On Recognition of Group of Human Beings in Images with Navigation Strategies

Using Efficient Matching Algorithms
with Parallelization

Von der Fakultaet Informatik, Elektrotechnik und
Informationstechnik, der Universitaet Stuttgart
zur Erlangung der Wuerde eines Doktors der
Naturwissenschaften (Dr.rer.nat) genehmigte Abhandlung

Vorgelegt von

Douglas Antony Louis PIRIYAKUMAR
aus Paramakudi, Tamil Nadu, Indien

Hauptberichter : Prof.Dr.Paul Levi
Mitberichter : Prof.Dr.C.Siva Ram Murthy
Mitberichter : Dr. Kenji Hanakata

Tag der muendlichen Pruefung :
24.7.2003

Institut fuer Parallelle und Verteilte Systeme der
Universitaet Stuttgart
2003

This thesis is dedicated to my paternal aunt **Panchavathy** *whose discipline and sacrifice improvised my life*

Acknowledgments

My doctoral work was pursued at the Institute of Parallel and Distributed Systems (formerly known as Parallel and High-performance Systems) in the department of Computer Science, Electrical Engineering and Information Technology, University of Stuttgart, Stuttgart in Germany. Without GAES (German Academic Exchange Service - in German DAAD, Deutscher Akademischer Austauschdienst) this study would not be possible. My gratitude will remain ever with DAAD for the fellowship from June 1996 to September 1999.

First of all, I must thank Prof.Paul Levi for his acceptance as the guide for my doctoral study and for his critical comments without mincing with words apart from his financial support from October 1999 till the last date. The amount of academic freedom I enjoyed through him stands as a testimony of german higher education system. His friendly suggestions to acclimatize with german system outside the campus were really helpful in the early days. I thank Prof.C.Siva Ram Murthy, my guide in IIT Madras for his acceptance as joint referee. His valuable comments are useful from the beginning. I also thank Dr.Kenji Hanakata for being as second joint referee. His continuing criticism with philosophical background improved my thoughts at various stages.

For the innumerable assistance and suggestions, I am thankful to the former colleagues especially, Prof. Thomas Braeunl, Dr. Nobert Oswald, Niels Mache, Frank Herrman, Dr. Matthias Muscholl, Michael Becht and Dietmar Lippold, and the present colleagues Viktor Avrutin, Thorsten Buchheim, Guenter Hetzel, Georg Kindermann, Olga Kornienko, Serguei Kornienko, Reinhard Lafrenz, Dr. Michael Schanz, Frank Schreiber, Moritz Schule, Monika Tepfenhart, Georg Wackenhut and Peter Burger. Special thanks to Olga, Serguei and Monika for their help and participation in taking images for the experiments. For the timely and tireless help, I thank our secretary Ms. Ute Graeter.

The useful discussions which I held with Prof. Egbert Lehmann, Dr. Rolf Rabenseifner, Prof. Walter Knoedel and Prof. Ulrich Hertrampf clarified several of the fundamental issues associated with my doctoral study. I thank also Prof. Klaus Lagally and Prof.Kurt Rothermal for their relevant courses. For the friendly advices and assistances, I thank Dr. Kenji Hanakata, Dr. Holger Petersen, Bernd Holzmueller and Hartmut keller. For those who gave flavour to student life in Allmandring 20c, I thank specially Marcel, Juergen, Daniela, Jose Daniel, Demian and Selvin. My regards will remain with families of Dr.Kenji Hanakata, Josef Spanniger, Otto Weiss, Kurt Weizanegar and Stephan Machmer.

No words could adequately portray the joyful days I had with my DAAD friends, Prof.S.V.Joga Rao, Dr. P. Jaisankar, Dr. R. Purvaja, Dr. A.M. Sembian, Kamal Sharma, V. Ashok Kumar, Dr.S. Hazra, Dr. Balaram and their families. For the cherishable reminiscences and their timely supports, I thank my indian friends in stuttgart Fr. Francis Xavier, Fr. Jeyaraj Boniface, Fr. Denis Ponniah, Prof. K.P. Karunakaran, Prof.Y.G.Srinivasa, Shri.S. Raghunath, Mr.Miranda, Dr. Kripesh, Dr.V.S.Srinivasan, N.Manickam, Dr. R. Jayaganthan, N. Sivakumar, R. Manohar, S. Bipin, Dr. Niraimathi, R. Ramakrishnan, Dr. B. Koushik, J. Kartik, P. Prasannakumar, Dr. S. Shashi, J. Surabin, N. Srihari, K.Sameer, Dr. Amin Iqbal, M. Anand, C. Ramesh, Dr.R.Sarathi, R.Magesh, Dr.Tamil Selvan, A.P. Sahayaraj, Dr.N. Sivakumar, Dr. R.Subashri, and many others who still remain in my tranquility with their families along with my classmates S. Pachaiammal, R.S. Ramesh and N. Ramesh.

Consistent and timely encouragements would not be available to me without my former colleagues in Pondicherry University, Prof. S.Kuppuswami, Prof. S. Gunasekaran, Prof. K.S.Mathew, Prof. P.Jothilingam and the faculties in computer science and mathematics departments especially M.S.Ashok, V. Prasanna, M. Sundaramohan, N. Amoudha, Dr. K .M . Tamizhmani, Dr. Tamazharasi, Dr. M.Subbiah and Dr. V.Muruganantham. To my friend and colleague Dr. R.Subramanian to whom I owe which I may not able to repay, my sincere brotherly affection will continue. For standing with me in the tempest time in Pondicherry university, I profoundly thank M. Velayudham. I thank also Dr. Paul, M. Shanmugasundram and K.R. Ramesh.

In my opinion, success starts with the family. I must thank God for such a blessing. Uniquely most of my family members were my teachers. From my father Late.J.Douglas Thangadurai and my mother Mary Alangaram, I learned my english and mathematics respectively which helped me to top in every one of my academic achievements. I thank my paternal uncle J.E.Chelladurai for excellence in english grammar which I follow. For the love and affectionate care, I thank my paternal aunts Late.Jeyaseeli, Late. Mara, Late. Margaret, Kamala Chelladurai and Sebastinal Florence and my uncle R.Anthonisamy and aunt Maria Louis for their timely assistance. With the love and affection of my brothers and sisters, Pascal Jeevaraj, Irene Punitha, John Barnabas and Arul Mariapackiam, I am able to lead my life contented. Finally, without the cooperation and love of my affectionate wife Teresa and my sweetest daughter Ursula, I could have not accomplished the strenuous tasks peacefully during the final stages of this study.

Contents

List of Figures

List of Tables

Notations

G - Graph(V,E)
V - Set of Vertices
v_i - i^{th} Vertex in V
E - Set of Edges
e_{ij} - Edge between v_i and v_j
$< v_i, v_j >$ - Directed Edge from v_i to v_j
c_{ij} - Cost involved in matching v_i to v_j
$f^*(x)$ - Cost of Node x in A^* Algorithm
$g^*(x)$ - Cost of getting Node x from the start node in A^* Algorithm
$h^*(x)$ - The Lower Bound estimation of the cost at Node x in A^* Algorithm
P_i - The regular polygon with i vertices (in images)
v_i^j - j^{th} vertex in P_i
g_i - Neighbour Isomorphic Group
L_i - Line of Symmetry
h_i - Set of vertices divided by L_i
θ_i - Threshold grey value
ϕ - Distance Function
a_{ij} - Binary valued Image
a_{ij}^k - a_{ij} at k^{th} iteration
A,B - Sets of points
H(A,B) - Hausdorff Distance between A and B
h(A,B) - Directed Hausdorff Distance of A from B
||...|| - Distance Norm(DN)
$d_B(a)$ - Minimum Distance value at a point a to the point set B
r - the number of Images
q - the number of Models
p - the number of processors
R(i,j) - Read i^{th} Image and j^{th} Model
c(i,j) - Find the matching positions of Model j in Image i
r(k) - Exchange Model k with the neighbouring processor
T_i - Task i
P_k - Processor k (in Scheduling of Tasks)
μ_{P_k} - Service rate or speed of the processor P_k
λ_i - The Static Level of T_i
τ_i - The set of Parent Tasks of T_i
η_i - The set of Children Tasks of T_i
$\zeta_{T_k P_k}^{N_j}$ - The Completion time of Task T_k on P_k in node N_j
Γ_{N_i} - Set of Tasks assigned in node N_i

Zusammenfassung

Einleitung

Mit leistungsfaehiger Prozessoren und Bildsensoren wurde die Bildverarbeitung zu einem realisierbaren und nuetzlichen Verfahren. Starke Interessen sind mit Bildverarbeitung fuer die hohen Anforderungen des neuen Jahrtausends verbunden. Das zentrale Problem der kuenstlichen Intelligenz (Nachahmung menschlicher Faehigkeiten oder Turing Tests) wird weit untersucht. Das Sehvermoegen ist eine derjenigen menschlichen Faehigkeiten, deren Nachahmung durch Kameras eine echte Herausforderung fuer die Computerwissenschaftler darstellt.

Um das Computersystem intelligenter zu machen richten die Anstrengungen richten sich so dass es die von der Kamera erfassten Bilder "versteht". Neben der Nachahmung der Sehfaehigkeit muessen solche Systeme Bilder verarbeiten koennen, um die Effizienz zu steigern. In Umgebungen wo Menschen und Maschinen zusammenarbeiten, darf die Maschinen den Menschen keinen Schaden zufuegen. In zahlreichen industriellen Umgebungen bewegen sich Industrieroboter bei ihrer Arbeit in der Nachbarschaft von Menschen. Eine bessere Aufbereitung der erfassten Bildinformationen ist notwendig um die schwere Schaeden zu vermeiden bei der Steuerung von Maschinen durch Menschen, wie z.B. durch Fahrer Assistenzsysteme in Automobilen. Das Erkennen von Menschen, um deren Gefaehrdung in solchen Umgebungen zu verhindern, etablierte sich als neuer Fachbereich des computergestuetzten Sehens (Computer Vision) und wird teilweise als "Looking at People" (Ausschau nach Menschen) bezeichnet.

Das Erkennen von Menschen in Bildern von einaeugigen (monocular) Kameras ohne die ueblichen Einschraenkungen ist das Ziel der vorliegenden Arbeit. Das zuverlaessige Hausdorff Matching Verfahren wird erweitert, so dass Menschen anhand von vielfaeltigen Modellen und mittels veraenderter Abstandsmessungen erkannt werden. Da die Strategie der Vereinigung mehrerer Algorithmen, um trotz Okklusionen bessere Ergebnisse zu erzielen, im Grundsatz bereits parallel ist, lieferte die parallele Implementierung des Systems auf einem Cray 3TE richtige Ergebnisse in wesentlich kuerzerer Rechenzeit.

Allgemeines Matching Verfahren

Ob zwei abstrakte Darstellungen (zumindest teilweisen) gleich sind, ist das Grundproblem fuer die Entwicklung von Kuenstliche Intelligenz Systemen mit menschlichen Faehigkeiten (computergestuetzten Sehen). Aufgrund ihrer Leistungsfaehigkeit werden fuer die abstrakte Objekt Darstellung oft Graphen benutzt. Parametrische oder syntaktische Vorgehensweisen in entsprechender Komplexitaet findet man in der Literatur.

Bei der Mustererkennung und beim maschinellen Sehen werden Graphen als Darstellungen der a priori bekannten Objekt-Modelle und der zu erkennenden, unbekannten Objekte benutzt. Bei dieser Darstellungsform der Objekte wird das Problem des Erkennens zu einem Problem der Graphen Uebereinstimmung (Graph Matching). Fuer das Graph Matching werden die beiden wichtigsten Verfahren, die Verzweigungs- und Abgrenzungsmethode (Branch and Bound), sowie nichtlineare Optimierungsverfahren in grossem Umfang eingesetzt, da die Komplexitaet NP ist. Obwohl auch andere Methoden vorwiegen, werden das Matching mit Graphen Isomorphie oder mit Raumzustands Verfahren (State Space), wie etwa die A^* Strategie, fuer rechnerintensive exakte oder optimale Loesungen vorgezogen. Auch fuer das fehlertolerante bzw. das fehlerkorrigierende Graph Matching werden diese Verfahren eingesetzt. Die vorliegende Arbeit schlaegt Verfahren vor, um effizientes Graph Matching mit weniger Rechenzeit zu erreichen.

Definition der Nachbar Isomorphie(Neighbour Isomorphism)

Die Nachbar Isomorphie zwischen zwei Knoten v_1 und v_2 in einem Graphen G_1 ist wie folgt definiert:

v_1 und v_2 sind Nachbar Isomorph, wenn es eine Anzahl k von Nachbarn im Abstand r fuer v_1 in G_1 gibt, so dass es exakt eine Anzahl k von Nachbarn im Abstand r fuer v_2 in G_1 gibt. (Wenn der Graph gewichtet wurde, so sollten die entsprechenden Knoten paarweise Nachbar isomorph sein).

Sei G der Graph mit der Menge V Knoten und der Menge E Kanten. g_i soll die Menge Knoten bezeichnen, die Nachbar isomorph sind. Offensichtlich ist dann $g_i \subset V$. Es sei $V_i, V_j \in V$.

$V_i, V_j \in g_k$ wenn $N(V_i, l, g_p) = r$ und $N(V_j, l, g_p) = r, \forall l = 1$, ..., Durchmesser von G, wobei $N(V_i, l, g_p) = r$ bedeutet, dass es eine Anzahl r von Nachbarn von g_p mit dem Abstand l gibt.

Eigenschaften der Nachbar Isomorphie

1. Es seien g_p und g_q zwei NI Gruppen in G_i. Dann schliessen sie sich gegenseitig aus. D.h. $v_i \in g_p$, dann $v_i \notin g_q$ und umgekehrt.

2. Wenn g_p eine NI Gruppe in G_i ist und g_q eine NI Gruppe in G_j ist und $i \neq j$, dann sind g_p und g_q aehnlich, wenn $v_i \in g_p$ und $v_j \in g_q$, $N(v_i, l, g_p) = r$ und $N(v_j, l, g_q) = r, \forall l = 1$, ..., Durchmesser von G im entsprechenden Graphen.

3. NI ist invariant zu Translationen, Rotationen und Skalierungen (mit einheitlichen Skalierungsfaktoren).

Bharathanatyam Tanzstellungen

Einer der aeltesten klassischen Taenze Indiens ist Bharathanatyam und besteht aus genau festgelegten Bewegungen und Tanzstellungen. Die kombinierte Methode von der Nachbar Isomorphie und den verbesserten A^* Al-

gorithmus mit Bharathanatyam Tanzstellungen als Datenbank findet fuer eine bestimmte Tanzstellung als Eingabe die richtige Uebereinstimmung. Bei Eingabe einer anderen Stellung, die nicht in der Datenbank enthalten ist, findet der Algorithmus den am aehnlichsten aussehenden Stellung(en).

Abstands Transformation - Chamfering

Die Eigenschaften des Bildes sind die wichtigste Elemente und muessen eindeutig von den Nicht-Eigenschaften des Bildes unterschieden werden. Als Eigenschaften koennen Ecken, Kanten, helle Punkte oder Bereiche mit einer besonderen Textur in Frage kommen. Um die Ecken und Kanten im Bild zu erkennen gibt es zahlreiche Algorithmen, wie etwa SUSAN Filter. In unserem Verfahren werden Kanten als Eigenschaft gewertet. Das Ziel dabei ist, jeden Bildpunkt (Pixel) einer Nicht-Kante (Nicht-Eigenschaft) als Abstandsmass zum naechsten Kanten-Pixel (einer Eigenschaft) zu erhalten. Offensichtlich erhalten Kanten-Pixel den Wert Null. Wenn die echte euklidische Abstaenden berechent werden, ist das nicht nur sehr rechenintensiv, sondern braucht auch grossen Speicherplatz. Die Berechnung der Abstaende erfordert daher ein gutes Naeherungsverfahren. Die Abstands Transformation konvertiert ein binaeres Bild als ein Abstandsbild. Das binaeres Bild hat den Grauwert als 0 an den Eigenschaftspunkten und Maximalwert an den anderen Punkten. Danach kann jede der Abstandsfunktionen fuer die Berechnung der Abstaende benutzt werden.

Durch Propagation von lokalen Abstaenden werden die globalen Abstaende im Bild, d.h. den Abstaenden zwischen benachbarten Pixeln des Bildes, angenaehert. Bei diesem Verfahren der Abstands Transformation wird die lokale Operation wiederholt, um die naheliegenden globalen Abstaende zu erhalten. Diese Propagation kann sequentiell oder parallel ablaufen. Eine solche sequentielle Abstands Transformation wird als "Chamfering" bezeichnet. Eine 3 * 3 Nachbarschaft fuer die lokalen Abstaende wird benutzt. Die Chamfering 3-4 Methode ergibt im Vergleich zum euklidischen Abstand einen maximalen Unterschied von 8 Abstaenden ausgehend von durch Rauschen ungenau gewordenen Kanten in realen Bildern ist reine Zeitverschwendung.

Matching mittels Chamfering

Nach Abschluss der Abstands Transformation und nach Erzeugung des Abstandsbildes, kann das Modell verglichen werden. Beim Modell bilden alle Punkte (Pixel), die eine Eigenschaft darstellen (Kanten-Punkte) eine Liste von Koordinatenpaaren, wobei sich jedes Zahlenpaar aus der Spalten- bzw. Zeilennummer des betreffenden Kanten-Pixels zusammensetzt. Das Modell wird an jedem moeglichen Punkt dem Bild ueberlagert. In jedem einzelnen Fall wird dabei das Matching Mass aus der Liste der Koordinatenpaare

(nach einer Translation entsprechend dem aktuellen Ueberlagerungspunkt) berechnet. Handelt es sich um eine vollstaendige Uebereinstimmung (Perfect Match) muss das Matching Mass logischerweise Null betragen. Das Matching Mass in Chamfering ist ein quadratischer Mittelwert (RMS).

Bei weniger komplizierten und auch bei hochkomplexen Faellen funktionierte der Algorithmus gut. Der parallele Algorithmus funktionierte ebenfalls zufriedenstellend, erbrachte jedoch wenig Verbesserungen bezueglich der Rechenzeit. Die Schwankungen bei der Rechenzeit haengen vor allem von der Bildgroesse ab, sowie von der Groesse der Modelle und der Lage im Bild, an der ein Bestandteil des Modells positiv vorliegt. Auch im Falle des parallelen Algorithmus ergibt sich keine groessere Zeitersparnis, da die Prozessoren jedes Mal, wenn sie einen Modell Bestandteil im Bild finden, Daten untereinander austauschen muessen, damit die anderen Prozessoren solche Bereiche vermeiden koennen. Durch Hardware Zwaenge muessen saemtliche Prozessoren fuer einen Datenaustausch zwischen ihnen synchronisiert werden, so dass hierfuer fast mehr Leerlauf-Zeit (idle time) verbraucht wird als fuer die eigentliche Rechenarbeit. In den genannten Faellen wird ein Prozessor angewiesen, nur in dem ihm zugewiesenen Bereich zu rechnen, ohne sich um die anderen Prozessoren zu kuemmern. Die Zuverlaessigkeit geht bei schnellen Berechnungen verloren, daher wurde fuer eine tiefergehende Analyse das Hausdorff Verfahren gewaehlt.

Definition des Hausdorff Verfahrens

Der Hausdorff Abstand ist wie folgt definiert: $A = a_1, a_2, ..., a_n$ und $B = b_1, b_2, ..., b_m$ seien zwei finite Punkte-Mengen. Hausdorff Abstand H(A,B) = max(h(A,B),h(B,A)) wobei $h(A, B) = max_{a \in A} min_{b \in B} DN(a - b)$ Dabei ist DN ein bestimmter Norm-Abstand (nur der Abstand zwischen den beiden Punkten).

Eigenschaften des Hausdorff Verfahrens

1. Der Hausdorff Abstand zwischen zwei Punktmengen H(A,B) ist invariant bezueglich Translation und Rotation wenn fuer beide Punktmengen A und B, die Translation und Rotation mit dem gleichen Mass gemacht werden.

2. Der Hausdorff Abstand zwischen zwei Punktmengen H(A,B) wird fuer den euklidischen Abstand in allen Richtungen mit einem einheitlichen Skalierungsfaktor skaliert. Wenn der Skalierungsfaktor unterschiedlich ist oder wenn eine nichtlineare Abstandsfunktion benutzt wird, muss die Skalierung von H(A,B) nicht stattfinden.

3. Ist im Bild ein Modell vorhanden, wird h(model, image) (NOT H(model,image)) durch das Vorhandensein von Rauschen im Bild nicht beeinflusst.

4. Ist im Bild ein Modell exakt vorhanden, ist h(model, image) (NOT H(model,image)) Null, was dasselbe ist wie der Chamfering Abstand wenn ein Modell mit Eigenschaftspunkten im Bild vorliegt.

5. Das Hausdorff Verfahren kann Verdeckungen (Okklusionen) wirkungsvoll behandeln.

Effiziente Berechnung von Abstaenden

Die Abstandsfunktion ist so gewaehlt, dass Rechenzeit eingespart wird. Zum Beispiel kann die Berechnung der "City Block Distance" weniger Zeit beanspruchen als diejenige des euklidischen Abstands, da hierbei Quadratwurzeln berechnet werden muessen. In beiden Faellen koennen die Abstaende auch nur einmal berechnet und in einer grossen Tabelle gespeichert werden, so dass spaeter nur in der Tabelle nachgeschlagen und nicht mehr neu berechnet werden muss.

Die Abstands Transformationen, wie etwa Chamfering 3-4, wird um asymptotisch naehere Distanzen zu finden benutzt. Da dieses Verfahren in zwei Schritten ablaeuft, ist es sehr wirkungsvoll und benoetigt, im Gegensatz zu den Nachschlage Verfahren (Lookup Method), weder eine Vorausberechnung der Abstaende und eine grosse Tabelle fuer Abstandswerte.

Fuer das Problem gibt es mindestens drei Moeglichkeiten der Parallelisierung. Eine Moeglichkeit ist, jedes Bild in einen Prozessor aufzunehmen und es mit allen Modellen zu vergleichen. Ein anderer Weg waere, alle Bilder einzeln in alle Prozessoren aufzunehmen und die Menge der Modelle gleichmaessig auf die Prozessoren zu verteilen. Die dritte Moeglichkeit besteht darin, jedes Bild auf die Anzahl vorhandener Prozessoren aufzuteilen und diesen Teil des Bildes dann mit allen Modellen zu vergleichen. Im letzten Fall ist es unerlaesslich, fuer eine gewisse Bildueberlappung zu sorgen, um richtige Loesungen zu erhalten. Die zweite Moeglichkeit ist besser um eine Person zu verfolgen. In der vorliegenden Arbeit wurden die ersten beiden Moeglichkeiten implementiert.

Beschreibung des Menschen

Bei jedem Erkennungssystem muss das in einer Umgebung (dem Bild) zu erkennende Objekt (das Modell) entsprechend beschrieben oder definiert sein. Ein Kopf, ein Koerper, zwei Haende und zwei Beine bilden einen Menschen, ohne dabei auf Einzelheiten, wie Nase oder Augen, die fuer die Gesichtserkennung wichtig sind, einzugehen. Je nach Zweck der Anwendung kann die detaillierte Einzelheiten in der Beschreibung stark variieren. Bei einem Fahrer Assistenzsystem zum Beispiel reicht es aus, einen Menschen als Fussgaenger zu erkennen. Bei einem Ueberwachungssystem ist das Erkennen der Anwesenheit von Menschen genug um eine Kette von Handlungen auszuloesen. Eine detailreichere Beschreibung dieses Menschen ist spaeter

erforderlich.

Groesse eines Menschen

Bei den meisten Erkennungssystemen ist die Groesse eines Menschen fest vorgegeben oder muss innerhalb eines bestimmten Bereichs liegen. Wegen dieser Groessen Vorgaben wird ein gehendes Kind nicht erkannt. Auf der Grundlage einer vorgegebenen Groesse des Menschen koennen diese Systeme ausserdem moegliche Regionen fuer den Kopf, den Koerper, eine Hand oder ein Bein erkennen. Dennoch ist die Erkennung eines Menschen durch solche Systeme schwierig und sie wuerde noch schwieriger, wenn wie im vorliegenden Fall, keine Beschraenkungen hinsichtlich der Groesse vorgegeben sind. Die Aufhebung der Groessenbeschraenkung bedingt allerdings, dass das System alle Bildbereiche durchsucht, was mehr Rechenzeit erfordert.

Segmentierung des interessierenden Bereichs (Mensch)

Einige Aenderung im Bild durch Subtraktion des Hindergrunds genugt bei einigen Systemen um die Anwesenheit eines Menschen abzuschliessen. In der Literatur gibt es Verfahren, um durch weitergehende Analyse dieses Bereichs einen Menschen genauer zu erkennen. Interessanterweise spielen die Farbkomponenten bei der Segmentierung des interessierenden Bereichs (Mensch) immer noch eine grosse Rolle. Die Transformation einer solchen Wavelet (Elementarwelle) oder eine Fourier Analyse koennen ebenfalls fuer die Segmentierung des interessierenden Bereichs (Mensch) benutzt werden. Die Subtraktion des Hintergrunds funktionert bei Bildern in offenene Umgebung nicht.

Die Segmentierung eines interessierenden Bereichs (Mensch) auf der Grundlage der Farbe kann nicht zuverlaessig funktionieren, wenn man nicht alle moeglichen Farben von Menschen durchprobiert, da Menschen bekanntlich verschiedene Hautfarben haben koennen. Eine Erkennung laesst sich auch ohne Hintergrund Subtraktion durchfuehren, allerdings zu Lasten der Rechenzeit. In solchen Faellen kann man fuer die Erkennung z.B. das Hausdorff Verfahren anwenden. Es gibt aber noch weitere Probleme wie etwa die Verdeckung (Okklusion) durch andere Objekte, oder die Verdeckung durch das Objekt selbst, wie sie bei bestimmten Sichtwinkeln vorkommen kann.

Verdeckung (Okklusion)

Die Verdeckung von Bildteilen ist eines der groessten Probleme fuer Erkennungssysteme, da dadurch manche kritische oder Haupteigenschaften eines Objekts verdeckt werden koennen und eine korrekte Erkennung unmoeglich machen. In solchen Situationen muss eine Definition der minimal erforderlichen Eigenschaften, die fuer die Erkennung eines Menschen notwendig sind, vorliegen, z.B. das Vorhandensein eines Kopfs. Wenn die Person dabei nicht in die Kamera schaut, ist eine Moeglichkeit, nach einer

annaehernd runden oder elliptischen Form zu suchen. Die Systemleistungen haengen dabei sehr stark vom Sichtwinkel ab. Ebenso ist wohlbekannt, dass Eigenschaften durch das Objekt selbst, je nach Stellung oder Pose, verdeckt werden koennen.

Einige Verfahren, wie z.B. das Hausdorff Verfahren, ermoeglichen es dennoch einige Parameter zu korrigieren und so trotz Verdeckung eine Erkennung durchzufuehren. In solchen Faellen lassen sich Fehl-Erkennungen nicht voellig vermeiden. Interessanterweise laesst sich mit dem Hausdorff Verfahren eine Verdeckung sehr einfach mit vielen Modellen modellieren.

Ausreichende Modelle

Mit Sicherheit lassen sich durch eine groessere Anzahl von Modellen mehr Faelle des Auftretens eines Objekts sicherer erkennen. Die wichtigste Frage ist allerdings wie viele Modelle ausreichend sind, um in jedem Fall das Auftreten eines Objekts zu erkennen. Angesichts der Freiheitsgrade bei der Bewegung von diversen Koerperteilen des Menschen duerfte diese Anzahl Modelle allerdings sehr hoch sein. Viele Systeme, mit denen Bewegungen oder Taetigkeiten von Menschen erkannt werden sollen, sind in hohem Mass auf die ihnen vorgegebenen Modelle beschraenkt. Bei der Erstellung dieser Modelle spielt die Groesse wiederum eine wichtige Rolle. Ein Erwachsener ist keine groessenskalierte Version eines Kindes und umgekehrt. Mit Sicherheit muessen daher unterschiedliche Skalierungsfaktoren fuer die verschiedenen Koerperteile oder verschiedene Modelle fuer jede Groesse benutzt werden.

Im ersteren Fall laesst sich damit zwar die Anzahl notwendiger Modelle reduzieren, aber es muessen verschiedene Skalierungsfaktoren benutzt werden, was wiederum die Rechenzeit enorm erhoeht. Beim zweiten Fall ist es von entscheidender Bedeutung in welcher Reihenfolge die Modelle abgearbeitet werden, denn ein rein sequentieller Vergleich mit jedem Modell wird die notwendige Rechenzeit ebenfalls so stark erhoehen, dass sie fuer bestimmte Anwendungen, wie z.B. Fahrer Assistenzsysteme, inakzeptabel wird. Daher spielen bei diesem Verfahren die Vergleichs Reihenfolge und eine entsprechende Indexierung der Modelle eine wichtige Rolle.

Probleme durch mehrfaches Auftreten

Obwohl die im vorigen Abschnitt definierte Beschreibung eines Menschen akzeptabel ist, kommt es durch das Auftreten von m Beinen, n Haenden, x Koepfen usw... im Bild zu einer neuen Situation. Die genaue Anzahl der in einem Bild oder einer Szene vorkommenden Menschen zu finden, ist vergleichsweise schwierig. Das groesste Problem dabei sind Okklusionen von Eigenschaften. Findet man z.B. eine ungerade Anzahl von Haenden oder Beinen, so ist die paarweise Zuordnung dieser Haende oder Beine zu

einer Person eine reine Frage der Wahrscheinlichkeit ganz zu schweigen von der Moeglichkeit, dass eine behinderte Person tatsaechlich nur ein Bein oder einen Arm haben kann.

Probleme durch Bewegungen

Unabhaengig voneinander auftretende Bewegungen von Personen verursachen eine grosse Anzahl von Okklusionen. Wenn z.B. zwei Personen in Blickrichtung der Kamera hintereinander gehen, besteht eine endliche Wahrscheinlichkeit dafuer, dass eine Person, je nach den jeweiligen Koerpergroessen, die andere vollkommen verdeckt. Auch wenn sich zwei Personen begegnen und anschliessend in verschiedene Richtungen auseinander gehen, ist die Feststellung, welche Person in welche Richtung geht, relativ komplex. Wenn sich, wie etwa bei Militaerparaden, eine Reihe von Personen in derselben Richtung mit derselben Geschwindigkeit bewegen und die Abtastfrequenz des Bildes zufaellig noch mit dem Abstand der Personen uebereinstimmt, kann sich die Erkennung der Bewegung extrem schwierig gestalten.

Grundkonzept der Fusionsarchitektur

Durch die mit dem Chamfering und dem Hausdorff Verfahren gewonnenen Erfahrungen und deren Implementierung in einem System wird klar, dass nur ein einziger Algorithmus nicht ausreicht. Um ausserdem Okklusionen und Groessenunterschiede zu verarbeiten, muessen unterschiedliche Verfahren angewendet werden. Es ist daher notwendig, alle diese notwendigen Methoden in einer Architektur zu kombinieren, die als Fusionsarchitektur bezeichnet wird.

Hausdorff Verfahren bei Okklusionen

Es ist nur natuerlich, dass in realen Bildern gewisse Bildteile verdeckt (okkludiert) sind. Die Anwendung des Hausdorff Verfahrens auf das Modell kann je nach verdecktem Bildteil zu verschiedenen Ergebnissen fuehren. Um Okklusionen zu behandeln, kann man anstelle des bekannten Hausdorff Verfahrens, bei dem die Abstaende zwischen Punkten berechnet werden, ein neues Verfahren anwenden, indem man den Prozentsatz von Punkten berechnet, die von einem vorher festgelegten Grenzwert (Threshold) abweichen. Liegt diese Prozentzahl unter einer bestimmten Grenze (die vorher festgelegt wurde oder den maximal zulaessigen Verdeckungsgrad des Modells darstellt), so kann man davon ausgehen, dass das Modell an dieser Stelle des Bildes vorliegt.

Hausdorff bei unterschiedlichen Groessen

Eines der klassischen Probleme beim Matching von Bildern ist das Problem mit der Groessenunterschied durch unterschiedliches Zoomen. Falls im Folgenden nichts anders gesagt ist, wird fuer die Groessenanpassung grund-

saetzlich in allen Richtungen mit demselben Faktor skaliert. Falls die erforderliche Skalierung so gering ist, dass der Gesamteffekt kleiner ist als der Grenzwert, dann wird sie zuverlaessig arbeiten. Wenn der Gesamteffekt allerdings groesser ist - was den interessanteren Fall darstellt - wird die Aenderung so vorgenommen, dass je nach Skalierung fuer einige angegebene Skalierungspegel ein Abstandshistogramm erstellt wird. Dann wird der Histogramm-Pegel (Abstand) mit der groessten Anzahl Punkte gewaehlt. Die Maximalanzahl Punkte mit dem hoechsten Pegel im Histogramm ueber der Gesamtzahl von Punkten oder Ecken im Modell liefert einen bestimmten Prozentsatz. Wenn dieser Prozentsatz hoeher als ein vorgegebener Wert ist (beispielweise 90

Herausragende Vorteile der Fusionsarchitektur

1. Die Fusionsarchitektur erlaubt es, Okklusionen und Groessenunterschiede zu behandeln.

2. Die Fusionsarchitektur ist grundsaetzlich parallel angelegt.

3. Die Fusionsarchitektur mit Datenkommunikation kann allgemein den Parallel-Rechenaufwand kleiner halten.

4. Auch ohne Datenkommunikation funktioniert die Fusionsarchitektur gut wenn weniger Kommunikation erwartet wird.

5. Die kombinierten Effekte der Fusionsarchitektur ergeben bessere Erkennungsleistungen.

6. Die Fusionsarchitektur ist so allgemein angelegt, dass sie sich auch fuer allgemeine Erkennungsaufgaben einsetzen laesst.

7. Das Konzept der Kombination mehrerer Algorithmen in der Fusionsarchitektur kann auch fuer Suchverfahren in der kuenstlichen Intelligenz und fuer Optimierungsstrategien genutzt werden.

Navigations-Strategien

Da industrielle Anwendungen der Bilderkennung auch fuer die Bewegungssteuerung von Industrierobotern eingesetzt werden, um den Zielpunkt ohne Kollisionen mit Menschen zu erreichen, wurden Algorithmen fuer das Umfahren von Hindernissen entwickelt. In einer bekannten Umgebung liegt das Hauptgewicht auf der Suche nach dem optimalen Weg mit parallelen Strategien. In unbekannten Umgebungen wird nach Verfahren zur Verringerung der Bewegungsaufwands gesucht. Bei der Umfahrung gemaess dem "Problem das Handlungsreisenden" ("Travelling Salesman Problem" TSP) wurde eine parallele Implementierung auf dem Cray T3E mit Datenkommunikation der Prozessoren untereinander eingesetzt, um die Loesung schnell zu finden.

Modelle

Bei Bilderkennungssystemen spielen die Modelle eine herausragende

Rolle: nicht nur fuer die Zuverlaessigkeit der Erkennung, sondern auch bei der Verringerung der Rechenzeit. Von reinen Stab-Modellen eines Menschen zu 3-dimensionalen Koerper-Modellen (Blob model) nimmt die Komplexitaet zu, aber die spaetere Verfolgung wird vergleichsweise schneller. Um Menschen trotz Okklusionen erkennen zu koennen, sind entweder explizite Modelle mit Okklusionen erforderlich oder beim Hausdorff Verfahren die richtige Auswahl gewisser Parameter. Naturgemaess ist es schwierig, Werte fuer Parameter zu finden, die moeglichst viele Faelle abdecken. Die spezielle Erstellung von Modellen mit Okklusionen ist dabei hilfreich und erlaubt eine bessere Behandlung von Okklusionen, vergroessert aber die Anzahl notwendiger Modelle um ein Vielfaches.

Okklusionsmodelle

Bei Modellen mit Okklusionen wird das Modell eines Menschen genommen und der oder die durch Okklusion verdeckte(n) Teil(e) werden entfernt. Jedes Modell mit einem oder mehreren entfernten Teilen ergibt ein Okklusionsmodell. Wie die Experimente zeigen, erzielt man durch die explizite Modellierung von Okklusionen eine hoehere Erkennungswahrscheinlichkeit. Wie bereits angesprochen, erhoeht sich dadurch jedoch die Anzahl Modelle erheblich. Um diese Zunahme der Modell-Anzahl zu umgehen, fuehrt man ein generisches Modell ein.

Generische Modelle

Um die Anzahl Modelle zu reduzieren, wird die ausgewaehlte Menge Modelle uebereinander gelegt. Ein solches Modell nennt man dann ein generisches Modell. Das generische Modell ist die Vereinigung aller Punktmengen jedes gewaehlten Modells. Bei Kanten-Modelle geht man aehnlich vor. Obwohl sich mit generischen Modellen somit die Anzahl erforderlicher Modelle verringern laesst, ergeben sich Probleme beim Hausdorff Verfahren, da fuer das Matching viele Punkte fehlen koennen. Auch bei der Behandlung von Okklusionen erreicht man mit diesem Verfahren hoehere Erkennungswahrscheinlichkeiten. Um das Problem der fehlenden Punkte bei den generischen Modellen zu umgehen, fuehrt man ein Basismodell ein.

Basismodelle

Diese Art der Modell-Erstellung ist praktisch eine gegenseitige Ergaenzung der generischen Modelle. Statt der Vereinigung der ausgewaehlten Punktmengen - wie beim generischen Modell - nimmt man fuer ein Basismodell die Schnittmenge. Obwohl dieses Verfahren zunaechst vielversprechend erscheint, verbleibt ein Haar in der Suppe. In vielen Faellen ist die Schnittmenge der Punkte natuerlich viel geringer, so dass sie mit zufaelligen Punktmengen uebereinstimmen und zu Fehlerkennungen fuehren. Wenn man allerdings die Modelle gut auswaehlt, so dass viele von ihnen

untereinander weitgehend aehnlich sind, funktioniert dieses Verfahren recht ordentlich.

Rueckwaertserkennung bei Menschengruppen

Die Idee bei diesem Verfahren ist, dass wenn eine Person in einem Bild innerhalb einer Menschengruppe erkannt wurde, man auf das Vorhandensein dieser Person in einem vorhergehenden Bild, auf dem sie nicht erkannt wurde, schliessen kann. Da hierbei viele Probleme auftreten, wie etwa unterschiedliche Groesse, Bewegungen der Menschen, Bewegungen der Kamera usw... wurde zu Untersuchungszwecken eine eingeschraenkte Umgebung gewaehlt. In dieser eingeschraenkten Umgebung wurden mit der Rueckwaertserkennung gute Ergebnisse erzielt. Im vorliegenden Fall wird zuerst die Anwesenheit einer Einzelperson erkannt und deren relative Position wird dann in das zeitlich davor liegende Bild einer Menschengruppe "rueckprojiziert". Moegliche Okklusionen spielen dabei eine wesentliche Rolle fuer das Gelingen des Erkennungsprozesses. In solchen Situationen ist dieser Fall besonders kritisch, da das Nichterkennen einer Einzelperson dazu fuehren kann, dass die ganze Gruppe nicht erkannt wird.

Schlusswort

Obwohl das Ziel der vorliegenden Arbeit die Erkennung von Menschen in von einaeugigen Kameras aufgenommenen Bildern ohne die ueblichen Einschraenkungen war, wurden die urspruenglich auf Graphen basieren Matching Verfahren mit neuen Nachbar Isomorphie Verfahren analysiert. Das fuer das Matching sehr zuverlaessige Hausdorff Verfahren wurde erweitert, um Menschen anhand von vielfaeltigen Modellen und veraenderten Abstandsmessungen erkennen zu koennen. Da die Strategie der Vereinigung mehrerer Algorithmen, um trotz Okklusionen bessere Ergebnisse zu erzielen, im Grundsatz bereits parallel ist, lieferte die parallele Implementierung des Systems auf einem Cray 3TE richtige Ergebnisse in wesentlich kuerzerer Rechenzeit. Als moegliche Anwendung im Umfeld von Industrierobotern wurden drei beispielartige Situationen durchgespielt: eine in bekannter Umgebung, eine in unbekannter Umgebung und die dritte mit herumgehenden Menschen. Dazu dienten einfache Modelle mit Menschen als Hindernissen. Wenn es nicht gelingt, gute Modelle mit einer sinnvollen Indizierung zu waehlen, bleibt die computergestuetzte Erkennung von Menschen eines der nur schwer zu loesenden Probleme.

Abstract

In the realm of computer vision, the recognition of human beings in the images is one of the challenging problems which has ample applications in many fields from industry environments to surveillance systems. Most of the previous works on the problem were based on many strict assumptions which paved way for reducing the computation time to recognize. In this study, beginning with graphs to the real images, various strategies to recognize objects and human beings are developed which are based on graph matching and distance transformation methods leaving many strict assumptions. To exploit the inherent parallelism in the methods, parallel algorithms are developed and implemented on high-performance parallel systems viz. supercomputer Cray T3E.

As graphs are the powerful representation of objects, graph matching is considered primarily. Initially, A^* *Algorithm* is used for optimal matching of graphs. With *lower bound* and *upper bound* techniques, the computation time of the A^* Algorithm is reduced considerably. With various strategies for modifying heuristic functions and expansion mechanisms, the efficiency of the A^* Algorithm for optimal matching of graphs is analyzed. A new isomorphism (*Neighbour Isomorphism*) is introduced to reduce the computation time of *graph matching* enormously. The same isomorphism is used to find the symmetries in the regular polygons which are repeatedly attached at various corners of the polygons. Combining both neighbour isomorphism and A^* strategy, to match the postures of human beings from the indian classical dance, Bharathanatyam, a new algorithm is developed which produces accurate results.

Due to noises in real images, such graph based methods are not directly applicable. At the low level image processing, only corners and edges are used to recognize the human beings. The standard matching algorithms, *Chamfering* and *Hausdorff* methods are used to match the human beings in the images. New modified Hausdorff based measures are introduced to recognize human beings to the possible extent. The *Fusion Architecture* combining various algorithms to produce better results is discussed in the study. All the strategies including the Fusion architecture are implemented on *Cray T3E supercomputer* as they exhibit ample parallelism.

For the robots to reach the destination point without colliding with human beings in industrial environment, algorithms for navigation through obstacles are developed. In a known environment, the focus confined is to find the optimal path efficiently with parallel strategies. In unknown environments, the methods to minimize the cost of traversal are formulated. In case of going around like TSP (Travelling Salesman Problem), parallel

implementations on Cray T3E with communication among the processors are carried out to effectively find the optimal solution.

The crucial problem of occlusion is handled in a better way using *Occlusion Models* and *Generic models*. An attempt is made to describe the positional relationships between human beings in the sequence of images ontologically. The modelling of the group of human beings and their recognition are experimented with examples. The industrial applications with robots to optimize the distance covered and the scheduling of vision tasks onto parallel systems are also discussed. In a sequence of images, with the back propagation of relative positions of single human beings recognized separately, recognizing groups of human beings is also possible in restricted environments. The experimental results show that the recognition of human beings with strategies discussed, is possible and depends heavily on the models. The inherent parallelism in the strategies can be exploited with the efficient implementation on high-performance systems to reduce the computation time.

Succinctly, the aim of the study is to recognize the human beings in the images from monocular camera without usual constraints. Initially, the graph theory based methods for matching are analyzed with new neighbour isomorphism. The robust Hausdorff method for matching is extended to recognize the human beings with ample models and modified distance measures. As the strategy to fuse different algorithms to get better results despite occlusions is inherently parallel, it is implemented on Cray T3E which also produced correct results with appreciable reduction in computation time.

Chapter 1

Introduction

1.1 General Overview

With the dawn of technological innovations particularly powerful processors and variety of sensors, Image processing became viable and more useful. Lot of interests are evinced on Image Processing owing to the compelling demands prevailing in the new millennium. The quest of imitating a human being, a cornerstone problem in Artificial Intelligence or the pompous Turing Test, is widely investigated. Among other activities of human being, Vision is considered to be really challenging for computer scientists to emulate the functions of eyes with cameras.

Eventhough number crunching by computing systems was prevalent in the beginning stages of computing, the state of art of the computing systems with ubiquitous sensors paved way for image processing widely. The wide spectrum of the applications of image processing ranges from surveillance to medical applications, automobile industry to defense applications, robotics to helping physically handicapped persons, text analysis to image understanding and so on. Albeit the notion of pixel was present in texts or alphabets from natural languages, the images which are depicted by pixels posed crucial problems.

First of all, where is the relevant information in an image? How can the relevant information be extracted from the whole image? Which object resembles the group of the extracted portions or segment of the image?. These questions distinguished the images from characters which are well defined, properly arranged and legible enough to categorize as a particular character or alphabet of a natural language.

Visual input provides more information and only in recognizing the same

as known object is a problem which is investigated extensively. The tactile sensors or lasers or ultrasonic sensors may have some impact if the objects are continuously or frequently exposed to these sensors. However, camera is totally harmless and can be used continuously over a specified time. The related hardware viz.Frame Grabber enhanced the utility of camera with the computing system in a naive way.

Given the computing system with camera, the efforts are made to make the computing system more intelligent by understanding the images captured through the camera. Apart from emulating human visual activity, the applications of image processing require such facility to improvise the efficiency. For example, in environments where the machines and human beings interact, the machines are also expected to cause no damage to human beings. In various industrial environments, robots move among the workers to accomplish their tasks. Not only in such cases, better information input to the human controller can avoid severe damages as in driver assistance systems in automobiles. Thus, finding human beings to avoid any damage which is essential in such environments emerged as a new domain in computer vision(sometimes called as "Looking at People").

1.2 Image Processing

The primary reason for interests in digital image processing stems from two principal applications area: improvement of pictorial information for human interpretation and processing of scene data for autonomous machine perception [1]. To start with, the image is defined to be a two-dimensional light intensity function f(x,y), where x,y denote spatial coordinates and the value of f at any point (x,y) is proportional to the brightness (or grey level) of the image at that point. Such an image is usually referred as *monochrome image*. A multispectural image f is a vector-valued function with components $(f_1, f_2, ..., f_n)$ [2]. A colour image is referred as a vector-valued function with the components which denote the brightness values of each of the three basic colours, $f(x, y) = \{f_{red}(x, y), f_{blue}(x, y), f_{green}(x, y)\}$. Time-varying images f(x,y,t) have an added temporal argument. In reality, an image is a two dimensional distribution of light intensity on the focal plane of a camera pointed at a natural scene. A generalized image processing system is portrayed in Fig.1.1 [3].

