UILU-ENG-89-2235 CSG-112

COORDINATED SCIENCE LABORATORY College of Engineering

PARALLEL IMPLEMENTATION AND EVALUATION OF MOTION ESTIMATION SYSTEM ALGORITHMS ON A DISTRIBUTED MEMORY MULTIPROCESSOR USING KNOWLEDGE BASED MAPPINGS

Alok N. Choudhary Mun K. Leung Thomas S. Huang Janak H. Patel

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Approved for Public Release. Distribution Unlimited.

	REPORT DOCU	MENTATION I	PAGE		
REPORT SECURITY CLASSIFICATION		16. RESTRICTIVE	MARKINGS		
Unclassified	None				
SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION	AVAILABILITY OF	REPORT	President and the	
	Approved	for public r	elease;		
DECLASSIFICATION / DOWNGRADING SCHED	distribut	ion unlimite	d		
PERFORMING ORGANIZATION REPORT NUM	BER(S)	5. MONITORING	ORGANIZATION REP	PORT NUMBER(S)
				1. 1. 1. 1. 1.	Salar (Salar)
UILU-ENG-89-2235 (CSG-112)					
A. NAME OF PERFORMING ORGANIZATION	7a. NAME OF MO	DNITORING ORGAN	IZATION		
Coordinated Science Lab	NT/A	NASA / NSE			
University of Illinois	N/A		State and ZID C	and al	
ADDRESS (City, State, and ZIP Code)		NASA Langle	v Research Co	enter	
1101 W. Springfield Avenue		Hampton, VA	23665		
Urbana, IL 61801		National Sc 1800 G Stre	ience Founda	tion	20550
a. NAME OF FUNDING / SPONSORING	8b. OFFICE SYMBOL	9. PROCUREMENT	INSTRUMENT IDE	NTIFICATION NU	MBER
ORGANIZATION	(if applicable)	NASA NAG 1-	613		
NASA / NSF		NJF TKI 07-	03400		Street and
c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF F	UNDING NUMBERS	TACK	INORK LINIT
		ELEMENT NO.	NO.	NO.	ACCESSION N
see 7b		Charles Contractor			14 1
I. TITLE (Include Security Classification)					
Parallel Implementation and	Evaluation of Mo	tion Estimati	on System Al	gorithms or	na
Distributed Memory Multipro	cessor using Know	ledge Based M	lappings		
2. PERSONAL AUTHOR(S)					
Alok N. Choudhary, Mun K. L	eung, Thomas S. H	uang and Jana	k H. Patel	11.11.11.11.11.11.11.11.11.11.11.11.11.	
3a. TYPE OF REPORT	COVERED	14. DATE OF REPO	RT (Year, Month, D	ay) 15. PAGE	COUNT
3a. TYPE OF REPORT 13b. TIME Technical FROM_	TO	14. DATE OF REPO 89/10	RT (Year, Month, D	(ay) 15. PAGE	COUNT
3a. TYPE OF REPORT 13b. TIME Technical FROM_ 6. SUPPLEMENTARY NOTATION	COVERED TO	14. DATE OF REPO 89/10	RT (Year, Month, D	(15. PAGE	
3a. TYPE OF REPORT 13b. TIME Technical FROM_ 6. SUPPLEMENTARY NOTATION	COVERED TO	14. DATE OF REPO 89/10	RT (Year, Month, D	9 ay) 15. PAGE 25	COUNT 5
3a. TYPE OF REPORT 13b. TIME Technical FROM_ 6. SUPPLEMENTARY NOTATION 7. COSATI CODES	TO TO	14. DATE OF REPO 89/10	RT (Year, Month, D	identify by bloc	COUNT
3a. TYPE OF REPORT 13b. TIME Technical FROM 6. SUPPLEMENTARY NOTATION 7. COSATI CODES FIELD GROUP SUB-GROUP	COVERED TO TO 18. SUBJECT TERMS	14. DATE OF REPO 89/10	RT (Year, Month, D	identify by bloc	COUNT
3a. TYPE OF REPORT 13b. TIME Technical FROM_ 6. SUPPLEMENTARY NOTATION 7. COSATI CODES FIELD GROUP SUB-GROUP	TO 18. SUBJECT TERMS computer v	14. DATE OF REPO 89/10	RT (Year, Month, D e if necessary and estimation,	identify by bloc motion est	COUNT 5 ck number) imation
3a. TYPE OF REPORT 13b. TIME Technical FROM_ 6. SUPPLEMENTARY NOTATION 7. COSATI CODES FIELD GROUP	COVERED TO 18. SUBJECT TERMS computer v system algo	14. DATE OF REPC 89/10 (Continue on revers	e if necessary and estimation,	identify by bloc motion est	COUNT 5 :k number) imation
3a. TYPE OF REPORT 13b. TIME Technical FROM	TO TO 18. SUBJECT TERMS computer v: system algo any and identify by block blov a sequence of image	14. DATE OF REPO 89/10 (Continue on revers ision, motion orithms number) understanding vis	e if necessary and estimation, ion algorithms in y	identify by bloc motion est	COUNT 5 :k number) imation ut of
3a. TYPE OF REPORT 13b. TIME Technical FROM	COVERED TO TO TO 18. SUBJECT TERMS computer v: system algo computer v: system algo system algo any and identify by block block bloy a sequence of image sequence of image xt algorithm in the sequence sequence	14. DATE OF REPC 89/10 (Continue on revers ision, motion prithms number) understanding visionce. Vision syste	e if necessary and estimation, ion algorithms in w ms consist of algorithms	identify by bloc motion est which the output prithms that exh	COUNT 5 (k number) imation ut of hibit
3a. TYPE OF REPORT 13b. TIME Technical FROM	TO	14. DATE OF REPC 89/10 (Continue on reverse ision, motion orithms number) understanding vise ence. Vision syste decomposition an	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba	identify by bloc motion est which the output prithms that exh alancing technic	COUNT 5 tk number) imation the of hibit ques
3a. TYPE OF REPORT 13b. TIME Technical FROM	18. SUBJECT TERMS computer v: system algo and identify by block bloy a sequence of image xt algorithm in the seque re, require different data ver, since the input data	14. DATE OF REPO 89/10 (Continue on reverse ision, motion orithms number) understanding vise ence. Vision system decomposition and of a task is produce	e if necessary and estimation, ion algorithms in ms consist of algo d efficient load ba eed as the output d	identify by block motion est which the output prithms that exh alancing technic lata of the previo	COUNT tk number) imation ut of hibit ques ious
3a. TYPE OF REPORT 13b. TIME Technical FROM	18. SUBJECT TERMS computer v: system algo any and identify by block bloy a sequence of image xt algorithm in the sequence, require different data ver, since the input data ited to perform knowled	14. DATE OF REPC 89/10 (Continue on reverse ision, motion orithms number) understanding vision ence. Vision syste decomposition an of a task is produc ge based data decomposition	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba eed as the output d composition and load techniques for vis	identify by bloc motion est which the output orithms that exh alancing technic lata of the previ ad balancing.	COUNT 5 imation imation ut of hibit ques ious This becce
3a. TYPE OF REPORT 13b. TIME Technical FROM	18. SUBJECT TERMS computer v: system algo any and identify by block bloy a sequence of image xt algorithm in the sequence, require different data ver, since the input data ited to perform knowled perform static and dyna bloy at they canture the computer the computer	14. DATE OF REPC 89/10 (Continue on reverse ision, motion orithms number) understanding vise ence. Vision syste decomposition an of a task is produc ge based data deco mic load balancing putational requirem	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba ced as the output d omposition and loa techniques for vis	identify by bloc motion est which the output orithms that exh alancing technic lata of the previation systems. The examining the	COUNT ck number) imation ut of hibit ques ious This hese data
 3a. TYPE OF REPORT Technical 13b. TIME FROM_ 6. SUPPLEMENTARY NOTATION 7. COSATI CODES FIELD GROUP SUB-GROUP 9. ABSTRACT (Continue on reverse if necess. Computer vision systems emp an algorithm is the input of the nervarying characteristics, and therefor for parallel implementation. Howe task, this information can be explor paper presents several techniques to techniques are novel in the sense th when it is produced. Furthermore. 	18. SUBJECT TERMS computer v: system algo and identify by block ploy a sequence of image xt algorithm in the sequence, require different data ver, since the input data ited to perform knowled perform static and dyna they capture the comp they can be applied to m	14. DATE OF REPC 89/10 (Continue on reverse ision, motion orithms number) sunderstanding vise ence. Vision syste decomposition an of a task is produce ge based data decomic load balancing putational requirem any vision systems	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba ced as the output d omposition and loa techniques for vis nents of a task by because many algo	identify by block motion est which the output inters that exh alancing technic lata of the previous ad balancing. The sion systems. The examining the o orithms in diffe	COUNT ck number) imation int of hibit ques ious This hese data rent
 3a. TYPE OF REPORT Technical 13b. TIME FROM	18. SUBJECT TERMS computer v: system algo any and identify by block ohy a sequence of image xt algorithm in the sequence, require different data ver, since the input data ited to perform knowled operform static and dyna hey can be applied to minilar computational char	14. DATE OF REPO 89/10 (Continue on reverse Ision, motion orithms number) understanding vise ence. Vision system decomposition and of a task is produce ge based data decomic load balancing putational requirem any vision systems acteristics. These	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba eed as the output d omposition and loa techniques for vis hents of a task by because many algo techniques are eva	identify by bloc motion est which the output orithms that exh alancing technic lata of the previ- ad balancing. The examining the o orithms in diffe aluated by apply	COUNT ck number) imation it of hibit ques ious This hese data rent ying
