A. Kour, V. Yadav, V. Maheshwari and D. Prashar, "A Review on Image Processing", *International Journal of Electronics Communication and Computer Engineering*, vol. 4, no. 1, 2013.
- [26] B. Benchamardimath and R. S Hegadi, "A STUDY ON THE IMPORTANCE OF IMAGE PROCESSING AND ITS APLICATIONS", *International Journal of Research in Engineering and Technology*, 2020 [Online]. Available: https://www.researchgate.net/publication/340827009_A_STUDY_ON_THE_IMPORTANCE_OF_IMAGE_PROCESSING_AND_ITS_APLICATIONS. [Accessed: 15- Mar- 2022]
- [27] T. Prabakaran, P. Periasamy, V. Mugendiran and Ramanan, "Studies on application of image processing in various fields: An overview", *IOP Conference Series: Materials Science and Engineering*, vol. 961, 2020.
- [28] M. Ekstrom, *Digital image processing techniques*. Burlington: Elsevier Science, 2012
- [29] T. Acharya and A. Ray, *Image processing*. Hoboken, N.J.: Wiley, 2005.
- [30] S. S.Agaian, K. P.Lentz and r. M. Grigoryan, "A New Measure of Image Enhancement", *IASTED International Conference on Signal Processing & Communication*, 2000 [Online]. Available: https://www.researchgate.net/publication/244268659_A_New_Measure_of_Image_Enhancement. [Accessed: 15- Mar- 2022]
- [31] J. R. S. Tavares, "Image Processing and Analysis: Applications and Trends", *ES-ATEMA' 2010 Fifth International Conference on Advances and Trends in Engineering Materials and their Applications*, 2010 [Online]. Available:

https://www.researchgate.net/publication/228658054_Image_Processing_and_Analysis_Applications_and_Trends. [Accessed: 15- Mar- 2022]

- [32] E. Fomalont, "Image Analysis", *Synthesis Imaging in Radio Astronomy*, vol. 180, p. 301, 1999.
- [33] S. Umbaugh, *Computer imaging*. Boca Raton: Taylor & Francis, 2005.
- [34] P. Hockberger, *Why Do We Need Image Processing?* J Biomol Tech, 2013 Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3635309/#>. [Accessed: 15- Mar- 2022]
- [35] R. S Hegadi, "Image Processing: Research Opportunities and Challenges", *National Seminar on Research in Computers*,, 2010.
- [36] A. Gupta, "Current research opportunities of image processing and computer vision", *Computer Science*, vol. 20, no. 4, 2019.
- [37] R. S Hegadi, "Image Processing: Research Opportunities and Challenges", *National Seminar on Research in Computers*,, 2010.
- [38] A. Duffy, A. Martin, A. Orlandi, G. Antonini, T. Benson and M. Woolfson, "Feature Selective Validation (FSV) for Validation of Computational Electromagnetics (CEM). Part I—The FSV Method", *IEEE Transactions on Electromagnetic Compatibility*, vol. 48, no. 3, pp. 449-459, 2006.
- [39] A. Martin, "QUANTITATIVE DATA VALIDATION (AUTOMATED VISUAL EVALUATIONS)", PhD Thesis, De Montfort University, 1999.
- [40] V. Rajamani, C. Bunting, A. Orlandi and A. Duffy, "Introduction to feature selective validation (FSV)", 2006 IEEE Antennas and Propagation Society International Symposium, 2006.
- [41] O. Ogundapo, A. Duffy, C. Nche, Z. Gang, F. Akinuoye and V. Kang, "Cable Measurement Assessment using the Feature Selective Validation Method", *Cable Measurement Assessment using the Feature Selective Validation Method*, 2013.
- [42] A. Orlandi, A. Duffy, B. Archambeault, G. Antonini, D. Coleby and S. Connor, "Feature Selective Validation (FSV) for Validation of Computational Electromagnetics (CEM). Part II— Assessment of FSV Performance", *IEEE Transactions on Electromagnetic Compatibility*, vol. 48, no. 3, pp. 460-467, 2006.
- [43] D. Di Febo, F. de Paulis, A. Orlandi, G. Zhang, H. Sasse, A. Duffy, L. Wang and B. Archambeault, "Investigating Confidence Histograms and Classification in FSV: Part I. Fuzzy FSV", *IEEE Transactions on Electromagnetic Compatibility*, vol. 55, no. 5, pp. 917-924, 2013.

- [40] to [44] G. Zhang, H. Sasse, A. Duffy, L. Wang, D. Di Febo, F. de Paulis, A. Orlandi and B. Archambeault, "Investigating Confidence Histograms and Classification in FSV: Part II-Float FSV", IEEE Transactions on Electromagnetic Compatibility, vol. 55, no. 5, pp. 925-932, 2013.
- [45] A. P. Duffy, A. Orlandi and G. Zhang, "Notice of Retraction: Review of the Feature Selective Validation Method (FSV). Part I—Theory," in IEEE Transactions on Electromagnetic Compatibility, vol. 60, no. 4, pp. 814-821, Aug. 2018.
- [46] L. Hiltz, "Characterization study of the Feature Selective Validation (FSV) technique on simple and complex waveforms", 2009 IEEE International Symposium on Electromagnetic Compatibility, 2009.
- [47] G. Hankins and D. Lewis, "Validating the FSV Method using reverberation chamber measurements", 2010 IEEE International Symposium on Electromagnetic Compatibility, 2010.
- [48] A. Colin, M. Perotoni, K. Marconi, E. Ferreira, M. Andrade, S. Marchiori, M. Menezes and A. Nogueira, "Feature selective validation analysis applied to measurement and simulation of electronic circuit electromagnetic emissions", 2017 International Symposium on Electromagnetic Compatibility - EMC EUROPE, 2017.
- [49] X. Jiang, J. Mei and T. Jiang, "Verify microstrip antenna using feature selective verification method - IEEE Conference Publication", Ieeexplore.ieee.org, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8052014>.
- [50] P. Xu, X. Jiang and D. Ming, "Application of feature selective validation to the design of microstrip antenna", 2017 Progress In Electromagnetics Research Symposium - Spring (PIERS), 2017.
- [51] R. Telleria, F. Silva, A. Orlandi, H. Sasse and A. Duffy, "Factors influencing the successful validation of transient phenomenon modelling", 2010 Asia-Pacific International Symposium on Electromagnetic Compatibility, 2010.
- [52] B. Archambeault and J. Diepenbrock, "Quantifying the quality of agreement between simulation and validation data for multiple data sets", 2010 Asia-Pacific International Symposium on Electromagnetic Compatibility, 2010.
- [53] N. Bi, X. Wang, T. Jiang and Y. Zhang, "Analysis and evaluation of one-dimensional sea surface features based on FSV", 2017 International Applied Computational Electromagnetics Society Symposium - Italy (ACES), 2017.

- [54] M. Johnson, B. Hays, J. Bean and O. Ali, "Comparison of in situ aircraft electromagnetic environment measurements with time domain simulations", 2017 IEEE International Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI), 2017.
- [55] Y. Wu, H. Zhou, H. Li, X. Bao and L. Zhong, "Validation of Electromagnetic model for an Aircraft Under HIRF Condition With the Non-Intrusive Polynomial Chaos Methods", 2019 International Symposium on Electromagnetic Compatibility - EMC EUROPE, 2019.
- [56] P. Xu, X. Jiang and D. Ming, "Application of feature selective validation to the design of microstrip antenna", 2017 Progress In Electromagnetics Research Symposium - Spring (PIERS), 2017.
- [57] Yonghu Zeng, Lei Gao, Liandong Wang and Jinliang Li, "Comparison analysis of calculation results for target scattering cross section based on feature selective validation", 2016 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEMC), 2016.
- [58] G. Zhang, L. Wang, A. Duffy, D. Di Febo, A. Orlandi and H. Sasse, "Applying the Analytic Hierarchy Process (AHP) to an FSV-Based Comparison of Multiple Datasets", *IEEE Transactions on Electromagnetic Compatibility*, vol. 57, no. 3, pp. 477-483, 2015.
- [59] G. Zhang, L. Wang, A. Duffy, H. Sasse, D. Di Febo, A. Orlandi and K. Aniserowicz, "Downsampled and Undersampled Datasets in Feature Selective Validation (FSV)", *IEEE Transactions on Electromagnetic Compatibility*, vol. 56, no. 4, pp. 817-824, 2014.
- [60] A. Duffy and G. Zhang, "FSV: State of the art and current research fronts", *IEEE Electromagnetic Compatibility Magazine*, vol. 9, no. 3, pp. 55-62, 2020.
- [61] A. Martin, A. Ruddle and A. Duffy, "Comparison of measured and computed local electric field distributions due to vehicle-mounted antennas using 2D feature selective validation", 2005 International Symposium on Electromagnetic Compatibility, 2005. EMC 2005.
- [62] A. Orlandi, G. Antonini, C. Polisini, A. Duffy and H. Sasse, "Progress in the development of a 2D Feature Selective Validation (FSV) method", 2008 IEEE International Symposium on Electromagnetic Compatibility, 2008
- [63] G. Zhang, A. Duffy, A. Orlandi, D. Febo, L. Wang and H. Sasse, "Comparison of Data With Multiple Degrees of Freedom Utilizing the Feature Selective Validation Method", *IEEE Transactions on Electromagnetic Compatibility*, vol. 58, no. 3, pp. 784-791, 2016.

- [64] G. Zhang, A. Orlandi, A. Duffy and L. Wang, "Comparison of Three-Dimensional Datasets by Using the Generalized n-Dimensional (n-D) Feature Selective Validation (FSV) Technique", *IEEE Transactions on Electromagnetic Compatibility*, vol. 59, no. 1, pp. 103-110, 2017.]
- [65] A. Drozd, B. Archambeault, A. Duffy and I. Kasperovich, "Development of next generation FSV tools and standards", *2012 IEEE International Symposium on Electromagnetic Compatibility*, 2012
- [66] A. Martin, A. Ruddle and A. Duffy, "Comparison of measured and computed local electric field distributions due to vehicle-mounted antennas using 2D feature selective validation", *2005 International Symposium on Electromagnetic Compatibility*, 2005. EMC 2005.
- [67] J. Bai, G. Zhang, A. Duffy and L. Wang, "Dimension-Reduced Sparse Grid Strategy for a Stochastic Collocation Method in EMC Software", *IEEE Transactions on Electromagnetic Compatibility*, vol. 60, no. 1, pp. 218-224, 2018.
- [68] G. Zhang, A. Orlandi and A. Duffy, "Using Image Quality Assessment (IQA) Databases to Provide an Appraisal of the Ability of the Feature Selective Validation Method (FSV) to Compare Two-Dimensional Datasets", *IEEE Transactions on Electromagnetic Compatibility*, vol. 60, no. 4, pp. 890-898, 2018.
- [69] Archambeault et al., "Challenges in developing a multidimensional Feature Selective Validation implementation," *2010 IEEE International Symposium on Electromagnetic Compatibility*, Fort Lauderdale, FL, 2010, pp. 726-731.
- [70] G. Zhang, H. Sasse, L. Wang, and A. Duffy, "A Statistical Assessment of the Performance of FSV", *ACES Journal*, vol. 28, no. 12, pp. 1179–1186, Sep. 2021.
- [71] Z. Gang, A. Duffy, H. Sasse and W. Lixin, "The use of probability density functions to improve the interpretation of FSV results", *2012 IEEE International Symposium on Electromagnetic Compatibility*, 2012.
- [72] J. S. Simonoff *Smoothing methods in statistics*. New York: Springer Verlag 199

CHAPTER - 3

FSV IMPLEMENTATION

As seen in Chapter 2, Section 2.3, the Feature Selective Validation (FSV) method was developed to aid comparisons performed as part of the validation of Computational Electromagnetics (CEM). It primarily focuses on applications in Electromagnetic compatibility (EMC).

The FSV is a data comparison technique based on the decomposition of the original data into amplitude and feature data [1] [4]. In addition to these outputs, the FSV method also provides a single-value goodness of fit, which is a statistical measure of how well the two data sets match. This measure can be used to determine the overall similarity between the data sets. Moreover, density functions and statistical analysis are also calculated to provide further insights into the comparison results.

Therefore, in this chapter, steps on how to apply the FSV method to data sets and interpret the results, making it a valuable tool for comparison and validation in various fields, particularly in the field of Electromagnetic Compatibility (EMC) is discussed in detail. The implementation procedures for the FSV method are explained in this chapter, Section 3.1, including the 2-D FSV method and its extension to the n-D FSV approach in Section 3.2.

3.1 IMPLEMENTATION OF 1-DIMENSIONAL FSV

The basic procedure to implement FSV consists of 6 major steps, which are performed on a range of working datasets. Figure 1, below represents the process flow chart of the steps developed to implement FSV method.

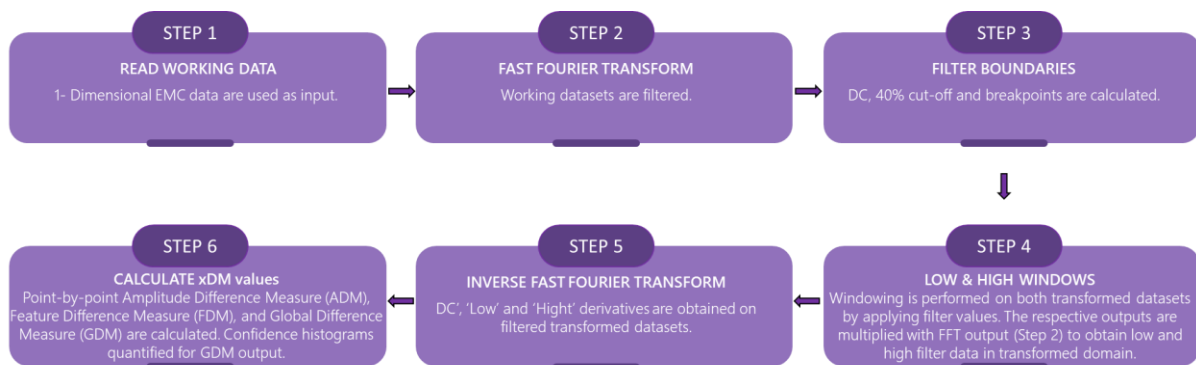


Figure 1: Steps to Implement 1- Dimensional FSV

These steps were further developed using MATLAB, a programming language developed by MathWorks and used as a computation tool in various applications including Mathematics, Science and Engineering [2]. MATLAB is proprietary software that is widely used for numerical computing, data analysis, and visualisation. It has a vast range of capabilities and

tools that make it ideal for working with large and complex datasets. Moreover, it provides a user-friendly graphical interface that allows users to interact with their data and visualise it in various ways, including 2D and 3D plots, histograms, and heatmaps. Additionally, MATLAB supports the integration of algorithms and interfacing programmes written in other programming languages, such as C, C++, and Java, which makes it more versatile and powerful [2] [3]. Therefore, it was a practical choice to use MATLAB in this study instead of developing a native language implementation, which could be more time-consuming and less efficient.

The FSV implementation steps developed using MATLAB is given in Appendix 1. Figures provided in each step were obtained using this program.

3.1.1 STEP 1: To read working data set

To implement the FSV method, a range of overlapping data sets are used. These data are read and interpolated, to ensure that the data points are coincident, which allows the later manipulation of the data and presentation of the results. If the length of the data sets is not the same, data with the smaller length is zero padded to equalise the largest length [4][5].

For example, two 1-dimensional data sets obtained from a mode-stirred reverberation chamber that show small variations in their features are used as the input data sets to provide a detailed understanding of the FSV implementation. These input data sets are shown in Figure 2 below.

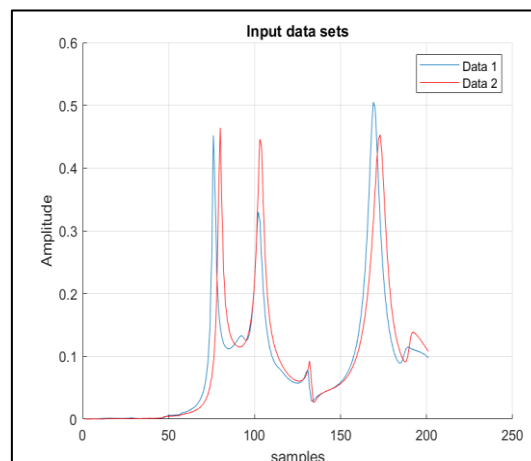


Figure 2: Input Data 1 and 2 for comparison

3.1.2 STEP 2: Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) is used to simplify the filtering of the data. This is performed for both the working data sets [4][5]. The output provides transformed data sets as shown in the figure 2 for the original data shown in Figure 3.

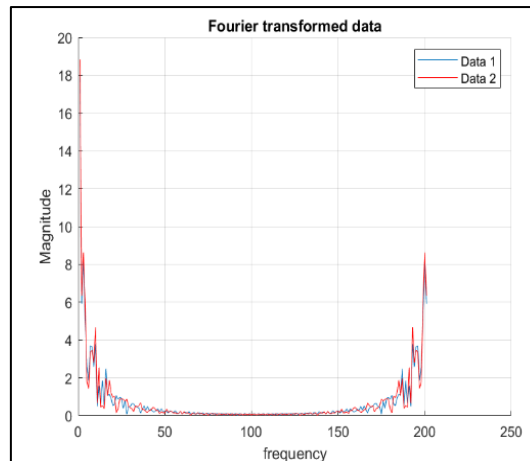


Figure 3: Fourier transformed data 1 and 2

3.1.3 STEP 3: To calculate the filter boundaries (the DC, 40% cut off and break point)

Now, the input data is considered in three different parts in the filtered domain. The first region includes the DC level which is called “Low pass” or “DC”. The second region includes the data overall envelop of the data called the “Band pass” or “Low”. This “Low” region contains approximately 40% of the area under the resulting output. Lastly, the third region that includes the varying features of the data sets called the “High pass” or “High”. Therefore, this step calculates all three filter boundaries as discussed below.

The first five points provide the low pass filter, incorporating the 0-point term, hence is called the DC set for convenience. These points are then not included in the remaining band-pass and high-pass filters. This is done to avoid offset terms swamping the other data[4][6]. The DC is obtained for both transformed data sets as shown in Figure 4 below:

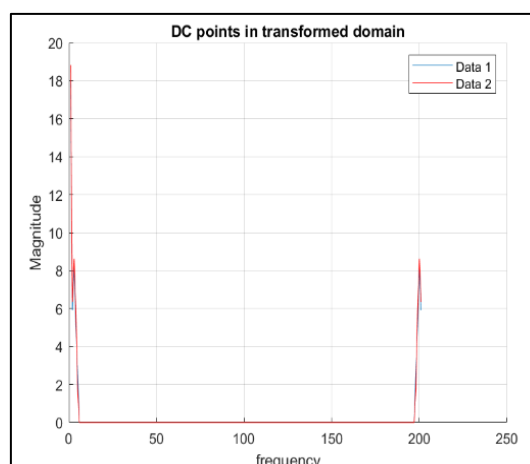


Figure 4: DC points for data 1 and 2 in the transformed domain

Next, the 40% cut off, which determines the upper bound of the band-pass filter is obtained by first determining the energy in the remaining data by summing the values of the independent variable, starting from the fifth point of the data to end of the data as in equation 1 below [5][6].

$$S = \sum_{i=5}^N TWDS(i) \quad (1)$$

Where,

$S \rightarrow$ Sum of independent variables

$i \rightarrow$ Element of the data set

$N \rightarrow$ Total number of data points in the set

$TWDS(i) \rightarrow$ i^{th} independent variable in the transformed domain.

As 40% cut off point is determined, it is verified with the sum of independent variables as given in the equation 2 below [5][6].

$$\sum_{i=5}^{i_{40\%}} TWDS(i) \geq 0.4 \times S \quad (2)$$

Where,

$i_{40\%} \rightarrow$ Element containing 40% location.

The value of 'i' is systematically increased from the fifth data point to the 40% cut off until the condition is satisfied. Now, the break points are determined by adding five data points above the 40% location, to allow transition between Low and High windows [4][5][6]. This is represented in equation 3 below,

$$i_{bp} = i_{40\%} + 5 \quad (3)$$

Where,

$i_{bp} \rightarrow$ Element number of the break point

In step 3, the 40% points for data sets 1 and 2 are at points 14 and 12, respectively. Thus, the breakpoints are calculated by adding a small offset of 5 points, providing 19 and 17 for data sets 1 and 2.

The addition of five data points to the 40% cut-off value improves the accuracy and reliability of the results obtained from the analysis. This is because these additional points provide a comfortable transition window between the low and high results, ensuring that no important

data is excluded during the analysis. Finally, the lowest break point value was chosen from the two individual break points calculated.

3.1.4 STEP 4: To obtain low and high windows

Windowing is performed on both transformed datasets. This is achieved by applying filter values from the Table 1 and 2 to the independent variables within the transformed domain. [5][6]. The Table 1 represents the element number and filter values to calculate Low values while Table 2 represents the same for High values.

Element number	Filter value
I_{bp-3}	1.000
I_{bp-2}	0.834
I_{bp-1}	0.667
I_{bp}	0.500
I_{bp+1}	0.334
I_{bp+2}	0.167
I_{bp+3}	0.000

Table 1: Filter Values for Low windows

Element number	Filter value
I_{bp-3}	0.000
I_{bp-2}	0.167
I_{bp-1}	0.334
I_{bp}	0.500
I_{bp+1}	0.667
I_{bp+2}	0.834
I_{bp+3}	1.000

Table 2: Filter Values for High windows

The results from the above step provide ‘Low’ and ‘High’ windows as shown in Figure 4 and 5. Figure 5 represents the ‘Low’ and ‘High’ window for data 1, while Figure 6 represents the ‘Low’ and ‘High’ window for data 2.

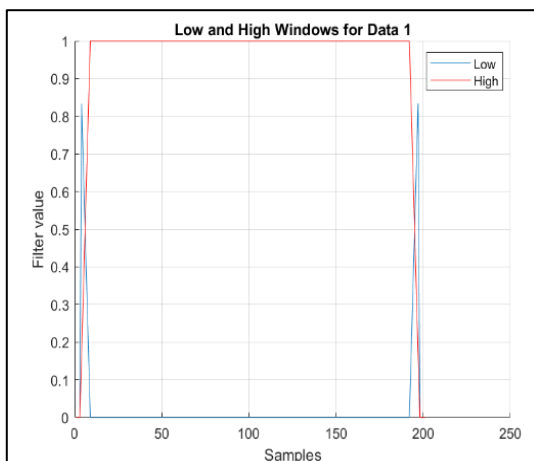


Figure 5: Low and High windows for data 1

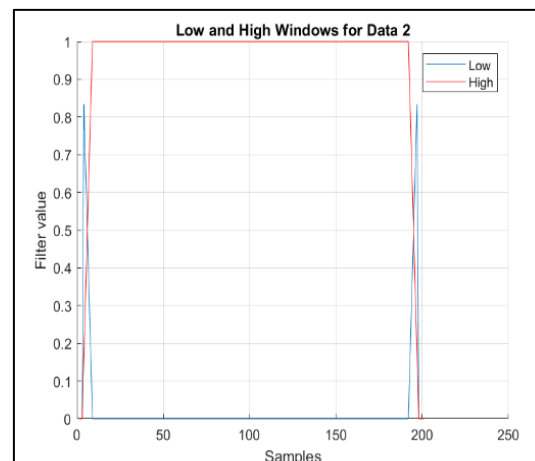


Figure 6: Low and High windows for data 2

The above results are then multiplied with the output from STEP 2; which is the transformed data sets to provide Low filter and High Filter data sets in the transformed domain as shown in Figure 7 and 8 respectively.

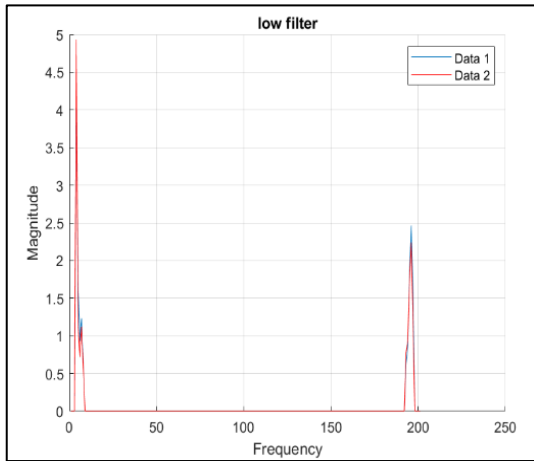


Figure 7: Low filter data 1 and 2

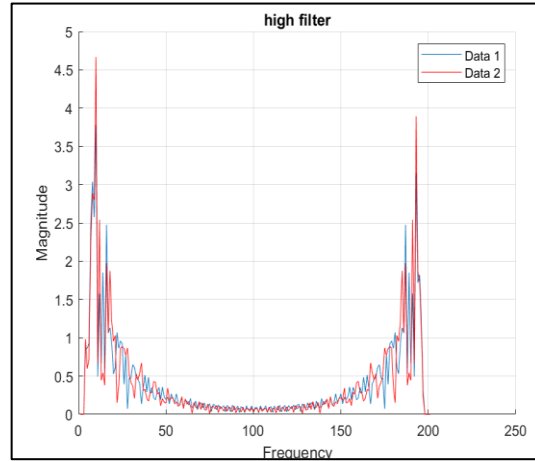


Figure 8: High filter data 1 and 2

3.1.5 STEP 5: Inverse Fast Fourier Transform (IFFT)

Inverse Fast Fourier Transform (IFFT) is applied on the filtered transformed datasets [5][6]. Now the derivatives of ‘DC’, ‘Low’ and ‘High’ are obtained as shown in Figure 9,10 and 11 respectively.

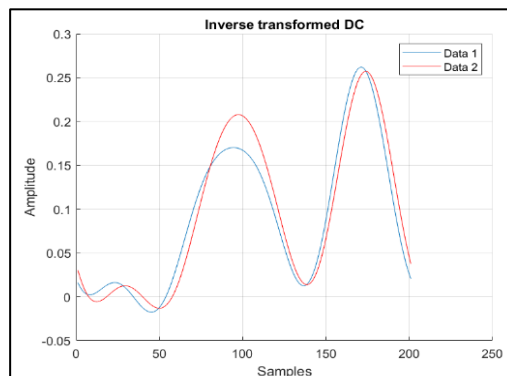


Figure 9: Inverse transformed DC for data 1 and 2

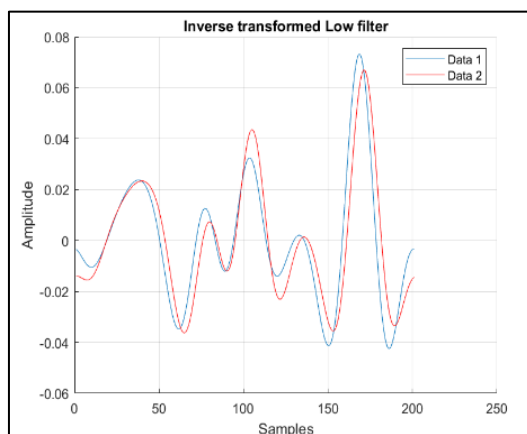


Figure 10: Inverse transformed low filter for data 1 and 2

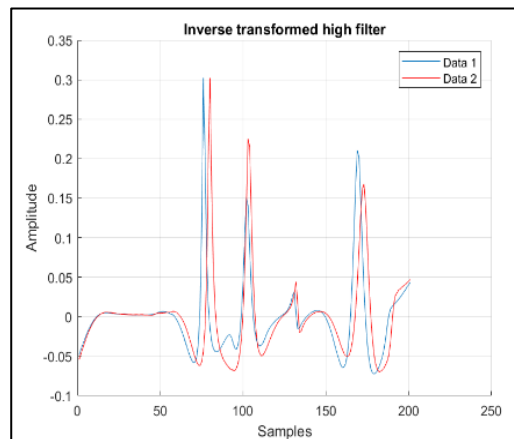


Figure 11: Inverse transformed High filter for data 1 and 2

3.1.6 STEP 6: To calculate difference measures

In this step, point-by-point Amplitude Difference Measure (ADM), Feature Difference Measure (FDM), and Global Difference Measure (GDM) are calculated. These are abbreviated as xDM values, where x stands for A, F, G.

3.1.6.1 Amplitude Difference Measure (ADM)

The point-by-point ADM is calculated considering filtered data the ‘DC’ and ‘Low’ from step 5 as given in the equation 4, so that it could be directly compared with the input data sets obtained from step 1 [5][6]. Here, the DC provides the offset between the input datasets while the Low gives the trend information [10]. This is abbreviated as ADM_i

$$AMD(n) = \left| \frac{\alpha}{\beta} \right| + \left| \frac{\chi}{\delta_M} \right| \exp \left\{ \left| \frac{\chi}{\delta_M} \right| \right\} \quad (4)$$

Where,

$$\alpha = (|Lo_1(n)| - |Lo_2(n)|)$$

$$\beta = \frac{1}{N} \sum_{i=1}^N (|Lo_1(i)| + |Lo_2(i)|)$$

$$\chi = (|DC_1(n)| - |DC_2(n)|)$$

$$\delta_M = \frac{1}{N} \sum_{i=1}^N (|DC_1(i) + Lo_1(i)| + |DC_2(i) + Lo_2(i)|)$$

Where,

$N \rightarrow$ Number of points in the inverse transformed, filtered, data sets / domain

$n \rightarrow$ is the nth data point.

$i \rightarrow$ Independent axis ranges from maximum to minimum

This step produces data for the ADM as shown graphically in Figure 12. This is then compared with the input data sets to identify areas of poor comparison, giving the user an opportunity to identify these areas and potentially investigate the cause for this. On the other hand, this output also but provides quantitative evidence of corrective measures or changes in the model to be able to investigate it themselves [6].

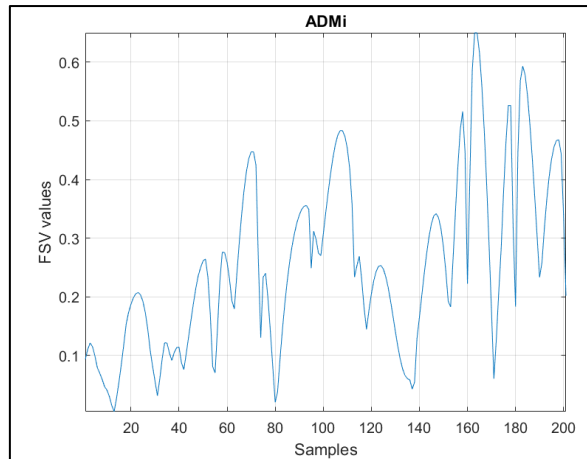


Figure 12: Point-by-point ADM

(a) ADM Total:

Here, the mean value of ADM which determines the quality of the result with single number is calculated using the equation 5.

$$ADM_{tot} = \frac{\sum_{n=1}^N ADM(n)}{N} \quad (5)$$

Where, N is the total number of samples used.

(b) ADM Histograms:

The confidence histograms were developed to aid the visual comparison technique [8]. In FSV, histograms quantify the data comparison into a number of natural language categories. Here, the value of FSV falls clearly into one of the various categories [9]. Currently, the range of xDM values are divided into six categories, described using natural language as Excellent, Very Good, Good, Fair, Poor, and Very Poor [6][7].

These qualitative descriptors were introduced to aid the comparison with visual evaluation directly [7][8]. Then the histograms are determined by counting the number of data point that fall into one of the quantitative categories as shown in the Table 3 and then divided by total number of samples in the data. This provides a presentation of the FSV data that is most easily compared with visual assessment of a group as discussed in [8]

FSV value (quantitative)	FSV interpretation (qualitative)
Less than 0.1	Excellent
Between 0.1 and 0.2	Very good
Between 0.2 and 0.4	Good
Between 0.4 and 0.8	Fair
Between 0.8 and 1.6	Poor
Greater than 1.6	Very poor

Table 3: FSV Interpretation scale [6]

Figure 13 represent the confidence histograms for ADM.

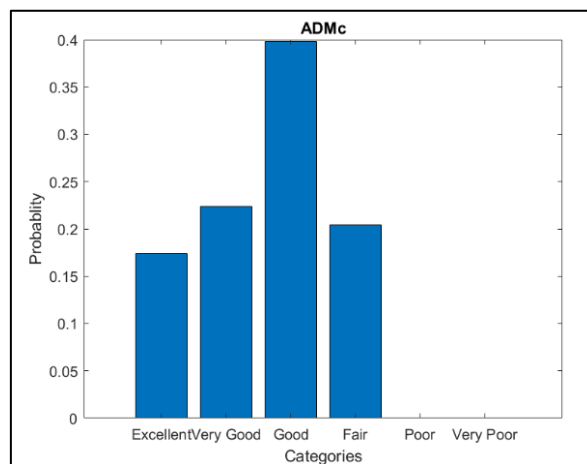


Figure 13: Histogram for ADM

The confidence histogram output for ADM indicates that a majority of data points fall into the "Good" bin category when considering the overall trend comparison. However, there are some notable data points that fall into the "Excellent", "Very good", or "Fair" categories. However, no data points are found in the "Poor" or "Very poor" bin categories.

3.1.6.2 Feature Difference Measure (FDM)

The FDM is calculated to determine the change in features in the input data sets. To calculate the point-by-point FDM, initially the first derivatives of low and high data sets are obtained using central difference approach as shown in the equation 6 [5][6].

$$Lo' = Lo(f + N_d) - Lo(f - N_d) \quad (6)$$

Where $N_d = 2$ for first derivative

Then the second derivatives of ‘High’ data set is calculated using the method shown in the equation 7 below,

$$Hi'' = Hi'(f + Na) - Hi'(f - Na) \quad (7)$$

Where $Na = 3$ for second derivative

Now, using the calculated derivatives from equation 6 and 7, the FDM is formed using three parts as shown in equation 8, 9 and 10. Here, the numerator calculation describes point-by-point values while the denominator describes calculations approaches using the mean value method as in ADM [6].

$$FDM_1(f) = \frac{|Lo_1'(f)| - |Lo_2'(f)|}{\frac{2}{N} \sum_{i=1}^N (|Lo_1'(i)| + |Lo_2'(i)|)} \quad (8)$$

$$FDM_2(f) = \frac{|Hi_1'(f)| - |Hi_2'(f)|}{\frac{6}{N} \sum_{i=1}^N (|Hi_1'(i)| + |Hi_2'(i)|)} \quad (9)$$

$$FDM_3(f) = \frac{|Hi_1''(f)| - |Hi_2''(f)|}{\frac{7.2}{N} \sum_{i=1}^N (|Hi_1''(i)| + |Hi_2''(i)|)} \quad (10)$$

Here, the values 2, 6 and 7.2 in the denominator represents the weighting scheme used to balance the FDM value with the visual evaluation [6]. Now, the equation 8, 9 and 10 are combined to form equation 11 as shown below. This in turn is used to calculate the FDM value for the input data sets [5][6].

$$FDM(n) = 2(|FDM_1(f) + FDM_2(f) + FDM_3(f)|) \quad (11)$$

Figure 14 represents the point-by-point FDM output for the input data sets obtained from step 1.

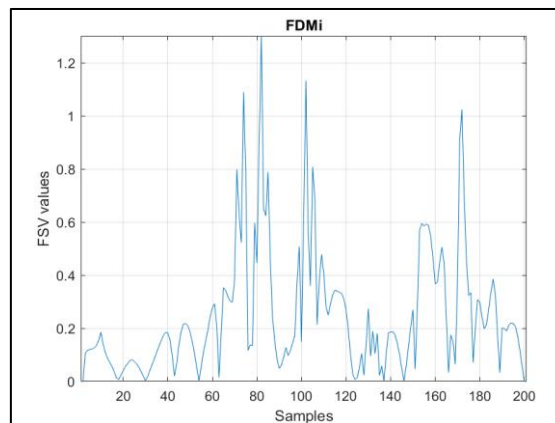


Figure 14: Point-by-point FDM

(a) FDM total:

The FDM also provides mean value, abbreviated as FDM_{tot}. This is calculated in the similar approach as of ADM_{tot} given in equation 5.