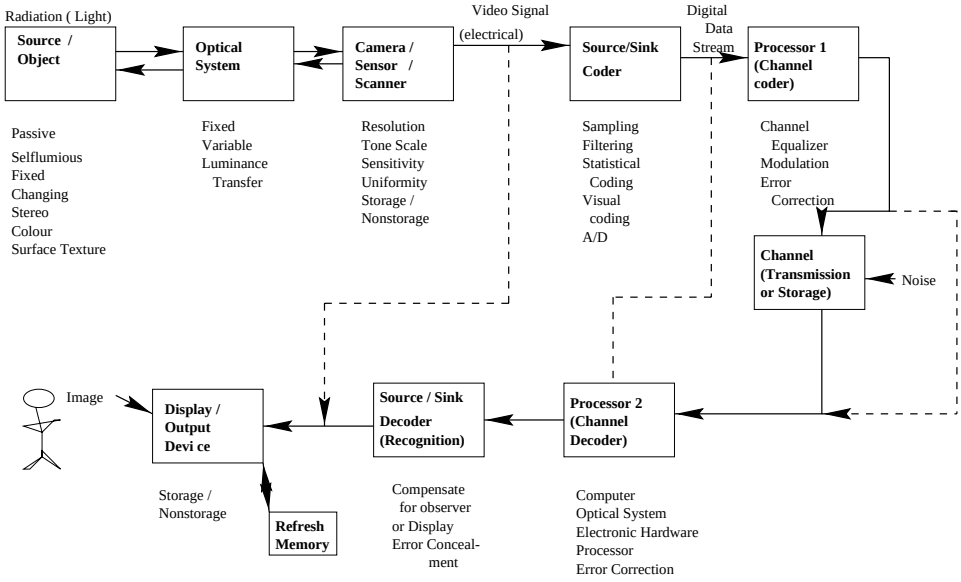


Figure 1.1: A Generalized Image Processing System

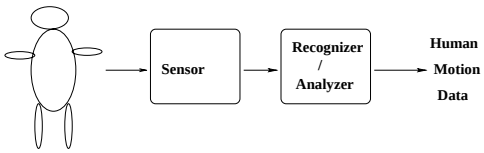


Figure 1.2: A simple sensor and recognizer system

1.3 Object Recognition

To answer the question, what is meant by object recognition, one may conclude naming an object in the scene. Sometimes in recognizing an object, it may be required to identify an individual object or a specific token (such as my wife). In some other cases, recognition means identifying the object as a member of a certain class or a type, (a girl) [4]. Moreover, an object may belong to a number of classes or categories simultaneously (e.g., my wife, an indian lady, woman, human being). An image may contain multiple different objects. Thus, the recognition of an object is an ability to retrieve information associated with an object, or a class of objects, that is not apparent in the image itself(for e.g., the name of an object). For any recognition system, sensors and recognizer(analyzers) form the crucial parts as in Fig. 1.2.

As the sensor complexity increases from passive sensors (which do not affect the surroundings or environment) to active sensors, the complexity of the recognizer or analyzer decrease [5]. So, a discussion on sensors as presented by the nature along with the proper use of the same follows next.

1.3.1 Biological Vision Systems

Intuitively a new born child can recognize the mother within few weeks after the birth without explicit learning. Effortlessly young children recognize the objects and classify them as well. In one study [6], pigeons were trained to sort 320 slides of natural scenes into two arbitrary categories, positive and negative. The pigeons learned the task rapidly, performed it with a high level of accuracy and repeated the same after two years without any additional practice. It may be recalled pigeons have only pea-pod sized brain. Even insects such as the bees use visual recognition for the purpose of navigation, finding the bee-hives and identifying the flower shapes [7]. It is universally well known that dogs easily identify the owners and distinguish them from other people.

Other Sensory Systems

Nature has provided a variety of sensors especially to animal kingdom. The cats use their whiskers as tactile sensors to get the information whether through the gap they can squeeze in. Snakes with their skins feel the vibrations on the surface to identify the approach of other animals. Deers and such other animals use their ears to detect the minute sound to escape from falling into the prey for some other animals. Ants use their "smelling" sense to get their food and follow the pheromonal trails despite being almost blind according to ethologists [8]. Bats use some sort of ultrasonic waves to detect objects while flying. Photophilic plants grow in the direction of region of light to synthesize their food. However, for computing systems trying to recognize objects, it is still a long distance goal.

1.3.2 Why is Object Recognition Complex?

To be more specific with object recognition, there is a large collection of patterns ($P = p_1, p_2, ..., p_n$) where each p_i represents the object in a different viewing position. Given an input pattern q, the direct approach will retrieve the pattern p_i which is most similar to q. The first question is what is meant by similar and secondly how large the patterns set should be so that it is sufficient to recognize every input pattern of the object. The similarity is

defined in many ways as it is a measure between two patterns. A simple case may be Hamming distance. But it can not be so efficient for its simplicity. However, the L_2 norm between the grey level images, is the sum of the squared differences between image intensity values at corresponding points.

Coming to the issue of how large the set P should be to recognize any input q as the object, beyond any iota of doubt it is prohibitively large. This is a crucial source of problem in object recognition. Secondly, what are the minimum number of features that must be present in the input so that it can be correctly recognized? Often due to occlusions in real environments, the input may not have all the required features. In such cases, how the situation can be handled, still remains to be investigated.

At the same time more fundamentally, what are the sources which bring the variability in the input pattern which is very difficult for recognition? As explained vividly in [4], the major factors are the following,

- Viewing Positions
- Photometric Effects
- Object Settings (or Occlusions)
- Changing Shape

In fact in real scenarios, an input pattern will have a combination of all these problems. For e.g., one input pattern for recognizing human being may be of the type from a view taken from the back (which means face, eyes, mouth, nose and such other features will be absent). In the presence of multiple coloured lighting sources, most of the colour based features become unsuitable. The silhouette of the person may be disproportionate also. Despite all these impediments, researchers are doing their best to cope up with the demands for accurate image processing.

1.4 Related Works

In human motion analysis survey [9], three major categories were mentioned namely, body structure analysis (both model based and non-model based), tracking (single camera and multiple cameras) and recognition (by state space method and Template matching methods). Another survey on the visual analysis of Human movements [10] distinguishes based on dimensionality (2D or 3D) combined with or without explicit shape models. A survey

on computer vision-based Human Motion Capture [11] discusses the general structure for systems analyzing human body motion as combination of Initialization, Tracking, Pose Estimation and Recognition.

In general the related works can be grouped based on the type of models used (stick figure-based, volumetric, statistical), the dimensionality of the tracking space (2D or 3D), sensor modality (visible light, infra-red, range), sensor multiplicity (monocular or stereo), sensor placement (centralized or distributed) and sensor mobility (stationary or mobile). However some more relevant issues are taken into consideration for forming Table 1.1. They are Initiation (Ini)(automatic (A) or predefined/manual), background (BG) (plain (P) or cluttered (C)), occlusion (Ocl) (permissible (Y) or not (N) or partially), human beings present (HuBe) (single (S) or Multiple (M)), motion (Mo) (predefined (P) or free (F)), motion detection (MoDec) (background subtraction (B), motion models, others (O)), size restrictions (Size)on human beings (Fixed (F) or not (N)), sensors (Sensor) (vision based (Monocular(Mo) or Stereo(St) or multiple (Mu) cameras), others like range (O)), and Segmentation (Segm)(colour (C), grey (G), others (O)). In the Table 1.1 noted contributions in the field of Looking at People are mentioned.

1.4.1 Typical Assumptions

In general there are lot of assumptions made which paved way for easy computation or recognition of the human being. A ranked list according to the frequency of usage discussed in [11] is reproduced here for the sake of continuity and further explanations.

Assumptions related to Movements

1. The object remains inside the workspace.
2. No or constant camera motion is allowed.
3. Only one person can be in the workspace at any time.
4. The object faces the camera at all times.
5. Movements must be parallel to the camera-plane.
6. No occlusion is allowed.
7. Slow and continuous movements are allowed.

Paper	*Ini*	*BG*	*Ocl*	*HuBe*	*Mo*	*MoDec*	*Size*	*Sensor*	*Segm*
[12]	A	C	N	S	P?	?	F	Mo	O
[13]	A?	C	Y	M	F	B	F	Mo	O
[14]	A	C?	Y	M	F	O	N	Mo	C
[15]	A	C	N	S	P	O	F?	Mo?	O
[16]	A	C	Y?	M	F	B	F	Mo	O
[17]	A	C	N	M?	P	O	F	Mo	O
[18]	A	C	Y	M?	P	O	F?	Mo	O
[19]	A?	C	Y	S?	F?	?	N	Mo?	O
[20]	A?	C	Y?	S	F	O	N	Mo?	O
[21]	A	C	N?	S	F	B	N	Mo	O
[22]	A	C	N?	S	F	B	N	Mo	O
[23]	A?	C?	N?	M	P	O	N?	Mo	O
[24]	A?	P	N	S	P	O	F?	Mo?	O
[25]	A	C?	Y?	M	F	B	F	Mo?	C,G
[26]	A	C?	N?	S	F?	B	A?	Mo	O
[27]	A	P	N	S	F?	O	F	Mu	O
[28]	A?	P	N	S	F	O	F	Mo	O
[29]	A?	C	N	S	F	O	F	Mo	O
[30]	A?	C	Y	S	F	O	F	Mu	O
[31]	A?	P	Y	S	P	O	F?	Mu	O
[32]	A?	C	Y?	M?	P	O	F	Mo?	O
[33]	A	P	Y?	S	P	O	F	Mo	O
[34]	A	C	Y?	M	F	O	F	Mo,O	O
[35]	A	C	Y?	M	F	O	F	Mo	O
[36]	A?	C?	N	S	P	B	F?	Mu	O

Table 1.1: Comparison of Different Recognition Systems in Field of Looking at People.

(Refer to Section 1.4 for abbreviations)

8. Only one or few limbs can move.
9. Fixed movements alone are allowed.
10. Object moves on a flat ground plane.

Assumptions related to Environment

1. Constant lighting alone is allowed.
2. The background must be static.
3. The background ought to be uniform.
4. Camera parameters are known.
5. Special hardware units are necessary.

Assumptions related to Object

1. The starting pose must be known.
2. The object is known for recognition.
3. Markers are placed on the object.
4. Special coloured clothes must be worn.
5. Tight-fitting clothes must be worn.

1.4.2 Major Three Strategies for Looking at People

Most of the methods were based on statistical moments or Single Gaussian or multiple Gaussian or Bimodal or Hidden Markov models to detect the presence of a human being in the scene. Chamfering method is based on distance transformations. Third method is to find a distance measure between the model and image based on the selected points. One such method is Hausdorff method which is robust also. Detailed discussions about the methods are presented in the following chapters.

1.5 Salient Features of My Approach

As assumptions helped a lot in reducing the computation and recognizing earlier, the principal deviation I made here is to leave many of the assumptions. This as expected increased the computation time. Yet the attempt to solve such things was fascinating to me atleast. The objects (human beings) need not to remain inside the workspace. Multiple persons are allowed. They need not look at the camera at all times. They can move in any direction and no restriction is imposed. The inclusion of occlusion, eventhough it is not new, really strengthens the use of the approach.

The relaxation of the assumptions related to environment is another vital issue in my approach. Most of the images are outdoor images. So, constant lighting is not possible, background can not be always static and uniform. Indeed in some extreme cases it is totally cluttered. No requirement of special hardware is recommended. However such things may reduce the computation time.

The major advantage of my approach is that there is no need of start positions of the objects. There is no need of wearing special markers as the approach is just grey level based. No compulsion on wearing specially coloured clothes or tight ones are made as normal dressing is allowed. But the number of models has to be increased accordingly.

Eventhough the decrease in the sensor complexity of the passive sensors increases the recognizer complexity, the passive sensor such as camera is chosen without the help of active sensors, as disturbing the environment will be viewed seriously by the social laws. As expected, the recognizer complexity is high. However, all possible efforts are taken to minimize the recognizer complexity.

Initialization is one of the major problems in such human motion capture or looking at people domain. At the start, normally a correct model of the person involved in motion is chosen and then the person is tracked. In my approach such initialization is not required. Of course if it is included, it will enormously reduce the computation time and further tracking will also be easier.

Generating models is one of the laborious problems. That too, they must be more generalized. The hand drafted models or simulated models are possible. However, the models are taken from the set of images and silhouettes and/or corners are taken for consideration. For chamfering edges are used. So, creating new models is not at all a tough task. The care must be taken however to include many poses. Occlusion models can also be included which is discussed in detail later.

Most of the systems use Kalman filter for tracking or particle based systems. Eventhough both these approaches can be incorporated, motion models are formed depicting various possible motions. Self-occlusion must be handled properly. Thus the models themselves can be used if motion models are included. As explained earlier, since the representation or model is image based, the time to construct the models is no longer a severe problem.

The crucial part of the whole system is recognition. In my approach, many possible systems can be integrated easily so that the combined system in turn produces the output positively despite taking more time. The methods such as Hausdorff and Chamfering are combined so that only those which are missed by Chamfering are tried by the Hausdorff method. This will reduce the unnecessary computation time.

The introduction of Fusion Architecture where different algorithms are combined so that different problems viz. Occlusion and scaling can also be handled to recognize human beings efficiently. The inherent parallelism in the Fusion Architecture paved way for parallel implementations.

As extension to make ontological descriptions or finally recognizing the actions after a period of time such as entering into the room or crossed each other at the door and so on can be easily inferred as the locations matched on the images can be used for that purpose despite having multiple persons. However care must be taken to identify the individuals to track them correctly after both of them meeting at a point. In restricted environments, recognizing groups of human beings is also possible with the back propagation of relative positions of single human beings recognized separately in a sequence of images.

1.5.1 The investigated Problem

The aim of this study is to recognize the group of human beings in the images from monocular camera without usual constraints. The graph theory based methods for matching are analyzed with new neighbour isomorphism at the beginning. The distance transformation based Chamfering is also applied to recognize the human beings. The robust Hausdorff method of matching is extended to recognize the human beings with ample models and modified distance measures. As the strategy to fuse different algorithms to get better results despite occlusions is inherently parallel, it is implemented on Cray T3E which also produced correct results in appreciably lesser computation time. At the outset, some navigation strategies to find the optimal path with obstacles in known and unknown environments are also developed.

Chapter 2

Matching based on Graph Theory

2.1 General Matching

Given two abstract representations, to find whether they match (atleast partially) lies in the heart of the development of artificial intelligence systems with human-like abilities such as computer vision. Due to the representational powers, graphs are often used for abstract representation. Parametric and syntactic approaches at the suitable levels are also available in the literature [37]. In pattern recognition and machine vision, graphs are used to represent the object models which are known a priori and the unknown objects which are to be recognized. Using these representation formalism, the recognition problem becomes a graph matching problem. Two main approaches namely, branch and bound methods and nonlinear optimization methods are widely used for graph matching as the complexity is NP. Eventhough other methods are also prevalent, graph isomorphism based matching and state space method such as A^* strategy are preferred in case of exact or optimal solutions which are computationally intensive. These approaches are used in error-tolerant and error correcting graph matching [38], [39]. Here, methods have been suggested to reduce the computation time for efficient graph matching.

2.2 Modified A^* strategy for Graph Matching

2.2.1 Previous Works

In most of the core application problems viz. artificial intelligence, code optimization in compilers, CAD and computer vision, manoeuvring the combinatorial search remains still to be solved efficiently. Especially in computer vision, the crux of the problem is to match two abstract representations (Graphs) [40].

As early as in 1964 [41], a heuristic program for testing pairs of directed line graphs for isomorphism was designed. Using representative graphs and reordered graph, another efficient algorithm for graph isomorphism is presented in [42]. With backtrack procedure, directed graph isomorphism is solved in [43]. A fast backtracking algorithm for the same not necessarily running in polynomial time was developed [44]. An algorithm for subgraph isomorphism using graph theoretical methods is presented in [45].

Mostly, two approaches viz. state-space method with branch and bound techniques [46] and nonlinear optimization methods with heuristic approximations [47] are employed to match graphs efficiently. Recently, noise included graph matching [48] and parallel algorithms [9] are also investigated. Various strategies and applicability of graph matching to computer vision is explained in [2].

However, the two approaches are combined to get the optimal matching always efficiently. The optimality is guaranteed by using A^* algorithm [51] with the proper h^* function aptly suiting to the problem. This demands the formulation of the problem in terms of A^* approach and developing heuristics for supplanting the upper bound for matching. The optimal results have been verified by the enumeration of permutations method.

2.2.2 The Matching Problem

The Definition

Given two graphs, G_1 and G_2 with vertex sets V_1 and V_2 along with edge sets E_1 and E_2, it is considered that the number of vertices in both the graphs are equal(say n). A cost matrix C is defined with c_{ij} as the cost involved in matching v_i of G_1 and v_j of G_2. Several issues are taken into consideration for incorporating them in the process of matching viz. degree of mismatch [9] and such others like difference between indegree and outdegree. The problem is to find a matching vector M where m_i is the vertex in G_2 matched with the vertex v_i in G_1 such that

$\Sigma c_{i,m_i}$ is minimal $\forall$ i, i= 1..n.

The Formulation

Each child node in the state-space of A^* (explained in the next section) denotes a partial assignment i.e., assigning a non-assigned vertex in G_1 with a non-assigned vertex G_2 apart from the already available such assignments made in the parent node. Here, $f^*(x) = g(x) + h^*(x)$ where f^* is the cost of the node, g is the cost of getting the node from the start node and h^* is the lower bound on the cost of arriving at a solution node from the node i.e., the sum of the static levels of the non-assigned vertices in G_1. The rest is the same as the general A^* strategy [51].

2.2.3 The New A^* Based Algorithm for Graph Matching

General A^* Algorithm

As the algorithm is based on the A^* algorithm, for the sake of clarity and explaining the algorithm, the general A^* algorithm used in most of the artificial intelligence problems is explained here as in [52]. In A^* algorithm, the state space graph is a tree called search tree. Each node in the tree corresponds to the assignment of a particular vertex in a graph with a specific vertex. All the internal nodes in the tree correspond to partial (or incomplete) matching and all external (leaf) nodes in the tree, correspond to either pruned node or complete graph matching. The problem here is to find the goal node, a leaf node corresponding to the optimal matching. Associated with a node v in the search tree is a cost function $f^*(x) = g(x) + h^*(x)$, which is an underestimate for the minimum cost of an assignment, given that it includes the partial matching. The function g(v) is the cost of the path from the root to v and the function h^*(v) is a lower bound estimation of the minimum cost function h(v), from the node v to a leaf node which corresponds to an optimal matching in the subtree rooted at node v. The search tree of A^* Algorithm with a random cost function for the graphs in Fig.2.2 is given in Fig.2.1.

The Heuristics Solution

To set the upper bound so that any node with the cost of partial matching or together with h^* also can be pruned, an effective heuristic

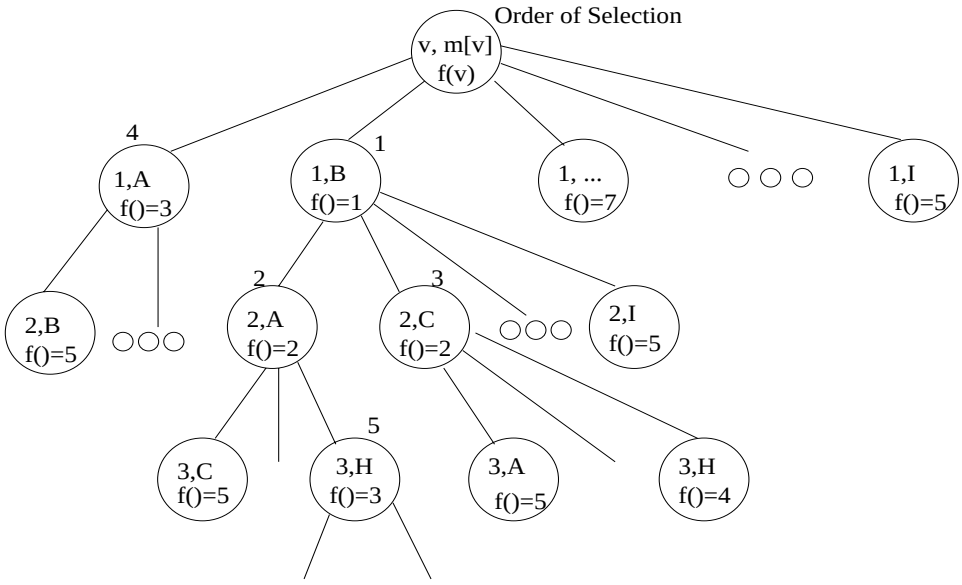


Figure 2.1: The Search Tree of A^* Algorithm for Fig.2.2

is defined here. A priority list is defined based on the following partial order,

v_i has more priority than v_j provided $\Sigma c_{ik}, \forall k$ is not less than $\Sigma c_{jk}, \forall k$.

The heuristic chooses each time, a vertex v_p from set of all non-assigned vertices such that no other vertex is having a higher priority. And, the vertex v_p is assigned to a vertex v_q in the set of all non-assigned vertices in G_2 such that c_{pq} is minimal considering all such non-assigned vertices. It can be also tried separately with N-Queen problems solutions as heuristics.

The Heuristics Function

The heuristic function is defined here as follows. At node x, let there be $V_1^{'}$ vertices which are already assigned. Then,

$g(x) = \Sigma c_{i,m_i}$, $\forall$ i $\in V_1^{'}$.

Now, to find the f(x) value, h(x) heuristic function is needed. To produce always optimal solution, indeed $h^*(x)$ is required. The $h^*(x)$ is defined as,

$h^*(x) = \Sigma c_{i,i'}$, $\forall$ i $\in V_1 - V_1^{'}$,

where $c_{i,i'}$ be the minimum in the row i. In fact, it is easy to verify that $h^*(x) < h(x)$ to ascertain the optimality.

2.2.4 New Techniques for Reducing Space and Time

Lower Bound

The lower bound is for the solution which is the minimum possible attainable solution. In the A^* algorithm, the algorithm has to continue even after finding a solution as it need not necessarily be optimal. Now the question lies how can it be proved that the given solution is the optimal solution so that the algorithm can be stopped at once. The only possible way is when the given solution is equal to the lower bound solution. As there could not be a better solution obviously, the algorithm can be terminated. Now, the problem boils down to finding the lower bound solution to the problem which is normally difficult in the general case. But it is not so in graph matching.

Let $c_{i,i'}$ be the minimum in the row i. Then, the lower bound is defined as the sum of all such row minima. i.e.,

$LB = \Sigma c_{i,i'}, \forall i = 1, n.$

The major problem with lower bound is that the possibility of many to many mapping is probable. However, in case of multiple similar objects, denoted as several occurrences of the same subgraph, this will indeed be more desirable. One should be always careful that all feasible optimal solutions need not necessarily be lower bound solutions. The main advantage is that if the given problem has the lower bound solution, the algorithm terminates at once it finds such solution, thereby reducing both the memory space required for the further expansions and the time to compute the same.

Upper Bound

The upper bound is a solution which is the minimum solution among the available solutions. In the A^* algorithm, the algorithm has to evaluate the function f(x) at every node. Supposing that f(x) is greater than upper bound, that node need not to be expanded further. This will not affect the optimality as anyhow by expanding the node, the solution obtained will be more than that of the already available solution.

However to start with, one should have a heuristic solution. So, the algorithm obviously needs a heuristic method to solve the problem. This also helps in another way drastically. Supposing that the heuristic solution is equal to the lower bound solution, then the algorithm stops without creating even a single node. Even otherwise, the heuristic solution found initially will serve as the upper bound. So, using upper bound, the number of nodes generated are minimized thereby reducing the memory space and CPU time.

2.2.5 The Algorithm for Optimal Graph Matching

1. Compute the lower bound solution, LB.
2. Find a heuristic solution, UB (using N-queen problem or so).
3. IF ($UB! = LB$) THEN
4. Construct the priority list of vertices.
5. c = 0 (* node count *).
6. Build the initial node N_0 and insert it in the list with f(N_0) = 0.
7. REPEAT
8. Select the node N_k with smallest f value.
9. IF (N_k is not a solution) THEN
   (a) Generate the successors i.e., trying with all unassigned vertices.
   (b) Do the following for each such vertices
   compare the vertex with all other vertices and assign.
   (c) FOR each such assignment N_i DO
   - Check whether it is already there in the list to eliminate the duplication
   - IF (already available) THEN
   Don't add the node
   ELSE
   Compute f(N_i) = g(N_i) + h(N_i) for this node N_i.
   IF (f(N_i) < UB)
   c = c + 1
   Insert it in the list
   IF (N_i is a solution) THEN
   IF (f(N_i) == LB) THEN
   Print the solution and quit.
   IF (f(N_i) < UB) THEN
   UB = f(N_i).
   ENDIF
   ENDIF
   ENDIF
   ELSE

Algorithm	*No. of nodes generated*	*CPU time in sec*
Variation A	285	0.183
Variation B	0	0.083
Variation C	45	0.083
Variation D	0	0.067
Permutation	362880	18.133

Table 2.1: Comparison of the variations of A^* with permutation Algorithm

Prune the node N_i
ENDIF
ENDIF

ENDIF
ELSE
Print the solution and quit

10. UNTIL (N_k is solution OR list is empty). ELSE
    Print the solution and quit

2.2.6 The Algorithms Developed with Variations

Four variations of the A^* algorithm with the new techniques [51] of lower bound and upper bound are developed here. Variation A is a simple A^* without employing any technique. Variation B is a simple A^* with the above techniques. Variation C is at each level of state-space tree only one vertex is selected based on the priority list. Variation D is the same as variation C together with these techniques.

2.2.7 Result and Analysis

Before analyzing the algorithm throughly, the snapshots of the algorithm for the input as given in Fig.2.2 are presented here.

Variation D

Here, for the sake of explaining a very simple function is taken as a cost matrix for the example. The cost function is taken to be the difference in the degrees of the corresponding vertices. As per the algorithm, the lower bound has to be calculated. For that, the static levels of the vertices have

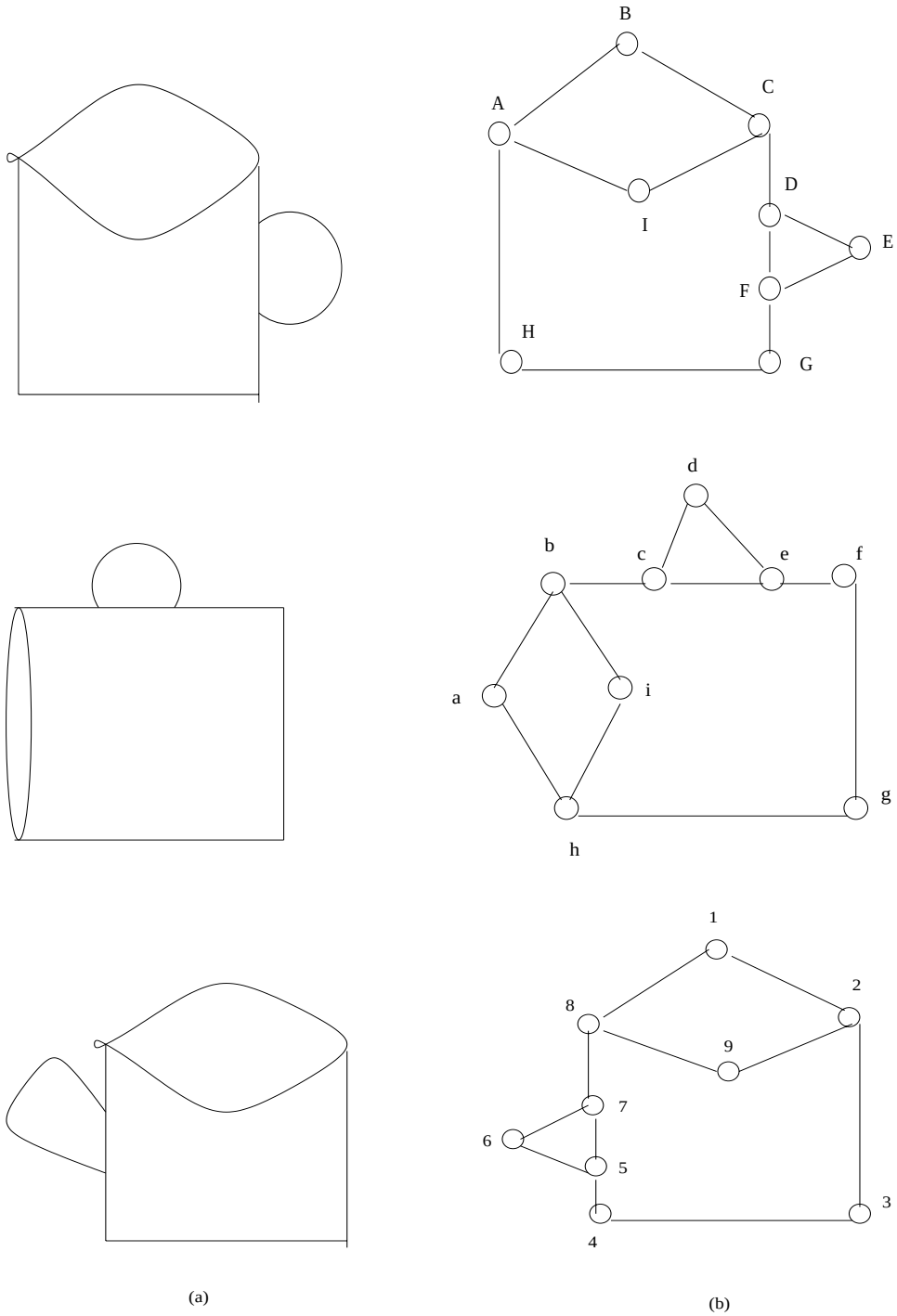


Figure 2.2: Recognition of Objects. (a) a real life Objects (b) a corresponding Graphs

to be computed from the cost matrix. From Fig.2.2, it is easy to compute the static levels. Since, minimum one vertex with the same degree is there, the static levels of each vertex is zero. Hence, the lower bound which is a sum of all the static levels is also zero.

Now, the heuristic solution as described earlier has to be computed. For a possible numbering of the vertices, given the cost function as the degree of mismatch (taken only the absolute value), the vertices form two groups with priority 4 and 5 depending upon the degree of the vertex 2 or 3 respectively. The heuristic algorithm chooses the vertex having no other vertex with higher priority. In the process, it finds a matching with the matching cost zero.

At this point, important event happens. Had the lower bound technique been not included in the algorithm, the algorithm would have proceed with the A^* algorithm. However, due to the inclusion of the new technique, the algorithm stops here, confirming the solution obtained as the optimal solution as it is equal to the lower bound solution (both are zero). It is valid because no solution can be better than a lower bound solution. So, any lower bound solution is the optimal solution. It is very easy to verify that the optimal solution need not necessarily be the lower bound solution.

Variation C

In this case, it is same as the general A^* algorithm excepting for the fact that only one node is expanded. The node selected for expansion is the node having the minimum f(x) value. For example in the case, the vertex zero will be matched with all other vertices making a node each time. However, instead of expanding all these nodes, only one node is selected and expanded. That is why there is a reduction from 285 nodes to mere 45 nodes. By expanding all nodes, more duplicate nodes alone will be generated at the expense of CPU time and memory space apart from eliminating such duplicate nodes also.

2.2.8 Analysis of the Results

The Table 2.1 is for the example given in Fig.2.2. It is very evident that the algorithms at any case outperform the general permutation method which also assures the optimality. The exact matching of vertices are given in Table 2.3.

The variation A is same as the A^* algorithm. By adding these techniques to the variation B, the reduction in both memory space and CPU time is

Algorithm	*No. of nodes generated*	*CPU time in sec*
Variation A	140	0.1
Variation B	0	0.083
Variation C	28	0.083
Variation D	0	0.067
Permutation	5040	8.7

Table 2.2: Comparison of the variations of A^* and permutation Algorithms for lesser number of nodes

Vertices of Cup	*Matched Vertices of Cup1*	*Matched Vertices of Cup2*
1	B	a
2	C	b
3	G	f
4	H	g
5	F	e
6	E	d
7	D	c
8	A	h
9	I	i

Table 2.3: Results of the Graph Matching Algorithm for Cups

evident. The prime difference between the set A,B and C,D is that in step 9, only one vertex is selected each time considering all unassigned vertices depending upon the priority. It may be also recalled by this, the optimality of the algorithm is not sacrificed. By allowing that only there will be lot of duplications which are also properly handled. The variation D is always better than Variation B, eventhough both produces the lower bound solution due to the power of the effective heuristic defined whenever possible. These variations are only for those who long for the optimality. The heuristic defined here in the chapter as well as the N-Queen problem will also serve the purpose of those who are not interested in optimal solution, but a quick reasonable sub-optimal solution. These algorithms are highly parallelizable. The permutation of the input cost matrix is carried out. However, the results are almost the same for the cases tried. To check, the number of nodes were reduced from 9 to 7 and the whole procedure is repeated. Table 2.2 portrays the same vividly.

2.2.9 Conclusion

The problems in artificial intelligence including computer vision which demand expensive and computationally intensive searches are tried for optimal solution. Eventhough several strategies are applied to solve the matching of abstract representation, mostly graphs, the A^* algorithm is chosen to ascertain optimality. The efficient A^* based algorithm for optimal graph matching with the incorporation of two techniques additionally is presented in the chapter. The lower and upper bound techniques help to reduce the number of nodes generated and the execution time to the appreciable amount. The heuristic used in the algorithms provides further reductions asserting the proper choice. The role of initial heuristic solution, in both upper and lower bound techniques are also exemplary and well suited to the graph matching problem. The variants of the algorithm presented here strengthen the claim of optimality clearly.

2.3 Symmetry based Graph Matching

2.3.1 Previous Works

Recently, there has been lot of interests evinced on matching objects in the images in the fields of Robotics, Satellite Imagery and Pharmaceuticals [50]. The paramount factor especially in the field of computer vision continues to be matching. The well-known algorithm such as A^* algorithm [51] [53] demands more memory and the computational time. The graph theory based methods have been used to the large extent [49] [48]. So, in the chapter also, a graph theory based isomorphism is introduced.

Most of the images have often regular repeated polygons, i.e., polygons which are regular(the edges with same length) are repeatedly placed at various points in the image. Thus, it will be useful if the basic units which are like building blocks are found.

The novel *Neighbour Isomorphism* defined in the chapter helps to categorize the vertices of the images into isomorphic groups and then the symmetric lines are formed to find the basic unit. Thereby, instead of matching the whole image, it is sufficient if the matching is done for the basic unit.

2.3.2 The Symmetry Problem

Given an image with lot of polygons connected in different ways, the symmetry lines have to be found to form the basic unit. Mathematically, let R be the set of all regular polygons. Let P_i denote the regular polygon with i

vertices, each denoted as v_i^j. Two vertices v_i^j and v_k^l are nailed together to connect the two polygons P_i and P_k. The remaining vertices are called hanging vertices. The given image contains a connected graph which is nothing but a regular repeated polygons. The *Neighbour Isomorphism* categorizes the vertices into disjoint class of isomorphic groups, g_i. Let there be n such isomorphic groups in the image. Try to find a line which will divide equally the number of vertices in each isomorphic group g_i. This line L_i is called the symmetry line of the image. There may be more than one line. Let H_1 be the set of all such vertices divided by the line L_1. Similarly, H_2 by another line L_2. Then, $H_{12} = H_1 \cap H_2$ is the basic unit provided no other symmetry lines are available [54].

2.3.3 The Neighbour Isomorphism Definition

The Neighbour isomorphism between two vertices v_1 and v_2 in a graph G_1 is defined as follows [51],

v_1 and v_2 are Neighbour isomorphic iff there exists k numbers of r distant neighbours for v_1 in G_1, then exactly there must exist k numbers of r distant neighbours for v_2 in G_1. (If the graph is a weighted graph, pairwise the corresponding vertices should be Neighbour isomorphic.)

Consider a graph with linear list of vertices v_0 to v_4. The Neighbour isomorphic groups of vertices are (v_0,v_4), (v_1,v_3) and v_2. The vertex v_0 is not Neighbour isomorphic with the vertex v_1 because v_0 has a neighbour at the distance of 4, where as v_1 has none at the distant of 4. Similarly, v_2 is not Neighbour isomorphic with v_3 as v_2 has two neighbours at the distant of 2 where as v_3 has only one. It is very easy to verify that all the vertices of a regular polygon are NI isomorphic, i.e., there is only one isomorphic group containing all the vertices. It is also true for hypercubes of all dimensions and complete graphs.

Mathematically, Let G be the graph with V as the set of vertices and E as set of edges. Let g_i denote the set of vertices which are neighbour isomorphic. Obviously, $g_i \subset V$. Let $V_i, V_j \in V$.

$V_i, V_j \in g_p$ iff $N(V_i, l, g_p) = r$ and $N(V_j, l, g_p) = r, \forall l = 1$, ..., diameter of G, where $N(V_i, l, g_p) = r$ denotes there are r number of l distant neighbours of g_p.

For a vertex numbered as *1* in the first figure of Fig.2.4, the function N with the following arguments forms the Table 2.4. In the Fig.2.4 and in all figures, if a vertex has a label i, then it belongs to g_i.

l (distance)	g_i *(isomorphic groups)*	*r (neighbours)*
1	g_1	2
1	g_2	2
2	g_1	1
2	g_2	4
2	g_3	1
3	g_2	2
3	g_3	2
4	g_3	1

Table 2.4: The Function N of Neighbour Isomorphism for the vertex numbered 1 in the first figure of Fig.2.4

2.3.4 Lemmas based on Neighbour Isomorphism

In the section, the lemmas are stated without proof as they can be easily derived from definition of Neighbour Isomorphism.

Lemma 1. If $P_i \in R$, then there exists only one neighbour isomorphic group g_1 containing all vertices. i.e., $V = g_1$. Refer Fig.2.3. As all the vertices are isomorphic to each other, they form only one group. It may also be noted that the dotted lines show the symmetry in the figures.

Lemma 2. If $P_i, P_j \in R$ and P_j is nailed at all the i vertices of P_i, then also all the i vertices of P_i form only one neighbour isomorphic group, g_1. (However, the isomorphism in P_j is changed.) Refer also Fig.2.4. It is important to note that the isomorphism in P_i is preserved where as the isomorphism in P_j is changed. The vertices with same number belongs to same group.

Lemma 3. If g_i and g_j are two neighbour isomorphic groups of P_k, then $g_i \cap g_j = \phi$. i.e., neighbour isomorphic groups are disjoint. It is easy to verify from the definition of Neighbour Isomorphism, if they are not disjoint, they would have merged into one group instead of two groups.

Lemma 4. Let $P_i, P_j, P_k \in R$. P_j is nailed at all the i vertices of P_i and P_k is nailed at all the remaining hanging vertices of P_js at j points, then also all the i vertices of P_i form only one neighbour isomorphic group. (But, the isomorphism is changed for both P_j and P_k.) This process can be done iteratively and then also the lemma holds good as shown in Fig.2.6. By adding the polygon P_j on each vertices of P_i, the isomorphic property of the vertices of P_i remains unaltered. Again, this is not true for the polygons P_j and the subsequently added polygon P_k also.

Lemma 5. Let $P_i, P_j, P_k \in R$. P_j and P_k are nailed at all the i vertices of P_i, then also all the i vertices of P_i form only one neighbour isomorphic group. (But, the isomorphism is changed for both P_j and P_k.) This process can be done iteratively and then also the lemma holds good as depicted in Fig.2.5. By adding any number of polygons on each vertices of P_i, the isomorphic property of the vertices of P_i remains unaltered. Again, this is not true for the polygons P_j and P_k.

Lemma 6. Let $P_i, P_i^1 \in R$, where P_i and P_i^1 differ only in size. If P_i^1 is smaller than P_i, P_i^1s are embedded in P_i at all the vertices with possible edges (normally two) too coinciding, then also all the vertices in P_i form only one neighbour isomorphic group. (However, the isomorphism is changed in all P_i^1s.) This is obvious from Fig.2.7.

Lemma 7. Lemmas 1,2,3,4,5 and 6 hold good in 3D also as shown in Fig.2.8.

Lemma 8. Neighbour isomorphism is invariant to translation, rotation and scaling with same scaling factors in all directions. As none of the conditions of neighbour isomorphism changes due to translation and rotation, it is obviously invariant. However, the distances are changed in scaling. That is why uniform scaling is required so that even if the distances changes, overall change in all the vertices will nullify the net effect on neighbour isomorphism.

Lemma 9. Nonregular polygons can also exhibit neighbour isomorphism. Few examples are provided in Fig.2.9.

Lemma 10. There exists atleast one symmetry line for all repetitions as per lemmas 1,2, 4, 5, and 6. As all regular polygons have atleast one symmetric line, divide each neighbour isomorphic groups into two sets unless they have odd number of vertices. In such case of odd number of vertices, the symmetry line should pass through one of the vertex in such group making the odd vertex common to both the divided sets.

2.3.5 Analysis of the result

If L_1 and L_2 are the two symmetry lines of the image(given as repeated regular polygons), and H_1 and H_2 are the corresponding set of vertices divided by L_1 and L_2 respectively, then probably $H_1 \cap H2$ gives the basic unit of the image. This may not always true. It has been observed in few repetition. If the image exhibits neighbour isomorphism, then there is finite chance in most of the cases that there exists atleast one symmetry line.

The major aspect is to find the symmetry lines so that basic units can be found easily. By constructing the neighbour isomorphic groups g_i, it is

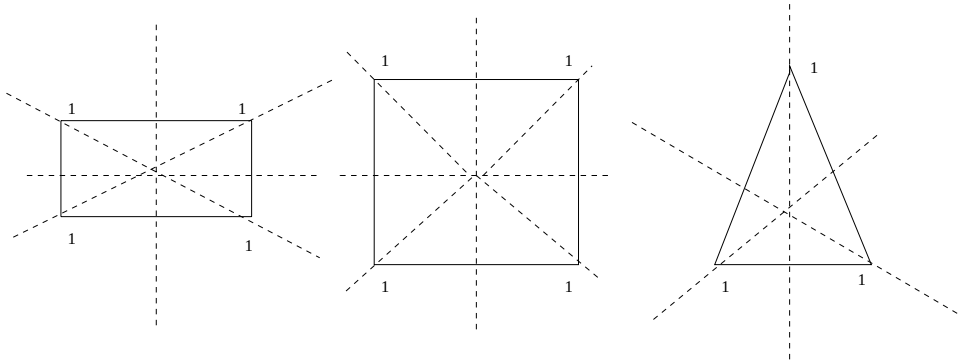


Figure 2.3: One level regular Objects

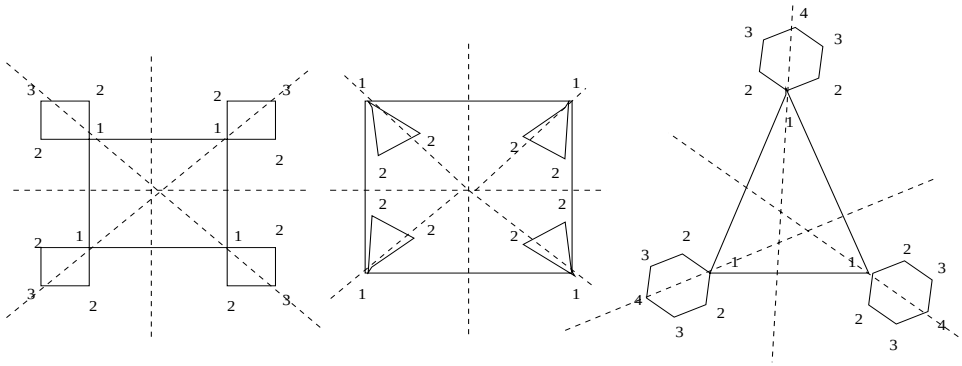


Figure 2.4: Two level regular Objects with nailing

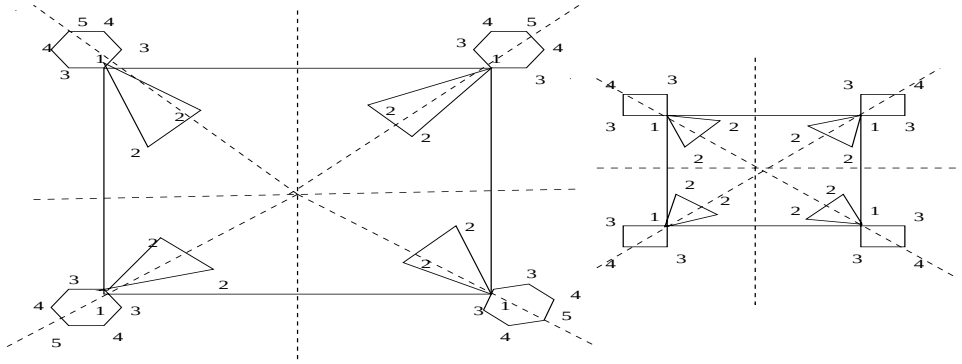


Figure 2.5: Two level regular Objects with double nailing

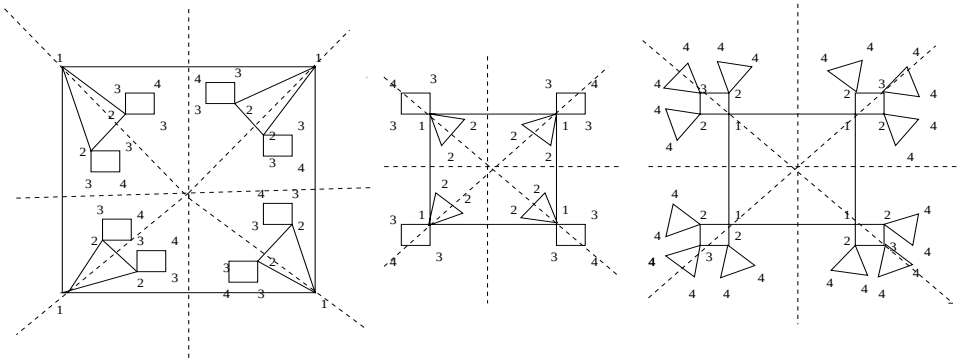


Figure 2.6: Three level regular Objects with nailing

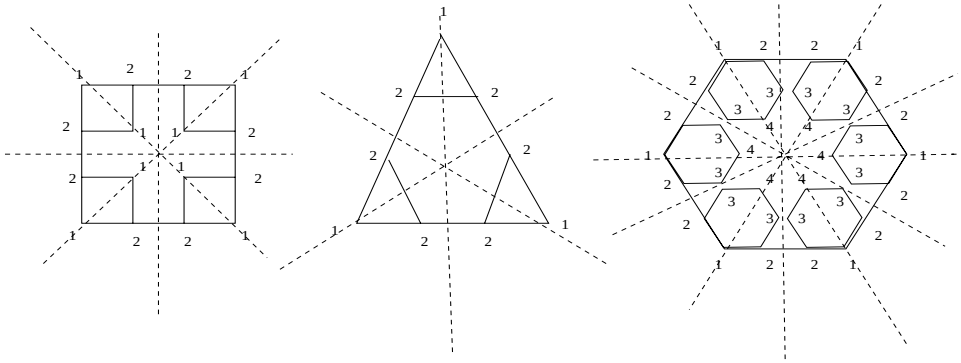


Figure 2.7: Two level regular Objects with embedding

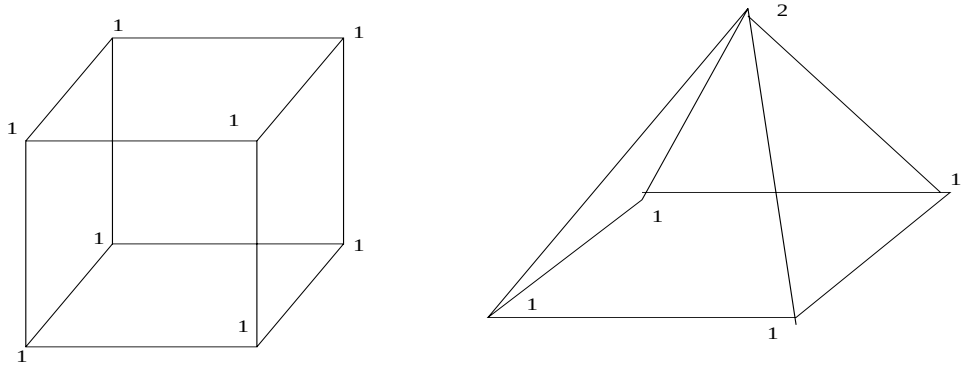


Figure 2.8: Three Dimensional regular Objects

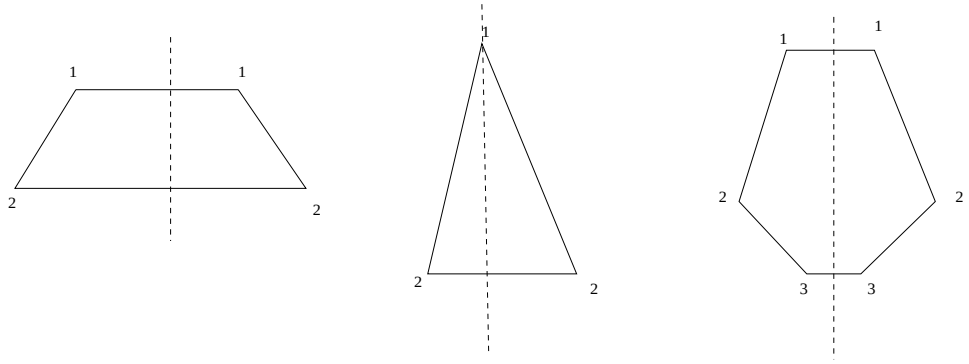


Figure 2.9: Nonregular Objects

O(n) time to the symmetry line, where n is the total number of isomorphic groups. Then, it is constant time to find H_1 and H_2 and the basic unit also.

