

Improving Performance of Edge Tracing and Tracking of a Linearly-Moving Object using Mesh and Torus connected Parallel Processors

Sonal Gupta



**DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI**

MARCH 2025

© **Indian Institute of Technology Delhi (IITD), New Delhi, 2025**

Improving Performance of Edge Tracing and Tracking of a Linearly-Moving Object using Mesh and Torus connected Parallel Processors

by

Sonal Gupta

DEPARTMENT OF ELECTRICAL ENGINEERING

submitted

in fulfilment of the requirement of Doctor of philosophy

to the



INDIAN INSTITUTE OF TECHNOLOGY DELHI

MARCH 2025

Dedicated to my beloved grandparents

Late Shri B. L. Gupta

Late Smt. Kamla Devi

Shri T. C. Gupta

Smt. Pushpa Devi

CERTIFICATE

This is to certify that the thesis titled **Improving Performance of Edge Tracing and Tracking of a Linearly-Moving Object using Mesh and Torus connected Parallel Processors**, submitted by **Sonal Gupta (2018EEZ8143)**, to the Indian Institute of Technology Delhi for the award of the degree of **Doctorate of Philosophy**, is a bona fide record of the research work done by her under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Subrat Kar

Professor

Deptt. of Electrical Engg.

IIT-Delhi, 110016

Place: New Delhi

Date: March 26, 2025

ACKNOWLEDGEMENTS

I would like to extend my deepest thanks to all those who have supported me throughout the research journey encapsulated in this thesis. The journey of academic research is seldom a solitary endeavour but rather a collaborative effort enriched by the support and guidance of many. The following is a recognition of those who have played a significant role in this academic journey.

I extend my deepest gratitude to my Supervisor, Prof. Subrat Kar, for his invaluable guidance, support, and encouragement during my research. His profound insights, expertise, and constructive critiques have been pivotal in shaping my thesis.

Special thanks go to the members of the Student Research Committee, Prof. Seshan Srirangarajan, Prof. Sumantra DuttaRoy, and Prof. B.S. Panda. Their expert advice and feedback have been essential, and I deeply appreciate their assistance.

I am immensely thankful to my husband Dr. Gaurav Singh Raghuwanshi and our family - Mahesh Gupta, Manju Gupta, Vikas Gupta, Dr. Ravindra Singh Raghuwanshi, Anita Raghuwanshi, Sonia Raghuwanshi, Piyush Thakur, Tejaswini Thakur, Sourabh Singh Raghuwanshi, Dr. Vidhi Raghuwanshi, Madhurima Ghose, Dr. Satya Gupta - for their unwavering support and encouragement.

I also express my gratitude to my colleagues and friends - Dr. Bodhibrata Mukhopadhyay, Dr. Naresh Reddy, Dr. Dolar Khachariya, Dr. Debashis Adhikari, Dr. Pallabi Sarkar, Dr. Ankur Beohar, Dr. Abha Sharma, Dr. Krishna Chauhan, Vikrant Giri, Dr. Sahil Anchal, Animesh Bhattacharya, Prateek Yadav, Aditi Gupta, Rahul Srivastava, Arpita Agarwal, Sneha Aggarwal, and Zeal Sheth. Their support has been a pillar of strength throughout this journey.

Lastly, I acknowledge the higher power, whose mysterious workings remind us of our humble place in the universe. This acknowledgement goes beyond a mere thank you, recognizing the enigmatic forces that weave the tapestry of our existence.

Sonal Gupta

ABSTRACT

KEYWORDS: interconnection networks, mesh, torus, twisted torus, parallel processing, FPGA, GPU, supercomputer, parallel processing arrays, line scan camera, edge detection, soccer ball, object tracking, image processing, computer vision

We proposed and developed parallel processing algorithms (and corresponding hardware) to speed-up two image processing applications - edge tracing and tracking of a linearly moving object - using the connecting properties of the interconnecting network. Further, we used a twisted Rectangular Torus-based interconnection network to improve the performance of such parallel processing applications.

First, we developed *three* contour tracing algorithms (Adapted and Segmented (AnS) algorithms), adapting three major families of contour tracing algorithms (pixel-following, vertex-following, and run-data-based-following algorithms) for the mesh and torus-connected architectures. These algorithms used distributed data processing for tracing contours which exploited the specific interconnect properties of torus network. We simulated these algorithms on a two-dimensional torus-connected multiprocessor model. Using a Python environment, we achieved a speed-up of 468 times (considering clock ticks) over single-processor systems. Our implementation on Tesla K40c and Quadro RTX 5000 GPUs showed a speedup of 194 (and 47 respectively) compared to the existing parallel processing contour tracing implementations.