 3a. TYPE OF REPORT Technical 13b. TIME FROM_ 6. SUPPLEMENTARY NOTATION 7. COSATI CODES FIELD GROUP SUB-GROUP 9. ABSTRACT (Continue on reverse if necessa Computer vision systems emp an algorithm is the input of the nervarying characteristics, and therefor for parallel implementation. Howe task, this information can be explopaper presents several techniques to techniques are novel in the sense th when it is produced. Furthermore, systems are either same, or have sin them on a parallel implementation 	TO TO 18. SUBJECT TERMS computer v: system algo ary and identify by block bloy a sequence of image xt algorithm in the sequence, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the com they can be applied to m milar computational char of the algorithms in a m	14. DATE OF REPC 89/10 (Continue on reverse ision, motion orithms number) understanding vise ence. Vision system decomposition and of a task is produced ge based data decomic load balancing putational requirem any vision systems acteristics. These otion estimation system	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba ced as the output d omposition and loa techniques for vis nents of a task by because many algo techniques are eva ystem on a hyperc	identify by bloc motion est which the output orithms that exh alancing technic lata of the previation sion systems. The examining the orithms in differ aluated by apply ube multiproce	COUNT ck number) imation imation thof hibit ques ious This hese data rent ying ssor
3a. TYPE OF REPORT 13b. TIME Technical FROM	18. SUBJECT TERMS computer v: system algo and identify by block ploy a sequence of image xt algorithm in the sequence, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the complicational char of the algorithms in a matem of the algorithms in a matem	14. DATE OF REPC 89/10 (Continue on reverse Ision, motion orithms number) sunderstanding vise ence. Vision system decomposition and of a task is produce ge based data decomic load balancing putational requirem any vision systems acteristics. These otion estimation systems acteristics. These otion estimation systems acteristics. These	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba ed as the output d omposition and loa techniques for vis nents of a task by because many algo techniques are eva ystem on a hyperc action of features,	identify by bloc identify by bloc motion est which the output orithms that exh alancing technic lata of the previa ad balancing. The examining the orithms in diffe aluated by apply ube multiproce 2) stereo match	COUNT ck number) imation imation t of hibit ques ious This hese data rent ying ssor h of f
3a. TYPE OF REPORT 13b. TIME Technical FROM	18. SUBJECT TERMS computer v: system algo and identify by block oloy a sequence of image xt algorithm in the seque re, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the compliant they can be applied to millar computational char of the algorithms in a millar consists of the follo match of images from the complexity of motion perform	14. DATE OF REPO 89/10 (Continue on reverse tsion, motion orithms number) understanding vise ence. Vision system decomposition and of a task is produce ge based data decomic ic load balancing outational requirem any vision systems acteristics. These otion estimation systems otion estimation estimation systems otion estimation estimation sy	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba eed as the output d omposition and loa techniques for vis nents of a task by because many algo techniques are eva ystem on a hyperc action of features, nts, 4) stereo matures	identify by bloc identify by bloc motion est which the output orithms that exh alancing technic lata of the previa ad balancing. The amining the orithms in differ aluated by apply ube multiproce 2) stereo matc ch to compute for e gains when the	COUNT ck number) imation it of hibit ques ious This hese data rent ying ssor h of final hese
 3a. TYPE OF REPORT Technical 6. SUPPLEMENTARY NOTATION 7. COSATI CODES FIELD GROUP SUB-GROUP 19. ABSTRACT (Continue on reverse if necessa Computer vision systems emp an algorithm is the input of the net varying characteristics, and therefor for parallel implementation. Howe task, this information can be explopaper presents several techniques to techniques are novel in the sense th when it is produced. Furthermore, systems are either same, or have sin them on a parallel implementation system. The motion estimation system. The motion estimation system and load balance. 	18. SUBJECT TERMS computer v: system algo ary and identify by block bloy a sequence of image xt algorithm in the sequence, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the compliant they can be applied to minilar computational char of the algorithms in a mitem consists of the follo match of images from tation of motion parame cing techniques are use	14. DATE OF REPO 89/10 (Continue on reverse 1. sion, motion orithms number) understanding vise ence. Vision system decomposition an of a task is produce ge based data decomic putational requirem any vision systems acteristics. These otion estimation systems acteristics. These otion estimation systems inferent time instates the significant acteristics and	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba eed as the output d omposition and loa techniques for vision techniques for vision techniques are evaluated ystem on a hypercon action of features, ints, 4) stereo matures and the performance and the overhead of	identify by bloc motion est which the output orithms that exh alancing technic lata of the previation ad balancing. Sion systems. The examining the of orithms in differ aluated by apply ube multiproce 2) stereo match of using these to	COUNT ck number) imation imation t of hibit ques ious This hese data rent ying ssor h of final hese ech-
3a. TYPE OF REPORT 13b. TIME Technical FROM	18. SUBJECT TERMS computer v: system algo any and identify by block bloy a sequence of image xt algorithm in the sequence, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the commister of the algorithms in a mean consists of the follo match of images from tation of motion parameter of mean of motion parameter of mean	14. DATE OF REPC 89/10 (Continue on reverse asion, motion orithms number) sunderstanding vise tence. Vision system decomposition and of a task is produced ge based data decomic load balancing putational requirem any vision systems acteristics. These otion estimation systems acteristics. It is shown the d are significant and	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba ced as the output d omposition and loa techniques for vis hents of a task by because many algo techniques are eva ystem on a hyperc action of features, nts, 4) stereo mature at the performance nd the overhead of	identify by block motion est which the output orithms that exh alancing technic lata of the previa ad balancing. The examining the orithms in diffe aluated by apply use multiproce 2) stereo match ch to compute the of using these to	COUNT ck number) imation imation th of hibit ques ious This hese data rent ying ssor h of final hese ech-
3a. TYPE OF REPORT 13b. TIME Technical FROM	18. SUBJECT TERMS computer v: system algo and identify by block oloy a sequence of image xt algorithm in the seque re, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the compliant they can be applied to m milar computational char of the algorithms in a m tem consists of the follo match of images from t tation of motion parame cing techniques are use	14. DATE OF REPC 89/10 (Continue on reversed ision, motion orithms number) understanding vise ence. Vision system decomposition and of a task is produce ge based data decomic load balancing putational requirem any vision systems acteristics. These otion estimation systems acteristics. These otion estimation systems inferent time instate ters. It is shown the d are significant and 21. ABSTRACT S	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba ced as the output d omposition and loa techniques for vis nents of a task by because many algo techniques are eva ystem on a hyperc action of features, mat the performanc nd the overhead of ECURITY CLASSIFIC	identify by block motion est which the output orithms that exh alancing technic lata of the previa ad balancing. The sion systems. The examining the orithms in differ aluated by apply ube multiproce 2) stereo match the gains when the of using these to ATION	COUNT ck number) imation imation th of hibit ques ious This hese data rent ying ssor h of final hese ech-
