A Parallel Implementation of the CIS Sea-Ice Motion Tracking Algorithm for Coarse-Grained Multicomputers

by

Francis Lee

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute of Computer Science

School of Computer Science
Carleton University
Ottawa, Ontario
December 5, 1996

© Copyright
1996, Francis Lee

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

—

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

—

ISBN 0-612-19485-X

Name FRANCIS LEA

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

COMPUTER SCIENCE

SUBJECT TERM

0984 UMI
SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language 0479
General 0289
Ancient 0290
Linguistics 0291
Modern 0291
Literature 0401
General 0294
Classical 0295
Comparative 0297
Medieval 0298
Modern 0298
African 0316
American 0591
Asian 0305
Canadian (English) 0352
Canadian (French) 0355
English 0593
Germanic 0311
Latin American 0312
Middle Eastern 0315
Romance 0313
Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion 0318
General 0321
Biblical Studies 0319
Clergy 0320
History of 0322
Philosophy of 0469
Theology 0469

SOCIAL SCIENCES

American Studies 0323
Anthropology 0324
Archaeology 0326
Cultural 0327
Physical 0310
Business Administration 0272
Accounting 0770
Banking 0454
Management 0338
Marketing 0385
Canadian Studies 0501
Economics 0503
General 0505
Agricultural 0508
Commerce-Business 0509
Finance 0510
History 0511
Labor 0358
Theory 0366
Folklore 0351
Geography 0578
Gerontology 0578
History 0578
General 0578

Ancient 0579
Medieval 0581
Modern 0582
Black 0378
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0333
United States 0337
History of Science 0585
Law 0398
Political Science 0615
General 0616
International Law and Relations 0617
Public Administration 0614
Recreation 0614
Social Work 0452
Sociology 0626
General 0627
Criminology and Penology 0738
Demography 0631
Ethnic and Racial Studies 0628
Individual and Family 0629
Studies 0630
Industrial and Labor 0700
Relations 0344
Public and Social Welfare 0709
Social Structure and Development 0999
Theory and Methods 0453
Transportation 0453
Urban and Regional Planning 0453
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture 0473
General 0285
Agronomy 0475
Animal Culture and Nutrition 0476
Animal Pathology 0359
Food Science and Technology 0478
Forestry and Wildlife 0479
Plant Culture 0480
Plant Pathology 0817
Plant Physiology 0777
Range Management 0746
Wood Technology 0306
Biology 0287
General 0308
Anatomy 0309
Biostatistics 0329
Botany 0332
Cell 0333
Ecology 0339
Entomology 0369
Genetics 0793
Immunology 0410
Microbiology 0307
Molecular 0317
Neuroscience 0416
Oceanography 0433
Physiology 0821
Radiation 0778
Veterinary Science 0472
Zoology 0786
Biophysics 0760
General 0425
Biochemistry 0976
Geochemistry 0976

Geology 0370
Geography 0372
Geophysics 0373
Hydrology 0411
Mineralogy 0345
Paleobotany 0426
Paleontology 0418
Paleozoology 0427
Polymology 0368
Physical Geography 0415
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences 0566
General 0300
Audiology 0992
Dentistry 0567
Education 0350
Hospital Management 0769
Human Development 0758
Immunology 0982
Medicine and Surgery 0564
Mental Health 0347
Nursing 0569
Nutrition 0570
Obstetrics and Gynecology 0380
Occupational Health and Therapy 0354
Ophthalmology 0381
Pathology 0571
Pharmacology 0419
Pharmacy 0572
Physical Therapy 0382
Public Health 0573
Radiology 0574
Recreation 0575

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences 0485
Chemistry 0749
General 0486
Agricultural 0487
Analytical 0488
Biochemistry 0738
Inorganic 0490
Nuclear 0491
Organic 0494
Pharmaceutical 0495
Physical 0754
Polymer 0405
Radiation 0605
Mathematics 0786
Physics 0606
General 0608
Acoustics 0748
Astronomy and Astrophysics 0607
Elementary Particles and High Energy 0798
Fluid and Plasma 0659
Molecular 0610
Nuclear 0752
Optics 0756
Radiation 0611
Solid State 0463
Statistics 0346
Applied Sciences 0984
Aeronautics 0984
Computer Science 0984

Engineering 0537
General 0538
Aerospace 0539
Agricultural 0540
Automotive 0541
Biomedical 0542
Chemical 0543
Civil 0544
Electronics and Electrical 0545
Heat and Thermodynamics 0546
Hydraulic 0547
Industrial 0794
Marine 0548
Materials Science 0743
Mechanical 0551
Metallurgy 0552
Mining 0549
Nuclear 0553
Packaging 0554
Petroleum 0765
Sanitary and Municipal 0554
System Science 0790
Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0794

Psychology 0621
General 0384
Behavioral 0622
Clinical 0620
Developmental 0623
Experimental 0624
Industrial 0625
Personality 0989
Physiological 0349
Psychobiology 0632
Psychometrics 0631
Social 0631

The undersigned hereby recommend to the Faculty of Graduate Studies

and Research acceptance of the thesis,

A Parallel Implementation of the CIS Sea-Ice Motion Tracking

Algorithm for Coarse-Grained Multicomputers

submitted by

Francis Lee

in partial fulfilment of the requirements

for the degree of Master of Computer Science

Thesis Supervisor

Thesis Co-Supervisor

Director, School of Computer Science

Carleton University

January 14, 1997

Abstract

A parallel version of the sea-ice motion tracking algorithm implemented at the Canadian Ice Service, a branch of Environment Canada, was designed. The purpose of the algorithm is to derive the ice motion from time-sequential remotely sensed imagery. This research examined the vector initialization and guided auto-correlation phases of this algorithm because they are the most time consuming. Proposed are parallel algorithms based on the Coarse-Grained Multicomputer model for these phases. Both algorithms were implemented, and their running times analyzed. The speedup observed for the vector initialization phase was close to linear. The guided auto-correlation phase was more difficult to parallelize. Yet, the results still showed a very good speedup for up to 16 processors. This thesis concludes by discussing the possible weaknesses arising with this implementation as the number of processors increases, as well outlines the possibilities for further performance improvement.

Acknowledgments

The completion of this thesis would not have been possible without the assistance of a number of people and organizations.

I would first like to acknowledge my thesis supervisors, Prof. Frank Dehne and Prof. Jörg-Rüdiger Sack for their valuable commentary and guidance throughout this research project. Without their support, this thesis might well have fallen short of its objectives.

Financial support for this project was provided by Natural Sciences and Engineering Research Council of Canada and Almerco Inc. Also, I would like to thank the Canada Centre for Remote Sensing, Natural Resources Canada for providing the imagery for testing the algorithm.

Technical assistance in overcoming hardware/software system problems was provided by Mr. Peter Choynowski.

Finally I would like to thank my wife Lillian for her unconditional support over the course of this thesis project. The patience and encouragement provided to me by my family was invaluable.

Table of Contents

Abstract	iii
Acknowledgments	iv
Table of Contents	v
List of Tables	vii
List of Figures	viii
Acronyms and Abbreviations	ix
1 Introduction	1
1.1 Background	1
1.2 Problem	4
1.3 Overview of Results	5
1.4 Related Applications	5
1.5 Overview of the Document Structure	6
2 Parallel Model of Computation	7
2.1 Bulk-Synchronous Parallel Model	7
2.2 Coarse-Grained Multicomputer Model	8
3 The Sequential Sea-Ice Motion Tracking Algorithm (IMTA)	10
3.1 Background	10
3.2 Detailed Description	11
4 Parallelization of the Ice Motion Tracking Algorithm	20

4.1	Overview	20
4.2	Parallelization of those IMTA Phases that are Based on Low-Level Image Processing	22
4.2.1	Overview	22
4.2.2	Overlap Mapping Versus Non-overlap Mapping	23
4.3	Parallel 2-Dimensional Correlation Matching	25
4.3.1	Overview	25
4.3.2	Load Balancing	28
4.3.2.1	Uniform Partitioning	28
4.3.2.2	Static Partitioning	29
4.3.2.3	Dynamic Partitioning	30
4.3.2.4	Selection of Partitioning Schemes	31
4.3.3	The Vector Initialization Phase	32
4.3.3.1	Overview	32
4.3.3.2	Uniform Partitioning Algorithm	33
4.3.3.3	Vector Initialization Algorithm	33
4.3.4	The Guided Auto-Correlation Phase	35
4.3.4.1	Overview	35
4.3.4.2	Static Partitioning Algorithm	36
4.3.4.3	Guided Auto-Correlation Algorithm	38
5	Implementation of the Vector Initialization and Guided Auto-Correlation Phases	40
5.1	Overview	40
5.2	Create Image Rotation Look-Up-Table	42
5.3	Read Input Disparity File	44
5.4	Vector Initialization Phase Routine	46
5.5	Guided Auto-Correlation Routine	48
5.6	Output Updated Disparity File	51
5.7	Implementation Issues	53
6	Experimentation Results	55
6.1	Overview	55
6.2	Timing Results for the Vector Initialization Phase	56
6.3	Timing Results for the Guided Auto-Correlation Phase	57
7	Conclusion	62
	References	64