Moreover, if there exists L_1 and L_2 for P_i, the basic unit will have only $i/4$ vertices. Suppose there are k symmetry lines, the basic unit will have only $i/2^k$ vertices. This shows that the order of reduction is very high for matching. Because, no longer the whole image has to be matched, it is sufficient if the basic unit is matched. Thus, the method can reduce the amount of time required to match considerably which solves the paramount factor in many of the real-time applications.

2.3.6 Conclusion

In the chapter, a novel graph theory based isomorphism, namely, *Neighbour Isomorphism* has been introduced. This isomorphism categorizes the vertices into disjoint groups. From these groups, it is easy to form the symmetry lines of the image. If more than one symmetry line exists, the intersection of the vertices partitioned by the symmetry lines forms the basic unit of the image. Many properties of the isomorphism are stated as lemmas here. The most important is the invariance despite translation, rotation and scaling(with equal factors). If the basic unit of image is found, no longer the whole image has to be matched. But, it is sufficient to match the basic unit only. Thus, the method appreciably reduces the amount of matching which is the major impediment in computer vision problems.

2.4 New Isomorphism based Matching

2.4.1 Previous Works

In most of the core application problems in Robotics, Satellite Imagery and Medical Imaging, recognizing the crucial parts or structures in the given images continues to attract more attention. Thus, the paramount factor in these computer vision related fields is matching the objects in the images [40], [48]. Even though the representational power of the graphs is high, the graph matching problem is a classic NP-complete problem [48]. The problem of graph matching in general is solved by various methods, viz. backtracking [44], [43], branch and bound [46], heuristic approximations [47], state-space method [53], [52] and isomorphisms [42].

Lot of interests are evinced recently following different approaches, viz., Eigendecomposition [55], Graduated Assignment [48], Subgraph Isomorphism [38], Error Correcting [39], and Single Value Decomposition [56]. Some of these algorithms which are based on matrix inverse or decomposition and linear programming methods have problems with large values of n. Some state-space based algorithms and branch and bound algorithms are of exponential time worst-case complexity. Few error correcting and error tolerant methods concentrated on noise in the image instead of the complexity of the matching algorithm primarily.

The classical algorithms for graph matching compute an incremental vertex-to-vertex mapping consisting of a backtrack tree search, perhaps with forward checking [45]. The complexity of such methods is NP. Here, the Neighbour Isomorphism (NI) is used which efficiently matches two graphs in $O(n^4)$ where n is the number of vertices in both the graphs which can be weighted and attributed. Using the NI (heuristics), the vertices of the graphs are grouped mutually exclusively. Instead of matching all the vertices, only the relevant NI vertices alone are matched paving way for ample efficiency by reducing the number of matching operations. The results are also compared with the standard A^* algorithm for exhaustive enumeration approach to portray vividly the reduction in the execution time.

2.4.2 Graph Matching and Graph Isomorphism

Graph Matching

A weighted attributed graph G is a set of vertices V which are attributed and edges E which are weighted with nonnegative real value. It may be recalled vertices can also be weighted and edges can have attributes. Given

two weighted Graphs, G_1 and G_2, matching is to find a match vector M such that,

$\Sigma c_{i,M_i}$ is minimal $\forall$ i ranging from 1..n.

where c_{ij} is the cost involved in matching v_i of G_1 and v_j of G_2 and $M_i = v_j$.

Graph Isomorphism

Given two graphs, $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$, for every edge, e_{ij} between the vertices v_i and v_j in V_1, there exists an isomorphism f such that there exists also an edge e_{pq} in E_2 such that f(v_i) = v_p and f(v_j) = v_q.

Further Properties of NI (Neighbour Isomorphism)

1. g_p and g_q are two NI groups in G_i. Then, they are mutually exclusive. i.e., $v_i \in g_p$, then $v_i \notin g_q$ and vice versa $\forall v_i \in V$.

2. If g_p is a NI group in G_i and g_q is a NI group in G_j and $i \neq j$, g_p and g_q are similar iff $v_i \in g_p$ and $v_j \in g_q$, $N(v_i, l, g_p) = r$ and $N(v_j, l, g_q) = r, \forall l = 1$, ..., diameter of G of the respective graph.

3. NI is invariant to translation, rotation and scaling(with uniform scaling factors).

Use of the Properties in Graph Matching Algorithm

The first property is about the equivalence classes grouped by NI. By grouping a vertex uniquely to a class(group), the same vertex need not be considered again any more. This results in ample reduction in computation. Moreover, the feature paves way for designing parallel algorithm for the graph matching.

The second property is the crucial one for graph matching. Since the vertices in each graph are grouped into equivalence classes, it is obvious that there exists only one matching between the respective groups(otherwise it will violate the equivalence class principle). The crux of the reduction in matching is derived from this property. Instead of matching each vertex in one graph with every vertex in the other graph, here only the corresponding groups are matched. Then the vertices in the groups are associated depending upon their edge connectivity.

The third property is very important for image processing. Most of the times, the images to be compared differ in orientation and size apart from translation. As NI is invariant to translation, rotation and scaling (with

uniform scaling factors), using the method the images depicted as graphs can be efficiently matched. The examples chosen for demonstration explain the salient features vividly. Indeed the linear combination of the operations preserving NI invariance are permissible. So, given two graphs if one is a linear combination of the operations of the other, they will be matched by the algorithm given in the next subsection.

2.4.3 Graph Matching Algorithm using NI

1. For the graphs G_1 and G_2, individually NI is computed to form isomorphic groups (g_p, g_q).

2. For each g_p in G_1, following steps are executed for matching with each g_q in G_2.

3. If the number of vertices in g_p is equal to the number of vertices in g_q, continue the steps else match with another g_s in G_2 as g_q from step 2.

4. Let $v_i \in g_p$ and $v_j \in g_q$. If $N(v_i, l, g_p) = r$ and $N(v_j, l, g_q) = r, \forall l = 1$, ..., diameter of G of the respective graph, then g_p and g_q are similar. Goto step 2 with new pairs of g_p and g_q.

5. If all groups are matched, $\forall g_p \in G_1, \exists g_q \in G_2$ such that g_p and g_q are similar. (If it is onetoone, then the isomorphism itself is found as well.) else print the available matching.

6. To match individual $v_i \in g_p$, choose any $v_j \in g_q$ taking into consideration of the edges to preserve the connectivity.

7. . Repeat Step 6 for each such similar groups g_p and g_q.

2.4.4 Snapshots of the Algorithm

In this section, the salient features of the NI based graph matching algorithm are explained with the snapshots for the example in Fig.2.10, a wrench. First individually NI is computed for each graph of (a) wrench-1, (say G_1 the graph on the left) and (b) wrench-2, (G_2 the graph on the right). The isomorphic groups in G_1 are g_1 with vertices (v_0, v_6), g_2 with vertices (v_1, v_2, v_3, v_4) and g_3 with vertex (v_5). It is evident that v_0 in G_1 has two neighbours at a distance 2 and one neighbour at a distance 3. All the vertices except v_5, are having similar neighbours. But all of them are not having pairwise NI corresponding vertices. For example, for the vertex

v_0, there is a vertex v_5 at a distance of 3, which has degree 2. Except the vertices v_6 and v_0, no other vertex has a neighbour at the distance of 3 which is similar to v_5. Vertex v_5 has 2 neighbours at a distance of 3 which no other vertex has. So, only v_0 and v_6 are grouped together by NI and v_5 separately. In the same vein, all vertices in each group can be explained. Similarly for G_2, the isomorphic groups are g_a with vertices (v_0, v_4), g_b with vertex (v_1) and g_c with vertices (v_2, v_3, v_5, v_6).

At the second step, g_1 in G_1 is considered for matching. First, g_a in G_2 is considered for the match as it has equal number of vertices. At step three, it is obvious from the graphs that $N(v_0, l, g_1) = N(v_4, l, g_a), \forall l = 1$, ..., diameter of G_1 and G_2 respectively(in this case l = 2,3,5 and 6). It may be noted that g_b and g_c will not be considered at all as the number of vertices in these groups differ from g_1. It is also easy to show that instead of v_4 in g_a, for v_0 in g_a also the previous condition is valid. So, g_1 in G_1 and g_a in G_2 are matched together. It must be noted that unlike exhaustive cases where each and every pair of vertices are compared for matching, here very less comparisons are made to match which results in reducing the execution time of the algorithm.

The steps two and three are repeated for g_2 and g_3 matching correctly with g_c and g_b respectively. As all groups are matched, the matching is over. However, individually any vertex in the matched groups can be matched with any other vertex in the corresponding matching group and it can be random done also. For example, the vertex v_0 in g_1 in G_1 can be matched with either the vertex v_0 in g_a in G_2 or v_4 in g_a in G_2 so as the vertex v_6 in g_1 in G_1. Similarly matching is done only with the corresponding matched groups and the overall matching is found. One possible matching with pairs first from G_1 and second from G_2 as follows, (v_0,v_4),(v_1,v_5),(v_2,v_6),(v_3,v_2),(v_4,v_3), (v_5,v_1) and (v_6,v_0). The same is elaborately explained in Fig.2.11.

2.4.5 Time Complexity

As the important step in the algorithm is NI computation of both the graphs, the time complexity of NI plays the major role in the entire algorithm. For each graph separately the distance between the vertices can be found by any standard algorithm [57] in O(n^3). Now to find NI in each graph, for each vertex (n times) for each length (maximum n) for each other vertex (n times) for every length (maximum n times), $N(v_i, l, g_p) = r$ has to be compared. This results in the time complexity of O(n^4). Since rest of the steps take lesser time complexities, the complexity of the algorithm is O(n^4).

Model	Vertices	Time by NI (in Sec)	Time by A^* (in Sec)
Wrench	7	0.01	0.05
Humanface	24	0.05	18.43
DNA	32	0.09	129.18

Table 2.5: Graph Matching with NI

Vertices	Edges	Time (in Sec)
5	5	0.01
10	9-14	0.01
16	30-32	0.02

Table 2.6: Graph Matching with NI for Random Graphs

2.4.6 Experimental results and Analysis

Here as shown in Fig.2.12, three crucial examples, (a,b). wrenches for industrial applications, (c,d). DNA molecules (Thymine base) for genetics and (e,f). Human faces for image understanding are considered for graph matching. The Vertices can be attributed as O for oxygen, C for Carbon and so on as in DNA molecule. The edges have always weights specified. Otherwise, it is assumed to be the same unit for all. Time is given in seconds in Table 2.5 Table 2.6. The algorithm is coded in C and executed on Sun Ultra 10 Microsystems. The results are tabled in Table 2.5. The exact matched vertices are presented in Table 2.7 and Table 2.8.

Indeed, the results clearly shows that it is a graph isomorphism in the given cases even though it ought not to be so all the times. For checking the

Vertex in Wrench (a)	Matched Vertex in Wrench (b)
0	4
1	5
2	6
3	2
4	3
5	1
6	0

Table 2.7: Graph Matching with NI for Wrenches

DNA (c)	DNA (d)	Humanface(e)	Humanface(f)
0	17	0	19
1	14	1	18
2	15	2	0
3	16	3	17
4	13	4	11
5	10	5	12
6	11	6	9
7	12	7	10
8	1	8	13
9	0	9	16
10	7	10	15
11	6	11	14
12	5	12	8
13	8	13	7
14	9	14	6
15	2	15	5
16	3	16	4
17	4	17	3
18	18	18	2
19	22	19	1
20	21	20	22
21	29	21	23
22	28	22	20
23	30	23	21
24	31		
25	23		
26	24		
27	25		
28	26		
29	27		
30	19		
31	20		

Table 2.8: Matched Vertices of DNA Molecules and Human faces with NI

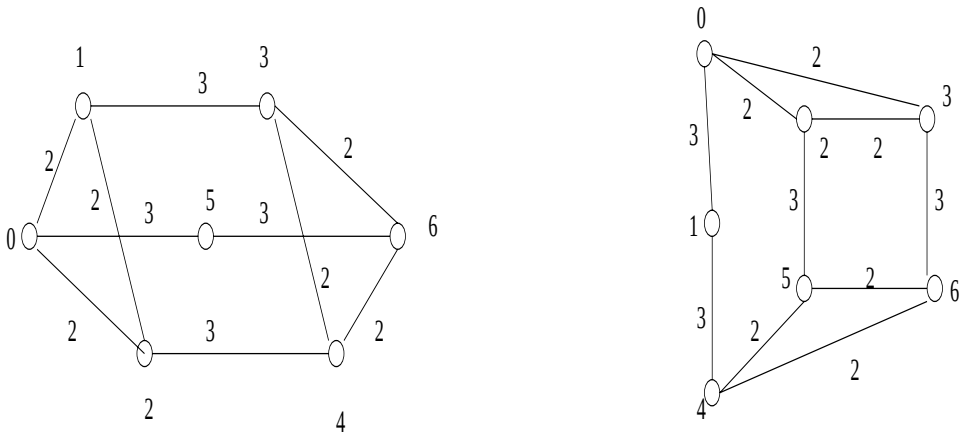


Figure 2.10: The given two graphs representing Wrenches

optimality here only the exact matching has the minimum cost and others are set as very high cost. With the amount of time, getting optimality with the stringent conditions to ensure isomorphism itself exemplifies the power of NI and the performance of the matching algorithm. In all these examples, the algorithm finds the graph isomorphism itself. However, it is not guaranteed for every case eventhough the optimal matching guaranteed. The examples are chosen from different fields to ascertain that NI is possible in many of the graphs from these fields.

Comparing the time required between the NI algorithm and the standard A^* algorithm [52] which matches exhaustively from Table 2.5, the reduction in the execution time is very impressive. It may be recalled as the number of vertices increases, the reduction in execution time also increases in these cases. This proves that as more and more vertices are grouped, the efficiency of the algorithm also improves.

For the three classical examples from various fields, the efficiency of matching with linear programming and backtracking methods are also investigated. For the first example with 7 vertices alone, it is possible to compare as other methods demand either enormous memory space or exponential computing time. For the first example, linear programming method took 17.75 seconds and backtracking methods found only sub-optimal solution also takes 0.05 seconds. There is no iota of doubt that NI outperforms all these methods.

For the random graphs by varying the number of edges also, the algo-

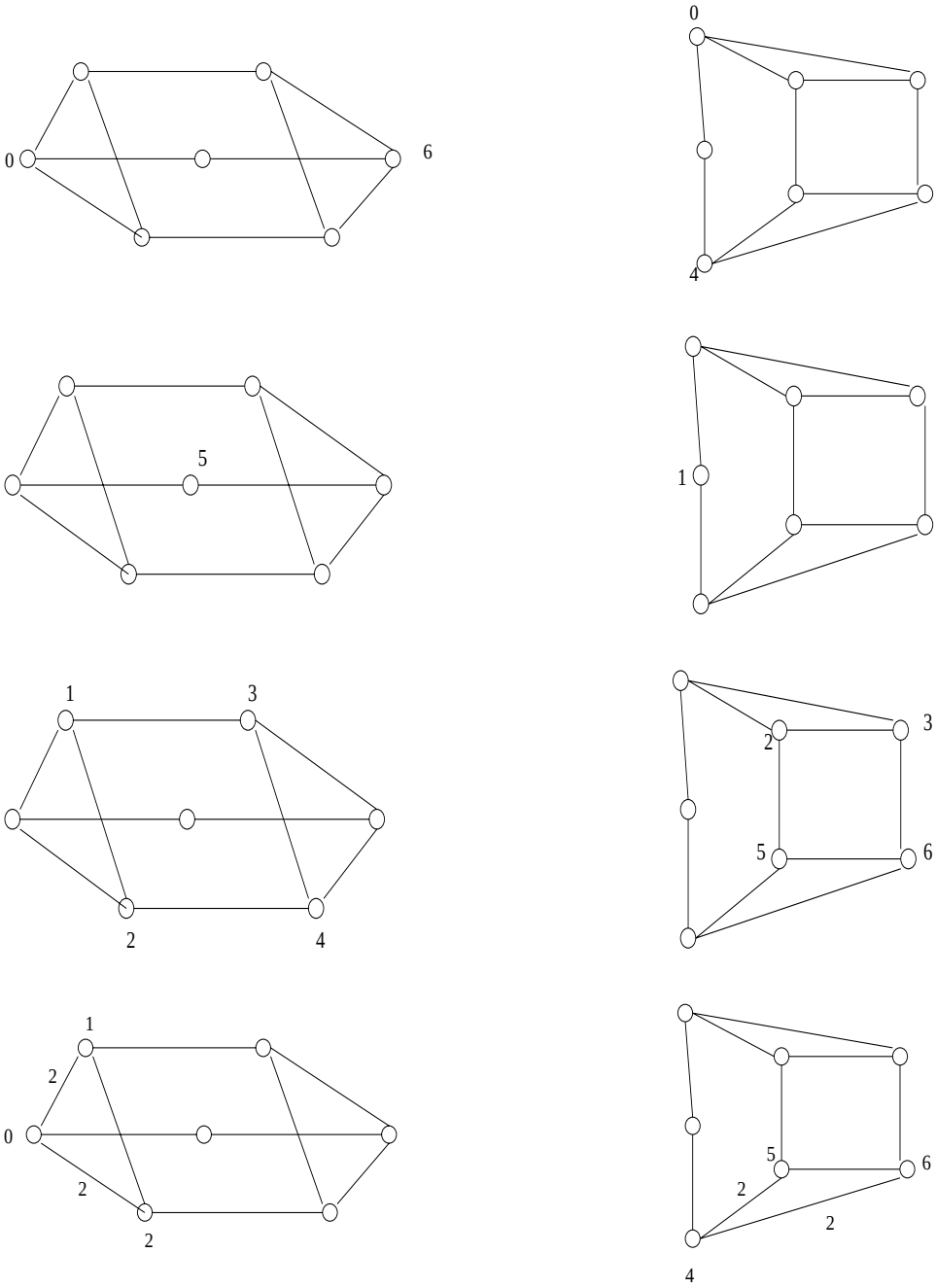


Figure 2.11: The isomorphic groups g_1 and g_a, g_3 and g_b, g_2 and g_c in Wrenches with matching vertices

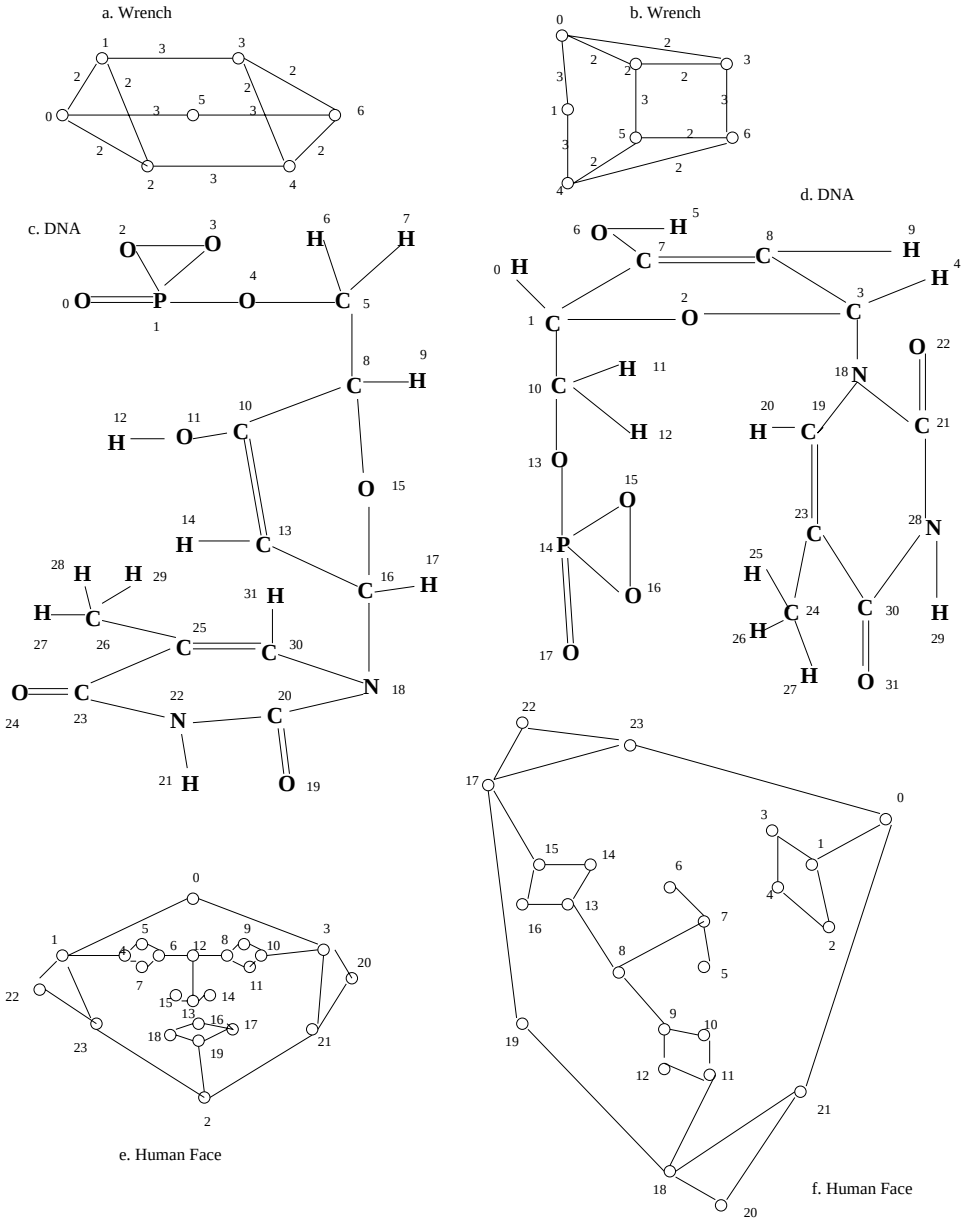


Figure 2.12: Examples for Graph Matching using Neighbour Isomorphism

rithm performs well as portrayed in Table 2.6. Time given in seconds in Table 2.6 are the mean time for such random graphs. Here, the graphs may exhibit NI or not. Despite the absence of NI, the algorithm matches the graphs correctly within the reasonable time. This asserts that the overhead involved in NI grouping is negligible compared to the total execution time.

2.4.7 Future Works

The order of the algorithm can be reduced to $O(n^2m)$, where m is the number of edges with suitable modification in the algorithm. Currently, the algorithm finds similar to graph isomorphism rather as graph matching, matching both ways. Incorporating subgraph isomorphism requires considerable modification in the algorithm. As the algorithm is inherently parallel, developing a parallel algorithm may not be tough. The inclusion of error-correcting and error-tolerant in the algorithm should also be investigated.

2.4.8 Conclusion

A novel method using Neighbour Isomorphism is introduced which efficiently matches two graphs in general and in many application specific areas. For recognizing the crucial parts or structures of the objects in such applications which are represented as graphs, the efficient matching of the objects with the models is inevitable. A graph matching algorithm based on Neighbour Isomorphism (NI) defined in the chapter is presented here. For the chosen classical examples from various fields, the algorithm found the optimal graph matching which in the examples concerned are indeed graph isomorphism itself. Instead of matching all the vertices, only the relevant NI vertices alone are matched paving way for ample efficiency by reducing the number of matching operations. The performance of the algorithm is demonstrated with crucial examples and random graphs. The algorithm outperforms many of the standard algorithms which is also investigated in the chapter. The order of the algorithm can be reduced and the algorithm can also be executed in parallel.

2.5 Bharathanatyam Postures - Posture Matching

2.5.1 Bharathanatyam Postures

One of the oldest classical dances in India is Bharathanatyam which has very well-defined movements and postures [58]. Here, combining both Neighbour

Isomorphism and the improved A^* Algorithm, given a set of bharathanatyam postures as database, if a particular posture is given as input, the combined algorithm efficiently finds the correct match. In case of a different input that is not in the database, the algorithm tries to match to the nearest resembling posture or postures also. It may be noted that the regions corresponding to the features viz. head, hands, foot and so are given along with the input so that they can form a graph. Since, the human structure is constant, edges are not specified as they are obvious. Once the graph corresponding the input posture is formed as shown in Fig.2.17, from the database of postures where the respective graphs for each posture is stored, according to A^* Algorithm the optimal matching is found efficiently.

2.5.2 The Combined Algorithm

The combined algorithm of Neighbour Isomorphism and the improved A^* Algorithm, is depicted in the figure Fig.2.13. A set of bharathanatyam postures presented in [58], are chosen to form the database. Some of the postures are presented in the following figures, Fig.2.14, Fig.2.15, Fig.2.16 and Fig.2.17. As shown in Fig.2.17, each posture is represented as a graph. When an unknown graph representing a posture is given, the algorithm tries to match with optimally closest graph in the set of graphs.

2.5.3 Results and Analysis

The algorithm is implemented on Sun Microsystems Ultra 10. One set of input is chosen from the database itself to check the veracity of the implementation. For all the cases in the database, the algorithm correctly identified the posture. The results are summarized in Table 2.9. Yet another set of input postures is tested with the available database so that the optimal matching posture or postures can be found. For the tested cases which are not in the database, the algorithm correctly identified the posture also. The timings are tabulated in Table 2.10. In all the cases, the timings presented are the averages over repeated trials. From the tables, Tab. 2.9 and Tab.2.10, it is clear that irrespective of whether the input is from the database or not, the computation time does not vary considerably. Because, the Neighbour Isomorphism categories the parts (features such as head) in the same time for similar size graphs, it becomes straight forward for the A^* algorithm to find the optimal match efficiently. This has also proved that for stick model of Human beings, Neighbour Isomorphism can be used effectively.

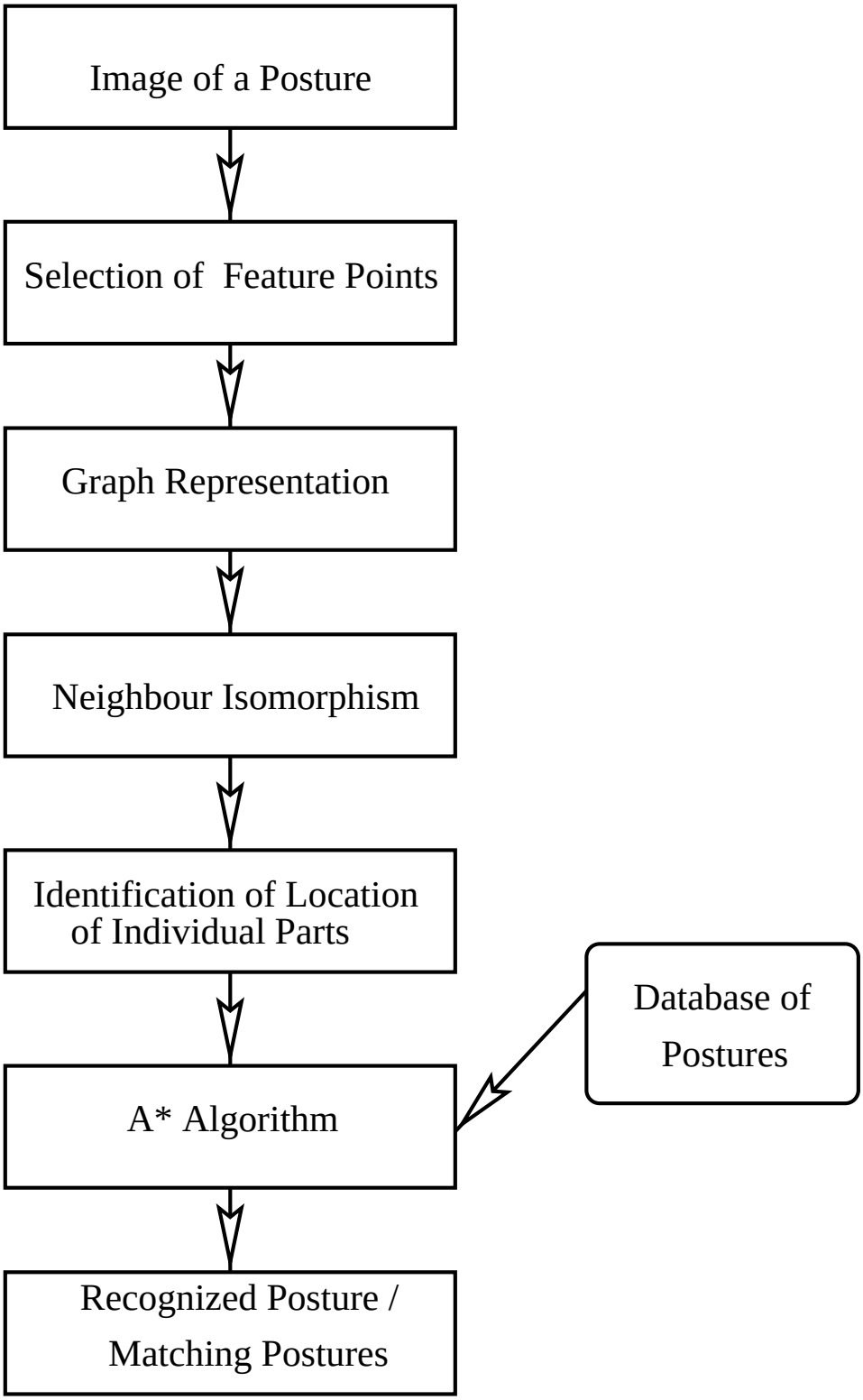


Figure 2.13: The Combined Algorithm for Matching Bharathanatyam Postures

Figure 2.14: Bharathanatyam Postures Set 1

Figure 2.15: Bharathanatyam Postures Set 2

Figure 2.16: Bharathanatyam Postures Set 3

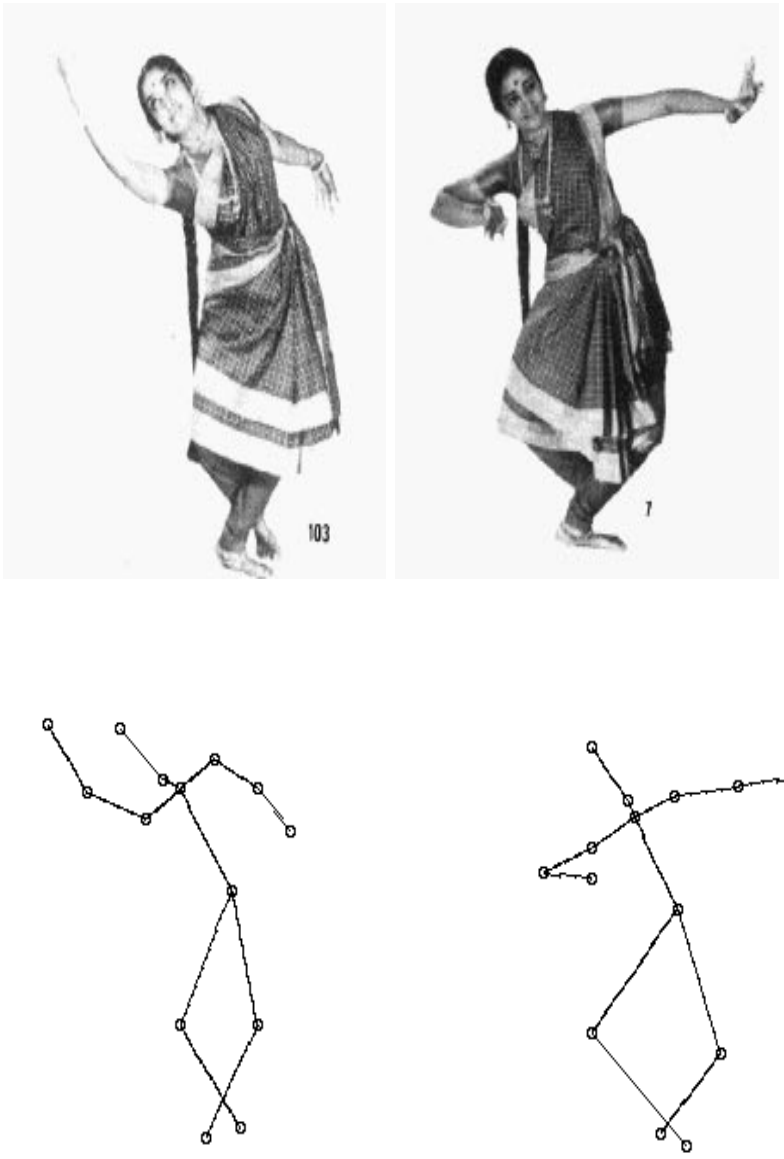


Figure 2.17: Bharathanatyam Postures Set 4 with respective Graph Representations

Model	Time in Sec	Found Correctly?
15-m7	0.14	yes
16-m11	0.12	yes
17-m25	0.14	yes
18-m34	0.13	yes
19-m42	0.12	yes
20-m48	0.12	yes
21-m58	0.13	yes
22-m60	0.13	yes
23-m70	0.12	yes
24-m77	0.12	yes
25-m89	0.12	yes
26-m92	0.13	yes
27-m103	0.11	yes
28-m112	0.12	yes

Table 2.9: Bharathanatyam Posture Matching (input from Database)

Input Graph	Time in Sec	The Matched Model
1	0.13	16-m11
2	0.14	17-m25
3	0.14	15-m7
4	0.13	20-m48
5	0.12	22-m60
6	0.14	18-m34
7	0.13	19-m42
8	0.14	21-m58
9	0.15	23-m70
10	0.14	27-m103
11	0.15	26-m92
12	0.12	24-m77
13	0.14	28-112
14	0.15	25-m89

Table 2.10: Bharathanatyam Posture Matching (input not from Database)

Chapter 3

Chamfering based Matching

3.1 Basic Concepts

3.1.1 Segmentation with Thresholding

In many of the pattern recognition problems, description of what is in the image or what are the possible descriptions of the various subsets of the image (Segments or objects) and their properties is one of the major corner stone problems. Pattern recognition systems must be capable of singling out the appropriate image subsets (segmentation). Eventhough there is no universal method for segmentation, from simple thresholding to colour cues, from connectivity to distance based approaches, there are many methods for segmentation.

Thresholding

Specifying a subset of a picture is equivalent to specifying its 'Characteristic function' i.e., the function whose value is 1 at the points of the subset and 0 elsewhere [59]. To segment an image or single out the subsets of the image, one way is to obtain the characteristic function of the subset by thresholding the given image. If f is an image, $f^{'}$ is the transformed image by thresholding such that $f^{'}(i,j) = 1$ if $f(i,j) > \theta$ (a threshold value) or zero otherwise. It can be extended in many ways including $f^{'}(i,j) = 1$ if $\theta_1 < f(i,j) < \theta_2$. By thresholding, one can get isophote images, smooth the image by eliminating noises both at the low and high levels, sharpen the image and match using cross correlation also. However, one of the major problems is to choose the value for threshold. But the distance based methods help to find the distance/similarity measure between the given two images or image and a

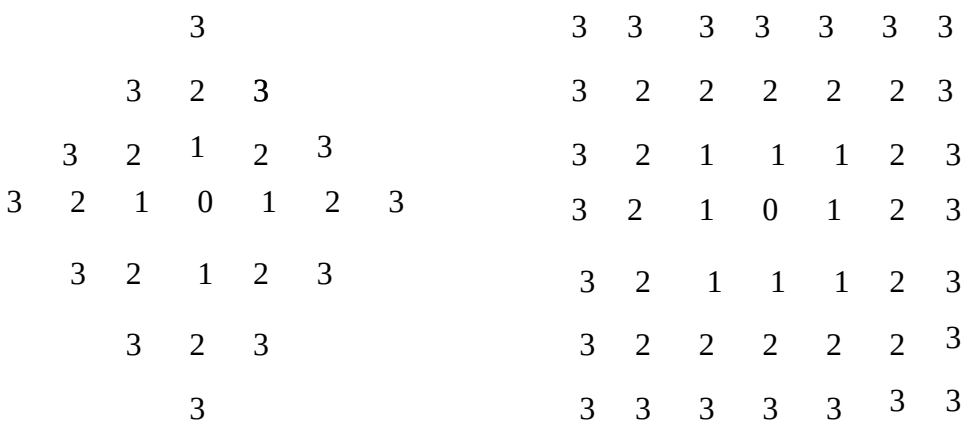


Figure 3.1: (a) Diamond (b) Square Distances

model to be matched.

3.1.2 Distance Functions

A function ϕ is called a distance function, if it is positive definite (ϕ(x,y)=0 iff x = y), symmetric (ϕ(x,y)= ϕ(y,x)) and satisfies triangle inequality (ϕ(x,z) <= ϕ(x,y) + ϕ(y,z), where x,y,z are points with coordinate positions.

The city block distance function is defined as ϕ_1((i,j),(h,k)) = |i-h| + |j-k|. Another function ϕ_2 is defined as the maximum of |i-h| and |j-k|. Fig.3.1 portrays a visualization of these distance functions.

The main point to be remembered which is used in distance transformation is the following theorem [59].

Let ϕ be a distance function, let (a_{ij}) be an n-by-n binary valued image, let S be the set of (i,j) for which $a_{ij} = 0$, let the sequence of integer-valued images $(a_{ij}^{(k)})$ be defined by

$a_{ij}^{(k)} = min_{\phi((u,v),(i,j))\leq 1} a_{uv}^{(k-1)} + a_{ij}^{(0)}$

for k=1,2, ... , where $a_{ij}^{(0)} = a_{ij}$. Then for sufficiently large k (for ϕ_1, k = 2n and for ϕ_2, k = n suffices),

$a_{ij}^{(k)} = \phi((i,j), S) \forall (i,j)$.

3.2 Distance Transformation

Given the image with model, the features are the most important and they have to be distinguished from nonfeatures in the image. The features can be

corners, edges, bright spots or areas of particular texture. There are many algorithms for finding edges and corners such as SUSAN [60] filters. In the approach, the edges are taken as features. The aim is to have a measure of the distance from each non-edge (nonfeature) pixel to the nearest edge pixel (feature) at each non-edge pixel. Obviously, edge pixels get the value zero. If the true Euclidean distance has to be found, it is not only computationally intensive, it also demands more memory [61]. So, a good approximation is needed to get these distances. The operation converting a binary image to an approximate distance image is called as a distance transform. Once an image is converted into a binary image with features points 0 and others some maximum value, any of the distance functions can be used for calculating the distances.

Based on the theorem cited above in the previous section, the global distances in the images are approximated by propagating the local distances, i.e., distances between neighbouring pixels over the image. In the approach of distance transformation, the local operation of propagating the local distances is iterated to get the closest global distances. The propagation can be done sequentially or in parallel. Such a sequential distance transformation is called chamfering.

It may be recalled a 3 * 3 neighbourhood is used for local distances. Chamfering 2-3 method has the maximum difference (compared to Euclidean distance) of 13 percent. The city block distance has the maximum difference of 59 percent. Chamfering 3-4 method used in the approach has only 8 percentage of difference [61]. It is essential to note that taking efforts to compute exact distances (Euclidean) from inexact edges due to noise in real images is a waste.

3.3 Chamfering 3-4

Given the binary image (feature pixels at zero and others at maximum value), here it is shown how chamfering 3-4 method can be programmed both for sequential and parallel implementation.

for(k=1 to some fixed value n) do
{
for(each pair (i,j)) do in parallel
$a_{ij}^{k} = min(a_{i-1,j-1}^{k-1} + 4, a_{i-1,j}^{k-1} + 3, a_{i-1,j+1}^{k-1} + 4, a_{i,j-1}^{k-1} + 3, a_{i,j}^{k-1}, a_{i,j+1}^{k-1} + 3, a_{i+1,j-1}^{k-1} + 4, a_{i+1,j}^{k-1} + 3, a_{i+1,j+1}^{k-1} + 4)$.
}

Iterations can continue until no changes occur.

The sequential algorithm performs both forward and backward passes from left to right, but from top to bottom and bottom to top respectively.

Forward step:
for i = 2, ... , row do
for j = 2, ... , columns do
$a_{ij} = min(a_{i-1,j-1} + 4, a_{i-1,j} + 3, a_{i-1,j+1} + 4, a_{i,j-1} + 3, a_{i,j})$.

Backward Step:
for i = row-1, ... , 1 do
for j = columns-1, ..., 1 do
$a_{ij} = min(a_{i,j}, a_{i,j+1} + 3, a_{i+1,j-1} + 4, a_{i+1,j} + 3, a_{i+1,j+1} + 4)$.

3.4 Matching using Chamfering

Once the distance transformation is over and the distance image is produced, the model can be matched. From the model, all the points (pixels) which are feature points (which form the edges) form a list of coordinate pairs, each pair being the row and column numbers of the corresponding edge pixel. The model is superimposed on the image at every possible points. In each case, using the list of coordinate pairs (translated depending upon the point of superimposition) the matching measure is calculated. The matching measure if it is a perfect match must be zero. In the approach, root mean square average is taken as matching measure.

3.5 Results and Analysis

The matching using Chamfering is implemented both on the Sun Microsystems and on Cray T3E. Obviously, for various examples the algorithm works perfectly. However in some very difficult conditions especially for night shots and highly cluttered environments notably inside shops, the algorithm failed to identify the persons correctly. In such cases, Hausdorff method performed well proving the robustness. Of course, it is not true that the algorithm does not work in such highly cluttered environments. Atleast in one of the given examples, the Chamfering perfectly recognized a person (eventhough left many such incidents which might be due to lack of adequate models). At the same time, it must be noted that the algorithm takes appreciably less amount of time compared to the Hausdorff method.

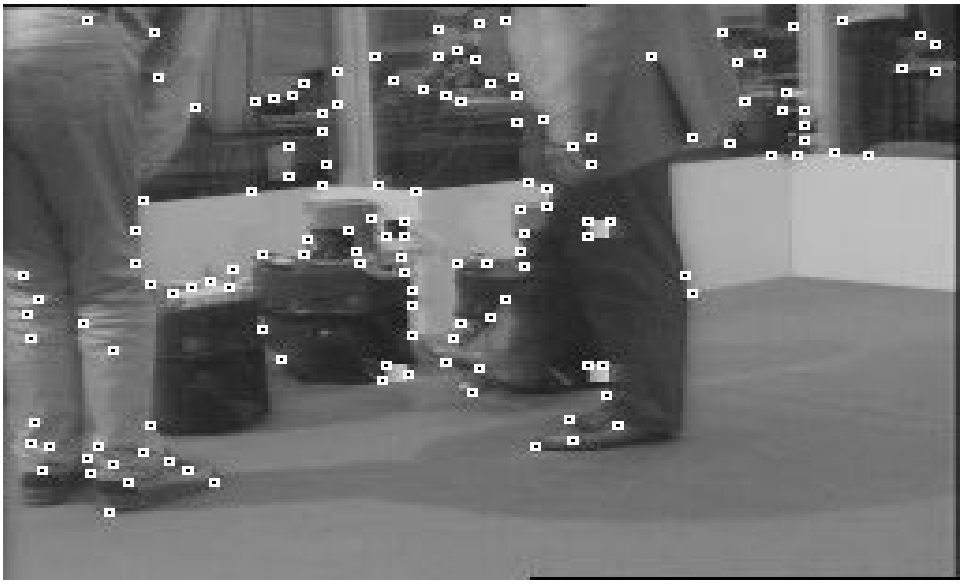


Figure 3.2: An image in a robotic field

For the image sets and models shown here, the computation times are tabulated. The figures in Fig.3.2, Fig.3.3, Fig.3.4, Fig.3.5, Fig.3.6 and Fig.3.7 are taken as input images. The models are depicted in Fig.3.8. A sample edge image of Fig.3.4 is shown in Fig.3.9 and the corresponding distance transformed image is presented in Fig.3.10.