Second, we implemented our developed algorithms on a Zynq-7000 FPGA-based platform, building a hardware accelerator based on our Adapted-and-Segmented (AnS) algorithms. This implementation used a mesh-interconnected multiprocessor architecture and was much faster than existing methods - 55 times faster (if input-output overheads were excluded) and 12.5 times faster (if input-output overheads were included).

Third, we developed an efficient and cost-effective system for tracking linearly moving objects in sports fields using a Panning Line-Scan Camera and Lens (P-LSCL) setup. We introduced a novel angle of motion prediction algorithm based on edge tracing, specifically designed for tracking a white football

in a football game within a stadium (with artificial grass as a uniform green background) which is 1.15 times more accurate, $O(n^2)$ times less complex, and only 1/6th of the cost of current methods.

Fourth, and last, we enhanced the connectivity properties in Rectangular Torus (RT) networks by introducing twists to reduce the Average Inter-node Distance (AID). We proposed a methodology for implementing twists in RT networks, studying the Average Inter-node Distance as a function of the twist (t-shift) in one dimension. This methodology can be used for mapping *any* Rectangular Twisted Torus network onto a 2D VLSI chip, potentially enhancing performance and connectivity in parallel processing systems.

सारांश

सारांश कीवर्ड: इंटरकनेक्शन नेटवर्क, मेष, टोरस, ट्विस्टेड टोरस, समानांतर प्रसंस्करण, एफपीजीए, जीपीयू, सुपर-कंप्यूटर, समानांतर प्रसंस्करण ऐरेज, लाइन स्कैन कैमरा, एज डिटेक्शन, फुटबॉल, ऑब्जेक्ट ट्रैकिंग, इमेज प्रोसेसिंग, कंप्यूटर विजन

हमने इंटरकनेक्टिंग नेटवर्क के जुड़ने की विशेषताओं का उपयोग करके दो इमेज प्रोसेसिंग अनुप्रयोगों - एज ट्रेसिंग और रैखिक रूप से चलने वाली वस्तु की ट्रैकिंग की गति बढ़ाने के लिए समानांतर प्रसंस्करण एल्गोरिदम (और संबंधित हार्डवेयर) का प्रस्ताव और विकास किया। इसके अलावा, हमने ऐसे समानांतर प्रसंस्करण अनुप्रयोगों के प्रदर्शन में सुधार के लिए एक ट्विस्टेड रेक्टैंगुलर टोरस-आधारित इंटरकनेक्शन नेटवर्क का उपयोग किया।

सबसे पहले, हमने तीन कंटूर ट्रेसिंग एल्गोरिदम (अनुकूलित और विभाजित (AnS) एल्गोरिदम) विकसित किए, जिन्होंने मेष और टोरस-कनेक्टेड आर्किटेक्चर के लिए तीन प्रमुख कंटूर ट्रेसिंग एल्गोरिदम परिवारों (पिक्सेल-अनुसरण, वर्टेक्स-अनुसरण, और रन-डेटा-आधारित-अनुसरण एल्गोरिदम) को अनुकूलित किया। इन एल्गोरिदम ने टोरस नेटवर्क की विशिष्ट इंटरकनेक्ट संपत्तियों का लाभ उठाते हुए कंटूर को ट्रेस करने के लिए वितरित डेटा प्रसंस्करण का उपयोग किया। हमने इन एल्गोरिदम को एक दो-आयामी टोरस-कनेक्टेड मल्टीप्रोसेसर मॉडल पर सिमुलेट किया। Python पर्यावरण का उपयोग करते हुए, हमने एकल-प्रोसेसर सिस्टम की तुलना में 468 गुना की गति वृद्धि (घड़ी टिक्स को ध्यान में रखते हुए) हासिल की। हमारा कार्यान्वयन Tesla K40c और Quadro RTX 5000 GPUs पर मौजूदा समानांतर प्रसंस्करण कंटूर ट्रेसिंग कार्यान्वयनों की तुलना में क्रमशः 194 (और 47) गुना तेज था।