 3a. TYPE OF REPORT 13b. TIME Technical 6. SUPPLEMENTARY NOTATION 7. COSATI CODES FIELD GROUP SUB-GROUP 9. ABSTRACT (Continue on reverse if necess. Computer vision systems emp an algorithm is the input of the ner varying characteristics, and therefor for parallel implementation. Howe task, this information can be explo paper presents several techniques to techniques are novel in the sense th when it is produced. Furthermore, systems are either same, or have sin them on a parallel implementation system. The motion estimation system. The motion estimation system images in one time instant, 3) time unambiguous points and, 5) comput data decomposition and load balar niques is minimal. 20. DISTRIBUTION / AVAILABILITY OF ABSTRAES 	18. SUBJECT TERMS computer v: system algo and identify by block bloy a sequence of image xt algorithm in the seque re, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the comp they can be applied to m milar computational char of the algorithms in a m tem consists of the follo match of images from tation of motion parame cing techniques are use CT AS RPT. DTIC USEF	14. DATE OF REPC 89/10 (Continue on reverse 15:101, motion orithms number) understanding vise ence. Vision system of a task is produce ge based data decomic ic load balancing outational requirem any vision systems acteristics. These otion estimation systems acteristics. These acteristics. Th	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba eed as the output d omposition and loa techniques for vis nents of a task by because many algo techniques are eva ystem on a hyperc action of features, nts, 4) stereo matures the overhead of ECURITY CLASSIFIC ified	identify by bloc motion est which the output orithms that exh alancing technic lata of the previa ad balancing. The amining the orithms in diffe aluated by apply ube multiproce 2) stereo matc ch to compute the e gains when the of using these the ATION	COUNT ck number) imation at of hibit ques ious This hese data rent ying ssor h of final hese ech-
3a. TYPE OF REPORT 13b. TIME Technical FROM	TO 18. SUBJECT TERMS computer v: system algo ary and identify by block bloy a sequence of image xt algorithm in the sequence of image re, require different data ited to perform knowled perform static and dyna at they capture the compliant they can be applied to mention milar computational char of the algorithms in a mentem consists of the follo match of images from tation of motion parame cing techniques are use CT AS RPT. DTIC USEF	14. DATE OF REPC 89/10 (Continue on reverse ision, motion orithms number) understanding vise ence. Vision system of a task is produce ge based data decomic based data decomic of a task is produce ge based data decomic of a task is produce ge based data decomic putational requirem any vision systems acteristics. These otion estimation systems acteristics. These otion estimation systems acteristics. It is shown the d are significant and 21. ABSTRACT S Unclass 22b. TELEPHONE	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba eed as the output d omposition and loa techniques for vision techniques for vision techniques are evaluated ystem on a hypercon action of features, ints, 4) stereo matures at the performance and the overhead of ECURITY CLASSIFIC ified (include Area Code)	identify by bloc motion est which the output orithms that exh alancing technic lata of the previa ad balancing. The examining the orithms in diffe aluated by apply ube multiproce 2) stereo match of using these to a balancing the original store of the previous of the original and balancing. The examining the original and balancing the original store of the original and balancing the original and the original and the original and the original and the original and the original and the original and the original	COUNT ck number) imation imation th of hibit ques ious This hese data rent ying ssor h of final hese ech- SYMBOL
3a. TYPE OF REPORT 13b. TIME Technical FROM	TO 18. SUBJECT TERMS computer v: system algo any and identify by block oloy a sequence of image xt algorithm in the sequence, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the commister of the algorithms in a milar computational char of the algorithms in a milar consists of the follo match of images from tation of motion parame cing techniques are use CT AS RPT. DTIC USEF	14. DATE OF REPC 89/10 (Continue on reverse ision, motion orithms number) understanding vise ence. Vision system decomposition and of a task is produce ge based data decomic load balancing putational requirem any vision systems acteristics. These otion estimation systems acteristics. It is shown the d are significant and 21. ABSTRACT S Unclass 22b. TELEPHONE	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba- ced as the output do omposition and load techniques for vision techniques for vision techniques are evan ystem on a hypercondition of features, ants, 4) stereo matures, ant the performance and the overhead of ECURITY CLASSIFIC iffied (Include Area Code)	identify by bloc motion est which the output orithms that exh alancing technic lata of the previa ad balancing. The examining the orithms in diffe aluated by apply ube multiproce 2) stereo matc ch to compute fi e gains when the of using these to ATION	COUNT ck number) imation imation th of hibit ques ious This hese data rent ying ssor h of final hese ech-
3a. TYPE OF REPORT 13b. TIME Technical FROM	TO TO 18. SUBJECT TERMS computer v: system algo and identify by block oloy a sequence of image xt algorithm in the seque re, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the compliant they can be applied to mentational char of the algorithms in a mentation of motion parame cing techniques are use CT AS RPT. DTIC USEF 3 APR edition may be used	14. DATE OF REPC 89/10 (Continue on reversed ision, motion orithms number) understanding vise ence. Vision system decomposition and of a task is produce ge based data decomic ision systems acteristics. These otion estimation systems acteristics. These acteristics. These otion estimation systems acteristics. These acteristics. These a	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba ed as the output d omposition and loa techniques for vis nents of a task by because many algo techniques are eva ystem on a hyperc action of features, ints, 4) stereo mate at the performanc nd the overhead of ECURITY CLASSIFIC ified (include Area Code)	identify by bloc motion est which the output ad balancing. The sion systems. The examining the orithms in diffe aluated by apply ube multiproce 2) stereo match ch to compute the e gains when the of using these to ATION 22c. OFFICE S CLASSIFICATION	COUNT CO
3a. TYPE OF REPORT Technical 13b. TIME FROM_ 6. SUPPLEMENTARY NOTATION 7. COSATI CODES FIELD GROUP SUB-GROUP 9. ABSTRACT (Continue on reverse if necess. Computer vision systems emplition and load balan algorithm is the input of the nervirging characteristics, and therefor for parallel implementation. Howe task, this information can be exploid paper presents several techniques to techniques are novel in the sense the when it is produced. Furthermore, systems are either same, or have sit them on a parallel implementation system. The motion estimation system anabiguous points and, 5) computed at decomposition and load balan inques is minimal. 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT (CARSIFIED/UNLIMITED SAME_OF RESPONSIBLE INDIVIDUAL BUNCLASSIFIED/UNLIMITED SAME 22a. NAME OF RESPONSIBLE INDIVIDUAL	18. SUBJECT TERMS computer v: system algo and identify by block bloy a sequence of image xt algorithm in the seque re, require different data ver, since the input data ited to perform knowled perform static and dyna at they capture the compliant they can be applied to menta of the algorithms in a menta of the algorithms in a menta ited to fimages from the complexition of motion parame cong techniques are used CT AS RPT. DTIC USEF 3 APR edition may be used All other editions are	14. DATE OF REPC 89/10 (Continue on reversed 15 ion, motion orithms number) 9 understanding vise ence. Vision system of a task is produce ge based data decomic ic load balancing putational requirem any vision systems acteristics. These otion estimation sy wing steps: 1) extr different time instaters. It is shown the d are significant and 21. ABSTRACT S Unclass 22b. TELEPHONE until exhausted. obsolete.	e if necessary and estimation, ion algorithms in y ms consist of algo d efficient load ba eed as the output d omposition and loa techniques for vis nents of a task by because many algo techniques are eva ystem on a hyperc action of features, nts, 4) stereo matu hat the performance d the overhead of ECURITY CLASSIFIC ified (Include Area Code) SECURITY UNICLAS	identify by bloc motion est which the output rithms that exh alancing technic lata of the previa ad balancing. The sion systems. The examining the of orithms in diffe aluated by apply ube multiproce 2) stereo matc ch to compute the e gains when the of using these the ATION 22c. OFFICE S CLASSIFICATION SIFIED	COUNT COUNT COUNT Count Co