As mentioned earlier, the algorithm performed well for both less complex cases and highly cluttered cases as presented in Table 3.1. The parallel algorithm also performed in the same lines without much improvement as far as the computation times are considered as tabulated in Table 3.2. The variation in the computation time depends on the size of the image, the size of the model and the position in the image where the instance of the model is present positively. In case of parallel algorithm not much reduction is visible as the processors have to exchange whenever they find an instance of the model in the image. This will help the other processors to avoid such areas. However, due to hardware constraints related to interprocessor communications, all processors must be synchronized to communicate among themselves which involves lot of idle time which is more than the computation time mostly. So, in these cases, irrespective of the other processors, a processor will compute in the area specified to it. The parallelization strategy is extensively discussed in the following Chapter. The robustness is lost in the fast computation and which is why Hausdorff method is chosen for

Figure 3.3: An image near city center in Rothenburg ob der Tauber

Figure 3.4: An image of students cross the road before the institute

Figure 3.5: An image of a dangerous crossing over the rails

Figure 3.6: An image of a busy cash counter inside a shopping complex

Figure 3.7: An image of a night shot near my house

Image	Model	Posi	Accu	Found	Time
1	1	0,0	100%	Yes	0.71
2	2			No	1.49
3	3	390, 200	100%	Yes	2.5
4	4			No	2.3
5	5			No	2.6
6	6			No	2.3

Table 3.1: Image Matching with Chamfering for Human being Recognition

Image	Model	Posi	Accu	Found	2 pcr Time	4 pcr	8 pcr
1	1	0,0	100%	Yes	0.43	0.36	0.37
2	2			No	1.51	1.26	1.22
3	3	390, 200	100%	yes	2.67	2.31	2.14
4	4			No	2.62	2.2	2.09
5	5			No	2.94	2.31	2.09
6	6			No	2.74	2.20	2.08

Table 3.2: Parallel Time of Chamfering for Human Being Recognition

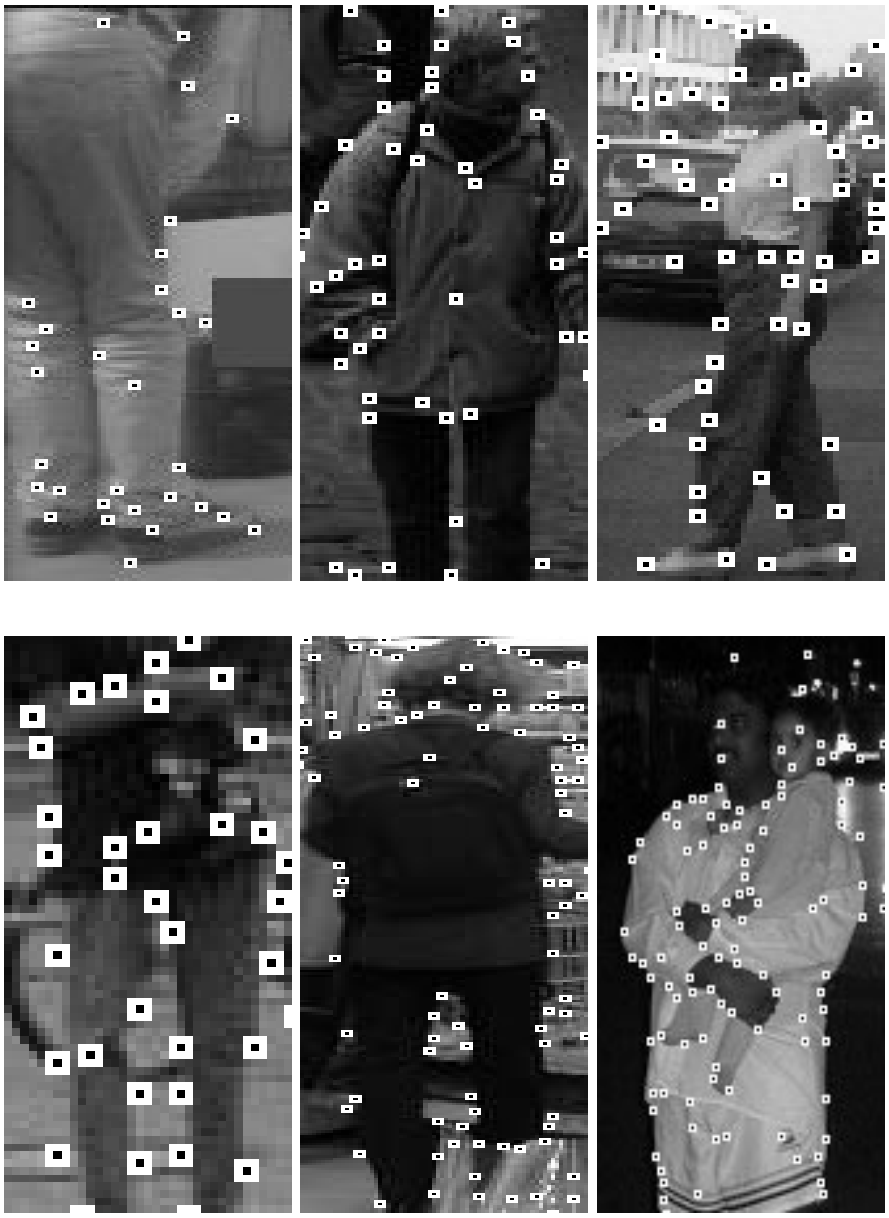


Figure 3.8: Some of the Models with Corners

Figure 3.9: The edge image of Fig.3.4

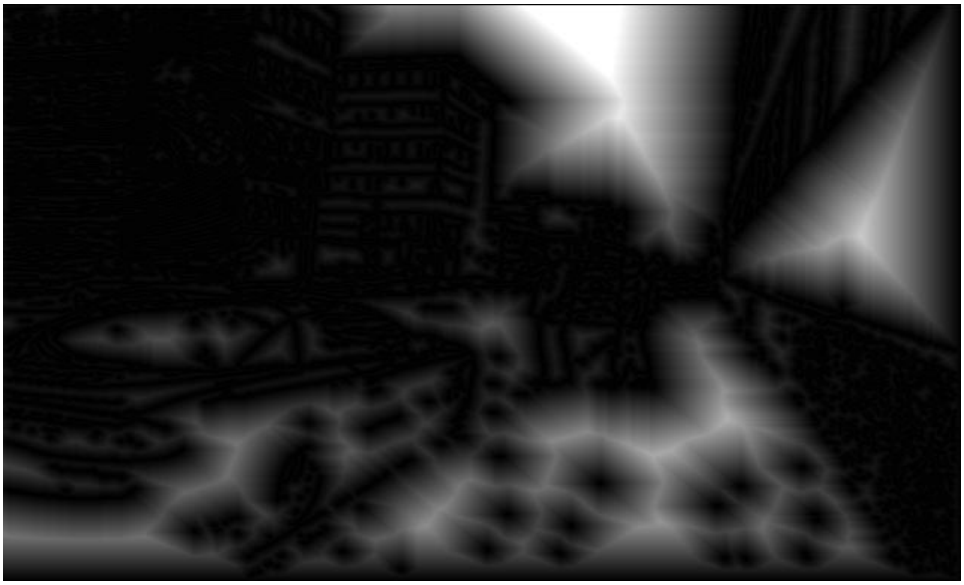


Figure 3.10: The Distance transformed image of Fig.3.9

further analysis in depth.

Chapter 4

Hausdorff Method for Matching

4.1 Definition of Hausdorff Method

The Hausdorff distance [12] is defined as follows:

> Let the two given finite point sets be $A = a_1, a_2, ..., a_n$ and $B = b_1, b_2, ..., b_m$.
> Hausdorff Distance H(A,B) = max(h(A,B),h(B,A)) where
> $h(A, B) = max_{a \in A} min_{b \in B} DN(a - b)$
> where DN is a distance norm (the distance between the two points a and b).

In Fig.4.1, sets of points are pictorially depicted to understand better. In Hausdorff method, each point (denoted by circles) in the point set A (left side stick figure) ranks each point (denoted by hexagon) in the point set B. That is each point in A tries to find which is the closest point in B and the distance is DN($a_i - b_j$). The maximum distance among such distances for each point in A is h(A,B). In the Fig.4.1, a possible nearest points based on some distance method is shown. This is as if one is superimposed on other and nearest points are found.

It may be recalled h(A,B) need not be necessarily equal to h(B,A). A simple example to check the veracity is with set $A = \{(0, 0), (0, 4)\}$ and set $B = \{(3, 0)\}$. Let DN be the Euclidean norm. Then, DN((0,0),(3,0)) = 3, DN((0,4),(3,0)) = 5. Hence, h(A,B) = 5. Since B has only one point, there is no need to first find the minimum unlike in the following case to find h(B,A). DN((3,0),(0,0)) = 3. DN((3,0),(0,4)) = 5. The minimum of both is 3. Since

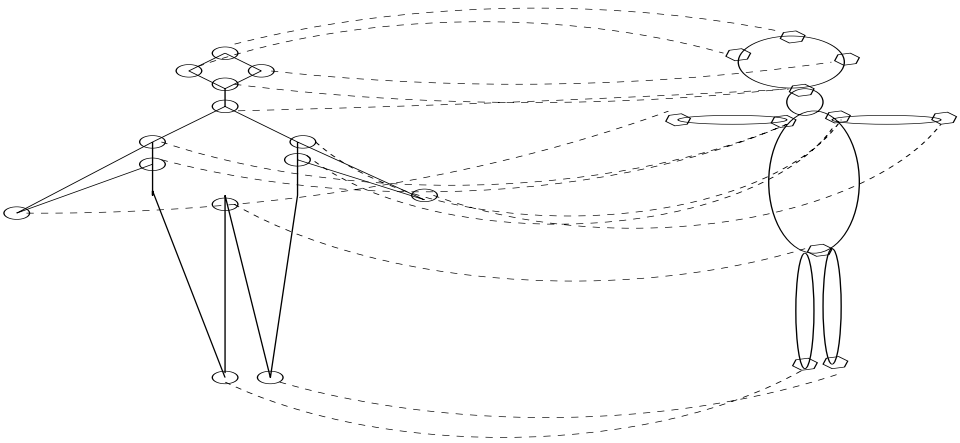


Figure 4.1: Two sets of points to be matched

there is only one point in B unlike the previous case, there is no need to find the maximum. Thus, h(B,A) = 3 which is not equal to h(A,B). However, H(A,B) is symmetric, i.e., Hausdorff distance is symmetric eventhough the directed Hausdorff distance need not be symmetric.

Starting from locating the objects using Hausdorff distance [62], Hausdorff distance has been applied for active tracking [63], tracking real scenes [64], visual target detection [65] and video sequence matching [66]. Modified measures of Hausdorff distances are used in plenty of applications, image matching [67], [68], [69], [70], model-based object recognition [71] and object matching [72]. In 3D segment matching [73], page similarities [74], human face recognition [75],[76] and occlusion contour detection [77], Hausdorff distance is applied.

4.1.1 Image Matching

Here, images which have to be analyzed are matched with the models which depict the pattern to be recognized, in the present case human beings. Using the general corner detecting algorithm (here SUSAN filter is used [60]), the corners of the images are found which serve as the point set B. Similarly, all the model images are filtered with the same SUSAN filter to find the corners and are stored as point set A. For the application, it is sufficient to find h(A,B). The main Hausdorff distance H(A,B) being a metric has to satisfy the symmetry property and that is why it is defined as maximum of the directed h(A,B) and h(B,A). The model is placed over all possible positions on the image and for each such position, h(A,B) is computed which

is computationally intensive. The model is found to be present in the image if h(model,image) is less than some predefined threshold. Against various models, the image is matched using the method.

4.1.2 Salient Features of Hausdorff Method

1. The Hausdorff distance between the two point sets H(A,B) is invariant to translation or rotation provided both A and B are translated or rotated with same measures.

2. The Hausdorff distance between the two point sets H(A,B) is scaled with the uniform scaling factor on all directions in case of Euclidean distance. If either the scaling factor is different or some nonlinear distance function is used, this scaling of H(A,B) need not to be true.

3. If a model is present in the image, h(model, image) (NOT H(model,image)) is unaffected by the presence of noise in image.

4. If a model is present in the image exactly, h(model, image) (NOT H(model,image)) is zero which is also same as the chamfering distance if feature points wise model is present in the image.

5. The Hausdorff method can handle occlusions effectively.

4.2 The Conventional Hausdorff Measures

Apart from varying the distance function used in Hausdorff method, various modifications are made. In [12] the proposed partial HD (PHD) measure is defined as directed distance of K^{th} rank,

$h_K(A, B) = K_a^{th} \in Ad_B(a)$

where $d_B(a)$ represents the minimum distance value at a point a to the point set B and $K_a^{th} \in A$ denotes the K^{th} ranked value of $d_B(a)$. This method need a parameter $f = K/N_A$ which has good matching when f = 0.6.

While comparing two binary images in [78] another directed distance CHD is defined as

$h_{P,Q}(A, B) = P_a^{th} \in AQ_b^{th} \in B\|a - b\|$

where $P_a^{th} \in A$ denotes the P^{th} ranked value of $Q_b^{th} \in B\|a - b\|$, with $Q_b^{th} \in B\|a - b\|$ representing Q^{th} ranked value of the Euclidean distance. Some parameters p and q need to be fixed to get good matching.

To compare the synthetic images contaminated by the four types on noise [72] a directed distance MHD is defined as

$h_{MHD}(A,B) = \frac{1}{N_A}\sum_{a\in A} d_B(a)$.

In order to eliminate outliers by replacing the Euclidean distance by the cost function, another directed distance $d_M(A,B)$ is defined as follows,

$h_M(A,B) = \frac{1}{N_A}\sum_{a\in A} \rho(d_B(a))$

where the cost function ρ is convex and symmetric and has a unique minimum at zero. Mostly $\rho = |x|$, if $|x| \leq \tau$ or $\rho = \tau$ if $|x| > \tau$. One more HD measure based on $h_M(A,B)$ by taking the averages of the i^{th} distance values is also presented in [78].

There are number of ways by which the Hausdorff distance calculations can be efficient. Principally calculating the relative distances requires more time. Lookup method storing the distance between the integer points 1..N,1..N is one such method. Using Distance transformations viz. chamfering also the distances can be computed. This can be also parallelized to get the results in lesser time. A proper scheduling of the parallel program onto parallel processors will further improvise to get the solution in lesser time.

4.3 Efficient Implementation of Hausdorff Method

4.3.1 Efficient Computation of Distances

Fundamentally, the distance function itself can be chosen such that the computation time is lesser. For e.g., city block distance may require lesser time than Euclidean distance as they involve finding square roots. At the same time for both cases, the distances can be computed once and stored in a big array and later only look up is involved and not the actual computation.

Always the distance transformations such as chamfering 3-4 can be used to find the asymptotically closer values instead of the direct calculations. This being two phase method, it is very efficient and unlike the look up methods, it requires neither precomputing the distances nor big array to store the distance for look up operation later.

In general, all these methods can be parallelized. Indeed further scheduling of the parallel program can still produce results in lesser time provided such MIMD systems are available for access.

From the resultant images Fig.4.2, Fig.4.3, Fig.4.4, Fig.4.5, Fig.4.6 and Fig.4.7, it is clear that the Hausdorff method is very robust compared to the chamfering method. However the computation time is more in the direct method i.e., without using distance transformation. The last column is for the same with the distance transformation as presented in Table 4.1. There

Image	Model	Position	No DT Time	With DT time
1	1	0,0	0.71	2.81
2	2	160,350	13.38	3.51
3	3	390, 200	24.08	4.31
4	4	127,235	31.51	4.3
5	5	330,200	8.7	4.15
6	6	240,120	2.3	4.14

Table 4.1: Sequential Times of Hausdorff with and without Distance Transform

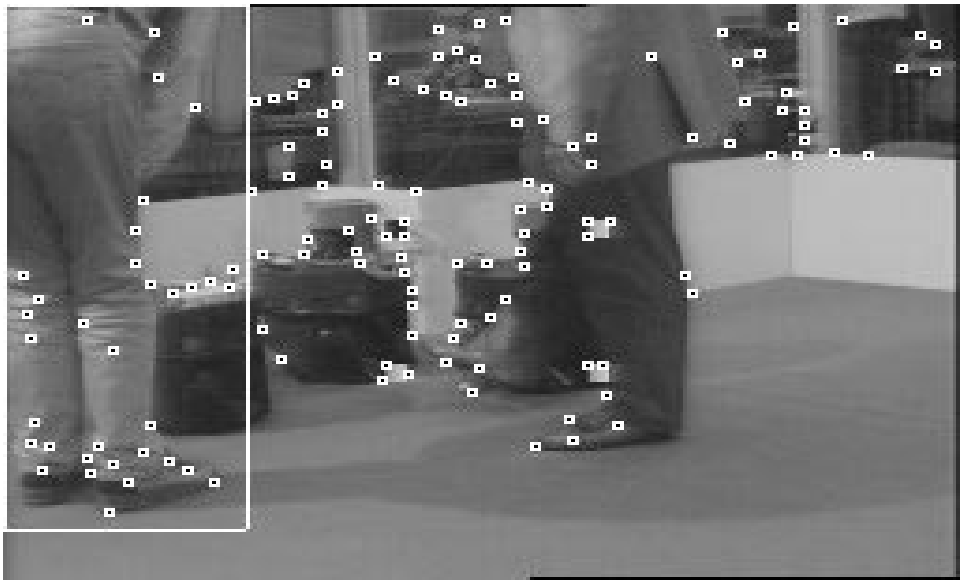


Figure 4.2: The Recognized Human being in the robotic field

Figure 4.3: The Recognized Human being near city center in Rothenburg ob der Tauber

Figure 4.4: The Recognized Human being on the road before the institute

Figure 4.5: The Recognized Human being in the dangerous crossing over the rails

is a slight occasional change of position by a pixel which is also accepted as 100 % accuracy as one pixel deviation does not matter much. But in the case of lookup method, it is very strange to note that computation times are 0.94, 16.65, 25.62, 16.97, 8,93 and 23.65 respectively. The main reason for such a huge computation times is due to reading a very large lookup array of size 1024 times of 1024. Referring each item also takes considerable time. It must be noted because of this the parallel algorithm with lookup method is not at all efficient as the file reading (I/O time) offsets the parallel computation time.

Figure 4.6: The Recognized Human being in the busy cash counter inside a shopping complex

Figure 4.7: The Recognized Human being in the night shot near my house

Image	Model	Position	Accuracy	C1 Time	C2 Time	C3 Time
1	0	50,2	100%	0.13	0.13	0.08
3	3	75,2	100%	0.14	0.17	0.14
10	20	205,60	100%	8.12	2.40	2.24
12	24	205,205	100%	9.10	7.05	7.08
15	31	160,160	100%	19.24	19.16	17.91
21	34	80,80	100%	2.62	1.94	1.99
23	52	240,110	100%	3.42	2.93	2.87
28	56	271,80	100%	6.56	6.41	6.18
34	45	365,195	100%	23.25	15.39	15.01
38	62	325,210	100%	26.52	15.59	15.43

Table 4.2: Sequential Times of Hausdorff Method for 1-1 exact position

4.4 Various Algorithmic Investigations on Hausdorff Method

As the Hausdorff method is very robust in recognizing human beings in cluttered scenes, the ability of the algorithm is investigated with different aspects in mind. First of all, one to one, one to many, many to one and many to many (Images Vs Models) variations are experimented with different sets (See Appendix D for image and model indices). The efficiency of the algorithms can be improved if the number of persons in the image is known or atleast the maximum number can be specified. Moreover, instead of exact locations, if the approximate locations are sufficient how the algorithms perform are also investigated. Finally what is the amount of false positives when the images do not have the respective models is also discussed in detail. The unit of time is seconds always.

4.4.1 Image Matching 1-1

In the category of matching one image with one model, four cases are investigated. Firstly, to see whether atleast one person (C1) is there in the image is implemented. Secondly, if the number of persons is known (C2) how efficient is the algorithm is tested. Thirdly, if exact position of the person is not a paramount factor but a detection of a person (C3) in the image, the impact of this aspect is analyzed comparing the efficiency. The results are tabulated in Table 4.2. Finally if the image does not have the model (C4), how the algorithm functions is also presented in Table 4.3.

Image	Model	C4 Time
0	0	0.57
4	1	0.52
17	20	19.95
20	33	0.63
16	15	16.32
21	40	1.85
25	53	2.73
41	55	0.01
33	43	18.06
37	47	19.46

Table 4.3: Sequential Times of Hausdorff Method for 1-1 No Match Cases

Model	No of Images	C1 Time	C2 Time	C3 Time	C4 time
1	2	0.69	0.65	0.68	9.55
1	4	0.95	0.92	0.75	33.61
1	8	78.98	78.40	80.44	108.75
16	2	15.97	15.92	15.63	7.21
16	4	35.05	35.10	34.42	29.54
16	8	66.74	73.80	67.56	83.09
50	2	2.98	2.97	2.87	0.11
50	4	3.68	3.62	4.06	0.42
47	8	10.75	12.06	10.66	25.43

Table 4.4: Sequential Times of Hausdorff Method for 1-n exact position

4.4.2 Image Matching 1-n

In this case, whether a model is present in the given set of images is analyzed with the same strategies as discussed in the previous subsection. Here, the number of images are doubled from 2 to 4 and then to 8 to analyze the performance as shown in Table 4.4.

4.4.3 Image Matching n-1

Given an image and a set of models, all the previous strategies are analyzed. The models are increased from 2 to 4 and then to 8. In Table 4.5, the results are presented.

Image	No of Models	C1 Time	C2 Time	C3 Time	C4 time
2	2	0.60	0.63	1.08	1.07
2	4	0.63	0.97	0.86	1.41
2	8	0.60	1.05	1.37	2.61
9	2	13.65	14.41	15.35	41.64
9	4	13.70	20.45	31.65	81.08
9	8	43.65	20.48	34.26	98.50
28	2	17.24	18.29	19.03	6.86
28	4	17.07	27.00	30.71	15.42
28	8	17.65	35.04	49.00	27.21

Table 4.5: Sequential Times of Hausdorff Method for n-1 exact position

No.Images	No.Models	A1 Time	A2 Time	A3 Time	A4 time
8	8	451.72	431.33	408.30	405.19
8	8	498.24	460.06	453.45	449.83

Table 4.6: Sequential Times of Hausdorff Method for n-n

4.4.4 Image Matching n-n

Given a set of images and models, the analysis is made for finding the exact positions if the number of persons are known (first case - A1), the approximate positions of the persons if the maximum number of persons are known (second case - A2), all persons without the knowledge about the number of persons with exact positions (third case - A3) and with approximate positions (fourth case - A4) as shown in Table 4.6.

4.4.5 Critical Investigation of Hausdorff Distance

The experiments are conducted to find how much similar figures among themselves vary as far as Hausdorff distance is concerned and how much they differ with different environmental brightness. The investigation is also carried out to find how much the figures of different sizes vary as far as Hausdorff distance is concerned. The two sets of figures with different postures are presented in Fig.4.8, Fig.4.9, Fig.4.10, Fig.4.11 and Fig.4.12 with l10 (10%more), l20 (20% more), s10 (10%less) and s20(20%less). Here, Hausdorff distance is set as h(A,B) and not H(A,B) which is the maximum of h(A,B) and h(B,A). In Fig.4.13, despite several hands, the matching is

Posture	1P1	1P2	1P3	1P4	1P5	1P6	1P7	1P8
1P1	0	85	104	246	278	285	167	247
1P2	267	0	112	183	233	222	130	184
1P3	277	116	0	187	319	258	233	144
1P4	280	114	197	0	296	258	219	193
1P5	281	91	62	78	0	82	84	71
1P6	280	71	68	65	76	0	86	61
1P7	270	123	84	147	168	186	0	143
1P8	269	299	282	270	286	296	226	0

Table 4.7: Hausdorff Distance h(A,B) for Set1

Posture	3P1	3P2	3P3	3P4	3P5	3P6	3P7	3P8
3P1	0	99	101	104	186	202	177	168
3P2	238	0	216	277	216	218	211	205
3P3	299	177	0	267	324	302	171	147
3P4	241	260	260	0	422	376	298	334
3P5	156	106	108	142	0	221	121	92
3P6	163	123	145	162	213	0	119	100
3P7	222	182	210	205	247	198	0	125
3P8	230	241	242	241	243	260	183	0

Table 4.8: Hausdorff Distance h(A,B) for Set2

found because some combination of two hands matched with atleast one model.

Important Conclusions about Hausdorff Distance

From the tables, Table 4.7, Table 4.8, Table 4.9 and Table 4.10, the following important conclusions are obtained.

- h(A,B) = h(B,A) mostly if A = B.
- $h(A, B) \neq h(B, A)$ mostly if $A \neq B$.
- Hausdorff distance h(model,image) is not affected by noise in image.
- Hausdorff distance h(image,model) may be affected by noise in image.

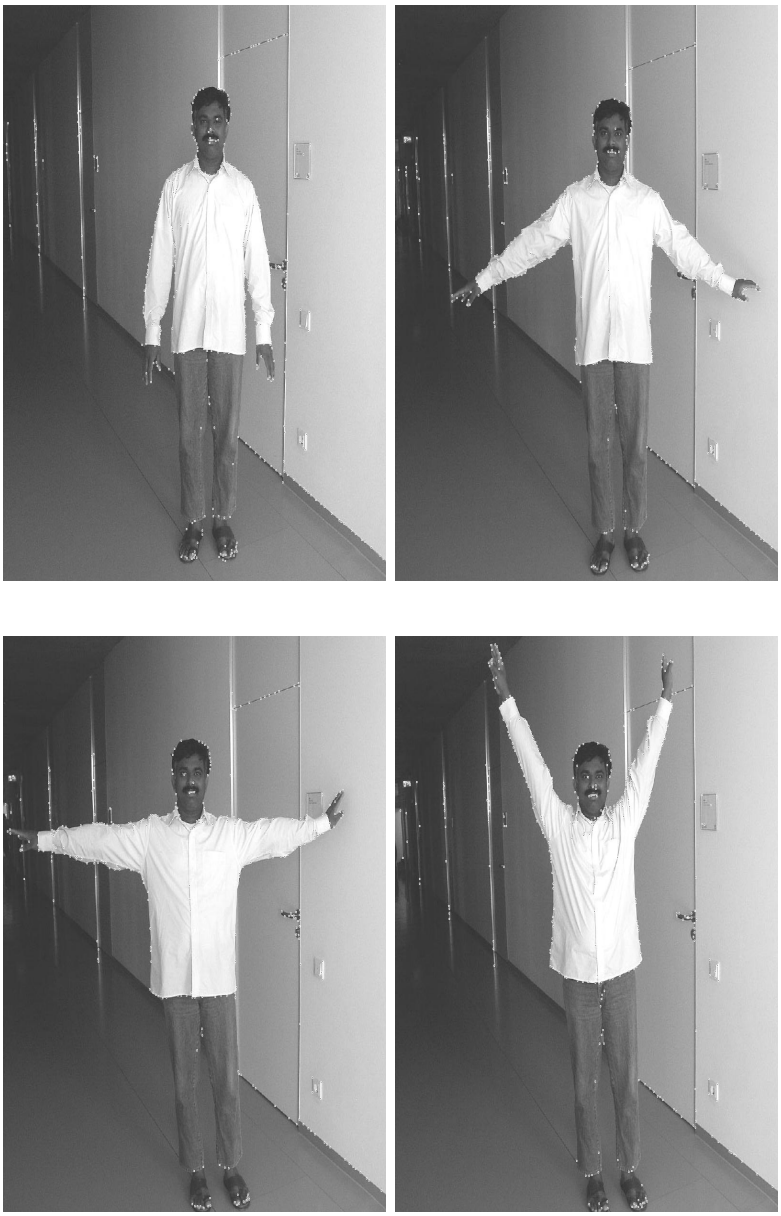


Figure 4.8: Different Postures 1p1-1p4

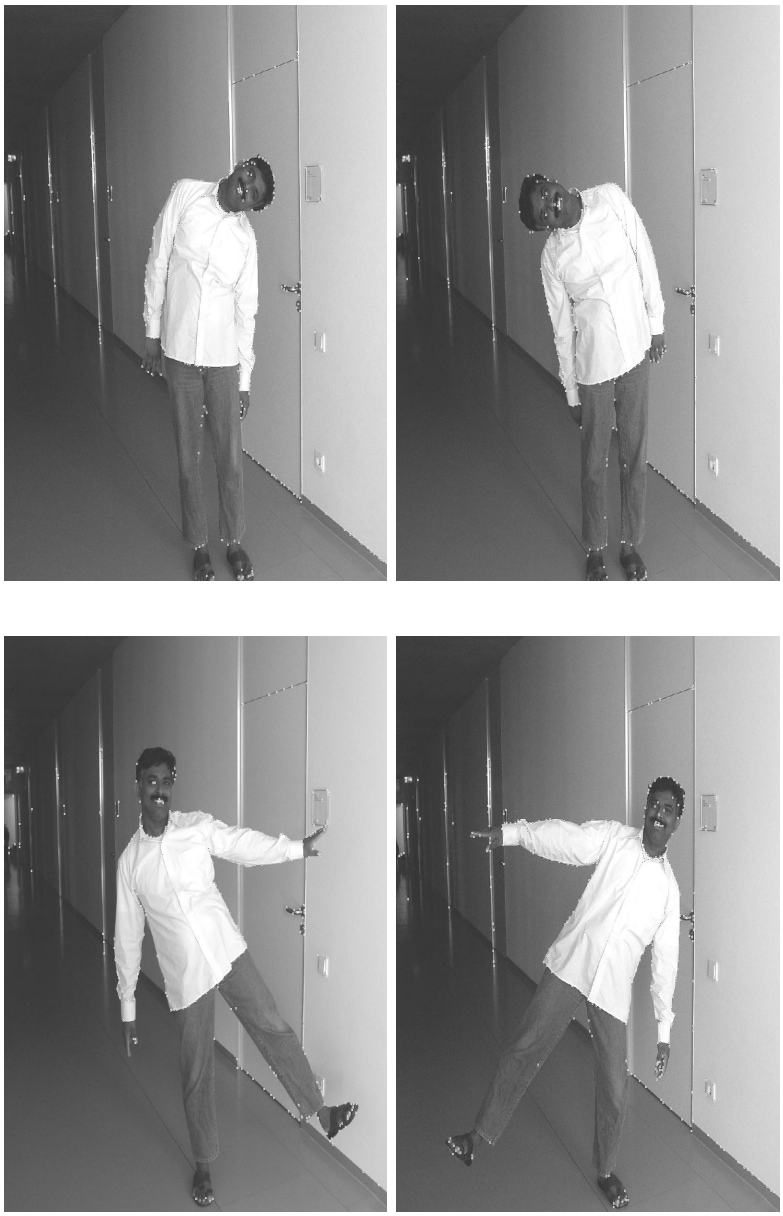


Figure 4.9: Different Postures 1p5-1p8

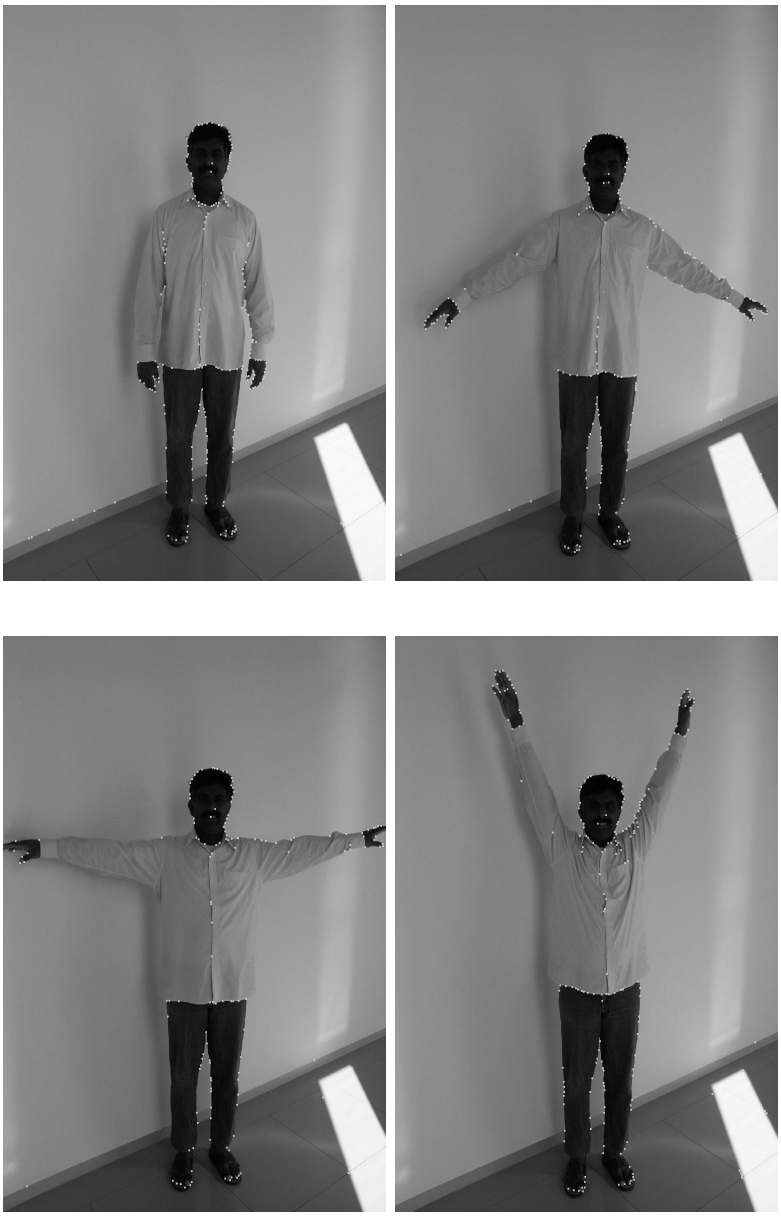


Figure 4.10: Different Postures 3p1-3p4

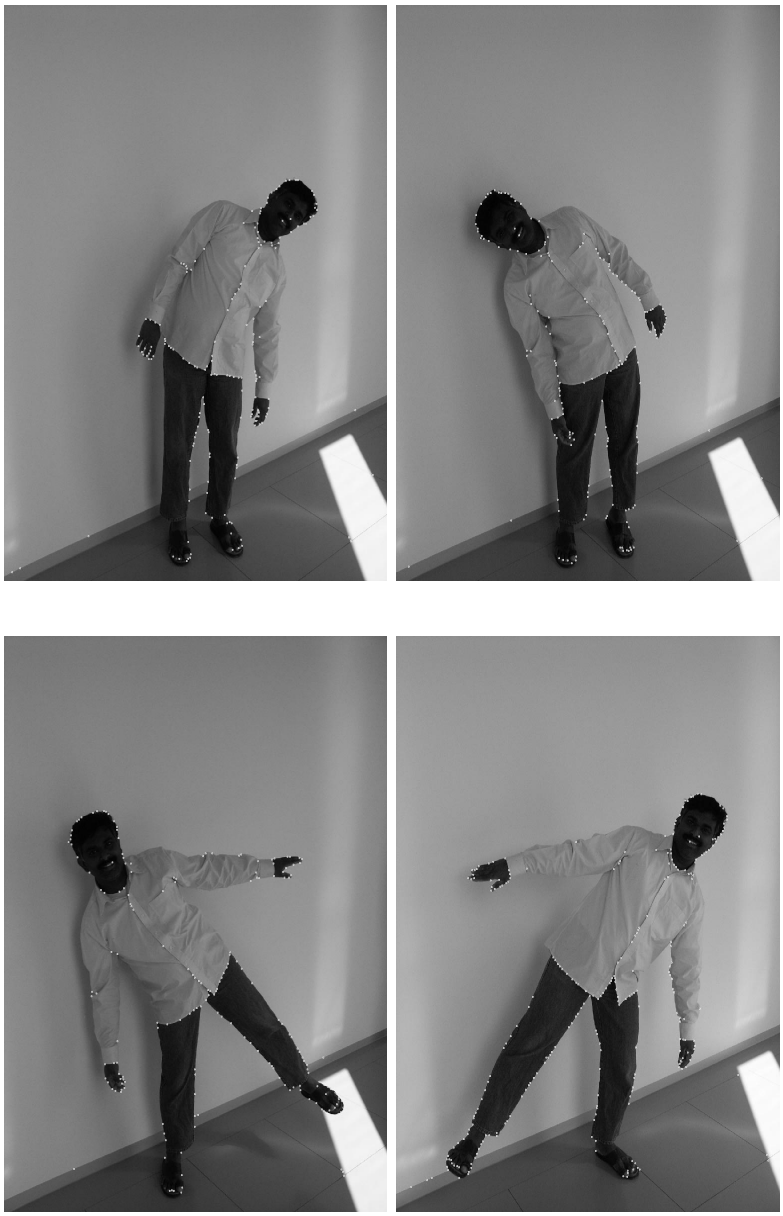


Figure 4.11: Different Postures 3p5-3p8

Posture (A)	Posture (B)	h(A,B)	h(B,A)	Time in Sec
1P1	3P1	335	378	0.06
1P2	3P2	230	175	0.08
1P3	3P3	249	144	0.05
1P4	3P4	175	155	0.12
1P5	3P5	174	162	0.03
1P6	3P6	165	179	0.03
1P7	3P7	190	199	0.07
1P8	3P8	215	196	0.08

Table 4.9: Hausdorff Distance h(A,B) for similar ones in Set1 and Set2

Posture	3P2	s10	s20	l10	l20
3P2	0	107	221	90	124
s10	186	0	159	119	140
s20	133	87	0	149	176
l10	177	198	278	0	116
l20	236	284	355	170	0

Table 4.10: Hausdorff Distance h(A,B) for Different Sizes

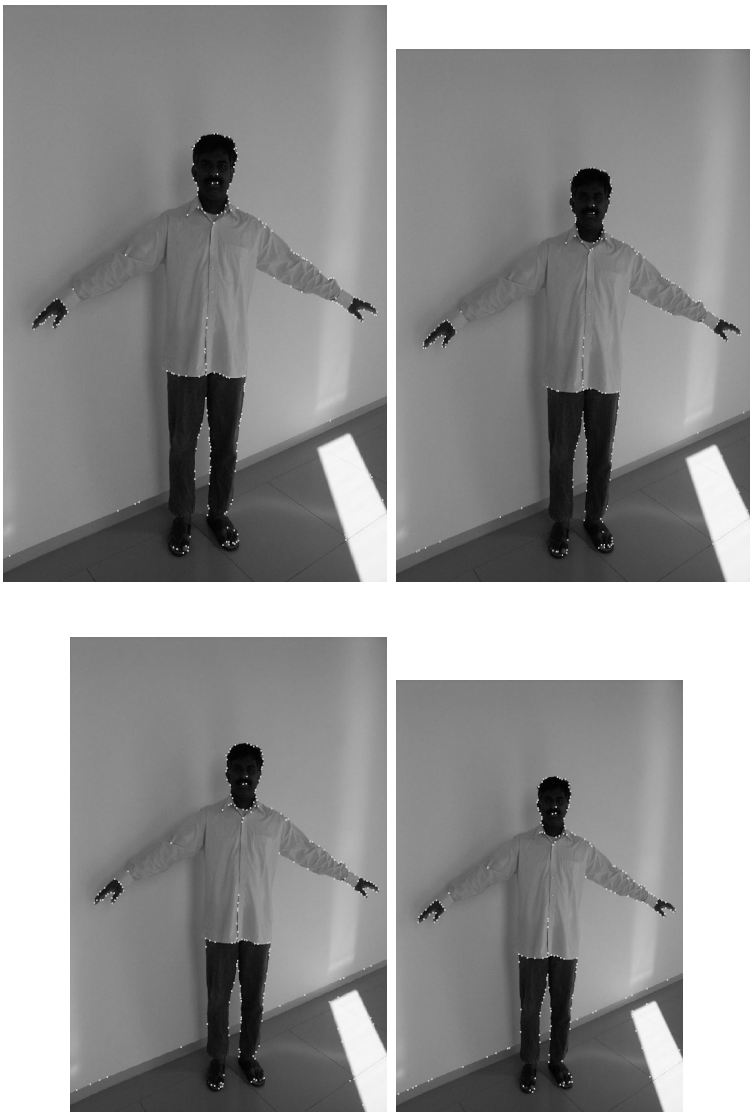


Figure 4.12: Different Postures of Different Sizes

Figure 4.13: Avathar Manifestations

- Even for slight changes, h(A,B) varies.
- For different sizes of the same image, H(image,differentsize-image) are not necessarily to be proportional.
- As the environmental light intensity affects the image, feature extraction is affected, resulting in different Hausdorff distance due to presence or absence of some corners.

4.4.6 Parallel Algorithm for Image Matching

Parallelization

For this particular problem, there can be at least three methods to parallelize. One way is to take each image by a processor and match with all models. The other way is to take one by one all images by all processors and divide the set of models equally among the processors. The third way is to divide each time one image by the number of processors and match that portion of image with all models. In the last model, overlapping is essential to get the proper solution. The second model is preferable in the situation while tracking a person is the major factor. The first and second methods are implemented here. A good insight to various strategies of parallelization can be found in [79], [80] and [81].

Outline of Parallel Program for First Method

Let r be the number of images, q be the number of models and p be the number of processors.

```
MPI_Comm_rank(MPI_COMM_WORLD,&i);
MPI_Comm_size(MPI_COMM_WORLD,&p);
each processor handles one image, image = i in parallel do
{ fscanf(...);
  for each model k = 1 .. q do
    {      fscanf(...);  /*each processor i reads all models k one after the other */
      h = Hausdorff distance of the image i matched with model k for all positions.
      whenever the h(k,i) < threshold, the position is notified.
      /* if required the best matching model is also found depending upon
      the minimum threshold. */
    }
}
```

Image	Model	Posi	Accu	Found	2 pcr Time	4 pcr	8 pcr
1	1	0,0	100%	Yes	0.28	0.27	0.27
2	2	160,350	100%	Yes	2.68	1.48	1.07
3	3	390, 200	100%	yes	5.20	2.94	2.07
4	4	127,235	100%	Yes	5.6	2.97	2.08
5	5	330,200	100%	Yes	2.68	1.8	1.48
6	6	240,120	100%	Yes	13.17	6.68	3.85

Table 4.11: Parallel Time of Hausdorff without Distance Transform

Image	Model	Posi	Accu	Found	2 pcr Time	4 pcr	8 pcr
1	1	0,0	100%	Yes	0.51	0.52	0.53
2	2	160,350	100%	Yes	1.52	1.42	1.39
3	3	390, 200	100%	yes	2.57	2.36	2.26
4	4	127,235	100%	Yes	2.64	2.4	2.29
5	5	330,200	100%	Yes	2.4	2.26	2.2
6	6	240,120	100%	Yes	2.47	2.29	2.23

Table 4.12: Parallel Time of Hausdorff with Distance Transform

Results and Analysis for First Method

As no communication is required unlike in the second method, the results are highly appreciable as tabulated here in Table 4.11 and Table 4.12. The unit of time is seconds. For the comparison purpose the same image and model sets are taken.

Outline of Parallel Program for Second Method

Let r be the number of images, q be the number of models and p be the number of processors.

```
MPI_Comm_rank(MPI_COMM_WORLD,&j);
MPI_Comm_size(MPI_COMM_WORLD,&p);
for each image i=1..r do
{ MPI_File_open(...,MPI_COMM_WORLD,...);
  MPI_File_read(...);  /*all processors read i^th image */
  MPI_File_close(...);
  for each model k = j, j+p, j+2*p, .. q do
    { MPI_File_open(...,MPI_COMM_SELF,...);
      MPI_File_read(...);  /*each processor j reads only the corresponding model k */
      MPI_File_close(...);
```

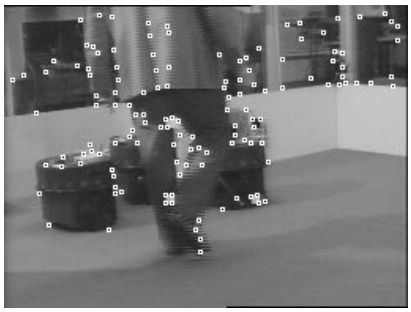


Figure 4.14: A sample Image

Figure 4.15: A sample Model

System	*Processors*	*Total time*
Cray T3E	1	273.7
Cray T3E	2	198.5
Cray T3E	4	100.5
Sparc 10	1	374.8

Table 4.13: Image Matching Parallelization

```
        h = Hausdorff distance of the image i matched with model k for all positions.
        h_partial_min = min ( h, h_partial_min);
        whenever the h(k,i) < threshold, the position is notified.
        /* if required the best matching model is also found depending upon
        the minimum threshold. */
      }
    MPI_Allreduce(h_partial_min,&h_min,1,MPI_FLOAT,MPI_MIN,MPI_COMM_WORLD );
}
```

4.4.7 Results and Analysis

As the focus is more on parallelization, here 4 sample images (one is shown in Fig.4.14) and 4 sample models are considered (one is shown in Fig.4.15). Normally in image processing application, number of models are obviously more. It may be recalled to suit to such real situation the proper strategy for parallelization is chosen for implementation. The single processor case along with Sparc Ultra 10 Sun Microsystems is also presented. The algorithm is tested with 2 and 4 processors to suit to the sample application and results are tabulated in Table 4.13 with the total time in seconds for 4 images.

In all the images, the accurate positions (to one pixel resolution) of the human beings have been found. The best model (or models) for each image has been found correctly without fail. From the overall timing (given in seconds), it is evident that parallelization produced good results without any iota of doubt. This also paves way for improving the results with more processor for increased number of model to suit to real application.

Computation and Communication Complexities

As the best model among the models suited to image is required in particular cases, the communication is required among the processors. As the method is implemented, as mentioned above the communication is essential to compute the minimum and to keep the minimum to compare across the iterations. Here, the communication is kept at the minimum as compared to the computation. The communication complexity per image is *O(sy)* where y is the cost involved in finding the minimum and broadcasting the value among the processors and $s = q/p$.

Let the size of an image be v and the size of a model be u. Totally for each image, s communications are required to find and broadcast the minimum for h(model,image) among the models. The computation per processor per image is *svu*. So, the complexity is *O(suv)* where the serial complexity is *O(quv)*. For r images, both these complexities are multiplied by r. It may be noted, in the method of parallelization only the given complexities are true as the other methods vary substantially.

4.4.8 General Scheduling Aspects for Optimal Solutions in Computer Vision

Apart from computing the distances efficiently, the computation time can be further reduced in case of multiprocessing systems of type MIMD(Multiple Instruction Multiple Data). Here, how the general scheduling can improve the reduction in computation time for computer vision related tasks is discussed. The scheduling of tasks onto processors is an important step in exploiting the capabilities of a multiprocessor system. The multiprocessor scheduling problem can be stated as finding a schedule for a task graph (that represents a parallel program) to be executed on a multiprocessor system so that the completion (or execution) time of the graph (program) can be minimized. The motivation for the objective of minimal completion time is that, in many cases, a poor schedule results in excessive interprocessor communication and inefficient processor utilization. Since the problem is known

to be NP-hard in the strong sense [116], in all but a few restricted cases [117] [118], the main research efforts in the area are focused on heuristic methods for obtaining near-optimal solutions [119] [120] [121] [122] [123] [124] [125] in a reasonable amount of computation time.

The multiprocessor scheduling problem at compile-time is considered here. Compile-time scheduling has two benefits: (i) it is easier to realize and (ii) it eliminates run-time overheads. A new A^* based algorithm for solving the problem is presented here. To alleviate the impediments of large space and time requirements of the algorithm, three new effective techniques, namely, processor isomorphism, task isomorphism and node isomorphism have been developed and the concepts of upper bound and lower bound theory are used effectively.

The algorithm developed here, unlike the heuristic approaches, always produces optimal schedules (or solutions). There are two main reasons for finding optimal solutions. Firstly, when a particular problem, say a computer vision task, has to be repeated virtually several times, it is more efficient if an optimal solution is used as one can afford to find time to get the optimal solution rather than wasting more time in each execution. Secondly, an optimal solution is a yardstick for the non-optimal solutions for measuring their closeness to the optimality. In case one may have to choose among the available non-optimal solutions based on their relative closeness, and the computational time and the memory space required to arrive at the corresponding non-optimal solution, the yardstick provides a better perspective of the choice of non-optimal solutions. Moreover, the decision of improving the heuristics can be better evaluated only when an optimal solution is at hand.