List of Tables

6.1	Total Processing Time of the Vector Initialization Phase of the 2-D Correlation Module	56
6.2	Average Total Processing Time of the Guided Auto-Correlation Phase of the Parallel 2-D Correlation Module	58
6.3	Average Total Communication Time of the Guided Auto-Correlation Phase of the Parallel 2-D Correlation Module	59
6.4	Sample Local Processing Time of a 2-Node and 4-Node Results	60

List of Figures

1.1	Data Acquisition of Canadian Ice Service	2
1.2	Two Satellite Passes Cover the Same Area	3
2.1	Distributed Memory Parallel Systems	7
3.1	Overview of the Ice Motion Tracking Algorithm	12
3.2	Filtering and Subsampling of Image	13
3.3	Pixel Cornerity	14
3.4	Relationship between Partitions, Grid Cells, Pixels and Candidates within IMTA	15
3.5	2-Dimensional Area Correlation between Two Areas	16
3.6	Consistency in Motion	16
3.7	Error Detection Module	17
3.8	Sample Result of IMTA	19
4.1	IMTA Modules Based on Low-Level Image Processing	22
4.2	Overlap Mapping	23
4.3	IMTA Modules Based on Parallel 2-D Correlation Matching	26
4.4	Processing Element Local Memory Map	31
4.5	Detailed Design of the Vector Initialization Phase	32
4.6	Detailed Design of the Guided Auto-Correlation Phase	35
4.7	Static Load-Balance Partitioning Scheme	37
6.1	Processing Time Results of the Vector Initialization Phase	57
6.2	Processing Time Results of the Guided Auto-Correlation Phase	58
6.3	Average Total Communication Time versus Number of Processors in Guided Auto-Correlation Phase	60

Acronyms and Abbreviations

AVHRR	Advanced Very High Resolution Radiometer
BSP	Bulk-Synchronous Parallel
CCRS	Canada Centre for Remote Sensing
CGM	Coarse-Grained Multicomputer
CIS	Canadian Ice Service, Environment Canada
DMSP	Defense Meteorological Satellite Program
ERS	European Remote Sensing Satellite
IMTA	Ice Motion Tracking Algorithm
ITS	Ice Tracking System
LAM	Local Area Multicomputer
LUT	Look Up Table
MPI	Message Passing Interface
NOAA	National Oceanic and Atmospheric Administration
PRAM	Parallel Random Access Machine
SAR	Synthetic Aperture Radar
SLAR	Side-Looking Airborne Radar
SSM/I	Special Sensor Microwave Imager

1. Introduction

1.1 Background

The Canadian Ice Service, Environment Canada (CIS) is a Canadian Government Agency that is responsible for providing ice related information about Canada's offshore area to marine clients, in particular the Canadian Coast Guard. Remote sensing data in the form of imagery from a variety of sources, including NOAA AVHRR, SSM/I, ERS-1 and aircraft SLAR [26], are provided to the ice/oceans analysts who interprets the environmental conditions. The products include a variety of image, chart and alphanumeric information. Figure 1.1 illustrates the flow of image data from collection to analysis and distribution. The data is captured by satellite and downloaded to the Ground Receiving Station at Gatineau, Quebec. The signal data is processed and converted to imagery at the Ground Receiving Station, then using a dedicated T1 link is transferred to the Canadian Ice Service (CIS) located in Ottawa, Ontario. After the images arrive at CIS, a series of automated ice interpretation processes are applied and products are generated. The automatically generated products are further analyzed by the ice/oceans experts to ensure their accuracy. The resulting products can then

be delivered to the Coast Guard ice breakers or the ship based helicopters. The entire operational process requires a turnaround time of less than 6 hours. With the launch of the RADARSAT satellite in 1995 [25], CIS is acquiring a large volumes of data which required a faster processing facility in order to keep up with the incoming data volume.

Figure 1.1 : Data Acquisition of Canadian Ice Service

The Sea-Ice Motion Tracking Algorithm (IMTA) is one of the automatic processing steps employed at CIS. The purpose of the IMTA is to calculate the ice displacements from two coincident (in geographic space) image pairs. The resultant ice displacement field helps the

ice analysts forecast ice conditions for the upcoming days. Figure 1.2 shows two sample satellite passes over the same area. The area covered by the coincident scenes can be input to the IMTA in order to track the ice motion between scenes and calculate the ice drift.

Figure 1.2 : Two Satellite passes Cover the Same Area (Area of Interest)

The most time consuming portion of the calculation is the matching of the ice floes in the two images. The coverage of a RADARSAT swath is approximately 5 times the width of previously available imagery for ice tracking, i.e., Star-2 airborne SAR, ERS-1 [10,25]. The resulting image sizes are approximately 25 times larger. The current implementation of the IMTA requires about 30 minutes to process an image pair. By projecting these times for a RADARSAT image pair, 12.5 hours will be necessary to process the data. Thus, in order to

provide timely reports to the Canadian Coast Guard, more efficient processing methodology are required to meet their needs.

1.2 Problem

In order to increase the processing power of the IMTA, parallelizing the algorithm is the logical step to improve its speed and adapt it to current and future high performance parallel hardware architectures [2,17]. Most existing multicomputer machines consist of a set of p state-of-the-art processors, each of which has a large amount of local memory. We refer to these machine as Coarse-Grained Multicomputer Systems. These processors are connected by some interconnection network. In order to utilize these existing machines, the algorithm designed must be *scalable* [3,5], that is, the speed of the algorithm should relate directly to the number of processors and the local memory available. The advantage of this is that the number of processors and local memory can be increased when additional processing power is required. The bottleneck of parallel processing, in general, is the relatively large communications overhead. This overhead can be attributed to three sources: (1) the message overhead, (2) the latency due to the interconnection topology, and (3) bandwidth limitations. Hence, the number of communication rounds of the scalable algorithm should be minimized. The goal of this thesis is to apply the coarse-grained parallel algorithm approach to the IMTA problem.

1.3 Overview of Results

A parallel version of the IMTA was designed and is described in this thesis. The most time consuming module, the two-dimensional cross-correlation module, was implemented. This module is divided into two phases, the vector initialization phase and the guided auto-correlation phase. The vector initialization phase is easy to parallelize and hence a close to linear speedup resulted. However, the guided auto-correlation phase is harder to parallelize and the implementation results shows a very good speedup when the number of processor is less than 16. It is suspected that the speedup will start to decrease as the number of processors grows considerably beyond that. We analyzed the reason and examined possibilities for further improvement.

1.4 Related Applications

The technique used to parallelize the IMTA can be modified to apply to other applications. In particular, applications which use a combination of traditional image processing methods and computational geometry algorithms are good candidates for this technique. One of the main advantages of this parallelization technique is that it will improve the algorithms' performance without having to re-design the entire algorithm. Some examples are as follows:

Real-life operational applications:

- Ship detection
- Oil spill detection
- Ice motion prediction model
- Flood area detection

Digital signal and image processing applications:

- Connected-Component [3]
- Stereo-matching, or feature-based range estimation [28]
- Edge detection
- Object thinning

1.5 Overview of the Document Structure

Section 2 of this thesis describes the computational models used for this parallelization process. Section 3 describes the sequential IMTA in detail and Section 4 describes the parallelization of the IMTA. Section 5 gives the parallel IMTA implementation details. The timing results are analyzed in Section 6 and Section 7 concludes the thesis.

2. Parallel Model of Computation.

2.1 Bulk-Synchronous Parallel Model

Valiant [30] introduced the Bulk-Synchronous Parallel (BSP) model in 1990. The BSP approach simulates Parallel Random Access Machine (PRAM) algorithms which are optimized to be used on distributed memory parallel systems, see Figure 2.1. Valiant points out that an architecture independent parallel algorithm design must utilize local computations and minimize global operations. The BSP computer consists of a set of processors and memory modules

connected by a point-to-point message passing router. Parallel computation under BSP is divided into a sequence of supersteps separated by barrier

Figure 2.1 : Distributed Memory Parallel Systems

synchronizations. Each superstep involves local computation and communications steps [11,30,31,32]. The model involves 4 parameters:

- n : the size of the problem
- p : the number of processors available
- L : the minimum time between synchronization steps
- g : the number of messages of unit size that can be delivered by the router.

Let λ be the total number of supersteps for the algorithm, then the total computation cost is:

$$T_{comp} = \sum_{i=1}^{\lambda} \max\{L, t'_1, \dots, t'_p\}$$

where the t'_j is the number of basic computation operations performed by processor j on the i -th superstep. In addition, assuming that processor p_j receives messages of lengths $r'_1, \dots, r'_j$ and sends messages of lengths $s'_1, \dots, s'_j$ during the i -th superstep, the total communication cost of the algorithm is:

$$T_{comm} = \sum_{i=1}^{\lambda} (\max\{L, g(\sum_{u=1}^{j'} r'_{iu} + \sum_{u=1}^{j''} s'_{iu})\})$$

2.2 Coarse-Grained Multicomputers Model

A variation of the BSP model, referred to as the *Coarse Grained Multicomputer (CGM)*, was introduced by Dehne *et al.* [5] in 1993. The model is a specialized case of the BSP which also consists of a set of p processors with $O(n/p)$ local memory on each processor, connected

by some interconnection network. A CGM algorithm consists of an alternating sequence of computation rounds and communication rounds separated by barrier synchronizations. A computation round is equivalent to a computation superstep in the BSP model. A communication round is a single h -relation where $h \leq n/p$. Therefore, the total communication cost T_{comm} of a CGM algorithm with λ communication rounds is simply, $T_{comm} = \lambda H_{n,p}$ where $H_{n,p}$ is the cost of each communication round. The unique characteristic of the CGM model is that the local memory size is considerably large, e.g., $O(n/p) > O(p)$ or $O(n/p) > O(p^2)$ [3,5,6,7]. The CGM model works well for algorithms where the overall computation speed is considerably larger than the overall communication speed. This type of configuration is found in most of the current multicomputer machine architectures. Another characteristic of the model is that it minimizes the message overhead associated with sending a single message regardless of its length. These flexibility provide the designer a simple way to maximize the throughput of their algorithms.