Parallel Implementation and Evaluation of Motion Estimation System Algorithms on a Distributed Memory Multiprocessor using Knowledge Based Mappings

Alok N. Choudhary, Mun K. Leung, Thomas S. Huang and Janak H. Patel

Coordinated Science Laboratory University of Illinois 1101 W. Springfield Urbana, IL 61801

Address for Correspondence

Corresponding Author : Alok N. Choudhary Department of Electrical and Computer Engineering Science and Technology Center Syracuse University Syracuse, NY 13244

Mun K. Leung, Thomas S. Huang and Janak H. Patel Coordinated Science Laboratory University of Illinois 1101 W. Springfield Urbana, IL 61801

Conference : Computer Architectures Vision and Pattern Recognition

Neither this paper nor any version close to it has been or is being offered for publication elsewhere. If accepted, the paper will be personally presented at the designated 10ICPR by the author or one of the co-authors.

Alok N. Choudhary

Parallel Implementation and Evaluation of Motion Estimation System Algorithms on a Distributed Memory Multiprocessor using Knowledge Based Mappings

Alok N. Choudhary, Mun K. Leung, Thomas S. Huang and Janak H. Patel

Coordinated Science Laboratory University of Illinois 1101 W. Springfield Urbana, IL 61801

Abstract

Computer vision systems employ a sequence of image understanding vision algorithms in which the output of an algorithm is the input of the next algorithm in the sequence. Vision systems consist of algorithms that exhibit varying characteristics, and therefore, require different data decomposition and efficient load balancing techniques for parallel implementation. However, since the input data of a task is produced as the output data of the previous task, this information can be exploited to perform knowledge based data decomposition and load balancing. This paper presents several techniques to perform static and dynamic load balancing techniques for vision systems. These techniques are novel in the sense that they capture the computational requirements of a task by examining the data when it is produced. Furthermore, they can be applied to many vision systems because many algorithms in different systems are either same, or have similar computational characteristics. These techniques are evaluated by applying them on a parallel implementation of the algorithms in a motion estimation system on a hypercube multiprocessor system. The motion estimation system consists of the following steps: 1) extraction of features, 2) stereo match of images in one time instant, 3) time match of images from different time instants, 4) stereo match to compute final unambiguous points and, 5) computation of motion parameters. It is shown that the performance gains when these data decomposition and load balancing techniques are used are significant and the overhead of using these techniques is minimal.

This research was supported in part by National Aeronautics and Space Administration Under Contract NASA NAG-1-613, and in part by National Science Foundation Grant IRI 87-05400.

1. Introduction

Computer vision tasks employ a broad range of algorithms. In vision system many algorithms with different characteristics and computational requirements are used in a sequence where output of one algorithm becomes the input of the next algorithm in the sequence [1,2]. An example of such a system is a motion estimation systems. In such a system, a sequence of images of a scene are used to compute the motion parameters of a moving object in the scene. Figure 1 shows the computational flow for a motion estimation system in which stereo images (L_{im} and R_{im}) at each time frame are used as the input to the system. Briefly, the involved tasks (or algorithms) in this system are as follows. The first algorithm is computation of zero crossings of the images (edge detection (L_{zc} and R_{zc})). The zero crossings are used as feature points for both stereo and time matching. The stereo match algorithm provides points to compute 3-D information about the object in the scene. Using these matched points (*Lsm* and *Rsm*), the corresponding points in the image in the next time frame (L_{tm}) are located and this task is performed by time match algorithm. Again, stereo match is used to obtain the corresponding 3-D points in the next image frame. These two sets of points provide information to compute the motion parameters. The above process is repeated for each new set of input image frame.

This paper presents techniques to perform efficient data decomposition and load balancing for vision systems for medium to large grain parallelism. Two important characteristics of these techniques are that they are general enough to apply to many vision systems, and that they use statistics and knowledge from execution of a task to





perform data decomposition and load balancing for the next task. For example, in the motion estimation system sufficient knowledge can be obtained about the output data from the zero crossing step to perform data decomposition and load balancing for the stereo matching step. The advantages of such schemes are as follows. First, these techniques use characteristics of tasks and data, and therefore, work well no matter how data changes. Second, many vision systems consist of such tasks and exhibit the above described computation flow, and therefore, these techniques can be used in any system (e.g., object recognition, optical flow etc.) [2].

The performance of the proposed techniques is evaluated by using a parallel implementation of the motion estimation system algorithms on a hypercube multiprocessor system. The results show that using uniform partitioning, without considering the computations involved, parallel processing does not provide significant performance improvements over sequential processing. Furthermore, by applying the proposed data decomposition and load balancing techniques significant performance gains (as much as 6 fold) can be obtained over uniform partitioning.