(b) FDM Histogram:

The histogram for point-by-point FDM is abbreviated as FDM_c which are calculated in the similar approach as of the ADM_c using the FSV interpretation given in table 3. For the input data sets used here, the FDM histogram is as shown in Figure 15 below.

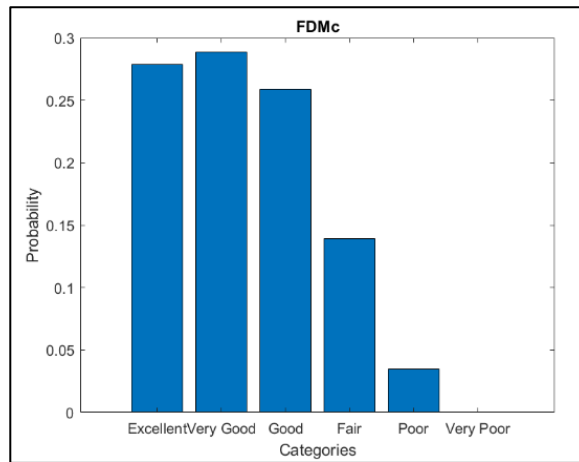


Figure 15: Histogram for FDM

3.1.6.3 Global Difference Measure (GDM)

GDM determines the overall comparison. The point-by-point GDM is calculated as in equation 12, using the independent values of ADM and FDM [4][6]

$$GDM(n) = \sqrt{ADM(n)^2 + FDM(n)^2} \tag{12}$$

Figure 16 represents the point-by-point GDM output for the input data sets.

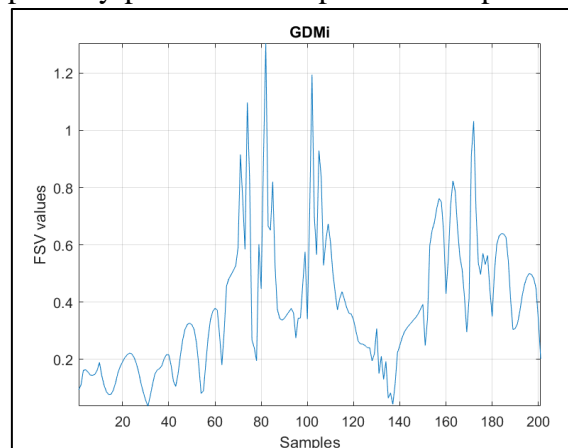


Figure 16: Point-by-point GDM

(a) GDM total

The GDM also provides mean value, abbreviated as GDM_{tot}. This is calculated in the similar approach as of ADM_{tot} given in equation 5.

(a) GDM Histogram

The histogram for point-by-point GDM is abbreviated as GDM_c which are calculated in the similar approach as of the ADM_c using the FSV interpretation given in table 1. For the input data sets used here, the GDM histogram is as shown in Figure 17 below.

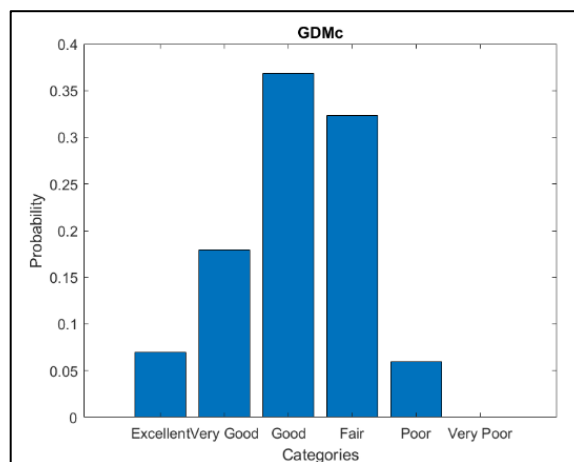


Figure 17: Histogram for GDM

3.2 MULTI DIMENSIONAL FSV

In this thesis, repeated 1-D FSV is used to perform 2-D FSV. The evolution of 1-D FSV into nD FSV was explained in the chapter 2 section 2.5, this section provides detailed steps to implement 2-D FSV using repeated 1-D FSV on the x and y direction of the datasets.

3.3 IMPLEMENTATION OF 2 DIMENSIONAL FSV

The basic procedure to implement 2-D FSV consists of five major steps, as shown in Figure 18 below. Each step is discussed in detail in the following sub-sections:

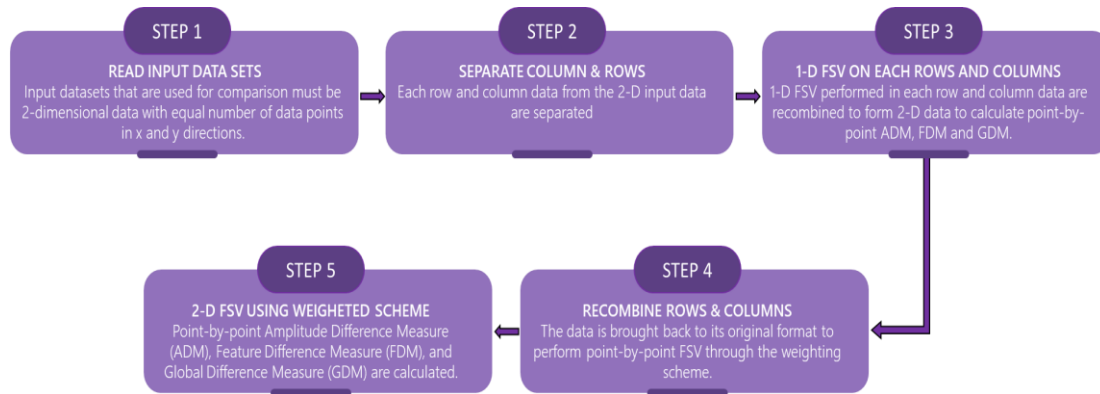


Figure 18: Steps to Implement 2- Dimensional FSV

3.3.1 STEP 1: Read Input data

The input data sets, that are used for comparison must be 2-dimensional data with equal number of data points in x and y direction as shown in Figure 19. Here, I1 and I2 are the 2-D input data sets with the same size of $M*N$, where M is the total number of column data while N is the total number of row data [11].

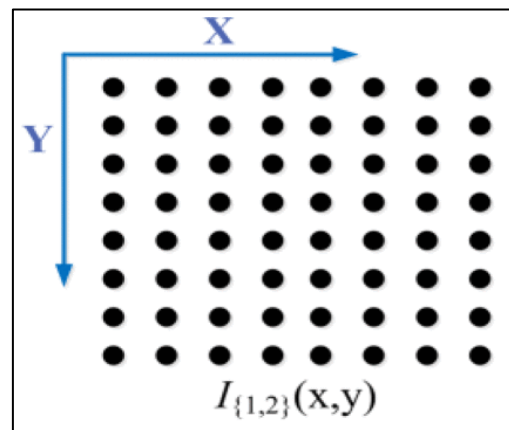


Figure 19: 2-dimensional Input data

The 2-D data must have same number of data points in both the x and y direction. In case of different number of data points, the overlap window is verified, and interpolation is performed

in the data for the overlap region to obtain equal number of data points [11] as described in section 3.1 above.

The overlap window requires a minimum of 32 data points in the filtered domain to identify the data in the Low and High regions. Therefore, when using the FSV method for data processing, it is essential that any input data contains a minimum of 32 data points. Interpolation is necessary if the length of the input data is less than 32 data points. This ensures that the data is expanded to the maximum number of data points required for accurate analysis and comparison.

3.3.2 STEP 2: To separate rows and column data

Now, each row and column data from the 2-D input data are separated as shown in the Figure 20. Here, each row and column data are given by $I_{\{H1,H2\}}$ and $I_{\{V1,V2\}}$ respectively. The ‘H’ represents data points in the horizontal direction while ‘V’ represents data points in vertical direction [11].

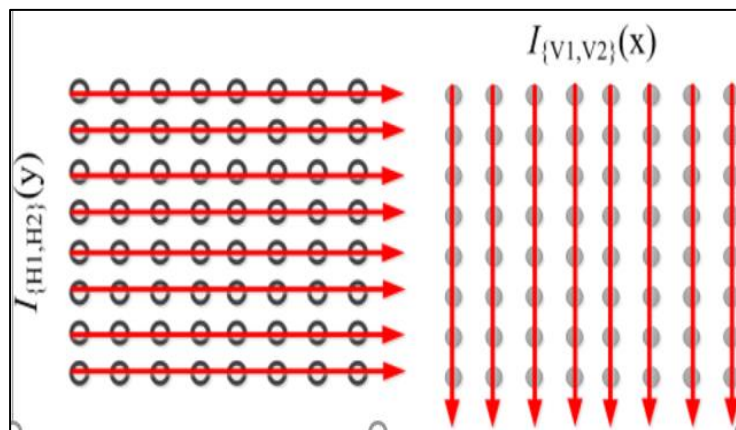


Figure 20: 2-D data separated vertical and horizontally

These data are then treated separately as two data sets as shown in the equation 13 and 14 below.

$$I_{\{H1, H2\}}(y); (y = 1, 2, 3, \dots, N) \quad (13)$$

$$I_{\{V1, V2\}}(x); (x = 1, 2, 3, \dots, M) \quad (14)$$

3.3.3 STEP 3: Perform 1-D FSV on each row and column separately

In this step, 1-D FSV is performed as explained in section above, to calculate point-by-point ADM, FDM and GDM [11]. Similarly, the mean value (xDMtot) and histograms (xDMc) are

also calculated using the same approach given in section 3.1. Equation 15 and 16 represents the repeated 1-D FSV performed on x and y directions.

$$xDMi_V(x) = FSV(IV1(x), IV2(x)), \quad x=1,2,3,\dots,M$$

$$xDMi_H(y) = FSV(IH1(y), IH2(y)) \quad y=1,2,3,\dots,N$$

3.3.4 STEP 4: Recombine the rows and column to bring it back to 2-D

Here, the 1-D FSV performed in each row and column data are combined to form 2-D data. This provides ADMi, FDMi and GDMi output into 2-D format. Therefore, the data to be compared is brought back to its original format to perform point-by-point FSV through the weighting scheme. This is further explained in step 5.

3.3.5 STEP 5: Perform 2-D FSV using weighted Scheme

The 1-D FSV performed on each row and column data separately and combined are treated on point-by-point basis through weighting scheme. Equation 17 and 18 represents this approach [11].

$$ADMi(x, y) = \sqrt{K_V ADMi_V(x, y)^2 + K_H ADMi_H(x, y)^2} \quad (17)$$

$$FDMi(x, y) = \sqrt{K_V FDMi_V(x, y)^2 + K_H FDMi_H(x, y)^2} \quad (18)$$

Here, $ADMi_V$ and $ADMi_H$ represents the 1-D FSV outputs of column and row data of the combined 2-D data respectively. Similarly, $FDMi_V$ and $FDMi_H$ represents the same. The K_V and K_H are the weighting factors used to perform 2-D FSV, which ranges from 0 to 1 [11][12]. This weighting factor is calculated using the approach shown in equation 19 and 20.

$$K_V = \frac{M}{M+N} \quad (19)$$

$$K_H = \frac{N}{M+N} \quad (20)$$

Where M and N are the number of data points in column and row data respectively.

Now, the point-by-point GDM is calculated by combining the ADMi (x,y) and FDMi(x,y) as shown in the equation 21, without including any weighting factor to it[11].

$$GDMi(x, y) = \sqrt{ADMi(x, y)^2 + FDMi(x, y)^2} \quad (21)$$

Moreover, the Mean value xDMtot and Histograms are obtained using the same approach as in 1-D FSV, explained in section 3.1

Further the 2-D FSV can be extended to 3-dimensional and n-dimensional data. Figure 21 below represent a 3-D data. Here, the repeated 1-D FSV is performed for x, y and z directions [12]. Similar implementation steps as explained in section 2 (2-D FSV) could be used to perform nD FSV.

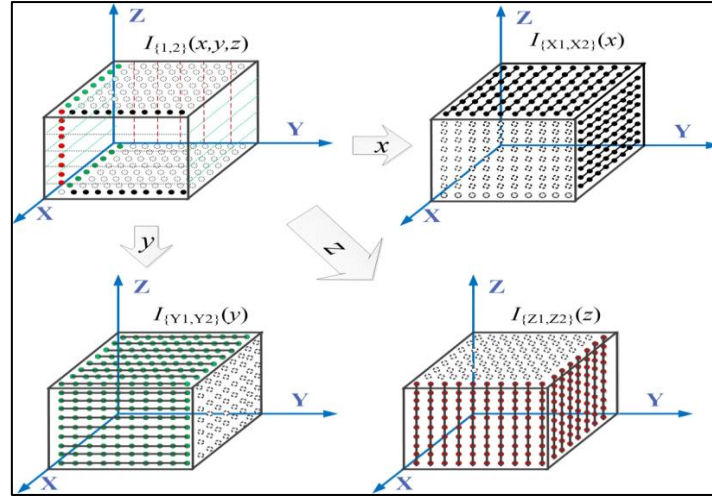


Figure 21: n-D FSV method on a 3D data [12]

Now, the point-by-point FSV values from x, y and z directions are recombined to bring it back to the original input data format [12]. Then using weighting scheme, point-by-point 3-D FSV is obtained using the approach given in equation 22 and 23.

$$ADMi(x, y, z) = \sqrt{K_X \cdot ADMi_X(x, y, z)^2 + K_Y \cdot ADMi_Y(x, y, z)^2 + K_Z \cdot ADMi_Z(x, y, z)^2} \quad (22)$$

$$FDMi(x, y, z) = \sqrt{K_X \cdot FDMi_X(x, y, z)^2 + K_Y \cdot FDMi_Y(x, y, z)^2 + K_Z \cdot FDMi_Z(x, y, z)^2} \quad (23)$$

The K is the weight factor used which ranges from 0 to 1 [11][12]. The calculation of K in x, y and z direction are obtained using the equation 24, 25 and 26 respectively [9].

$$K_X = \frac{L}{L+M+N} \quad (24)$$

$$K_Y = \frac{M}{L+M+N} \quad (25)$$

$$K_Z = \frac{N}{L+M+N} \quad (26)$$

Where L , M and N represents the number of data points along x , y and z direction respectively [12]. Finally, the overall comparison combining ADMi and FDMi is given by GDMi as shown in the equation 27.

$$GDMi(x, y, z) = \sqrt{ADMi(x, y, z)^2 + FDMi(x, y, z)^2} \quad (27)$$

The total value and the histograms are obtained in the similar method used in the 1-D FSV, section 3.1.

3.4 PROBABILITY DENSITY FUNCTIONS (STATISTICAL ANALYSIS OF FSV)

As discussed in chapter 2, section 2.6, to overcome the issues with histograms in the FSV method, continuous Probability Density Function (PDFs) is derived from the point-by-point FSV [13]. This further allows statistical analysis of the FSV results. On the other hand, as the PDF's provide continuous distribution, it allows multiple comparisons to be analysed using non-parametric test [13] [14]. The calculations of PDF's, statistical analysis on its results and non -parametric tests are given in the below sub-sections.

3.4.1 PDF CALCULATIONS

To obtain detailed results of the datasets being compared, the PDFs are calculated for the point-by-point ADM, FDM and GDM.

However, before, calculating the continuous PDF's; it is important to pre-process the point-by-point FSV obtained using the piece-wise linear interpretation given in table 4. This is because, the FSV interpretation values given in table 3 provides a non-linear relationship between the quantitative and qualitative FSV results given [14]. Therefore, the FSV interpretation values are linearized in table 4 to directly compare the calculated PDF's to the confidence histograms [13].

FSV value (point-by-point)	Linearized Value
$X \leq 0.1$	X
$0.1 < X \leq 0.2$	X
$0.2 < X \leq 0.4$	$0.2 + (X - 0.2)/2$
$0.4 < X \leq 0.8$	$0.3 + (X - 0.4)/4$
$0.8 < X \leq 1.6$	$0.4 + (X - 0.8)/8$
$1.6 < X \leq 3.2$	$0.5 + (X - 1.6)/16$
$X > 3.2$	0.6

Table 4: Piecewise linear FSV interpretation [13]

The probability density function $f(x)$ given in equation 28 is estimated using the normal kernel function [13] [15]. The kernel function K is used here, as the kernel bandwidth h aids in providing the smoothness for the probability density curve [13].

$$f(x) = \frac{d}{dx} F(x) = \lim_{h \rightarrow 0} \frac{F(x+h) - F(x-h)}{2h} \quad (28)$$

Where,

$x \rightarrow$ Random variable

$F(x) \rightarrow$ Cumulative distribution function of the random variable

$h \rightarrow$ Bandwidth

Now the Empirical Cumulative Distribution Function (ECDF) for a random sample with size n from the density f , $X: \{x_1, x_2, \dots, x_n\}$ has the form given in equation 29 [13][14].

$$\hat{F}(x) = \frac{N\{X \leq x\}}{n} \quad (29)$$

Here, $N\{X \leq x\}$ signifies the number of elements less than or equal to x in X [13] [14].

Therefore, the equation 28 is obtained as shown in 30 below.

$$\hat{f}(x) = \frac{N\{x-h < X \leq x+h\}}{2nh} \quad (30)$$

The equation 31 can be re-written, including the Kernel density estimator with uniform Kernel function, K as given in equation 31 [13] [14].

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right) \quad (31)$$

Where,

$$K(u) = \begin{cases} \frac{1}{2}, & -1 < u < 1, \\ 0, & \text{otherwise.} \end{cases}$$

3.4.2 STATISTICAL ANALYSIS

As the PDF's are obtained, it makes it possible to analyse FSV performance using statistical moments. The results help in describing the characteristics of the distribution [14].

The first four statistical moments, namely mean, variance, skewness and kurtosis respectively are calculated using the point-by-point FSV values as given in equation 32,33, and 34.

Firstly, the mean is calculated as shown in equation 5 which provides the average value of the distribution and represented as μ . Secondly, variance represented as σ determines the variability measure of the datasets. Thirdly, skewness is a measure of symmetry, which determines how much the distribution varies with respect to the normal distribution. Fourthly, the kurtosis determines the tails, peaks of a distribution [14].

$$\text{Variance} = \sigma^2 = \frac{1}{N} \sum_{i=1}^N (xDM(i) - \mu)^2 \quad (32)$$

$$\text{Skewness} = \frac{1}{N} \sum_{i=1}^N \frac{(xDM(i) - \mu)^3}{\sigma^3} \quad (33)$$

$$\text{Kurtosis} = \frac{1}{N} \sum_{i=1}^N \frac{(xDM(i) - \mu)^4}{\sigma^4} \quad (34)$$

Where, $xDM(i)$ refers to point-by-point ADM, FDM and GDM. The total number of points in $xDM(i)$ is given by N .

3.4.3 THE KOLMOGOROV-SMIRNOV (KS) TEST:

The KS-test is a non-parametric test which determines the goodness-of-fit by calculating the maximum difference between the cumulative distribution functions of both datasets [16]. The results test the null hypothesis which determines if the input datasets are from same or different distribution. If the Null hypothesis is true, then the input datasets can be confidently regarded as being from the same distribution. While the null hypothesis is rejected or false, then the input datasets are from the different distributions. However, the null hypothesis is obtained by obtaining the test statistic value D and then compared with the critical value decided by the significance level known as D_{critical} [14][16].

The test statistic D is determined using the approach given in equation 35. This is calculated using the maximum vertical deviation between the two curves of the Empirical Cumulative Distribution functions (ECDFs) of the datasets [14].

$$D = \max (|CDF_1(x) - CDF_2(x)|) \quad (35)$$

Here, CDF_1 and CDF_2 represents the number of values less than or equal to x in the first and second datasets respectively [11]. Now, the $D_{critical}$ value for the test statistic D for various significance level is decided using the approach given in equation 36 [14][17].

$$D_{critical} = k \cdot \sqrt{\frac{(N_1 + N_2)}{(N_1 \cdot N_2)}} \quad (36)$$

The length of the datasets to be compared are N_1 and N_2 respectively. The K value is determined from the tables given in [14]. If the test static D value is greater than the $D_{critical}$ value, the null hypothesis is false/rejected. On the other hand, if D value is less than the $D_{critical}$ value, the null hypothesis is true/accepted [14] [17].

The major advantage of KS-test is that it is a robust test and hence the results are not affected by the linearizing procedure sued to obtain the PDF's as explained in section [14]. The KS-test process can be used to perform one-sample test, where the CDF is compared with theoretical distribution. On the other hand, the KS-test can also be performed on two-samples that provide the same distribution. Here, the CDF's from two the datasets are compared [18]. This is further explained is chapter 4, as two-sample KS-test approach is used in this research.

3.5 REFERENCES

- [1] B. Archambeault, S. Connor and A. Duffy, "Comparing FSV and human responses to data comparisons", *2005 International Symposium on Electromagnetic Compatibility, 2005. EMC 2005.*, 2005.
- [2] 2022. [Online]. Available: <https://uk.mathworks.com/help/matlab/getting-started-with-matlab.html>. [Accessed: 12- Apr- 2022]
- [3] T. Siemers, *An Introduction to Matlab and Mathcad*. 2014.
- [4] A. Duffy, A. Martin, A. Orlandi, G. Antonini, T. Benson and M. Woolfson, "Feature Selective Validation (FSV) for Validation of Computational Electromagnetics (CEM). Part I—The FSV Method", *IEEE Transactions on Electromagnetic Compatibility*, vol. 48, no. 3, pp. 449-459, 2006.
- [5] "IEEE Standard for Validation of Computational Electromagnetics Computer Modeling and Simulations".

- [6] "IEEE Recommended Practice for Validation of Computational Electromagnetics Computer Modeling and Simulations", 2011.
- [7] D. Di Febo, F. de Paulis, A. Duffy and A. Orlandi, "Histogram density for the Feature Selective Validation (FSV) method", 9th IET International Conference on Computation in Electromagnetics (CEM 2014), 2014.
- [8] D. Di Febo, F. de Paulis, A. Orlandi, G. Zhang, H. Sasse, A. Duffy, L. Wang and B. Archambeault, "Investigating Confidence Histograms and Classification in FSV: Part I. Fuzzy FSV", IEEE Transactions on Electromagnetic Compatibility, vol. 55, no. 5, pp. 917-924, 2013.
- [9] G. Zhang, H. Sasse, A. Duffy, L. Wang, D. Di Febo, F. de Paulis, A. Orlandi and B. Archambeault, "Investigating Confidence Histograms and Classification in FSV: Part II-Float FSV", IEEE Transactions on Electromagnetic Compatibility, vol. 55, no. 5, pp. 925-932, 2013.
- [10] B. Archambeault, A. Duffy, H. Sasse, X. Li, M. Scase, M. Shafiullah, A. Orlandi and D. di Febo, "Challenges in developing a multidimensional Feature Selective Validation implementation", 2010 IEEE International Symposium on Electromagnetic Compatibility, 2010.
- [11] G. Zhang, A. Duffy, A. Orlandi, D. Febo, L. Wang and H. Sasse, "Comparison of Data With Multiple Degrees of Freedom Utilizing the Feature Selective Validation Method", IEEE Transactions on Electromagnetic Compatibility, vol. 58, no. 3, pp. 784-791, 2016.
- [12] G. Zhang, A. Orlandi, A. Duffy and L. Wang, "Comparison of Three-Dimensional Datasets by Using the Generalized n-Dimensional (n-D) Feature Selective Validation (FSV) Technique", IEEE Transactions on Electromagnetic Compatibility, vol. 59, no. 1, pp. 103-110, 2017.
- [13] Z. Gang, A. Duffy, H. Sasse and W. Lixin, "The use of probability density functions to improve the interpretation of FSV results", *2012 IEEE International Symposium on Electromagnetic Compatibility*, 2012.
- [14] G. Zhang, H. Sasse, L. Wang, and A. Duffy, "A Statistical Assessment of the Performance of FSV", *ACES Journal*, vol. 28, no. 12, pp. 1179–1186, Sep. 2021#.
- [15] J. S. Simonoff Smoothing methods in statistics. New York: Springer Verlag 1996.
- [16] F. Massey, "The Kolmogorov-Smirnov Test for Goodness of Fit", *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68-78, 1951.
- [17] N. Smirnov, "Table for Estimating the Goodness of Fit of Empirical Distributions", 2022. .
- [18] V. Berger and Y. Zhou, "Kolmogorov–Smirnov Test: Overview", *Wiley StatsRef: Statistics Reference Online*, 2014.

CHAPTER - 4

PROPOSED METHOD

In this Chapter, a proposed method is demonstrated on regularly shaped (rectangular and square) images to allow a detailed analysis. However, this method can be readily extended to irregular structures which include features such as spaces or gaps within the image structure. Moreover, this chapter also reviews the challenges and advantages of the developed approaches.

The proposed methodology is developed into two approaches to perform 2-Dimensional FSV, as shown in Figure 1 below. The first approach (A1) performs 2-D FSV on the original full structure of the input image. Whereas, in the Approach (A2), the original image is segmented into multiple blocks, and 2-D FSV is performed on each blocks and then concatenated to the original structure for results comparison. The outputs from Approach 1 and Approach 2 are discussed in Chapter 5.

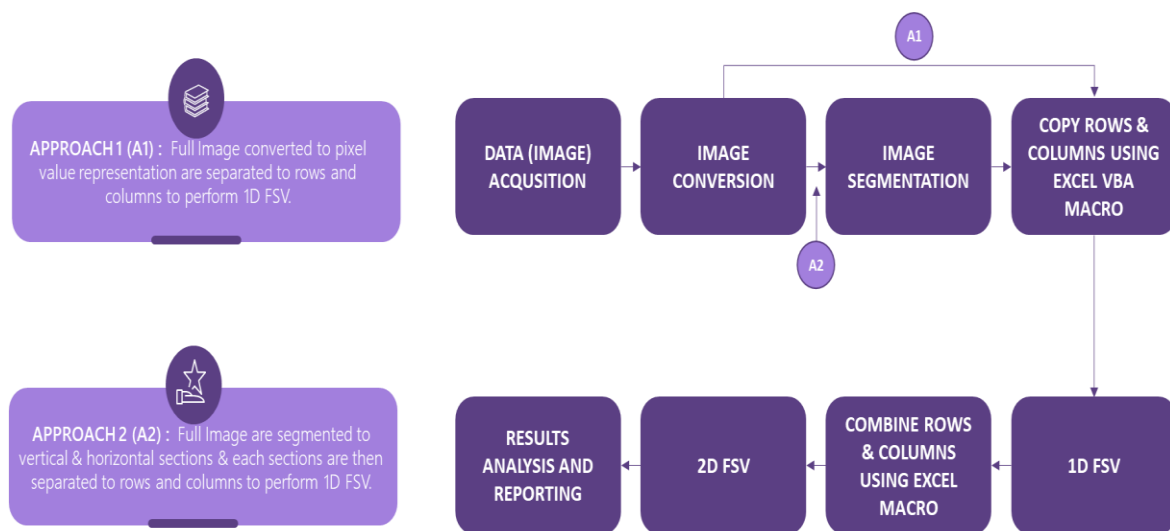


Figure 1: Illustration of the proposed methodology

Figures 2 and 3 provide visual illustrations of the step-by-step process in Approach 1 (A1) and Approach 2 (A2), respectively. The subsequent sections will elaborate on each of the steps depicted in Figures 2 and 3.

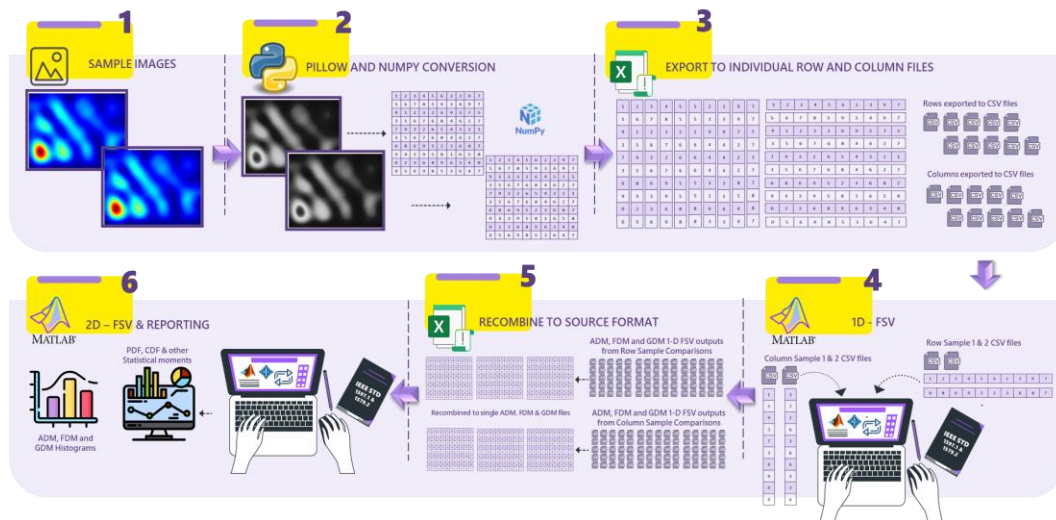


Figure 2: Illustration of steps involved in Approach 1

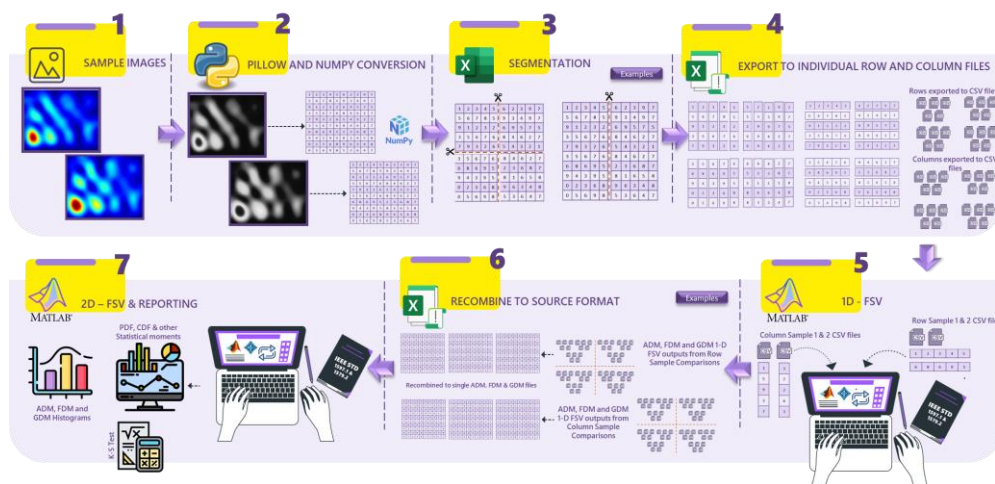


Figure 3: Illustration of steps involved in Approach 2

This proposed methodology is developed using various programming languages and toolsets to demonstrate the feasibility of the research study. Here, the Python programming language is used for image processing and conversion into an array, as this provides a range of image manipulation and storage functions using the Python Imaging Library (PIL) and Numerical Python (NumPy). In this section, the JetBrains PyCharm Integrated Development Environment (IDE) version 2020.3 and Python 3.8.6 are used to develop and test the Python program responsible for image conversion.

Microsoft Excel is employed for visually representing and interpreting data points.

Additionally, a Microsoft Excel Visual Basic for Applications (VBA) Macro is employed to

automate specific steps involved in the 2-dimensional Feature Selective Validation (2-D FSV) process.

Finally, MATLAB programming is used for performing FSV steps and generating graphs and other outputs for final comparisons. The MATLAB programming developed for the FSV steps and results has been successfully tested to be compatible with the MATLAB R2020a version and on the Windows 10 operating system.

4.1 DATA PREPARATION

The symbolic representation of information is known as data. These data are collected for various purposes such as in machine learning, computation electromagnetics and for simple data mining. However, before collecting the data, it is important to understand the experimental purpose, which specifies how the data could be presented for its analysis. Hence, it is a major step to organize the data into an appropriate form. Therefore, the process of pre-processing, organizing and manipulating data before analysis is known as data preparation [1][2].

In this thesis, a raster image and a vector image are utilised as input data for the proposed methodology. These image types are extensively discussed in Chapter 2, Section 2.1.5.

Here, an image of “Einstein” and an “E-field output of an EMC structure” through a slice of a mode stirred reverberation chamber at different stirrer positions are utilised as examples of raster and vector images, respectively. Figures 4 and 5 represent a 2-dimensional image of Einstein and E-field outputs of an EMC structure used in this research study to verify the feasibility of the proposed methodology.

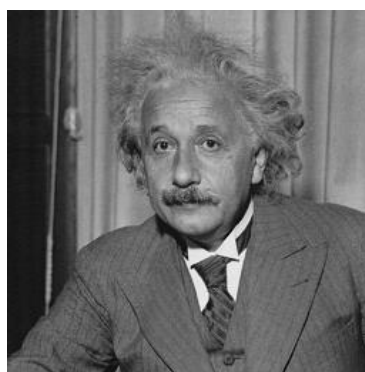


Figure 4: Image of Einstein [3]

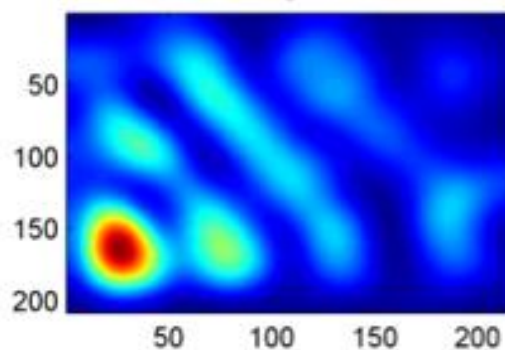


Figure 5: E-Field output of an EMC structure [3]

The selection of the specific image to be analysed can greatly affect the outcomes of the FSV process. Hence, choosing an image is often determined based on the need for the given or a proposed analysis purpose. For this reason, data preparation is a critical part of the research methodology [1].

Thus, it is important to analyse the anomalies and augment to prepare specific set of input data. When an input image is selected, it is important to verify the quality of the image which is known as a data anomaly check. This includes, duplications, missing values, inconsistency and other distortion that could cause imperfections in the specified images [4][5]. It is a common criterion to check for data anomaly to make sure the correct input image is selected.

The other important criteria are the data augmentation, which is commonly used method to enhance the images to provide specific number of data sets [6]. The general augmentation used are rotation, translation, scaling, adding noise, blurring an image, etc [6][7].

In this thesis, the data used comes from 2-Dimensional images which is to be processed as explained in the upcoming sections below.

4.2 IMAGE CONVERSION

An image may be transformed in order to meet the unique criteria of an experiment. It is important to understand how the image may be presented for analysis [1]. This is also known as pre-processing the original input data into another form for the experimental purposes. In this thesis, RGB and greyscale images are used as the input data as shown in figure 2 and 3. These images are then converted to an array of pixel values, which holds the dimensions and colour information of the input images [8].

As discussed in Chapter 2, Section 2.1.4, greyscale images only contain brightness information and are represented by pixel values ranging from 0 to 255, where 0 represents black and 255

represents white [9] [10]. In contrast, RGB images are colour images composed of three colour intensities: Red, Green, and Blue. Each colour intensity is represented by 8 bits per pixel, resulting in a total of 24 bits per pixel and a range of 0-255 for each colour [10] [11]. Figure 6 represents the Red, Green, and Blue channels of an “E-Field” image, as shown in Figure 5.

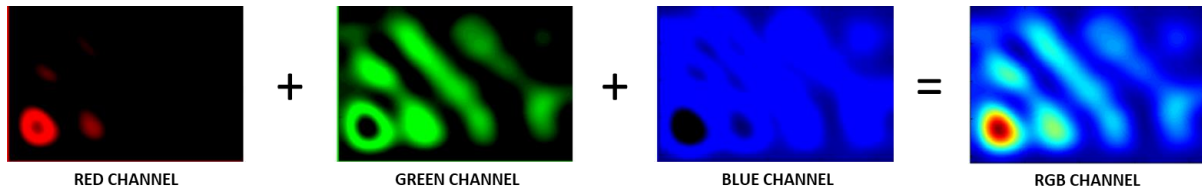


Figure 6: RGB Channels of an E-field image

This section discusses about two image conversions methods used for experimental purpose. Firstly, to convert an RGB image into a greyscale image and then convert it into an array of pixel values. Secondly, a greyscale image is directly converted into an array of pixel values for further processing which are explained in the upcoming sections. However, it is important to understand why an RGB image requires greyscale conversion here.