दूसरे, हमने हमारे विकसित किए गए एल्गोरिदम को Zynq-7000 FPGA-आधारित प्लेटफॉर्म पर लागू किया, जिसमें हमारे अनुकूलित-और-विभाजित (AnS) एल्गोरिदम पर आधारित हार्डवेयर त्वरक बनाया गया। इस कार्यान्वयन ने मेष-इंटरकनेक्टेड मल्टीप्रोसेसर आर्किटेक्चर का उपयोग किया और मौजूदा विधियों की तुलना में बहुत तेज था - यदि इनपुट-आउटपुट ओवरहेड्स को छोड़ दिया जाए तो 55 गुना तेज और यदि इनपुट-आउटपुट ओवरहेड्स को शामिल किया जाए तो 12.5 गुना तेज।

तीसरे, हमने खेल मैदानों में रैखिक रूप से चलने वाली वस्तुओं को ट्रैक करने के लिए एक कुशल और लागत प्रभावी प्रणाली विकसित की, जिसमें पैनिंग लाइन-स्कैन कैमरा और लेंस (P-LSCL) सेटअप का उपयोग किया गया। हमने एक नवीन कोण मोशन प्रेडिक्शन एल्गोरिदम पेश किया जो एक फुटबॉल गेम में सफेद फुटबॉल की ट्रैकिंग के लिए विशेष रूप से डिज़ाइन किया गया था, जिसमें समान हरे रंग की कृत्रिम घास के साथ स्टेडियम का उपयोग किया गया था, जो मौजूदा विधियों की तुलना में 1.15 गुना अधिक सटीक, $O(n^2)$ बार कम जटिल, और केवल लागत का 1/6वां है।

चौथा, और अंत में, हमने रेक्टैंगुलर टोरस (RT) नेटवर्कों में कनेक्टिविटी की विशेषताओं को बढ़ाया, औसत इंटर-नोड दूरी (AID) को कम करने के लिए ट्विस्ट पेश किया। हमने एक आयाम में ट्विस्ट (t-shift) के रूप में औसत इंटर-नोड दूरी का अध्ययन करने के लिए RT नेटवर्कों में ट्विस्ट्स को लागू करने की एक पद्धति प्रस्तावित की। यह पद्धति किसी भी रेक्टैंगुलर ट्विस्टेड टोरस नेटवर्क को 2D VLSI चिप पर मैप करने के लिए उपयोग की जा सकती है, जिससे समानांतर प्रसंस्करण प्रणालियों में प्रदर्शन और कनेक्टिविटी को संभवतः बढ़ाया जा सकता है।

Contents

CERTIFICATE	i
ACKNOWLEDGEMENTS	ii
ABSTRACT	iii
List of Figures	x
List of Tables	xiii
1 INTRODUCTION	1
1.1 Motivation	1
1.2 Related Work Review	5
1.2.1 Edge Tracing and its FPGA implementation	5
1.2.2 Real-Time Ball Tracking in Sports	6
1.3 Methodology	6
2 Speeding Up Edge Tracing on Mesh or Torus Connected Architectures	9
2.1 Introduction	9
2.2 Methodology	10
2.2.1 Hardware Framework	10
2.2.2 About Mesh and Torus Interconnection Network	11
2.2.3 Overview of the methodology	12
2.2.4 Choice of an appropriate test image	13
2.2.5 Partitioning the test image as a square planar tessellation with 4-connectedness to match underlying multiprocessor hardware	13
2.2.6 Details of Algorithms (Preprocessing and conventional contour tracing in the test image)	14

2.2.7	Converting conventional contour tracing algorithm for a multiprocessor implementation	15
2.2.8	Adaptation of Pixel-Following algorithm to multiprocessor implementation	16
2.2.9	Adaptation of Vertex-Following algorithm to multiprocessor implementation	18
2.2.10	Adaptation of Run-Data-Based-Following algorithm to multiprocessor implementation	19
2.2.11	Efficient parallel algorithms for a multiprocessor system using segmentation	20
2.2.12	Metric used for comparison between different algorithms and their adaptations	21
2.2.13	Proposed implementation on NVIDIA GPUs used in existing parallel processing algorithms	22
2.3	Results	23
2.3.1	Comparison between the conventional algorithms and their multiprocessor versions	23
2.3.2	Comparison of the increased efficiency in the three algorithms	24
2.3.3	Performance comparison between the existing parallel processing algorithms and our implementation	26
2.4	Discussion	27
3	An FPGA based Implementation as a Proof-of-Concept of Accelerated Edge Tracing in Real-Time Imaging	31
3.1	Introduction	31
3.2	Methodology	31
3.2.1	Overview	31
3.2.2	About the Interconnection Network	32
3.2.3	Test image used	32
3.2.4	Hardware Framework implementation on the FPGA	33
3.2.5	Software Framework - Description of the Algorithms and their Implementation using Verilog Testbench	36
3.2.6	Projecting our implementation on GPU hardware	38
3.2.7	Testing on an image from ImageNet database	40
3.3	Results and Discussion	42