The parallel Sea-Ice Motion Tracking Algorithm (IMTA) examined in this study has been designed using a technique for designing scalable parallel matching algorithms for CGM architectures, which are optimal or at least efficient for a wide range of data to processor ratios. Another feature characteristic of this design technique is that most of the existing uni-processor modules, which have been fully optimized, can be reused.

3 The Sequential Sea-Ice Motion Tracking Algorithm (IMTA)

3.1 Background

Filey and Rothrock [8,9] described a sea-ice motion tracking algorithm, based on two-dimensional area correlation of sample pixel intensity arrays, that will derive ice motion from time-sequential remotely sensed imagery. The weakness of the design is that the algorithm can only work on a predefined small subset of the Synthetic Aperture Radar (SAR) imagery. Hirose and McNutt [15] proposed a modified version of the algorithm to overcome limitations encountered in more difficult ice tracking situations. In addition, the possibility of tracking sea ice in an operational environment was addressed. An Ice Tracking System (ITS) based on this algorithm has been developed as part of a joint project between the Canada Centre for Remote Sensing (CCRS) and the Canadian Ice Service (CIS) [16,26]. The ITS is used operationally at the CIS to generate sea-ice kinematic information over navigable Canadian waterways. In an operational environment the Ice Tracking System must efficiently extract ice motion from sequential image scenes with minimal errors, and operate in different

ice regimes using data acquired from a variety of sensors without human intervention. Several remote sensing satellites, including NOAA AVHRR [14] (Advanced Very High Resolution Radiometer), aircraft radar [4,16] (Synthetic Aperture Radar and Side Looking Airborne Radar), and DMSP SSM/I [1] (Special Sensor Microwave Imager), provide information for operational ice analysis and forecasting at CIS. Automated tracking of sea ice using the ITS has successfully demonstrated its functionality and importance by using these sensors.

With the launch of the RADARSAT satellite in November 1995, large volumes of data was acquired by the CIS for their operational uses. RADARSAT is Canada's first earth observation satellite, and features a ScanSAR wide mode beam [10] that can image an area approximately 500 km x 500 km per scene. The current implementation of the tracking algorithm requires improvements in its processing strategy in order to keep up with the large volumes of incoming data. Parallelizing the algorithm is one of the logical steps towards improving the algorithm's performance, and to adapt to the current processing requirements and future hardware architectures.

3.2 Detailed Description

The Ice Motion Tracking Algorithm (IMTA) calculates ice displacements from time

sequential imagery in a 5-step procedure; each step is referred to as a phase [14]. Phase 1 reduces the resolution of the input images. Fewer pixels mean a faster processing time in the following phases. Phase 2 automatically identifies a small number of initial matches. Phase 3 uses the initial matches found in Phase 2 to guide the search for more matches. Phase 4 uses the results of Phase 3 to guide a more detailed search for matches. Finally, in Phase 5 the resolution of the images is returned to the original level providing more accurate results.

An overview of the data-flow and processing scheme of the ice motion algorithm is shown in Figure 3.1. A coarse resolution image is created by low pass filtering and sub-sampling of the original input image as shown in Figure 3.2. The resulting coarse resolution image is used as input into phases 2 through 4. The number of reductions the input image goes through will depend on the image quality, pixel resolution and the ice characteristics of the

Figure 3.1 : Overview of IMTA

imaging area. In general, the pixel resolution of the working level image is between 500 and 1000 metres.

Before the end of Phase 1, the coarser resolution (or working level) images are high-pass

Figure 3.2 : Filtering and Subsampling of Image

filtered. The filter is designed to highlight corners of floes and ridges on the ice which then provide candidate locations in the imagery [13]. The purpose of identifying these locations is to give the matching module a higher probability of finding pairs by matching corners and/or edges of the ice from the two scenes. In addition, the searching portion of all the phases only has to search for the locations that have higher corner values, hence reducing the area search time. Figure 3.3 illustrates a high cornerity versus a low cornerity pixel value.

The purpose of Phase 2 is to find a few initial matching locations evenly spread over the coarse resolution image pair in a timely fashion. Three matching strategies are possible, one based on two-dimensional area correlation, one based on feature matching [18], and a third

Figure 3.3 : Pixel Cornerity: (a) has higher cornerity value than (b).

based on manually selecting matches. Only the area correlation matching strategy was examined in this study.

A two-dimensional area correlation involves comparing an array of pixel intensities surrounding a candidate point from the reference image (the first image) with an array of pixel intensities from a match image (the second image) scene. A 9×9 pixel array was identified as the optimal window size in previous studies [1, 14,15,16,20]. A poor correlation value corresponds to 0.0 and an exact match corresponds to 1.0. If the arrays are similar, a high correlation results. A detailed methodology for tracking sea ice using this technique is described by Hirose and McNutt [15].

To implement the area correlation technique efficiently, images are partitioned into non-overlapping blocks, or partitions. Each partition is then further subdivided into grid cells. Each cell within that grid is made up of a matrix of pixels. Figure 3.4 illustrates the

relationship between partitions, grid cells and pixels. The image is divided into four partitions. Each partition is subdivided into nine grid cells each made up of nine pixels. Each grid cell can store one candidate, or match. Thus, match points must be at least one grid cell apart. The match candidates are identified using the high-pass filter, described above.

Figure 3.4 : Relationship between Partitions, Grid Cells, Pixels and Candidates within IMA

In Phase 2 (the initialization phase), a 9×9 pixel window centered at each candidate point is cross-correlated with a corresponding window in the match image. Figure 3.5 illustrates the matching strategy of the initialization phase. The maximum search area on the match image is defined as an input parameter. Within the maximum search area on the match image, each candidate point is cross-correlated with the reference window. In addition, to account for the internal ice floe rotation, each potential candidate window on the match image is rotated in 32 directions ($= 360 \text{ degree} / 11.25 \text{ degree increments}$) to obtain the best rotation angle and the best match location within the area.

Figure 3.5 : 2-Dimensional Area Correlation between Two Areas

Phase 3 propagates the match field by finding a match for locations over the entire image at a fine spacing. Beginning at the locations initialized in Phase 2, a match for each of its neighbours on the reference scene is

searched for on the second image. In

order to reduce the number of locations to test on the second image, based on the consistency of the ice motions, the assumption was made that the neighbour tends to move in the same direction with a similar displacement as the initialized locations, see Figure 3.6. This heuristic is used to guide the search

Figure 3.6 : Consistency in Motion

for matches of the neighbours. For example, in Figure 3.6, to determine the search area of location B, the vectors at location A and location C are projected onto the location B to define the search area. A small search area centered on the predicted location is cross-correlated with the reference location and the best match (highest correlation) is compared with an acceptance threshold. If it is greater than the threshold it is assigned as a matching pair, otherwise no match is made. The process is repeated until no new matches are found. A high correlation threshold is used to inhibit the propagation of incorrect matches that may have occurred in the initialization phase.

Figure 3.7 : Error Detection Module:
(a) Forward Error Detection.
(b) Backward Error Detection.

At the end of Phase 3, an attempt is made to detect and eliminate incorrect matches. It is assumed that locations in a local neighbourhood will move in a similar way (in both direction and magnitude). If a match vector does not move like the local motion field, all the vectors within the local neighbourhood are removed from the matched location's list, see Figure 3.7(a). Conversely, the vectors in the local neighbourhood must be come from the same area, otherwise

all the vectors in that neighbourhood are removed, see Figure 3.7(b).

Phase 4 is identical to Phase 3 but the acceptance threshold is lowered enabling more locations to be matched. Phase 5 refines the positional accuracy of the matched pairs by finding their corresponding locations on each of the higher resolution images. Figure 3.8¹ shows a sample output of the IMTA. The dot on each vector indicates the start position, and the vector represents the ice motion of the two-time sequential images within a 1½ hour time period.

¹ Image courtesy of Canada Centre for Remote Sensing, Natural Resources Canada.

Figure 3.8 : Sample Result of IMTA

4 Parallelization of the Ice Motion Tracking Algorithm

4.1 Overview

Because of the reliability of the current design of the CIS Ice Motion Tracking Algorithm (IMTA), the original processing strategy was kept. Parallelizing Phases 1 and 5 of the algorithm is relatively simple by partitioning the image into p (number of processors) partitions, or segments, and each processor can work on its local partition with the appropriate window filters, i.e., the sub-sampling filter, high-pass filter, and the error detection filter. This filtering technique is classified as the “Low-Level Image Processing Operator” in the classical image processing application [21,22,24]. An advantage of this partitioning design is that no processor-to-processor communications are required.