Initially when a 2-dimendional RGB image was converted into an array of pixel values; each pixel value of the array contained consecutive values of red, green and blue. Therefore, FSV had to be performed for each colour pixel array separately and then every colour pixel array had to be recombined using the weighted approach to perform 2-D FSV.

For example, when the E-Field image in RGB colour mode was converted into array of RGB pixel values using python which is explained further in this chapter, each pixel holds the value of red, green and blue as shown in Figure 7. Thus, the total number of data points are three times higher than the total pixel count. Hence to perform FSV, the red, green and blue values must split into 3 separate image arrays and then recombined to analyse the final results.

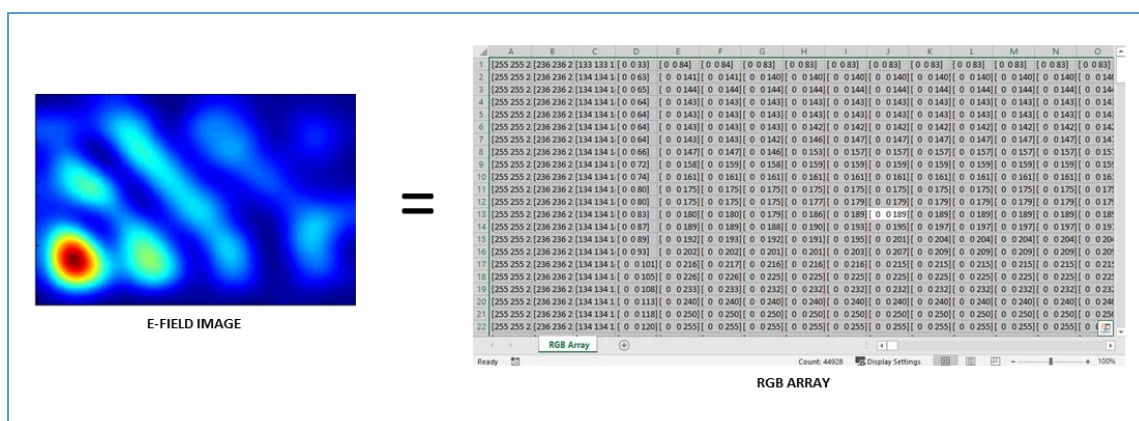


Figure 7: E-Field Image converted in to RGB array

This increases computation time, effort, and risk of introducing errors during repeating FSV steps for each colour array and reforming using weighted scheme. Therefore, RGB images are converted to grey scale images before performing FSV for experimental purposes in order to avoid computing overheads and to demonstrate the proposed methodology.

This is a complex solution which is further discussed in the future work. However, to reduce the complexity and to provide the proof of proposed concept, RGB images are converted to grey scale for experimental purpose.

4.2.1 PYTHON FOR IMAGE CONVERSION

An image contains pixel values or numbers that can be read and interpreted by computer programming languages [12] [13]. Here, Python is one such programming language which can be used to convert an image into pixel value, read and write these numbers in an input data format which is the fundamental step of this thesis. Python is an interpreted high-level, functional programming language in scientific computing platform with an object-oriented approach that aids in the straightforward construction of logical programmes, using its own standard libraries and garbage collection [14] [15].

In the current software marketplace, various Python Integrated Development Environment (IDEs) are available for usage. The Integrated Development Environment or IDE is a software program that comprises the code editor, compiler/interpreter, and debugger into a single package. IDEs ease the process of developing new programs without having to set up several packages or learn new tools to run a programme [16]. JetBrains PyCharm IDE version 2020.3 and Python 3.8.6 is used in this section to develop and test the image conversion python program.

In this section, the RGB to greyscale conversion and greyscale to array of pixel values are obtained using the Python program developed as shown in Appendix 2. The input image conversion is performed step-by-step as shown in the Figure 8. These steps are further explained in the upcoming sub sections below.

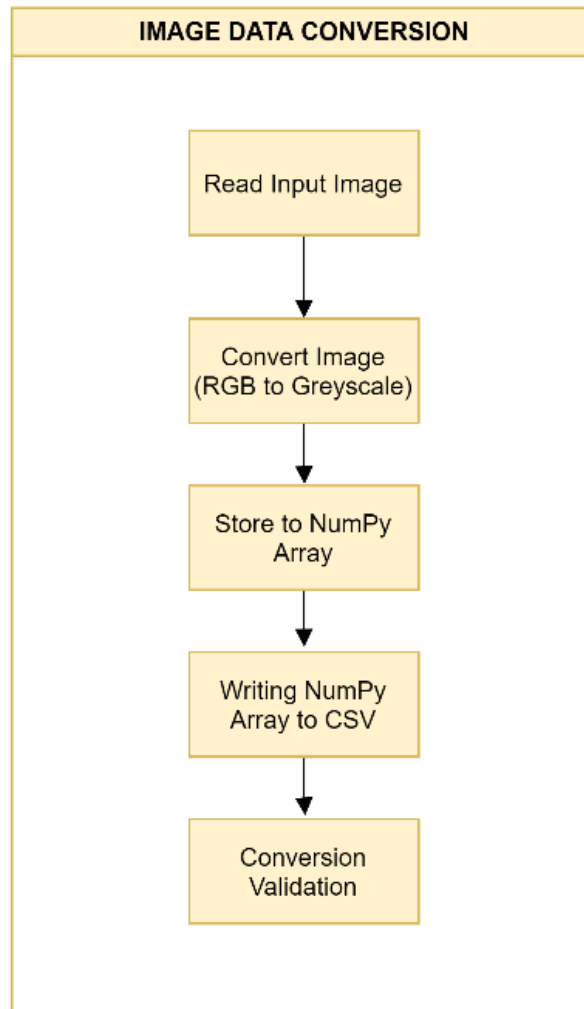


Figure 8: Steps for Image conversion using Python

Python offers a library for image processing operations called the Python Imaging Library (PIL). The PIL provides a range of image manipulation functions which can be written using Python programming language. Moreover, PIL supports file formats such as .png, .jpeg, .gif, .tiff, and .bmp [17]. In order to import and work with PIL, Pillow fork should be installed to the programming workplace using Python package installer (pip) as follow [17],

\$sudo pip install pillow

The PIL also offers basic functions such as open (), save (), show (), convert (), copy (), resize (), etc [17]. Some of these PIL syntaxes are used to perform image conversion for this research. PIL is initialised in a python program as shown below [17],

import PIL

from PIL import Image

4.2.1.1 READ INPUT IMAGE

Here, the input images are read using *open ()* function for the given image by file name in a specified file path. Following is the syntax for open an image [17][18].

image.open(filename)

Below is an example python statement to open an E-filed image as shown in Figure 9. Either a filename or the path of the file can be mentioned in this the *open ()* function. Image must be stored in a variable defined in the program for further image operations.

inputImage = Image.open(E-field.png)

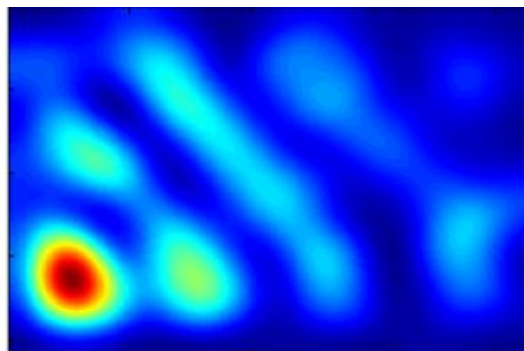


Figure 9: E-field image in RGB colour mode

4.2.1.2 CONVERT RGB IMAGE TO GREYSCALE IMAGE

As discussed in section 4.2, RGB and Greyscale images are converted into array of pixel values for experimental purposes. However, an RGB image must be converted to a greyscale image before storing it into an array of pixel values.

This step describes how the Python Imaging Library (PIL) Image module is utilised to obtain this outcome [17].

The PIL image module's '*Convert ()*' function is used to convert an RGB colour image to a greyscale colour image. The syntax for converting the image to a different mode is shown below [18],

image.convert(mode)

“*Mode*” in the above syntax is categorised into, ‘*L*’ and ‘*RGB*’. The letter ‘*L*’ represents a greyscale image conversion, whereas the letter ‘*RGB*’ represents a colour image conversion [18]. An example Python statement for converting an RGB image to a greyscale image is given below and Figure 10 represents the converted image into grayscale image.

```
inputImage = Image.open(E-field.png)
greyScaleInputImage = inputImage.convert('L')
```



Figure 10: E-field image converted into a greyscale image

When converting the greyscale image shown in Figure 10 to image intensity pixel values, it is important to note that the same value can be assigned to both a very high field strength and a very low field strength if they have the same colour intensity. This can potentially create confusion and misleading comparisons, especially when interpreting the data without access to the original information that generated the image.

This phenomenon is a well-known problem in image intensity representation, as different data distributions can lead to the same pixel value in the greyscale image. However, it is crucial to highlight that in the context of this specific case, the purpose is to present the concept. The intention is to introduce the idea and demonstrate its potential applications.

While this limitation is acknowledged and understood, it does not affect the validity of the approach being presented. Therefore, in practical implementations, the proposed approach can be implemented effectively.

4.2.1.3 STORE IMAGE IN TO NUMPY ARRAY

In this step the greyscale images are stored into an array of pixel values. An array is a grid of numerical values which contains information about the pixel values, how to locate and interpret a pixel value [20]. Numerical Python (NumPy) arrays is a standard data structure for fast storing and reading multidimensional arrays [19] [20] in Python and can be used to perform a wide variety of mathematical operations on arrays. NumPy array is used in this step to store the pixel values as it provides various image processing functions [21].

Similar to Pillow fork installation as discussed in section 4.2.1., NumPy can be installed to the programming workspace using python package installer (pip) as shown below [22] [23].

```
$sudo pip install numpy
```

Now NumPy package can be imported using the below syntax,

```
import numpy as np
```

Following the example in section 4.2.1.2, after an RGB image is converted to a greyscale image, the image will be stored to a NumPy array as shown in the below statements. JetBrains PyCharm IDE version 2020.3 and Python 3.8.6 on a Windows 10 operating system is used in this section to store the converted greyscale image into NumPy array.

```
# converting RGB to Greyscale image  
inputImage = Image.open(E-field.png)  
greyScaleInputImage = inputImage.convert('L')  
  
# storing to a NumPy Array  
  
inputImageNumPyArray = np.array(greyScaleInputImage)
```

4.2.1.4 WRITING NUMPY ARRAY TO CSV

As, the images are converted into NumPy array, it is important to store this array to perform further operations explained in the upcoming sections. Here, these NumPy array are stored into a Comma separated Value (CSV) files. These files are used as input files for further FSV process discussed in this chapter.

In any computer programming, storing a data is an important step as it allows to save information for further processing; however, when transferring data from one software to another, data saved in files will ease the programming complexities. Python provides extensive functionality for interacting with files of various formats. For this research the NumPy arrays are stored in CSV formats with .CSV extension [24] [25].

The CSV is a general import and export format used in spreadsheets and databases. Moreover, it is simple to read and perform operations. Here, Python CSV module allows to store the NumPy array from the section 4.2.1.3 into a tabular data format which can be used in Microsoft Excel for experimental purposes [26]. Figure 11 represents the NumPy array stored in a CSV file which is opened using Microsoft Excel.

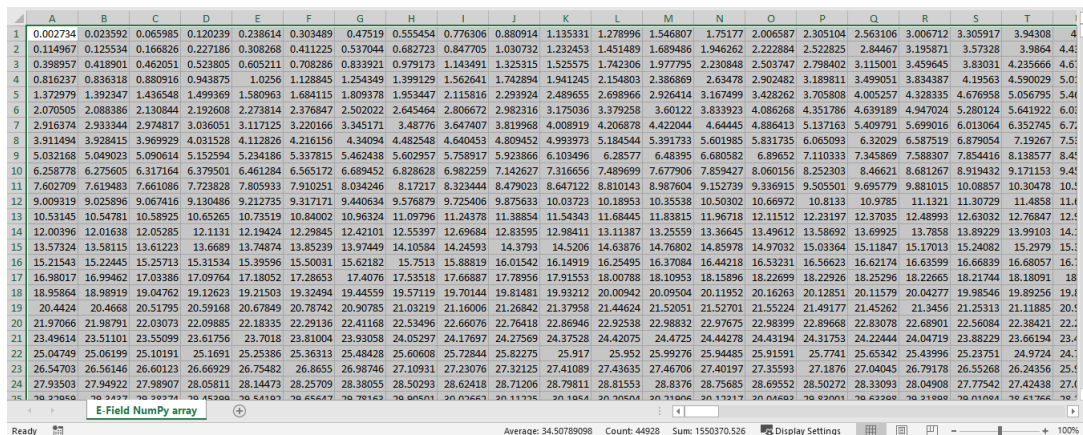


Figure 11: A illustrative example of a Greyscale image converted into NumPy array and opened in Microsoft Excel.

Similar to other functions such as PIL and NumPy, the 'csv' module performs a number of functions, including read and write. A *csv.reader()* function is used to read each lines of a CSV file by rows or columns; while, a *csv.writer()* function is used to write lines of data points to a CSV file by rows or columns. In order to perform the above, a 'csv' package is imported using the below syntax [26].

```
import csv
```

Following the example in section 4.2.1.3, after storing the pixel values in a NumPy array, the array is written to a *csv* file to a user-specified file location, as indicated in the sample python statements shown below.

```
# storing to a NumPy Array

inputImageNumPyArray = np.array(greyscaleInputImage)

# save the image Greyscale numpy array as a CSV to a user selected file path

ftypes = [('CSV files', '*.csv')]

with open(asksaveasfilename(filetypes=ftypes, initialfile = "E-field_GreyScaleArray"
, title = "Select the path to save the outputs", defaulttextextension=".csv"), 'w', newline=")
as csvfile:

writer = csv.writer(csvfile, delimiter=',')
writer.writerows(inputImageNumPyArray)
```

4.2.1.5 VERIFYING CONVERSION METHOD

This is one of the most important steps in Image conversion approach. Here, each step followed to perform image conversion are verified. This is done to identify and correct procedural errors

at the early stage of the experiment which in turn aids in saving time and effort spent on repetitive data processing tasks. This step is not a comprehensive model for verifying the final system; nonetheless, it will verify that the images converted, stored, and written are the same as when they were inputted [27].

The conversion is verified using the PIL and NumPy libraries. Firstly, the image mode is verified to ensure the image is converted to greyscale [16]. Secondly, the total number of pixel values are checked to ensure that the image pixel count and the NumPy array count are the same and thus no data is lost [22]. Finally reforming the NumPy array back to the original Image to verify whether the image is rightly converted [28]. This aids in visually determining if the converted image is reconstructed from the NumPy array. The Python statements used to perform the verification process is as given below,

```
print (inputImage.mode)  
  
width, height = inputImage.size  
  
print ("Total Pixels:", width*height)  
  
print ("Total Pixels:", np.size(inputImageNumpyArray))  
  
im = Image.fromarray(inputImageNumpyArray)  
  
im.show()
```

4.3 IMAGE SEGMENTATION

Image segmentation is a technique for breaking down a digital image into multiple segments in order to minimise the image's complexity and make it simpler to represent and classify it meaningfully to analyse an image effectively. The segments can be collectively regrouped to form a full original image [29].

In the beginning stage of this research, images as shown in Figure 12 below were considered as input images as the aim of this research is to compare data with irregular structure. Figure 12 represents surface currents outputs from a Computer Simulation Technology (CST) for an Ultra High Frequency (UHF) Radio Frequency Identification (RFID) tag, designed to be mounted directly on the human skin [30].

Unlike, the other properties of an antenna like S-parameters, radiation, efficiency and gain; the surface current distribution outputs are represented on the antenna structure and thus the FSFV

can be applied on the device structure itself. This could provide an accurate measure of the device. Moreover, the EMC devices were studied in UHF range as shown in the reference [32][33], which could lead on this research in current research areas.

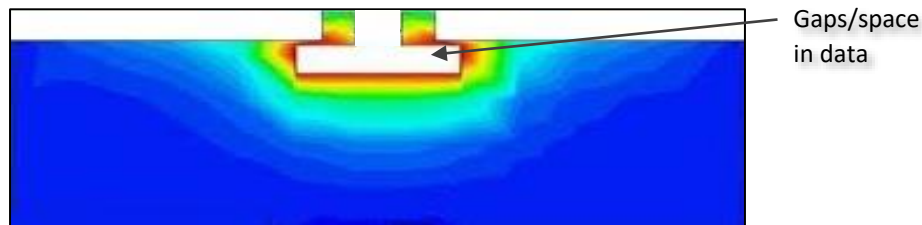


Figure 12: Simulated surface current distribution of On-skin passive UHF RFID tag [31]

As seen in Figure 12, here are a gap or space in the EMC device structure. These images are computer-generated; which are vector images as discussed in Chapter 2, Section 2.1.5.1 [33]. Here, the major aim is to perform FSV to compare on the original EMC device structure. However, a gap or space in an image does not represent any data points to be validated which increases computation time and effort of the experimental procedure. Therefore, a masking approach was used to cover the gaps in the image and to process FSV for the targeted area of the structure.

Masking is a technique used to cover or hide a portion of a device structure or an image with a mask layer. Here, a mask layer is a binary image comprising zero and non-zero values [34]. With respect to the input image structure shown in the above Figure 12, the gap or space is the area around the image which has no data points or not required to be validated is called as non-targeted area. Masking is performed visually, by layering a mask over the non-targeted (gap/space) area [35] [36]. Therefore, inserting a mask on the non-targeted area allows to perform FSV for the targeted area on any EMC devices given in the above Figure 12.

A mask was layered according to the gaps or space in the image as shown in the Figure 13 below. Here, the masked layers are treated as non-targeted area; while the remaining part known as the targeted area of the image is treated as major structure for validation and converted into data points for further experimental procedures.

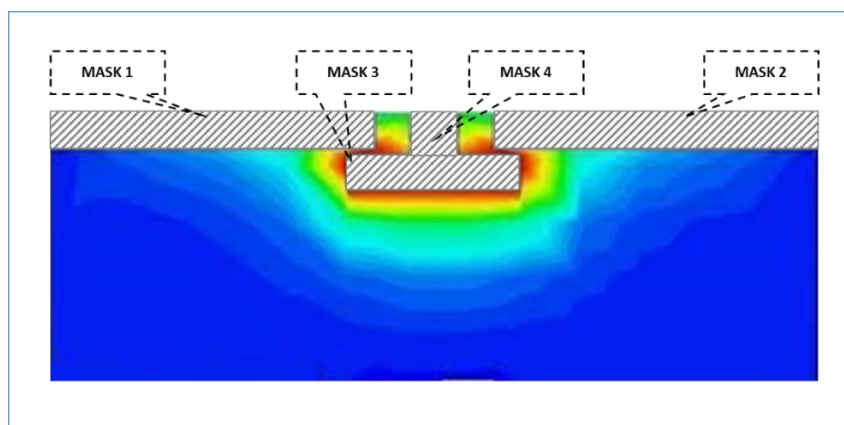


Figure 13: Masks layered on gap or space

Although the masking approach looks simple and easy to perform, it fails from certain drawbacks. Firstly, as the masking is layered visually; there are possibilities is adding human errors to identify the appropriate boundaries of the targeted area [35] [36]. Thus, there are chances to ignore some key data points for validation. For example, in Figure 13 above which represents UHF RFID tag, the area around the gap is contains key EM signals to be validated. In this case when a mask is layered visually, there are possibilities to ignore the key data points.

Secondly, this approach adds noise to the overall structure increasing the non-targeted data points. This in turn crowds the targeted area around the structure [35] [37]. On the other hand, while performing FSV, it is important to consider the transient phenomena; which is an impulsive noise found for a brief time period [38]. By using masking approach, the transient signals may vary as the EMC structure may not be in steady state. This happens in case of the mask layer precedes or surpasses the non-targeted area due to human error caused by visual masking [39].

To overcome the above issues of masking, a simple segmentation method was developed to compare structures with gap or space in it. The process of partitioning a digital image into multiple regions of pixel values is known as image segmentation. Each segment is treated separately which has similar characteristics such as texture, colour and pixel values [29] [40].

Moreover, segmentation can be processed using various methods such as manual and automatic segmentation, pixel based, edge or pattern, and region-based segmentation. These techniques are broadly classified into block-based and layer-based segmentation [40] [41].

Block based Image segmentation technique is the process of partitioning non-overlapping block of pixels in an image. On the other hand, layer-based segmentation techniques are performed by portioning an image into foreground, background, and mask layers [42].

In this research, Block based image segmentation technique is uses as it performed based on various features such as colour, pixel values and texture in image [43]. As discussed above, the aim of this research is to perform FSV on each segmented block where it compares the feature and trend and the input segmented blocks. Figure 14 below represents segmented blocks from original image shown in Figure 12.

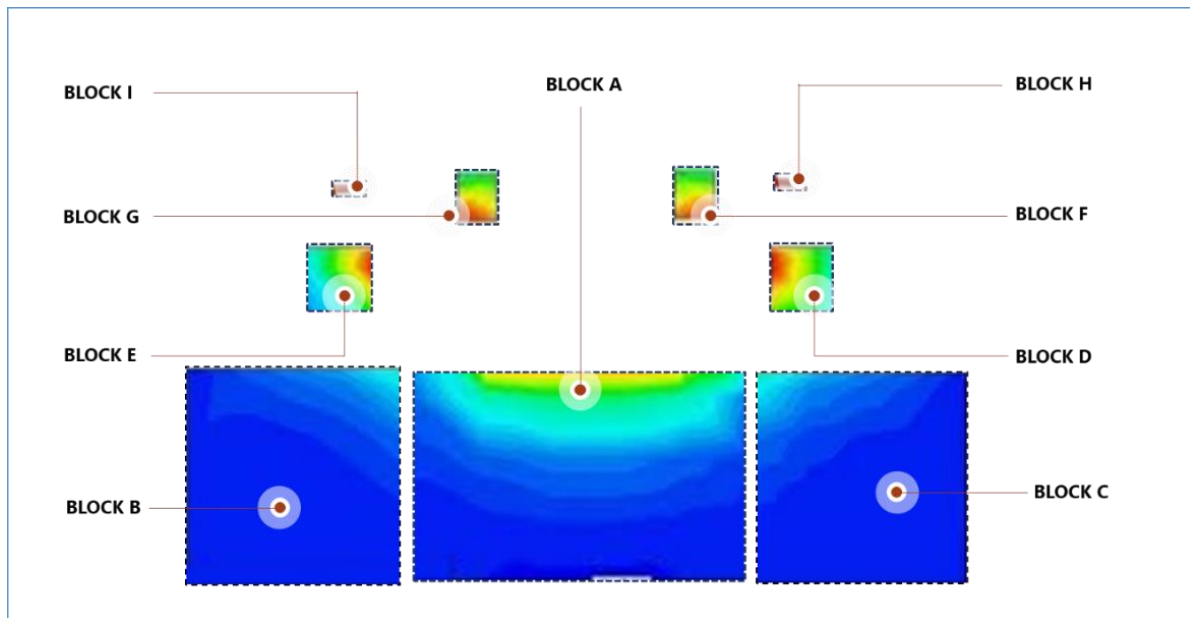


Figure 14: Segmented blocks of a RFID tag

This block-based technique usually depends on the pixel values in each block. Therefore, this method was used to manually segmented portions in an EMC device structure as show in figure below.

Figure 15 below demonstrates a simple example of how a 2-D array of pixel values with a dimension of 10×10 is segmented using three approaches; vertically segmented to form two 10×5 arrays, horizontally segmented to form two 5×10 arrays and finally segmented both vertically and horizontally to form four 5×5 arrays. The segmented 2-D array blocks are then exported and stored in a CSV file format.

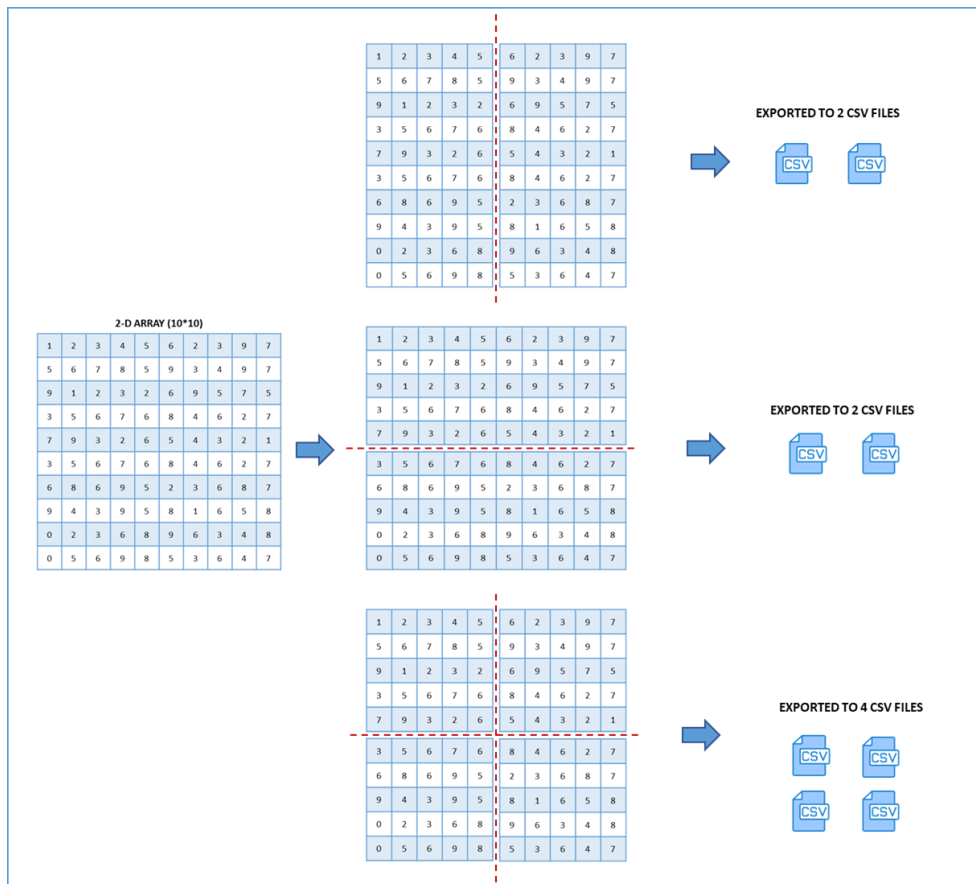


Figure 15: Example of Pixel level segmentation

These segmentations were performed manually by importing the Numpy CSV file output from the section 4.2 into Microsoft Excel tool. Although a CSV file may be viewed and edited in any programme, Microsoft Excel is the most used spreadsheet application for working with CSV files [44].

4.4 DATA MANIPULATION - COPY ROWS & COLUMNS USING EXCEL VBA MACRO

Now, an image is segmented into various 2-D arrays. Next step is to perform 2-D FSV for these arrays. However as seen in chapter 3, to perform 2-D FSV repeated 1-D FSV method was stated efficient. Therefore, in this section, the rows and columns form the 2-D array are separated. The operations performed in this section are the pre-requisite steps before performing a 1-D FSV.

The 2-D FSV was perform on the original structure of the input data and in various segmented arrays of the input data. For example, if a 2-D array is of 10x10 and the segmented 2-D array is 5x 5 in size are considered as shown in Figure 16 and 17 respectively. In Figure 16 where the original structure of the image is considered where each rows and columns are copied,

separated and exported as 10 individual sheets with row data, 10 individual sheets with column data are exported to individual CSV files.

While in Figure 17, where the input data is segmented into two 5x5 arrays, 5 individual sheets with row data and 5 individual sheets with column data are exported to individual CSV files. Figure 16 and 17, represents an illustrative example to split and export 1-D row and column data. However, the data should have at least 32 points to perform FSV. If not, interpolating the data points to get adequate number of points is required. These data points are then interpolated back to the original number of points.

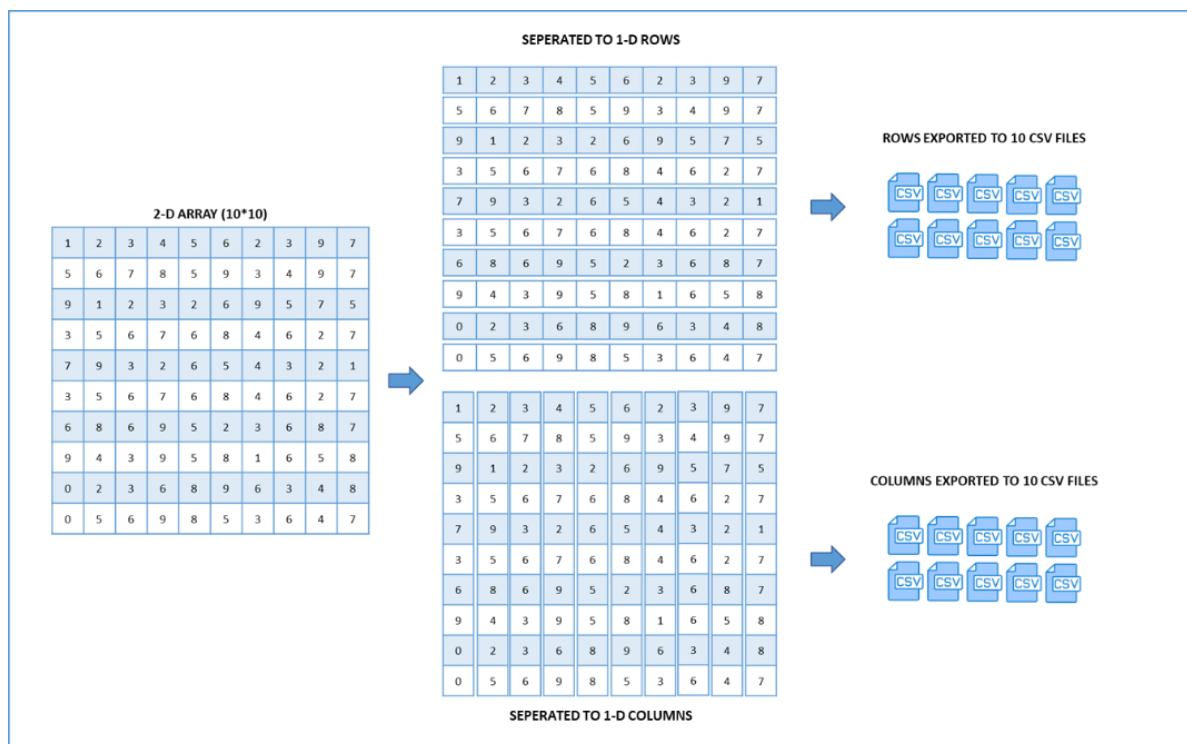


Figure 16: Example of 2-D full array copied to individual row and column CSV files

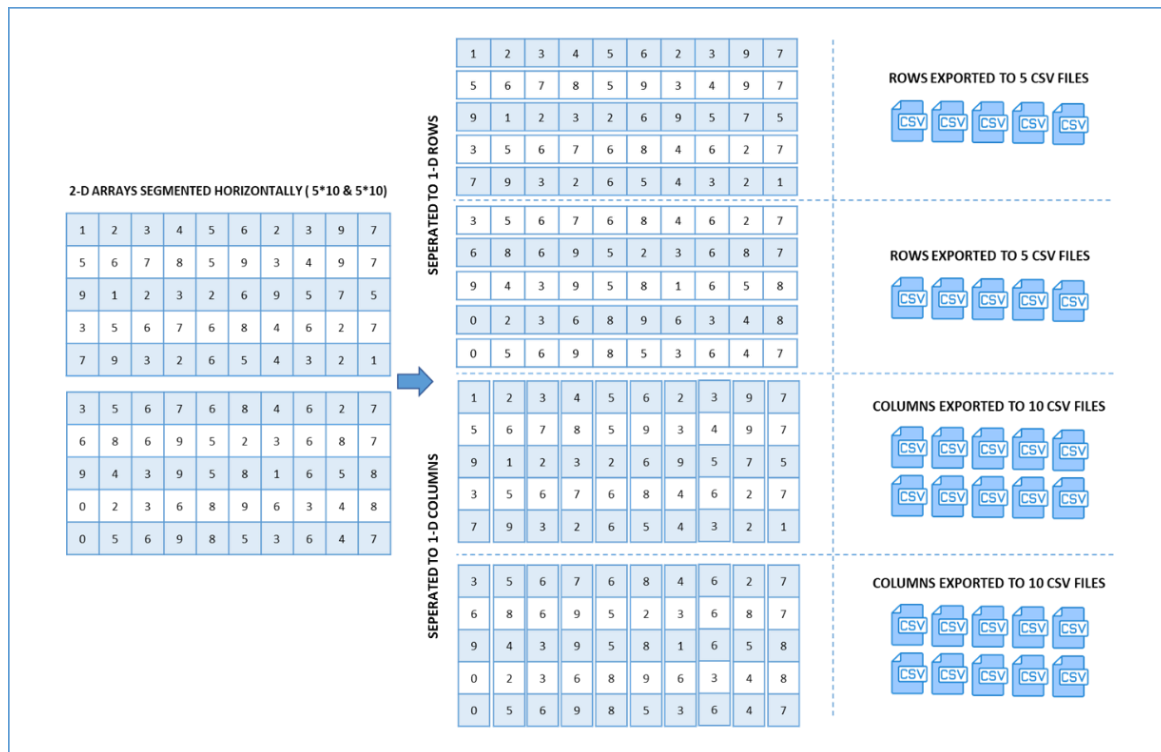


Figure 17: Example of 2-D segmented array copied to individual row and column CSV files

As mentioned above, all the output files in this section are exported in a CSV (comma-separated values) format; which is the most basic and widely used format for importing and exporting data in a computer programme or function. As previously discussed in section 4.2, the CSV file format offers several advantages, including the ability to visualize data in a table structure, the ability to store data in a smaller file size, CSV files are machine-readable and CSV files are non-proprietary [45] [46]. Although a CSV file may be viewed and edited in any programme, Microsoft Excel is the most used spreadsheet application for working with CSV files. Microsoft Excel is an excellent visual inspection tool to monitor dimensions and number of the input data points throughout the process. This ensures there is no data loss during this step. Hence, the process of separating 1-D rows and columns from 2-D array in original input image structure and in segmented arrays to CSV files were performed using Microsoft Excel.

A simple approach for performing this process would be manually copying, pasting, and saving each row and column to individual files and saving each file to a .CSV file format. These repetitive manual steps will consume a huge amount of man hours depending upon the dimension of the 2-D data and the number of 2-D data planned for the experimental purpose. Adapting automation for performing the repetitive manual steps will help in reducing the time taken and in reducing the manual error rate of this process [47] [48].

Microsoft Excel’s Visual Basic Application allows users to automate several aspects using an Excel feature known as Macro, which is developed to consistently complete a task based on the instructions written in visual basic programming codes. Macro feature help in saving the time taken for completing a series of repeated steps faster when compared manually. This also helps in reducing the error rate created during the manual task [47]. The Macro in this section is developed in a macro-enabled Excel worksheet using Visual Basic for Applications (VBA) programming language. The macro has been successfully tested to be compatible with Excel 365 version and on Windows 10 operation system.

Figure 18 below represents the steps followed to separate each row and column data to individual sheets and save it.