The main difference between general Data Flow Graphs (DFGs) and the signal processing DFGs is the associated delay elements (registers) in the directed edges [82]. An edge without a register represents precedence between tasks within iteration. If an edge has n registers, it describes the precedence between tasks of different (i,n+i) iterations which differ by n iterations. A simple example of a nonterminating, iterative, data-flow program with feedback is given in program 1. The DFG in Fig.4.16 corresponds to program 1. It may be recalled in image matching using different strategies each iterating independently, to avoid those areas where a model has been already matched by one strategy, there need to be some communication during the iterations. It must be also understood that since finding the match with some model occurs at runtime, it can not be directly modelled by the method. However, iterations can wait after specified number of iterations and then the communications can be made which is very easy to model using the delay register

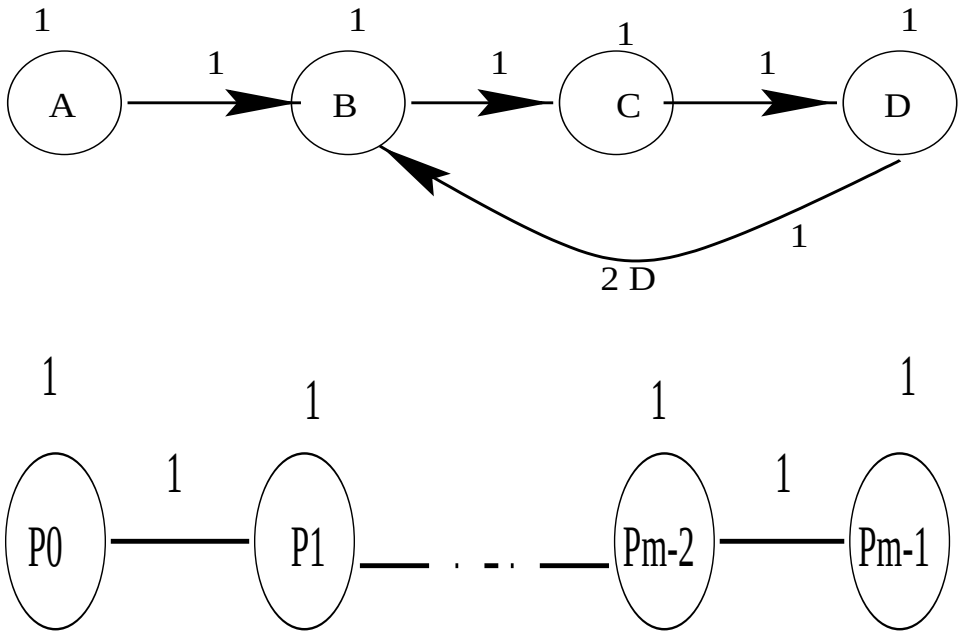


Figure 4.16: Data-Flow Graph with Linearly Connected Multiprocessor

approach. In the appendix the general case is simulated as the accessibility to such machine is restricted.

A complete discussion with appropriate examples for explaining the data-flow model is available in [82].

Program 1: Initial conditions : db(-1),db(0)

for (i=1 to $infinity$)
ab(i) = f_{ab} [x(i)]
bc(i) = f_{bc} [ab(i) db(i-2)]
cd(i) = f_{cd} [bc(i)]
db(i) = f_{db} [cd(i)]
y(i) = f_y [cd(i)]

Issue	*Hausdorff*	*Chamfering*
Features in Image	Points/Corners	Edges
Features in Model	Points/Corners	Points or Corners
Distance Measure	Maximum difference	Root mean square
Distance (D)	Euclidean or others	Distance function
Distance Calculation	Direct or DT or Look up	DT

Table 4.14: Comparison between Hausdorff and Chamfering

4.5 Comparison of Hausdorff and Chamfering Methods

To start with the principal difference as given in Table 4.14, chamfering methods use edges as features and Hausdorff method uses corners. The distance measure in case of chamfering is the square root of the average of the sum of the squared difference of the distances between the model and the image at a particular position, where it is the maximum of the differences of the distances in case of Hausdorff method. Distance transformation is essential for Chamfering. Lookup or distance transformation can be used for Hausdorff method.

Chapter 5

Human Being Recognition

As mentioned earlier, recognizing a human being has become a most interesting field not only due to the applications, but also for the challenging complexity. From Surveillance to Driver assistance in cars, recognizing a human being is almost a compelling demand. At first, the problems with single human being are discussed and then the problems with group of human beings are discussed.

5.1 Fundamental Complexities involved in HBR

5.1.1 Description of Human Being

In any recognition system, the object (model) to be recognized in an environment (image) must be described or defined. In the present case, human being must be defined or described in the best sense. A head, a torso, two hands and two legs form a human being without going into internal details such as nose and eyes as in the case of face recognition. It may be recalled the biometric measures based on iris pattern or finger print are required when a particular person has to be identified. Thus depending upon the need of the application, the level of details of description varies considerably. For e.g., in case of driver assistant system it is sufficient that a pedestrian (human being) is recognized. However in surveillance mere detection of a human being in the protected area can ignite further course of actions and more particulars about the human being may be required later.

5.1.2 Size of the Human Being

In most of the systems, size of the human being is fixed or approximated to an interval i.e., a human recognition system may not recognize a walking child due to the size restrictions. Moreover based on the size of the human being (which is fixed), the systems are able to identify the possible regions of head or torso or hand or leg. Eventhough recognizing the human being in such systems is hard, even more when there is no restriction on the sizes as in the study. However, proper modelling including for a child may possibly help in recognition. Yet removing the size restriction will force the system to check all regions in the image incurring more computation time. Nonetheless, it is worth trying.

5.1.3 Segmenting the Region of Interest(Human Being)

Just a change in the region by background subtraction suffice some systems to conclude the presence of human being. Further analyzing only such regions to precisely recognize the human being is also found in the literature. Interestingly the colour components still play an important role in segmenting the regions of interest (Human being). Such a systems will not work where nonwhite people are present. Transformation such a wavelet or Fourier can also be used to segment the region of interest (human being).

The background subtraction as such will not work flawlessly in outdoor environments. It is unreasonable to think that only one person alone will drive a car in road. Systems which overcome small changes due to leaves waving use a larger area or predefined area size to choose only those regions for further processing (recognizing as human being).

Without background subtractions also recognition can be made at the cost of computation time. Methods such as Hausdorff can be used to recognize in such cases. However, there are other problems like occlusion and self-occlusion as other objects can occlude or due to the viewing angle also occlusion is possible.

5.1.4 Occlusion

Occlusion is one of major problems in recognition systems as some of the critical or main features may be occluded resulting in incorrect recognition. In such situations, there must be a definition of minimal features required to identify a human being such as head alone is sufficient (for sake of arguments). Then if the person is not looking at the camera, a possible way is

to look for a circular or elliptical object approximately. This constraints the system very much due to viewing angle.

Furthermore, due to viewing angle self-occlusion is also possible. If a person stands erect with hands kept close to torso and legs one behind the other and if the viewing direction is perpendicular to the facing direction, possibly head and a leg may be recognizable. It is well known that depending upon the pose, features can be occluded.

Yet the methods such as Hausdorff provide ample opportunity to fix some parameters approximately and still recognize despite occlusion. In such cases, false positives can not be avoided. Interestingly the occlusion itself can be modelled in Hausdorff method very easily, of course increasing the number of models many times.

5.1.5 Sufficient Models

For sure with more models, more instances of the objects can always be recognized. But the prime question is how many models are sufficient to positively recognize every instance of the object. This is likely to be extremely large considering the degree on freedom movements of various parts of the human being. Many of systems which attempt to recognize the motion or action of the human being is highly restricted to the given models or predefined models. While forming a model, obviously the size of the model comes into play. It is very difficult to accept that an adult (human being) is a scaled version of a child (human being). On the streets, people with different ages can walk together. Moreover, bending phenomenon is observed with the very elderly people (leave alone the use of sticks).

Confirmly, one must use different scaling factors or different models for each scaling factor (in case of floating point what is next probable useful number). In the first case, as it only reduces the number of models, there is every possibility that different scaling factors must be considered which in turn will enormously increase the computation time. In the second case, which model can be preferred over the others becomes crucial as sequentially trying with each model will take unacceptable amount of time depending upon the application such as driver assistance. Succinctly matching and indexing of the models take predominant role. Some of the hand drawn models of the human beings are shown in Fig.5.1 and Fig.5.2.

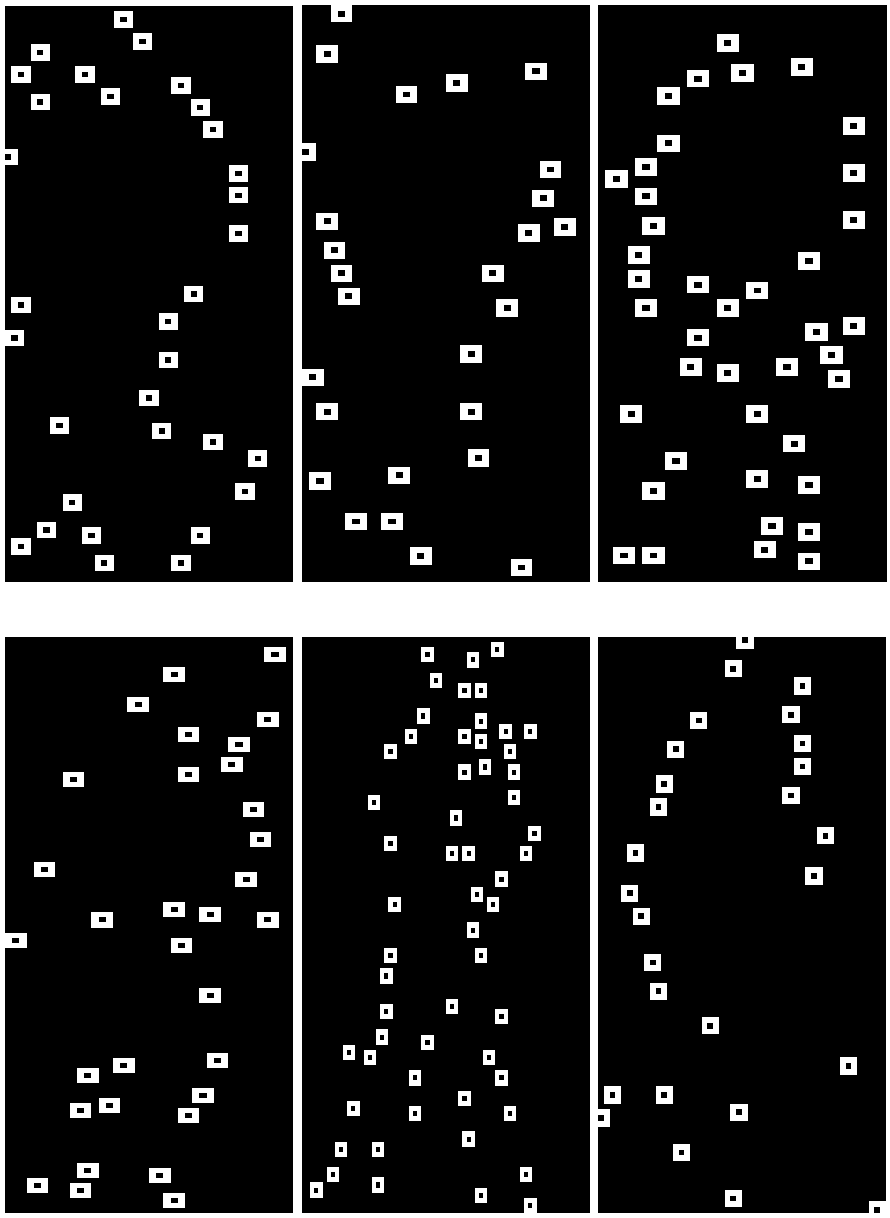


Figure 5.1: Some examples of hand drawn Models of the Human Beings - Set 1

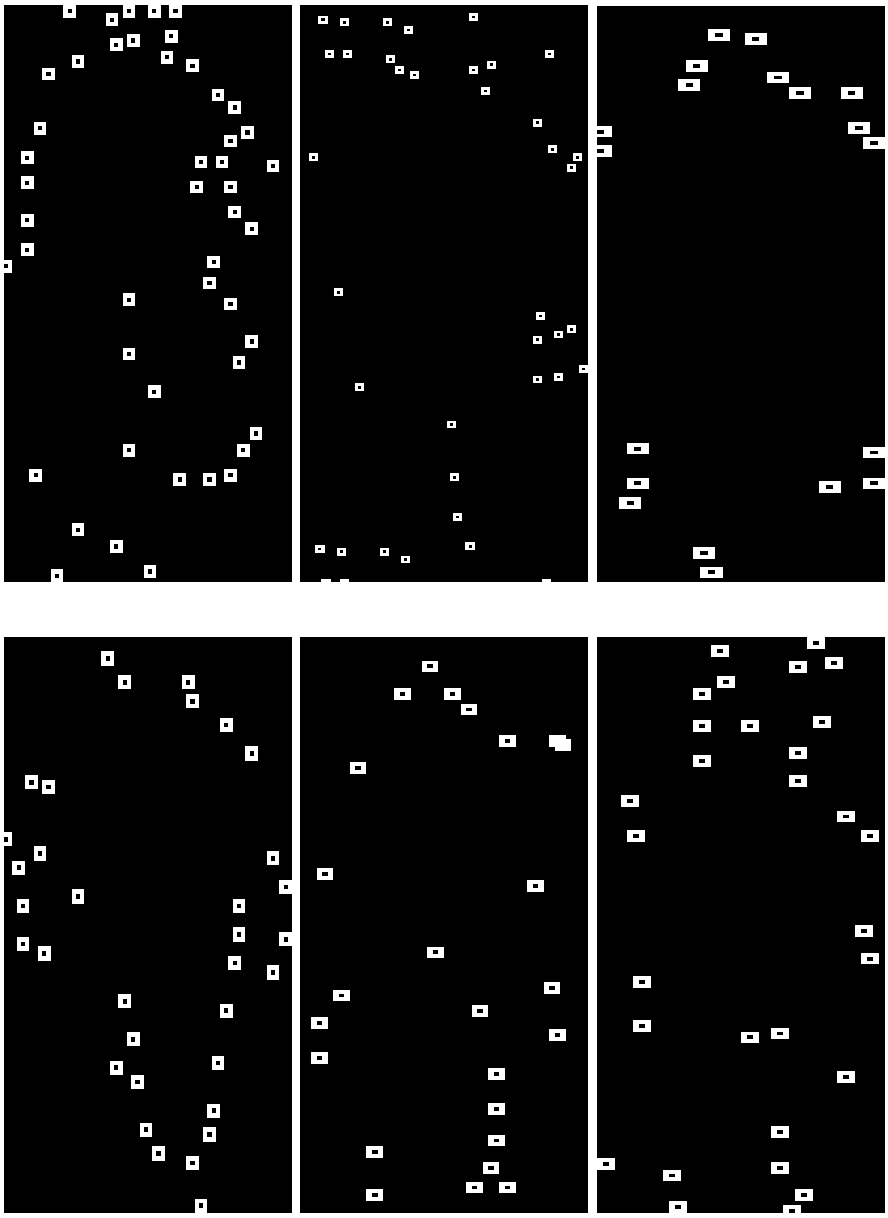


Figure 5.2: Some examples of hand drawn Models of the Human Beings - Set 2

5.1.6 Threshold Value

Matching is an integral part of complex recognition systems. To recognize an object (model) in an environment (image), the respective features have to be matched. This involves a matching measure. Unless it is a perfect match, there is always some deviation due to occlusion and such other issues. So, a threshold value is required to decide categorically whether the portion of the image (region of interest) match with the model (human being).

To set a value for the threshold itself is a problem. If the value is set to be large, the possibilities of false positives will be more, i.e., the objects will do not necessarily match, may be matched. If the value is very less, only it accepts exactly matching instances which are rather infrequent in real scenarios due to noises. A through discussion on Models, Matching and Indexing is presented in the following chapter.

5.2 Recognition of Group of Human Beings

On the streets or inside the shops, there are mostly groups of human beings. Recognizing a group of human beings is more difficult than a single human being. The main aspect is the motions involved which results in occlusion, difficulty in identification or in arbitrary movements.

5.2.1 Problem due to Multiple Occurrences

Eventhough the definition of the human being as defined in the previous section is acceptable, the new situation is encountered with m legs, n hands, x heads and so on. It is comparatively difficult to find the exact number of persons in the scene or image. The principle reason is occlusion. With odd number of hands and legs (leave alone those people with one leg or amputated limbs), considering which combination belongs to one person is only a matter of probability.

5.2.2 Problem due to Movements

Independently joint movements cause lot of occlusion. When two people walk together parallel to the focal plane of the camera, there is a finite chance of one person occluded entirely depending upon the sizes of the persons. Similarly when two persons meet and disperse in different directions, to identify which person moved in which direction is relatively complex. If sequence of persons move in the same directions with same velocity like

in military parades and by chance the sampling frequency matches with respective time interval, possibility of detecting the very motion may be questionable. Or in an industry, when the parts to be picked or checked arrive in regular intervals, with the camera above the conveyor belt may not able to say whether there was a movement of the conveyor belt or not without special marking or such other identification markers.

5.2.3 Problem in Segmentation

Segmenting a region of interest (human being) based on colour will not work until unless all possible colours of the human beings are tried as each human being can be in different colours.

5.3 Results and Analysis

Principally, three major cases are presented for discussion. One sequence (Sequence 1) with almost plain background Fig.5.3, second sequence (Sequence 2) with one human being but in a cluttered environment Fig.5.4 and third sequence (Sequence 3) Fig.5.5 with more human beings in a cluttered environment are detailed here. The analysis is focussed on the computation time and accuracy with all the four methods viz chamfering, Hausdorff direct method, Hausdorff distance transform method and Hausdorff with lookup method. The results of parallel versions of the same are also presented. A prototype to demonstrate the possibility of ontological description is also included. The resultant figures are shown in Fig.5.6, Fig.5.7 and Fig.5.8. The ontological descriptions produced by the program are reproduced here.

5.3.1 Ontological Description for Sequence 1

Person 0 possibly moving right from frames 0 to 1
Person 0 possibly moving right from frames 1 to 2
Person 0 possibly moving right from frames 2 to 3
Person 1 possibly moving left from frames 0 to 1
Person 1 possibly moving left from frames 1 to 2
Person 1 possibly moving left from frames 2 to 3
At frame 0 person 0 is left of person 1
At frame 1 person 0 is left of person 1
At frame 2 person 0 is right of person 1
Person 0 and person 1 crossed between frames 2 1
At frame 3 person 0 is right of person 1

Figure 5.3: Sequence 1 Frame Number 1,2,3 and 4

Figure 5.4: Sequence 2 Frame Number 1,2 and 3

Figure 5.5: Sequence 3 Frame Number 1 and 2

Figure 5.6: Human beings in Sequence 1 frame Number 1,2,3 and 4

Figure 5.7: Human beings in Sequence 2 frame Number 1,2 and 3

Figure 5.8: Human beings in Sequence 3 frame Number 1 and 2

Set No	*Chamfering*	*HD direct*	*HD with DT*	*HD with Lookup*
Set1	10.37	53.13	40.43	45.17
Set2	10.24	208.18	20.98	186.18
Set3	121.04	14.23	135.20	4.87

Table 5.1: Computation Times of Different Algorithms

Set No	*Chamfering*	*HD direct*	*HD with DT*	*HD with Lookup*
Set1	3.97	3.84	3.93	9.76
Set2	5.2	19.36	4.78	25.99
Set3	3.39	19.37	5.86	30.77

Table 5.2: Parallel Computation Times of Different Algorithms

5.3.2 Ontological Description for Sequence 2

One person at 365 195 in frame 0 is NOT identified
One person at 350 190 in frame 1 is NOT identified
Person 0 possibly is stationary from frames 0 to 1
Person 0 possibly is stationary from frames 1 to 2

5.3.3 Ontological Description for Sequence 3

Person 0 possibly is stationary from frames 0 to 1
Person 1 possibly moving left from frames 0 to 1
At frame 0 person 0 is left of person 1
At frame 1 person 0 is left of person 1

The Table 5.1 shows the computation times involved in sequential case with different algorithms. It must be recalled that only the Hausdorff direct method performed well and correctly in all the cases. The reduction in other algorithms are due to their inability to recognize all the instances. Thus they resulted in comparably lesser time. In case of parallel implementation, the same is true as algorithmically there is no difference as it is only parallelized. The undue increase in Lookup method is because of the large file reading resulting in heavy I/O which offsets all the gains due to parallelization as shown in Table 5.2.

5.4 Fusion Architecture to recognize Human Beings

5.4.1 Basic Concept

From the experience with Chamfering and Hausdorff methods and their implementations, it is clear that one algorithm is not sufficient. Moreover to handle occlusions and size differences, different approaches are required. So, it is necessary to combine the required methods into one architecture, which is named as Fusion Architecture.

Hausdorff Method for Occlusion

It is natural that some of the parts might be occluded in the image. So, trying to apply Hausdorff as such for the model may give different results depending upon the occluded portions. In some cases, few corners may be absent or so. In order to handle such occlusions, instead of calculating distances between the points as in original Hausdorff method, in the new approach to handle occlusions, one finds the percentage of points which differ from the threshold predefined. If the percentage is less than a particular level (to be fixed or the maximum permissible level of occlusion of the model), then it is considered that the model is present at the position in the image.

It is very easy to verify that the modification of Hausdorff distance for occlusion is the same as the original Hausdorff method if the percentage of points that is more than the threshold is zero. However, such strict permissible levels of percentage will not match well with occlusion in the images. Of course, one can verify the implementation of the modified method setting the level of percentage as zero to check up with the original Hausdorff method. The results of a parallel implementation of the method for a sample image and model are presented in Fig.5.9.

Eventhough the scaling of the parallelism is evident, the idle time due to synchronous communications shares around 47 percentage of the total computation time. The less Input-Output time shows the efficient use of parallel Input-Output mechanism implemented on CRAY T3E.

Hausdorff Method for Different Sizes

One of the classical problems in matching is the problem with zooming or different sizes. Here, scaling is considered to be equal on all dimensions unless otherwise stated. For sure, if the scaling is so less that the overall effect is less than the threshold, then it will work correctly. However, if the overall

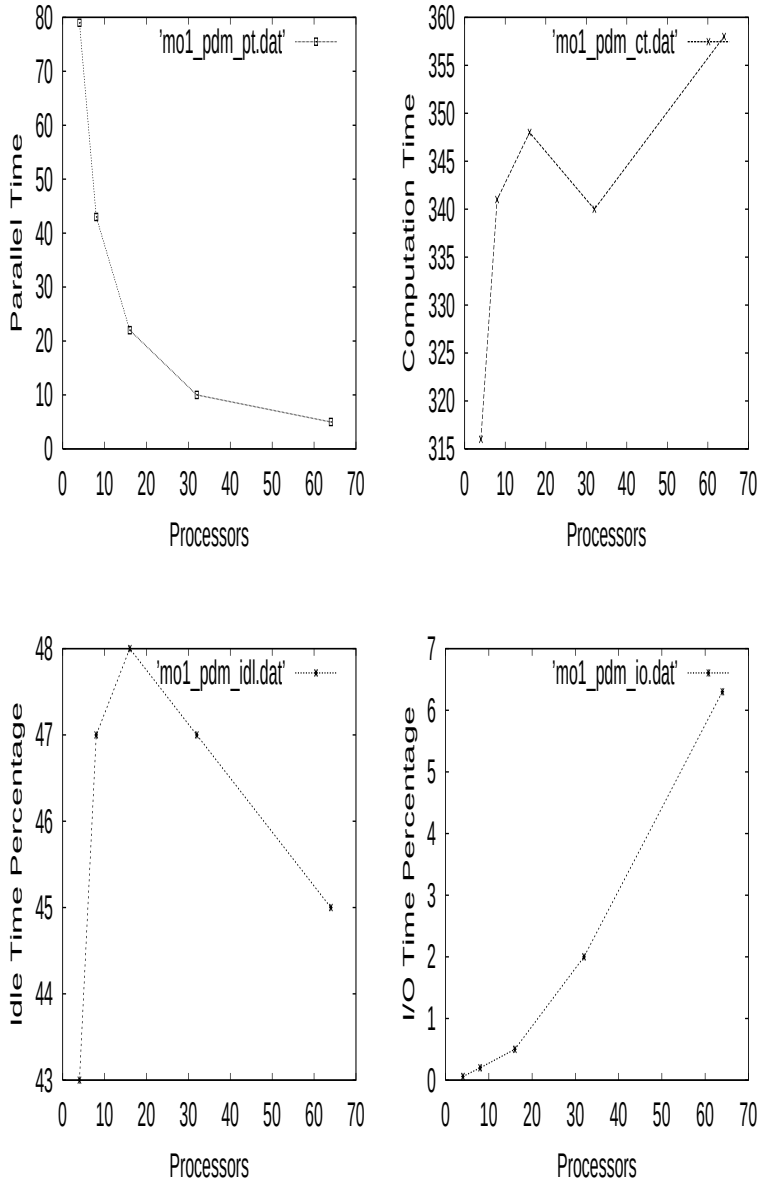


Figure 5.9: Parallel Implementation of Hausdorff for Occlusion

effect is more, which is a more interesting case, the modification is done such that a histogram of distances for some specified levels depending upon the scaling is constructed. Select the histogram level (distance) which has maximum number of points. If the percentage of points (maximum number of points at the highest level in the histogram versus the total number of points (or corners) in the model) is higher than some predefined (say 90 percentage), then the model is matched with the relevant portion of the image.

The key idea is that if every point in the model is displaced with some distance depending upon the zooming or scaling, then at the displayed distance in the histogram, it will be maximum. Since all the points (assuming that there is no occlusion here) are in that level, the percentage will be 100 percentage which is more than the predefined percentage (90 percentage). In such cases, the modification is same as that of the original Hausdorff method.

5.4.2 Fusion Architecture

As explained earlier, the required methods are combined together so that the recognition can be effectively done. The entire recognition of human beings is presented as the Fusion Architecture as in Fig.5.10.

5.4.3 Results and Analysis of Parallel Implementations

For various sets of image sequences, the results are obtained from the parallel implementation of the fusion architecture for recognizing human beings on CRAY T3E with MPI (Message Passing Interface). Different algorithmic approaches such as with 1 image and 1 model category, 1 image with n models category and finally m images and n models category are implemented. The sample results are presented in the Fig.5.11 and Fig.5.12. The sequences of images from Moehringen Tram station are shown in Fig.5.13, Fig.5.14 and Fig.5.15 with the results in Fig.5.16, Fig.5.17 and Fig.5.18. Further results of the sequence of images from Zebra Crossing in Koenigstrasse, Stuttgart are shown in Fig.5.19, Fig.5.20 and Fig.5.21. Yet another results of the sequence of images from City Centre, Stuttgart are shown in Fig.5.22, Fig.5.23, Fig.5.24, Fig.5.25 and Fig.5.26. From Table 5.3 for multiple images and multiple models, eventhough the reduction is evident, it is not scaled properly.

As the fusion architecture has ample parallelism to be exploited inherently, the parallel implementations becomes an obvious choice. From the

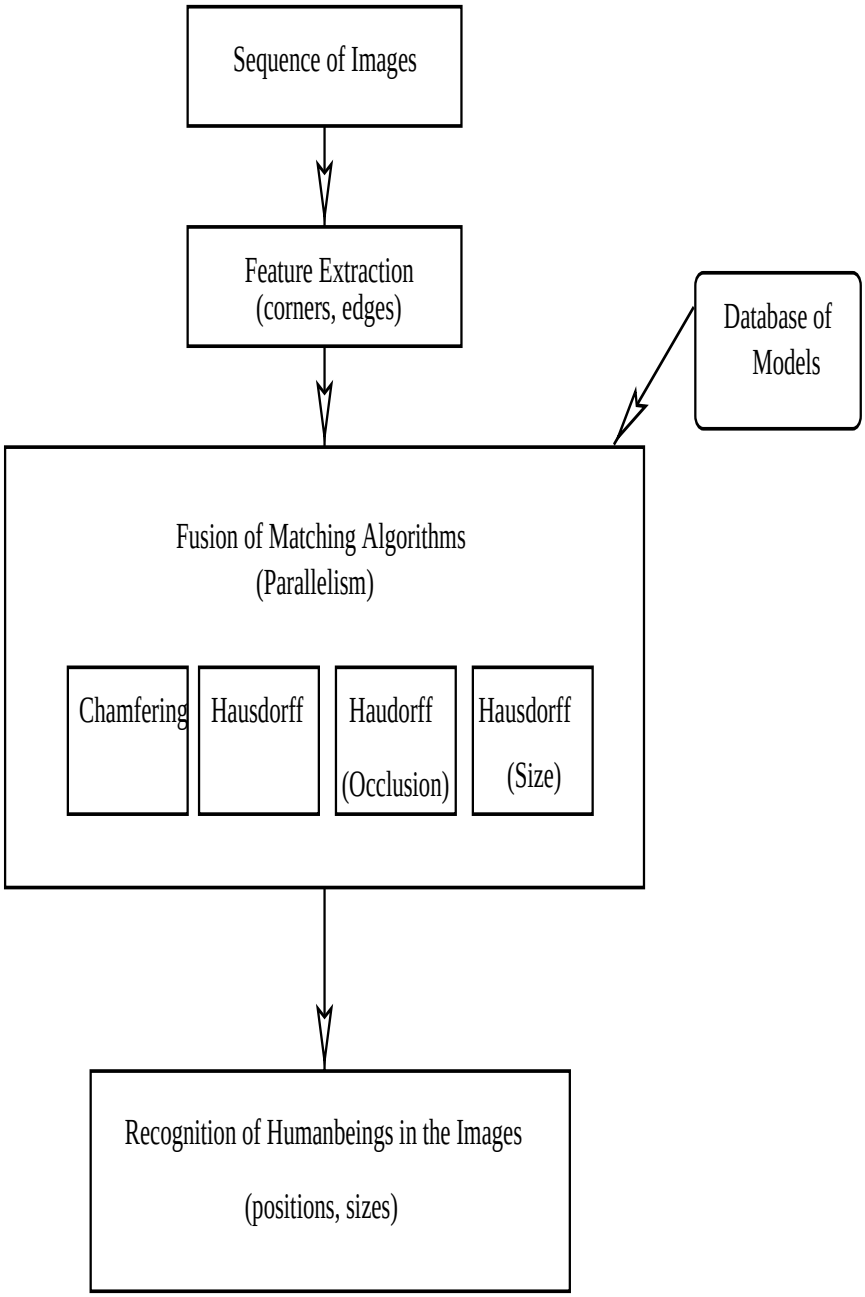


Figure 5.10: Fusion Architecture for Recognizing Human beings

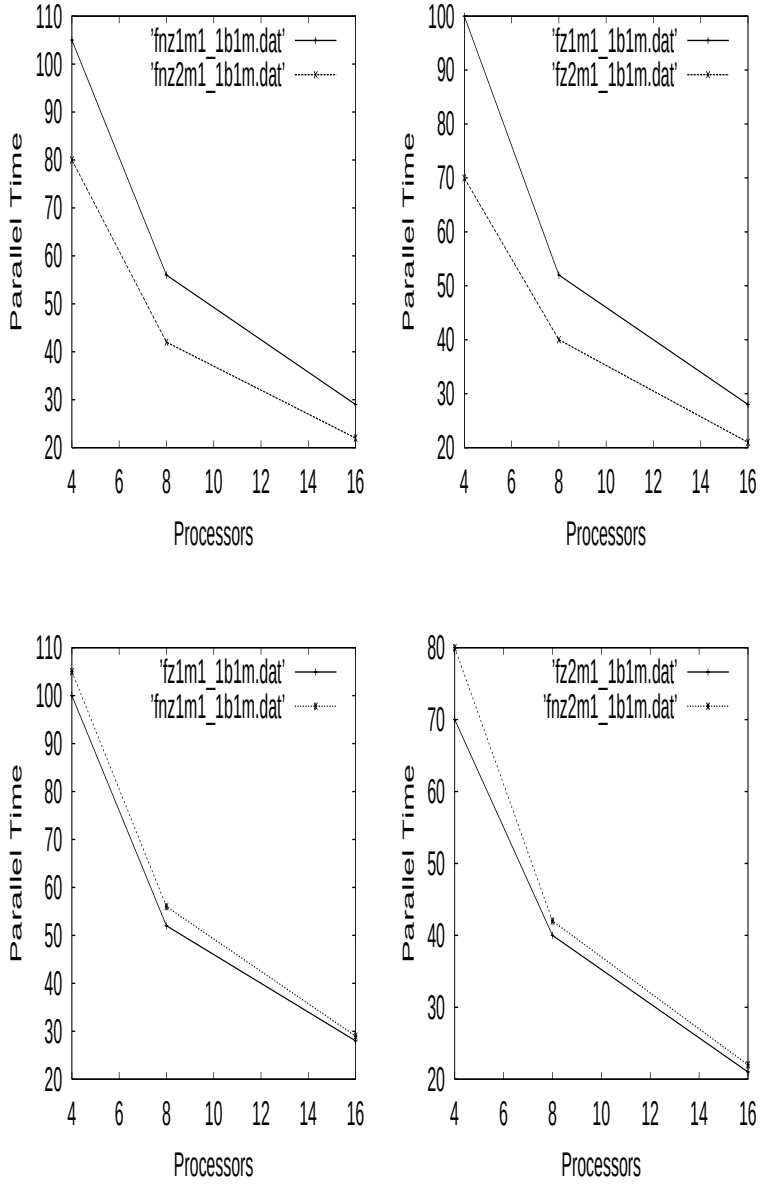


Figure 5.11: Parallel Implementation of Fusion Architecture (a) No Communication (b) With Communication (3) Difference for Set 1 (4) Difference for Set 2

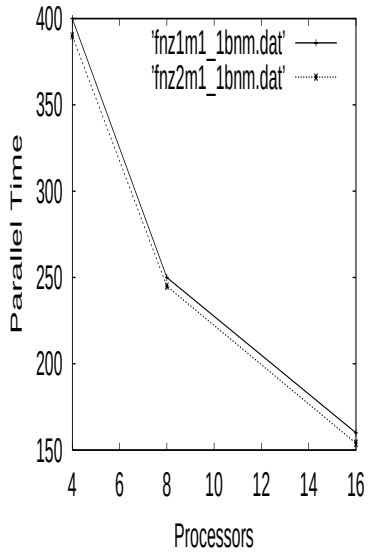


Figure 5.12: Parallel Implementation of Fusion Architecture for 1 Image with Multiple Models

Number of Processors	*Time in Sec*
4	850
16	500
48	270

Table 5.3: Parallel Implementation of Fusion Architecture for m Images with Multiple Models

Figure 5.13: Sequence of Images from Moehringen Tram station - Part1

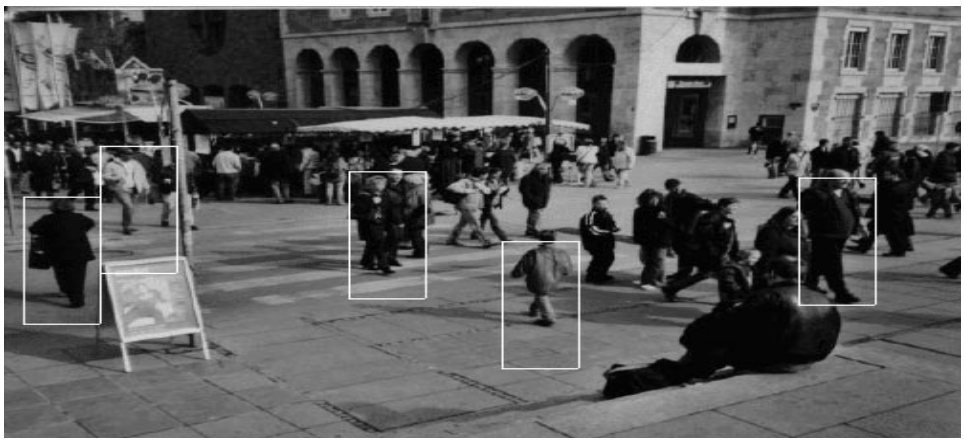
Figure 5.14: Sequence of Images from Moehringen Tram station - Part2

Figure 5.15: Sequence of Images from Moehringen Tram station - Part3

Figure 5.16: Identified Human beings in Images from Moehringen Tram station - Part1

Figure 5.17: Identified Human beings in Images from Moehringen Tram station - Part2

Figure 5.18: Identified Human beings in Images from Moehringen Tram station - Part3



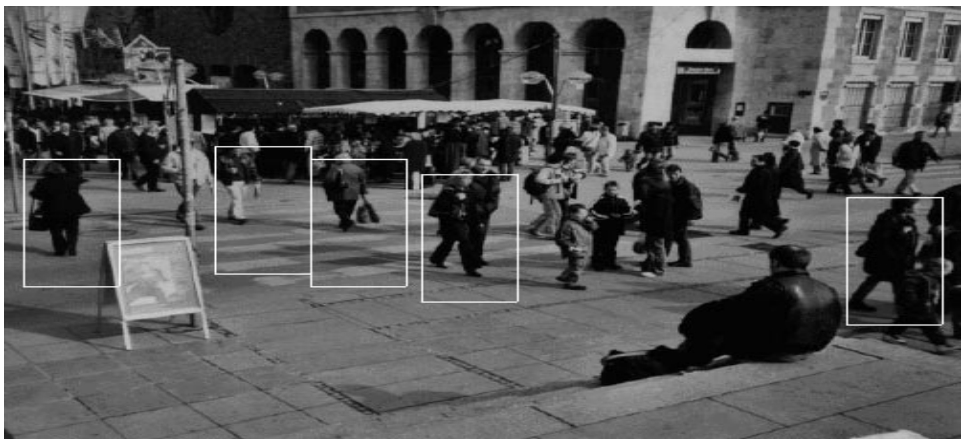
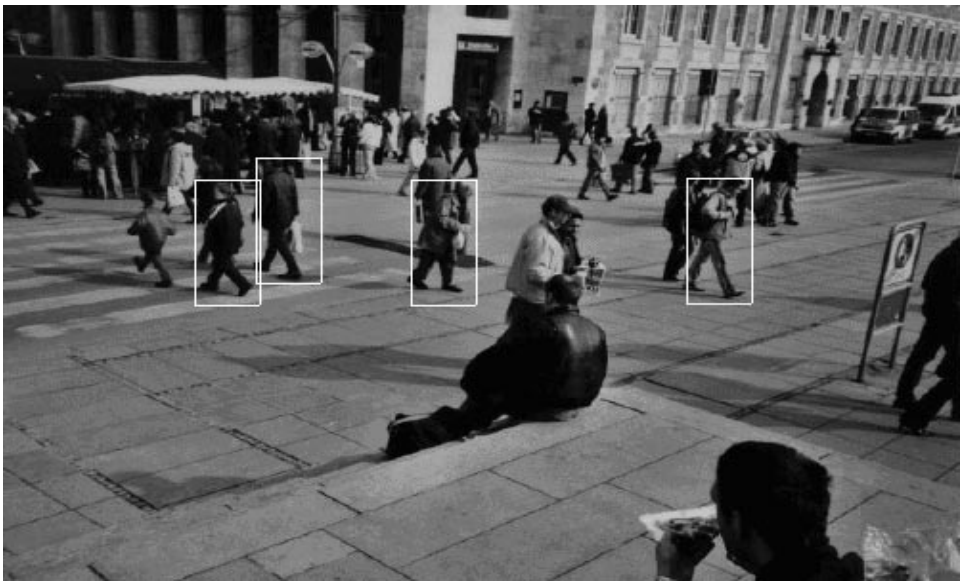


Figure 5.19: Identified Human beings in Images from Zebra Crossing in Koenigstrasse - Part 1

Figure 5.20: Identified Human beings in Images from Zebra Crossing in Koenigstrasse - Part 2



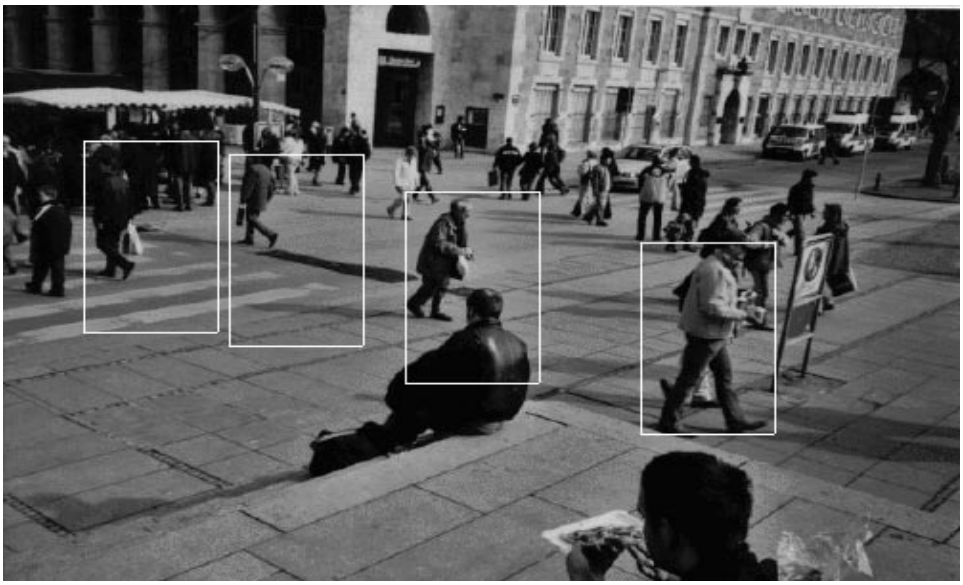
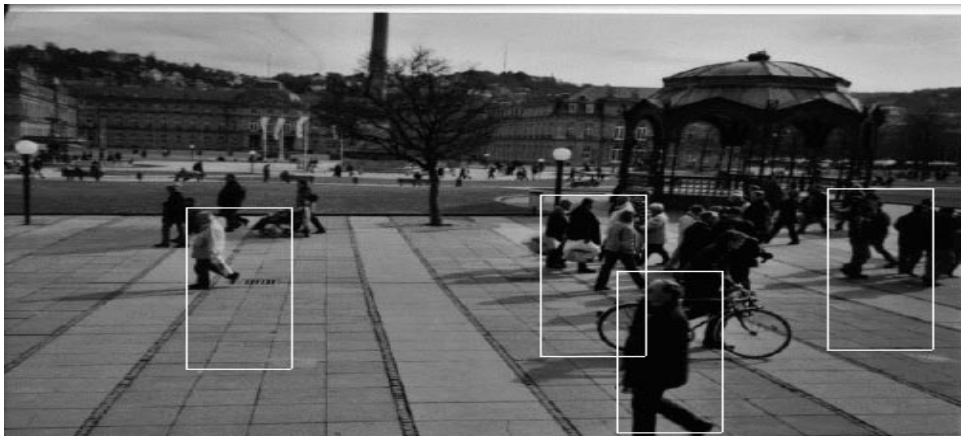


Figure 5.21: Identified Human beings in Images from Zebra Crossing in Koenigstrasse - Part 3

Figure 5.22: Identified Human beings in Images from City Centre in Stuttgart - Part 1



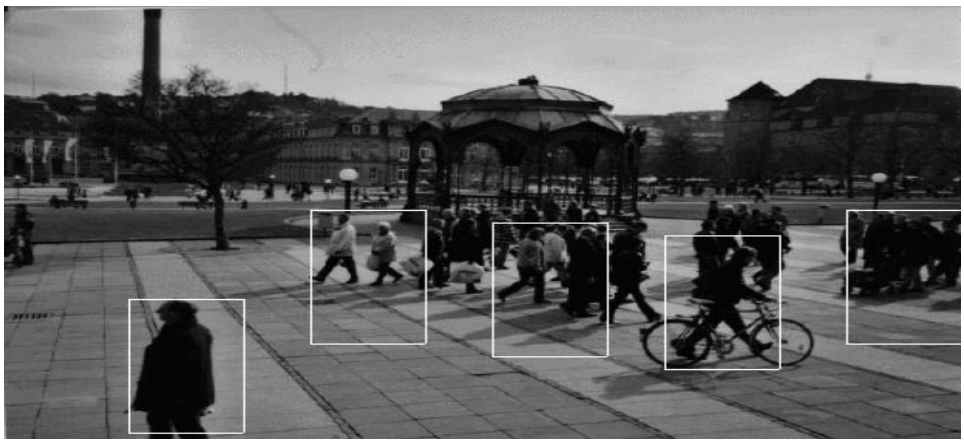


Figure 5.23: Identified Human beings in Images from City Centre in Stuttgart - Part 2

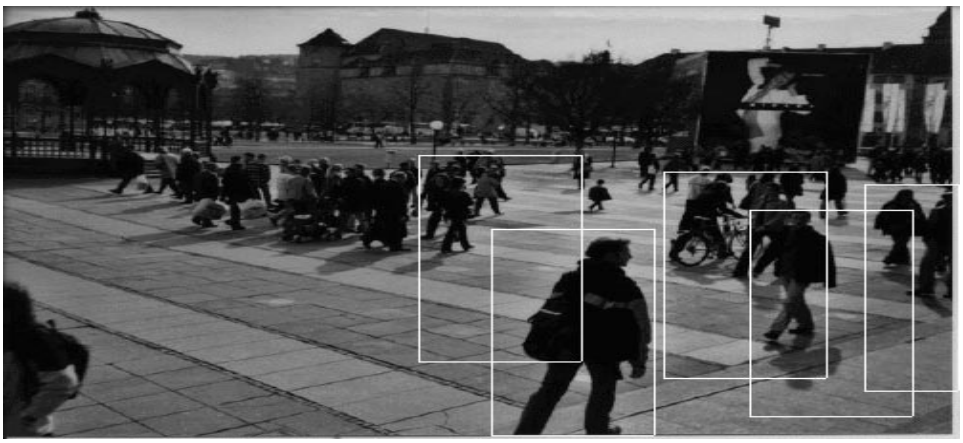


Figure 5.24: Identified Human beings in Images from City Centre in Stuttgart - Part 3

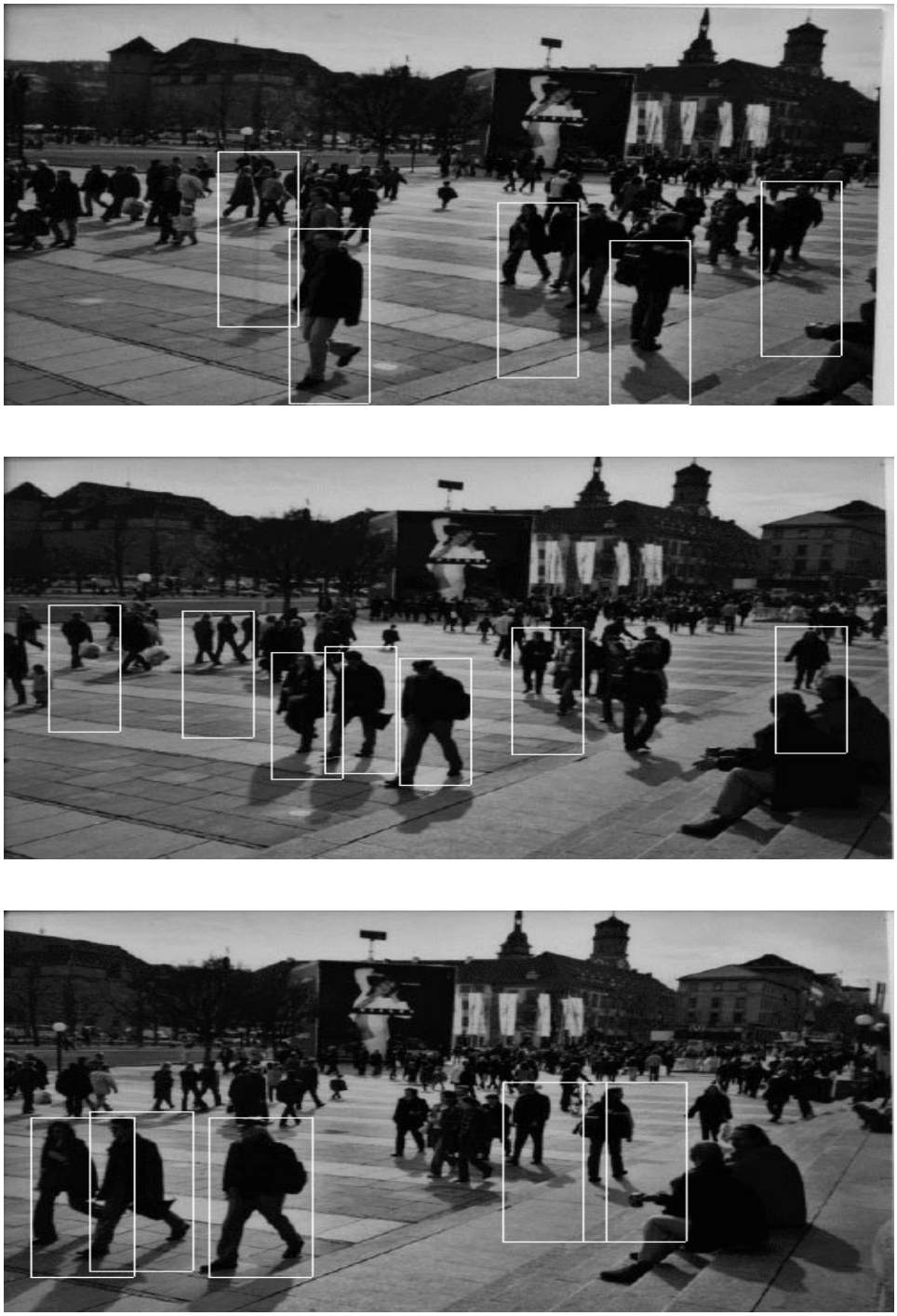


Figure 5.25: Identified Human beings in Images from City Centre in Stuttgart - Part 4

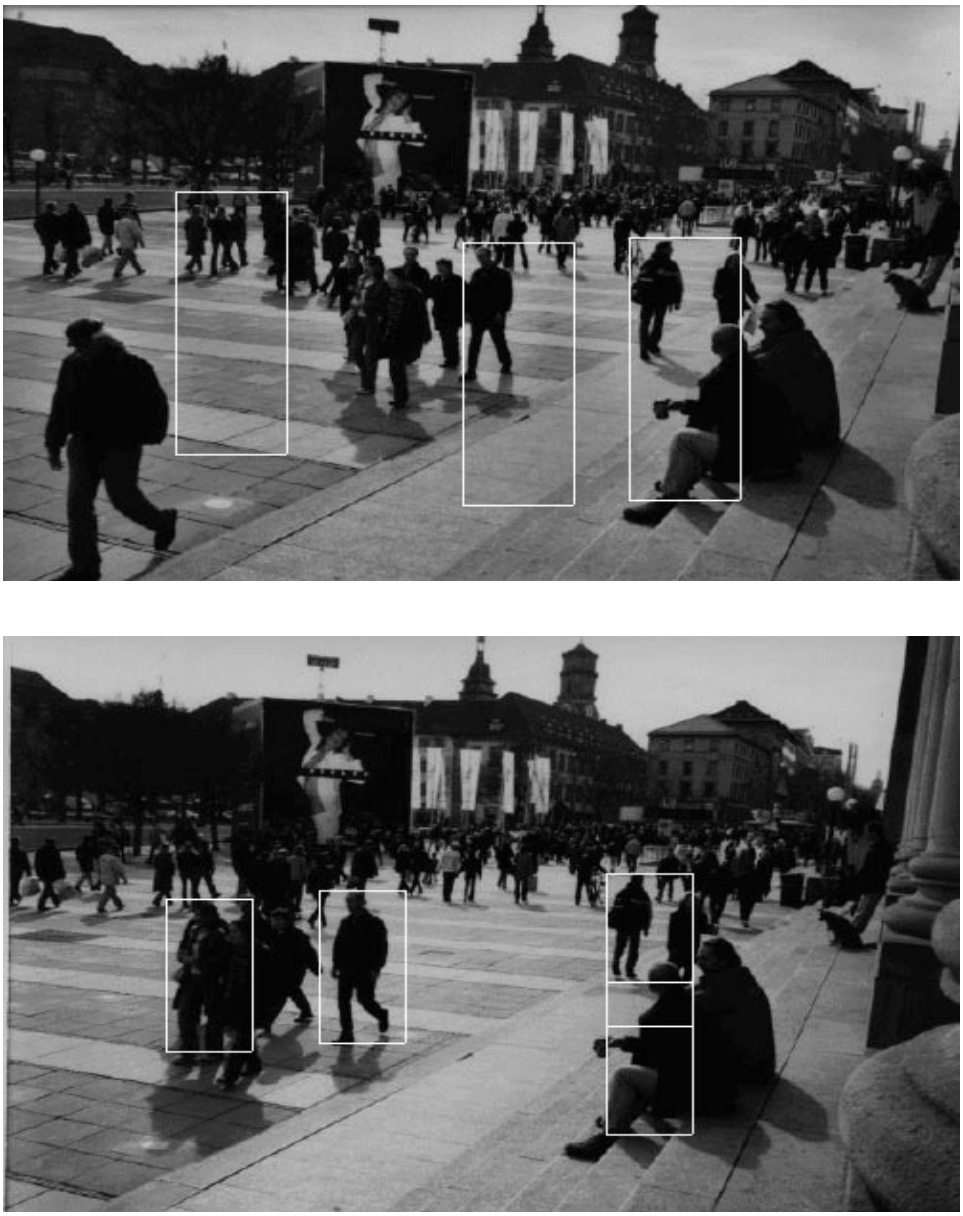


Figure 5.26: Identified Human beings in Images from City Centre in Stuttgart - Part 5

results, it is desirable to have communications among the processors. However, it may be recalled that as the implementation is on CRAY T3E, the synchronous communications demand all the processors to be synchronized before the communication can start, lot of idle time is encountered during the computation. In case to 1 image and 1 model category, it is evident that the communication improves the parallel computation time. In case of 1 image with multiple models, for 8 processors, due to communication it is more than the no-communication case as with communication it is 300 and without communication it is 250. So also with 16 processors, with communication it is 175 and without communications 160. This prohibited the use of communication method as the number of processors increase with the number of models and images.