In Phases 2, 3, and 4 the two-dimensional cross-correlating of the two images to find the matching floes is the most computational intensive (and most time consuming) portions of the algorithm. This study concentrated on designing a parallel version of the 2-D cross-

correlation module.

In the vector initialization phase (Phase 2), each processor holds two segments of the image pair in local memory: (1) one partition of the reference image, and (2) the maximum search area of the match image related to the partition of the reference image. The size of this maximum search area is based on the maximum predicted ice drift. The sequential implementation of the matching algorithm is then executed on each processor. This technique, again, requires no processor-to-processor communications.

The partitioning scheme for the guided auto-correlation phases (Phases 3 and 4) are similar to the vector initialization phase except that one grid cell along the edges of the partitions are overlapping with their neighbouring processors. This overlapping area is used for information exchange and communication during the propagation of the vectors. When the local propagation (local computation) of the vectors is done (using the original highly optimized matching module), the vector information along the edge of each partition is sent to its neighbour processors. When a processor receives information from a neighbour, it updates the information for the edge cells of the partition by keeping the highest correlation value of the grid cell. Then the local propagation process starts again until no new vectors found. The following subsections describe this parallelization strategy in more detail.

4.2 Parallelization of those IMTA Phases that are Based on Low-Level Image Processing

4.2.1 Overview

The goals of Phase 1 and Phase 5 of the IMTA are: (1) to reduce the size of the image to a workable level, (2) to reduce the noise of the image and enhanced the features of the working level image, and (3) to outline the potential corners and edges of the ice floes. The error-detection routine in Phase 3 and Phase

Figure 4.1 : IMTA Modules (Shaded) Based on Low-Level Image Processing

4 is used to eliminate the incorrect matches generated in the cross-correlation routine by employing the local neighbouring information. All these goals can be met by using low-level image processing techniques developed over the last twenty years. The main objective of low-level image processing is to enhance image features and to make it more suitable for subsequent processing, and for human analysis and understanding [27,29]. The low-level digital image processing algorithms involve different computational aspects and produce different results, but all of them require an enormous amount of computational power. Fortunately, these computationally intensive operations can be easily parallelized using a

window-based operator in the local partitions. With local operators, the value of a processed pixel depends only on its value and on values of the pixels in its neighbourhood. These operators, such as the mean filter or high-pass filter, use different neighbourhoods (windows), but in general have a square or rectangular geometry. The shaded modules in Figure 4.1 can be parallelized by using this technique. Section 4.2.2 describes two popular mapping methods which can be used to map the input image to the multicomputer's processor local memory.

4.2.2 Overlap Mapping Versus Non-Overlap Mapping

Assume that p is the number of processors, $M \times M$ is the size of the local window filter, and $N \times N$ is the size of the image. As far as mapping the $N \times N$ image is concerned, there are two approaches for the parallelization of local filters or subsampling techniques [27]:

(1) Overlap Mapping:

each processor p stores a partition of the image plus a neighbourhood so that it can compute all its partition values

Figure 4.2 : Overlap Mapping:
(a) Partition for which processor i is responsible
(b) Actual subimage that processor i is required to store

without communication. For example, the processor responsible for the $n/\sqrt{p} \times n/\sqrt{p}$ partition shown in Figure 4.2(a) actually stores the $(n/\sqrt{p}-m) \times (n/\sqrt{p}+m)$ subimages, shown in Figure 4.2(b).

- (2) Non-overlap Mapping: The image is decomposed into $n/\sqrt{p} \times n/\sqrt{p}$ blocks. Each processor only gets the image block corresponding to its partition. When the processor requires extra information from its neighbours, the processor requests the information from its neighbours through inter-processor communication.

The advantage of the overlap mapping technique is that no inter-processor communication is required. The only disadvantage is that each processor requires more local memory to store the information for processing than when the non-overlap mapping technique is used. However, this large local memory size still fits under the CGM model, i.e., $O(n^2/p)$. It is also important to note that the communication overhead in the local filtering and sub-sampling problem is small relative to the computing cost. When the overlap mapping is used, $O(NM\sqrt{p} + PM^2)$ additional data is stored in the local nodes (i.e., in addition to the transfer N^2 image values). However, since the host can send data to several nodes in parallel, the overhead penalty is not as severe. While the same amount of data has to be transferred between processors when the non-overlap mapping is used, the p processors can work in parallel so that the transfer time is approximately equal to that for the transfer of $O(NM\sqrt{p} + M^2)$ data. In either case, this overhead is expected to be small compared with the time required for the local computing, which is $O(N^2M^2/P)$, to be carried out by each processor.

For small window sizes, the non-overlap method is significantly slower than the overlap method. For larger window sizes the difference in run time is not significant [27]. Much of the difference in the run time is attributable to the following:

- (1) The program for the non-overlap case is considerably more complex, and has greater overhead than that for the overlap case.
- (2) The data transfer rate from the host to the node is much higher than that between nodes.
- (3) For larger window sizes the computation time significantly dominates the communication time.

Therefore, the overlap mapping processing strategy is recommended when implementing the filtering and sub-sampling module of the IMTA.

4.3 Parallel 2-Dimensional Correlation Matching

4.3.1 Overview

This section describes the parallel two-dimensional correlation matching technique that can be used in the shaded module in Figure 4.3. The input parameters to the module are as follows:

Description	Type
Reference (first) image name in sequence	Character String
Match (second) image name in sequence	Character String
Match Image candidate map file name	Character String
Input Disparity file name	Character String
Output Disparity file name	Character String
Correlation threshold	Real number
Correlation window size	Integer
Maximum ice displacement	Integer
Grid size of the partition	Integer
Rotation flag (1=rotate, 0=fix)	Integer
Counter clockwise rotation number (0-31)	Integer
Phase Option: Phase 2 = 1, Phase 3 and Phase 4 = 2	Integer

In order to have a highly parallel algorithm, the design of the 2-D area correlation matching module must have the following properties:

- (1) It should minimize the inter-processor communications.
- (2) The total processing time should linearly decrease as the number of processors increases.

In addition, the highly optimized sequential module should be reused if possible to reduce the development cost.

**Figure 4.3 : IMTA Modules (Shaded)
Based on the Parallel
2-D Correlation Matching**

The following issues are addressed during the implementation process to improve the processing speed [19]:

- (1) Partition/Paging: The ice motion displays a high level of consistency and locality.
- (2) Over-partitioning: Load balancing on the partitions to allow an equal workload on each processor.
- (3) Message Grouping: To reduce the communication rounds, all small messages are grouped along the edges of the partitions.

The study by Filey and Rothrock [8] concluded that in most of the cases, the ice motions have a high level of consistency and locality. This characteristic must be taken into account for the design of the parallel version of the correlation module. Moreover, when considering the local processing time for an individual processor, balancing the workload between the processors is another major factor that must be considered to improve the overall timing results. For example, in a 4-node network, if three processors finished their local processing jobs in 10 seconds while the other one processor requires 100 seconds, all three processors have to wait 90 seconds until the last processor finishes its local processing. Therefore, each processor, within the network must be balanced in a way that all processors can finish the local task in almost the same time. The next section discusses the load-balancing scheme, which is a standard parallel processing issue.

4.3.2 Load Balancing

When considering the workload of each processing region, the computational load represented by each processor is proportional to the image area. Since the potential distribution of match candidates in the image is non-uniform, and most input images after geocoding, will have an area of zero fill pixels (dark area), see the example in Figure 3.8. Furthermore, sometimes a large portion of the image may be covered by land where no ice motion should be detected. In all of these cases, a load balancing technique is needed to utilize every processor in a parallel system effectively. Three load-balancing techniques [28] have been explored: (1) uniform partitioning, (2) static partitioning, and (3) dynamic partitioning.

4.3.2.1 Uniform Partitioning

If the vectors are uniformly distributed over the image plane, then a fixed uniform size partitioning scheme would be optimal. Given a p -processor machine, the image plane would be divided into p equal sized partitions; one partition per processor. No explicit load balancing would be necessary to utilize each processor equally due to the uniform distribution of features in the image plane.

4.3.2.2 Static Partitioning

If the vectors are not distributed uniformly (which is most often the case), then a load-balancing technique is needed to make efficient use of every processors in the system. The major workload of each processor in the 2-D area correlation module is the computation performed for each candidate. The load-balancing function of this module can be broken down into two parts: (1) the workload required to scan a grid cell with no matches as neighbours, W_G , and (2) the workload required to process a grid cell that has matches (vectors) as its neighbours, W_C .

$$\text{Workload} = \text{Function}(W_G, W_C)$$

Assume that the computation cost of the regular grid cell is 'a' and the cost of propagating the vector is 'b'. Then, we have,

$$\begin{aligned} \text{Cost of regular grid cell} &= a \\ \text{Cost of the grid cell with vector} &= a + b \end{aligned}$$

by normalizing the above two values by 'a', we have:

$$\begin{aligned} \text{Cost of regular grid cell} &= 1 \\ \text{Cost of the grid cell with vector} &= 1 + b/a \\ &= 1 + k \end{aligned}$$

where $k = b/a = \text{ratio between } a \text{ and } b$.