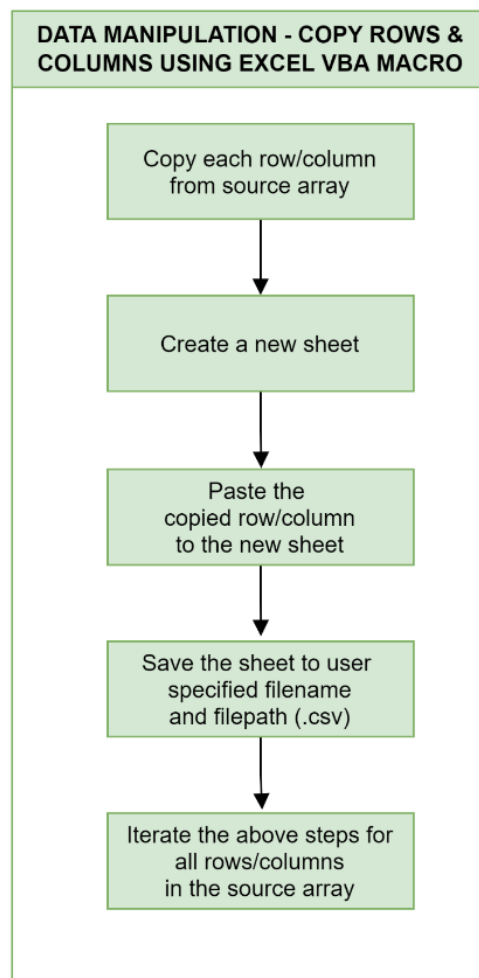


Figure 18: Steps to separate rows and columns from a 2-D array into individual CSV files

To perform the above steps in Figure 18, modularizing macros for row and column-wise export to separate CSV files is a viable method. The process of modularization is to group a selection of programming instructions that performs a certain function. Modularisation allows for the

reuse of modules or functions in a macro, breakdown the complexity of a problem statement and to save development time [48]. For example, if a module is created for copying and saving rows to separate sheets in.csv formats, the module can be reused for storing columns data by changing the range selection from the row to the column section. The VBA Macro functions in Appendix 3 and Appendix 4 were created to copy each row/column from the source sheet and export it to independent CSV files. The copy, paste, and save operations employ conventional excel operational keyboard action statements and loop conditional statements were used for intended iterations. The VBA Macro functions developed in this section are given in Appendix 5 and Appendix 6.

The Implementation of each steps as shown in Figure 18 are explained below.

4.4.1 COPY EACH ROWS AND COLUMNS FROM SOURCE ARRAY

To perform a row-wise or column-wise copy from a 2-D source array, the following syntax for select and copy keyboard action statements were used.

Syntax for copying one row,

'Select the first cell reference of the row to be copied

Range("Cell reference").Select

'Select full row

Range(Selection, Selection.End(xlToRight)).Select

'Copy the row selected in the above step

Selection.Copy

Syntax for copying one column,

'Select the first cell reference of the column to be copied

Range("Cell reference").Select

'Select full column

Range(Selection, Selection.End(xlToDown)).Select

'Copy the column selected in the above step

Selection.Copy

4.4.2 CREATE A NEW SHEET

After a row or a column data is copied from the 2-D source array workbook, a new destination workbook had to be created to paste the copied row or column data. This is performed by using the excel workbook creating syntax.

' Creating a new workbook

Workbooks.Add

4.4.3 PASTE THE COPIED ROW/COLUMN TO THE NEW SHEET

After a destination workbook is created, the copied row or column data will be pasted using the paste keyboard action statement. The paste operation is performed on the selected and copied row or column data from section 4.4.2; below is the syntax for paste operation.

'Special Paste operation form a selected and copied range

'Paste:=xlPasteAll : i.e. pasting original copied selection

'Operation:=xlNone : i.e. do not perform multiply, divide, add from the selected range

'SkipBlanks _ :=False : i.e. paste blanks data as well from the selection

Transpose:=False : i.e. pasting rows to columns or vice versa. This is based on the destination formatting requirement for next stages of data processing

Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _ False, Transpose:=True

4.4.4 SAVE THE SHEET TO USER SPECIFIED FILENAME AND FILEPATH (.CSV)

After the row or column data are pasted to the destination workbooks, the file has to be saved to .CSV format for next stages of the experiment. The CSV files can be saved to any user-fed file name and file location. The operation is performed using below workbook save syntax.

'Save the workbook in .csv format with user-fed filename to the user-fed filepath

ChDir _ <mention file location>

*ActiveWorkbook.SaveAs fileName:=<mention file name> _, FileFormat:=xlCSV,
CreateBackup:=False*

4.4.5 ITERATE THE ABOVE STEPS FOR ALL ROWS/COLUMNS IN THE SOURCE ARRAY

Finally, dependent on the number of rows and columns of data that need to be exported to individual CSV files, all of the above statements from sections 4.4.1 to 4.4.4 will be iteratively executed. If the number of row and column data in the 2-D source array is ten, the for-loop statement iterates ten times for each row and column. The following is the for-loop syntax based on row or column data.

'For-loop to iterate row/column copy, paste and save operations

*numberOfRowsOrColumn = <mention the number of rows or columns in a 2-D
source array>*

For i = 1 To numberOfRowsOrColumn

<Select, copy, paste and save operations>

Next i

4.5 PERFORM 1-D FSV ON EACH ROW AND COLUMN SEPERATED

From the above section, a 2-D array of pixel values are separated into 1-D rows and columns and stored individually. In this section, 1-D FSV is performed on each row and column data separately. The detailed procedure to perform FSV using IEEE STD 1597.1 and 1579.2 [49][50] is explained in chapter 3. Further, these steps were developed in MATLAB as shown in Appendix 1. However, in this case, there are several 1-D data separated as rows and columns from a 2-D array. Therefore, the FSV process has to be repeated according to the number of row/column data. This is achieved by adding a For-loop which is a conditional iterative statement executed for the specified number of iterations.

For example, when 1-D FSV is iterated for 10 rows and 10 column files, each row and column data provide 10 files of ADM, FDM and GDM for row and column data separately. These are then exported and stored in CSV format as shown in Figure 19 below.

The MATLAB program developed to perform repeated 1-D FSV is given in Appendix 7.

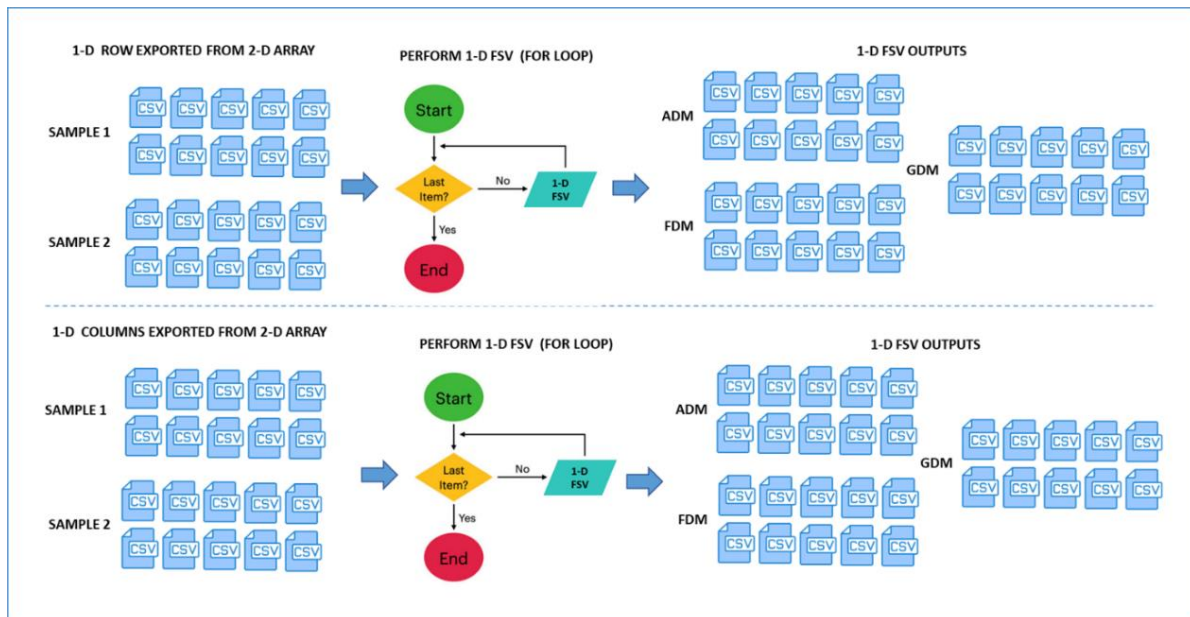


Figure 19: Repeated 1-D FSV using For-Loop

4.6 DATA MANIPULATION TO COMBINE ROWS & COLUMNS USING EXCEL VBA MACRO

The ADM, FDM and GDM outputs obtained by performing 1-D FSV for rows and columns in section 4.5 are recombined in this section to bring back 2-D data representation. As a result, this approach aids in visualizing and representing the data to be compared in its early 2-D array format, similar to how the greyscale images were transformed to 2-D NumPy arrays in section 4.2. Once the data is recombined to 2-D data representation, point-by-point 2-D FSV is performed using the weighting scheme. This provides ADM_i, FDM_i and GDM_i outputs in 2-D formats respectively; which is further explained in section 4.7.

The 1-D FSV outputs ADM, FDM and GDM obtained in Section 4.5 were stored in CSV formats. Hence, Microsoft Excel is used for working with CSV file formats as discussed in Section 4.3. Moreover, a Macro function developed in Visual Basic for Application (VBA) is used to automate the manual recombination steps. The Macro in this section is developed in a macro-enabled Excel worksheet using Visual Basic for Applications (VBA) programming language [47]. The macro has been successfully tested to be compatible with Microsoft Excel 365 version and on Windows 10 operation system.

For example, considering ADM output of 10 rows and column data from section 4.5, the macro developed here will open the 10 ADM output CSV files one by one in Microsoft Excel, copy the full 1-D ADM data, and paste it into a single output file as 10 consecutive rows, resulting in a 10×10 ADM row output. Similarly, the 10 ADM output CSV files from the 1-D column

are opened in Microsoft Excel, copied and pasted as 10 consecutive columns into the output file, resulting in a 10×10 ADM column output as shown in Figure 20.

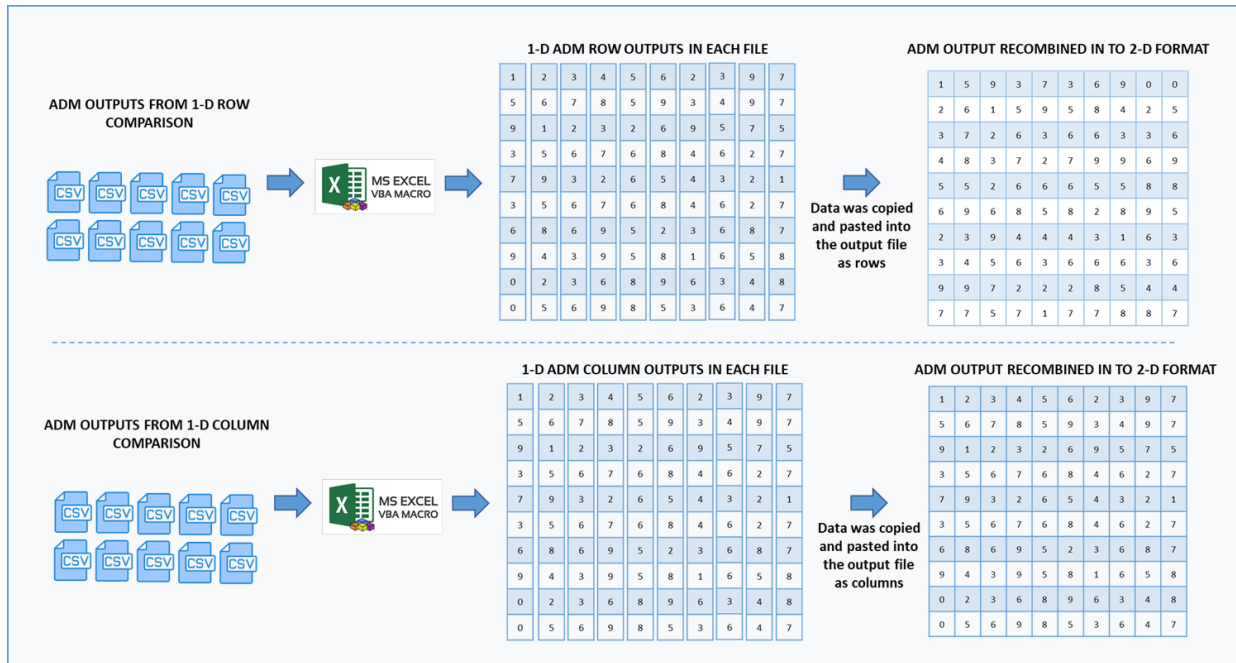


Figure 20: Recombining row and column data into a single output file using Excel Macro

The approach used to recombine row and column data to single file is divided in to 4 major steps as shown in the below Figure 21 and the implementation of each step are discussed below.

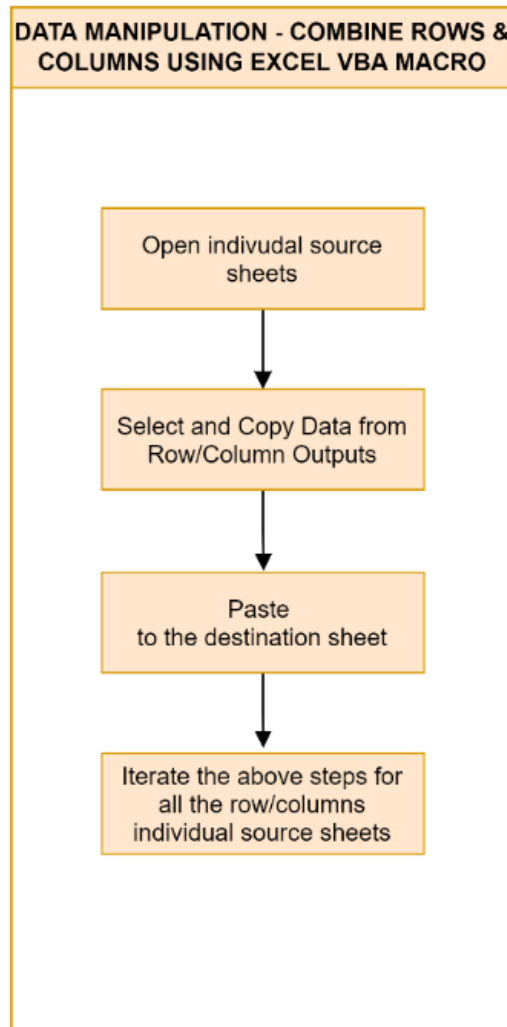


Figure 21: Steps to recombine row and column data into a single output file

4.6.1 OPEN INDIVIDUAL SOURCE SHEETS

To open each row/column output source sheets in Microsoft Excel, the following workbook syntax is used,

'Steps to open individual source sheet and copy column

inputFile = filePath & fileName & ".csv"

Workbooks.Open inputFile

4.6.2 SELECT AND COPY DATA FROM THE ROWS/COLUMNS OUTPUTS

The full data from the row/column output files are selected and copied using keyboard action statements. The syntax to perform this operation is as follows,

'Select the first cell reference of the row to be copied

Range("Cell reference").Select

'Select full column. Data in the ADM, FDM and GDM outputs are stored as columns

Range(Selection, Selection.End(xlToDown)).Select

'Copy the selection in the above step

Selection.Copy

4.6.3 PASTE TO THE DESTINATION SHEET

After a data is copied from the row/column outputs, the copied data will be pasted using the paste keyboard action statement. Below is the syntax for paste operation,

'Special Paste operation form a selected and copied range

'Paste:=xlPasteAll : i.e. pasting original copied selection

'Operation:=xlNone : i.e. do not perform multiply, divide, add from the selected range

'SkipBlanks _ :=False : i.e. paste blanks data as well from the selection

Transpose:=False : i.e. pasting rows to columns or vice versa. This is based on the destination formatting requirement for next stages of data processing

Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _ False,

Transpose:=<True or False>

4.6.4 ITERATE THE ABOVE STEPS FOR ALL THE ROW/COLUMN SOURCE SHEETS

Finally, dependent on the number of rows and columns outputs that need to be recombined into 2-D data format, all the above statements from sections 4.6.1 to 4.6.3 will be iteratively executed. The following is the for-loop syntax based on row or column data,

'For-loop to iterate open, copy and paste

numberOfRowsOrColumn = <mention the number of rows or columns outputs>

For i = 1 To numberOfRowsOrColumn

<Open, Select, copy and paste operations>

Next i

4.7 PERFORM 2-D FSV USING WEIGHTED SCHEME

The 1-D FSV performed on each row and column data separately and combined are treated on point-by-point basis through weighting scheme.

The final step of the proposed methodology is to apply point-by-point 2-D FSV formula on the 1-D xDMi outputs obtained from previous sections. The 2-D FSV formulas given in equation (17) (18) and (21) from chapter 3, section 3.3.5 was used to calculate point-by-point 2-D FSV for x and y direction. Moreover, the weighting factors K_V and K_H were also calculated as given in equations (19) and (20) in chapter 3 section 3.3.5.

The total average value and histogram for ADM, FDM and GDM were also obtained in the similar method used in the 1-D FSV, section 3.1 Further, from the above point-by-point FSV outputs, statistical analysis is obtained by calculating continuous Probability Density Function (PDF) as given in chapter 3 section 3.4.

On the other hand, to analyse the performance of the FSV outputs; the first four statistical moments, namely mean, variance, skewness and kurtosis respectively are calculated using the point-by point FSV values as given in equation 32, 33, and 34 in chapter 3 section 3.4.2.

Finally, the KS -test were performed for the FSV outputs obtained from original structure and segmented approach. These results were analysed to verify the performance of the proposed methodology. In Chapter 5, various 2-D data are processed using the steps explained in this chapter and detailed analysis of each results are given. These steps are developed using MATLAB. The program for this section is given in Appendix 8, 9 and 10.

4.8 REFERENCES

- [1] Z. Abdallah, L. Du and G. Webb, "Data Preparation", *Encyclopedia of Machine Learning and Data Mining*, pp. 318-327, 2017.
- [2] J. Brownlee, *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Machine Learning Mastery, 2022.
- [3] G. Zhang, A. Duffy, A. Orlandi, D. Febo, L. Wang and H. Sasse, "Comparison of Data With Multiple Degrees of Freedom Utilizing the Feature Selective Validation Method", *IEEE Transactions on Electromagnetic Compatibility*, vol. 58, no. 3, pp. 784-791, 2016.
- [4] S. Roy, P. Sharma, K. Nath, D. Bhattacharyya and J. Kalita, "Pre-Processing: A Data Preparation Step", *Encyclopedia of Bioinformatics and Computational Biology*, pp. 463-471, 2019.
- [5] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection", *ACM Computing Surveys*, vol. 41, no. 3, pp. 1-58, 2009.
- [6] C. Shorten and T. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning", *Journal of Big Data*, vol. 6, no. 1, 2019.

- [7] M. Dong, Y. Cui, X. Jing, X. Liu and J. Li, "End-to-End Target Detection and Classification with Data Augmentation in SAR Images", *2019 IEEE International Conference on Computational Electromagnetics (ICCEM)*, 2019.
- [8] C. Bauckhage, F. Beaumont and V. Toborek, "Notes on Image Processing in Python Reading and Writing Images", 2021 [Online]. Available: https://www.researchgate.net/publication/356969703_Notes_on_Image_Processing_in_Python_Reading_and_Writing_Images. [Accessed: 06- May- 2022]
- [9] S. Jeyalakshmi and S. Prasanna, "Measuring distinct regions of grayscale image using pixel values", *International Journal of Engineering & Technology*, vol. 7, no. 11, p. 121, 2017.
- [10] K. Padmavathi and K. Thangadurai, "Implementation of RGB and Grayscale Images in Plant Leaves Disease Detection – Comparative Study", *Indian Journal of Science and Technology*, vol. 9, no. 6, 2016.
- [11] C. Saravanan, "Color Image to Grayscale Image Conversion", *2010 Second International Conference on Computer Engineering and Applications*, 2010.
- [12] M. Nixon and A. Aguado, *Feature extraction and image processing for computer vision*. Elsevier Science, 2019.
- [13] N. Alaa and I. Abidne, "Image Processing with Python: An Introduction", 2021.
- [14] T. Oliphant, "Python for Scientific Computing", *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10-20, 2007.
- [15] B. Malloy and J. Power, "Quantifying the Transition from Python 2 to 3: An Empirical Study of Python Applications", *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2017.
- [16] A. Sayeth Saabith, T. Vinothraj and M. Fareez, "A Review on Python Libraries and IDEs for Data Science", *International Journal of Research in Engineering and Science (IJRES)*, vol. 9, no. 11, 2021.
- [17] H. Gujar, P. Mhatre, S. Ghanate, S. Chile, S. Kadam, D. Kurle and S. Shitole, "Python Based Image Processing", 2016.
- [18] R. Chityala and S. Pudipeddi, *Image processing and acquisition using Python*. Taylor & Francis, 2014.
- [19] C. Harris, K. Millman, S. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. Smith, R. Kern, M. Picus, S. Hoyer, M. van Kerkwijk, M. Brett, A. Haldane, J. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke and T. Oliphant, "Array programming with NumPy", *Nature*, vol. 585, no. 7825, pp. 357-362, 2020.
- [20] S. van der Walt, S. Colbert and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation", *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22-30, 2011.
- [21] C. Bauckhage, "NumPy / SciPy Recipes for Image Processing: Creating Fractal Images", 2015.

- [22] Z. Karimi, "NumPy Quick Review", 2021.
- [23] S. van der Walt, S. Colbert and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation", *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22-30, 2011.
- [24] C. Blumzon and A. Pănescu, "Data Storage", *Good Research Practice in Non-Clinical Pharmacology and Biomedicine*, 2019.
- [25] T. Döhmen, H. Mühleisen and P. Boncz, "Multi-Hypothesis CSV Parsing", *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, 2017.
- [26] J. Hunt, *Advanced Guide to Python 3 Programming*. Springer International Publishing, 2019.
- [27] C. U. Smith and M. Woodside, "Performance Validation at Early Stages of Software Development", 2000.
- [28] A. Müller and S. Guido, *Introduction to machine learning with Python*. O'Reilly Media, 2016.
- [29] S. Prabu and J. Gnanasekar, "A Study on Image Segmentation Method for Image Processing", *Recent Trends in Intensive Computing*, 2021.
- [30] M. Ziai and J. Batchelor, "Temporary On-Skin Passive UHF RFID Transfer Tag", *IEEE Transactions on Antennas and Propagation*, vol. 59, no. 10, pp. 3565-3571, 2011.
- [31] M. Rutschlin, "5G Antenna Design for Mobile Phones | The SIMULIA Blog", *The SIMULIA Blog*, 2020.
- [32] M. Gopikrishna, D. Krishna and C. Aanandan, "A Compact Rectangular Monopole Antenna Design with a Novel Feed for an Improved UWB Performance", *Radioengineering*, vol. 27, no. 1, pp. 63-69, 2018.
- [33] L. Tan, "Image file formats", *Biomedical Imaging and Intervention Journal*, vol. 2, no. 1, 2006.
- [34] U. Qidwai and C. Chen, *Digital image processing*. Boca Raton, Fla.: CRC/Chapman & Hall, 2010.
- [35] J. Enns and V. Di Lollo, "What's new in visual masking?", *Trends in Cognitive Sciences*, vol. 4, no. 9, pp. 345-352, 2000.
- [36] J. Wu, W. Lin, G. Shi, X. Wang and F. Li, "Pattern Masking Estimation in Image With Structural Uncertainty", *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4892-4904, 2013.
- [37] N. Strobel, "Nonlinear unsharp masking methods for image contrast enhancement", *Journal of Electronic Imaging*, vol. 5, no. 3, p. 353, 1996.
- [38] R. Jauregui, Gang Zhang, J. Rojas-Mora, O. Ventosa, F. Silva, A. Duffy and H. Sasse, "Analyzing Transient Phenomena in the Time Domain Using the Feature Selective Validation (FSV) Method", *IEEE Transactions on Electromagnetic Compatibility*, vol. 56, no. 4, pp. 825-834, 2014.

- [39] S. Macknik and M. Livingstone, "Neuronal correlates of visibility and invisibility in the primate visual system", *Nature Neuroscience*, vol. 1, no. 2, pp. 144-149, 1998.
- [40] S. Tripathi, K. Kumar, B. Singh and R. Singh, "Image Segmentation: A Review", *International Journal of Computer Science and Management Research*, vol. 1, no. 4, 2022.
- [41] A. Aly, S. Bin Deris and N. Zaki, "Research Review for Digital Image Segmentation Techniques", *International Journal of Computer Science and Information Technology*, vol. 3, no. 5, 2011.
- [42] S. B and R. B, "COMPARATIVE STUDY OF BLOCK-BASED IMAGE SEGMENTATION TECHNIQUE", *International Research Journal of Mathematics, Engineering and IT*, vol. 3, no. 12, 2016.
- [43] Maheswari and Radha, "Enhanced Hybrid Compound Image Compression Algorithm Combining Block and Layer-based Segmentation", *The International journal of Multimedia & Its Applications*, vol. 3, no. 4, pp. 119-131, 2011.
- [44] E. Billo, *Excel for scientists and engineers*. Hoboken, NJ: John Wiley, 2007.
- [45] P. Carvalho, P. Hitzelberger, B. Otjacques, F. Bouali and G. Venturini, "Information Visualization for CSV Open Data Files Structure Analysis", *Proceedings of the 6th International Conference on Information Visualization Theory and Applications*, 2015.
- [46] G. van den Burg, A. Nazábal and C. Sutton, "Wrangling messy CSV files by detecting row and type patterns", *Data Mining and Knowledge Discovery*, vol. 33, no. 6, pp. 1799-1820, 2019.
- [47] A. Ishak, R. Ginting and T. Amalia, "Macro excel (VBA) implementation in designing booking information systems in uniform convection (Case Study: Kholidi Taylor SME, Medan Denai)", *IOP Conference Series: Materials Science and Engineering*, vol. 1003, no. 1, p. 012019, 2020.
- [48] P. Blayney and Z. Sun, "Using Excel and Excel VBA for Preliminary Analysis in Big Data Research", *Advances in Data Mining and Database Management*, pp. 110-136, 2019.
- [49] "IEEE Standard for Validation of Computational Electromagnetics Computer Modeling and Simulations".
- [50] "IEEE Recommended Practice for Validation of Computational Electromagnetics Computer Modeling and Simulations", 2011

CHAPTER - 5

EXPERIMENT RESULTS AND ANALYSIS

The main objective of this research study is to perform 2-Dimensional FSV on irregular image datasets. To achieve this, the proposed methodology developed in Chapter 4 is applied to regular (without space or gaps in the original image structure) and irregular (with space or gaps in the original image structure) image datasets in two stages.

In the first stage, the 2-Dimensional FSV is performed on regular image datasets using the two approaches from the proposed methodology, as shown in Chapter 4, Figure 1. The outcomes of both approaches are compared to demonstrate the feasibility of performing 2-D FSV on irregular image datasets. In the second stage, the proven segmented approach (approach 2) shown in Chapter 4, Figure 1, of the proposed methodology is applied to irregular image data sets containing gaps or spaces within the image structure. The 2-D FSV results obtained from the segmented approach are analysed and evaluated for effectiveness in achieving the core objectives.

In summary, this chapter analyses and discusses the outcomes of the results to demonstrate the feasibility and effectiveness of the proposed methodology for performing 2-D FSV on irregular image datasets.

5.1 INPUT DATASETS

To perform the proposed methodology in two stages, suitable regular and irregular image data sets are identified to be used as input data for the 2-D FSV processing. For the first stage, images of “Einstein” and “E-field” through a slice of a mode stirred reverberation chamber at different stirrer positions output images were obtained are chosen as the input data sets [1]. These images were selected because they represent typical examples of regular image data.

Before using these images as input data, the quality of the images is verified, and they are augmented to provide a specific number of data sets, as discussed in Chapter 4, Section 4.1. The aim of this augmentation is to enhance the images and ensure that the data sets are suitable for the experiments.

The “E-field” and “Einstein” images are shown in Figures 1 and 2, respectively, and their respective names are listed in Table 1. Throughout this chapter, the image names listed in Table 1 are used to display, analyse, and discuss the outcomes of the stage 1 results. By using these specific image datasets and following the proposed methodology, the feasibility of performing

2-D FSV on regular image data sets can be demonstrated. This is an important first step towards achieving the core objectives of the research study.

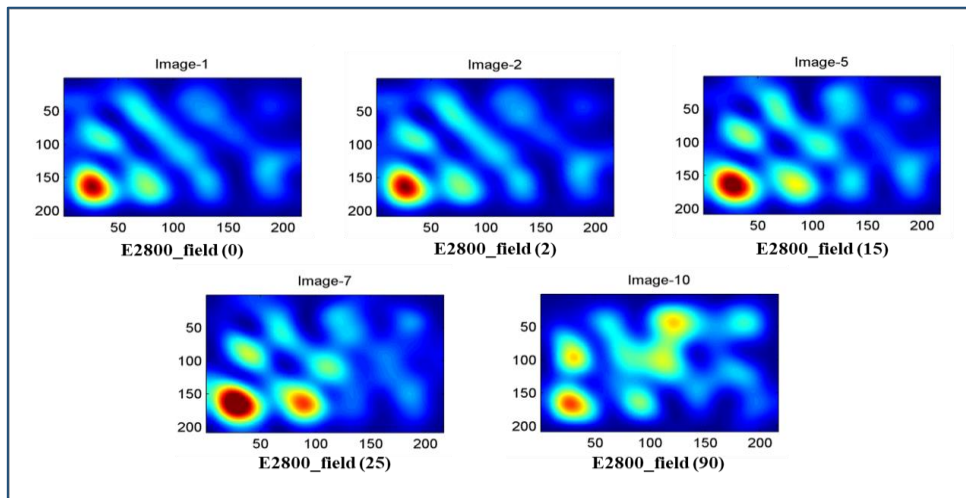


Figure 1: E-field image dataset used for Stage- 1 FSV processing [1]

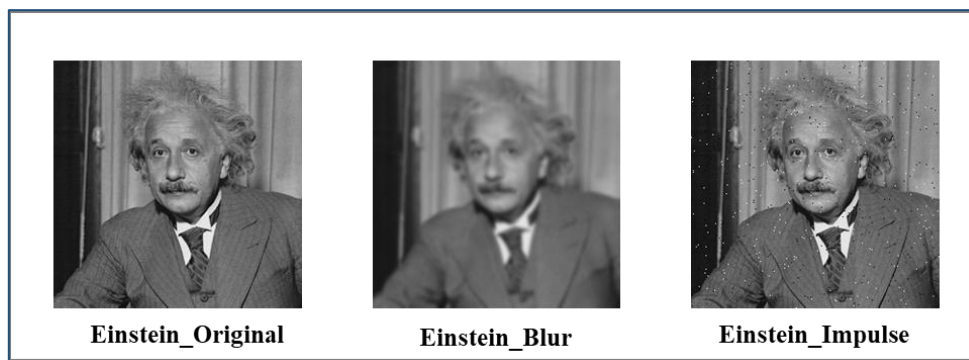


Figure 2: Einstein Image dataset used for Stage- 1 FSV processing [1]

Dataset Category	Image Reference	Image Name
E-Field output images of EMC data	Image -1	E2800_field (0)
	Image -2	E2800_field (2)
	Image -5	E2800_field (15)
	Image -7	E2800_field (25)
	Image -10	E2800_field (90)
Real-time Einstein images	Einstein - 1	Einstein_Orignal
	Einstein - 2	Einstein_Blur
	Einstein - 3	Einstein_Impulse

Table 3: References of Input Image datasets used for Stage-1 FSV processing

For the second stage of result analysis, irregular image data sets with gaps or spaces within the device structure are used to perform 2-D FSV. To achieve this, a microstrip patch antenna was designed using Computer Simulation Technology (CST). The CST software is commonly used for designing, analysing, and optimising electromagnetic (EM) components.

The microstrip patch antenna was designed and chosen to operate at different frequencies, including 5.2 GHz, 5.3 GHz, and 5.6 GHz, and its simulated surface current outputs were exported as images to ensure that the data sets are appropriate for conducting experiments. These images, which are the surface current outputs of the microstrip patch antenna at various frequencies are displayed in Figure 3, and their names and references are provided in Table 2. Throughout this chapter, the results of Stage 2 are presented, analysed, and discussed using the image names listed in Table 2. The antenna was designed with gaps or spaces within the device structure to show that the proposed methodology could be applied to real-world problems in the field of electromagnetic (EM) components and systems that have irregular structures.

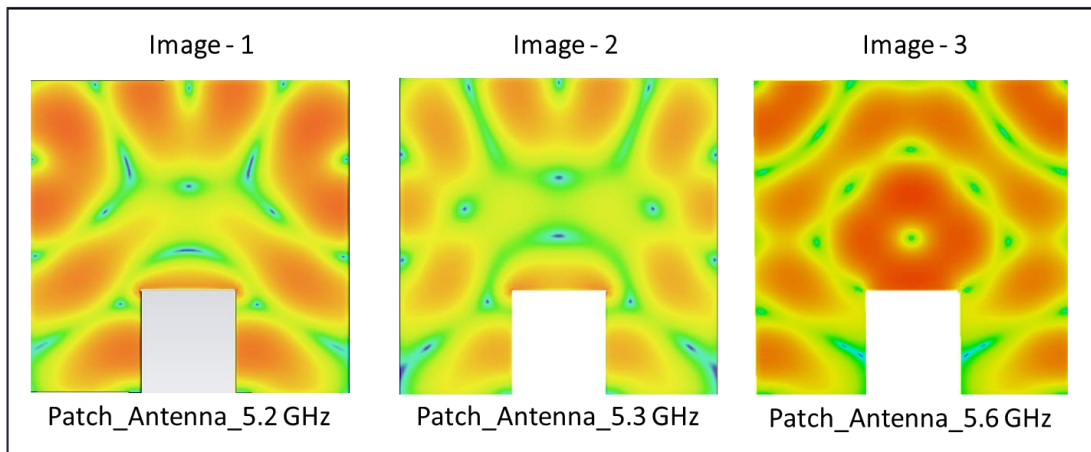


Figure 3: Microstrip patch antennas simulations operating at frequencies of 5.2 GHz, 5.3 GHz, and 5.6 GHz used for Stage-2 FSV processing

Dataset Category	Image Reference	Image Name
Simple microstrip patch antennas	Image -1	Patch_Antenna_5.2 GHz
	Image -2	Patch_Antenna_5.3 GHz
	Image -3	Patch_Antenna_5.6 GHz

Table 4: References of Input Image datasets used for Stage-2 FSV processing

5.2 STAGE -1: RESULT ANALYSIS OF 2-D FSV ON REGULAR IMAGE DATA SETS

In this section, the regular image data sets, such as E-field and Einstein images, are processed using the proposed methodology from Chapter 4. This methodology is performed using two approaches: Approach 1 involves performing 2-D FSV on the full image structure, and Approach 2 involves performing 2-D FSV by segmentation and recombination. The results from both approaches are compared to validate the segmented approach's feasibility on irregular images with gaps or spaces in the image structure, which is the novelty of this research study. This is further discussed in this chapter, Section 5.3.

The steps to perform Approach 1 and Approach 2 of the proposed methodology are discussed in detail in Chapter 4. In addition, to demonstrate Approach 2 to perform 2-D FSV, the E-field and Einstein images from Figures 1 and 2 are divided into multiple vertical and horizontal blocks.

The 2-D FSV outputs, such as confidence histograms, total values, and statistical analysis from approaches 1 and 2, are obtained and analysed here. To evaluate the results from the two approaches, the Kolmogorov-Smirnov test (KS-test) was performed. This is a statistical test used to compare two probability distributions [5]. The KS-test aids in determining any significant difference between the outputs of Approach 1 and Approach 2 and determining whether the segmented approach (Approach 2) is feasible for irregular image data sets. The steps for performing the K-S test are discussed in Chapter 3, Section 3.4.3.