3.3.1	The contour outputs of AnS-VF and AnS-RDBF algorithms	42
3.3.2	Number of clock cycles required	42
3.3.3	Timing details of the hardware implementation	45
3.3.4	Area Utilization and Power Parameters	46
3.3.5	Comparison of performance with the existing implementations and discussion	47
4	Efficient and Inexpensive Tracking of Linearly Moving Object Using Panning Line-Scan Camera	52
4.1	Introduction	52
4.2	Methodology	53
4.2.1	Overview	53
4.2.2	Experimental Setup	54
4.2.3	The pixels occupied by a spherical football in the image stream	55
4.2.4	Initial Position Detection	55
4.2.5	Pre-processing Steps	56
4.2.6	Differential Position Detection and Trajectory Prediction	56
4.2.7	Calculations for the specifications of the line-scan camera and the convex lens required to scan a football field	59
4.2.8	Hardware Used for Implementation of Proof of Concept	61
4.3	Results and Discussion	61
4.3.1	Metrics Used for Analysis	63
4.3.2	Comparison with Existing Methods	64
4.4	Future Scope	64
5	Connectivity Improvement using Rectangular Twisted Torus Networks and Mapping them on 2D VLSI Chip	66
5.1	Introduction	66
5.2	Properties of the Torus Networks	67
5.2.1	Rectangular Torus Network	67
5.3	Average Internode (Shortest) Distance Calculation and Trend	69
5.4	Mapping t-shift RTT-network	71

5.5	Discussion	75
6	Discussion and Conclusions	77
6.1	Results and Discussion	77
6.2	Conclusion	79
6.3	Summary	80
6.4	Future work	81
	References	81
	List of Publications from this Thesis	91
	Author Biography	92

List of Figures

2.1	General supercomputer and GPU network architecture where PE is Processing Element, NI is Network Interface, CTRL is Control Unit, and SCHED is Scheduler. Boxes with a cross are the Routers.	11
2.2	A $c \times r$ 2-D mesh connected network of processors with 8 columns and 6 rows, <i>i.e.</i> , a total of 48 processing elements.	12
2.3	A $c \times r$ 2-D torus connected network of processors with 8 columns and 6 rows – a total of 48 processing elements.	12
2.4	(a) Standard Image and (b) its $4 \times$ scaled version.	13
2.5	All ten local patterns which our algorithms can encounter in a digital binary image[44].	14
2.6	An $n \times m$ 2-D torus connected network of processors with 26 rows and 18 columns, <i>i.e.</i> , a total of 468 processing elements with the standard image overlaid – the wraparound connections (similar to Fig.2.3) are present but are not explicitly shown in this figure.	15
2.7	(a) Tracing of the standard image using our implementation of the pixel-following algorithm; dot represents the first active pixel detected; (b) The resultant pixels in the contour stack.	18
2.8	(a) Tracing of the standard image using our implementation of the vertex-following algorithm; dot represents the starting of the first vertex detected; (b) The resultant pixels in the contour stack.	19
2.9	(a) Tracing of the standard image using our implementation of the run-data-based-following algorithm; dot represents the first active pixel detected; (b) The resultant pixels in the contour stack.	20
2.10	Pictorial explanation of the concept of the chapter – (a) Distribution of image pixels in a multiprocessor system; Each dotted square in the 2-D array is a processing element; (b) The multiprocessor hardware system divided into 3×3 segments. . . .	21
2.11	Speed-up as a function of Number of Segments in log-log scale.	24