Since the processing time for the grid cell with no vector is much less than the grid cell with a vector (i.e., $b \gg a$), we will set the value of k to a large value.

The static partitioning scheme, which will be described in detail later, can be used if the number of vectors per processor does not change too rapidly between different iterations of the entire guided auto-correlation in Phase 3 and Phase 4; otherwise a load imbalance will appear in later iterations.

4.3.2.3 Dynamic Partitioning

If the scene content changes rapidly, such that the number of vectors per processor fluctuates greatly from one iteration to the next, then the estimate function described in Section 4.3.2.2 is not accurate enough to represent the computation time required by the processor. To correct such cases, a higher order model of the computation time may be formulated considering the dynamic factors. A simple method is to have a controlling processor distribute the data to slave processors from a task queue of processors. The processors are assigned new data as they finish processing their current data. This method of dynamic allocation is practical only if the communication network between the controlling task scheduler and the processing element does not saturate. While this approach may reduce the load imbalance, it creates considerable overhead for re-distributing the work. This overhead

cost could be larger than the gain from the improved load balance.

4.3.2.4 Selection of Partitioning Schemes

In most parallel machines, the bottleneck is in the communication overhead [23]. In order to reduce the local processing time, the uniform partition scheme was used in the implementation of the vector initialization phase (Phase 2) because there is no information to predict the presence of vectors. The static partitioning scheme was used in the implementation of the guided auto-correlation phases (Phase 3 and Phase 4) because: (1) it reduces the load imbalance for the partition caused by the non-uniformly distributed vectors, and (2) it avoids all the heavy inter-processor communications.

To satisfy the requirements of the CGM model, the number of communication rounds should also be minimized. Therefore, all the communication between processors and information exchange along the edges can be grouped together into a large message. This inter-processor communication can take place after the local processing is done,

Figure 4.4 : Processing Element Local Memory Map

exchanging all information in one communication round. The overlap mapping processing scheme was also used in all phases to minimize the number of messages. Figure 4.4 shows the memory allocation of each processing element.

4.3.3 The Vector Initialization Phase

4.3.3.1 Overview

This section outlines the design of the Vector Initialization Phase (see Figure 4.5) of the parallel 2-D area correlation module. Detailed descriptions of the individual algorithms are also presented. In general, the purpose of the parallel 2-D area correlation module is to find the matching locations on two time sequential images by using the candidates found on the coarse, filtered reference and match image.

Figure 4.5 : Detailed Design of the Vector Initialization Phase

4.3.3.2 Uniform Partitioning Algorithm

In the vector initialization phase, each candidate point on the reference image has equal probability of finding a matching point on the match image. There is no additional information provided by the previous phase on what the workload will be for each partition. Therefore, the uniform partitioning scheme was used for this phase. In this implementation of the uniform partitioning scheme, the number of partitions must be equal to a power of 2. The algorithm initializes the entire image as one partition. Each partition of the image is then recursively split into two sub-partitions until the number of the partitions is equal to the number of processors available.

- Step 1: Partition the grid cell map into two halves by splitting the longer edge of the map.
- Step 2: If the number of rows/columns is not divisible by 2, take the FLOOR for the 1st half and the rest for the 2nd half.
- Step 3: Repeat the above two steps for each sub-partition until the total number of partitions is equal to the number of processors.

4.3.3.3 Vector Initialization Algorithm

The following explains the initialization phase of the parallel 2-D area correlation module.

First, we use the uniform partitioning scheme described above, to partition the image into p

(number of processors) partitions. Second, according to the partitioning results, we read all the necessary image information and data into local memory. Third, the 2-D area cross-correlation process is started to search for matching locations on each candidate on the match image until all the vectors are found in the local partition. The advantages of this method are that no inter-processor communication is required and that the original optimized sequential code for matching module can be reused.

- Step 1: Partition the grid cell map according to the number of processors using the uniform partition scheme (the number of processors is assumed to a power of 2, i.e., hypercube like network)
- Step 2: Read the i^{th} partition of the reference image into processor i^{th} local memory.
- Step 3: Read the disparity file grid cell information into memory.
- Step 4: Determine the maximum ice drift (search area) area for the match image.
- Step 5: Read the corresponding maximum ice drifts area from the match image into memory.
- Step 6: Read the corresponding ice drifts match file candidate file into memory.
- Step 7: For each grid cell with a candidate,
 - Determine the search area of the match image for the current grid cell.
- Step 7.1: For each grid cell on the match image within the search area,
 - If a candidate exists then
 - Cross-correlated the two areas with rotation (if set to ON).
 - Save the maximum correlation value and destination x, y-location
 - End-if
- End-for
- If the maximum cross-correlation value is greater than the given correlation threshold, update the disparity grid cell with the destination x, y-location,

correlation value, and set the grid cell candidate flag to positive value (means that a match point is found in this grid cell).

End-for.

Step 8: Output disparity grid cell records.

4.3.4 The Guided Auto-Correlation Phase

4.3.4.1 Overview

The purpose of the guided auto-correlation phase is to propagate the neighbour vectors of matches found in the initialization phase at in a finer grid spacing, see Figure 4.6. The static partitioning scheme was used in this phase because the vectors computed in the previous phase are not uniformly distributed across the image.

Figure 4.6 : Detailed Design of the Guided Auto-Correlation Phases

4.3.4.2 Static Partitioning Algorithm

This section describes the static partition scheme in a step-by-step fashion. A workload is assigned to each grid cell based on the vectors found in the initialization phase. This workload is dependent on two factors: (1) the cost of processing the grid cell with the vector, and (2) the local memory size. In the beginning, the cost to process a grid cell which has a vector is set to dominate the workload by setting k to a large value. If any partition exceeds the local memory size of a processor, we use a binary search to find the maximum k such that all partitions fit into local memory. The algorithm splits the grid cell map according to the workload weight of the partitions. Between the neighbouring partitions, one grid cell overlap is maintained. The splitting process recursively divides each partition into two sub-partitions until the number of partitions equals to the number of processors. See Figure 4.7 for an illustration.

- Step 1: Assign a large value (k) to the vector map if a vector exists in the grid cell.
- Step 2: Partition the map into two halves by splitting the longer edge of the map. The weight difference between the two partitions must be minimized.
- Step 3: Maintain one row/column (depending on the split orientation) overlap between two partitions.
- Step 4: Repeat Step 2 and Step 3 until the number of partitions is equal to the number of processors.
- Step 5: If any partition size is over the limit of the local memory, use the binary search method to find the largest k value for which the local memory can be store the partition. Goto Step 2.

Figure 4.7 : Static Load-Balance Partitioning Scheme. The shaded area represents the overlapping portion between processors. The number outside the grid is the sum of the row/column which are used by the splitting process

4.3.4.3 Guided Auto-Correlation Algorithm

This section outlines the procedure for the guided auto-correlation phases of the parallel 2-D area-correlation module. The vectors found in Phase 2 are used as a guide to the search area in each grid cell. This propagation process is repeated until no new vectors are found in the local partition. All grid cell information along the edges are then exchanged with the neighbouring processors. Each processor keeps the highest correlation value and its vector destination location in the grid cell. This new information can lead to another iteration of the vector propagation. The entire local propagation and information exchange processes are repeated until no new information is changed between all the processors. The following describes the steps of this algorithm:

- Step 1: Read the disparity file and create a blank cost matrix.
- Step 2: If there is a vector, set the cost of this particular cell of the matrix to $1+k$, otherwise, set it to 1 .
- Step 3: Partition the image by using the static partitioning scheme and use the cost matrix as a reference.
- Step 4: Read the i^{th} partition of the reference image into i^{th} processor's local memory.
- Step 5: Determine the maximum ice drift area for the match image.
- Step 6: Read the corresponding ice drift area in the match image into memory.
- Step 7: For each grid cell with a candidate OR a vector,
 - Determine the match image search area for the current grid cell guided by using its immediate 8 neighbouring vectors (if they exist).
- Step 7.1: For each area in the match image search area:

If the bitmap shows that a candidate exists on the match image then

Cross-correlate the two areas.

Keep only the maximum correlation value and destination x,
y location.

End-if

End-for

If the cross-correlation value is greater than the threshold, update the disparity
grid cell with the destination x, y-location, and correlation value, and set the
grid cell candidate flag to a positive value (which means that a match point is
found in this grid cell).

End-for.

- Step 7.2: Repeat this step, until no NEW vectors are added to the disparity file.
- Step 8: Exchange disparity file information along the overlap area between partitions.
- Step 9: If the destination points of the edge cell along the edges are different, save the one
with the highest correlation threshold.
- Step 10: After the information is exchanged, for each partition, if any of the vectors along
the edges changed, go to the *Step 7*. Otherwise, end of routine.