5.2.1 FSV RESULT ANALYSIS FOR E-FIELD INPUT IMAGE DATA SETS

In this section, the E-Field input images shown in this chapter, Figure 1, are used as input samples for 2-D FSV processing. The detailed steps for Approach 1 and Approach 2 of the proposed methodology are discussed in Chapter 4. Figure 4 and Table 3 below demonstrate the various E-field images used as input datasets for 2-D FSV processing. For the first comparison, E2800_Field (0) and E2800_Field (2) are used and shown as comparison-1 in Figure 4. Similarly, for second, third, and fourth comparisons, E2800_Field (2) and E2800_Field (15), E2800_Field (2) and E2800_Field (25), and E2800_Field (2) and E2800_Field (90) are used and named as comparison-2, comparison-3, and comparison-4, respectively.

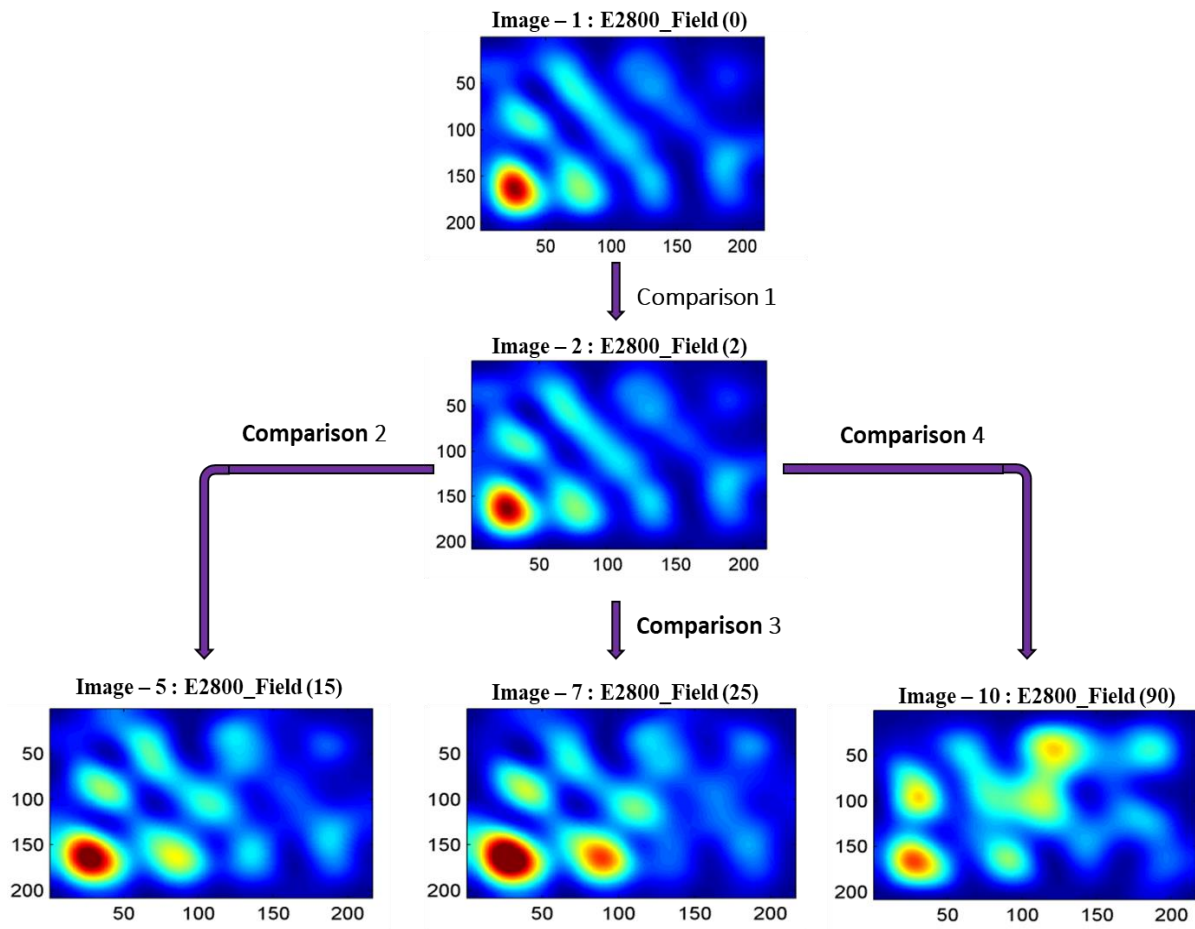


Figure 4: Different comparisons of E-field images used as input datasets for 2-D FSV processing

Comparison Reference	Images compared for 2-D FSV Processing
Comparison - 1	E2800_Field (0) vs E2800_Field (2)
Comparison - 2	E2800_Field (2) vs E2800_Field (15)
Comparison - 3	E2800_Field (2) Vs E2800_Field (25)
Comparison - 4	E2800_Field (2) Vs E2800_Field (90)

Table 5: Comparison references and E-field images used as input datasets for 2-D FSV processing

The outputs and results of the 2-D FSV for Comparison - 1 are shown and examined in the following section, 5.2.2. Meanwhile, the outputs and results of the 2-D FSV for Comparison - 2, Comparison - 3, and Comparison - 4 are shown and discussed in the subsequent section, 5.2.5. Finally, the overall conclusions and implications of the 2-D FSV results obtained for the E-Field images are analysed and discussed in section 5.3.

5.2.2 FSV RESULTS OF E2800_FIELD (0) VS E2800_FIELD (2)

The FSV results obtained by performing approach 1 and approach 2 on E2800_field (0) and E2800_field (2), which is comparison - 1 in Figure 4 and table 3 are shown and discussed in detail in this section.

5.2.3 APPROACH 1: 2-D FSV RESULTS FOR FULL STRUCTURE

E2800_field (0) and E2800_field (2), as shown in figure 5 below, are used to perform 2-D FSV for the full structure of input images.

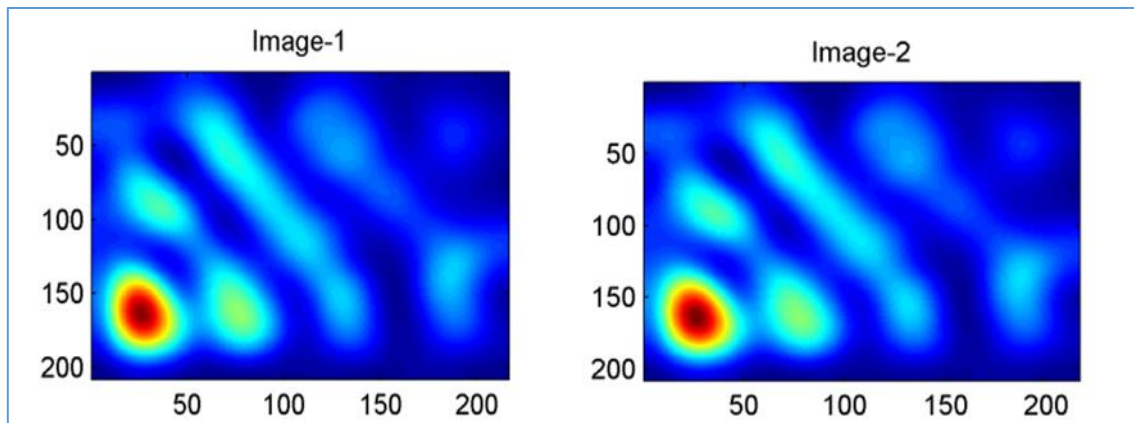


Figure 5: Full structures of E2800_field (0) vs E2800_field (2) images used in approach 1

Figure 6 represents the confidence histogram for ADM, FDM, and GDM outputs obtained from performing point-by-point 2-D FSV.

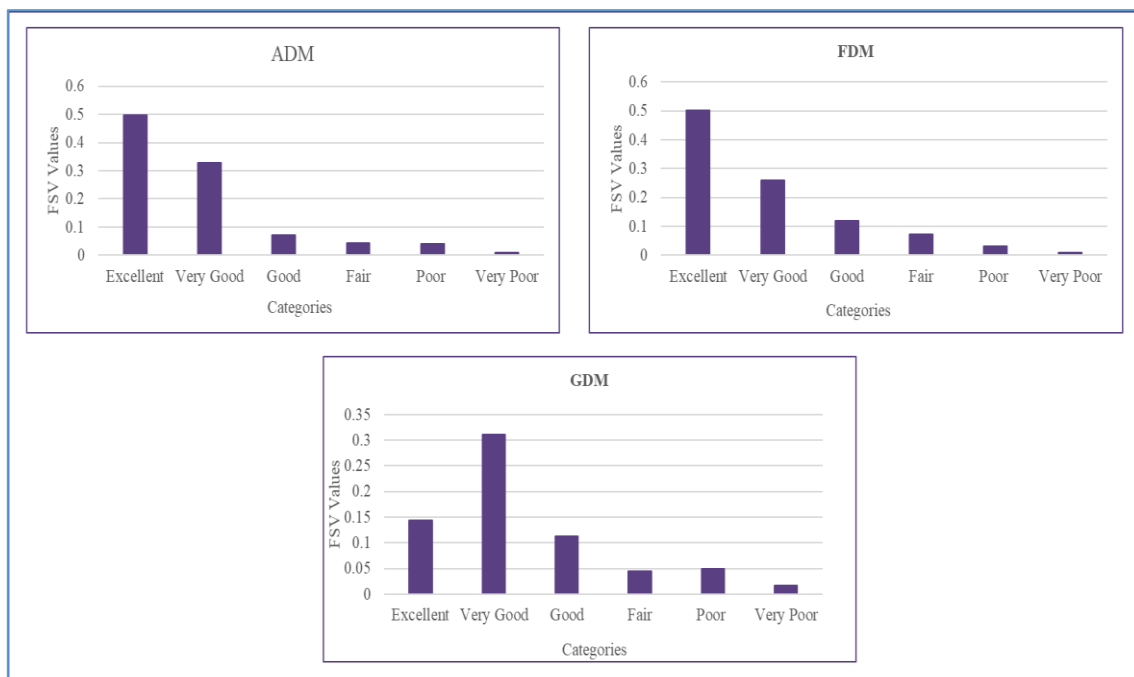


Figure 6: Confidence histograms for ADM, FDM and GDM

The mean values obtained for ADM, FDM and GDM as shown in table 4.

ADMtot	FDMtot	GDMtot
0.2706	0.2026	0.3858

Table 6: Total Value for ADM, FDM & GDM

From the above point-by-point FSV output, the overall measure, GDM, is used to determine the continuous Probability Density Function (PDF) and continuous Cumulative Density Function (CDF). As the PDFs provide a continuous distribution, it allows multiple comparisons to be analysed using a non-parametric test [2] [3]. However, before calculating the continuous PDFs, it is important to pre-process the point-by-point 2-D FSV obtained using the piece-wise linear interpretation table given in Chapter 3, Section 3.4.1.

Figures 7 represents the linearized GDM output, PDF, and CDF, respectively.

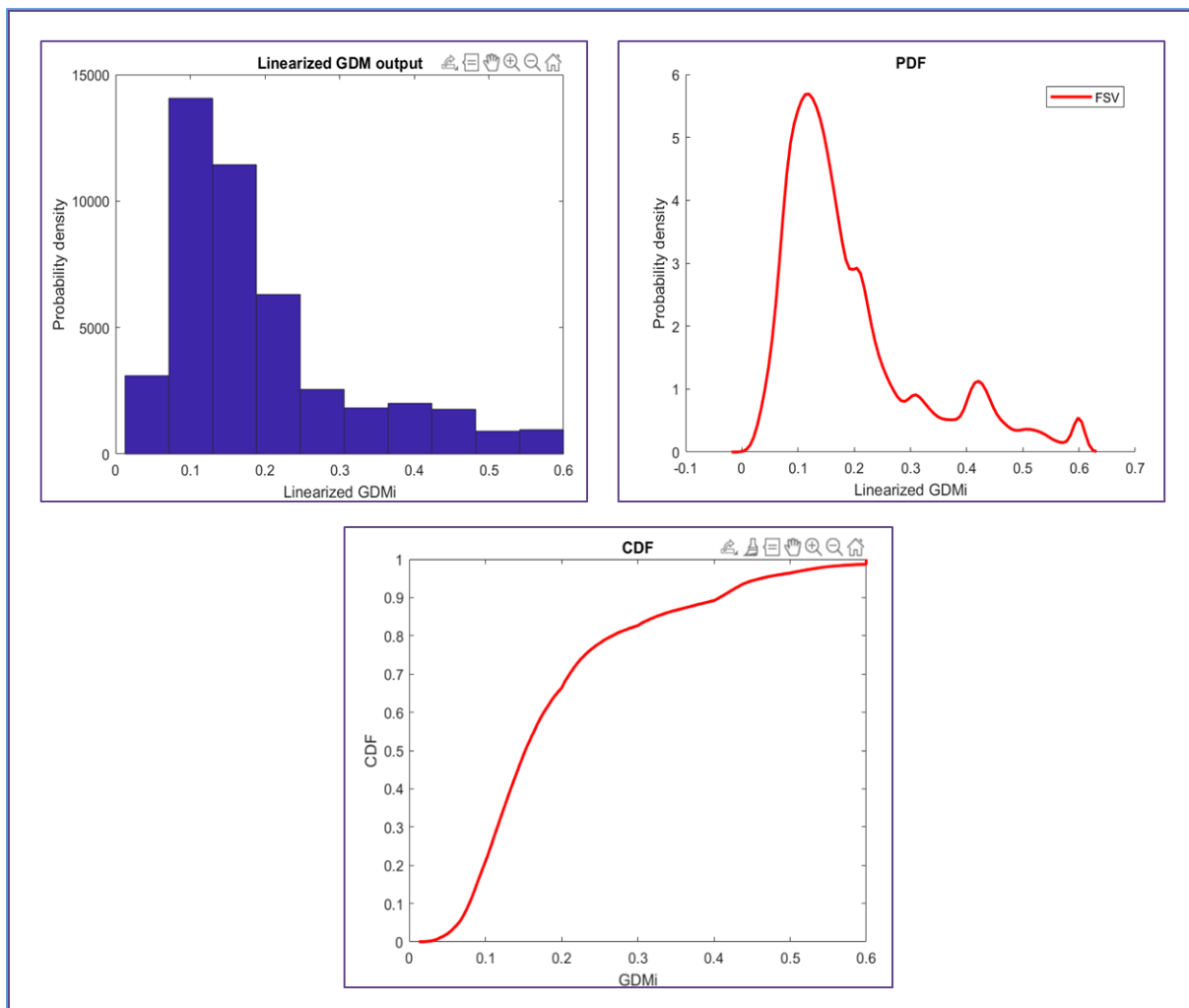


Figure 7: Linearized GDM, PDF and CDF Outputs

To analyse the performance of the FSV outputs, the first four statistical moments, namely mean, variance, skew, and kurtosis, were calculated [3]. The mean or total values were calculated using equation 5 in Chapter 2, and the respective outputs are given in Table 4 above. This value provides an absolute single value of a distribution, allowing it to aid the FSV interpretation. Similarly, other moments were also calculated to improve the analysis of FSV results using a single value. Here, the variance is used to determine the dispersion of the source data. The third-moment skewness provides the symmetry shape of the distribution. The fourth moment, kurtosis, provides the measure of the peaks and tails of a distribution [2] [3]. The variance, skewness, and kurtosis were calculated using the 32, 33, and 34, respectively, in Chapter 3, Section 3.4.2. Each value for the outputs is shown in Table 5.

Variance	Skewness	Kurtosis
0.0155	1.4003	4.3722

Table 7: Statistical Moments

5.2.4 APPROACH 2 - 2-D FSV RESULTS FOR SEGMENTED STRUCTURE

E2800_field (0) and E2800_field (2), as shown in figure 8 below, are used to perform 2-D FSV for the segmented structure of input images. The below input images were horizontally segmented into top and bottom blocks. where 2-D FSV is performed for E2800_field (0) top block versus E2800_field (2) top block and E2800_field (0) bottom block versus E2800_field (2) bottom block, respectively. Here, each block is treated as an individual 2-D image, and 2-D FSV is performed on each block separately. Finally, the FSV outputs from the top and bottom blocks are concatenated to bring the image back to its original structure.

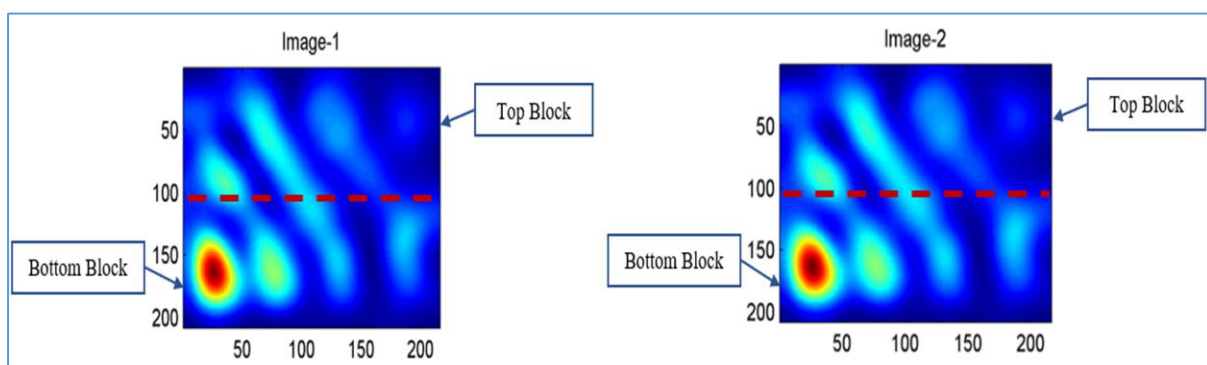


Figure 8: E2800_field (0) vs E2800_field (2) images segmented horizontally used in approach 2

Figure 9 represents the confidence histogram for ADM, FDM and GDM outputs obtained from performing point-by-point 2-D FSV respectively.

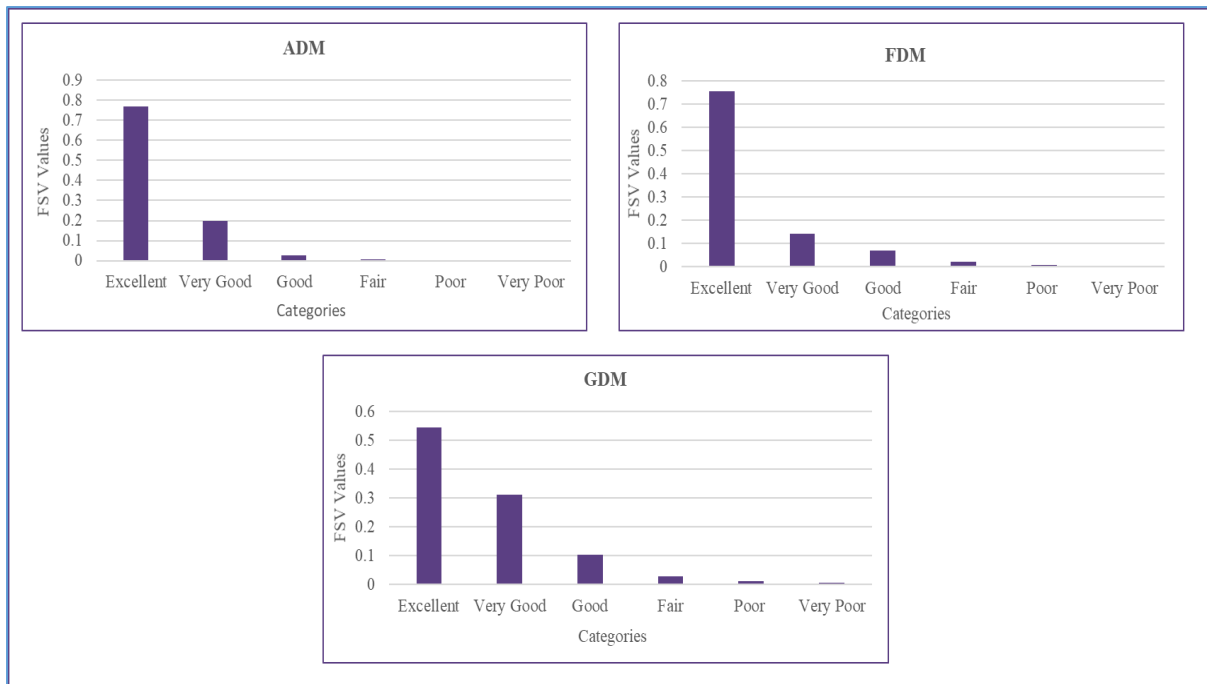


Figure 9: Confidence histograms for ADM, FDM and GDM

Below are the total average values obtained for ADM, FDM and GDM as shown in table 6.

ADMtot	FDMtot	GDMtot
0.0799	0.1033	0.1424

Table 8: Total Value for ADM, FDM & GDM

Further, from the above point-by-point 2-D FSV, the linearized GDM output, continuous Probability Density Function (PDF), and continuous Cumulative Density Function (CDF) were calculated, and their outputs are shown in figure 10.

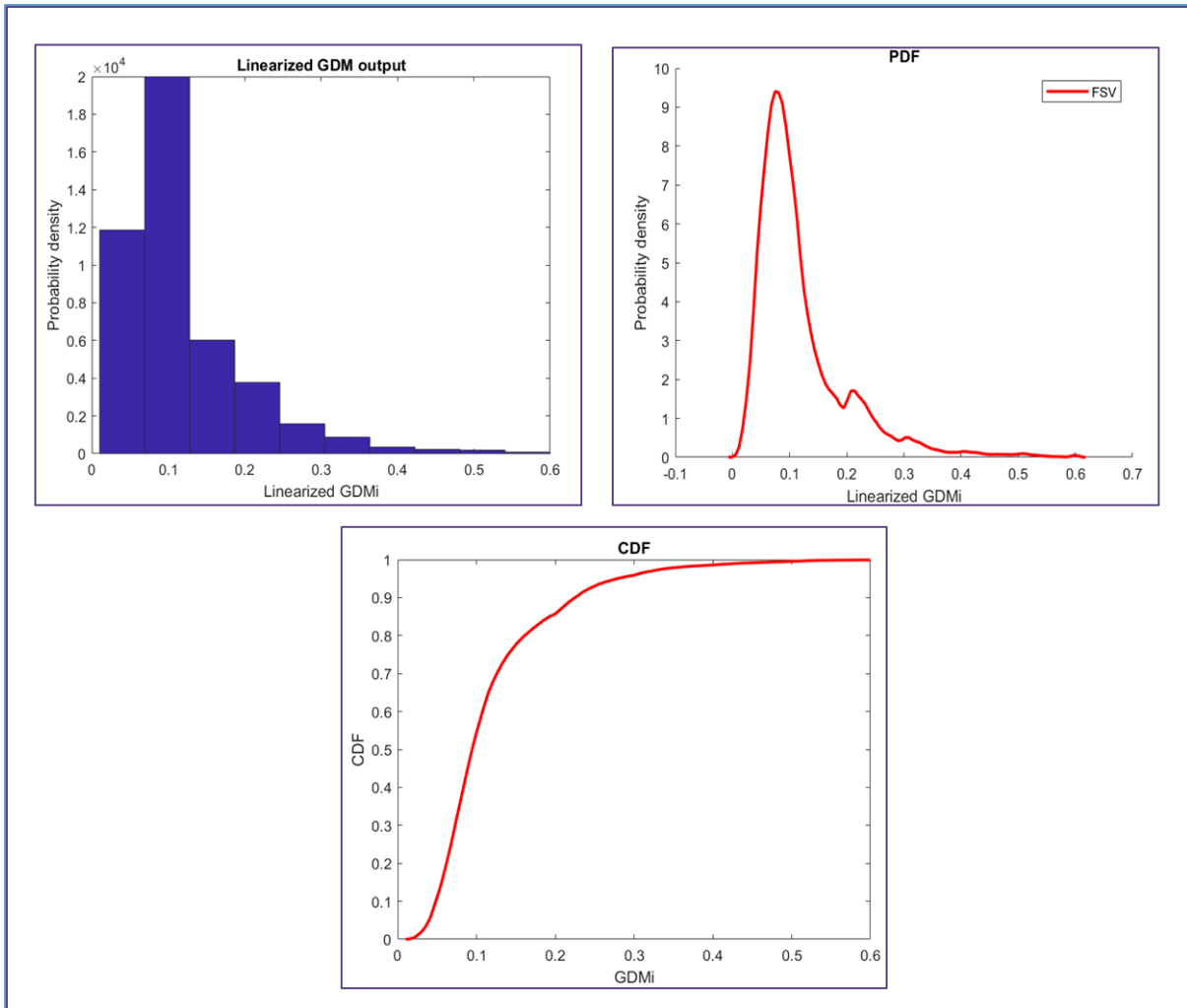


Figure 10: Linearized GDM, PDF and CDF Outputs

As the continuous distributions were obtained as shown above, this makes it possible to analyse the performance of the FSV outputs using statistical moments, namely mean, variance, skewness, and kurtosis [3]. The mean or total values were calculated and given in table 6 above. The other moments were calculated using the equations 32, 33, and 34 in Chapter 3, Section 3.4.2. Each value for the outputs is shown in the below table 7.

Variance	Skewness	Kurtosis
0.0064	2.0587	8.6140

Table 9: Statistical Moments

5.2.5 K-S TEST RESULTS FOR APPROACH 1 AND APPROACH 2

Finally, the KS - test steps explained in Chapter 3, Section 3.4.3, were performed for the FSV outputs obtained from the original structure and segmented approach. This test was performed to determine whether the distributions are from the same input data sets, i.e., the KS-test

verifies whether the original image structure and the segmented and then recombined image structure are the same. The null hypothesis of the data sets is determined to verify the test results. When the null hypothesis is true, it means that the distributions must be from the same data sets. While the null hypothesis is false, it means the distributions are from different data sets [4] [5]. For this, the test statistic values D and $D_{critical}$ values were calculated as shown in Chapter 3, Section 3.4.3. The test results are shown in Table 8. These results were analysed to verify the performance of the proposed methodology. As $D_{critical} > D$, the null hypothesis is true or accepted.

Test statistic (D)	$D_{critical}$ value
0	0.132

Table 10: K-S Test Results for Approach 1 and Approach 2

5.2.6 FSV RESULTS OF OTHER E-FIELD DATA SETS

Similar to the above steps shown in 5.2.2.1, the proposed methodology of Approach 1 and Approach 2 are applied to the other E-Field image data sets as shown in Figure 4 and Table 3. The confidence histogram of ADM, FDM, and GDM obtained from performing point-by-point 2-D FSV, the total average values obtained for ADM, FDM, and GDM, linearized GDM outputs, continuous Probability Density Function (PDF), Cumulative Density Function (CDF), statistical moments, namely mean, variance, skewness, kurtosis and KS test results for each comparison dataset are calculated and shown below.

The FSV results from Approach 1, Approach 2, and KS- test outputs obtained by comparing E2800_Field (2) and E2800_Field (15) are shown in Figures 11, 12, and 13, respectively.

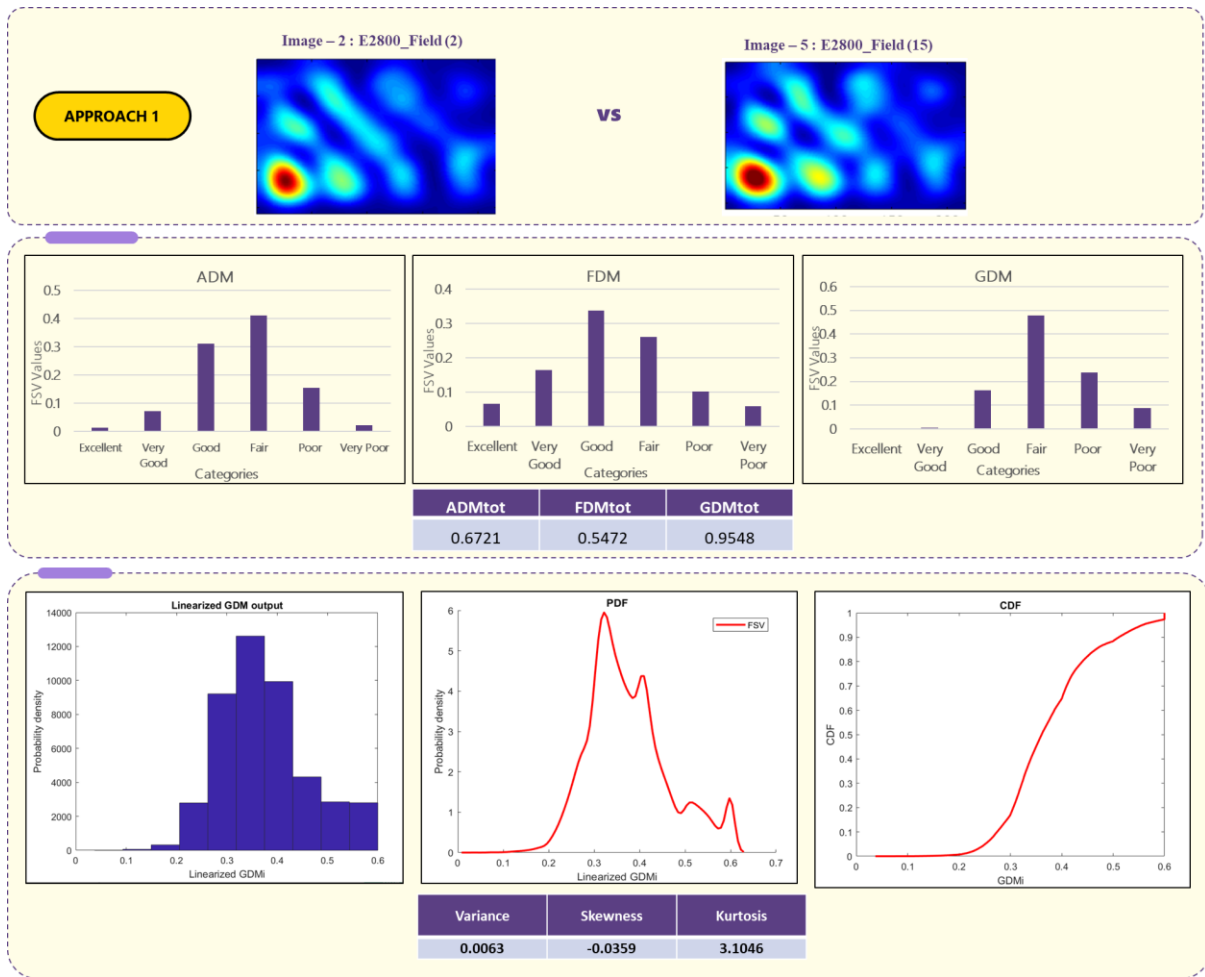


Figure 11: 2-D FSV Results of E2800_Field (2) vs E2800_Field (15) using Approach 1

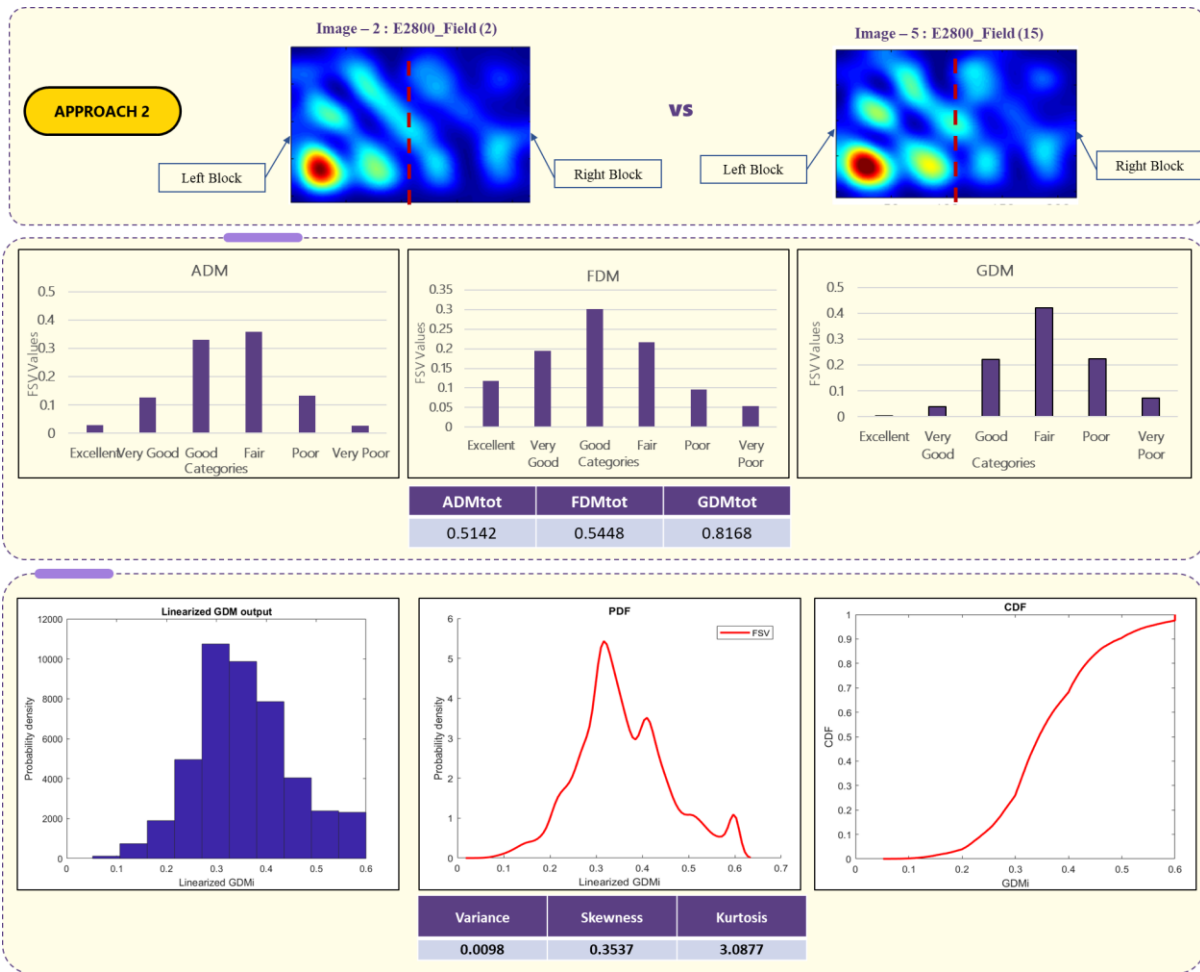


Figure 12: 2-D FSV Results of E2800_Field (2) vs E2800_Field (15) using Approach 2

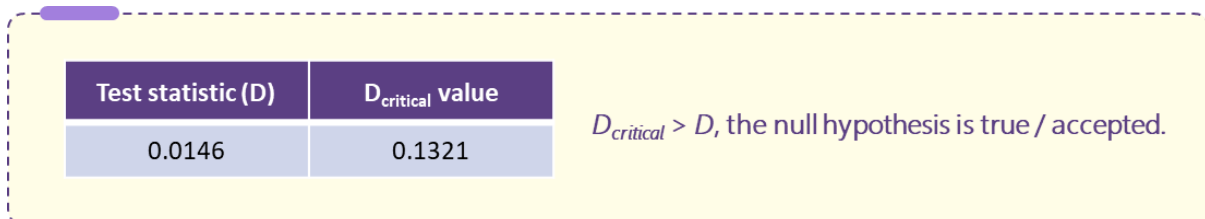


Figure 13: K-S Test results of E2800_Field (2) vs E2800_Field (15) comparing Approach 1 and Approach 2

The FSV results from approach 1, approach 2 and KS- test outputs obtained by comparing E2800_Field (2) Vs E2800_Field (25) are shown in Figure 14 , 15 and 16 respectively.