This paper is organized as follows. In Section 2 we provide a brief description of each step in the motion estimation system. For a detailed description, the reader is referred to [3, 4]. These algorithms will provide insight into the involved computations in the motion estimation system. Section 3 describes the proposed load balancing and data decomposition techniques. In section 4 we present a parallel implementation of these algorithms in an integrated environment on a hypercube multiprocessor, and discuss the performance results for each of these algorithms and data decomposition and load balancing schemes. Some of these techniques have been applied to other integrated vision systems and have been shown to work well [2, 5]. Finally, concluding remarks are presented.

2. Steps in the Motion Estimation System

The motion estimation system consists of the following steps: 1) extraction of features, 2) stereo match of images in one time instant, 3) time match of images from different time instants, 4) stereo match to compute final unambiguous points and, 5) computation of motion parameters. We will not discuss, the last process, calculation of motion parameters, but a discussion on how to compute them can be found in [6]. The matching algorithms use stereo image pairs, and the algorithms are designed to find point correspondences between two consecutive time instants, i.e., t_{i-1} and t_i . From the point correspondences, we can estimate the motion parameters. Typical stereo image pairs at two consecutive time instants (t_7 and t_8) used in this paper are shown in Figure 2, which are outdoor scenes of truck at different locations. The images are segment out from larger images of size 1024×1024 . The imaging setup used in taking the images is parallel axis method. The feature points used in the matching process are

edge points which are considered as the more reliable features obtained from an image. In order to save considerable computation time, the matching process in done by employing non-iterative procedures with the assumption of limited displacement (or disparity) between frames. We apply the matching algorithm on two stereo image pairs at two consecutive time instants t_7 and t_8 . The following is a brief description for each major step of the motion estimation system.

2.1. Feature Points

The feature points used in this algorithm are zero crossing points of an image. We use the method suggested by Huertas and Medioni in [7] to extract the zero crossings of an image. In order to eliminate non-significant zero crossing points and maintain enough details, we threshold the zero crossing image based on the intensity gradient at each zero crossing point. Figure 3 depicts one of the thresholded zero crossing images, l_7 . We associate each zero crossing point with one the sixteen possible zero crossing patterns as suggested and used by Kim and Aggarwal [8]. The patterns are not used directly; instead, we assign each pattern a value according to its local connectivity. These pattern values are used in the matching process.

2.2. Matching

Once zero crossings are extracted in all the involved images, the matching process is applied to find point correspondences among the images (two stereo image pairs at two consecutive time instants). The evidences used in this process to obtain matched point pairs are the normalized correlation coefficient, and the zero crossing pattern values. Furthermore, in order to limit the search space, the assumption of limited displacement or disparity between frames is exploited. The matching process consists of six steps as follows:

- 1) Perform stereo (from left to right) matching in the t_{i-1} stereo image pair.
- Obtain unambiguous matched point pairs by eliminating multiple matches.
- 3) Perform time matching between the unambiguous matched points in the left t_{i-1} image and the feature points of the left t_i image.
- Obtain unambiguous matched point pairs from the time matched points by eliminating multiple time matches.

- 5) Perform stereo matching between the unambiguous matched points (obtained in step (4)) in the left t_i image and the feature points of the right t_i image.
- 6) Obtain unambiguous matched point pairs from the results of t_i stereo matching by eliminating multiple matches.

The results of the above steps are two sets of unambiguous stereo matched point pairs at time instant t_{i-1} and t_i . These two sets are related through steps (3) and (4), the matching over time; therefore, we can pick out all the unambiguous matched points that correspond to each other among the two stereo image pairs at time instants t_{i-1} and t_i . The matching algorithm was applied to the images shown in Figure 2. The final results are depicted in Figure 4, which shows that we have enough point correspondences for the motion estimation.

3. Data Decomposition and Load Balancing Techniques for Parallel Implementation

In a multiprocessor system the simplest method to implement a task in parallel is to decompose the data and equally and uniformly among the processors. In a completely deterministic computation in which the computation is independent of the input data such schemes perform well, and normally, the processing time is comparable on all the processors. That is, efficient utilization and load balancing can be obtained. For example, regular algorithms such as convolutions, filtering or FFT exhibit such properties. The amount of computation to obtain each output point is the same across all input data. Therefore, uniform decomposition of data results in load balanced implementation.

Most other algorithms do not exhibit a regular structure, and the involved computation is normally data dependent. Furthermore, the computation is not uniformly distributed across the input domain. In such cases, a simple decomposition of data does not provide efficient mapping, and results in poor utilization and low speedups. Also, the performance cannot be predicted for a given number of processors, and a given data size, because the computation varies as type of data and its distribution varies. For example, in the stereo match algorithm, the computation is more where feature points are dense, and is comparatively small where number of features is small and sparsely distributed (Figure 3).

In a vision system, it is important to efficiently allocate resources and perform load balancing at each step to obtain any significant performance gains overall. An important characteristic of such systems is that the input data of a task is the output of the previous task. Therefore, while computing the output in the previous task enough





(a) Left and right images at time instant t_7



(b) Left and right images at time instant t_8 Figure 2 : Images set of t_7 and t_8



7







(a) : At time instant t7



(a) : At time instant t_8 Figure 4 : Unambiguous matched points of Figure 2

knowledge about the data can be obtained to perform efficient scheduling and load balancing.

Consider a parallel implementation of a task on n processor parallel machine. Let T_i $(1 \le i \le n)$ denote the computation time at processor node i. Then the overall computation time for the task is given by

$$T_{\max} = \max\{T_1, \dots, T_n\} \tag{1}$$

The total wasted time (or idle time) T_w is given by

$$T_{w} = \sum_{i=1}^{i=n} (T_{\max} - T_{i})$$
(2)

If $T_{\text{max}} = T_i$ for all i, $1 \le i \le n$, then the task will be completely load balanced. Another measure of imbalance is given by the variation ratio V,

$$V = \frac{T_{\max}}{T_{\min}}, \quad T_{\min} = \min\{T_1, ..., T_n\}$$
(3)

The goal in performing load balancing is to minimize T_w , or move V as close to 1 as possible. In the best case, $T_w = 0$ or V = 1.

If T_{seq} is the time to execute the same task on a sequential machine then the speedup is given by

$$S_p = \frac{T_{seq}}{T_{\max}} \tag{4}$$

Therefore, by minimizing T_w , the achievable speedup can be maximized. In the following we discuss such techniques, and in the next section we present the performance results for a parallel implementation of algorithms in the motion estimation system.

3.1.1. Uniform Partitioning

Data decomposition using uniform partitioning performs well as a load balancing strategy for input data independent tasks, because equally dividing the data distributes the computation equally among processors. If total input data size is D then total computation time to execute a task is $T = k \times D$, where k is determined by the computation at each input data point. For example, in convolution of an image with $m \times m$ kernel, $k = 2 \times m^2$ floating point operations. Hence, for an n node multiprocessor, the data decomposition methods to balance the computation is to make the granule size to

$$d_i = \frac{D}{n} \tag{5}$$

For data independent algorithms, such a partitioning guarantees equal distribution of computation among processors. Therefore, if communication time can be minimized, then optimal performance can be obtained on a given multiprocessor.