5. Implementation of the Vector Initialization and Guided Auto-Correlation Phases

5.1 Overview

The Message Passing Interface (MPI) was used as the communications library between processors in this implementation. The reason is that the MPI is a new library specification for message-passing which is proposed as a standard by a broadly based committee of vendors, implementors, and users [12]. It is designed for high performance on both massively parallel machines and on workstation clusters. It is an attempt to collect the best features of many existing message-passing systems, improve them where appropriate, and standardize them. Hence, these advantages allow programmers to write truly portable parallel algorithms (i.e., message-passing algorithms can be executed on any computer platform that implements this message-passing model). MPI is widely available and free of charge with both source code and executable format. The MPI library is currently implemented on Intel iPSC/860, Intel Paragon, IBM SP-2, Silicon Graphics, Cray T-3D etc., and network clusters

implemented on systems such as Linux, SunOS, Solaris, IBM AIX, and PC-Windows/NT. These characteristics provide the IMTA with the power of portability across many platforms. The remainder of this section presents more details of the parallel 2-D cross-correlation module implementation. Both, the vector initialization phase and the guided auto-correlation phase, are combined into one routine. The phases are parameterized and controlled by the 'phase_option' parameter. The bold statements in the routine will be described in more details.

```
MAIN PROGRAM    Parallel 2-D area correlation module
BEGIN
    Initialize MPI environment.
    Get Input parameters.
    Create Image Rotation Look-Up Table.
    Read Input Disparity file (contains the vectors of the grid cell) into memory.
    Get Reference Image header information (i.e., # of pixels, # of lines)
    Get Match Image header information (i.e., # of pixels, # of lines)
    IF (phase_option equal to Phase 2: initialization phase) THEN
        Run Vector_Initialize_Phase Routine.
    ELSE IF (phase_option equal to Phase 3 or 4: guided auto-correlation phase) THEN
        Run Guided_Auto_Correlation_Phase Routine.
    ELSE
        print, 'Error -- Invalid processing option !!!'
    Output Updated Disparity File.
    Terminate MPI environment.
END
```

5.2 Create Image Rotation Look-Up Table

This routine generates a rotation Look-Up Table (LUT) for 11.25 degree increments in counter clockwise direction for the offset of the match image. The purpose of this LUT is to speed up the rotation of the match image during the matching process. Instead of rotating the image every time during the process, this routine calculates the offset of the window's index so that the rotated image pixel can be reached without actually rotating the image. For example, the top-left quadrant of a 0 rotation for a 5×5 LUT will look like:

-2, -2	-1, -2	0, -2
-2, -1	-1, -1	0, -1
-2, 0	-1, 0	0, 0

The LUT is generated for both the x and y coordinates. When the matrix is rotated by n degrees about (xc, yc) , the x and y LUTs are used to point to the location of a pixel $(xc+i, yc+j)$ in a matrix centered at (xc, yc) . The results of the above matrix after 45 degree rotation is:

0, -3	1, -2	1, -1	
-1, -2	0, -1	1, -1	45.0 degrees rotation LUT.
-1, -1	-1, -1	0, 0	

Input Parameters: Maximum Rotation Number (i.e., $32=360/11.25$)

Correlation Window Size.

Output Parameters: Delta X-direction LUT.

Delta Y-direction LUT.

Returns: ERROR - Error in processing.

OK - Success.

BEGIN

increment = $360.0 / \text{max_rotation_number}$

rotation = 0.0

half_window_size = correlation_window_size / 2

FOR r=0 TO max_rotation_number DO

line_index = 0

FOR j=-half_window_size TO half_window_size DO

pixel_index = 0

FOR i=-half_window_size TO half_window_size DO

$\text{DxLUT}[r][\text{line_index}][\text{pixel_index}] =$
 $i * \cos(\text{rotation}) + j * \sin(\text{rotation})$

$\text{DyLUT}[r][\text{line_index}][\text{pixel_index}] =$
 $-i * \sin(\text{rotation}) + j * \cos(\text{rotation})$

Pixel_index = pixel_index + 1

END-FOR

line_index = line_index + 1

END-FOR

return OK

END

5.3 Read Input Disparity File

This routine is used to read the entire disparity file into memory. In addition, the vector map is set to value '1' if a vector exists, otherwise it is set to '0'. The vector map is used by the static partitioning routine to correctly identify the vector location and hence assign the appropriate weight function to the grid cell. The format of the disparity file is as follows:

Record 0 (header information record):

- Record Number.
- Total Number of Records in File.
- Number of Grid Cell in Y-direction, Number of Grid Cell in X-direction.
- Grid Cell Size in the Current Working Level.
- Sub-sampling Rate.
- Number of Reduction.
- X- and Y-direction Offsets between Reference and Match Images

Record 1 to n (grid cell records):

- Record Number.
- Grid Cell Flag (-999=NO CORNER(candidate), -500=CANDIDATE, 1=FOUND match area)
- Corner Point in the Cell (y-location), Corner Point in the Cell (x-location)
- Displacement Vector (y-direction), Displacement Vector (x-direction)
- Correlation Value of the Two Areas
- Link-list Pointer (NOT applicable in this version)
- Rotation Number.

Input Parameters: Input Disparity File Name.

Output Parameters: Disparity Records Information.

Number of Vectors Encountered.

Vector Map.

Returns: ERROR - Error in processing.
OK - Success.

```
BEGIN
    if open file::Disparity file equal ERROR
        Return ERROR

    Counter = 0

    WHILE read(record number) NOT equal to End-of-File DO
        IF (record number EQUAL 0) THEN
            Read (Header information record)
        ELSE
            Read Grid cell information into memory.

            IF grid_cell_flag EQUAL 'Vector' THEN
                Counter = Counter + 1
                Set Vector Map[location] = 1
            ELSE
                Set Vector Map[location] = 0
            END-IF
        END-IF
    END-WHILE

    return OK
END
```

5.4 Vector Initialization Phase Routine

The section shows the high-level pseudo-code of the vector initialization phase (Phase 2). The routine searches the entire match image (or the maximum ice drift) for the match area of each candidate on the reference image. For each candidate in the reference image, a matching location is found by cross-correlating a correlation window with all locations in the search area (area-correlation). The highest correlation value will be selected. An assignment is made if the value is greater than the input threshold.

Input Parameters: Reference and Match Image File Name.
Match Image Candidate Map.
Correlation Threshold
Correlation Window Size
Delta X-direction LUT.
Delta Y-direction LUT.
Disparity Vector Information
Maximum Ice Drift
Rotation Flag
Rotation Number

Output Parameters: Updated Disparity Vector Information

Returns: ERROR - Error in processing.
OK - Success.

BEGIN

Uniform Partition: Reference Image

Uniform Partition: Match Image + (Maximum Ice Drift)

Uniform Partition: Match image candidate map + (Maximum Ice Drift)

According to the local process ID,

read the own portion of the reference and the match image.

read the own portion of the match candidate point.

FOR x_index=start_partition_x TO end_partition_x DO

FOR y_index=start_partition_y TO end_partition_y DO

IF (disparity[x_index,y_index].mflag == CANDIDATE) THEN

Return_rotation_number = 0

Return_correlation_value = -999

Calculate Maximum Displacement window on match image.

FOR x_win=start_window_x TO end_window_x DO

FOR y_win=start_window_y TO end_window_y DO

IF rotation_flag = NO_ROTATION THEN

Correlation two area windows with the
given rotation number.

Keep the highest return correlation
value

Set the Return_rotation_number =
rotation_number

ELSE

Correlation two area windows with
ALL rotation number by using
the Delta X/Y-direction LUT.

Keep the highest return correlation
value and the
rotation_number.

END-IF

END-FOR {y_win}

END-FOR {x_win}

IF the Return_correlation_value >= threshold THEN

Update disparity information.

END-IF

END-IF

END-FOR {y_index}

END-FOR {x_index}

Return OK

END

5.5 Guided Auto-Correlation Routine

This routine is used for the guided auto-correlation phases (Phase 3 and Phase 4) of the IMTA. The match vectors found in the initialization phase are used to guide the search for neighbouring unassigned grid cell nodes. The displacement vectors of its neighbours (8-neighbours) are used to predict the new locations. The definition of search areas is also limited to the distance of the maximum ice drift.

Input Parameters:

- Reference and Match Image File Name.
- Match image Candidate Map.
- Correlation Threshold
- Correlation Window Size
- Delta X-direction LUT.
- Delta Y-direction LUT.
- Disparity Vector Information
- Maximum Ice Drift

Output Parameters: Updated Disparity Vector Information

Returns:

- ERROR - Error in processing.
- OK - Success.

BEGIN

Static Partition: Reference Image

Static Partition: Match Image + Maximum Ice Drift

Static Partition: Match Image candidate map + Maximum Ice Drift

FOR all the Grid Cells on the edge DO

Determine and Save the neighbour's process ID.

According to the local processor ID,

read the own portion of the reference and the match image.

read the own portion of the match candidate point.

REPEAT

Iteration = 0

REPEAT

Update_vector_flag = 0

Iteration = Iteration + 1

FOR x_index=start_partition_x TO end_partition_x DO

FOR y_index=start_partition_y TO end_partition_y DO

IF (disparity[x_index,y_index].mflag == CANDIDATE) OR

(Any of its eight neighbours.mflag == VECTOR) THEN

Return_rotation_number = 0

Return_correlation_value = -999

Define the displacement windows according to its' neighbours
vectors.

FOR x_win=start_window_x TO end_window_x DO

FOR y_win=start_window_y TO end_window_y DO

IF rotation_flag = NO_ROTATION THEN

Correlation two area windows with the
rotation Number.