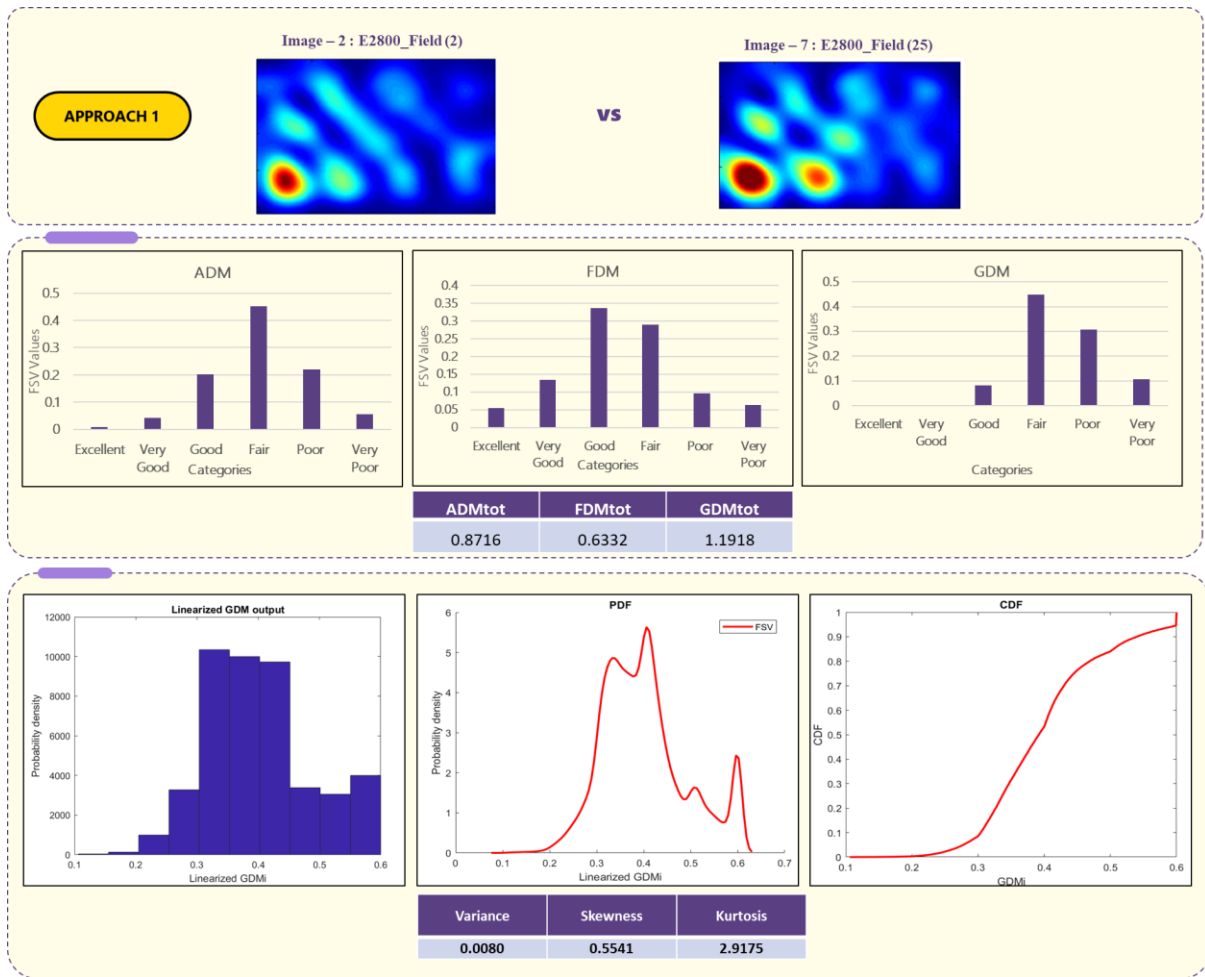


Figure 14: 2-D FSV Results of E2800_Field (2) vs E2800_Field (25) using Approach 1

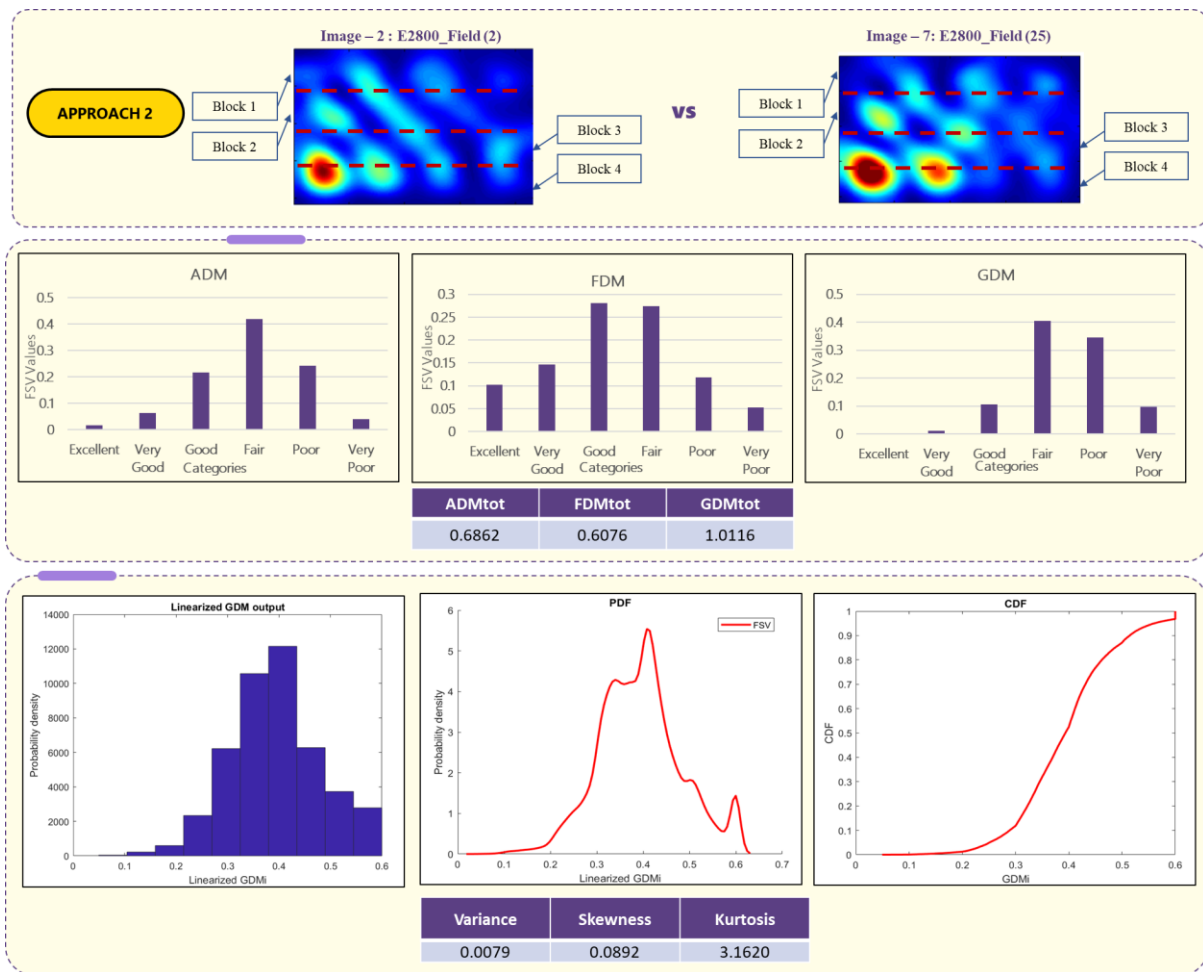


Figure 15: 2-D FSV Results of E2800_Field (2) vs E2800_Field (25) using Approach 2

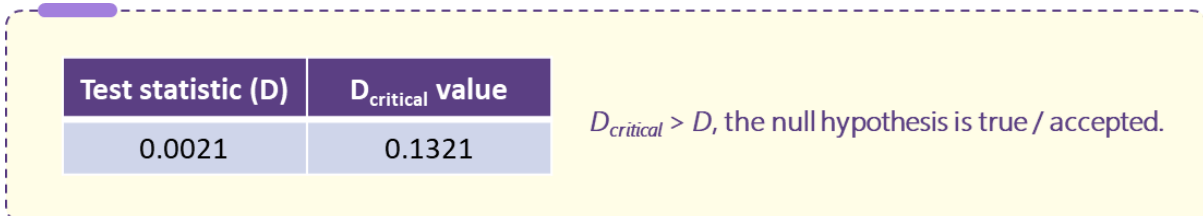


Figure 16: K-S Test results of E2800_Field (2) vs E2800_Field (25) comparing Approach 1 and Approach 2

The FSV results from approach 1, approach 2 and KS- test outputs obtained by comparing E2800_Field (2) Vs E2800_Field (90) are shown in Figure 17 , 18, and 19 respectively.

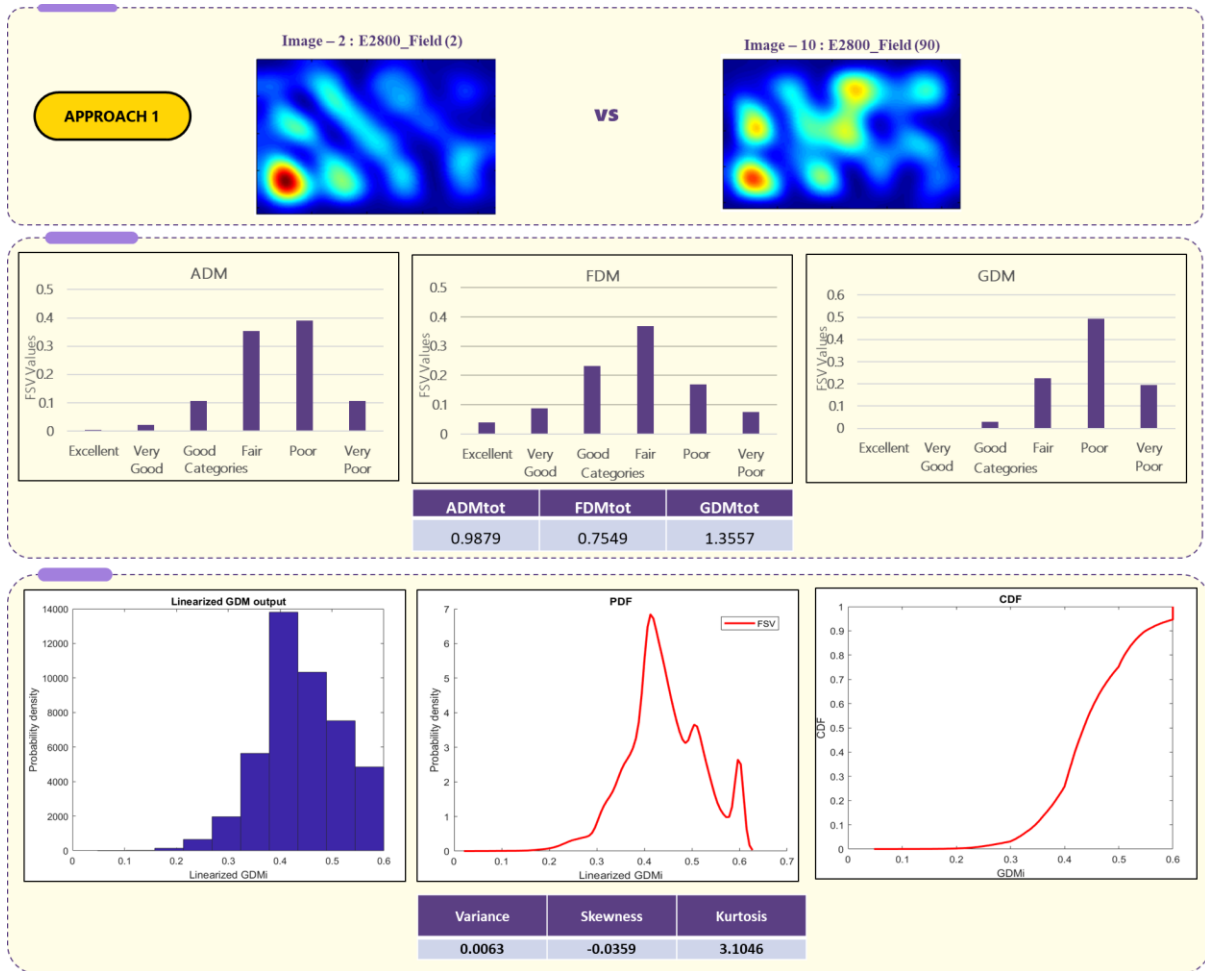


Figure 17: 2-D FSV Results of E2800_Field (2) vs E2800_Field (90) using Approach 1

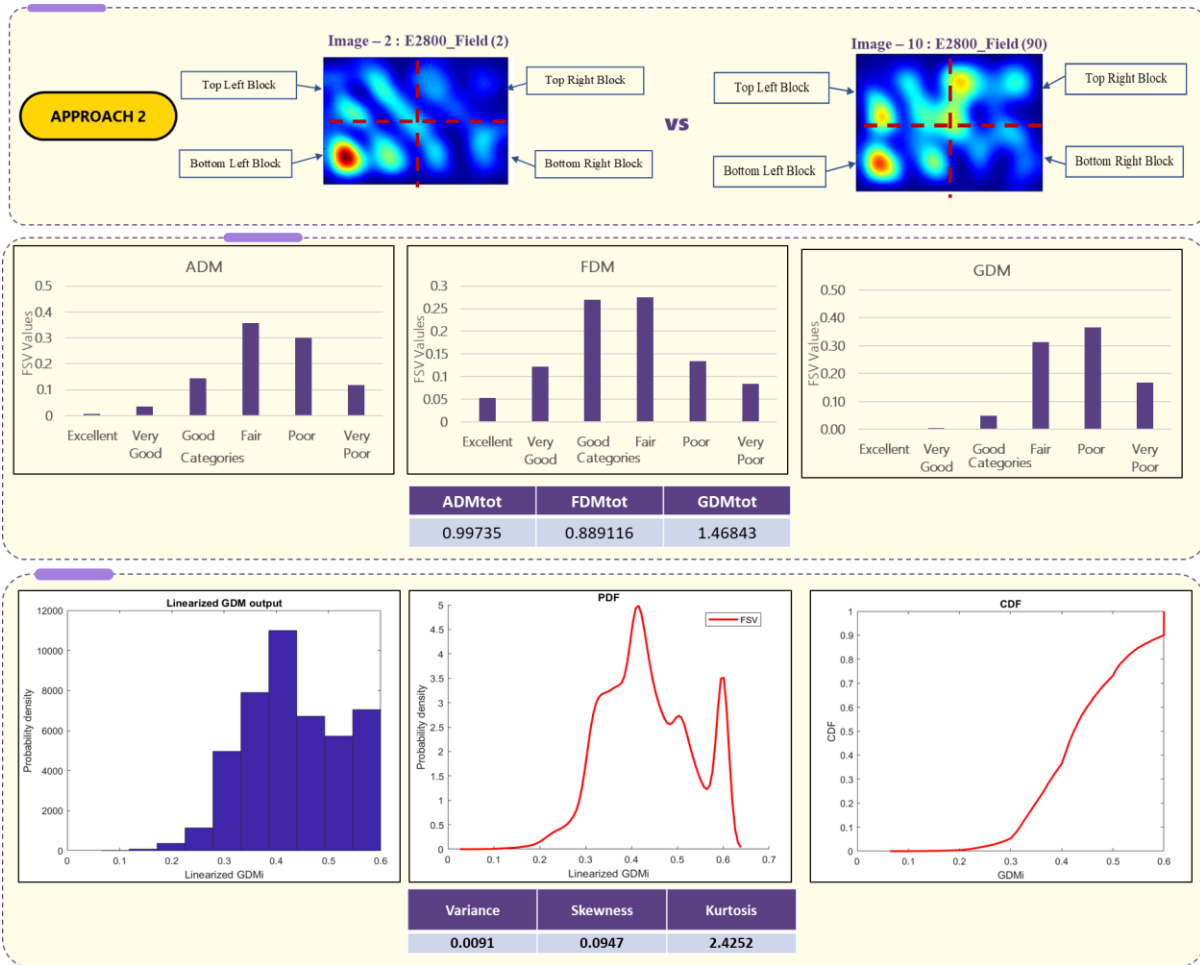


Figure 18: 2-D FSV Results of E2800_Field (2) vs E2800_Field (90) using Approach 2

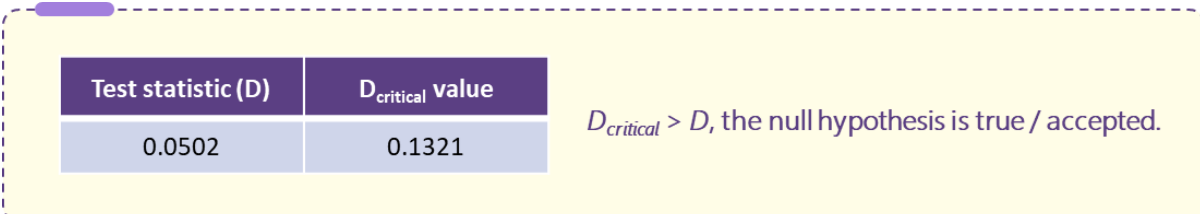


Figure 19: K-S Test results of E2800_Field (2) vs E2800_Field (90) comparing Approach 1 and Approach 2

5.2.6 FSV RESULT ANALYSIS FOR EINSTEIN IMAGE DATA SETS

In this section, the Einstein input images shown in this chapter, Figure 2, are used as input samples for two sets of 2-D FSV comparisons. These were carried out for additional demonstration of the effectiveness of the proposed methodology. Figure 20 and Table 9 below demonstrate the two sets of comparisons of Einstein images that were used as input datasets for 2-D FSV processing. Einstein_Original vs. Einstein_Blur is used for comparison 1, and Einstein_Original vs. Einstein_Impuse is used for comparison 2.

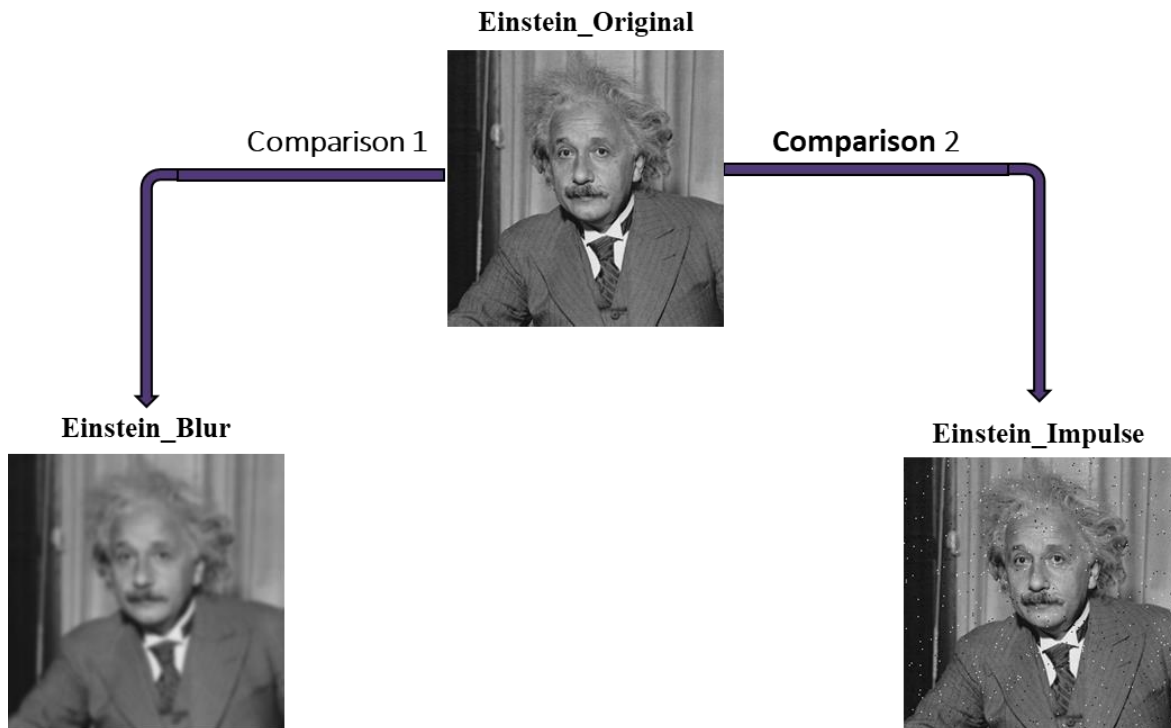


Figure 20: Different comparisons of Einstein images used as input datasets for 2-D FSV processing

Comparison Reference	Images compared for 2-D FSV Processing
Comparison - 1	Einstein_Original vs Einstein_Blur
Comparison - 2	Einstein_Original vs Einstein_Impulse

Table 11: Comparison references of Einstein images used as input datasets for 2-D FSV processing

The results from the 2-D FSV for Comparison - 1 and Comparison - 2 are shown and discussed in this section. The overall conclusions and implications of the 2-D FSV results obtained for the Einstein images are analysed along with the E-Field images and discussed in Section 5.3.

Figure 21, 22, and 23 represent point-by-point FSV outputs and KS- test for comparison-1, which is Einstein_Original vs Einstein_Blur respectively.

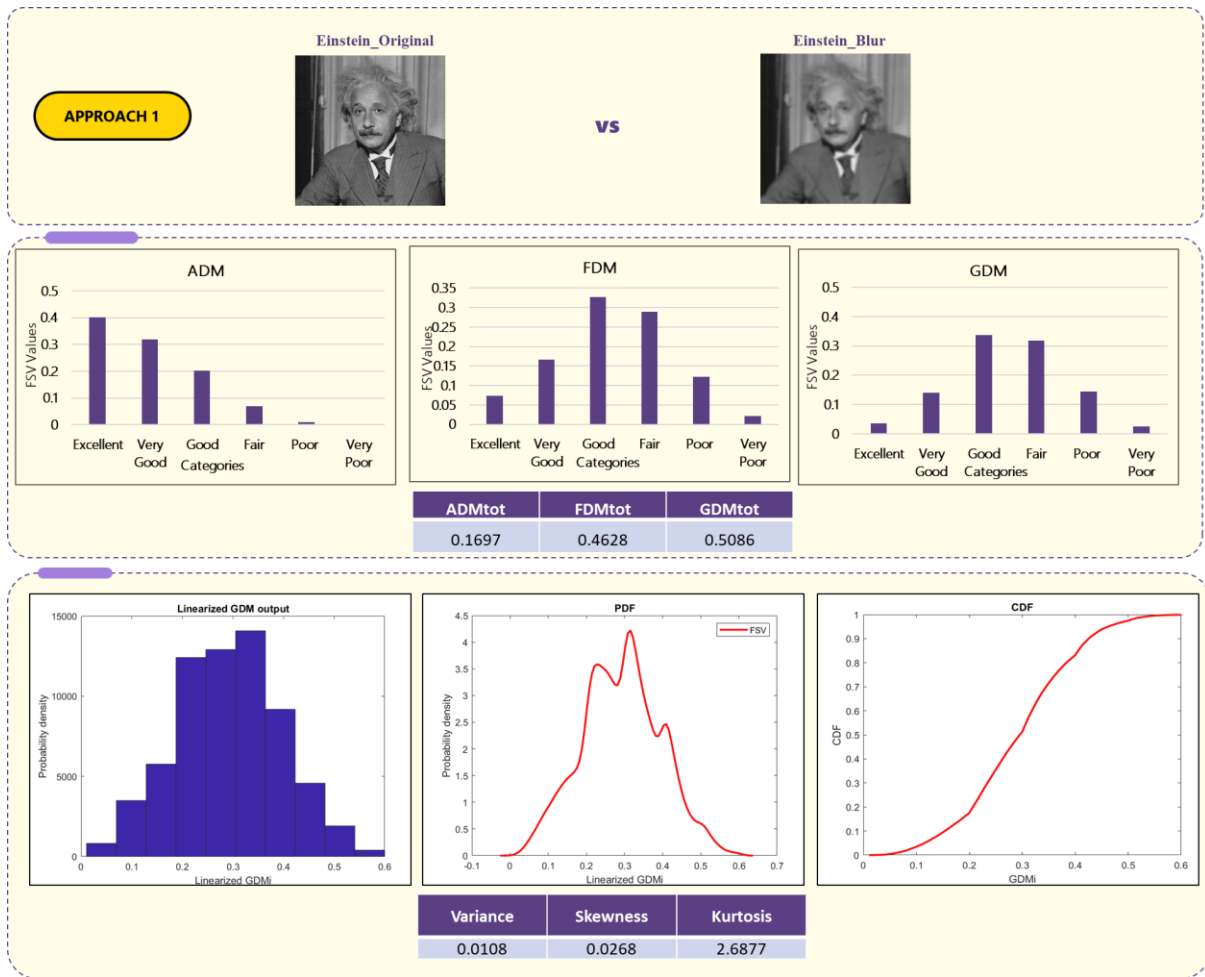


Figure 21: 2-D FSV Results of Einstein_Original vs Einstein_Blur using Approach 1

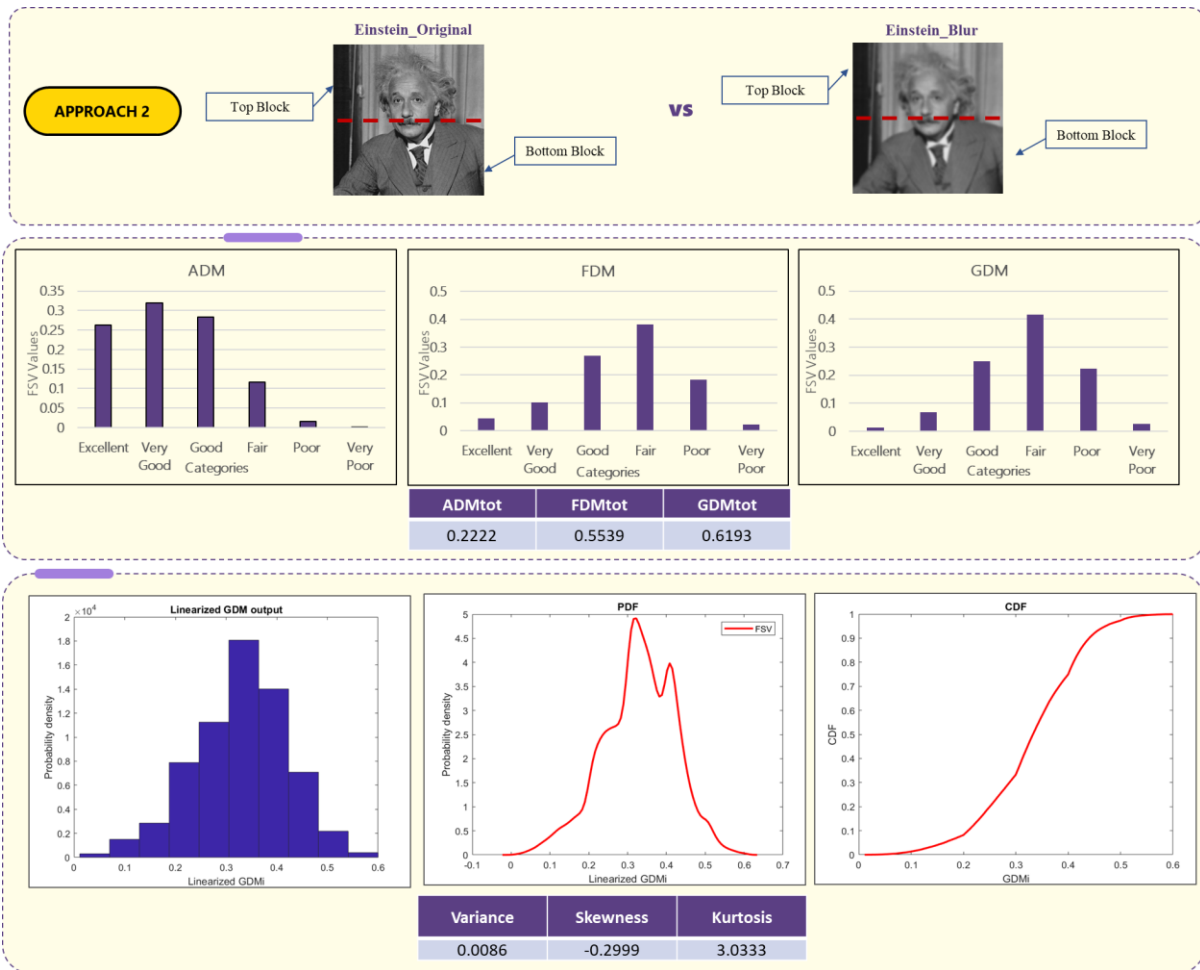


Figure 22: 2-D FSV Results of Einstein_Original vs Einstein_Blur using Approach 2

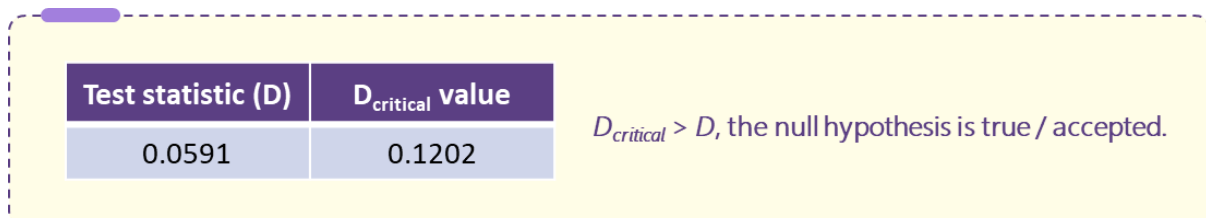


Figure 23: K-S Test results of Einstein_Original vs Einstein_Blur comparing Approach 1 and Approach 2

The results for comparison -2, which is Einstein_Original vs Einstein_Impulse is represented in Figure 24, 25, and 26.

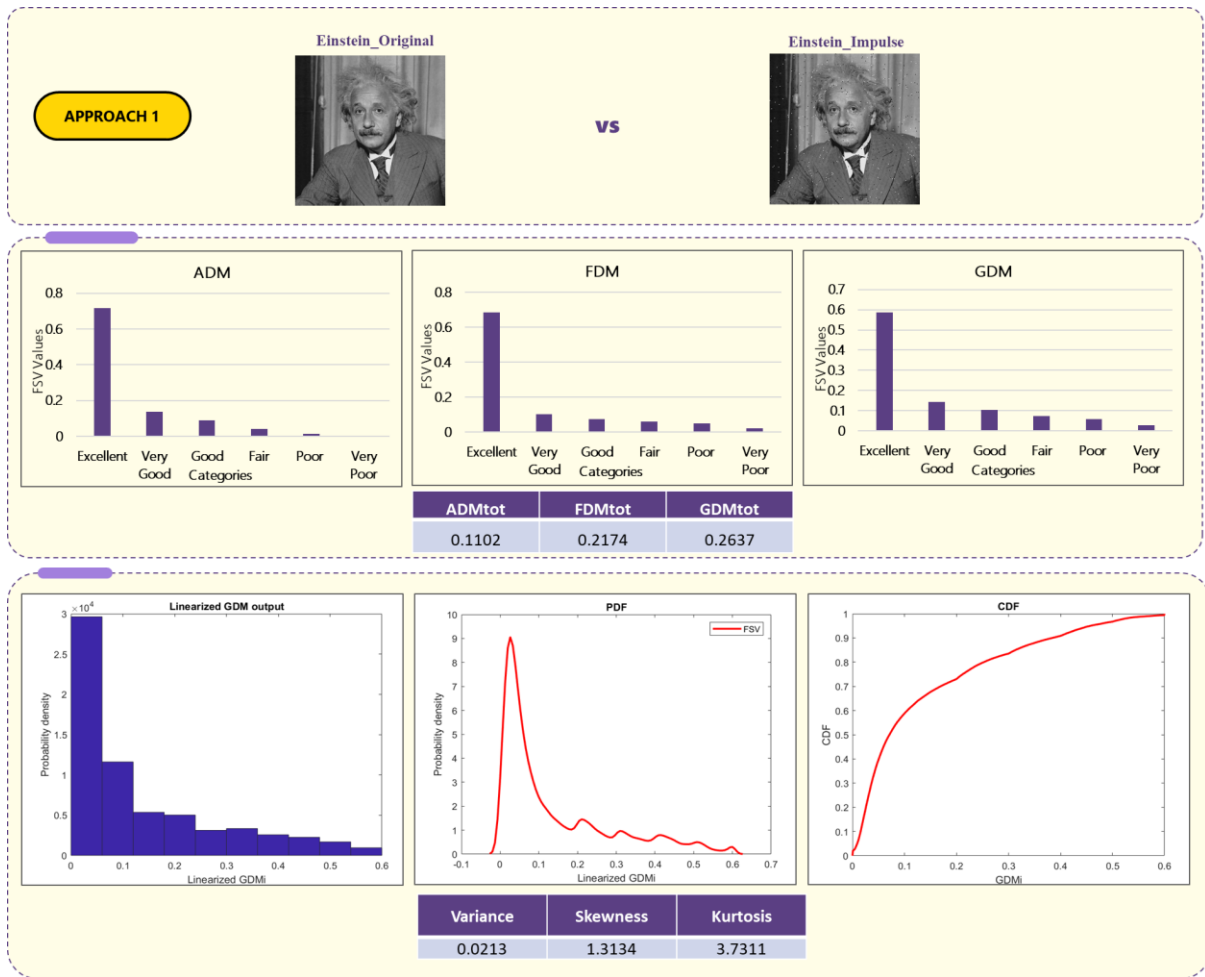


Figure 24: 2-D FSV Results of Einstein_Original vs Einstein_Impulse using Approach 1

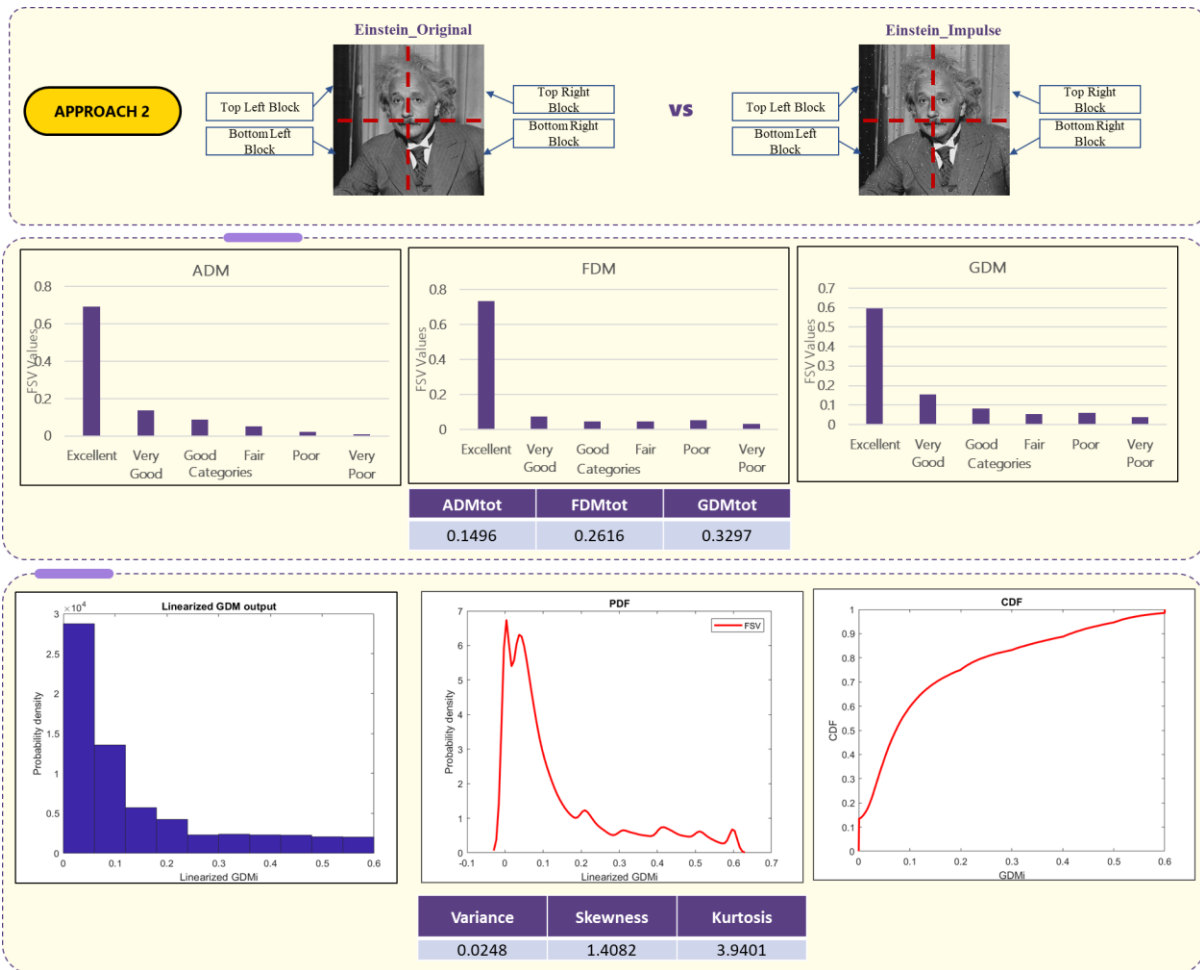


Figure 25: 2-D FSV Results of Einstein_Original vs Einstein_Impulse using Approach 2

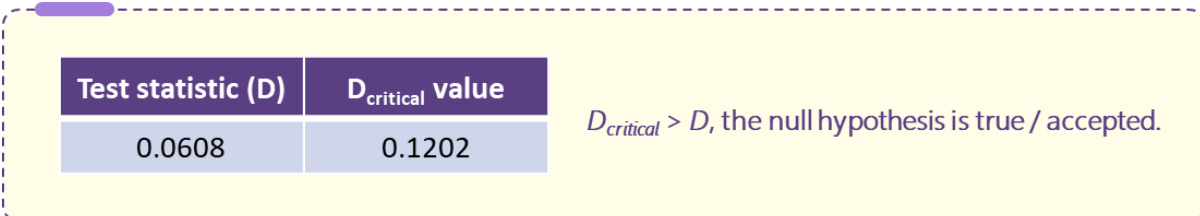


Figure 26: K-S Test results of Einstein_Original vs Einstein_Impulse comparing Approach 1 and Approach 2

5.3 OVERALL INFERENCE FOR STAGE -1 RESULTS:

In this section, the confidence histograms and density functions were analysed to compare the results from approach 1 with approach 2. From the analysis of 2-D FSV outputs of E-field and Einstein data, it is determined that there was no significant movement of data observed between the bin categories of the confidence histograms obtained from point-by-point FSV. Instead, the number of points in the confidence histograms was either moved towards the immediate left or right bin categories.