3.1.2. Static

When computation is not uniformly distributed across the input domain, and is data dependent, uniform partitioning does not work well for load balancing. Normally, computation depends on significant data elements in a partition. Many vision algorithms exhibit this behavior. For example, in stereo match, hough transform etc., the computation is proportional to the number of features (edges) or significant pixels in a granule rather than on the granule size. Therefore, equal size granules do not guarantee load balanced partitioning because of the data dependent nature of the computation. In many such algorithms, the computation time for a granule (i), T_i , is proportional to a certain extent on the granule size (fixed overhead to process a granule), and to the number of significant data in a granule. That is,

$$T_i = A \times d_i + B \times f_i \tag{6}$$

where, d_i is the granule size, f_i is a measure of significant data in granule (i), and A and B are arbitrary constants which depend on the algorithm. The objective is to divide the computation among processors such that each processor receives equal measure of computation. One way to assign a granule to a processor is to compute the total measure of computation and partition is as follows:

$$T_i = \frac{\sum_{i=i}^{i=g} A \times d_i + B \times f_i}{T_i}$$
(7)

where, g is the total number of granules in the input domain (Note that the number of granules for the current task is n for an n processor system).

For example, consider computing hough transform of an edge image to detect line segments. If there exists a line whose normal distance from the origin is r, the normal makes an angle θ with the x-axis then if a point (x,y) lies on that line, the following equation is satisfied.

$$r = x\cos\theta + y\sin\theta$$

r and θ are quantized for desired accuracy and then for each significant pixel (where there is an edge), r is computed for all quantized θ values. If two partitions of equal size contain different number of edge pixels, then the

amount of computation will be different for the two partitions despite them being equal in size. In fact, the computation is directly proportional to the number of edge pixels in a partition. A way to perform static load balancing will be to decompose the input data such that each partition contains equal number of edge pixels. The computation to recognize this partioning can be performed in the task in which edges are detected by keeping a count of the number of edges detected by a processor. Once a task is completed, the data can be reorganized such that the number of edges with each processor is in the interval $(\frac{Z_a}{n} - \delta, \frac{Z_a}{n} + \delta)$, where Z_a is the total number of edges detected in the image, and δ is determined by the minimum granule size from fixed overhead considerations.

3.1.3. Weighted Static

When the computation in a granule not only depends on number of significant data points in the input domain, but it also depends on their spatial relationships, then data distribution needs to be taken into account as a measure of load to perform load balancing. For example, in stereo match or time match, not only does the computation depend on the number of zero crossings, but it also depends on their spatial distribution. If the zero crossings are densely spaced, then the computation will be more than that if the same number of zero crossings need to be matched with each corresponding zero crossing in the other image, whereas less number of zero crossings need to be matched if they are sparsely distributed. Hence, the computation also depends on the spatial density (such as features/row if one dimensional matching is performed). That is,

$$T_i = A \times d_i + B \times w_i \times d_i \tag{8}$$

where, w_i is the feature dependent spatial density. For example, if the minimum granule size is a row of the input data then $w_i = r_i^{\beta}$, where r_i is the number of features in row *i*, and β is a parameter, $0 \le \beta \le 1$. $\beta = 0$ means that the computation is independent of how the features are distributed within a row. Therefore, to divide the computation equally among *n* processors, the following heuristic can be used.

$$T_{i} = \frac{\sum_{i=0}^{i=R} A \times d_{i} + B \times w_{i} \times d_{i}}{n}$$
⁽⁹⁾

where, R is the number of rows in the image. Note that the above heuristics approximate the load and do not exactly divide the computation among processors. However, in the next section we will show that these schemes perform well.

3.1.4. Dynamic

Above three methods use the knowledge about the data when it is produced to perform load balancing for the next task. However, once decomposition is done, then the data is not reshuffled. Therefore, we consider the above methods as knowledged based static load balancing schemes. In the dynamic scheme, the data is decomposed into finer granules such that the number of tasks, (that is number of independent granules) M, is much larger than the number of processors.

At execution time the processors are assigned these tasks dynamically by a designated scheduler from a task queue containing these tasks. Processors are assigned new tasks as they finish their previously assigned tasks, if there are more tasks left to be assigned. However, the knowledge obtained from the previous step can be used again to anticipate the completion of a task, in order to assign a new task to a processor. That is, the task assignment can be pipelined, thereby reducing the overhead of dynamic assignment.

The following procedure illustrates the dynamic assignment of tasks onto the processor. The pseudo code essentially illustrates what the scheduler does in order to perform dynamic load balancing. The number of tasks (max tasks) are determined during the execution of the preceding step in the system, and the task_queue contains all the tasks including the computational information associated with each task. Initially, the scheduler assigns few tasks to each processor. The number of tasks to be assigned initially is a parameter (pipe_line_no). If this parameter is 1, it implies that there is no anticipatory scheduling. In other words, a processor is assinged a new task only when it finishes the task it is currently executing. A task is assigned to a processor only if the task contains significant computation. For example, in stereo match, if a task's data does not contain any zero crossings, then the task can be discarded because it is not going to produce any useful information anyway. In a blind scheme, where little is known about a task, the task will be assigned, which is an overhead, and can be avoided by using the knowledge obtained from the previous steps. Whenever a processor P_i completes the current task, it sends a compl_msg to the scheduler which assigns P_i a new task if the task_queue is not empty. Once the task_queue becomes empty, the scheduler sends a term_msg (terminate message) to all the processors. Upon receiving a term_msg from the scheduler, processors complete the remaining tasks in their task_queues, and sends a term_msg to the scheduler, terminating the computation. Note that by using the pipe_line_no, anticipatory dynamic scheduling can be performed, and a processor need not be idle when a new task is being assigned. By using this parameter, the amount of initial static assignment, and dynamic assignment can be controlled.

```
Dynamic Scheduling of Tasks
         /*Initial Assignment*/
         curr_task = 0;
1.
2.
         for j = 1 to j <= pipe_line_no do
3.
                  for i = 1 to i = num_{proc} do
                           if comp(task_queue(curr_task)) > 0
4.
5.
                                     schedule curr_task at proc. P_i;
                                     curr_task = curr_task+1;
6.
7.
                            else
8.
                                     curr_task = curr_task+1;
                                     go to 4.
9.
10.
                            end if
11.
                  end_for
12.
         end_for
         /*Scheduling*/
13.
         done = false; k = num_proc;
         while not done do
14.
15.
                   wait for msg from a processor;
16.
                  receive msg;
                  if (msg = compl_msg)
17.
                            P_i = sender \text{ processor};
18.
19.
                            if curr_task < max_tasks
                                     if comp(task_queue(curr_task)) > 0
20.
                                              schedule curr_task at proc. P_i;
21.
                                              curr_task = curr_task+1;
22.
23.
24.
                                     else
                                               curr_task = curr_task+1;
25.
                                               go to 19.
26.
                            else
27.
                                     send term msg to P_i.
28.
29.
                   else if (msg = term msg)
30.
                            k = k - 1;
                            if (\mathbf{k} \leq 0)
31.
32.
                                     done = true.
```

4. Parallel Implementation and Performance Evaluation

This section presents a parallel implementation of the algorithms that are part of motion estimation system and describes the performance of the algorithms and load balancing strategies.

4.1. Hypercube Multiprocessor

A hypercube multiprocessor system of size P has P processors, where P is an integral power of 2. P processors are indexed by the integers 0,...,P-1 and the following criteria is satisfied. If the processor numbers are represented by $\log_2(P)$ bits then two processors are connected by communication links if and only if their bit representation

differs by exactly one bit. Therefore, each processor is connected to $\log_2(P)$ processors with direct communication links. Diameter of the hypercube of size P is $\log_2(P)$ (diameter is the maximum distance between any two nodes). We used Intel ipsc/2 hypercube multiprocessor consisting of 16 nodes. Each node consists of an Intel 80386 processor, Intel 80387 co-processor, 4 megabyte memory, and a communication module.