5.4.4 Salient Advantages in Fusion Architecture

- Fusion Architecture handles cases of occlusion and size differences.
- Fusion Architecture is inherently parallel.
- Fusion Architecture with communication can reduce the parallel computation time in general.
- Even without communication also, Fusion Architecture performs well where less communication is expected.
- The combined effect in Fusion Architecture results in better recognition.
- The Fusion Architecture is so general, it can be used for general recognition problems also.
- The concept of combining various algorithms in fusion Architecture can be used in Artificial Intelligence search methods and in optimization strategies also.

5.5 Industrial Applications

One of the applications in the industries is that the robots have to be employed which need to move around in the presence of human beings. Here, the human beings are considered as obstacles which are represented as line segments. As the robot should not collide with the human being, this type of representation is sufficient. In some cases the area in which human beings work are specially demarcated and the robots have to avoid going through the area. The areas can be modelled as polygons with line segments. An efficient parallel algorithm to find the optimum path between any point in the industry based scenario to any other point given arbitrarily oriented obstacles is presented here. In an industry based scenario, a servicing mobile robot has to make decisions to optimize the throughput and minimize the waiting time by adopting an optimum path between the required points. Given the obstacles with varied orientations in general, the problem is computationally intensive. The work is based on the closely lower bound algorithm on intersecting segments using computational geometric approach. An efficient parallel algorithm for the problem with $O(n^2 \log(n))$ time complexity is presented here. Two different environments one where it is known a priori and other case, the robot have to explore are considered. In the third case, a multi agent system is used for an environment where the robots have to be employed in a hotel or so where they must go around the specific servicing points.

5.6 Robot Traversal in Known Environments

The interesting problem of optimum path finding [87], [85] has been solved through several approaches such as graph methods [89], [86], some enumeration techniques [96], with maps [93], [94], [92] and possibly with refinement techniques by active vision also [98], [84]. An optimal and efficient path finding in partially known environments was formulated in [97]. Recently, some more methods using computational geometry like Veronoi diagrams are also investigated. A through analysis of Robot motion planning is given in [91], [90]. Here, the principle of optimality in graph methods is combined with efficient computational geometric techniques to get an efficient parallel algorithm for the robot path planning.

5.6.1 The Formulation of the Problem

It is assumed here as in general the enclosed environment is rectangle in nature. The stationary obstacles are either assumed to be thin linear or approximated to an enclosing rectangle box. The only moving object is the mobile robot with the known dimensions. Here, it is assumed that n processors of CREW (concurrent read and exclusive write - a PRAM model) model are available. Given is a wheeled robot where the direction of motion is not impeded like car-like robot mentioned in [87]. The positions of the obstacles are known a priori and they are stationary. The initial position, the final position and the dimension of the robot (say cylindrical) are also provided. The aim is to find optimally the shortest path between the initial and final positions, if it is reachable. Here it is not only discussed about finding the shortest path but also optimally solving the problem using graph theoretical and computational geometrical methods.

5.6.2 The Parallel Algorithm

1. First sort the n obstacles lexicographically with x and y coordinates using parallel merge sort.

2. Close all intervals smaller than the mobile robot using simple geometry of circles, length of tangents, chord cutting lines for each obstacle in parallel.

3. Create all possible connections with permitted vertices only in parallel for each obstacle.

4. Given each such line segments from each obstacle, find the set of all cutting lines using the computational geometric method(to be removed).

5. Generate a graph from those line segments without cutting obstacles.

6. If the graph is a multistage graph, apply forward dynamic programming technique to get the shortest path or apply generally all pair shortest path algorithm in parallel through every vertex to get shortest path between any pair of vertices.

The Complexities of the steps are tabulated in Table 5.4. The column 2 shows the parallel time complexity of each step as explained before. In column 1, the entries, 6a and 6b denote when the graph is a multistage graph

Step No.	*Parallel Compl.*	*Workdone*
1	O(log(n)loglog(n))	O(nlog(n))
2	O(n)	O(n^2)
3	O(n)	O(n^2)
4	O($n^{'}$K log($n^{'}$))	O($n^{'2}$K log($n^{'}$))
5	O(N)	O(N^2)
6a		O(N+E)
6b	O(N^2)	O(N^3)

Table 5.4: Parallel Complexity of the Algorithm

or general graph as both are solved differently. In column 3, the work done totally by all processors (n processors here) is presented. This is also a close measure of the serial implementation of the algorithm on a uniprocessor system [80].

Here, n is the number of obstacles, $n^{'}$ is the sum of n and the number of newly added segments in step 3, K is the number of cuttings at the worst case $O(n^2)$, N is the number of vertices remaining and E is the number of edges remaining.

5.6.3 The Salient Features of the Algorithm

To begin with after getting the input, the algorithm first sorts the obstacles according to x and y co-ordinates. For the parallel merge sort is used which demands CREW, i.e., processors will be concurrently reading the data and at no circumstance, the processors will write on a datum or the data simultaneously. This algorithm as cited is given in [80]. Secondly, closing all the intervals through which the robot can not pass are made. This is done by drawing a circle of radius same as that of the robot and finding whether the end points of the obstacles lie within the circle or any of these obstacles cut the circle [83]. In both the cases, the gaps are closed. In the former case, it is easy to join the center of the circle and the corresponding end point of the obstacle which is inside the circle. However in the latter case, to which point on the line(obstacle here), the center of the circle should be connected with, becomes a subtle problem. It is easily resolved to the point on which the perpendicular line to the obstacle passing through the center of the circle cuts. Any other point may be creating further problems in becoming a member of the narrow gap for the robot. Now, all possible connections between every endpoint of an obstacle with every other endpoint of

the other obstacles are made and with the initial and final positions of the robot. It may be remembered that the endpoints cutting the periphery of the environment need not be considered. The crucial issue is finding all the cutting segments. Here, only when it cuts with an obstacle, then alone it is considered to be the cutting and not otherwise as the only moving object is the robot. This is found by the well-known computation geometric algorithm given in [95] with slight modifications suiting to the need as explained earlier. This is close to the lower bound on the algorithm finding all cutting segments. Then, the lines which do not cut with obstacles, remain to form a connectivity graph. This has to be transformed into a visibility graph. Normally the visibility graph is defined to the graph in which the vertices are the endpoints of the graph which are visible to each other, i.e., the line joining them does not cut any other edge and the line joining such endpoints form the edges. Eventhough, theoretically it is true, in practice this is not so as far as robots are concerned. For example, if A to B is visible and B to C is visible, does not mean A to C is visible. So, special care is taken to find the transitive visibility which is very crucial to the algorithm. This is carried out by taking two close points on each of the line segments AB and BC and joining them to check whether it cuts any obstacle. If so, then it is not visible. Otherwise, check the degree of the vertex B, if it is more than three and it is due to an obstacle, then it is obvious that it is not visible. At once the visibility graph is formed with transitive visibilities, it is very easy to find the shortest path using Dijkstra's algorithm or if the graph is a multistage graph, then dynamic programming can be used to reduce the complexity [88].

5.6.4 The Snap Shots of the Algorithm

Consider the example in Fig. 5.27. Here, the following is considered for the simplicity of explaining the algorithm. Only linear obstacles are taken into consideration. The robot is considered to be circular in shape with the known radius and the ability to turn to any direction. The robot has the facility to find obstacles. After getting the input data, the execution proceeds in the following way.

As the obstacles are denoted by their endpoints, the corresponding x_i and y_i points, viz. (2.0,5.0) < (3.0,4.0) and (2.0,3.0) < (2.0, 4.0) are sorted. As shown in Fig.5.28, the narrow gaps through which the robot can not pass are closed. They are highlighted with thick lines. The example is so chosen to explain both the cases of closing as explained earlier. Now, lines between the robot's initial position and the endpoints of the obstacles are

drawn including the line between robot's initial position and final position as shown in Fig.5.29. Then all lines cutting atleast one obstacle are removed. Only the lines which does not cut the obstacles are alone retained as in Fig.5.30. To differentiate the obstacles from the proposed paths for robot, the proposed paths of robot are shown with dotted lines. Similar lines are drawn from robot's final position to all endpoints of the obstacles. It is again repeated for the endpoints among the obstacles. The final connectivity graph having no cuts with the obstacles is given Fig.5.31. Now, all obstacles are removed and only the line segments having transitive visibility are retained. The visibility graph according to the algorithm is depicted in Fig.5.32. Then the optimal solution of shortest path is found and presented in Fig.5.33.

5.6.5 The Proof of the Algorithm

Time Complexity

Step 1 is a well-known parallel merge sort algorithm as given in [80] along with the proof of the complexity. In Step 2, each endpoint is taken and checked with other endpoints to find out whether the gap is sufficiently wide for the robot to pass through. This can be done in O(n) time as each check demands only constant time. Hence, the complexity is O(n) and the workdone is O(n^2). It is obvious to conclude that connecting each permitted vertex with other vertices requires O(n) in parallel with the workdone O(n^2). Step 4 is the algorithm given in [95] but the serial version. The parallelism exploited here is by separately checking of each vertex individually with the rest of the obstacles and the robot. So, each time one has to execute the serial algorithm. Hence, the time complexity and workdone are as given above. It is simple to arrive at the graphs having only the edges without cutting obstacles. It can be easily checked to have the time complexity O(N) in parallel and with n processors, the work done is O(N^2). Finally step 6a or 6b, both are well known dynamic programming algorithm and shortest path algorithm as in [88] and the parallelism exploited is just a "for loop" extension, reducing the complexity to O(N^2) with workdone of O(N^3).

Correctness

All that has to be proved is that if the final position is reachable, the algorithm should find the shortest path. Let the shortest path be p_k. Let the set of all paths be P = p_1, p_2, ... , p_m. The chance of eliminating that could happen, seems to be in step 2. Let NP $\subseteq$ P, be the set of paths eliminated in step 2. However, the chance is not possible because even if that is the

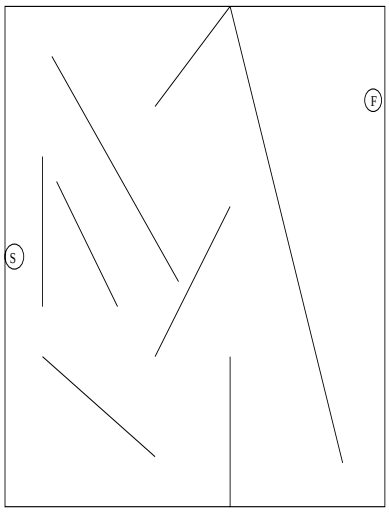


Figure 5.27: Robot in given Environment

shortest path, if the robot can not pass through, it is of no use. So, by step 2, no valid path is eliminated. Hence, if at all the shortest path exists, then $p_k \in P - NP$. Secondly, one should be assured that all paths are tried for finding shortest paths. This is guaranteed in step 3. Let CP $\subseteq$ P, be the set of path hindered by the obstacles. Steps 4 eliminates only CP. Hence, VP = P-NP-CP has p_k if at all final point is reachable. Now, the algorithm should find the shortest path given the set of paths VP. Now the graph G, having the paths VP, is alone generated in step 5. The shortest path p_k is found by algorithm in step 6. Since these being standard algorithms, it is guaranteed that if the final position is reachable, the algorithm finds the shortest path, p_k. The sole incorporated modification is the transitive visibility which eliminates only those paths which are not transitively visible. So, the algorithm finds correctly the shortest path, p_k provided the final position is reachable.

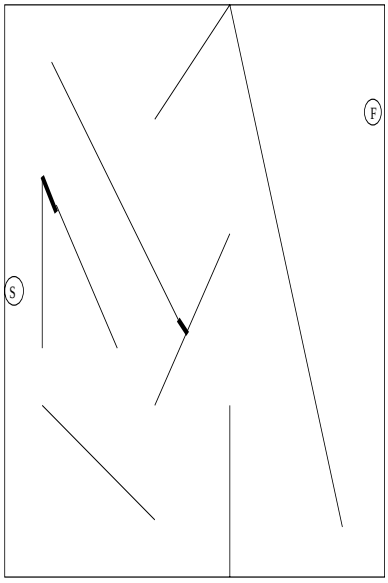


Figure 5.28: With the narrow gaps closed

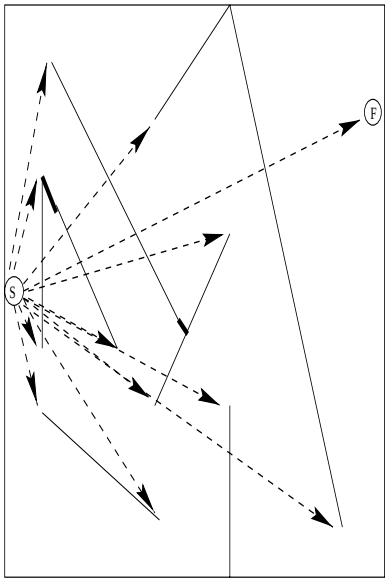


Figure 5.29: Lines between Initial and Final Positions

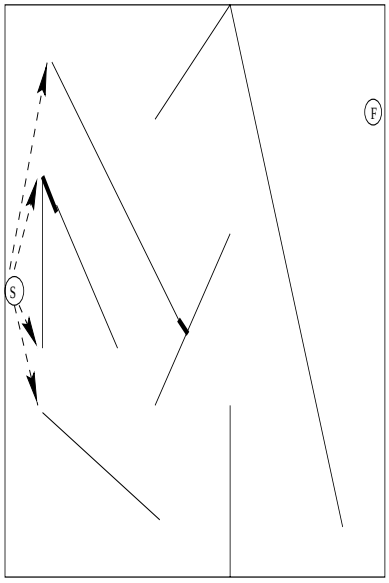


Figure 5.30: Lines not crossing Obstacles

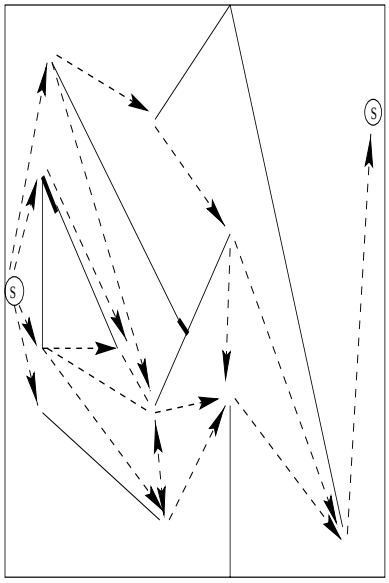


Figure 5.31: The Final Connectivity Graph

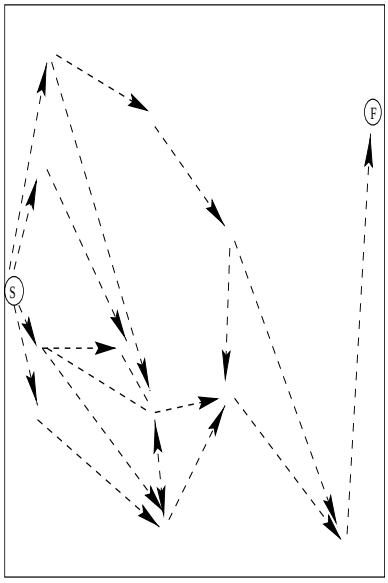


Figure 5.32: The Visibility Graph

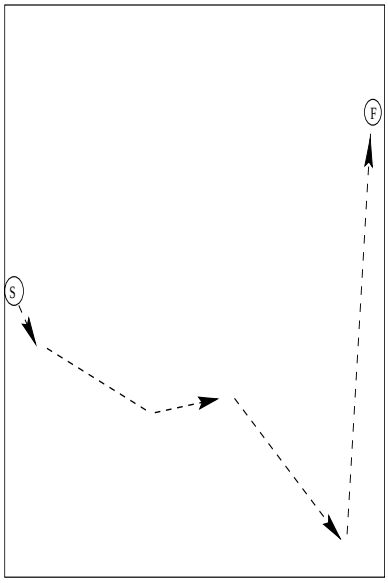


Figure 5.33: The shortest Path between Initial and Final Positions

5.7 Robot Traversal in Unknown Environments

To yield better products with higher efficiency in the modern industry scenarios, the role of the mobile robots is vital. Most of the times, the mobile robot has to reach the point of service requirement at the earliest possible time. Sometimes, due to frequent modification suiting to the current need, the operation may have to be repeated many times coping with the new environment taking into care about slight modifications. This compels to find an optimal solution which perhaps may be repeatedly used to minimize spending lot of time in traversing through some path each time rather than reaching every time through the optimal path minimizing the search effort. Recently, the researchers focussed their attention mainly on the sensor based navigation [100], [97], [89]. A dynamic graph search algorithm for motion planning in [99] describes a heuristically short motion in configuration space.

However, little attention was evinced to the efficiency of the search along the path. If a mobile robot has the required information about the environment a priori, it can precompute the optimal path performing searches in the computer memory itself using classical graph-search algorithms such as A^* [52], [103]. However, these algorithms are of less use as the mobile robot has to move physically to find the path in an unknown environment. During the course of finding the optimal path, given the initial and final positions, the robot has to make some "jumps" which are discontinuous in practical situations. Then, the robot has to spend more time in traversing the complicated path physically. The algorithms to minimize the search efforts are presented already in [103], [100], [96]. The algorithms [103], [100], make lot of search efforts leading to unbounded search in worst case [96]. In [96], the algorithm uses an A^* like behaviour to impose a bound on the depth of the search effort, and consequently to impose a bound on the search effort. It also used a DFS-like (Depth First Search) behaviour to minimize the search effort. Unlike the complicated hybrid strategy and cost propagation for updating the global costs of the nodes, three new techniques, namely, *Petri expansion, Markovian cost function, and Retaining shortest path nodes* are developed. These techniques automatically lead to minimizing the search efforts within the framework of the A^* strategy itself with minor modifications.

5.7.1 The Formulation of the Problem with Assumptions

A sensor-based cylinder-shaped mobile robot is considered here with the assumption that it has mechanism to distinguish the obstacle and locating the furthest point of visibility in the required directions. The obstacles are

assumed to be stationary at some unknown locations which can be detected by the mobile robot using the sensors. The only moving object is the mobile robot with the known dimensions. Given is a wheeled robot where the direction of motion is not impeded like car-like robot mentioned in [87]. The initial and final positions of the robot are known a priori. Given the initial and final position, the robot has to reach the final position by finding the optimal path without colliding with the obstacles and with the minimal search efforts. The focus is diverted to the minimal search efforts without sacrificing the optimality.

5.7.2 Three New Techniques to Reduce the Search Efforts in A^* Algorithm

The general A^* algorithm [52] is used here by the mobile robot to find the optimal path between the given points in an unknown environment. It may be recalled that the mobile robot standing at a point is required to go to the point which is selected for expansion by the A^* Algorithm. This discontinuous change of position is described as "jumps". The whole aim is to minimize the path traversed due to these jumps as the robot has to physically move. Three new techniques which aim to reduce the search efforts during the process of finding the optimal path always have been developed namely, Petri Expansions, Markovian Cost Function and Retaining Shortest Path Nodes. Each technique is explained in the following subsections how the search effort could be minimized in each case.

Petri Expansions

The concept of Petri net is well known and widely discussed [102], [104]. Succinctly explaining that a node is fired provided there is already one token in each of the incoming arc. This concept is transformed to suit to the need of the problem. The main idea is as follows, the search effort could be minimized if more information is available to the A^* algorithm to choose the next node to be expanded. There by unnecessary traversals can be easily eliminated. Now, the question boils down to how to get more information that too in an unknown environment. In fact, it is possible as most of the times two points P_1 and P_2 will be visible and P_1 will be expanded and P_2 waiting for its own turn for expansion. When the turn comes, the point P_2 being visible to the point P_1, will have one node with the last visited node being P_1. But, as it is known, P_1 has been already expanded. Now, without wasting time, all points visible from P_1 can be included as nodes provided

they are not visited earlier in the currently expanded node with the point P_2. These additional nodes can provide more information perhaps directly leading to the optimal path. Thereby, all intermediate nodes which were supposed to be visited will be not be visited in this case as a better choice is now available due to additional information.

The implementation is in a form of petri net. Whenever a point is expanded, it is noted as marked. During the expansion of some node, when the points which are marked are visible from the point of expansion, automatic petri expansion takes places, i.e., nodes are added with the nodes having their last point visited being visible to the marked points. This type of expansion happens as a firing as whenever the last node visited is already a marked node. This leads to lot of reduction in the search efforts as better node could be chosen for next viable expansion. The notable point is that there is very less additional effort required to do that. In fact, all the modifications are incorporated in the A^* algorithm itself. Even though, there seems to be increase in the number of nodes generated, it is in due course lead to reduction in the total number of nodes generated due to the selection of better node for expansion apart from minimizing the search efforts.

Markovian Cost Function

The characteristics of a Markov process include "forgetfulness" property [105], [101]. This forgetfulness property is used here while coining the cost function especially g(x). In the general A^* algorithm, g(x) denotes the cost involved in coming to the node x from the initial node i.e., to this point from the initial position of the mobile robot. However, due to the physical movement of the mobile robot and because of jumps, the mobile robot may have to retraverse the path to reach a point. Normally all distances traversed starting from the initial position till the point will be included as the cost incurred to reach the point. Because of the complicated paths and jumps, the measure eventhough it is really the cost incurred to reach this point, does not give much hopes to go further. Mathematically, the cost function g(x) eventhough it represents the cost incurred to reach this point, does not help in getting a better node selected as the path in which it previously traversed becomes immaterial and only the distance counts. In this juncture, the concept of Markovian forgetfulness property is introduced to the cost function.

All that is needed is a good measure of g(x) which will help further to get the optimal path with minimal search efforts. So, instead of having g(x) as the cost involved in getting the mobile robot come to the point now, which

is usually the case, the deviation is made to introduce the property to forget the past and determine the future from the point. Mathematically, g(x) need not to be the total distance traversed to reach the point, but the minimal distance required to reach the point from the initial point with the available knowledge known so far. This gives better results and with ample examples it is shown that it drives the mobile robot exactly in the optimal path as required. Of course, these may be rare examples. Yet these examples show concretely that such a possibilities are not remote. By the proper choice of the next node to be expanded, the searching effort will be reduced to the large extent. It may be also recalled, no extra effort is needed as it is the modification of the g(x) function in the A^* algorithm itself.

Retaining Shortest Path Nodes

This is one of the techniques in conjunction with other techniques yields highly appreciable results as far as the reduction of the search effort is concerned. The main idea is that what is the need of having a node with the last point P_l and another node with the same last point but a different path to reach P_l and it is shorter also. By expanding a node having the last point visited as P_l, and the distance from the initial point to P_l is not shorter compared to another node having the same last point visited as P_l and it is the shortest as per the knowledge at that moment of time available, is of no use as the path obtained as the solution can not be the optimal path because the path from the initial point to P_l is not optimal. This demands the node to be eliminated at the inception itself. This in turn will not only bound the explosion of the creation of new nodes, but also will reduce the search efforts.

The primary concern is to use this in conjunction with other techniques, eventhough individually it guarantees the optimal path also. The explosion of the creation of nodes are minimized as many nodes which are not having the shortest path to the last node P_l are summarily eliminated. Thereby any possibility of jumping to these nodes are once far all eliminated from the search space of the A^* algorithm itself. As explained, this can be easily incorporated into the $A*$ algorithm, by just checking the f(x) value as the g(x) value is already modified to incorporate the Markovian cost function.

5.7.3 The Improvised A^* Algorithm

Before the improvised A^* algorithm is presented, few techniques introduced in [53] which reduces the space requirement and the computational time are

briefed here as they are used in the algorithm.

Lower Bound

The lower bound is for the solution which is the minimum possible attainable solution. In the A^* algorithm, the algorithm has to continue even after finding a solution as it need not necessarily be optimal. Now the question lies how can it be proved that the given solution is the optimal solution so that the algorithm can be terminated at once. The only possible way is that when the given solution is equal to the lower bound solution, obviously there could not be a better solution. Hence, the algorithm can be terminated. Here, the lower bound is equal to Euclidean distance between the initial position and final position of the mobile robot. One should be always careful that all feasible optimal solutions need not necessarily be lower bound solutions. The main advantage is that if the given problem has the lower bound solution, the algorithm terminates at once it finds such solution, thereby reducing both the memory space required for the further expansions and the time to compute the same.

Upper Bound

The upper bound is a solution which is the minimum solution already available. In the A^* algorithm, the algorithm has to evaluate the function f(x) at every node. Supposing that f(x) is greater than upper bound, that node need not to be expanded further. This will not affect the optimality as anyhow by expanding the node, the solution obtained will be more than that of the already available solution. However to start with, it is assigned a very high value for example say the product of the length and breath of the unknown field if it is known. However, once a solution is found first, the upper bound is set to be the solution. Further, whenever new solutions are found, it is updated provided it is better than the already available upper bound. So, using upper bound, the number of nodes generated are minimized thereby reducing the memory space and CPU time.

The Heuristics Function

The A^* strategy mainly depends on the effectiveness of the heuristic function. At node x, let there be $P_1^{'}$ points already visited. p_0 is the initial position of the mobile robot.Then,

$g(x) = \Sigma$ Distance(p_i, p_{i-1}), $\forall p_i \in P_1^{'}$, which are visited in the node x.

Now, to find the f(x) value, h(x) heuristic function is required. To produce always optimal solution, indeed $h^*(x)$ is required. The $h^*(x)$ is defined as, $h^*(x)$ = Distance(p_l, p_f) where p_l is the last point visited in the node x and p_f is the final position of the mobile robot, and Distance function calculates the Euclidean distance between the given points. In fact, it is easy to verify that $h^*(x) < h(x)$ to ascertain the optimality.

5.7.4 The Algorithm for Finding the Optimal Path

1. Compute the lower bound solution, LB.
2. Set the upper bound UB as high value.
3. IF $(UB! = LB)$ THEN
4. c = 0 (* node count *).
5. Build the initial node N_0 with the initial point as first visited and insert it in the list with f(N_0) = LB.
6. REPEAT
7. Select the node N_k with smallest f value.
8. IF (N_k is not a solution) THEN
   (a) Generate the successors i.e., trying with all visible farthest points.
   (b) Do the following for each such points
   Include this point as the last point visited.
   (c) FOR each such visiting of points as N_i DO
   - Check for the duplication or shorter paths
   - IF (already available or not shorter) THEN
   Don't add the node
   ELSE
   Compute f(N_i) = g(N_i) + h(N_i) for this node N_i.
   IF (f(N_i) < UB)
   c = c + 1
   Insert it in the list
   IF (N_i is a solution) THEN
   IF (f(N_i) = LB) THEN
   Print the solution and quit.
   IF (f(N_i) < UB) THEN

Ex	*OP Dist*	*Nodes*	*AP Dist*	*Per Inc*
Eg1	6.0	13		
Eg2	7.23	16	10.54	45.68
Eg3	7.47	23		
Eg4	9.54	21	19.54	104.86
Eg5	26.64	24		
Eg6	17.25	3267	1211.69	6923

Table 5.5: The general A^* Algorithm

UB = f(N_i).
ENDIF
ENDIF
ENDIF
Start Petri expansion
ELSE
Prune the node N_i
ENDIF
ENDIF

ENDIF
ELSE
Print the solution and quit

9. UNTIL (N_k is solution OR list is empty). ELSE
Print the solution and quit ENDIF

5.7.5 Analysis of the Result and Future Work

To explain the effectiveness of the techniques, the simulations are carried out with various examples and few important cases are presented here in Fig.5.34 to explain the salient features of the improvised algorithm. For the sake of simplicity and explanation, linear obstacles are considered in the figures. At First, without these techniques the computations are made and then with techniques. They are tabulated in Table 5.5 and Table 5.6 respectively.

It is very evident from the examples that improvised algorithm outperforms well in the complicated situations and performs equally well in simple situations and never worse than the general algorithm excepting for

Ex	*OP Dist*	*Nodes*	*AP Dist*	*Per Inc*
Eg1	6.0	22		
Eg2	7.23	19	10.54	45.68
Eg3	7.47	27		
Eg4	9.54	38	19.54	104.86
Eg5	26.64	27		
Eg6	17.25	280	73.21	324.35

Table 5.6: The Improvised A^* Algorithm

a marginal increase in the computation time due to the additions few extra nodes. Here, Ex denotes the example sets, OP Dist denotes the Optimal Path Distance, Nodes denotes the number of nodes generated as a measure of computational time and memory space, AP Dist denotes Actual Path Distance traversed, and Per Inc shows the percentage of increase between OP Dist and AP Dist. Whenever AP Dist and Per Inc are not having values, it indicates that the path traversed is the optimal path and no extra distance is covered. It may be noted that it is same in Table 5.5 and Table 5.6 as the same f(x) is used for the sake of comparison. The most interesting is the last case, where there is commendable achievement obtained by the improvised algorithm as it is easy to check that the search effort is minimized from 1211.69 to 73.21 and that too equally good reduction in Per Inc also. This evidently shows that the three techniques in the complicated situations reduce the search efforts enormously. In the cases of Eg1, Eg3, and Eg5, the exact path traversed is the optimal path and there is absolutely no extra distance traversed. This shows that the search efforts are even minimized to zero in some cases as evident from the examples shown here. This clearly demonstrates the effectiveness of the new techniques, especially Markovian cost function. As an easy extension, the algorithm can be modified either to stop at the first solution or any ϵ optimal solution taking into consideration of the lower bounds as the optimal solution.

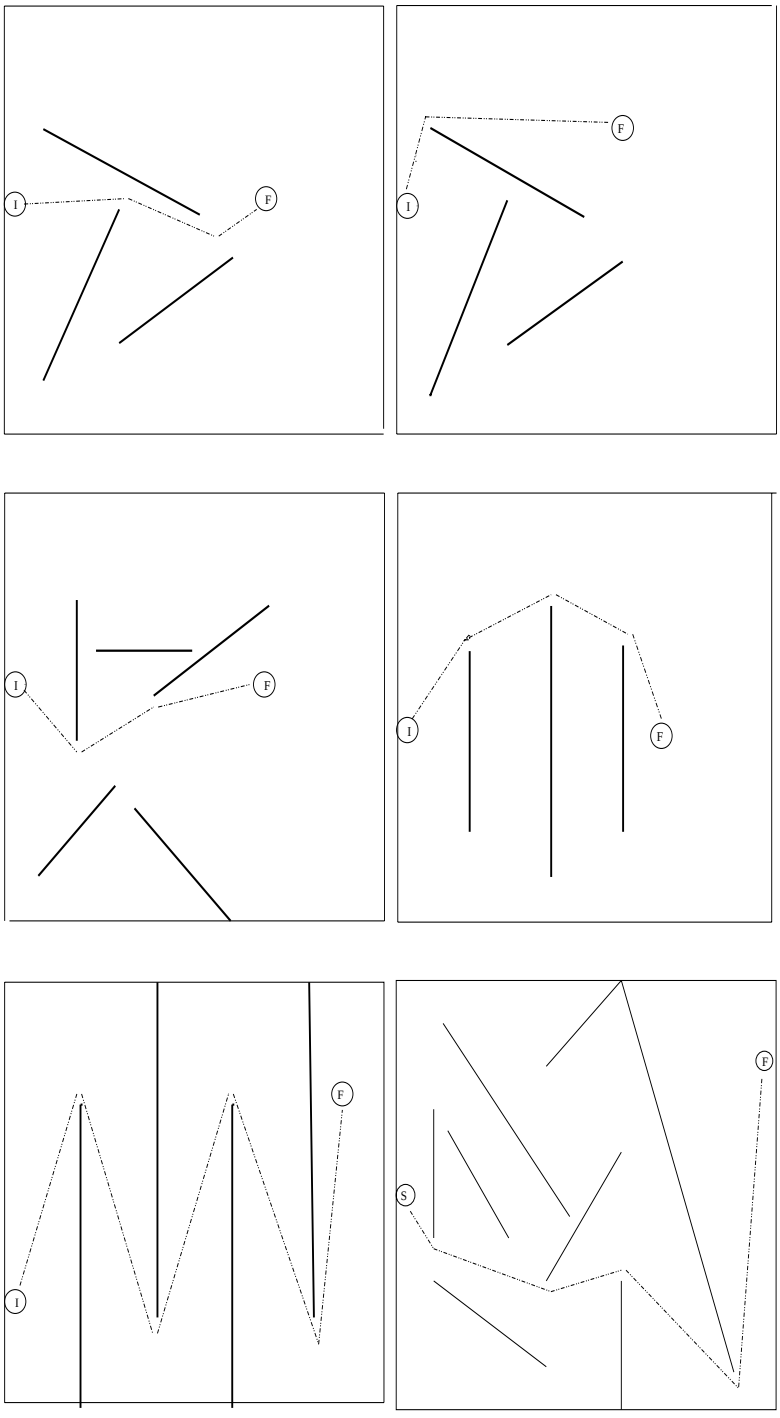


Figure 5.34: Robot in Unknown Environments Examples 1 - 6

5.8 Ants Colony Optimization Based Robot Traversal

Here the robots are employed in a restaurants where they have to go around some specific points like TSP (Travelling Salesman Problem) to serve the people. Here the concentration is on the traversal assuming the positions of the tables to be fixed where the human beings are expected to be seated. The Ant Colony Optimization, since introduced [106] has attracted more attention realizing the importance of Swarm Intelligence based on the natural phenomenon of real ants with Pheromone (trails) to trace their food. The strategy has been applied to many optimization problems including TSP. It may be recalled that initially Ants System was introduced for optimization by a colony of cooperating agents [107]. The Ant Colony System differs from the Ants System in three ways namely, different state transition rule, global updating rule and a separate local updating rule [106].

The ACO can be applied to many optimization problems including some multiobjective problems [108]. The standard TSP is chosen for experimentation as benchmarks (TSPLIB [109]) are available and widely analyzed. As TSP problem is NP-hard, obviously heuristic approaches are employed to get optimal solution. Moreover, it exhibits ample parallelism also.

In [110], both synchronous and asynchronous parallel methods are proposed and simulated. As no implementation was available, only discrete event simulation results are presented with proper assumptions. In [111], independent executions of the asynchronous method were implemented. In each execution, different or randomized initial positions are used. However, there is absolutely no communication. There is no information from the other processes which otherwise can improve the pheromone (trails). However, MMAS (Max-Min Ant System) improved the results [112]. In the method, the best ant alone updates the trails.

A new approach to implement the same, but to learn from others is introduced. As the processors in Cray T3E can communicate (unlike the independent executions), the communications with other processes (other Ant Colonies) paved way for learning from the other processes (Ant Colonies). Synchronous method with trail update after predefined iterations (to accumulate pheromones) is implemented on Cray T3E. As in the MMAS case, only the best ant in each colony is allowed to globally update the trails.

5.8.1 General TSP and ACO Approach

General TSP

A TSP can be represented by a complete weighted graph G = (V,E,f) where V is a set of cities to be visited V = {1,2,...,n}, E is a set of edges between the vertices E = {(i,j) such that (i,j) $\in$ V x V} and f is a function associated with the distances between the vertices or the length of the edges. The goal is to find a shortest cycle visiting each city once and returning back to the start city. Here, symmetric TSP is considered (f(E_{ij}) = f(E_{ji})) [109].

ACO Approach

Initially m ants are positioned among n cities according to some rule or randomly also. Each ants finds a solution using the state transition rule (in the case of TSP, builds a tour). While constructing the solution, ants modify the amount of pheromone (in the case on the edges). Once all ants have computed the solutions, once more the amount of pheromone on edges are modified according to the global update rule. As in the Ants System, ants construct the solution based on the heuristic information (nearest city) and the pheromone information (chosen by the most of the ants). After some fixed number of iterations, the best result among the ants is qualified as the optimal solution.

5.8.2 The New Parallel Algorithm for ACO

The approach is based on ACO [106] and the MMAS (Max-Min Ants System) [112]. The principal change is that instead of globally updating after every iteration (ants have constructed the solution), only after some predefined number of iteration global update is carried out. This will not only minimize the total communication time but also permits to accumulate more knowledge of the system through the accumulation of pheromone on the edges as the frequently used edges will have more pheromone. The need of MMAS is required as after some large number of iterations, there seems to be some saturation. Like in such other systems, local search is also included to improve the solution.

As the ants are independent, the algorithm is parallelized based on the number of ants, indeed on the number of colonies each having the same number of ants. Depending upon the number of parallel processors, the parallelism is scaled. Further details about the parallel implementation are provided in the following subsection. To match the number of parallel pro-

cessors and the ants, the number of colonies is approximated to the number of number of cities divided by the number of ants so that there can be at the maximum one ant positioned in one city. Only during the global update, the communication is involved and as in this case the communication is synchronous, idle time is also included.

The sketch of the parallel algorithm is given below.

Input: Coordinate positions of cities in TSP, Number of maximum iterations, Number of ants, Iteration interval, parameters rho, alpha and beta.
for each iteration do
for each colony do in parallel
for each ant do
Find a tour
Update locally pheromone (the trail matrix)
After each iteration interval
Only the best ant in each colony improves the solution with local search
globally update trail matrix by best ant in each colony only

The best tour among the best ants from each colony becomes optimal solution.

5.8.3 Outline of the ACO Parallel Program in MPI

Let n be the number of cities, m be the number of ants, r be the number of iterations and all other required parameters including the distances between the cities are read as input.
MPI_Comm_rank(WORLD,&j);
MPI_Comm_size(WORLD,&p);
MPI_File_open(WORLD,...);
MPI_File_read(...);
/*each processor j reads the input for TSP */
MPI_File_close(...);
q = n / m; /* q = the number of colonies */
Tau_{min} and Tau_{max} are calculated based on MMAS
for each iteration i=1..r do
{ for each colony k = j, j+p, j+2*p, .. q do /* in parallel */
{ for each ant w= 1 .. m do
find a tour
locally update }

```
if (i % iteration_interval == 0) /* pheromones accumulated */
for each colony k = j, j+p, j+2*p, .. q do /* in parallel */
{ do local search for each best ant in the colony to improve the solution
globally update only with the best ants from each colony
} }
MPI_Allreduce(...,MPI_MIN,WORLD); /* Optimal solution */
```

5.8.4 Parallel Implementation of the Algorithm on Cray T3E with MPI

Synchronous Communication

On the Supercomputer Cray T3E with MPI, the parallel algorithm is implemented. Eventhough MPI supports both synchronous and asynchronous communication, for the sake of simplicity of programming in parallel, synchronous communication is chosen despite the involvement of idle time required for synchronization before communication among the processors. In the implementation, only the best ant from each colony communicates with other colonies.

Here care must be taken to group the processes as in all cases the number of colonies need not to be integer multiples of the processors used. MPI has ample routines [114] to easily manoeuvre the situation. Otherwise, there might be infinite waiting presuming nonexisting processes to participate in the communication. This will wrongly increase the total idle time.

The wide variety of communication routines in MPI facilitates the global update of pheromone in a simple way of course with the above mentioned condition. It must be recalled that for local update no communication is required as each colony is associated with each processor having separate memory.

Communications are also required at the end of the algorithm to collect the results and choose the best as the optimal solution. MPI has efficient implementation of find maximum or minimum of a particular value among the processes in the parallel processors.

5.8.5 Parallelism in ACO Algorithm

As explained in the previous section, it is natural to parallelize the algorithm based on the number of ants (indeed the ant colonies)and the available parallel processors. In ACO algorithm developed in the work, each colony has same number of ants. So, instead of parallelizing at the ant level which will

increase the communication and idle times as global update requires all ants to update the pheromone, at the colony level the algorithm is parallelized. This will not only minimize the number of times communications ought to be carried out but also the amount of data to be communicated.

Moreover, instead of updating after each cycle which will also increase both the total communication and idle times, the algorithm permits only after a specific predefined interval (can be modified at the run time), the pheromones are globally updated among the processors. It may be recalled that local update as mentioned earlier does not demand any communication as all the ants in each colony will be associated with the same processor. The global update is also parallelized efficiently. The set of processors participate in global update are grouped to use efficient routines in MPI.

5.8.6 Experimental Results and Analysis

Here for the experimentation, a standard benchmark data for TSP Berlin52.dat [109] is chosen. To compare with other algorithms, (1)ant system(AS), (2)Min-Max AS (MMAS), (3)MMAS with local search (MMAS-LS), (4)ant colony system(ACS), (5)ACS with local search (ACS-LS) and (6) the algorithm developed in the work, the experiments were conducted on 4 parallel processors without changing any other parameter viz. rho for MMAS, alpha and beta for state transition rule. The five major issues [80], 1)length of the tour(optimal solution - Tour len), 2) total computation time ($Time_{par}$) 3)single processor time ($Time_{pcr}$) 4) percentage of total communication time with respect to total computation time ($Time_{comm}\%$) and 5) percentage of total idle time with respect to total computation time ($Time_{idle}\%$) are analyzed as shown in Fig.5.35 by varying the number of iterations from 100, 500 and 1000.