Moreover, it is observed that the density functions determined using the statistical moments and non-parametric tests gave a more detailed representation of the FSV outputs. The meta-KS-test results confirmed that the null hypothesis was true, indicating that the input images used for Approach 1 and Approach 2 for all the comparisons were from the same source.

The immediate movement of the data points in the confidence histogram approach is due to two major factors: one is the central difference scheme, and the other is the segmentation. Where the segmentation process enables the FSV process to focus and identify important details and features of the segmented blocks and effectively aids in generating granular comparison outputs, resulting in the confidence points falling to the immediate bins rather than falling to the extreme bins when compared with full-structure 2-D FSV results.

For example, in comparison -1 from Section 5.2.1 Table 3, the E2800_field(0) versus E2800_field(2) confidence histogram results showed that the maximum points in the 'Very Good' bin category were reduced, and the points were increased to the left into the "Excellent" bin category. Similarly, when results of Einstein_Original versus Einstein_Blur were compared in Section 5.2.7 Table 9, the maximum points in the "Good" bin category of the confidence histogram were reduced, and the points were increased to the right into 'Fair' bin category.

Based on all the observations above, it is evident that the input images used for 2-D FSV processing are sourced from the same dataset. This ensures that any differences in the confidence values of the data points are due to the segmentation approach and not to differences in the input images.

Secondly, the immediate movement of the data points in the confidence histograms further supports the validity of the proposed methodology using the segmented approach. This movement demonstrates that the methodology is effective in identifying and capturing the important details and features of the segmented blocks, resulting in more precise and granular FSV comparison results.

Finally, it is found that the confidence points in the segmented approach results are not moved to the extreme bins, which indicates that there are no radical faults in the research study.

Overall, these observations and analyses provide strong evidence for the reliability and accuracy of the methodology used in this research study. The approach used is appropriate for the proposed objective, and the results obtained are consistent and reliable.

5.4 STAGE - 2: APPLICATION AND RESULT ANALYSIS OF 2-D FSV ON IRREGULAR IMAGE DATA SETS

In sections 5.2 and 5.3, the 2-D FSV results of E-Field and Einstein from both Approach 1 and Approach 2 of the proposed methodology are compared, and the segmented approach is validated as a feasible approach that can be applied to irregular images with gaps or spaces in the image structure.

The detailed steps to carry out Approach 2 are discussed in Chapter 4, Figure 1.

5.4.1 FSV RESULT ANALYSIS FOR MICROSTRIP PATCH ANTENNA'S SURFACE CURRENT IMAGE DATA SETS:

In this section, the surface current output images of a Microstrip patch antenna designed to operate at frequencies of 5.2 GHz, 5.3 GHz, and 5.6 GHz are shown. These simulated surface current outputs are exported as images and used as input datasets for 2-D FSV processing, as shown in Section 5.1, Figure 3.

Figure 26 and Table 10 below demonstrate the comparison of Patch_Antenna_5.2 GHz and Patch_Antenna_5.3 GHz, Patch_Antenna_5.2 GHz, and Patch_Antenna_5.6 GHz, respectively.

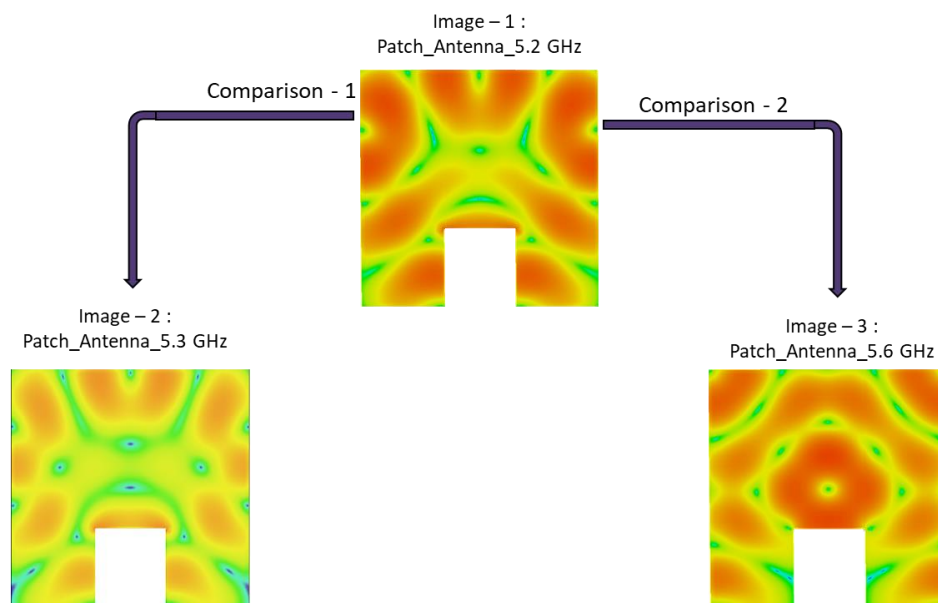


Figure 27: Different runs of Microstrip patch antenna surface current input datasets for 2-D FSV processing

Comparison Reference	Images compared for 2-D FSV Processing
Comparison - 1	Patch_Antenna_5.2 GHz vs Patch_Antenna_5.3 GHz
Comparison - 2	Patch_Antenna_5.2 GHz vs Patch_Antenna_5.6 GHz

Table 12: Comparison references of surface current output images from a microstrip patch antenna used as input datasets for 2-D FSV

In the second stage of the analysis, the segmented approach is used to analyse the irregular image datasets from Section 5.1, Figure 3, and Table 2, which were designed using Computer Simulation Technology (CST). The images had gaps or spaces, and these were identified in the bottom centre of the surface current outputs. Subsequently, the images were segmented vertically into three blocks: Block A, Block B, and Block C. The 2D FSV was then performed separately on each of the blocks, generating point-by-point confidence histograms. The blocks were then recombined to validate the results. Figure 23 below shows the outcomes of both the block-wise and recombined 2D FSV analyses.

Figure 27 presents the 2-D FSV results of the comparison between Patch_Antenna_5.2 GHz and Patch_Antenna_5.3 GHz. The figure demonstrates the vertical segmentation method used on the input datasets, and it displays the point-by-point 2-D FSV outcomes along with the mean values for each segmented block. It also shows the point-by-point 2D FSV outcomes when the segmented blocks are recombined.

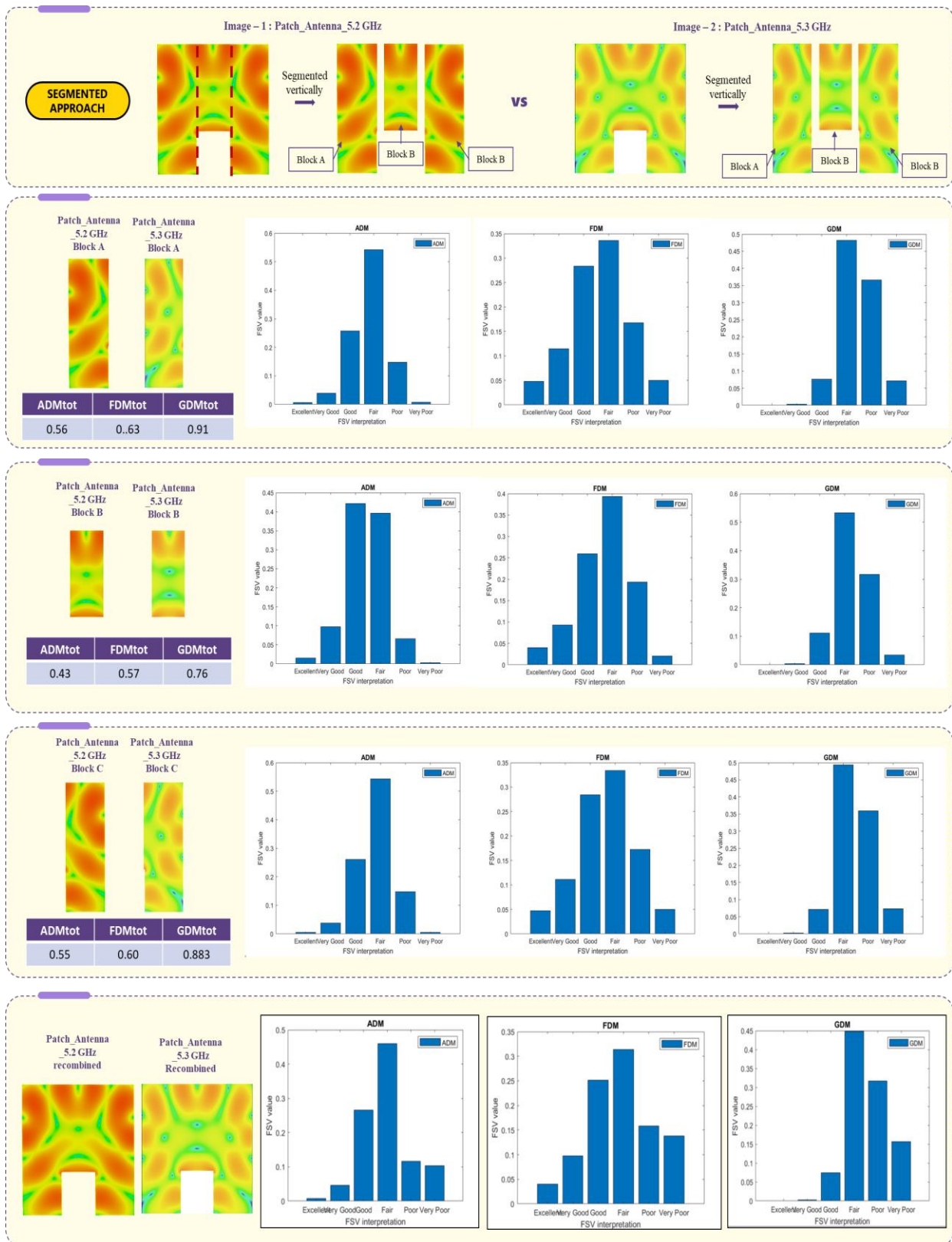


Figure 28: 2-D FSV Results of Patch_Antenna_5.2 GHz vs Patch_Antenna_5.3 GHz using Approach 2

Figure 28 presents the 2-D FSV results of the comparison between Patch_Antenna_5.2 GHz and Patch_Antenna_5.6 GHz.

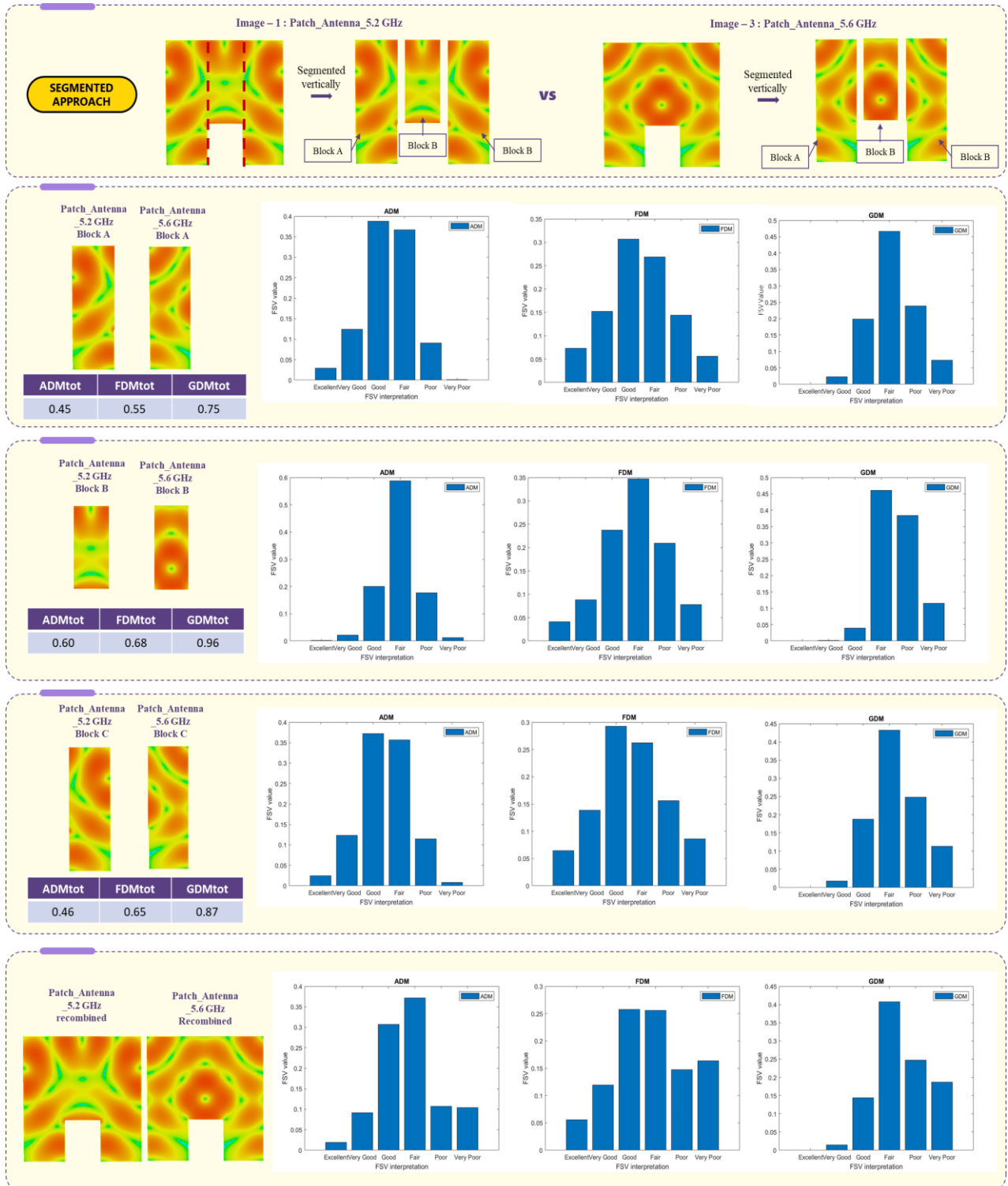


Figure 29: 2-D FSV Results of Patch_Antenna_5.2 GHz vs Patch_Antenna_5.6 GHz using Approach 2

5.4.2 OVERALL INFERENCE FOR STAGE -2 RESULTS

On analyzing the results from Figure 23 and 24, it can be observed that the segmentation approach can be successfully applied on irregular images with gaps or spaces in the image structure. Moreover, the segmentation and recombination approach method performed on the input images demonstrated negligible differences. This means that when the compared blocks were recombined, they produced similar outputs, which proves the robustness of the segmentation approach.

Therefore, the segmented approach developed in this study can be used to perform 2-D FSV on irregular image data sets without adversely impacting the accuracy of the results. This finding is significant because it indicates that the proposed methodology is a viable and effective solution for conducting 2D FSV on irregular data sets with spaces or gaps in the image structure.

In conclusion, this chapter presented an analysis and discussion of the results to demonstrate the feasibility and effectiveness of the proposed methodology for performing 2D FSV on regular and irregular image data sets. The inferences from stage 1 and 2 suggest that the segmented approach developed in this research study can be utilized to perform 2D FSV on irregular data with spaces or gaps within the image structure without compromising the accuracy of the results.

5.5 REFERENCES

- [1] G. Zhang, A. Duffy, A. Orlandi, D. Febo, L. Wang and H. Sasse, "Comparison of Data With Multiple Degrees of Freedom Utilizing the Feature Selective Validation Method", *IEEE Transactions on Electromagnetic Compatibility*, vol. 58, no. 3, pp. 784-791, 2016.
- [2] Z. Gang, A. Duffy, H. Sasse and W. Lixin, "The use of probability density functions to improve the interpretation of FSV results", *2012 IEEE International Symposium on Electromagnetic Compatibility*, 2012.
- [3] G. Zhang, H. Sasse, L. Wang, and A. Duffy, "A Statistical Assessment of the Performance of FSV", *ACES Journal*, vol. 28, no. 12, pp. 1179–1186, Sep. 2021#.
- [4] N. Smirnov, "Table for Estimating the Goodness of Fit of Empirical Distributions", 2022.
- [5] V. Berger and Y. Zhou, "Kolmogorov–Smirnov Test: Overview", *Wiley StatsRef: Statistics Reference Online*, 2014.

CHAPTER - 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In this research study, it was identified that performing 2-Dimensional Feature Selective Validation on irregular data which includes gaps or spaces within the image structure presented a significant challenge in the field of Computation Electromagnetics (CEM). To address this challenge, this study aimed to develop an approach that would allow for 2-D FSV on such data sets and the primary objective was identified to validate the feasibility of this approach and demonstrate its potential for practical applications in the field of CEM.

To achieve this objective, a proposed method with two approaches to perform 2-Dimensional FSV was developed in Chapter 4. . The first approach performs normal 2-D FSV on the original full structure of the input image. Whereas in the second approach, the original image is segmented into smaller blocks, and 2-D FSV is performed on each blocks and then concatenated to the original structure for results comparison.

To validate the effectiveness of developed proposed methodology, the approaches were applied on regular-shaped images such as E-Field and Einstein images in Chapter 5. The results demonstrated that the confidence histograms showed only minor shifts in the data points, and any differences in the confidence values were attributed to the more precise and granular FSV comparison results in the segmentation approach. To further validate the effectiveness of the proposed approaches, Statistical moments and non-parametric tests were performed to further validate the FSV outputs, and a meta Kolmogorov-Smirnov (K-S) Test was conducted, which confirmed that the input images used for the proposed methodology for all the comparisons were from the same source.

Therefore, these results confirmed that the proposed segmentation approach is feasible for irregular image datasets with gaps or spaces, fulfilling the primary objective goal of the study. It also indicated that the segmentation and recombination process does not unduly affect the FSV results, making the proposed methodology a reliable approach for performing 2-D FSV on irregular image structures.

In Chapter 5, the second stage of the objective was accomplished by implementing the segmented approach on an antenna designed to have gaps or spaces within its device structure. This was done to demonstrate that the proposed methodology could be used to address real-world problems in the field of electromagnetic (EM). The FSV results were analysed and observed that the segmentation approach was effective on irregular images with gaps or spaces

in the image structure as the point-by-point confidence histogram output data demonstrated only a negligible differences.

In conclusion, the research study successfully developed and validated an approach to perform 2-Dimensional FSV on data with irregularities such as gaps or spaces within the image structure. The results demonstrated the feasibility and effectiveness of the proposed methodology, which has practical applications in the field of EM components and systems.

However, the study had some limitations, including the conversion process used in Chapter 4, section 4.2. The process involved converting RGB colour scale input images into monochromatic grayscale images, which was necessary in reductions of the computational efforts and overhead due to repeating FSV steps for each color array, followed by reforming the data using a weighted scheme. This is a potential area for future research to explore in developing a methodology in performing 2-D FSV for separate arrays of Red, Blue, and Green pixel values and formulate a weighted scheme approach for results analysis.

Another limitation is this research study is the proposed methodology developed using different programming and toolsets to demonstrate the proof of concept. Though a comprehensive analysis and validation of the proposed approach is concluded; a single programming toolset solution will substantially increase the processing time and widen its applications in EMC.

The study's limitations should be considered into account for future work in this research area. Overall, the proposed approach presents a promising and a robust solution for analysing irregular image data sets.

6.2 FUTURE WORK

Future research should focus on applying the segmented approach developed in this research study to perform 2-D FSV on irregular data with spaces or gaps within the image structure, such as surface currents outputs for an Ultra High Frequency (UHF) Radio Frequency Identification (RFID) tag [1], as shown in Chapter 4 Figure 12. Also, this approach can be applied on the surface current outputs of a 5G mobile antenna in Figure 1 and a rectangular monopole antenna with CPW feed in Figure 2 [2] [3].

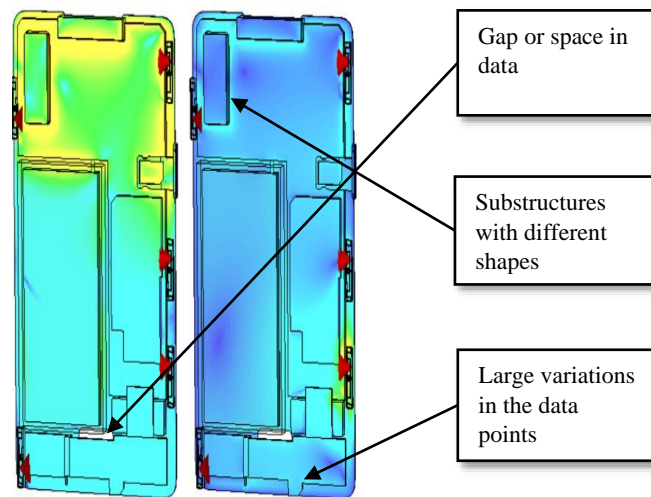


Figure 1: Surface currents distribution on 5G mobile antenna [2]

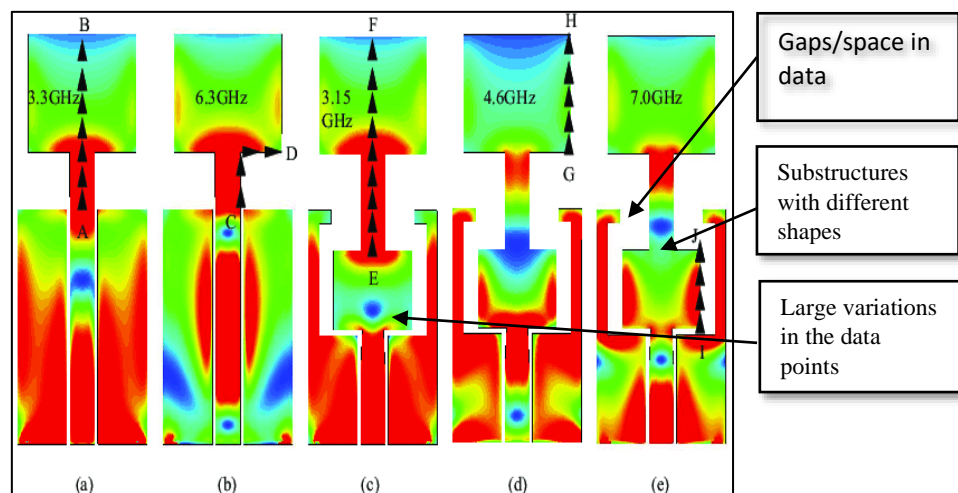


Figure 2: Surface currents plots on rectangular monopole antenna with CPW feed [3]

As shown in the Figure 1 and 2 above, it is identified that to perform 2-D FSV in EMC device, it is also important to consider the large variation in data points and different sub structures within the image. This can be further studied here.

To apply the segmented approach to an EMC device structure with a gap or space, the images should be segmented into several vertical and horizontal blocks, and 2- D FSV should be performed for each block, and subsequent recombination can be performed to calculate histograms, PDF, CDF, and statistical moments (Mean, Variance, Skewness, and Kurtosis) to evaluate FSV performance. However, the statistical moments derived from the preceding results need be studied in order to comprehend and infer the statistical distribution properties of the input data.

The next phase of research should focus on developing an approach to perform 2-D FSV on RGB images, which was noted as a challenge while developing the image conversion steps in the proposed method in chapter 4 section 4.2. When an RGB image is turned into an array of pixel values, each pixel value of the array has sequential values of red, green, and blue, as discussed in the literature in chapter 1. As a result, the approach to be developed should perform 2-D FSV for each colour pixel array independently by performing repeated 1-D FSV for x and y direction each colour pixels array. Then, these arrays must be recombined using weighted scheme to evaluate FSV performance from the calculated histograms, PDF, CDF, and statistical moment (Mean, Variance, Skewness, and Kurtosis).

While there can be a reasonable level of confidence that 2-D segmentation will work with structures like those above, one of the major challenges will be about the interpretation and use of the numerical data obtained from the comparison in the context of that device or structure.

Another 2-D challenge will be to deal with curved surfaces. It is likely that segmentation and then conformally mapping the curved surface to a linear one, performing the comparison and conforming back, might be the most sensible approach. This is a topic of future research

The next significant challenge will be sectorization, that is taking an irregular 3-D data-set and sectoring it into a number of regular sub-volumes for comparison and subsequent recombination. Further, while the concept is straightforward to do something similar to this to greater numbers of degrees of freedom, the method of splitting up the data into a number of contiguous 4D blocks is particularly challenging. Should that be solvable, higher numbers of degrees of freedom should be possible by extension.

6.3 REFERENCES

- [1] M. Ziai and J. Batchelor, "Temporary On-Skin Passive UHF RFID Transfer Tag", *IEEE Transactions on Antennas and Propagation*, vol. 59, no. 10, pp. 3565-3571, 2011.
- [2] M. Rutschlin, "5G Antenna Design for Mobile Phones | The SIMULIA Blog", *The SIMULIA Blog*, 2020.
- [3] M. Gopikrishna, D. Krishna and C. Aanandan, "A Compact Rectangular Monopole Antenna Design with a Novel Feed for an Improved UWB Performance", *Radioengineering*, vol. 27, no. 1, pp. 63-69, 2018.

APPENDIX

APPENDIX 1

1- DIMENSIONAL FSV IMPLEMENTATION

```

close all;
clc;
% importing input data 1 and 2
x1 = importdata('chart2a_new.txt');
x2 = importdata('chart2b_new.txt');
% to select the max length from both the inputs
maxlength1=max(size(x1));
maxlength2=max(size(x2));
if min(size(x1))==1
    ax=(1:maxlength1)';
    ay=x1(1:maxlength1);
    bx=(1:maxlength2)';
    by=x2(1:maxlength2);
else
    ax=x1(1:maxlength1,1);
    ay=x1(1:maxlength1,2);
    bx=x2(1:maxlength2,1);
    by=x2(1:maxlength2,2);
end
maxlength=max(size(x1)); % assigning the length of the input data
t=1:maxlength; %Initialisation of time (x-axis)
Halflength = fix(maxlength/2); % assigning the Half length of the input data
% to plot the input data 1 and 2
figure(1)
hold all;
plot(x1(:,1))
plot(x2(:,1),'r')
xlabel('samples');
ylabel('Amplitude');
title('Input data sets')
grid on;

```

```
set(gcf,'color','white')
```

```
⊘*****FFT*****⊘
```

```
y1=fft(x1); % to perform FFT on input data 1
y2=fft(x2); % to perform FFT on input data 2
iy1=abs(y1); % to determine the magnitude of the FFT signal 1
f=(0:length(y1)-1)*fs/length(y1);
```

```
iy2=abs(y2); % to determine the magnitude of the FFT signal 2
f=(0:length(y1)-1)*fs/length(y1);
```

```
% to plot the FFT outputs for data 1 and 2
```

```
figure(2)
hold all;
plot(iy1)
plot(iy2,'r')
xlabel('frequency');
ylabel('Magnitude');
title('Fourier transformed data');
grid on;
set(gcf,'color','white');
```

```
⊘*****DC*****⊘
```

```
d=5; % initializing DC value
dc=zeros(maxlength,1); % zero padding the Max length of the data
% Dc remains '1' for the specified interval & 0 for other data
for i=1:d
    dc(i)=1;
end
for i=(maxlength-d+2):maxlength % end point of the data is specified
    dc(i)=1;
```

```

end
% To obtain DC in the transformed domain
dc1 = y1(1:maxlength).*dc;
dc2 = y2(1:maxlength).*dc;
% to plot the DC outputs in the transformed domain for data 1 and 2
figure(3)
hold all;
plot(abs(dc1))
plot(abs(dc2),'r')
xlabel('frequency');
ylabel('Magnitude');
title('DC points in transformed domain')
grid on;
set(gcf,'color','white');

% *****CUT-OFF & BREAKPOINTS*****%

sum=0;    % to store the sum of values of the independent variable
% initialising arrays for sum
sum_1 = zeros(Halflength,1);
sum_2 = zeros(Halflength,1);
totala=0; % to store the values of total 1
totalb=0; % to store the values of total 2
n1=d+1;   % getting the 5th data point
n2=d+1;

% Independent variables are summed for the specified length
for i=d+1:Halflength
    totala = totala + iy1(i) + iy1(maxlength-i);
    sum_1(i) = totala;
    totalb = totalb + iy2(i) + iy2(maxlength-i);
    sum_2(i) = totalb;
end
% the cut off location is specified
cut_off_factor = 0.4;

```

% the cut off is verified

```
switch_1 = sum_1 <= totala*cut_off_factor;
switch_2 = sum_2 <= totalb*cut_off_factor;
```

```
selected_1 = sum_1(switch_1);
if (zeros(size(selected_1)) == selected_1)
    n1 = 0;
else
    n1 = find(max(selected_1));
end
```

```
selected_2 = sum_2(switch_2)
if (zeros(size(selected_2)) == selected_2)
    n2 = 0;
else
    n2 = find(max(selected_2));
end
```

% to obtain the breakpoint

```
if n1>=n2
    n1=n2;
else
    n2=n1;
end
```

```
n1=n1+5;
n2=n2+5;
```

⊘*****Windows*****⊘

```
L1=zeros(maxlength,1);           %Initializing Low window for data 1
H1=zeros(maxlength,1);           %Initializing High window for data 1
L2=zeros(maxlength,1);           %Initializing Low window for data 2
H2=zeros(maxlength,1);           %Initializing High window for data 2
```

% to ensure negative and positive frequencies get treated the same way, mirroring is done here

```
for i=d+1:n1-3
    L1(i)=1;
    L1(maxlength-i)=1;
end
```

```
for i=n1+3:Halflength+1
    H1(i)=1;
    H1(maxlength-i)=1;
end
```

% positive and negative frequencies are fixed at the same time

```
L1(n1-2)=0.834;
L1(maxlength-(n1-2))=0.834;
L1(n1-1)=0.6667;
L1(maxlength-(n1-1))=0.6667;
L1(n1)=0.5;
L1(maxlength-n1)=0.5;
L1(n1+1)=0.3333;
L1(maxlength-(n1+1))=0.3333;
L1(n1+2)=0.166;
L1(maxlength-(n1+2))=0.166;

H1(n1-2)=0.166;
H1(maxlength-(n1-2))=0.166;
H1(n1-1)=0.3333;
H1(maxlength-(n1-1))=0.3333;
H1(n1)=0.5;
H1(maxlength-n1)=0.5;
H1(n1+1)=0.6667;
H1(maxlength-(n1+1))=0.6667;
H1(n1+2)=0.834;
H1(maxlength-(n1+2))=0.834;
```

```
for i=d+1:n2-3
```



```
L2(i)=1;
L2(maxlength-i)=1;
end
for i=n2+3:Halflength+1
    H2(i)=1;
    H2(maxlength-i)=1;
end

L2(n2-2)=0.834;
L2(maxlength-(n2-2))=0.834;
L2(n2-1)=0.6667;
L2(maxlength-(n2-1))=0.6667;
L2(n2)=0.5;
L2(maxlength-(n2))=0.5;
L2(n2+1)=0.3333;
L2(maxlength-(n2+1))=0.3333;
L2(n2+2)=0.166;
L2(maxlength-(n2+2))=0.166;

H2(n2-2)=0.166;
H2(maxlength-(n2-2))=0.166;
H2(n2-1)=0.3333;
H2(maxlength-(n2-1))=0.3333;
H2(n2)=0.5;
H2(maxlength-(n2))=0.5;
H2(n2+1)=0.6667;
H2(maxlength-(n2+1))=0.6667;
H2(n2+2)=0.834;
H2(maxlength-(n2+2))=0.834;

high1=zeros(maxlength,1);
high2=zeros(maxlength,1);
low1=zeros(maxlength,1);
low2=zeros(maxlength,1);
```

```

for i = 1:maxlength
    high1(i)=H1(i);
    high2(i)=H2(i);
    low1(i)=L1(i);
    low2(i)=L2(i);
end

% to plot the Low and high windows for data 1 and 2
figure(4)
hold all;
plot(low1)
plot(high1,'r')
xlabel('Samples');
ylabel('Filter value')
title('Low and High Windows for Data 1')
grid on;
set(gcf,'color','white');

figure(5)
hold all;
plot(low2)
plot(high2,'r')
xlabel('Samples');
ylabel('Filter value');
title('Low and High Windows for Data 2')
grid on;
set(gcf,'color','white');

%to obtain low and high filters in the transformed domain
highfilter1 = y1.*high1
highfilter2 = y2.*high2;
lowfilter1 = y1.*low1;
lowfilter2 = y2.*low2;

%to plot the low and high filters

```

```

figure(6)
hold all;
plot(abs(Highfilter1))
plot(abs(Highfilter2),'r')
% title('highfilter1')
xlabel('Frequency')
ylabel('Magnitude')
title('high filter');
set(gcf,'color','white');
grid on;

```

```

figure(7)
hold all;
plot(abs(Lowfilter1))
plot(abs(Lowfilter2),'r')
xlabel('Frequency')
ylabel('Magnitude')
% title('lowfilter1')
title('low filter');
set(gcf,'color','white');
grid on;

```

⊘ * * * * * I F F T * * * * * ⊘

```

ihigh1=ifft(highfilter1);    % to perform IFFT on High 1
ilow1=ifft(lowfilter1);     % to perform IFFT on Low 1
idc1=ifft(dc1);            % to perform IFFT on DC 1
ihigh2=ifft(highfilter2);  % to perform IFFT on High 2
ilow2=ifft(lowfilter2);    % to perform IFFT on Low 2
idc2=ifft(dc2);           % to perform IFFT on DC 2
%to plot the inverse transformed DC filter data

```

```

figure(8)
hold all;
plot(t,idc1)

```

```

plot(t,dc2,'r')
xlabel('Samples')
ylabel('Amplitude')
title('Inverse transformed DC');
grid on;
set(gcf,'color','white');
%to plot the inverse transformed low filter data
figure(9)
hold all;
plot(t,ilow1)
plot(t,ilow2,'r')
xlabel('Samples')
ylabel('Amplitude')
title('ilow1')
title('Inverse transformed Low filter');
grid on;
set(gcf,'color','white');
%to plot the inverse transformed high filter data
figure(10)
hold all;
plot(t,ihigh1);
plot(t,ihigh2,'r');
xlabel('Samples')
ylabel('Amplitude')
title('Inverse transformed high filter')
grid on;
set(gcf,'color','white');
⊘*****ADM*****⊘

alfa=zeros(1,maxlength);
bita=0;
axs=zeros(1,maxlength);
delta=0;
for i=1:maxlength