4.2. Feature Extraction

Features used for stereo match algorithms are the zero crossings of the convolution of the image with Laplacian. Zero crossing computation involves 2-D convolution and extraction of zero crossings from the convolved image. Since convolution is a data independent algorithm uniform partitioning is sufficient to evenly distribute the computation. The mapping is a division of $N \times N$ image onto P processors. Each processor computes the zero crossings of share of N^2/P pixels. Data division onto the processors is done along the rows. This mapping reduces communication to only in one direction. The reason is that 2-D convolution can be broken into two 1-D convolution [7]. This not only reduces the computation from W^2 sum of products operations per pixel to $2 \times W$ sum of product operations per pixel (W is the convolution mask window size), but also reduces the communication requirements in a parallel implementation if the data partitioning is done along the rows. There is no need for communication when convolution is performed along the rows.

Table 1 shows the performance results for the above implementation for an image of size 256×256 and convolution window of size 20×20 . First column shows the number of processors in the cube(P). Second column represents the total processing time (t_{proc}) for convolution. Column 3 shows the number of bytes communicated by a processor to the neighboring processor, and column 4 shows the corresponding communication time which is small compared to the computation time. The second half of the table shows the computation time for extracting zero crossings from the convolved image. Corresponding speedups are also shown.

It can be observed that almost linear speedup is obtained for convolution. Two factors which contribute toward this result are that communication overhead is relatively small, and communication is constant as the number of processors increases. However, the speedup obtained in the elapsed time, which includes the program and data load time also, is sub-linear due to the following reason. The hypercube multiprocessor's host does not have a broadcast capability, and therefore, the overhead of loading the program increases linearly with the number of processors. However, data load time increment with the increase in the number of processors is comparatively small because amount of data to be loaded to one processor decreases as the number of processors increases. The only

	Computation for Convolution and Zero Crossings								
	Convolution Window Size = $20x20$								
No. Proc.	Conv. Comp. Time(sec.)	Conv. Comm. Bytes	Conv. Comm. Time(ms.)	Conv. Total Time(sec.)	Conv. Speed Up	ZC Comp. Time(sec.)	ZC Speed Up		
1	109.0	0	0	109.0	1	6.47	1		
2	54.76	2816	13	54.78	1.98	3.23	1.99		
4	27.51	5632	36	27.55	3.95	1.66	3.89		
8	13.88	5632	36	13.92	7.83	0.85	7.60		
16	7.07	5632	36	7.11	15.33	0.42	15.25		

Table	1:1	Performance	for	feature	Extraction	(Zero	Crossings	S)
-------	-----	-------------	-----	---------	------------	-------	-----------	----

Feature Extraction Performance (Elapsed Time)					
No. Proc.	Elapsed Time(sec.)	Speed up			
1	116.2	1			
2	58.8	1.97			
4	30.1	3.86			
8	16.1	7.22			
16	9.6	12.1			

increment in data load time results from the number of communication setups from the host to the node processors, which increases linearly with the number of processors.

4.3. Matching Features

This task involves matching features in stereo pair of images. Since the imaging setup uses the parallel axis method, the epipolar constraint is used to limit the search space for matching to one-dimension which is in the horizontal direction. Thus data partioning along the rows for parallel implementation results in no communication between node processors as long as each partition contains an integral number of rows.

The computation involved in stereo matching algorithm is data dependent. The computation varies across the image because it depends on the number of zero crossings, distribution of zero crossing across the image, and distribution of zero crossings along the epipolar lines. Therefore, partioning the data uniformly among the processors (i.e. assign each processor equal number of rows) may not yield expected speedups and processor utilization. A processor which has very few zero crossings, and sparsely distributed zero crossings will be under utilized, whereas a processor with a large number of zero crossings, and densely distributed zero crossings will become a bottleneck.

We used uniform partitioning, static load balancing, weighted static and dynamic load balancing schemes to decompose the computation on the multiprocessor. Static load balancing can be achieved by keeping a count of the zero crossings with each processor when the previous task (feature extraction) is executed. At the completion of the task, the data is reorganized using this information, and using the techniques described in the previous section.

Figure 5 shows the distribution of the computation times for 8 processor case. The X-axis shows the processor number, and the Y-axis shows the computation time for each scheme. As we can observe, uniform partitioning does not perform well at all because the variation in computation time is tremendous, and therefore, performance gains are minimal. The static load balancing scheme (shown as dashed bars) performs much better than uniform partitioning, but variation in computation times is still significant because the computation also depends on the distribution of zero crossings. The weighted static scheme performs better than static, and further reduces the variation in computation times. Note that these schemes only measure the load approximately, and therefore, will not divide the computation exactly uniformly. Furthermore, minimum granularity is a row boundary in order to avoid communication between processors. Finally, for 8 processor case, dynamic scheme performs the best. Table 2 summarizes the distribution for the 8 processor case. The Table shows the computation time, variation ratio, and improvement ratio for each processor under all four methods. Table 2 summarizes the distribution for the 8 processor case. The table



Figure 5 : Distribution of Computation Times for Stereo Match (P=8)

shows the computation time for each processor for all four methods. For example, the variation ratio is 44.25 for uniform partitioning, is 2.71 for static load balancing, is 1.50 for weighted static, and is 1.09 for dynamic load balancing. Improvement ratio is the ratio of speedup obtained with load balancing to that of uniform partitioning. The computation times shown include all the overhead of load balancing schemes. Figure 6 shows the speedup graph for varying size of multiprocessor from 1 processor to 16. We observe that uniform partitioning does not provide any significant gains in speedup as the number of processors increases. Dynamic scheme performs the best among all the schemes, and the two static scheme perform comparably with the dynamic scheme. We believe that as the number of processors is increased, the two static schemes will move even closer to dynamic scheme, or even perform better than the dynamic scheme, because for a larger multiprocessors, the overhead of dynamic scheme will be greater.

4.4. Time Match

The computation in time match algorithm is similar to that in stereo match except the search space is twodimensional, and the input to the algorithm is stereo match output. Other difference is that the number of significant points in the input data is much smaller than that in stereo match, because a great deal of input points get eliminated in stereo match. Table 3 shows the distribution of the computation times for the 16 processor case. We only present uniform partitioning and static load balancing cases. The most important observation is that uniform partitioning

Com	putation Time	Distribution fo	r Stereo Matc	h (P=8)	
Proc. No.	Uniform Partitioning	Static	Static Weighted	Dynamic	
	Time (ms.)	Time (ms.)	Time (ms.)	Time (ms.)	
0	364	1402	2439	2890	
1	164	3333	2606	2786	
2	878	3066	2219	2980	
3	7258	3327	2277	2967	
4	6827	3371	2798	2818	
5	5207	3269	3328	2913	
6	762	3063	2864	2803	
7	312	1243	3223	3051	
Max.	7258	3371	3328	3051	
Min.	164	1243	2219	2786	
Variation ratio	44.25	2.71	1.50	1.09	
Improvement ratio	1	2.15	2.19	2.38	

Table 2 : Distribution of Computation Times for Stereo Match



Number of Processors



performs worse than that in the case of stereo match, and static load balancing performs better.