From the graphs in Fig.5.35, it is clear that the algorithm developed in the work performs competitively well and it may be recalled all experiments were done with 4 processors. The only problem with the method was more idle time. It is because less processors were used. It becomes very evident when the number of processors are increased as shown in the tables. The least communication time by the algorithm promises that it can be used for larger size problems also.

For constructing the tables, only the algorithm developed in the work is used, but all parameters are varied and the results are analyzed. From Table 5.7 the increase in the number of iteration obviously produces better results. However, there is some sort of saturation or falling into local minimum is observed. From Table 5.8, by increasing the number of ants, ultimately

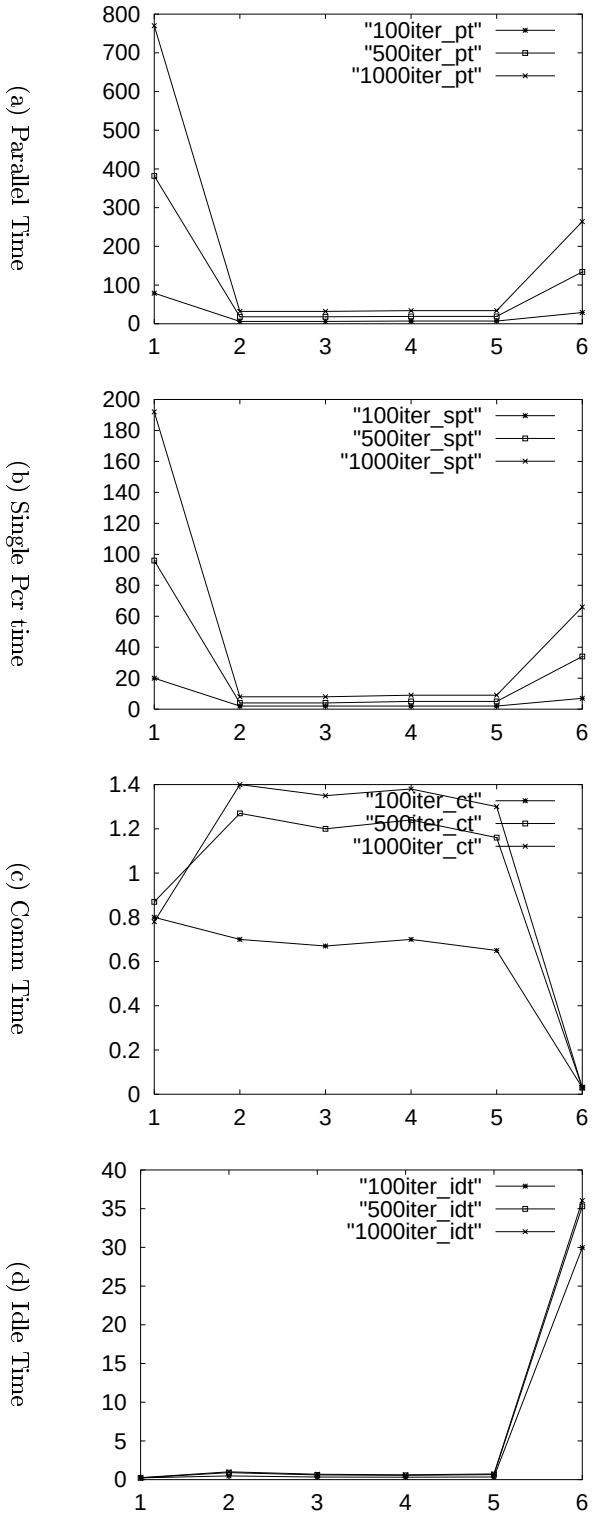


Figure 5.35: Parallel, Single Pcr, Communication and Idle Time Analysis with 6 Algorithms

No.of Iter	*Tour len*	$Time_{par}$	$Time_{pcr}$	$Time_{comm}\%$	$Time_{idle}\%$
500	10027	111	11	0.18	0.36
1000	9857	205	21	0.20	0.38
2000	9857	395	39	0.21	0.41

Table 5.7: Results based on varying Iterations

No.of Ants	*Tour len*	$Time_{par}$	$Time_{pcr}$	$Time_{comm}\%$	$Time_{idle}\%$
2	9838	428	16	0.75	0.45
4	9857	231	17	0.69	0.41
5	9857	205	20	0.41	0.38

Table 5.8: Results based on varying Ants

Interval	*Tour len*	$Time_{par}$	$Time_{pcr}$	$Time_{comm}\%$	$Time_{idle}\%$
10	9838	601	23	0.82	0.48
5	8200	869	33	1.12	0.49

Table 5.9: Results based on varying Interval

Rho	*Tour len*	$Time_{par}$	$Time_{pcr}$	$Time_{comm}\%$	$Time_{idle}\%$
0.9	8200	607	23	1.02	0.48
0.5	8200	607	23	1.04	0.46
0.1	8200	607	23	1.07	0.44

Table 5.10: Results based on varying Rho

Alpha,Beta	*Tour len*	$Time_{par}$	$Time_{pcr}$	$Time_{comm}\%$	$Time_{idle}\%$
1,5	8200	607	23	1.08	0.47
2,5	8200	609	23	1.04	0.46
5,1	8200	604	23	1.05	0.39

Table 5.11: Results based on varying Alpha and Beta

the number of colonies are reduced as (colonies = cities / ants). As the colony size reduces with the increase of number ants in a colony, no better results are produced. This defends idea of updating globally based on best ant in each colonies than with all ants. From Table 5.9 it is clear that by reducing the interval size, the number of communication increases. But this gives more dividend by producing better results because more global information is shared but in a restricted way. From Table 5.10 and Table 5.11, no marked change is observed by changing rho and alpha and beta. However more experiments with different data can prove the role of each parameter explicitly. The parallelism perfectly works giving better results and at the same time both communication and idle times are contained. This leaves less aspirations for using asynchronous communications for the algorithm. Eventhough the best result for the TSP problem is less than the obtained results, the strategy derives strength from restricting the global update not after every iteration but after some period of intervals. This in effect not only reduced the communication time and idle time but also shared the pheromone values which is the crucial aspect in any ants based multi agent systems.

5.8.7 Future Extensions

Apart from varying all the parameters, the role of local search must be addressed. After some iterations, with out local search, pheromone saturation seems to be evident. The change in state transition rule from AS to ACS which balances such cases also must be exclusively studied. After a large iterations by changing rho or alpha or beta does not always lead to better solutions. Modifying these at runtime or as the iterations proceed, the ability to get better results can also be investigated. The efficiency of MPI routines can also be found by varying the strategies. Finally, asynchronous mode of communication must also be studied to improvise both the algorithm and the implementation of MPI routines on Cray T3E also. Attempts can be made to solve other optimization problems apart from TSP.

Chapter 6

Model, Matching and Indexing

6.1 Models

In the recognition systems, the role of the models is very important not only in recognition but also in reducing the computation time. From stick models of human being to 3D blob models complexity increases but the tracking later becomes comparatively faster. To recognize a human being despite occlusion requires either explicit occlusion modelling or in Hausdorff method, proper selection of some parameters. It is always difficult to find values of the parameters which will suit to many instances in general. However, specially modelling for occlusion will help. But it will increase the number of models several times. Yet it manoeuvres occlusion better.

6.1.1 Occlusion Models

In Occlusion models, the model of a human being is taken and some of the parts are removed. Each model after such deletions of parts becomes an occlusion model. By explicitly modelling occlusion, the probability of recognition has been increased as evident from the experiments. As mentioned before, the method increases the number of models. To circumvent the increase in the number of models, a generic model is introduced.

6.1.2 Generic Models

To reduce the number of models, the selected set of models are superimposed together. Such a model is called a generic model. In case of points, a

generic model is the union of all the sets of points from each selected model. Similarly, for the edge models also it is defined like points.

However, generic models eventhough reduces the number of models, they have problem with Hausdorff method for matching as many points may be missing. Eventhough using the same technique for occlusion will increase further probability of recognition, to alleviate the problem of missing points in the generic models, basis models are introduced.

6.1.3 Basis Models

This modelling is almost complementary to the generic models. Instead of taking union among the selected sets of points as in the case with generic models, here intersection is taken. Eventhough the method looks rosy, there is a fly in the oil.

In many cases, the number of points in the intersection set is very less, so that they match with arbitrary set of points often increasing the number of false positives. However if the selection of the models is proper, in the sense that most of them are similar to the large extent, the method will work appreciably. It must be recalled that similar idea is used [10] in forming a hierarchical way of grouping the models.

6.2 Matching

As in the study, Hausdorff and Chamfering matching are appreciably efficient matching methods. However general matching methods have some inherent problems due to scaling which may be solvable by defining the matching measure differently.

6.2.1 Matching Problem with Scaling

Both the Hausdorff and Chamfering methods for matching do not cope up with scaling adequately. Albeit the problem can be solved by having models of varied sizes, the ultimate problem of fixing proper threshold which can still accept, continues. Moreover, theoretically there can be infinite scaling variations for a single model. Hence, it will only increase the number of models and complexity.

Taking a closer look at an image of different size, the matching is obvious visually. However, unless the corresponding model of the size is not there, the acceptance depends upon the value of the threshold. Instead of putting the onus on the value of threshold, a different outlook is taken which

defines the matching measure differently which will enhance the matching with different sizes.

6.2.2 A New Matching Measure

Defining a matching measure is as simple as defining a function over a set of points and values at those points. However, if it should suffice as good matching measure, especially in the situation where the sizes are the paramount problems, the following measure increases the confidence in successful matching. To recall the matching measure in Hausdorff, it is the maximum distance value in those feature points. That is why when a segment of an image is a different size of the same model, it is not able to accept as the maximum distance value may be more than the threshold. Similarly, in the case of Chamfering, the distance measure is the square root of the average of the sum of squares of the distances at the feature points. It is something like taking the average value only, but in a different way to suit to the distance transformation.

For the new measure, first the average of all the distance values at the feature points is taken and it is subtracted from the distance values at the feature points and the absolute value of the new difference is alone considered. Then, the maximum is found which is the value of the new measure.

The new measure combines both the chamfering and Hausdorff distance measures. For simple objects like circles, circles of any radius can be easily recognized with a single model with the new distance measure. When the objects are cylinders and blobs depending upon the sizes the differences also increase. For the models of human being, when the sizes are nearly same, then obviously the correct matching is found. Then again as the sizes increases, playing with the values of threshold, suitable match can be found positively.

In any matching method, given a set of models and an image, which model has to be chosen first and what is the order in which models can be chosen so that at the earliest all correct matching can be found depends on indexing.

6.3 Indexing

6.3.1 Problems of Outdoor environment compared to Image Databases

Image Indexing is one of the fields which attracts many of the researchers to focus the attention due to its wide applicability. From matching finger prints to finding the culprit from the image databases, image indexing plays an important role. But the problem with recognizing human beings in an outdoor environment is more complicated than the image databases. Firstly, image databases may have the entries of same size or atleast approximately same. Since the chances of preprocessing can be done off-line, most of the noises can be reduced in the images stored in the databases. More importantly, all the features of the image are in the image itself (if the feature exists).

In case of outdoor environments, neither the sizes can be same, nor there is any guarantee for noise free images. Above all, in such images how segmenting can be done in such a way that relevant features are available within the segment itself and the features will not be distorted due to segmentation or dividing the image into equal smaller sizes?

6.3.2 Segmenting Problems

There are lot of segmenting methods based on colours or texture or same grey values or connected components and so on. This will disturb the generality of the problem considerably. There is not much interest at this advanced stage to backtrack to use colour cues to segment the image.

At the same time, equally dividing the image into segments of smaller sizes, as mentioned before has to handle the features falling exactly at the dividing lines. Moreover, there is no the guarantee that the segment satisfying the conditions based on histogram must match with the model. Any arbitrary segment can have the same histogram like a model. So, after indexing applying robust matching methods are also essential.

6.3.3 Possible Indexing Strategies

Histograms or measures based on either Hausdorff or Chamfering can be used for indexing. But the primary problem is how to segment a given image such that if a model exists, the indexing will choose such a model first. Lot of work needs to done in the direction which will drastically reduce the computation time.

As histogram is the simplest and at the same time powerful one, indexing can be done based on the histogram values. A combination of maximum, minimum and median can be a combined indexing key.

6.4 Backward Recognition of Human Groups

The idea is if at a particular point of time a human being is recognized, from the previous image where the human being would have been, the human being can be recognized along with other human beings. As there are many problems including sizes, movements of human beings, movement of the camera and so on, a restricted environment is taken for study. For the restricted environment, the backward recognition works well as shown in Fig.6.1, Fig.6.2, Fig.6.3. Here, recognizing a single human being is done first and later the relative position is back propagated to group the other human beings. The occlusion plays a vital role in impeding the recognition process. In such situations, it is more crucial as even one human being is not recognized, may lead to the group not recognized.

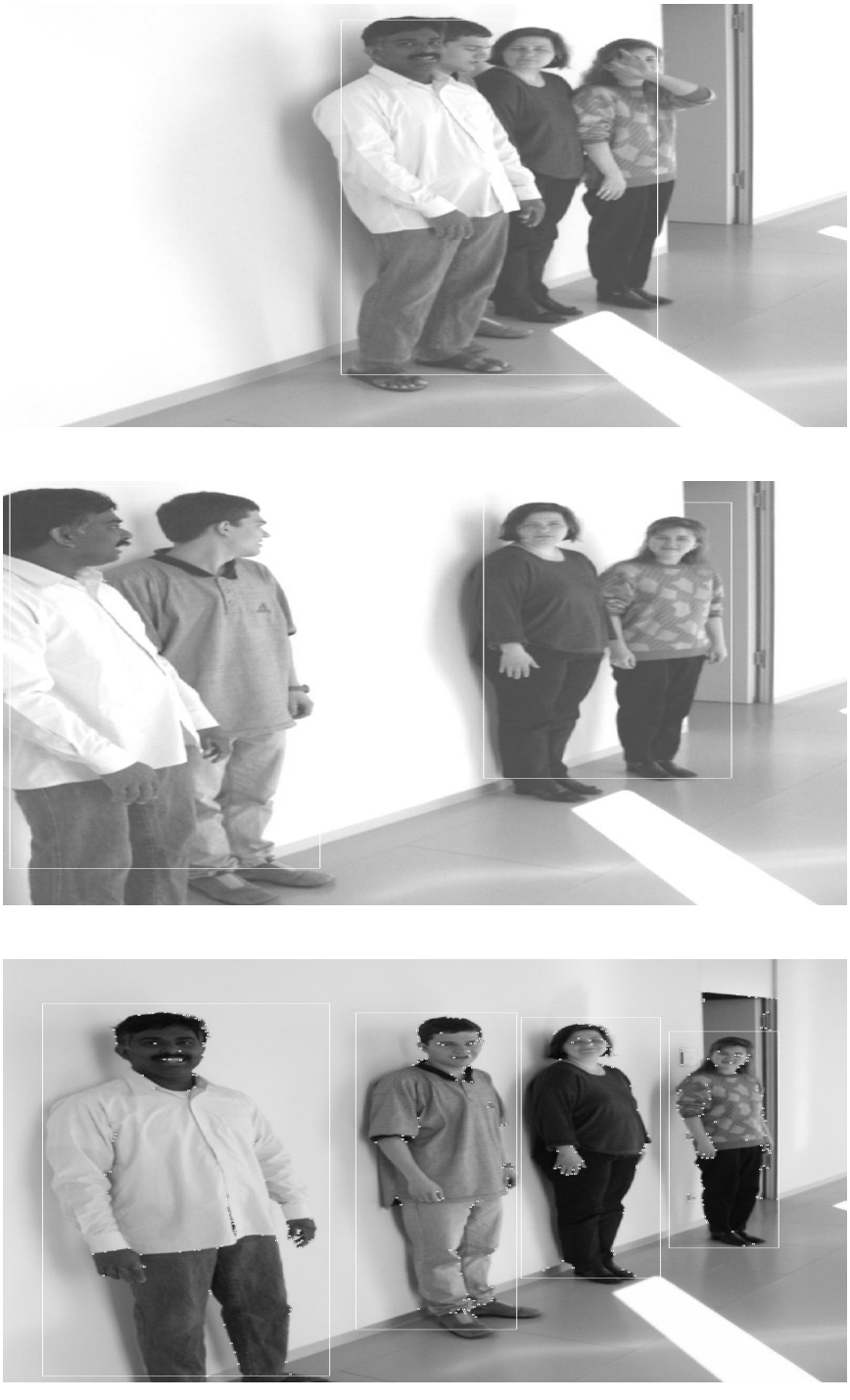


Figure 6.1: Identified Human Groups in Images from Image Understanding Group 1

Figure 6.2: Identified Human Groups in Images from Image Understanding Group 2

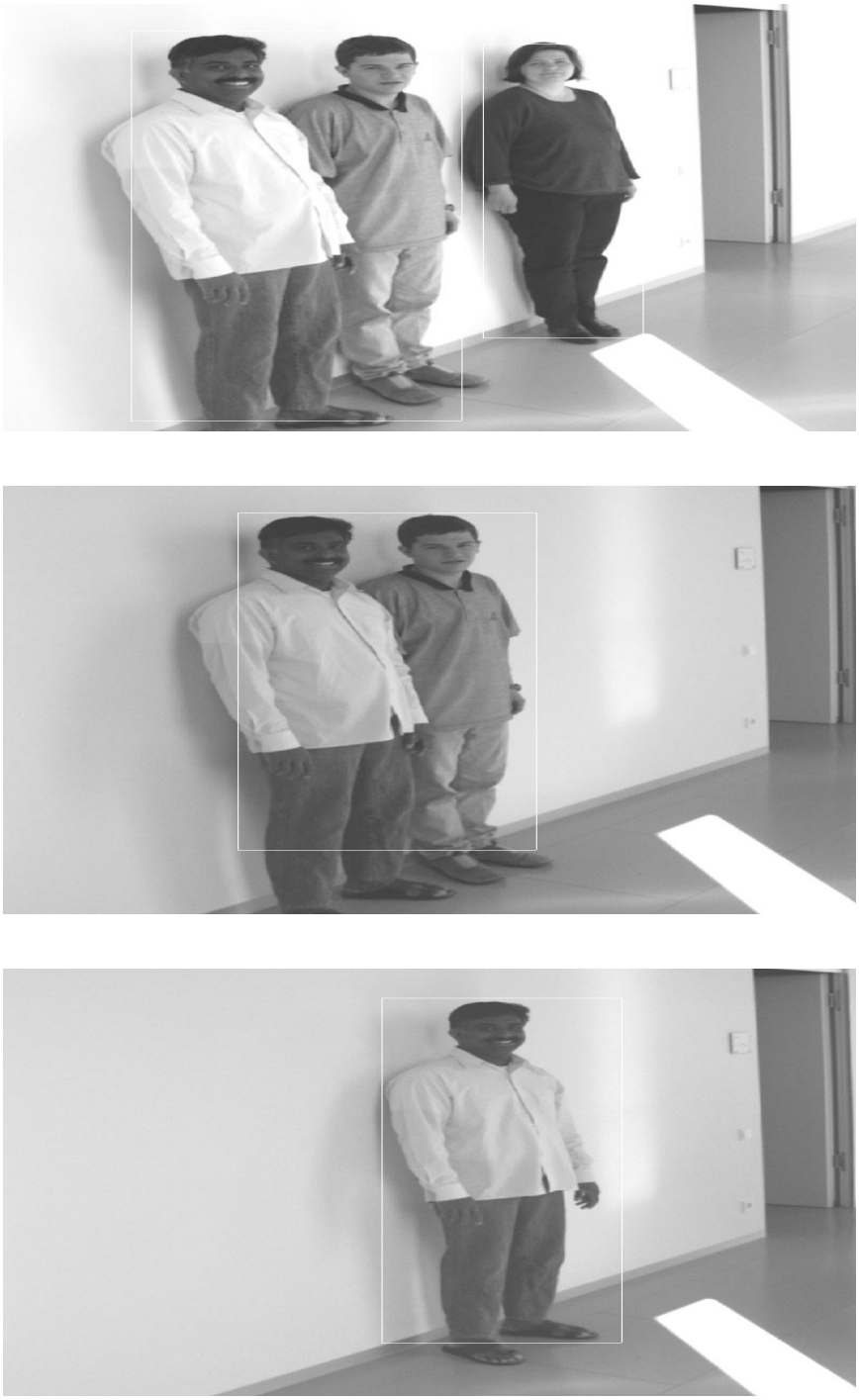


Figure 6.3: Identified Human Groups in Images from Image Understanding Group 3

Chapter 7

Conclusion

The growth of diversified applications that demand the recognition of human beings in the images has increased considerably. In the study, an attempt has been made to recognize human beings in images without many strict constraints which are normally applied and it has been shown that the possibility is high. To start with graph matching, the basic concept of matching two graphs is investigated as the graphs are the most powerful representation of objects mathematically. A new isomorphism (Neighbour Isomorphism) has been introduced which reduces the computation time to match the two graphs enormously. The same isomorphism is extended to find the symmetries in the regular polygons which are repeatedly present at various positions. The symmetry axis and the relationship with the new isomorphism is deduced clearly. Initially, A^* Algorithm is used for optimal matching of graphs also. Combining both Neighbour Isomorphism and A^* strategy, a new algorithm to match the postures of human being especially taken from the indian classical dance, Bharathanatyam, has been developed which produced correct results efficiently.

In the real images due to noises, such graph methods are not directly applicable in low level image processing. Here, the concentration is focussed on low level image processing to recognize a human being. The two standard matching methods, Chamfering and Hausdorff method are investigated. It has been found that the Chamfering method is faster than the Hausdorff method as far as computation times are concerned. But the Hausdorff method is more robust than the Chamfering method. The efficient computation of distances in the images with respect to the features based on distance transformations and lookup methods are discussed and it is found that distance transformation method is computationally better.

The crucial contribution of Fusion architecture in the study is basically highly general and not restricted only to human being recognition. The combination of various algorithms produced better recognition in the Fusion Architecture. The modified Hausdorff methods to handle occlusion and scaling or zooming improvised the combination in the Fusion Architecture. Being inherently parallel, the Fusion Architecture can be easily implemented on parallel machines.

To do lot of experiments and for reducing the computation time, all the strategies are parallelized. The parallel implementations are done on Cray T3E supercomputer using MPI. The results show that the parallelization can obviously reduce the computation time depending upon the amount of parallelization and the number of processors available.

Apart from the general problems involved in recognition systems, the aspects of model, matching and indexing strategy are analyzed with the different approaches of occlusion modelling with generic and basic models. To describe the positional relationships between the human beings ontologically, experiments are performed and the preliminary results are encouraging. As industrial applications in robotics, three situations, one with known environment, another with unknown environment and the third with going around have been discussed considering primitive models of human beings as obstacles. In restricted environments, recognizing groups of human beings is also possible by recognizing single human beings separately in a sequence of images and back propagating the relative positions in the previous images along with other human beings.

Albeit the aim of the study is to recognize the human beings in the images from monocular camera without usual constraints, initially the graph theory based methods for matching are analyzed with new neighbour isomorphism. The robust Hausdorff method for matching is extended to recognize the human beings with ample models and modified distance measures. As the strategy to fuse different algorithms to get better results despite occlusions is inherently parallel, it is implemented on Cray T3E Supercomputer which produced correct results in appreciably lesser computation time. Unless the method of choosing a proper model with the good indexing is available, the recognition of human beings will continue to remain as one of the hard problems to be solved in image processing.

Appendix A

Cray T3E

As the Supercomputer Cray T3E at HLRS [113] is accessible, the parallel algorithm for image matching is implemented and tested on the platform. It has 512 nodes, 64 GB DRAM memory, 128 MB DRAM memory per node with the peak performance of 461 GFLOPS/s. It has high communication and I/O Bandwidths and operates on Chorus based operating system.

The system is highly scalable with distributed memory. Message Passing Interface (MPI) model is supported by Cray T3E. This MPI allows parallel file I/O upto 200 Mb/sec. Both interactive and batch mode of executions are allowed on Cray T3E.

The MPI on Cray T3E has with standard MPI_Send and MPI_Recv a latency of 6 microseconds, and with messages longer than 8 kbytes a bandwidth of faster than 220 Mbytes/sec, with messages longer than 64 kbytes a bandwidth of faster than 300 Mbytes/sec, and with messages longer than 256 kbytes a bandwidth of about 315 Mbytes/sec.

The automatic MPI profiling and totalview software to debug make the programming and development really easier. Compilation can be done with variable number of processors so that the number of processors can be changed at runtime using the following commands.

cc -o <object file name> <source file name>

mpirun -np <number of processors> <object file name>

To use the file system in batchmode get_input and save_result shell scripts are also available.

Appendix B

Message Passing Interface

In the programming Language C, the message passing interface (MPI) is included as a library [114] as MPI being not a language as such. However, MPI can be included in many languages such as Fortran and C++. The set of library routines enable to communicate among the processors. In fact, MPI specification is portable which takes advantage of the specialized hardware and software offered by the individual vendors. MPI parallel file reading is used here as the individual file reading. Just to find the minimum, MPI-Allreduce is used. The same program is compiled and executed on different number of processors without any change in the program.

Here, a list of MPI standard routines is provided to understand and program in parallel systems. For exact arguments and their types, the useful references are [114] and [115].

1. MPI_Init(); Every program must start with routine.
2. MPI_Finalize(); It is the final routine called from MPI programs.
3. MPI_Comm_create(...) creates a new intercommunicator.
4. MPI_type_create_subarray(...) is to create subarrays from the main array such that they can be processed in parallel.
5. MPI_Send and MPI_Recv are some of the routines for transferring data between the processes.
6. MPI_Allgather(...) and MPI_Allreduce are collective operations where all processes contribute to the result which is received by all.
7. MPI_Gather(...) and MPI_Reduce are collective operations where all processes contribute to the result which is received by one.

8. MPI_Bcast(...) and MPI_Scatter are collective operations where one process contributes to the result which is received by all.

9. MPI_Barrier(...) is used to synchronize the processes.

10. MPI_File_open(...); opens the file identified by the filename on all processes in the communicator group. Files can be opened in read or write or read only and such other modes.

11. MPI_File_read(...) is to read a file by a processor (noncollective operation) and MPI_File_read_all(...) is to read a file by all the processors in the communicator group (collective operation).

12. MPI_File_write(...) is to write on a file by a processor (noncollective operation) and MPI_File_write_all(...) is to write on a file by all the processors in the communicator group (collective operation) at specific positions.

13. File operations can be done in blocking and nonblocking modes with explicit offsets or individual file pointers or shared file pointers such as MPI_File_read/write_at/at_all/shared/ordered.

Appendix C

File Interoperability with Parallel MPI File-I/O

The I/O operations especially file related operations are investigated. The significant optimizations required for efficiency can only be implemented if the parallel I/O system provides a high-level interface supporting partitioning of file data among processes and a collective interface supporting complete transfers of global data structures between process memories and files [115]. Parallel reading of the same image or model into the memory of several MPI processes can be implemented with the MPI_File_read_all. This collective routine enables the MPI library to optimize reading and broadcasting the file information into the memories of all processes. In image processing, there exists also a huge number of different formats to store the image data in files. The standard image processing software gives the options of a proprietary format or a standard ASCII format. Because most of the formats can be converted into ASCII file format in many systems, and to circumvent problems with the 64-bit internal integer format on the Cray T3E, the ASCII format is decided as the image(model also) file format. Therefore, it is mandatory to implement the conversion of ASCII file (mostly representing integers being pixel coordinates and grey values) so that file Interoperability in MPI can be used effectively for image processing. As the sizes of the files increase obviously the I/O overheads also increase. In image processing, there will be always many files required both for images and models. Hence, it is not only the sizes of the images, but also the number of them is a matter of concern for I/O overheads.

C.1 Data Access Routines

The file Interoperability means to read and write the information previously written or read respectively to a file not just as bits of data, but the actual information the bits represent. The data access routines provide the data movement between files and processes. There are three orthogonal aspects to data access, 1. positioning (with offset or implicit file pointer), 2. synchronism (blocking or non-blocking) and 3. coordination (collective or non-collective) [115]. Like data access routines, File interoperability has three aspects namely, 1. transferring the bits, 2. converting different file structures and 3. converting between different machine representations. The third being the concern here, the multiple data representations and the inability to read integer data stored in an ASCII file which is needed for image processing are explained in the following subsection.

C.2 Data Representations

MPI-2 defines the following three data representation, 1. *native*, 2. *internal* and 3. *external32* [115]. In *native* representation, the data is stored in a file exactly as it is in memory. In *external32* format, also a binary data representation is used. Obviously, it is impossible to use these formats directly to read integer data from ASCII files. The *internal* representation cannot be used for data exchange between MPI programs and other non-MPI programs that have provided the image data because the internal representation may be chosen arbitrarily by the implementer of the MPI library. MPI-2 has standardized also a method to use *user-defined* data representation. Here, the user can combine the parallel I/O capabilities with the byte-to-data conversion routines. The major constraint is that the representation of a given data type must have a well-defined number of bytes. As the number of digits of integers in an ASCII file vary (and each integer may end either with a blank or an end-of-line character), user-defined representation also cannot help reading integers efficiently from ASCII files.

C.3 Reading Integer Data from ASCII File with MPI I/O

The former constraints force the implementation of the following strategies:
Normal File Reading with fscanf In the first strategy, the files are read using normal file reading command *fscanf* instead of MPI for the sake of

ranks=0	1	2	3	4	5	6	7
R(0,0)	R(1,1)	R(2,2)	R(3,3)	R(0,4)	R(1,5)	R(2,6)	R(3,7)
c(0,0)	c(1,1)	c(2,2)	c(3,3)	c(0,4)	c(1,5)	c(2,6)	c(3,7)
r(1)	r(2)	r(3)	r(4)	r(5)	r(6)	r(7)	r(0)
c(0,1)	c(1,2)	c(2,3)	c(3,4)	c(0,5)	c(1,6)	c(2,7)	c(3,0)
r(2)	r(3)	r(4)	r(5)	r(6)	r(7)	r(0)	r(1)
c(0,2)	c(1,3)	c(2,4)	c(3,5)	c(0,6)	c(1,7)	c(2,0)	c(3,1)
r(3)	r(4)	r(5)	r(6)	r(7)	r(0)	r(1)	r(2)
c(0,3)	c(1,4)	c(2,5)	c(3,6)	c(0,7)	c(1,0)	c(2,1)	c(3,2)
R(8)	R(9)	R10)	R(11)	R(12)	R(13)	R(14)	R(15)
c(0,8)	c(1,9)	c(2,10)	c(3,11)	c(0,12)	c(1,13)	c(2,14)	c(3,15)
r(9)	r(10)	r(11)	r(12)	r(13)	r(14)	r(15)	r(8)
c(0,9)	c(1,10)	c(2,11)	c(3,12)	c(0,13)	c(1,14)	c(2,15)	c(3,16)
...	...	...	...	...	...	...	...
R(4,q-1)	R(5,q-2)	R(6,q-3)	R(7,q-4)	R(4,q-5)	R(5,q-6)	R(6,q-7)	R(7,q-8)
...	...	...	...	...	...	...	...

Table C.1: Parallelization scheme of I/O and computation.

comparison with MPI file I/O operations. It may be recalled that there is no need for conversion as fscanf can directly read the integers from the files.
Off-line Conversion In the second strategy, the ASCII file is converted into a native file by a separate program. This gives the facility to convert the required ASCII file off-line which enables the image processing program to read the native file without any difficulty. To achieve heterogeneity, MPI external 32 data representation can be used instead of the native format.
Runtime Conversion In the third strategy, the entire ASCII file is read into a large buffer of type CHAR, and then individually by reading every character till it is terminated either by a blank or by an end-of-line character, the same is converted into an integer at run-time. In fact, the original file remains as ASCII file and is still used. The conversion can be stored as a native file for further use, if the need is so. It may be recalled the ASCII to Integer conversion function is very easy to implement which is also system independent.

C.4 Optimizing the Parallel I/O

The image data usage pattern has two chances for optimization: (a) all image data must be reused (and probably reloaded) for comparing with several models, and (b) all models must be reused (and probably reloaded) for comparing with several images. In the sequential version of the software, each image is loaded once and all models are loaded again for comparing with each image. By reversing the sequence of models for each even image number, at least the latest models can be cached in memory. In the first parallel version loading of the images can be optimized with collective reading into all processes.

If more than one image can be analyzed in parallel, i.e., if one can accept an additional delay for the analysis of an image because not all available processors are used for analyzing and because the start of the analysis is delayed until a set of images is available, then the parallelization can be optimized according to the scheme in Table C.1. The scheme shows the analysis of 4 images in parallel on 8 processors. R(i,k) denotes reading of the image i and model k, R(k) is only reading of model k, r(k) is receiving of model k with point-to-point communication from the right neighbor (sending is omitted in the figure), and c(i,k) denotes the computation of the Hausdorff distance for image i and model k.

Looking at the scheme, note that reading the image into several processors at the same time (e.g., image 0 into processes 0 and 4) can be still optimized with collective reading (MPI_File_read_all) that internally should optimize this operation by reading once from disk and broadcasting the image data to the processes. Reading several images and models at the same time can be accelerated by the use of striped file-systems. The scheme is also optimized for a cluster of shared memory nodes (SMPs). The vertical bar between rank 3 and 4 may donate such a boundary between SMPs. One can see on each node, that only one model is received from another node (and another model is sent) while exchanging all models.

C.5 Results and Analysis

For the purpose of illustration, four sample images (one shown in Fig. 4.14) and four models (one shown in Fig. 4.15) are considered. The algorithm is tested with 1, 2 and 4 processors on the Cray T3E-900 at HLRS. As the interest is on I/O, the I/O timings per process are tabulated in Table C.2 for 4 images and 4 models. The timing is done with MPI_Wtime(). The wall clock time per process to handle the reading of 4 images and 4 models, including repeated reading or message exchanges of the model is shown in Table C.2. Before starting each I/O timing, a barrier is done to prohibit that any synchronization time is assessed as I/O time. Although the I/O requires only a small part of the total execution time in the current version of the program, it is expected that on faster processing systems and with better optimization of the Hausdorff algorithm, I/O will be a relevant factor for execution speed. In the *original* parallelization, each image is read by all processes (which may be optimized by the MPI library), and for each image, each process reads only a subset of the models according to the numbers of processors. In the *optimized* parallelization, each image is read by only one

process, and for each set of images analyzed in parallel, each model is read only once and then transferred with message passing to the other processes. Table C.3 shows the accumulated number of reading an image or model file or transferring a model for the test case with 4 images and 4 models. Each entry in Table C.3 shows the accumulated number of *images read* + *models read* + *models exchanged* by all processes with the different parallelization schemes, e.g., 4*2+16+0 means, that 4 times 2 identical images, and 16 models are read, and 0 models are exchanged by message transfer.

The experiments are started with normal reading with fscanf. The original parallelization incurred larger I/O time because each image had to be read on each processor again. In the second experiment the reading is so parallelized and each fscanf is substituted by MPI-2 file reading. Because reading of ASCII integers is not available in MPI-2, reading the same as characters is chosen. Normally each integer is expressed only with a few characters, therefore, the expected additional overhead was not expected very high. But the measurements have shown that the solution was 46 times slower than the original code. The MPI-2 I/O library on the Cray T3E could not be used in a similar way as fscanf() or getc() can be used. To overcome the high latency of the MPI I/O routines, reading the whole file with one (experiment No. 3) or only a few (No. 4) MPI operations were implemented. But there is still no benefit from parallelizing the I/O. The I/O time per process grows with the number of processes and the accumulated I/O time with 4 processors is therefore 4–6 times more than with one processor. In the last two experiments, the parallelization was optimized to reduce the number of reading of each image and model. This method achieves an optimal speedup for the I/O. But also with the optimization, the fscanf solution is about 10 %faster than the MPI I/O solution on 4 processes.

These experiments have shown that (a) MPI I/O can be used for ASCII files, (b) but only large chunks should be accessed due to large latencies of MPI I/O routines, and (c) optimizations that can be implemented by the applications should be preferred than optimizations that may be done inside the MPI library, (d) as long as many small or medium ASCII files should be accessed, it may be better to use standard I/O by many processes and classical message passing or broadcasting the information to all processes that need the same information, than using collective MPI I/O.

No.	*Parallelization*	*File Op*	*Conversion*	*I/O Entities*	*1 proc*	*2 proc*	*4 proc*
1	Original	fscanf	On-line	integers	0.126 s	0.130 s	0.142 s
2	Original	MPI	On-line	characters	7.087 s	6.173 s	6.563 s
3	Original	MPI	On-line	whole file	0.157 s	0.196 s	0.234 s
4	Original	MPI	Off-line	3*int, 2*array	0.189 s	0.182 s	0.195 s
5	Optimized	MPI	On-line	whole file	0.163 s	0.071 s	0.040 s
6	Optimized	fscanf	On-line	integers	0.129 s	0.068 s	0.036 s

Table C.2: I/O time per process for 4 images and 4 models

Parallelization	accumulated number of images + models read with		
	1 process	*2 processes*	*4 processes*
Original	4*1 + 16 + 0	4*2 + 16 + 0	4*4 + 16 + 0
Optimized	4*1 + 16 + 0	4*1 + 8 + 8	4*1 + 4 + 12

Table C.3: *images read + models read + models exchanged*

C.6 Conclusion

One of the computationally intensive image processing problem, *Image matching* which demands the solutions within real time constraints is investigated focusing the attention on MPI File Interoperability especially with ASCII files. Due to the domain specific nature of the problem, the images usually stored in files, differ in formats considerably. This poses an impediment to the efficient implementation of the parallel algorithm despite parallel I/O implementations in MPI-2. As most of the formats can be converted into ASCII file format in many systems, the three strategies namely, *Normal File Reading, Off-line Conversion and Run-time Conversion* for free format integer file reading and writing are implemented on Cray T3E with MPI-2. The modified parallelization presented here produced better results comparing the I/O timings. The important conclusion of the section is that the problem of file format conversion in image processing applications can be efficiently solved with the proper parallelization and MPI parallel I/O operations. In all the images, the accurate positions (to one pixel resolution) of the human beings with the corresponding best model are not only found correctly but also efficiently as the obtained results demonstrate.

Appendix D

Indices of the Images and Models

As presenting the large set of the image indices and model indices will be difficult, only some typical image indices and model indices are mentioned. It may be recalled that for sake of efficiency, mostly they will be preprocessed into coordinate files of relevant corners for Hausdorff methods. However, for chamfering respective edge images must be provided as a part of preprocessing or feature extraction.

D.1 Some typical Image Indices as coded in the program for reading the files

0 : ("/home/piriyaku/muserk/progs/pgm/img/rf6_c.pgm",);
1 : ("/home/piriyaku/muserk/progs/pgm/img/rf7_c.pgm",);
2 : ("/home/piriyaku/muserk/progs/pgm/img/rf8_c.pgm",);
3 : ("/home/piriyaku/muserk/progs/pgm/img/rf9_c.pgm",);
4 : ("/home/piriyaku/muserk/progs/pgm/img/rf0_c.pgm",);
5 : ("/home/piriyaku/muserk/progs/pgm/img/bv0_c.pgm",);
6 : ("/home/piriyaku/muserk/progs/pgm/img/bv1_c.pgm",);
7 : ("/home/piriyaku/muserk/progs/pgm/img/bv2_c.pgm",);
8 : ("/home/piriyaku/muserk/progs/pgm/img/bd1_c.pgm",);
9 : ("/home/piriyaku/muserk/progs/pgm/img/bd2_c.pgm",);
10 : ("/home/piriyaku/muserk/progs/pgm/img/bd3_c.pgm",);
11 : ("/home/piriyaku/muserk/progs/pgm/img/bd4_c.pgm",);
12 : ("/home/piriyaku/muserk/progs/pgm/img/bd5_c.pgm",);
13 : ("/home/piriyaku/muserk/progs/pgm/img/bd6_c.pgm",);

14 : ("/home/piriyaku/muserk/progs/pgm/img/bd7_c.pgm",);
15 : ("/home/piriyaku/muserk/progs/pgm/img/bd8_c.pgm",);
16 : ("/home/piriyaku/muserk/progs/pgm/img/bd9_c.pgm",);
17 : ("/home/piriyaku/muserk/progs/pgm/img/bd10_c.pgm",);
18 : ("/home/piriyaku/muserk/progs/pgm/img/bd11_c.pgm",);
19 : ("/home/piriyaku/muserk/progs/pgm/img/bd12_c.pgm",);
20 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0035_c.pgm" ,);
21 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0036_c.pgm" ,);
22 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0037_c.pgm" ,);
23 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0038_c.pgm" ,);
24 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0039_c.pgm" ,);
25 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0040_c.pgm" ,);
26 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0041_c.pgm" ,);
27 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0042_c.pgm" ,);
28 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0043_c.pgm" ,);
29 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0044_c.pgm" ,);
30 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0045_c.pgm" ,);
31 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0046_c.pgm" ,);
32 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0047_c.pgm" ,);
33 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0048_c.pgm" ,);
34 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0050_c.pgm" ,);
35 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0051_c.pgm" ,);
36 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0052_c.pgm" ,);
37 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0053_c.pgm" ,);
38 : ("/home/piriyaku/muserk/progs/pgm/img/PICT0054_c.pgm" ,);
39 : ("/home/piriyaku/muserk/progs/pgm/img/ut3_c.pgm" ,);
40 : ("/home/piriyaku/muserk/progs/pgm/img/ut7_c.pgm" ,);
41 : ("/home/piriyaku/muserk/progs/pgm/img/ut13_c.pgm" ,);

D.2 Some typical Model Indices as coded in the program for reading the files

0 : ("/home/piriyaku/muserk/progs/pgm/mod/cut7_c.crd",);
1 : ("/home/piriyaku/muserk/progs/pgm/mod/cut8_c.crd",);
2 : ("/home/piriyaku/muserk/progs/pgm/mod/cut81_c.crd",);
3 : ("/home/piriyaku/muserk/progs/pgm/mod/cut9_c.crd",);
4 : ("/home/piriyaku/muserk/progs/pgm/mod/cut91_c.crd",);
5 : ("/home/piriyaku/muserk/progs/pgm/mod/cut7_hs_c.crd",);

6 : ("/home/piriyaku/muserk/progs/pgm/mod/cut8_hs_c.crd",);
7 : ("/home/piriyaku/muserk/progs/pgm/mod/cut81_hs_c.crd",);
8 : ("/home/piriyaku/muserk/progs/pgm/mod/cut9_hs_c.crd",);
9 : ("/home/piriyaku/muserk/progs/pgm/mod/cut91_hs_c.crd",);
10 : ("/home/piriyaku/muserk/progs/pgm/mod/cut7_ds_c.crd",);
11 : ("/home/piriyaku/muserk/progs/pgm/mod/cut8_ds_c.crd",);
12 : ("/home/piriyaku/muserk/progs/pgm/mod/cut81_ds_c.crd",);
13 : ("/home/piriyaku/muserk/progs/pgm/mod/cut9_ds_c.crd",);
14 : ("/home/piriyaku/muserk/progs/pgm/mod/cut91_ds_c.crd",);
15 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd1_c.crd",);
16 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd2_1_c.crd",);
17 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd2_2_c.crd",);
18 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd2_3_c.crd",);
19 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd2_4_c.crd",);
20 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd3_1_c.crd",);
21 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd4_1_c.crd",);
22 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd4_2_c.crd",);
23 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd5_1_c.crd",);
24 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd5_2_c.crd",);
25 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd5_3_c.crd",);
26 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd5_4_c.crd",);
27 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd5_5_c.crd",);
28 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd6_1_c.crd",);
29 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd6_2_c.crd",);
30 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd7_1_c.crd",);
31 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd8_1_c.crd",);
32 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd9_1_c.crd",);
33 : ("/home/piriyaku/muserk/progs/pgm/mod/mbd9_2_c.crd",);
34 : ("/home/piriyaku/muserk/progs/pgm/mod/ants1_c.crd",);
35 : ("/home/piriyaku/muserk/progs/pgm/mod/ants2_c.crd",);
36 : ("/home/piriyaku/muserk/progs/pgm/mod/ants3_c.crd",);
37 : ("/home/piriyaku/muserk/progs/pgm/mod/ants4_c.crd",);
38 : ("/home/piriyaku/muserk/progs/pgm/mod/ants5_c.crd",);
39 : ("/home/piriyaku/muserk/progs/pgm/mod/ants6_c.crd",);
40 : ("/home/piriyaku/muserk/progs/pgm/mod/ants7_c.crd",);
41 : ("/home/piriyaku/muserk/progs/pgm/mod/ants8_c.crd",);
42 : ("/home/piriyaku/muserk/progs/pgm/mod/ants9_c.crd",);
43 : ("/home/piriyaku/muserk/progs/pgm/mod/ants10_c.crd",);
44 : ("/home/piriyaku/muserk/progs/pgm/mod/ants11_c.crd",);
45 : ("/home/piriyaku/muserk/progs/pgm/mod/ants12_c.crd",);

46 : ("/home/piriyaku/muserk/progs/pgm/mod/ants13_c.crd",);
47 : ("/home/piriyaku/muserk/progs/pgm/mod/ants14_c.crd",);
48 : ("/home/piriyaku/muserk/progs/pgm/mod/ants15_c.crd",);
49 : ("/home/piriyaku/muserk/progs/pgm/mod/ants16_c.crd",);
50 : ("/home/piriyaku/muserk/progs/pgm/mod/olga1_c.crd",);
51 : ("/home/piriyaku/muserk/progs/pgm/mod/olga2_c.crd",);
52 : ("/home/piriyaku/muserk/progs/pgm/mod/olga3_c.crd",);
53 : ("/home/piriyaku/muserk/progs/pgm/mod/olga4_c.crd",);
54 : ("/home/piriyaku/muserk/progs/pgm/mod/olga5_c.crd",);
55 : ("/home/piriyaku/muserk/progs/pgm/mod/olga6_c.crd",);
56 : ("/home/piriyaku/muserk/progs/pgm/mod/olga7_c.crd",);
57 : ("/home/piriyaku/muserk/progs/pgm/mod/olga8_c.crd",);
58 : ("/home/piriyaku/muserk/progs/pgm/mod/olga9_c.crd",);
59 : ("/home/piriyaku/muserk/progs/pgm/mod/olga10_c.crd",);
60 : ("/home/piriyaku/muserk/progs/pgm/mod/olga11_c.crd",);
61 : ("/home/piriyaku/muserk/progs/pgm/mod/olga12_c.crd",);
62 : ("/home/piriyaku/muserk/progs/pgm/mod/olga13_c.crd",);
63 : ("/home/piriyaku/muserk/progs/pgm/mod/ut3_1_c.crd",);
64 : ("/home/piriyaku/muserk/progs/pgm/mod/ut7_1_c.crd",);
65 : ("/home/piriyaku/muserk/progs/pgm/mod/ut13_1_c.crd",);

Appendix E

Scheduling of Tasks onto Multiprocessors for Optimal Solutions

E.1 Problem Formulation

A parallel program (algorithm) is represented as a weighted, directed acyclic graph, $G_t = \{V_t, E_t\}$, where $V_t = \{ v_i : \text{i=1,2,...,n} \}$ the set of vertices (tasks) with associated service demand s_i, and $E_t = \{< v_i, v_j > : \text{i,j} = 1,2,...,\text{n}, \text{i} \neq \text{j} \}$ the set of directed edges with associated intertask communication (data) from task T_i to task T_j, imposing the partial order that task T_j can be executed only after the execution of task T_i. As an example, a task graph with five tasks is given in Fig. E.1. Here, the numbers beside the nodes represent the service demands (s_i) of the tasks in the corresponding nodes and the numbers beside the edges represent the intertask communication (c_{ij}) between the corresponding tasks in the direction of the edge concerned. Here, task T_0 is the start task as it does not have any predecessor. The end task is task T_4 as it does not have any successor.