Record highest

Return_correlation_value

Return_rotation_number =

rotation_number

```

ELSE
    Correlation two area windows with
    ALL rotation number by using the
    Delta X/Y-direction LUT.
    Keep the highest return correlation
    value and rotation_number.
END-IF
END-FOR {y_win}
END-FOR {x_win}

IF the Return_correlation_value >= threshold THEN
    Update disparity information with mflag <-- Iteration.
    Update_vector_flag = 1
END-IF
END-FOR {y_index}
END-FOR {x_index}

UNTIL Update_vector_flag = 0

SEND the edge grid cell (Disparity) information to its neighbours.

RECEIVE the edge grid cell (Disparity) information from its neighbours.

IF the local grid cell correlation value < its neighbour's value THEN
    Set the Update_vector_flag = 1
    Replace the local Grid cell disparity information with the new data.
END-IF

SEND the Update_vector_flag to HOST node (node 0).
IF processor ID is HOST THEN
    RECEIVE from all nodes the Update_vector_flag.
    IF ALL Update_vector_flag equal '0' THEN
        BROADCAST Update_vector_flag = 0
    ELSE
        BROADCAST Update_vector_flag = 1
    ENDIF
    RECEIVE from HOST the New Update_vector_flag

UNTIL Update_vector_flag = 0

END

```

5.6 Output Updated Disparity File

This routine is used to output updated vector results to the disparity file. The format of the file is the same as the input disparity file. If the vector initialization phase is running, all processors write all the local grid cell information to the output file. If the guided auto-correlation phase is running, in order to prevent the output the overlapping portion of the grid cell, all processors only write the output start from the 2nd line and 2nd row of the local partition (except those processors that are located along the top and the left edges of the image).

Input Parameters: Output Disparity File Name.
 Output Vector Information for Local Partition.

Output Parameters: Output Disparity File.

Returns: ERROR - Error in processing.
 OK - Success.

```
BEGIN
    if open file::Disparity file equal ERROR
        Return ERROR

    IF (my partition is located along the LEFT edge of the original image) THEN
        Start_cell_X = 0
    ELSE
        Start_cell_X = 1

    IF (my partition is located along the TOP edge of the original image) THEN
        Start_cell_Y = 0
    ELSE
        Start_cell_Y = 1

    IF (process_id == HOST) THEN
        Write (Header information record)

    FOR x_index=Start_cell_X TO end_grid_cell_Y DO
        FOR y_index=Start_cell_Y TO end_grid_cell_Y DO
            Write (Disparity information at location:[x_index,y_index])

        Close file.

END
```

5.7 Implementation Issues

During the implementation of the algorithm, several obstacles were encountered. The following section discusses these issues:

- (1) In the parallel 2-D guided auto-correlation phase, an interesting load-imbalance of the local processing workload occurred after the first iteration, a detailed description is provided in Section 6.3.
- (2) The current working version of the sequential Ice Motion Tracking Algorithm was implemented in FORTRAN. The conversion into C, which is being conducted by Noetix Research Inc., is still in progress. Thus, source code could not be reused for either the vector initialization or guided auto-correlation phases. Both modules needed to be entirely re-written in C for this implementation.
- (3) The accuracy of a floating point number is different between C and FORTRAN compilers. When comparing the results of the parallel C version to the FORTRAN version, most of the floating point values differed by 0.000001. This discrepancy resulted in a significant increase in effort and time required for the verification process.

-
- (4) The byte-order of the binary data, both integer and floating point values, are different between the sequential and the parallel versions of the algorithm due to the different target systems. All the binary data transfers from the sequential IMTA must be converted into ASCII format in order to maintain the compatibility between the two systems.

 - (5) A parallel version of the connected-component labeling algorithm based on the CGM model which introduced by Atallah *et. al.* [3] was also implemented during the course of the thesis. The work experience gained on both the MPI library and the algorithm design technique from the implementation can be directly applied to the IMTA problem.

6 Experimentation Results

6.1 Overview

This section discusses the timing results of the parallel 2-D area correlation implementation outlined in Section 5. The Coarse-Grained Multicomputer used to obtain these results is a 16-node Pentium-based multi-processor machine. Each node has 32 MBytes of local memory and is connected through a fast Ethernet links hub. The Local Area Multicomputer (LAM) implementation of the Message Passing Interface (MPI) was used as the inter-processors communication protocol. Three RADARSAT image pairs were available to test this module. The sizes of the three pairs are:

<u>RADARSAT Image Date</u>	<u>Original Image (pixels)</u>	<u>Working Level (pixels)</u>
February 7 - 8, 1996	5000 x 5981	632 x 746
February 24 - 25, 1996	4000 x 5970	504 x 745
March 6 am - pm, 1996	4096 x 4096	512 x 511

All three image pairs were run on the machine with 1-node (sequential processing time), 2-node, 4-node, 8-node, and 16-node active. The output was verified by comparing it with the sequential implementation of the algorithm to confirm the correctness of the algorithm. Section 6.2 discusses the timing results of the initialization phase of the IMTA and Section 6.3 discusses the timing results of the guided auto-correlation phase.

6.2 Timing Results for the Vector Initialization Phase

Because of the unique design of the algorithm, the algorithm requires no inter-processor communications. Table 6.1 shows the results of the runtimes and Figure 6.1 shows the plot of the average total processing time versus the number of the processing nodes. The numbers in Table 6.1 are the maximum processing times among the nodes. The results show that the total processing time of the vector initialization module decreases almost linearly as the number of processors increases.

	1 node	2 nodes	4 nodes	8 nodes	16 nodes
Feb. 7 - 8, 1996	322.0291	182.7645	92.47736	47.19382	35.40811
Feb. 24 -25, 1996	217.0568	121.559	93.10613	55.69931	27.18627
Mar. 6 am-pm, 1996	216.2861	109.6465	56.48483	30.82703	18.09649
Average	251.790667	137.99	80.68944	44.573387	26.89695667

Table 6.1 : Total Processing Time (in seconds) of the Vector Initialization Phase of the 2-D Correlation Module

(a)

(b)

Figure 6.1 : Processing Time Results of the Vector Initialization Phase:
 (a) Average total processing time versus number of processors in vector initialization phase. (b) The speedup for the vector initialization phase.

6.3 Timing Results for the Guided Auto-Correlation Phase

In the guided auto-correlation phase, inter-processor communications are required for the information exchange along the edges of the partitions. Because of the communications requirement, a linear speedup for these phases is not as easy to obtain as in the vector initialization phase. The main achievement of our approach is that for this phase we do obtain a very good speedup. Table 6.2 shows the average total processing time of these phases and Figure 6.2 shows the plot of the data.

	Phase	1 node	2 nodes	4 nodes	8 nodes	16 nodes
Feb. 7 - 8, 1996	3	12218.11589	7790.04944	4766.55289	3192.33095	2077.141676
	4	14130.14927	9118.58151	6833.90598	4449.103365	3360.808812
Feb. 24 -25, 1996	3	1271.522688	895.859015	566.911193	364.726398	258.271989
	4	9262.038701	7784.00131	5131.53448	4004.723401	2793.491803
Mar. 6am-pm, 1996	3	3162.945083	2349.47409	1058.61816	672.257161	327.542655
	4	9411.489435	5714.41384	3875.95824	2975.437901	1664.129091
Average		8242.71018	5608.729868	3705.58016	2609.7632	1746.897671

Table 6.2 : Average Total Processing Time (in seconds) of the Guided Auto-Correlation Phase of the Parallel 2-D Correlation Module

Figure 6.2 : Processing Time Results of the Guided Auto-Correlation Phase:
 (a) Average total processing time versus number of processors in guided auto-correlation phase. (b) The speedup of the guided auto-correlation phase.

Table 6.3 and Figure 6.3 show the total communication time of the guided auto-correlation phases. With respect to the communication time, each node is required to send the grid cell

information along the edges to its immediate neighbour nodes. For comparing a 2-node process with a 4-node process, assuming that the image size is constant, we observe that the perimeter (number of grid cells along the edges) of the 2-node partition is generally larger than the 4-node partition. This can be used to explain the communication time decreases when the number of the processors increases, as shown in Table 6.3. On the other hand, one can argue that the number of messages increases as the number of processor increases. However, in most of the current multicomputer architectures multiple messages can be sent to other processors in parallel without much delay as long as they are within the total bandwidth of the router. It appears that this was the case for our experiments.

	Phase	1 node	2 nodes	4 nodes	8 nodes	16 nodes
Feb. 7 - 8, 1996	3	N/A	5491.356002	4098.58959	3023.7224	2028.20018
	4	N/A	5355.513905	5030.01859	3674.80566	3309.58686
Feb. 24 -25, 1996	3	N/A	158.024855	351.100224	279.631477	224.973703
	4	N/A	5023.798316	4056.641277	3867.66833	2762.54727
Mar. 6 am - pm, 1996	3	N/A	1157.029358	620.042881	521.055408	290.530905
	4	N/A	2514.403396	3349.973012	2895.0932	1630.19641
Average		N/A	3283.3543053	2917.727596	2376.996079	1707.672555

Table 6.3 : Average Total Communication Time (in seconds) of the Guided Auto-Correlation Phase of the Parallel 2-D Correlation Module

In summary, our experiments have shown a very good speedup for up to 16 processors but we are concerned about the algorithm's behaviour when considerably more processors are involved. In the remainder of this section, we study possibilities for improvement are examined.