The Table shows how the measure of computation (number of zero crossings left from stereo match step) is divided among the processors in the two cases. It is clear that the number of zero crossings are very evenly distributed (within the minimum granule of one row constraint) in the static case, whereas they are lumped with a few processors in the uniform partitioning case. Figure 7 shows the speedup graphs for the two schemes for a range of multiprocessor size. The speedup gains for the load balanced case is very significant over the uniform partitioning case. We computed the overhead of performing knowledge based static load balancing, and the overhead was 3 ms., which is negligible compared to the computation time, and the performance gains are significant.

4.5. Second Stereo Match

This step involves stereo match computation for features from images at time instant t_{i+1} after time point correspondence is established between images at time t_i and t_{i+1} . The matching is similar to that in first stereo match except that it need to be done only at those points at which time correspondence has already been established.

. . .

Computation for Time Match (Proc. = 16)								
Proc. No.	Uniform Partitioning			With Load Balancing				
	Matching (Sec.)	Total (Sec.)	No. Zcs	Matching (Sec.)	Total (Sec.)	No. Zcs		
0	0.14	0.22	3	9.35	10.00	47		
1	0.03	0.14	2	12.38	12.55	50		
2	0.02	0.13	0	13.12	13.21	53		
3	0.02	0.13	0	14.23	14.27	43		
4	0.02	0.13	0	11.88	11.91	45		
5	3.61	3.72	21	10.93	10.95	44		
6	13.45	13.56	55	12.82	12.85	53		
7	5.09	5.20	20	12.16	12.19	51		
8	26.65	26.76	93	11.41	11.44	45		
9	45.85	45.97	182	10.63	10.65	40		
10	73.82	73.93	259	13.89	13.91	50		
11	27.20	27.32	121	13.69	13.71	44		
12	0.31	0.42	3	15.07	15.09	43		
13	0.11	0.22	1	15.70	15.72	56		
14	0.42	0.53	4	14.36	14.39	56		
15	0.08	0.10	0	5.21	5.68	43		
	Max. time(sec.)	Min. time(sec.)	Variation ratio	Speed up	Improvement ratio			
Uniform	73.82	0.10	738	2.69	a series			
Balanced	15.72	5.68	2.76	12.63	4.7	S. Asta		

Table 3 : Distribution of Computation Time for Time Match



Number of Processors

Figure 7 : Speedup for Time Match

Consequently, the number of features to be matched are much less than that in the first computation, and hence, the importance of load balancing is further increased. Figure 8 depicts the distribution of computation times for the second stereo match step. The three load balancing algorithms used in this case are Uniform Partitioning, Static and Dynamic. We observe from the Figure that uniform partitioning does not perform well compared to the other two schemes. The variation in computation time is significant, and the static and dynamic schemes perform comparably.

Figure 9 presents the speedups for the same algorithm for various multiprocessor sizes. The Figure shows that the gains from these load balancing schemes are very significant over uniform partitioning. One important observation can be made by comparing results in Figure 6 and 9. Note that the performance of uniform partitioning in the second stereo match is much worse than that in the first stereo match. For example, for 16 processor case, the speedup in the first case is 5.55, whereas for the same multiprocessor size speedup is only approximately 2.3 for the second stereo match. Therefore, as the computation progresses in an integrated environment, the gains of these load balancing schemes become increasingly significant.

•







Number of Processors

Figure 9 : Speedups for Second Stereo Match

4.6. Summary of Results

In summary, the following important observations can be made from the results presented in this section. First, the improvement in performance (such as utilization and speedup) itself increases using the load balancing schemes as the number of processors increases. Therefore, performance gains are expected to be higher for larger multiprocessors. Second, in an integrated environment, the overheads of such methods are small because measure of load can be computed at run time as a bi-product of the current task. Finally, though we showed the performance results of the implementation on the hypercube multiprocessor, these methods can be applied when algorithms are mapped on any medium to large grain multiprocessor system, because these techniques are independent of the underlying multiprocessor architecture.

Consider the overall performance gains for the entire system. As the computation progresses from one step to the next, uniform partitioning performs worse because the data points reduce, but the computation at each point increases. Hence, the gains of using parallel processing are minimal. However, the load balancing techniques recognize the data distribution at each step, and the data is decomposed using the distribution. Therefore, performance

gains are expected to improve as the computation progresses in an integrated environment. For example, consider zero crossing, stereo match, time match, and second stereo match steps. In zero crossing computation, uniform partitioning performs well and load is balanced. Hence, the improvement ratio is 1. For stereo match the improvement of static over uniform partitioning is 2.15 for 8 processor case, and is 2.22 for 16 processor case. Similarly, for time match step, the improvement of static load balancing for 8 processor case is 3.38, and for 16 processor case, it is 4.2. Therefore, the improvement in performance itself increases as the number of processors increases as well as when the computation progresses in from one step to the next in a vision system.

5. Concluding Remarks

In this paper we presented techniques to perform efficient data decomposition and load balancing for vision systems, for medium to large grain parallelism. Two important characteristics of these techniques are that they are general enough to apply to any such integrated system, and that they use statistics and knowledge from the execution of a task to perform data decomposition and load balancing for the next task in the system. Knowledge from each step is used to perform load balancing in the next step. The advantages of such schemes are as follows. First, these techniques use characteristics of the tasks and the data, and therefore, work well no matter how the data changes. Secondly, many vision systems consist of such tasks and exhibit the above described computation flow, and therefore, these techniques can be used in any system.

Finally, the performance of the proposed techniques was evaluated by using a parallel implementation of the motion estimation system algorithms on a hypercube multiprocessor system. The results show that using uniform partitioning without considering the computations involved, parallel processing does not provide significant performance improvements over sequential processing. Furthermore, by applying the proposed data decomposition and load balancing techniques significant performance gains (as much as 6 fold) can be obtained over uniform partitioning.

REFERENCES

- C. Weems, A. Hanson, E. Riseman, and A. Rosenfeld, "An integrated image understanding benchmark: recognition of a 2 1/2 D mobile," in *International Conference on Computer Vision and Pattern Recognition*, Ann Arbor, MI, June 1988.
- [2] Alok N. Choudhary, "Parallel architectures and parallel algorithms for integrated vision systems," in *Ph.D. Thesis*, University of Illinois, Urbana-Champaign, Agust 1989.
- [3] Mun K. Leung and Thomas S. Huang, "Point matching in a time sequence of stereo image pairs," in Tech. Rep., CSL, University of Illinois, Urbana-Champaign, 1987.
- [4] M. K. Leung, A. N. Choudhary, J. H. Patel, and T. S. Huang, "Point matching in a time sequence of stereo image pairs and its parallel implementation on a multiprocessor," in *IEEE Workshop on Visual Motion*, Irvine, CA, March 1989.
- [5] Alok N. Choudhary, Subhodev Das, Narendra Ahuja, and Janak H. Patel, "Surface reconstruction from stereo images : an implementation on a hypercube multiprocessor," in *The Fourth Conference on Hypercubes, Concurrent Computers, and Applications*, Monterey, CA, March 1989.
- [6] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-square fitting of two 3-D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 698-700, September 1987.
- [7] A. Huertas and G. Medioni, "Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 651-664, September 1986.
- [8] Y. C. Kim and J. K. Aggarwal, "Positioning 3-D objects using stereo images," Computer and Vision Research Center, The University of Texas at Austin.