The multiprocessor system onto which tasks are scheduled, is assumed to be either homogeneous (all processors have the same service rate, memory capacity, link capacities, etc) or heterogeneous (the processors may differ in service rates). A processor is assumed to perform both computation and interprocessor communication at the same time like an INMOS transputer. The multiprocessor system is represented as a weighted undirected graph, $G_p = \{V_p, E_p \}$, where $V_p = \{ v_q : \text{q} = 1,2,...,\text{m}\}$ set of processors with associated service rates μ_q and $E_p = \{(\text{p,q}) : \text{p,q} = 1,2,...,\text{m}, \text{p} \neq \text{q}\}$ set of

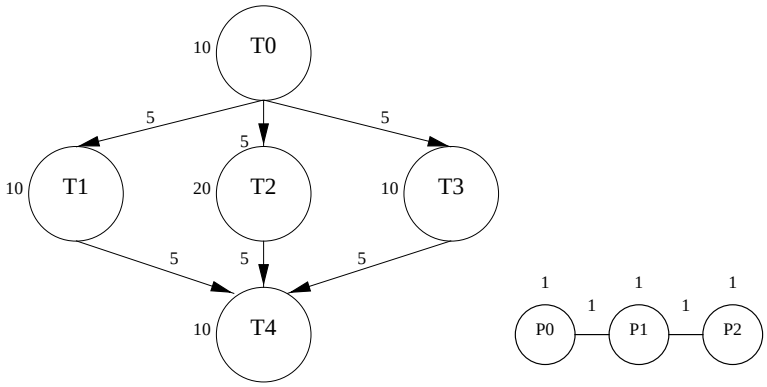


Figure E.1: Task Graph and Processor Graph

links with associated link capacities L_{pq}. The data communication between a pair of processors follows the shortest path. The shortest path between any two processors is the sequence of links (edges) in which the data reaches the destination processor in the shortest time. Here, the distance between the processors directly connected is the inverse of the link capacity of the link between the processors. In other words, the shortest path between any two processors is the sequence of links in which the total distance is minimal. The execution of a task on a processor is nonpreemptive. As an example, a processor graph with three linearly connected processors is given in Fig.E.1. Here the numbers beside the processors represent the service rates of the processors and the numbers beside the links represent the capacity of the link concerned.

The problem of scheduling parallel (concurrent) tasks onto multiprocessors can be stated as to find an optimal schedule (minimum schedule length), by allocating each task to one processor and executing them in such a way to satisfy the precedence constraints among the tasks. As the computational times of tasks are known a priori with their interdependence and no preemption is allowed, the tasks can be scheduled efficiently at compile-time, and the overhead associated with dynamic scheduling can be eliminated.

E.2 The New A^* Based Algorithm

E.2.1 General A^* Algorithm

The general A^* algorithm used in most of the artificial intelligence problems is given in [52]. In A^* algorithm, the state space graph is a tree called search

tree. Each node in the tree corresponds to the assignment of a particular task to a specific processor. All the internal nodes in the tree correspond to partial (or incomplete) schedule and all external (leaf) nodes in the tree, correspond to either pruned node or complete task schedule. The problem here is to find the goal node, a leaf node corresponding to the optimal schedule. Associated with a node v in the search tree is a cost function f(v) = g(v) + h^*(v), which is an underestimate for the minimum cost of an assignment, given that it includes the partial schedule. The function g(v) is the cost of the path from the root to v and the function h^*(v) is a lower bound estimation of the minimum cost function h(v), from the node v to a leaf node which corresponds to an optimal task assignment in the subtree rooted at node v.

The algorithm begins by creating a root node representing a null state (no task scheduled) and placing it in the unexpanded list, which is initially empty. Let v be a node in the unexpanded list with the minimum value of cost function f(v) i.e., f(v) $\leq$ f(u) for all other nodes u in the unexpanded list. Also, let T_i be the task scheduled at node v. If v is not a goal node, then it is removed from the unexpanded list and expanded by generating all possible assignment of ready tasks (tasks whose predecessors have already been assigned) without violating the precedence constraints. The algorithm computes the evaluation function f(u) for each node u and inserts u in the unexpanded list in the order of increasing value of the node evaluation function. The algorithm terminates when the node to be expanded happens to be the goal node.

E.2.2 New Techniques for Reducing Space and Time

The A^* algorithm described above, can be used to solve the problem of multiprocessor task scheduling. But the main impediment with the A^* algorithm is the requirement of large memory space and computational time. So, to reduce the space and time requirements of A^* algorithm, three new techniques apart from two more effective techniques are developed namely,

1. *Processor isomorphism*
2. *Task isomorphism*
3. *Node isomorphism*
4. *Upper bound*
5. *Lower bound theory.*

Before explaining the new A^* algorithm completely, these five techniques which reduce the execution time (computational time) and the memory requirements to arrive at an optimal solution are explained.

Processor Isomorphism

First, the processor isomorphism is defined. Two processors P_1 and P_2 are isomorphic iff

1. all their physical characteristics viz. processing speed, memory capacity, etc are the same.
2. if there exists r number of k distant neighbor processors for P_1, only the same r number of k distant neighbor processors should be there for P_2. Moreover, pairwise those neighbor processors of P_1 and P_2 should be isomorphic.

Take for example a linear chain of processors shown in Fig.E.1 The isomorphic groups of processors are (P_0, P_2), and P_1. P_0 is not isomorphic to P_1 because P_0 has a neighbor at a distance of 2, which P_1 does not have. P_2 is not isomorphic to P_1 because P_1 has two neighbors at a distance of 1 whereas P_2 has only one neighbor at a distance of 1. As P_1 has two neighbors at a distance of 1, and no other processor has such two neighbors, it forms its own group of isomorphic processors. Interestingly, in a homogeneous hypercube multiprocessor system of any dimension (n), all the processors are isomorphic to each other forming only one group. Similarly, in a completely connected homogeneous multiprocessor system, all the processors form only one single isomorphic group. In the same vein, a ring of homogeneous multiprocessor system also forms only one isomorphic group comprising of all the processors. So, when a start task is allocated to a processor, it will be allocated to all possible processors in the A^* algorithm. Now in the new A^* based algorithm, it is sufficient if such allocation for the start task is made with only one member in each of the isomorphic groups as this will not affect optimal solution due to the properties of isomorphic processors. The reduction this technique gives for various architectures for the best cases is given in Table E.1.

Task Isomorphism

Two tasks T_i and T_j are said to be isomorphic iff

1. The completion times of T_i and T_j on a processor P_k where $\mu_{P_k} = 1$ are the same, i.e., $s_i = s_j$.

Architecture	*No. of isomorphic groups*	*Reduction*
Linear array	n/2	n/2
Ring	1	n
Completely connected	1	n
Hypercube (n-dim)	1	2^n

Table E.1: Reduction Factor due to Isomorphic Groups

2. The static level λ_i of a task T_i in a task graph G_t is defined to be the sum of the completion time of the task T_i on the fastest processor and the maximum of the static levels of all its children. Then, the static levels of T_i and T_j should be the same, i.e., $\lambda_i = \lambda_j$.

3. If τ_i is the set of parent tasks of T_i, i.e., $\forall T_k \in \tau_i, < T_k, T_i > \in E_t$ and if τ_j is the set of parent t asks of T_j, i.e., $\forall T_l \in \tau_j, < T_l, T_j > \in E_t$, then $\tau_i = \tau_j$.

4. $\forall\ T_k \in \tau_i$, $c_{ki} = c_{kj}$.

5. If η_i is the set of child tasks of T_i, i.e.,$\forall T_k \in \eta_i, < T_i, T_k > \in E_t$ and if η_j is the set of child tasks of T_j, i.e.,$\forall T_l \in \eta_j, < T_j, T_l > \in E_t$, then $\eta_i = \eta_j$.

6. $\forall T_k \in \eta_i$, $c_{ik} = c_{jk}$.

Consider the task graph given in Fig.E.1. The tasks T_1 and T_3 are isomorphic tasks. The isomorphic groups of tasks here are T_0, (T_1 and T_3), T_2 and T_4. Whenever there are ready tasks, it is sufficient if one task from each of the task isomorphic groups is assigned, as assigning all the tasks will only lead to a futile attempt in generating the same optimal solution. The major meritorious point is that task isomorphism is calculated only once and has a time complexity of $O(n^3)$. It is also easy to verify that task isomorphism is transitive like processor isomorphism.

Node Isomorphism

Two nodes N_i and N_j in the state space are said to be isomorphic iff

1. Let T_i and T_j be the last tasks assigned in the nodes N_i and N_j, respectively. Then T_i and T_j should be isomorphic tasks.

2. Let the last tasks T_i and T_j in the respective nodes N_i and N_j be assigned to processors P_i and P_j, respectively. Then P_i and P_j should be isomorphic processors.

3. Let $\zeta_{kp}^{N_i}$ be the completion time of task T_k on processor P_p in the node N_i. Then, $\zeta_{T_iP_i}^{N_i}$ should be equal to $\zeta_{T_jP_j}^{N_j}$.

4. Let Γ_{Ni} be the set of tasks assigned in the node N_i. For each $T_k \in \Gamma_{N_i} \neq T_i$ assigned to the processor P_k in the node N_i, then task T_k should be assigned only to the respective P_k in the node N_j.

5. $\forall T_k \in \Gamma_{N_i}$, $\zeta_{T_kP_k}^{N_i}$ should be equal to $\zeta_{T_kP_k}^{N_j}$.

Considering the task graph and processor graph in Fig.E.1, assume that task T_0 is assigned to processor P_1. Now tasks T_1, T_2 and T_3 are ready. Then, for example, any one of the following schedules will itself guarantee an optimal solution, i.e., T_1 on P_1 or T_1 on P_3 or T_3 on P_1 or T_3 on P_3 as $\zeta_{T_1P_1}^{N_i} = \zeta_{T_1P_3}^{N_{i+1}} = \zeta_{T_3P_1}^{N_{i+2}} = \zeta_{T_3P_3}^{N_{i+3}} = 25$ units. So, when a node is isomorphic to the already existing node, then there is no need for adding the node in the unexpanded list as in the case of duplication of nodes.

Upper Bound

The logic behind the technique is how to reduce the number of nodes by finding them to be futile at the early stage itself. This will be possible only when some better solution is at hand. Hence, the heuristic algorithm produces a schedule which is taken as Upper Bound (UB). In the case, the same heuristics is used as in [126] for the sake of comparison. For the example in Fig.E.1, the heuristic algorithm produced a schedule with schedule length of 45 units, which is set as UB. A node whose f(x) value is greater than UB, need not to be included for expansion as a better solution is already available . This in turn reduces number of nodes in the subsequent levels. For the example, graphs given in Fig.E.1, at the node N_{15}, the f(N_{15}) = 55 which is more than UB, hence the node N_{15} is not added in the list.

Lower Bound Theory

This is based on the static levels as defined earlier. It is well known and obvious to prove that no optimal schedule can be lesser than the static level of the start task. It means that one can not parallelize a serial execution. Such serial execution only contributes to Lower Bound (LB). By the lower

bound theory, whenever in a node all tasks have been assigned and the f(x) = LB, the algorithm can be stopped at once. This is very much applicable in the tracking problem discussed in [126]. When there are multiple start tasks, the start task having the maximum static level is set to LB.

In the case (Fig.E.1), there is only one task T_0 as the start task with static level of 40 units which is set to LB. Fortunately, the algorithm finds the solution at the node N_{31} and then terminates producing the optimal solution. The important point to note in the lower bound theory is that no schedule better than LB can be found, irrespective of the number of processors and their interconnection structure.

E.2.3 The New Algorithm for Optimal Task Scheduling

The new A^* algorithm is explained succinctly as follows. First using a heuristic algorithm, find a schedule and set the schedule length to UB. Find the static level of each task and set LB as the static level of the start task. The basic idea behind the algorithm is that given a node (initially empty), find all the ready tasks. Assign one ready task from each task isomorphic group in every processor excepting for the start task or for the node isomorphism. In case of the start task, assign it to only one member from all the isomorphic groups of processors. With the above explanation, trying all isomorphic tasks which are ready is futile and one is sufficient to guarantee an optimal solution. In the same vein, isomorphic nodes are also deleted without impeding an optimal solution as the property ensures optimal solution. Compute the value of the heuristic evaluation function f(c) for each of these nodes. If the node does not occur earlier and $f(c) < UB$, add the node in the search tree as child of the recently expanded node. Check whether the node is a goal node for reaching a solution and if f(c)=LB, then also stop by producing the optimal solution. If the node to be expanded is a goal node, output the schedule as optimal schedule and stop. Otherwise, repeat the process until no more node could be expanded. Now the new A^* algorithm using the notations as specified earlier is presented. The Optimal Schedule for example given in Fig.E.1 is shown in Fig.E.2 with the timings in Table E.2. It is very evident that the new techniques produced the results in lesser time.

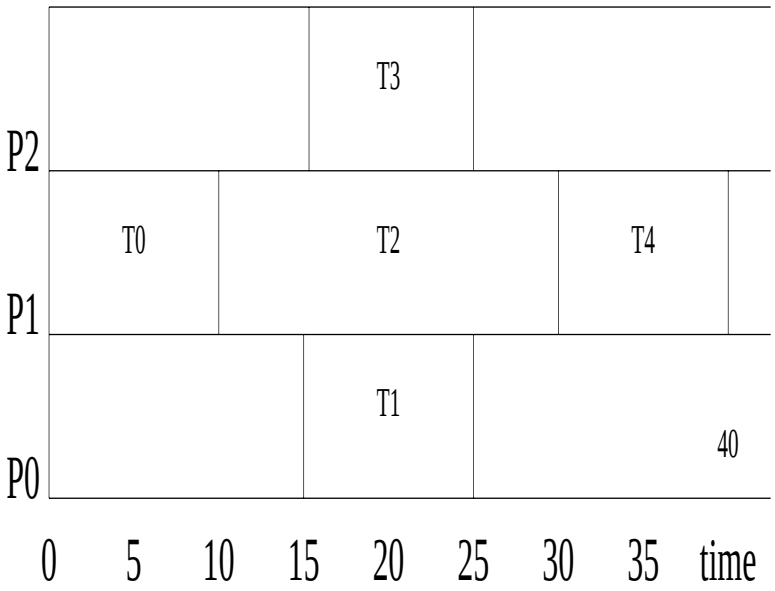


Figure E.2: The Optimal Schedule for Fig.E.1

Algorithm	*No. of nodes generated*	*CPU time in sec*
A^* with h(x)=0	609	0.4
Previous A^*	408	0.9
The new A^*	31	0.1

Table E.2: Comparison of Previous A^* with New A^* Algorithms

Appendix F

Scheduling Iterative Data-Flow Program Model

A nonterminating, iterative, data-flow program is represented as a weighted, directed acyclic graph [51], $G_t = \{V_t, E_t\}$, where V_t = { v_i : i=1,2,...,n } the set of vertices (tasks) with associated service demand s_i, and E_t = $\{< v_i, v_j >$: i,j = 1,2,...,n, i $\neq$ j } the set of directed edges with associated intertask communication (data) from task T_i to task T_j, imposing the partial order that task T_j can be executed only after the execution of task T_i as in Fig.4.16. The main difference between general DFGs(Data-Flow Graphs) and the signal processing DFGs is the associated delay elements (registers) in the directed edges [82]. An edge without a register represents precedence between tasks within iteration. If an edge has n registers, it describes the precedence between tasks of different (i,n+i) iterations which differ by n iterations. Scheduling Precedence Graphs in Systems with Interprocessor Communication times is discussed in [127].

F.1 The Effects of IPC on Periodic Multiprocessor Schedule

The data-flow programs can be scheduled onto multiprocessors in overlapped or non-overlapped manner with two other methods viz. fully-static and cyclo-static [82]. A multiprocessor schedule is said to be non-overlapped if the execution of the $(n+1)^{th}$ iteration begins only after the completion of all the tasks of the $(n)^{th}$ iteration, otherwise it is overlapped. A periodic schedule is said to be fully-static, if all the iterations of some task are scheduled on the same processor. A periodic schedule is said to be cyclo-static, if

the task T_i is scheduled in processor V_p at time t in the n^{th} iteration, then in the $(n+1)^{th}$ iteration the task T_i is scheduled in processor $V_{(p+K)modulom}$ at the time (t+T), where T is the time displacement (iteration period) and m is the total number of processors and K is the processor displacement.

Unlike in [82], the multiprocessor system onto which tasks are scheduled is not assumed to be completely connected which in practice may not always hold. It is assumed to be either homogeneous (all processors have the same service rate, memory capacity, link capacities, etc) or heterogeneous (the processors may differ in service rates). A processor is assumed to perform both computation and interprocessor communication at the same time like an INMOS transputer. The multiprocessor system is represented as a weighted undirected graph, $G_p = \{V_p, E_p\}$, where $V_p = \{ v_q : q = 1,2,...,m\}$ set of processors with associated service rates μ_q and $E_p = \{ (p,q) : p,q = 1,2,...,m, p \neq q\}$ set of links with associated link capacities L_{pq}. The data communication between a pair of processors follows the shortest path as defined in [51]. The execution of a task on a processor is nonpreemptive.

Here, the iteration period is considered as the parameter as it plays a vital role in the multiprocessor periodic schedules especially when IPC is included. An important point to note is that if iteration period is considered as defined earlier, it will not suffice to account for inter-iteration precedences more specifically when IPC is non-negligible. Hence, when IPC is included in the multiprocessor scheduling, the average iteration period is taken into consideration as it represents the steady state in the nonterminating programs. The DFG considered here is same as in Fig.4.16 with the processor graph.

F.2 The New A^* Algorithm for Optimal Scheduling of DFGs

F.2.1 New Techniques for Reducing Space and Time

The A^* algorithm described [52], [51], can be used to solve the problem of multiprocessor task scheduling. But the main impediment with the A^* algorithm is the requirement of large memory space and computational time. So, to reduce the space and time requirements of A^* algorithm, a new technique *Branch Join Path isomorphism* is developed.

Branch Join Path Isomorphism

This isomorphism is well pronounced in DFGs especially in digital signal processing applications. However, the isomorphism is only for homogeneous multiprocessor systems. The branch join path (BJP) isomorphism is defined as follows,

1. In a DFG, consider a task having 2 children. say c_1 and d_1. There can be more than 2 children but consider them in pairs.
2. Now, let c_1 has only one child c_2 and c_2 has only one child c_3 like this till some c_x for some positive $x > 2$.
3. Similarly, let d_1 has only one child d_2 and d_2 has only one child d_3 like this till some d_y for some positive $y > 2$.
4. Let the tasks c_x and d_y be same.
5. Let sum-c be the sum of the execution times of c tasks and sum-d be for d tasks.
6. If sum-c = sum-d, then it is sufficient that either the search is tried with c tasks or with d tasks, preferable with c tasks when $x < y$ or vice versa.

In the case for Fig.4.16 with m = 2 after 2-unfolding, there are task $T_0(A_1)$ and task $T_4(A_2)$ as the start tasks with static level of 4 units which can be considered as children of a fictitious node. It is logical to consider only in the case when the tasks are start tasks. Similarly, the tasks $T_3(D_1)$ and $T_7(D_2)$ are the end tasks which can be considered as the parents of another fictitious task. It is also logical to consider only in the case when the tasks are end tasks.

Now, BJP isomorphism exists as sum-c = sum-d = 4 and all other conditions also are fulfilled. Hence it is sufficient to try with one branch itself to get the optimal schedule. In fact, when x = y = 2, this reduces to node isomorphism. Hence, the original DFG itself can be modified so that there will be less number of tasks to schedule at the same time optimality is maintained. The only modification required is to merge all nodes c_1, c_2, ..., c_{x-1} into one node and similarly for d tasks. It is interesting to note that even when there is only one branch, one can merge these type of tasks forming a chain into a single task thereby not only reducing the number of tasks to be scheduled but also some of the isomorphisms previously mentioned to exhibit voluntarily. This portion of algorithm is of $O(n^2)$ complexity only.

Arch	Standard A^*		Isomorphisms		BJP Isomorphism	
	Nodes	CPU Time	Nodes	CPU Time	Nodes	CPU Time
l2	747	5.5	22	0.12	3	0.11
c3	2896	38.15	32	0.13	4	0.11
h2	11464	346.7	42	0.14	5	0.12
l4	11464	346.6	44	0.14	6	0.12

Table F.1: Comparison of Previous A^* with New A^* Algorithms for DFG

F.3 Performance Evaluation

For the DFG in Fig.4.16 with the schematic diagram of linearly connected *ln* with n processors along with other sets of different processor architectures such as completely connected with n processors *cn* and hypercube *hn* of dimension n, the number of nodes generated by A^* Algorithms are compared in Tab.F.1. It is very clear that BJP algorithm performs better than the other examples.

Appendix G

List of Publications

1. With Paul Levi, "A Novel Isomorphism Based on Nearest Neighbours for Efficient Graph Matching Algorithm", The Seventh International Conference on Control, Automation, Robotic and Vision, ICARCV 2002, December, 2002, Singapore.

2. With Paul Levi, "A New Approach to Exploiting Parallelism in Ant Colony Optimization", IEEE International Symposium on Micromechatronics and Human Science(MHS), October, 2002, Nagoya, Japan.

3. With Paul Levi and Rolf Rabenseifner, "Enhanced File Interoperability with Parallel MPI File I/O in Image Processing",Proceedings of 9th EuroPVMMPI-2002, Linz, Austria.

4. With Paul Levi, "On the symmetries of regular repeated objects using graph theory based novel isomorphism", The 5th International Conference on PATTERN RECOGNITION and IMAGE ANALYSIS: NEW INFORMATION TECHNOLOGIES PRIA-5-2000, October, 2000, Samara, The Russian Federation.

5. With Paul Levi, N.Manickam, R.Jayaganthan, "A Domain Knowledge Based Mutagenetic Algorithm for Multiprocessor Scheduling", The 2000 International Conference on Artificial Intelligence (IC-AI'2000), June, 2000, Las Vegas,USA.

6. With Paul Levi, "An Efficient A* based Algorithm for Optimal Graph Matching applied to Computer Vision", Pattern Recognition and Image Analysis, pp. 708-712, Vol.9, No. 4, Oct-Dec 1999.

7. With Paul Levi, "An Improvised A* Algorithm for Mobile Robots to Find the Optimal Path in an Unknown Environment with Minimized Search efforts", IEA/AIE-99, 1999, Cairo, Egypt.

8. With Paul Levi, R.Jayaganthan, "A Novel Algorithm for Task Mapping of Computer Vision Tasks using Thermodynamical Free Energy Approach ", Eleventh IASTED International Conference on Parallel and Distributed Computing and Systems, November, 1999, MIT, Boston, USA.

9. With Paul Levi, C. Siva Ram Murthy, "Optimal Schduling of Iterative Data-Flow Programs onto Multiprocessors with Non-negligible Interprocessor Communication", HPCN'99 , 1999, Amsterdam, The Netherlands.

10. With Paul Levi, "An Efficient Parallel Algorithm for Optimum Path Finding in Fixed Industry Oriented Scenarios by Mobile Robots", AMS'98, Karlsruhe, Germany.

11. With Paul Levi, C. Siva Ram Murthy, "A New A^* Based Optimal Task Scheduling in Heterogeneous Multiprocessor Systems Applied to Computer Vision", HPCN'98 , 1998, Amsterdam, The Netherlands.

Appendix H

Biography

Born on 20th May 1966 in Paramakudi, Tamil Nadu, India. Graduated in Physics from American College, Madurai in 1986 with the best student award. Received Goldmedal in M.Sc Computer Science from Madurai Kamraj University in 1988. Set a record in M.S Computer Science and Engineering in Indian Institute of Technology Madras in Chennai in 1994.

Taught in the department of Computer Science, Madurai Kamaraj University, India from October 1988 to September 1989 and from October 1989 to June 2001 in the department of Computer Science, Pondicherry University, India. Served as Teaching Assistent cum research scholar in Institute of Parallel and Distributed Systems (IPVS), department of Computer Science, University of Stuttgart, Germany from October 1999 till the submission of the thesis.

Received the prestigious DAAD Scholarship, Germany from June 1996 to September 1999.

Bibliography

[1] R.C.Gonzalez and P.Wintz, *Digital Image Processing*, Addison-Wesley, 1987.

[2] D.H.Ballard and C.M.Brown, *Computer Vision*, Englewood Cliffs, NJ, Prentice-Hall, 1982.

[3] W.F.Schreiber, *Fundamentals of Electronic Imaging Systems : Some Aspects of Image Processing*, Springer-Verlag, 1993.

[4] S.Ullman, *High-level Vision : Object Recognition and Visual Cognition*, MIT Press, 1996.

[5] T.B.Moeslund, "Computer Vision-based Human Motion Capture: A Survey," Technical Report LIA 99-02, University of Aalborg, 1999.

[6] W.Vaughan Jr. and S.L.Greene, "Pigeon visual memory capacity," Journal of Experimental Psychology: Animal Behaviour Processes, vol. 10, pp. 256-271, 1984.

[7] J.Gould, "How Bees remember Flower Shapes," Science, vol. 227, pp. 1492-1494, 1985.

[8] J.L. Denebourg and S. Goss, "Collective patterns and decision-making," Ethology, Ecology and Evolution, vol. 1, pp. 295-311, 1989.

[9] J.K.Aggarwal and Q.Cai, "Human Motion Analysis: A review," Workshop on Motion of Non-Rigid and Articulated Objects, Puerto Rico, USA, 1997.

[10] D.M.Gavrila, "The Visual Analysis of Human Movement: A Survey", Computer Vision and Image Processing, vol. 73, no. 1, pp. 82-98, 1999.

[11] T.B.Moeslund and E.Granum, "A Survey of Computer Vision-based Human Motion Capture," Computer Vision and Image Understanding, vol. 81, no. 3, pp. 231-268, March, 2001.

[12] D. Huttenlocher, G. Klanderman and W. Rucklidge., "Comparing images using Hausdorff distance", Transaction on Pattern Analysis and Machine Intelligence, vol. 15, no. 9, pp. 850-863, September, 1993.

[13] M.Isard, J.MacCormick, "BraMBLe: A Bayesian Multiple-Blob Tracker", pp. 34-41, ICCV 2001, Vancouver, Canada.

[14] J. Sherrah, S. Gong, "Continuous Global Evidence-Based Bayesian Modality Fusion for Simultaneous Tracking of Multiple Objects," pp. 42-49, ICCV 2001, Vancouver, Canada.

[15] D.M.Gavrila and J.Giebel, "Virtual Sample Generation For Template-Based Shape Matching," IEEE Conference on Computer Vision And Pattern Recognition, pp.676-681, Kauai, USA, 2001.

[16] N. T. Siebel and S. J. Maybank, "Real-time tracking of Pedestrians and vehicles", IEEE International workshop PETS'2001.

[17] D.M Gavrila and V.Philomin, "Real-Time Object Detection For "SMART" Vehicles," IEEE International Conference on Computer Vision, pp.87-93, Kerkyra, 1999.

[18] U.Franke, D.Gavrila, S.Gorzig and F.Lindner, "Autonomous Driving approaches Downtown," IEEE Intelligent Systems, vol.13, no.6, 1999.

[19] Dong-Gyu Sim and Rae-Hong Park, "Two-Dimensional Object Alignment Based on the Robust Oriented Hausdorff Similarity Measure," IEEE Transactions on Image Processing, vol.10, no.3, March, 2001.

[20] G.Rigoll, B.Winterstein and S.Mueller, "Robust Person Tracking in Real Scenarios with Non-Stationary Background Using A Statistical Computer Vision Approach," IEEE International Workshop on Visual Surveillance, pp.41-47, Fort collins, USA, 1999.

[21] A.M.Baumberg and D.C. Hogg, "Learning Flexible Models from Image Sequences", Technical Report 93.36, University of Leed, October, 1993.

[22] A.M.Baumberg and D.C.Hogg, "An Efficient Method for Contour Tracking using Active Shape Models", Technical Report 94.11, University of Leed, April, 1994.

[23] A.M.Baumberg and D.C.Hogg, "Learning Spatiotemporal Models From Training Examples", Technical Report 95.9, University of Leed, March, 1995.

[24] Jessica K.Hodgins, James F.O'Brien and Jack Tumblin, "Perception of Human Motion With Different Geometric Models," IEEE Transactions on Visualization and Computer Graphics, pp.307-316, vol.4, no.4, October-December, 1998.

[25] Ismail Haritaoglu and Myron Flickner, "Detection and Tracking of Shopping Groups in Stores," pp.431-438, IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[26] Aaron F.Bobick and Amos Y.Johnson, "Gait Recognition Using Static, Activity-Specific Parameters," pp.423-430, IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[27] G.Shakhnarovich, L.Lee and T.Darell, "Integrated Face and Gait Recognition from Multiple Views," pp.439-446, IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[28] Kentaro Toyama and Andrew Blake, "Probabilistic Tracking in a metric Space," Proceedings of International Conference on Computer Vision (ICCV), pp. 50-57(II), 2001.

[29] Pedro.F.Felzenszwalb, "Learning Models for Object Recognition," pp.1056-1063, IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[30] Q.Cai and J.K.Aggarwal, "Tracking Human Motion in Structured Environments Using a Distributed-Camera System," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.21, no. 11, pp. 1241-1247, November, 1999.

[31] I.Kakadiaris and D.Metaxas, "Model-based estimation of 3D Human Motion," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 12, pp. 1453-1459, December, 2000.

[32] Y.Ricquebourg and P. Bouthemy, "Real-Time Tracking of Moving Persons by exploiting Spatio-temporal Image slices," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 797-808, August, 2000.

[33] M.K.Leung and Y-H.Yang, "First Sight: A Human Body Outline Labeling System," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 4, pp. 359-377, April, 2000.

[34] I.Hariaoglu, D.Harwood and L.S.Davis, "W^4: Real-time Surveillance of People and their Activities," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 809-830, August, 2000.

[35] N.M.Oliver, B.Rosario and A.P.Pentland, "A Bayesian Computer Vision System for Modelling Human Interactions," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 830-843, August, 2000.

[36] Aaron F.Bobick and James W.Davis., "The Recognition of Human Movement using Temporal Templates", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.23, no.3, pp.257-267, March, 2001.

[37] M.A.Eshera and K-S.Fu, "An Image Understanding system Using Attributed Symbolic Representation and Inexact Graph-Matching," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 5, pp. 604-617, August, 2000.

[38] B.T.Messmer and H.Bunke, "A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 5, pp. 493-503, May, 1998.

[39] H.Bunke, "Error Correcting Graph Matching: On the Influence of the Underlying Cost Function," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 9, pp. 917-922, September, 1999.

[40] L.G. Shapiro and R.M. Haralick, "Structural Descriptions and Inexact matching," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 3, pp. 504-519, September, 1981.

[41] S.H.Unger, "GIT - A Heuristic Program for Testing Pairs of Directed Line Graphs for Isomorphism," Communications of Association for Computing Machinery, vol. 7, no. 1, pp. 26-34, January, 1964.

[42] D.G.Corneil and C.C.Gotleib, "An Efficient Algorithm for Graph Isomorphism," Journal of Association for Computing Machinery, vol. 17, no. 1, pp. 51-64, January, 1970.

[43] A.T.Berztiss, "A Backtrack Procedure for Isomorphism of Directed Graphs," Journal of Association of Computing Machinery, vol. 20, no. 3, pp. 365-377, July, 1973.

[44] D.C.Schmidt, "A Fast Backtracking Algorithm to Test Directed Graphs for Isomorphism Using Distance Matrices,"Journal of Association of Computing Machinery, vol. 23, no. 3, pp. 433-445, July, 1976.

[45] J.R.Ullmann, "An Algorithm for Subgraph Isomorphism," Journal of Association for Computing Machinery, vol. 23, no. 1, pp. 31-42, Jan. 1976.

[46] E.Lawler and D.Wood, "Branch and Bound Methods: A Survey," Operations Research, vol. 14, pp. 699-719, July-August, 1966.

[47] W.J.Christmas, J.Kittler and M.Petrou, "Structural Matching in Computer Vision using Probabilistic Relaxations," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 8, pp. 749-764, August, 1995.

[48] S.Gold and A.Rangarajan, "A Graduated Assignment Algorithm for Graph Matching," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no 4, pp.377-388, April, 1996.

[49] R.Allen, L.Cinque, S.Tanimoto, L.Shapiro and D.Yasuda, "A Parallel Algorithm for Graph Matching and its Maspar Implementation," IEEE Transactions on Parallel and Distributed Systems, vol. 8, no. 5, pp. 490-500, May, 1997.

[50] X.Y.Jiang and H.Bunke, " Exploration of Object Symmetries in Computer Vision and Robotics", *Modelling and Planning for sensor based Intelligent Robot System*, Dagstuhl, October, 1994.

[51] D.A.L.Piriyakumar, C.S.R. Murthy and P.Levi, "A new A^* Based Optimal Task Scheduling in Heterogeneous Multiprocessor Systems Applied to Computer Vision," Proceedings of International Conference on High-Performance Computing and Networking, HPCN'98, Amsterdam, April, 1998.

[52] N.J.Nilson, *Principles of Artificial Intelligence*, Palo Alto, Calif, Tiaga Publications, 1980.

[53] D.A.L.Piriyakumar and P.Levi, "An Efficient A* based Algorithm for Optimal Graph Matching applied to Computer Vision", Pattern Recognition and Image Analysis, pp. 708-712, Vol.9, No. 4, Oct-Dec 1999.

[54] D.A.L.Piriyakumar, and Paul Levi, "On the Symmetries of Regular Repeated Objects Using Graph Theory Based Novel Isomorphism," The 5 th International Conference on PATTERN RECOGNITION and IMAGE ANALYSIS: NEW INFORMATION TECHNOLOGIES (PRIA-5-2000), Samara, The Russian Federation, 2000.

[55] S.Umeyama, "An Eigendecomposition approach to weighted graph matching problems," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 5, pp.695-703, September, 1988.

[56] B.Luo and E.R.Hancock, "Structural graph matching using EM algorithm and single value decomposition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 10, pp. 1120-1136, October, 2001.

[57] A.V.Aho, J.E.Hopcroft and J.D.Ullman, *The Design and Analysis of Computer Algorithms*, Addison-wesley, 1975.

[58] Padma Subramaniam, *Bharathakkalai Kotpadu*, Vanathi Publishers, Fourth Edition, 2000.

[59] A.Rosenfeld, *Picture Processing by Computer* Academic Press, 1969.

[60] S.Smith and J.Brady, "SUSAN - a new approach to low level image processing", International Journal of Computer Vision, vol. 23, no. 1, pp. 45-78, 1997.

[61] G. Borgefors, "Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 6, pp.849-865, November, 1988.

[62] W.J.Rucklidge, "Locating Objects using the Hausdorff Distance," Proceedings of 5th International Conference on Computer Vision, pp. 457-464, June, 1995.

[63] V.Ayala-Ramirez, C. Parra and M.Devy, "Active tracking based on Hausdorff Matching," Proceedings of International Conference on Pattern Recognition, vol. 4, pp.706-709, 2000.

[64] E.S. Nielsen, J.L.Navarro and M.H. Tejera, "Increasing Efficiency of Hausdorff Approach for Tracking Real Scences with Complex Environments," Proceedings of 11th International Conference on Image Analysis and Processing, pp. 131-136, September, 2001.

[65] P. Gastaldo and R.Zunino, "Hausdorff Distance for robust and Adaptive Template Selection in Visual Target Detection," Electronics Letters, vol. 38, no.5, pp.1651-1653, December, 2002.

[66] S.H.Kim and R.H.Park, "An Efficient Algorithm for Video Sequence Matching Using the Modified Hausdorff Distance and the Directed Divergence," IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 7, pp. 592-596, July, 2002.

[67] J.You, E.Pissaloux, J-L.Hellec and P.Bonnin, "A Guided Image Matching approach using Hausdorff Distance with Interesting Points Detection," Proceedings of International Conference on Image Processing, vol. 1, pp. 968-972, November, 1994.

[68] J.You, E.Pissaloux and H.A.Cohen, "A Hierarchical Image Matching scheme based on the dynamic Detection of Interesting Points," International Conference on Acoustics, Speech and Signal Processing, vol.4, pp.2467-2470, May, 1995.

[69] D.P. Huttenlocher and W.J.Rucklidge, "A Multi-Resolution Technique for Comparing Images using the Hausdorff Distance," Proceedings of Computer Vision and Pattern Recognition, pp.705-706, June, 1993.

[70] O.K.Kwon, D.G.Sim and R.H. Park, "New Hausdorff Distances based on robust Statistics for Comparing Images," International Conference on Image Processing, vol.3, pp.21-24, September, 1996.

[71] I.Han, I.D.Yun and S.U.Lee, "Model-based Object Recognition using the Hausdorff Distance with Explicit pairing," International Conference on Image Processing, vol.4, pp.83-87, 1999.

[72] M-P.Dubuisson and A.K.Jain, "A Modified Hausdorff Distance for Object Matching," Proceedings of 12th International Conference on Pattern Recognition, pp. 566-568, Jerusalem, Israel, October, 1994.

[73] C. Guerra and V. Pascucci, "3D Segment Matching using the Hausdorff Distance," Proceedings of 7th International Conference on Image Processing and its Applications, vol.1, pp.18-22, 1999.

[74] C.Robertson and J.A.Robinson, "Page Similarity and the Hausdorff Distance," Proceedings of 7th International Conference on Image Processing and its Applications, vol.2, pp.755-759, 1999.

[75] K.H.Lin, B.Guo, K.M.Lam and W.C.Siu, "Human Face Recognition using a spatially weighted modified Hausdorff Distance," Proceedings of International Symposium on Intelligent Multimedia, Video and Speech Processing, pp.477-480, Hong Kong, May, 2001.

[76] B.Achermann and H.Bunke, "Classifying Range Images of Human Faces with Hausdorff Distance," Proceedings of International Conference on Pattern Recognition, vol.2, pp.809-813, 2000.

[77] X.Yi and O.I.Camps, "Robust Occluding Contour Detection using the Hausdorff Distance," Proceedings of International Conference on Computer Vision and Pattern Recognition, pp.962-968, June, 1997.

[78] D-G.Sim, O-k.Kwon and R-H.Park, "Object Matching Algorithms Using Robust Hausdorff Distance Measures," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 3, pp. 425-429, March, 1999.

[79] A. Baeumker and W. Dittrich., "Parallel algorithms for Image processing: Practical Algorithms with experiments", Technical re-

port, Department of Mathematics and Computer Science, University of Paderborn, Germany, 1996.

[80] J.F.JaJa, *Introduction to Parallel Algorithms*, Addison-Wesley, 1992.

[81] Rolf Rabenseifner, Parallel Programming Workshop Course Material, Internal report 166, Computer Center, University of Stuttgart, 2001.
`http://www.hlrs.de/organization/par/par_prog_ws/`

[82] K.K. Parhi and D.G. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding", IEEE Transactions on Computers, vol. 40, no. 2, pp. 178-194, Febuary, 1991.

[83] H.Anton, *Calculus with analytical geometry* Wiley, Newyork, 1980.

[84] J.T.Camillo and J.K.David, "Vision-based motion planning and exploration algorithms for mobile robots", IEEE Transaction on Robotics and Automation, pp. 417-426, June, 1998.

[85] Z.C.Danny, J.S. Robert and J.U.John, "A framed-quadtree approach for determining euclidean shortest paths in a 2D environment", IEEE Transactions on Robotics and Automation, pp. 668-681, October, 1997.

[86] J.A.Fernandes and J.Gonzalez, "Hierarchical graph search for mobile robot path planning", IEEE ICRA'98 pp. 656-661 April, 1998.

[87] D.Guy and S.Francois, "An efficient algorithm to find a shortest path for a car-like robot", IEEE Transaction on Robotics and Automation, pp. 819-828, December, 1995.

[88] E.Horowitz and S.Sartaj, " Fundamentals of Computer Algorithms", Computer Software Engineering Series, London Pitman, 1978.

[89] J.Y.J.Hsu and L.S.Hwang, "A graph based exploration strategy of indoor environments by a autonomous mobile robot", IEEE ICRA'98, pp. 1262-1268, April, 1998.

[90] Y.K.Hwang and N.Ahuja, "Gross motion planning - A survey", ACM Computing Surveys, vol. 24, no.3, pp. 219-291, September, 1992.

[91] J.-C.Latombe, *Robot Motion Planning*, Boston, M.A : Kluwer, 1991.

[92] L.E.Kavraki, Petr. S., J-C. Latombe and H.O.Mark, "Probabilistic Road maps for path planning in High-dimensional configuration spaces", IEEE Transactions on Robotics and Automation, pp. 566-580, August, 1996.

[93] D.Maio and S.Rizzi, "Map learning and clustering in autonomous systems", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 15, no 12, pp. 1286-1297, December, 1993.

[94] D.Pagac, E.M.Nebot and H.D.Whyte, "An evidential approach to probabilistic map building", IEEE ICRA'96 pp. 745-750, April, 1996.

[95] F.P.Preparata and M.I.Shamos, *Computational Geometry : an introduction*, Springer Verlag, 1985.

[96] L.Shmoulian and E.Rimon, "A_e^* DFS - an algorithm for minimizing search effort in sensor based mobile robot navigation", IEEE ICRA'98, pp. 356-362, April, 1998.

[97] A.Stentz, "Optimal and efficient path finding for partially known environments", IEEE ICRA'94, pp. 3310-3317, May, 1994.

[98] F.Wallner, R.Graf and R.Dillman, "Realtime map refinement by fusing sonar and active stereo vision", IEEE ICRA'95, pp. 2968-2973, 1995.

[99] P.C.Chen and Y.K.Hwang, "SANDROS: A dynamic graph search algorithm for motion planning", IEEE Transactions on Robotics and Automation, pp. 390-403, vol. 14, no. 3, June, 1998.

[100] G.Foux, M.Heymann and A.Bruckstein, "Two dimensional robot navigation among unknown stationary polygonal obstacles", IEEE Transactions on Robotics and Automation, vol. 9, no. 1, pp. 96-102, 1993.

[101] R.A.Howard, *Dynamic programming and Markov processes*, MIT Press, Cambridge Mass. 1960.

[102] J.L.Peterson, *Petri net theory and the modelling of systems*, Prentice-Hall, Engelwood Cliffs, 1981.

[103] J.Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, 1984.

[104] G.Rozenburg, *Advances in Petri nets*, Lecture Notes in Computer Science, Springer Verlag, Berlin, 1985.

[105] M.Sharpe, *General theory of Markov processes*, Academic Press, Boston, 1988.

[106] M.Dorigo and L.M.Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE. Transactions on Evolutionary Computation, vol.1, no.1, pp. 53-66, April, 1997.

[107] M.Dorigo, V. Maniezzo and A. Colorni, The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26, no.1, pp.29-41, February, 1996.

[108] K.Doerner, W.J.Gutjahr, R.F.Hartl, C.Strauss and C.Stummer, "Ant Colony Optimization in Multiobjective Portfolio Selection", Proceedings of 4th Metaheuristics International Conference, pp.243-248, Porto, Portugal, July, 2001.

[109] Gerhard Reinelt, "TSPLIB" , www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[110] C.Strauss, G. Kotsis and B. Bullnheimer, "Parallelization Strategies for the Ant System", High Performance Algorithms and Software in Nonlinear Optimization, Applied Optimization, vol. 24, pp. 87-100, Dordrecht, 1998.

[111] Thomas Stuetzle, "Parallelization Strategies for Ant Colony Optimization", Proceedings of Parallel Problem Solving from Nature PPSN-V, vol. 1498 of LNCS, pp. 722-731, Springer Verlag, Amsterdam, 1998.

[112] Thomas Stuetzle and Holger H. Hoos, "MAX-MIN Ant System", Future Generation Computer Systems Journal, vol. 16, no.8, pp.889–914, 2000.

[113] High Performance Computing Center Stuttgart (HLRS) *www.hlrs.de.*

[114] William Gropp, Ewing Lusk and Anthony Skejellum., *Using MPI*, MIT press, 1995.

[115] *MPI-2, Special Issue*, The International Journal of High Performance Computing Applications, vol. 12, no. 1/2, 1998.

[116] M.R.Gary and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H.Freeman and Co., 1979.

[117] E.G.Coffman and R.L.Graham, "Optimal Scheduling for Two-processor Systems", Acta Informatica, vol.1, pp.200-213,1972.

[118] R.Sethi, "Scheduling Graphs on Two Processors", SIAM Journal of Computing, vol.5, no.1, pp.73-82, 1976.

[119] H.El-Rewini and T.G.Lewis, "Scheduling Parallel Program Tasks onto Arbitrary Target Machines", Journal of Parallel and Distributed Computing, vol.9, no.2,pp.138-153, 1990.

[120] B.Shirazi, M.Wang, and G.Pathak, "Analysis and Evaluation of Heuristic Methods for Static Scheduling", Journal of Parallel and Distributed Computing, vol.10, pp.222-232, 1990.

[121] G.C.Sih and E.A.Lee, "A Compile-time Scheduling Heuristic for Interconnection-constrained Heterogeneous Processor Architectures", IEEE Transactions on Parallel and Distributed Systems, vol.4, no.2, pp.75-87, 1993.

[122] V.Chaudhary and J.K.Aggarwal, "A Generalized Scheme for Mapping Parallel Algorithms", IEEE Transactions on Parallel and Distributed Systems, vol.4, no.3, pp.328-346, 1993.

[123] S.Selvakumar and C.S.R. Murthy, "Scheduling Precedence-constrained Task Graphs with Non-negligible Intertask Communication onto Multiprocessors", IEEE Transactions on Parallel and Distributed Systems, vol.5, no.3, pp.328-336, 1994.

[124] G.L.Djordjevic and M.B.Tosic, "A Compile-time Scheduling Heuristic for Multiprocessor Architectures", The Computer Journal, vol.39, no.8, pp.664-667, 1996.

[125] Y. Kopidakis, M. Lamari, and V. Zissimopoulos, "On the Task Assignment Problem: Two New Efficient Heuristic Algorithms",

Journal of Parallel and Distributed Computing, vol. 42, no.1, pp.21-29, 1997.

[126] K.R.Pattipati, T.Kurien, R.T. Lee, and P.B. Luh, "On Mapping a Tracking Algorithm onto Parallel Processors", IEEE Transactions on Aerospace and Electronic Systems, vol.26, no.5, pp.774-791, 1990.

[127] J.J. Hwang, Y.C. Cow, F.D. Anger, and C.Y. Lee, "Scheduling Precedence Graphs in Systems with Interprocessor Communication times", SIAM Journal of Computing, pp. 244-257, 1989.