```

```

alfa(i)=abs(ilow1(i))-abs(ilow2(i));
bita=bita+(abs(ilow1(i))+abs(ilow2(i)));
axs(i)=abs(idc1(i))-abs(idc2(i));
delta=delta+(abs(idc1(i))+abs(idc2(i)));
end
bita=bita/maxlength;
delta=delta/maxlength;
cc=abs(axs./delta);
fraction=(alfa./bita);
ADM=abs(fraction)+cc.*exp(cc);

```

% to determine ADM total

```

ADM1=abs(fraction);
ADM2=cc.*exp(cc);
TADM=0;
TADM1=0;
TADM2 = 0;

```

```

for i=1:maxlength

```

```

    TADM=TADM+ADM(i);
    TADM1=TADM1+ADM1(i);
    TADM2=TADM2+ADM2(i);

```

```

end

```

```

TADM=TADM/maxlength;
TADM1=TADM1/maxlength;
TADM2=TADM2/maxlength;

```

%to plot the ADM output

```

figure(11)
plot(time,ADM);
% plot(ADM);
hold on;
grid on;

```

```

xlabel('Samples');
ylabel('FSV values');
title('ADMi')
axis tight;
set(gcf,'color','white');

%*****FDM*****%
% Initialising the First derivatives of inverse transformed Low & High data
d1ilow1 = zeros(maxlength,1);
d1ilow2 = zeros(maxlength,1);
d1ihigh1 = zeros(maxlength,1);
d1ihigh2 = zeros(maxlength,1);
% Initialising the second derivative of inverse transformed High data
d2ihigh1 = zeros(maxlength,1);
d2ihigh2 = zeros(maxlength,1);

% to obtain the First derivatives for specified interval
for i = 3:maxlength-2
    d1ilow1(i) = ilow1(i+2) - ilow1(i-2);
    d1ilow2(i) = ilow2(i+2) - ilow2(i-2);
    d1ihigh1(i) = ihigh1(i+2) - ihigh1(i-2);
    d1ihigh2(i) = ihigh2(i+2) - ihigh2(i-2);
end
% to obtain the First derivatives for specified interval
for i = 4:maxlength-3
    d2ihigh1(i) = d1ihigh1(i+3) - d1ihigh1(i-3);
    d2ihigh2(i) = d1ihigh2(i+3) - d1ihigh2(i-3);
end
% to calculate the denominator
F1=0;
F2=0;
F3=0;
for i=1:maxlength

```

```

F1=(abs(d1ilow1(i))+abs(d1ilow2(i)))+F1;
F2=(abs(d1ihigh1(i))+abs(d1ihigh2(i)))+F2;
F3=(abs(d1ihigh1(i))+abs(d2ihigh2(i)))+F3;
end
% to calculate the Numerator
FDM1=zeros(maxlength,1);
FDM2=zeros(maxlength,1);
FDM3=zeros(maxlength,1);
for i=1:maxlength
    FDM1(i)=(abs(d1ilow1(i))-abs(d1ilow2(i)))/(F1*2/maxlength);
    FDM2(i)=(abs(d1ihigh1(i))-abs(d1ihigh2(i)))/(F2*6/maxlength);
    FDM3(i)=(abs(d1ihigh1(i))-abs(d2ihigh2(i)))/(F3*7.2/maxlength);
end
FDM=2*abs(FDM1+FDM2+FDM3);
% to determine FDM total
TFDM = 0;
for i=1:maxlength
    TFDM=TFDM+FDM(i);
end
TFDM=TFDM/maxlength;
%to plot the FDM output
figure(12)
plot(time,FDM);
hold on
grid on
xlabel('Samples');
ylabel('FSV values');
title('FDMi')
axis tight;
set(gcf,'color','white');
%*****GDM*****%
GDM=zeros(maxlength,1);
for i=1:maxlength

```

```

GDM(i)=sqrt(ADM(i)*ADM(i)+FDM(i)*FDM(i));
end
% to determine GDM total
TGDM = 0;
for i=1:maxlength
    TGDM=TGDM+GDM(i);
end
TGDM=TGDM/maxlength;
%to plot the GDM output
figure(13)
plot(time,GDM)
hold on;
grid on;
xlabel('Samples');
ylabel('FSV values');
title('GDMi')
axis tight;
set(gcf,'color','white');
% ***** HISTOGRAMS ***** %
% initializing counters to store the FSV values
ADMScale=zeros(6,1);
FDMScale=zeros(6,1);
GDMScale=zeros(6,1);
% FSV interpretation results for ADM, FDM and GDM
for i=1:maxlength
    if ADM(i)<=0.1
        ADMScale(1)=ADMScale(1)+1;
    elseif ADM(i)>0.1 && ADM(i)<=0.2
        ADMScale(2)=ADMScale(2)+1;
    elseif ADM(i)>0.2 && ADM(i)<=0.4
        ADMScale(3)=ADMScale(3)+1;
    elseif ADM(i)>0.4 && ADM(i)<=0.8
        ADMScale(4)=ADMScale(4)+1;

```



```
elseif ADM(i)>0.8 && ADM(i)<=1.6
    ADMScale(5)=ADMScale(5)+1;
else
    ADMScale(6)=ADMScale(6)+1;
end

if FDM(i)<=0.1
    FDMScale(1)=FDMScale(1)+1;
elseif FDM(i)>0.1 && FDM(i)<=0.2
    FDMScale(2)=FDMScale(2)+1;
elseif FDM(i)>0.2 && FDM(i)<=0.4
    FDMScale(3)=FDMScale(3)+1;
elseif FDM(i)>0.4 && FDM(i)<=0.8
    FDMScale(4)=FDMScale(4)+1;
elseif FDM(i)>0.8 && FDM(i)<=1.6
    FDMScale(5)=FDMScale(5)+1;
else
    FDMScale(6)=FDMScale(6)+1;
end

if GDM(i)<=0.1
    GDMScale(1)=GDMScale(1)+1;
elseif GDM(i)>0.1 && GDM(i)<=0.2
    GDMScale(2)=GDMScale(2)+1;
elseif GDM(i)>0.2 && GDM(i)<=0.4
    GDMScale(3)=GDMScale(3)+1;
elseif GDM(i)>0.4 && GDM(i)<=0.8
    GDMScale(4)=GDMScale(4)+1;
elseif GDM(i)>0.8 && GDM(i)<=1.6
    GDMScale(5)=GDMScale(5)+1;
else
    GDMScale(6)=GDMScale(6)+1;
end

end
```

% to plot the histogram outputs

```
figure(14)
bar(ADMScale/length(x1));
set(gca,'XTicklabel',{'Excellent','Very Good','Good','Fair','Poor','Very Poor'});
xlabel('Categories');
ylabel('Probability')
title('ADMc');
set(gcf,'color','white')
```

```
figure(15)
bar(FDMScale/length(x1));
set(gca,'XTicklabel',{'Excellent','Very Good','Good','Fair','Poor','Very Poor'});
xlabel('Categories');
ylabel('Probability')
title('FDMc');
set(gcf,'color','white')
```

```
figure(16)
bar(GDMScale/length(x1));
set(gca,'XTicklabel',{'Excellent','Very Good','Good','Fair','Poor','Very Poor'});
xlabel('Categories');
ylabel('Probability')
title('GDMc');
set(gcf,'color','white')
```

APPENDIX 2

IMAGE CONVERSION USING PYTHON

```
# import PIL, NumPy and CSV libraries

from PIL import Image

import numpy as np

import csv

# import tkinter for user file selection to save and open a file from a file path

from tkinter import Tk

from tkinter.filedialog import askopenfilename

from tkinter.filedialog import asksaveasfilename

# import pathlib to save file name

from pathlib import Path

# steps for dynamic user file selection

Tk().withdraw()

inputImagePath = askopenfilename()

# verify input image name by extracting the file name from the file path. This file name can be
also used for writing file names.

inputImageName = Path(inputImagePath).stem

# Print to verify the input file name

print (inputImageName)

# store the selected image to a variable using Image.Open() function

inputImage = Image.open(inputImagePath)

# Convert RGB image to Greyscale image

greyScaleInputImage = inputImage.convert('L')
```

```
# Save input image as NumPy array

inputImageNumpyArray = np.array(greyScaleInputImage)

# save the image Greyscale numpy array as a CSV to a user selected file path

fotypes = [('CSV files', '*.csv')]

with open(asksaveasfilename(filetypes=fotypes, initialfile = "Einstein_GreyScaleArray", title =
>Select the path to save the outputs", defaultextension=".csv"), 'w', newline='') as csvfile:

    writer = csv.writer(csvfile, delimiter=',')

    writer.writerows(inputImageNumpyArray)

# Validate the image mode to verify after image conversion

print(greyScaleInputImage.mode)

#Validate the number of image pixels before storing to NumPy array

width, height = inputImage.size

print ("Dimensions:", inputImage.size, "Total pixels:", width * height)

#Validate the NumPy array size to compare with the input image pixel count

print(inputImageNumpyArray.shape)

# Reform NumPy array to input image and visually validate

im = Image.fromarray(inputImageNumpyArray)

im.show()
```

APPENDIX 3

VBA MACRO FUNCTION TO EXPORT EACH ROW FROM SOURCE SHEET TO INDIVIDUAL SHEETS

Sub copyRows()

'VBA Macro function to copy each row from the source sheet and transpose paste to new sheets in csv format(destination).

'For example: if the source sheet has 256 rows; each row will be copied, transposed and pasted to 256 new sheets

'Each new sheet will be named by source sheet original name + suffixing the count of row copied.

' Variables defined

Dim noOfRows

Dim i

Dim fileName

Dim filePath

' User to update the number of rows in the source sheet

noOfRows = 104

' User to update the desired file name of the destination sheets

fileName = "<filename>"

'User to update the file path of the destination sheets to be saved

filePath = "<filepath>"

' For loop to iterate the copy and paste operations

For i = 1 To noOfRows

' Loop starts from the selecting 1st cell of the row and increments to noOfRows

Range("A" & i).Select

' Select the full row of the cell selected in the above step

Range(Selection, Selection.End(xlToRight)).Select

' Copy the row selected in the above step

Selection.Copy

*' Steps to create individual destination sheet and transpose paste the selection
from above step*

Workbooks.Add

*Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:=
_ False, Transpose:=True*

' Clear the clipboard with copied row from the source sheet

Application.CutCopyMode = False

*' Save the workbook in .csv format with user-fed filename to the user-fed
filepath*

ChDir _filePath

*ActiveWorkbook.SaveAs fileName:=fileName & i & ".csv" _
, FileFormat:=xlCSV, CreateBackup:=False*

' Close the active workbook saved

ActiveWindow.Close

' loop the steps till noOfRows value

Next i

End Sub

APPENDIX 4

VBA MACRO FUNCTION TO EXPORT EACH COLUMN FROM SOURCE SHEET TO INDIVIDUAL SHEETS

Sub copyColumns()

'VBA Macro function to copy each column from the source sheet and paste to new sheets in csv format(destination).

'For example: if the source sheet has 256 columns; each column will be copied, and pasted to 256 new sheets

'Each new sheet will be named by source sheet original name + suffixing the count of column copied.

' Variables defined

Dim noOfColumn

Dim i

Dim fileName

Dim filePath

' User to update the number of columns in the source sheet (count should always be total count -1 as the column index starts from zero)

noOfColumns = 107

' User to update the desired file name of the destination sheets

fileName = "<filename>"

'User to update the file path of the destination sheets to be saved

filePath = "<filepath>"

' For loop to iterate the copy and paste operations

For i = 0 To noOfColumns

' Loop starts from the selecting 1st cell of the column and increments to noOfColumns

Range("A1").Select

ActiveCell.Offset(0, i).Select

' Select the full column of the cell selected in the above step

Range(Selection, Selection.End(xlDown)).Select

' Copy the column selected in the above step

Selection.Copy

' Steps to create individual destination sheet and paste the selection from above step

Workbooks.Add

```
Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _  
False, Transpose:=False  
Application.CutCopyMode = False  
' Save the workbook in .csv format with user-fed filename to the user-fed filepath  
ChDir _  
filePath  
ActiveWorkbook.SaveAs fileName:=fileName & i + 1 & ".csv" _  
, FileFormat:=xlCSV, CreateBackup:=False  
' Close the active workbook saved  
ActiveWindow.Close  
' loop the steps till noOfColumns value  
Next i  
End Sub
```


APPENDIX 5

REPEATED 1-D FSV (1-D FSV WITH A FOR LOOP)

```

close all;
clc;
% No of rows/column in a data set
x = < specify number of row or columns>;
% User to fill the filename and its path for input data sets
fileName1 = "specify file name";
filePath1 = "specify file path here";
fileName2 = "specify file name ";
filePath2 = "specify file path here";
% to iterate the process with respect to the number of rows/columns specified in x above
for l = 1:x
%to import and increment input data files
x1 = importdata(filePath1 + fileName1 + l + '.csv');
x2 = importdata(filePath2 + fileName2 + l + '.csv');
if min(size(x1))==1
    ax=(1:maxlength1)';
    ay=x1(1:maxlength1);
    bx=(1:maxlength2)';
    by=x2(1:maxlength2);
else
    ax=x1(1:maxlength1,1);
    ay=x1(1:maxlength1,2);
    bx=x2(1:maxlength2,1);
    by=x2(1:maxlength2,2);
end
maxlength=max(size(x1)); % assigning the length of the input data
t=1:maxlength; %Initialisation of time (x-axis)
Halflength = fix(maxlength/2); % assigning the Half length of the input data
kt=1;
fs=1/kt;

```

```

time=t;

%*****FFT*****%
y1=fft(x1);    % to perform FFT on input data 1
y2=fft(x2);    % to perform FFT on input data 2
iy1=abs(y1);   % to determine the magnitude of the FFT signal 1
f=(0:length(y1)-1)*fs/length(y1);

iy2=abs(y2);   % to determine the magnitude of the FFT signal 2
f=(0:length(y1)-1)*fs/length(y1);

%*****DC*****%
d=5; % initializing DC value
dc=zeros(maxlength,1); % zero padding the Max length of the data
% Dc remains '1' for the specified interval & 0 for other data
for i=1:d
    dc(i)=1;
end
for i=(maxlength-d+2):maxlength % end point of the data is specified
    dc(i)=1;
end
% To obtain DC in the transformed domain
dc1 = y1(1:maxlength).*dc;
dc2 = y2(1:maxlength).*dc;
%*****CUT-OFF & BREAKPOINTS*****%

sum=0;    % to store the sum of values of the independent variable
totala=0; % to store the values of total 1
totalb=0; % to store the values of total 2
n1=d+1;   % getting the 5th data point
n2=d+1;
n =0;
for i=d+1:Halflength
    totala = totala + iy1(i)

```

```

totalb = totalb + iy2(i)
end
while n==0
    sum=sum+iy1(n1);
    if (sum/totala)<0.4 % 40% of the total energy
        n1=n1+1;
    else
        n=1;
    end
end % 40% of the total energy
sum=0;
while n==1
    sum=sum+iy2(n2);
    if (sum/totalb)<0.4
        n2=n2+1;
    else
        n=0;
    end
end
% to obtain the breakpoint
if n1>=n2
    n1=n2;
else
    n2=n1;
end
n1=n1+5;
n2=n2+5; % five more points than the %40 point

%*****Windows*****%
L1=zeros(Halflength+1,1); %Initializing Low window for data 1
H1=zeros(Halflength+1,1); %Initializing High window for data 1
L2=zeros(Halflength+1,1); %Initializing Low window for data 2
H2=zeros(Halflength+1,1); %Initializing High window for data 2

```

% negative and positive frequencies were fixed at the same time and windows were calculated for Half Length of the data.

```
for i=d+1:n1-3
```

```
    L1(i)=1;
```

```
end
```

```
for i=n1+3:Halflength+1
```

```
    H1(i)=1;
```

```
end
```

```
L1(n1-2)=0.834;
```

```
L1(n1-1)=0.6667;
```

```
L1(n1)=0.5;
```

```
L1(n1+1)=0.3333;
```

```
L1(n1+2)=0.166;
```

```
H1(n1-2)=0.166;
```

```
H1(n1-1)=0.3333;
```

```
H1(n1)=0.5;
```

```
H1(n1+1)=0.6667;
```

```
H1(n1+2)=0.834;
```

```
for i=d+1:n2-3
```

```
    L2(i)=1;
```

```
end
```

```
for i=n2+3:Halflength+1
```

```
    H2(i)=1;
```

```
end
```

```
L2(n2-2)=0.834;
```

```
L2(n2-1)=0.6667;
```

```
L2(n2)=0.5;
```

```
L2(n2+1)=0.3333;
```

```
L2(n2+2)=0.166;
```

```

H2(n2-2)=0.166;
H2(n2-1)=0.3333;
H2(n2)=0.5;
H2(n2+1)=0.6667;
H2(n2+2)=0.834;

```

```

high1=zeros(a(1),1);
high2=zeros(a(1),1);
low1=zeros(a(1),1);
low2=zeros(a(1),1);
xx = mod(maxlength,2);

```

```

j=Halflength+2;
high1(Halflength+1)=1;
high2(Halflength+1)=1;

```

```

low1(Halflength+1)=0;
low2(Halflength+1)=0;

```

```

for i = 1:Halflength

```

```

    high1(i)=H1(i);

```

```

    high2(i)=H2(i);

```

```

    low1(i)=L1(i);

```

```

    low2(i)=L2(i);

```

```

    if(xx == 1)

```

```

        high1(j)=H1(Halflength+2-i);

```

```

        high2(j)=H2(Halflength+2-i);

```

```

        low1(j)=L1(Halflength+2-i);

```

```

        low2(j)=L2(Halflength+2-i);

```

```

    else

```

```

        if i~=Halflength

```

```

            high1(j)=H1(Halflength+1-i);

```

```

            high2(j)=H2(Halflength+1-i);

```

```

low1(j)=L1(Halflength+1-i);
low2(j)=L2(Halflength+1-i);
    end
end

j=j+1;
end

high1(Halflength)=1;
high2(Halflength)=1;
low1(Halflength)=0;
low2(Halflength)=0;

%to obtain low and high filters in the transformed domain
Highfilter1=y1(1:a(1)).*high1;
Highfilter2=y2(1:a(1)).*high2;
Lowfilter1=y1(1:a(1)).*low1;
Lowfilter2=y2(1:a(1)).*low2
%*****IFFT*****%
ihigh1=ifft(Highfilter1);
ilow1=ifft(Lowfilter1);
idc1=ifft(dc1);
ihigh2=ifft(Highfilter2);
ilow2=ifft(Lowfilter2);
idc2=ifft(dc2);
%*****ADM*****%

alfa=zeros(1,maxlength);
bita=0;
axs=zeros(1,maxlength);
delta=0;
for i=1:maxlength
alfa(i)=abs(ilow1(i))-abs(ilow2(i));
bita=bita+(abs(ilow1(i))+abs(ilow2(i)));

```

```

axs(i)=abs(idc1(i))-abs(idc2(i));
delta=delta+(abs(idc1(i))+abs(idc2(i)));
end
bita=bita/maxlength;
delta=delta/maxlength;
cc=abs(axs./delta);
fraction=(alfa./bita);
ADM=abs(fraction)+cc.*exp(cc);
% to determine ADM total
ADM1=abs(fraction);
ADM2=cc.*exp(cc);
TADM=0;
TADM1=0;
TADM2 = 0;

for i=1:maxlength
    TADM=TADM+ADM(i);
    TADM1=TADM1+ADM1(i);
    TADM2=TADM2+ADM2(i);
end
TADM=TADM/maxlength;
TADM1=TADM1/maxlength;
TADM2=TADM2/maxlength;

%*****FDM*****%
% Initialising the First derivatives of inverse transformed Low & High data
d1ilow1 = zeros(maxlength,1);
d1ilow2 = zeros(maxlength,1);
d1ihigh1 = zeros(maxlength,1);
d1ihigh2 = zeros(maxlength,1);
% Initialising the second derivative of inverse transformed High data
d2ihigh1 = zeros(maxlength,1);
d2ihigh2 = zeros(maxlength,1);

```

% to obtain the First derivatives for specified interval

for i = 3:maxlength-2

d1ilow1(i) = ilow1(i+2) - ilow1(i-2);

d1ilow2(i) = ilow2(i+2) - ilow2(i-2);

d1ihigh1(i) = ihigh1(i+2) - ihigh1(i-2);

d1ihigh2(i) = ihigh2(i+2) - ihigh2(i-2);

end

% to obtain the First derivatives for specified interval

for i = 4:maxlength-3

d2ihigh1(i) = d1ihigh1(i+3) - d1ihigh1(i-3);

d2ihigh2(i) = d1ihigh2(i+3) - d1ihigh2(i-3);

end

% to calculate the denominator

F1=0;

F2=0;

F3=0;

for i=1:maxlength

F1=(abs(d1ilow1(i))+abs(d1ilow2(i)))+F1;

F2=(abs(d1ihigh1(i))+abs(d1ihigh2(i)))+F2;

F3=(abs(d1ihigh1(i))+abs(d2ihigh2(i)))+F3;

end

% to calculate the Numerator

FDM1=zeros(maxlength,1);

FDM2=zeros(maxlength,1);

FDM3=zeros(maxlength,1);

for i=1:maxlength

FDM1(i)=(abs(d1ilow1(i))-abs(d1ilow2(i)))/(F1*2/maxlength);

FDM2(i)=(abs(d1ihigh1(i))-abs(d1ihigh2(i)))/(F2*6/maxlength);

FDM3(i)=(abs(d1ihigh1(i))-abs(d2ihigh2(i)))/(F3*7.2/maxlength);

end

FDM=2*abs(FDM1+FDM2+FDM3);

% to determine FDM total


```

TFDM = 0;
for i=1:maxlength
    TFDM=TFDM+FDM(i);
end
TFDM=TFDM/maxlength;

%*****GDM*****%
GDM=zeros(maxlength,1);
for i=1:maxlength
    GDM(i)=sqrt(ADM(i)*ADM(i)+FDM(i)*FDM(i));
end
% to determine GDM total
TGDM = 0;
for i=1:maxlength
    TGDM=TGDM+GDM(i);
end
TGDM=TGDM/maxlength;

%*****STORE OUTPUTS*****%
% ADM,FDM and GDM were exported as .CSV file
ADMOutFileName = "ADM"+ i+".csv";
FDMOutputfile = "FDM"+ i+".csv";
GDMOutFileFormat = "GDM"+ i+".csv";
csvwrite(ADMOutFileName,ADM)
csvwrite(FDMOutputfile,FDM)
end

```

APPENDIX 6

VBA MACRO FUNCTION TO COPY EACH ROW FROM SOURCE SHEET TO THE DESTINATION SHEET

Sub copyRowWiseOutputs()

'VBA Macro function to copy column from individual source sheets and transpose paste it to destination sheet.

'For example: Columns from 256 individual source sheets will be copied, transposed and pasted to 256 rows in the destination sheet.

'Individual source sheets are opened using user specified filename and filepath

' Variable declared

Dim noOfRows

Dim i

Dim fileName

Dim filePath

' User to update the number of rows in the source sheet

noOfRows = "<number of Rows>"

' User to update the file path of the destination sheets to be saved

filePath = "<filepath>"

'User to update the desired file name of the destination sheets

fileName = "<filename>"

' For loop to iterate the copy and paste operations

For i = 1 To noOfRows

Windows("Save output to single file.xlsm").Activate

Range("A1").Select

'Steps to open individual source sheet and copy column

```
inputFile = filePath & fileName & i & ".csv"
```

```
Debug.Print inputFile
```

```
Workbooks.Open inputFile
```

```
Windows(fileName & i & ".csv").Activate
```

```
Range("A1").Select
```

```
Range(Selection, Selection.End(xlDown)).Select
```

```
Selection.Copy
```

```
'Steps to activate destination sheet, transpose and paste as row
```

```
Windows("Save output to single file.xlsm").Activate
```

```
Range("A" & i).Select
```

```
Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _
```

```
False, Transpose:=True
```

```
'Close the individual source sheet
```

```
Windows(fileName & i & ".csv").Activate
```

```
ActiveWindow.Close
```

```
' Clear clipboard
```

```
Application.CutCopyMode = False
```

```
' loop the steps till noOfRows value
```

```
Next i
```

```
End Sub
```

APPENDIX 7

VBA MACRO FUNCTION TO COPY EACH COLUMN FROM SOURCE SHEET TO THE DESTINATION SHEET

Sub copyColumnWiseOutputs()

'VBA Macro function to copy column from individual source sheets and paste it to destination sheet.

'For example: Columns from 256 individual source sheets will be copied and pasted to 256 columns in the destination sheet.

'Individual source sheets are opened using user specified filename and filepath

' Variable declared

Dim noOfColumns

Dim i

Dim fileName

Dim filePath

' User to update the number of rows in the source sheet

noOfColumns = "<number of columns>"

' User to update the file path of the destination sheets to be saved

filePath = "<filepath>"

'User to update the desired file name of the destination sheets

fileName = "FDM"

' For loop to iterate the copy and paste operations

For i = 1 To noOfColumns

Windows("Save output to single file.xlsm").Activate

Range("A1").Select

'Steps to open individual source sheet and copy column

```
inputFile = filePath & fileName & i & ".csv"  
Workbooks.Open inputFile  
Windows(fileName & i & ".csv").Activate  
Range("A1").Select  
Range(Selection, Selection.End(xlDown)).Select  
Selection.Copy
```

'Steps to activate destination sheet, and paste as columns

```
Windows("Save output to single file.xlsm").Activate  
Range("A1").Select  
ActiveCell.Offset(0, i).Select  
Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _  
False, Transpose:=False
```

'Close the individual source sheet

```
Windows(fileName & i & ".csv").Activate  
ActiveWindow.Close
```

' Clear clipboard

```
Application.CutCopyMode = False
```

' loop the steps till noOfColumns value

```
Next i
```

```
End Sub
```

APPENDIX 8

PROGRAM TO CALCULATE 2-D FSV AND PLOT ITS OUTPUT IN MATLAB

```

close all;
clc;

m = 208; % length of row data
n = 216; % length of column data
% Import row and column xDM's to perform 2-D FSV
row_ADM = importdata('row_ADM.xlsx');
row_FDM = importdata('row_FDM.xlsx');
column_ADM = importdata('column_ADM.xlsx');
column_FDM = importdata('column_FDM.xlsx');
% to obtain the maximum length of the 2-D data
maxlength = m*n;

% to calculate the 2-D FSV
% Weighting factors Kv and KH are calculated
kv = n/(m+n);
kh = m/(m+n);
Column_ADM = (column_ADM).^2 % ADM calculated form column data sets
Column_FDM = (column_FDM).^2 % FDM calculated form column data sets
Row_ADM = (row_ADM).^2 % ADM calculated form row data sets
Row_FDM = (row_FDM).^2 % FDM calculated form row data sets

% applying the 2-D FSV formulae to calculate the point-by-point ADM, FDM and GDM
ADM = sqrt(kv*( Column_ADM) + kh*( Row_ADM));
FDM = sqrt(kv*( Column_FDM) + kh*( Row_FDM));
GDM = sqrt((ADM).^2+(FDM).^2);

% obtaining the confidence histograms for ADM, FDM and GDM
ADMScale=zeros(6,1);
FDMScale=zeros(6,1);
GDMScale=zeros(6,1);

```

% FSV interpretation results

```
for i=1:maxlength
    if ADM(i)<=0.1
        ADMScale(1)=ADMScale(1)+1;
    elseif ADM(i)>0.1 && ADM(i)<=0.2
        ADMScale(2)=ADMScale(2)+1;
    elseif ADM(i)>0.2 && ADM(i)<=0.4
        ADMScale(3)=ADMScale(3)+1;
    elseif ADM(i)>0.4 && ADM(i)<=0.8
        ADMScale(4)=ADMScale(4)+1;
    elseif ADM(i)>0.8 && ADM(i)<=1.6
        ADMScale(5)=ADMScale(5)+1;
    else
        ADMScale(6)=ADMScale(6)+1;
    end

    if FDM(i)<=0.1
        FDMScale(1)=FDMScale(1)+1;
    elseif FDM(i)>0.1 && FDM(i)<=0.2
        FDMScale(2)=FDMScale(2)+1;
    elseif FDM(i)>0.2 && FDM(i)<=0.4
        FDMScale(3)=FDMScale(3)+1;
    elseif FDM(i)>0.4 && FDM(i)<=0.8
        FDMScale(4)=FDMScale(4)+1;
    elseif FDM(i)>0.8 && FDM(i)<=1.6
        FDMScale(5)=FDMScale(5)+1;
    else
        FDMScale(6)=FDMScale(6)+1;
    end

    if GDM(i)<=0.1
        GDMScale(1)=GDMScale(1)+1;
    elseif GDM(i)>0.1 && GDM(i)<=0.2
        GDMScale(2)=GDMScale(2)+1;
```

```

elseif GDM(i)>0.2 && GDM(i)<=0.4
    GDMScale(3)=GDMScale(3)+1;
elseif GDM(i)>0.4 && GDM(i)<=0.8
    GDMScale(4)=GDMScale(4)+1;
elseif GDM(i)>0.8 && GDM(i)<=1.6
    GDMScale(5)=GDMScale(5)+1;
else
    GDMScale(6)=GDMScale(6)+1;
end
end
end

% to plot the Histograms
figure(1)
subplot(3,1,1)
bar(ADMScale/maxlength);
set(gca,'XTicklabel',{'Excellent','Very Good','Good','Fair','Poor','Very Poor'});
legend('ADM');
xlabel('FSV interpretation')
ylabel('FSV value')

subplot(3,1,2)
bar(FDMScale/maxlength);
set(gca,'XTicklabel',{'Excellent','Very Good','Good','Fair','Poor','Very Poor'});
legend('FDM');
xlabel('FSV interpretation')
ylabel('FSV value')

subplot(3,1,3)
bar(GDMScale/maxlength);
set(gca,'XTicklabel',{'Excellent','Very Good','Good','Fair','Poor','Very Poor'});
legend('GDM');
xlabel('FSV interpretation')
ylabel('FSV value')

```


% to calculate the total values for ADM, FDM and GDM

TADM = mean(ADM,'all');

TFDM = mean(FDM,'all');

TGDM = mean(GDM,'all')

APPENDIX 9

PROGRAM TO CALCULATE PDF, CDF AND STATISTICAL MOMENTS

```

close all;
clc;

m = 208; % length of row data
n = 216; % length of column data
% to obtain the maximum length of the 2-D data
maxlength = m*n;
% importing the data to be compared
GDM = importdata('E0E2_recombined_GDM.xlsx');

% Linearized FSV Rating Scale results
GG=zeros(maxlength,1);
for i=1:maxlength
    if GDM(i)>=0 && GDM(i)<0.2
        GG(i)=GDM(i);
    elseif GDM(i)>=0.2 && GDM(i)<0.4
        GG(i)=0.2+(GDM(i)-0.2)/2;
    elseif GDM(i)>=0.4 && GDM(i)<0.8
        GG(i)=0.3+(GDM(i)-0.4)/4;
    elseif GDM(i)>=0.8 && GDM(i)<1.6
        GG(i)=0.4+(GDM(i)-0.8)/8;
    elseif GDM(i)>=1.6 && GDM(i)<3.2
        GG(i)=0.5+(GDM(i)-1.6)/16;
    else
        GG(i)=0.6;
    end
end
% Obtaining Histograms for the Linearized GDM values
figure(1)
hist(GG);
xlabel('Linearized GDMi')
ylabel('Probability density')

```

```

title('Linearized GDM output')
set(gcf,'color','white')
% Probability Density Function
figure(2)
[visualff,x] = ksdensity(GG);
plot(x,visualff,'r','linewidth',2);
xlabel('Linearized GDMi')
ylabel('Probability density')
set(gcf,'color','white')
legend('FSV');
title('PDF')
% Emperhical Cumulative Distribution Function
figure(3)
[f,x] = ecdf(GG)
plot(x,f,'r','linewidth',2);
xlabel('GDMi')
ylabel('CDF')
title('CDF');
set(gcf,'color','white')

% Calculating the moments of statistics

S = std(GG,0,'all'); % standard deviation
V = var(GG,0,'all'); % variance
sk = skewness(GG,1,'all'); % skewness
k = kurtosis(GG,1,'all'); % Kurtosis

```

APPENDIX 10

PROGRAM TO PERFORM K-S TEST

```

close all;
clc;

m = 208; % length of row data
n = 216; % length of column data

% to obtain the maximum length of the 2-D data
maxlength = m*n;

% Importing GDM output obtained from original approach 1
GDM_fullimage = importdata('E0_E2_GDM.xlsx');

% Importing GDM output obtained from segmented approach 2
GDM_recombined = importdata('E0E2_recombined_GDM.xlsx');

%FSV interpretation results for full image
GG_fullimage=zeros(maxlength,1);
for i=1:maxlength
    if GDM_fullimage(i)>=0 && GDM_fullimage(i)<0.2
        GG_fullimage(i)=GDM_fullimage(i);
    elseif GDM_fullimage(i)>=0.2 && GDM_fullimage(i)<0.4
        GG_fullimage(i)=0.2+(GDM_fullimage(i)-0.2)/2;
    elseif GDM_fullimage(i)>=0.4 && GDM_fullimage(i)<0.8
        GG_fullimage(i)=0.3+(GDM_fullimage(i)-0.4)/4;
    elseif GDM_fullimage(i)>=0.8 && GDM_fullimage(i)<1.6
        GG_fullimage(i)=0.4+(GDM_fullimage(i)-0.8)/8;
    elseif GDM_fullimage(i)>=1.6 && GDM_fullimage(i)<3.2
        GG_fullimage(i)=0.5+(GDM_fullimage(i)-1.6)/16;
    else
        GG_fullimage(i)=0.6;
    end
end
end

```

%FSV interpretation results for segmented and recombined image

```
GG_recombined=zeros(maxlength,1);
for i=1:maxlength
    if GDM_recombined(i)>=0 && GDM_recombined(i)<0.2
        GG_recombined(i)=GDM_recombined(i);
    elseif GDM_recombined(i)>=0.2 && GDM_recombined(i)<0.4
        GG_recombined(i)=0.2+(GDM_recombined(i)-0.2)/2;
    elseif GDM_recombined(i)>=0.4 && GDM_recombined(i)<0.8
        GG_recombined(i)=0.3+(GDM_recombined(i)-0.4)/4;
    elseif GDM_recombined(i)>=0.8 && GDM_recombined(i)<1.6
        GG_recombined(i)=0.4+(GDM_recombined(i)-0.8)/8;
    elseif GDM_recombined(i)>=1.6 && GDM_recombined(i)<3.2
        GG_recombined(i)=0.5+(GDM_recombined(i)-1.6)/16;
    else
        GG_recombined(i)=0.6;
    end
end
```

% calculating and plotting Empirical Cumulative Distribution Function (ECDF) for GDM outputs obtained from approach 1 (original method) and approach 2 (segmented and recombined method)

```
figure(2)
subplot(2,1,1)
[f,x] = ecdf(GG_recombined)
plot(x,f,'r','linewidth',2);
xlabel('GDMi')
ylabel('CDF')
title('CDF - segmented and recombined');
set(gcf,'color','white')

subplot(2,1,2)
[f,x] = ecdf(GG_fullimage)
plot(x,f,'r','linewidth',2);
xlabel('GDMi')
```

```
ylabel('CDF')
title('CDF -full image');
set(gcf,'color','white')
% KS – test
% sorting the ECDF outputs
ecdf_GG_recombined = sort(GG_recombined);
ecdf_GG_full = sort(GG_fullimage)
% calculating test statistic value
D = abs(max(ecdf_GG_recombined - ecdf_GG_full))
% calculating D critical value
Dcritical = 1.36*sqrt((m+n)/(m*n));
```