Figure 6.3 : Average Total Communication Time versus Number of Processors in Guided Auto-Correlation Phase

To study the results of the load-balance partitioning scheme, the local processing time is further elaborated to show the workload of each node in each iteration. Table 6.4 shows a sample of the local processing time for a 2-node and 4-node process with 2 iterations.

Iteration	2 nodes		4 nodes			
	[0]	[1]	[0]	[1]	[2]	[3]
1	1265.706	1118.821	549.5301	344.9405	547.5635	575.1147
2	1063.559	73.43145	482.6554	93.01452	72.34383	75.33376

Table 6.4 : Sample local processing time of a 2-node and 4-node results

The table illustrates that in the first iteration, the workload for each processor is balanced (within 40% difference). However, in the second iteration, the workloads are not at all balanced (over 1000% difference). For example, node [0] of the 2-node process used roughly 1063.559 seconds, and node [1] only used 73.43145 seconds (more than 14 times faster than the node [0]). This result can be attributed to 2 sources: (1) the non-uniform vectors

distributed from one iteration to another (a static partitioning scheme was used in this implementation); and (2) the errors in the estimate of the workload. It would be preferable to maintain the static partitioning scheme because the dynamic partitioning scheme would introduce more inter-processors communication. Therefore, further investigation of the workload function of the vector distribution on the grid cells appears to be a good path towards further performance improvements.

7. Conclusion

Earlier, the processing requirements of the Canadian Ice Service's (CIS) Sea-Ice Motion Tracking Algorithm were discussed. In order to meet these requirement, the efficiency of the automated algorithms currently implemented must be improved. To achieve this, a parallel version of the automated ice motion algorithm based on the Coarse-Grained Multicomputer model was designed. The most time consuming portion, the two-dimensional area correlation matching, was implemented using the uniform partitioning scheme and static partitioning scheme to load-balance the workload of the processors. The overlap mapping technique was used to distribute the image data among the processors in order to reduce the communication rounds. The Coarse-Grained Multicomputer used to test the implementation was a 16-node Pentium-based multi-processor machine. The LAM implementation of the Message Passing Interface was used as the inter-processor communication protocols.

The experimental results, as shown in Section 6, were encouraging. They indicate that the total processing time of the parallel correlation matching module decreases almost linearly as the number of processors increases in the vector initialization phase. The speedup of the guided auto-correlation phase was not as good as the initialization phase. However, it still

maintained a very good speedup for up to 16-node processors. The results show that the technique used to implement the matching module successfully improved the algorithm. This speedup in the processing time could greatly improve the turnaround time of the EMTA in the operational environment at the Canadian Ice Service making it possible to process RADARSAT images within the time frame required.

However, the load-imbalance after the first iteration in the guided auto-correlation phase, indicates that further study of the load-balancing scheme and cost estimating function are recommended for obtaining further performance enhancements.

References

- [1] T.A. Agnew, H. Le and T. Hirose, "Estimation of Large Scale Sea Ice Motion From SSM/I 85.5 GHz Imagery," to appear in *Annals of Glaciology*, Volume 25, 1996.
- [2] S.G. Akl and K.A. Lyons, *Parallel Computational Geometry*, Prentice Hall, 1993.
- [3] M.J. Atallah, F. Dehne and S.E. Hambrusch, "A Coarse-Grained, Architecture-Independent Approach for Connected Component Labeling," in preparation, 1996.
- [4] M.J. Collins and W.J. Emery, "A Computational Method for Estimating Sea Ice Motion in Sequential Seasat Synthetic Aperture Radar Imagery by Matched Filtering," *Journal of Geophysical Research*, Volume 93, No. C8, pp. 9241-9251, August 15, 1998.
- [5] F. Dehne, A. Fabri and A. Rau-Chaplin, "Scalable Parallel Geometric Algorithms for Coarse Grained Multicomputers," *Proc. ACM Symposium on Computational Geometry*, pp. 298-307, 1993.
- [6] F. Dehne, C. Kenyon and A. Fabri, "Scalable and Architecture Independent Parallel Geometric Algorithms With High Probability Optimal Time," *IEEE Symposium on Parallel and Distributed Processing*, pp. 586-593, 1994.

- [7] F. Dehne, X. Deng, P. Dymond, A. Fabri and A. Kokhar, "A Randomized Parallel 3D Convex Hull Algorithm for Coarse Grained Multicomputers," *Proc. ACM Symposium on Parallel Algorithms and Architectures*, 1995.
- [8] M. Filey and D.A. Rothrock, "Extracting Sea Ice Data From Satellite Imagery," *IEEE Trans. Geosci. Remote Sensing*, GE-24, pp. 849-854, 1986.
- [9] M. Filey and D.A. Rothrock, "Sea Ice Tracking by Nested Correlation," *IEEE Trans. Geosci. Remote Sensing*, GE-25, pp. 570-580, 1987.
- [10] Geomatics Canada, *Remote Sensing in Canada*, Canada Centre for Remote Sensing, Geomatics Canada, Volume 23, Number 3, December 1995.
- [11] A.V. Gerbessiotis and L.G. Valiant, "Direct Bulk-Synchronous Parallel Algorithms," *Proc. of Third Scandinavian Workshop on Algorithm Theory, Algorithm Theory - SWAT '92*, Helsinki, Finland, pp. 1-18, July 1992.
- [12] W. Gropp, E. Lusk and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MIT Press, 1994.
- [13] R. Hartley and A. Rosenfeld, "Hierarchical Line Linking for Corner Detection," *Integrated Technology for Parallel Image Processing*, Academic Press, pp. 101-119, 1985.
- [14] T. Heacock, T. Hirose, F. Lee, M. Manore, and B. Ramsay, "Sea-Ice Tracking on The East Coast of Canada Using NOAA AVHRR Imagery," *Annals of Glaciology*, Volume 17, pp. 405-413, 1993.

- [15] T. Hirose and L. McNutt, "Toward an Automated Ice Tracking Algorithm," *Report prepared for the Canada Centre for Remote Sensing, Ice Applications Group*. Ottawa, Noetix Research, 1987.
- [16] T. Hirose, T. Heacock, and F. Lee, "Advances in Sea Ice Tracking Algorithms," *Proc. of the 14th Canadian Symposium on Remote Sensing*, 6-10 May, Calgary, Alberta Canada, pp. 292-295, 1991.
- [17] Z. Hussain, *Digital Image Processing: Practical Applications of Parallel Processing Techniques*, Ellis Horwood, 1991.
- [18] F.W. Leberl, *Radargrammetric Image Processing*. The Artech House Remote Sensing Library, 1990.
- [19] F. Lee, "A Parallel Implementation of The Sea Ice Tracking Algorithm for Coarse-Grained Multicomputers," to appear in *The Carleton Journal Of Computer Science*, No. 1, 1997.
- [20] Noetix Research, "Evaluation of an Ice Tracking Algorithm," *Report prepared for the Canada Centre for Remote Sensing, Ice Applications Group and the Canadian Ice Service, Environment Canada*. Ottawa, Noetix Research 1991.
- [21] M. Onoe, K. Preston, Jr and A. Rosenfeld, *Real-Time/Parallel Computing: Image Analysis*, Plenum Press, 1981.
- [22] T. Pavlidis, *Algorithms for Graphics and Image Processing*, CSP, 1982.
- [23] I. Pitas, *Parallel Algorithms: for Digital Image Processing, Computer Vision and Neural Networks*, John Wiley and Sons, 1993.

-
- [24] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer Verlag, 1985.
 - [25] RADARSAT International, *Sea Ice Monitoring*, RADARSAT International, October 1993.
 - [26] B. Ramsay, T. Hirose, T. Heacock, and A. Walton, "Automatic Synthetic Aperture Radar Information Extraction Algorithm for Sea Ice Applications," *Proc. IEEE International Geoscience and Remote Sensing Symposium, IGRASS '94*, Pasadena, pp. 505-507, August 1994.
 - [27] S. Ranka and S. Sahni, *Hypercube Algorithms: with Applications to Image Processing and Pattern Recognition*, Springer-Verlag, New York, 1990.
 - [28] R.E. Suorsa and B. Sridhar, "A Parallel Implementation of a Multisensor Feature-Based Range-Estimation Method," http://windchime.arc.nasa.gov/~grendel/papers/A_93131_RWOLRD/vpr/cpr.html, 1995.
 - [29] O. Vainio, L. Yin and Y. Neuvo, "Parallelism in Generalized Median Operations," *From Pixels to Features II*, Elsevier Science Publishers, pp. 359-371, 1991.
 - [30] L.G. Valiant, "A Bridging Model for Parallel Computation," *Communications of the ACM*, Volume 33, No. 3, pp.103-111, August 1990.
 - [31] L.G. Valiant, "General Purpose Parallel Architecture," *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed., Elsevier Science Publishers, Amsterdam 1990.

-
- [32] L.G. Valiant, "Why BSP Computer?," *IEEE Symposium on Seventh International Parallel Processing*, Newport Beach, California, pp.2-5, April 1993.