3.1	4x version of the standard image displaying the corresponding pixel values stored in PE registers. Corresponding pixel values stored in PE registers are also displayed.	33
3.2	4x version of the Standard image with Holes (SH) to include all the ten local patterns. Corresponding pixel values stored in PE registers are also displayed.	34
3.3	The Standard Holed (SH) image with all the ten local patterns covered.	34
3.4	Schematic of the two-bit comparator module	34
3.5	Schematic of the one-bit counter module	35
3.6	(a) Test image chosen from the ImageNet database[38], (b) Grayscaled and Thresholded image using Threshold = 150, (c) Grayscaled and Thresholded image using Threshold = 100, (d) Grayscaled and Thresholded image using Threshold = 50.	41
3.7	(a) Random 18x26 segment from Fig. 3.6c, (b) Second Random 18x26 segment from Fig. 3.6c, (c) Contour obtained using our accelerator on Fig. 3.7a, (d) Contour obtained using our accelerator on Fig. 3.7a. Red pixels indicate those that would appear twice in the array of contour pixels.	41
3.8	Output contour and the pixel values stored in the PE output registers for the Vertex-Following algorithm.	43
3.9	Output contour and the pixel values stored in the PE output registers for the Run-Data-Based-Following algorithm.	43
3.10	Output contour and the pixel values stored in the PE output registers for the Vertex-Following algorithm.	44
3.11	Output contour and the pixel values stored in the PE output registers for the Run-Data-Based-Following algorithm.	44
3.12	(a) Area utilization (%) report for the design implemented on Zynq-7000 FPGA, (b) Power report for the design implemented on Zynq-7000 FPGA.	46
3.13	Speed-up (in log-scale) by AnS-RDBF using the given GPU hardware (1) excluding IO delay (dark grey), and (2) including IO delay (light grey).	48
4.1	3D diagram of the football field being seen by the sensor array through the convex lens	54
4.2	Football occupying one to four pixels. (a) One Pixel, (b) Two Pixels, (c) Three Pixels, (d) Four Pixels.	55
4.3	The eight-connected neighbours and the 16 angles of football motion detected by our algorithm.	57

4.4	Flowchart of the algorithm for predicting the trajectory of the ball	58
4.5	Ray diagram of the football field being seen by the sensor array through the convex Lens (Note: Dimensions are not scaled to the actual values)	60
4.6	Representative diagram of the camera (with lens) capturing the movement of the football along a trajectory (on a computer screen)	61
4.7	The four ‘trajectories of the football’ considered for the experiment taken from [69]. The ball is assumed to move from left to right along the trajectories.	62
4.8	Image displaying the pixels detected in the four trajectories by our algorithm. The greyed-out pixels indicate the detected ones, while the white ones were deemed unnecessary.	62
5.1	A 0-shift 8×4 Rectangular Torus network	68
5.2	A 1-shift 8×4 Rectangular Twisted Torus (RTT) network	68
5.3	A 4-shift 8×4 Rectangular Twisted Torus (RTT) network	70
5.4	Average Internode Distance as a function of ‘ t ’ in t -shift $p_1 \times 4$ RTT-networks	70
5.5	Arrangement after corresponding operation done to the Original Row arrangement[87].	71
5.6	8×4 1-shift Rectangular Twisted Torus (RTT) network.	72
5.7	8×4 4-shift Rectangular Twisted Torus (RTT) network.	73
5.8	8×4 2-shift Rectangular Twisted Torus (RTT) network.	74
5.9	8×4 3-shift Rectangular Twisted Torus (RTT) network.	75

List of Tables

2.1	Performance speed-up of the three algorithms for the different number of segments.	25
2.2	Comparison between the Adaptation of the three algorithms on the basis of various parameters.	26
2.3	Performance of our methodology using GPU hardware used by existing parallel processing implementations.	28
2.4	Memory access time and total time of our methodology using GPU hardware used by existing parallel processing implementations.	29
2.5	Time comparison between the performance of our methodology and the existing parallel processing methodologies by Garcia-Molla <i>et al.</i> ,[24] and Zhou <i>et al.</i> ,[26].	30
3.1	Comparison between the memory saved on 4x image and Standard Holed (SH) image using AnS-PF, AnS-VF and AnS-RDBF	43
3.2	Number of Clock Cycles (CC) required for the contour tracing algorithms. SH Image stands for Standard Holed Image . . .	45
3.3	Algorithmic time and the input-output time taken by our AnS-RDBF algorithm using the same GPU hardware used by Garcia-Molla <i>et al.</i> in[24] (Tesla K40c and Quadro RTX 5000), and Zhou <i>et al.</i> in[26] (Tesla K20c).	50
3.4	Time comparison of our AnS-RDBF implementation with the existing methodologies by Garcia-Molla <i>et al.</i> [24], and Zhou <i>et al.</i> [26].	51
4.1	Performance of our algorithm using the two accuracy metrics for the four trajectories. Pixel is shortened as px.	61
4.2	Performance of our algorithm for the four trajectories.	63
5.1	't' with minimum AID for the given t -shift $p_1 \times 4$ RTT network	70