QUALITATIVE SPATIAL REASONING ABOUT DISTANCE AND ORIENTATION

A dissertation submitted in partial fulfilment of the

requirements for the award of the degree of

Master of Engineering in Electronics & Tele-communication Engineering

By

DEEP KUMAR PAL

Under the guidance of

Dr. AMIT KONAR

DEPARTMENT OF ELECTRONICS AND TELE-COMMUNICATION ENGINEERING

JADAVPUR UNIVERSITY

KOLKATA – 700032

WEST BENGAL, INDIA

2022

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

RECOMMENDATION

DATE : __ / __ / 2022

I hereby recommend that the thesis prepared under my direct supervision by **Mr. DEEP KUMAR PAL** (**University Registration No. –** 154080 of 2020 – 2021 and **Examination Roll No. –** M4ETC22011) entitled "**QUALITATIVE SPATIAL REASONING ABOUT DISTANCE AND ORIENTATION**" be accepted in partial fulfilment of the requirement for the award of the degree of Master of Engineering in Electronics & Tele-communication Engineering of Jadavpur University, Kolkata.

Dr. AMIT KONAR
Supervisor of the Thesis

Dr. ANANDA SHANKAR CHOWDHURY
Professor and Head of the Department

Prof. CHANDAN MAZUMDER
Dean of Faculty of Engineering and Technology

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL*

The foregoing thesis is hereby approved as a creditable study of Engineering subject to warrant its acceptance as a pre-requisite to obtain the degree for which it has been submitted. It is understood that by this approval the undersigned do no necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the thesis only got the purpose for which it is submitted.

…….

…….

……..…
Signature of Examiners

*Only in the case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

DECLARATION OF ORIGINALITY AND COMPLIANCE OF THESIS

I hereby declare that this thesis entitled "QUALITATIVE SPATIAL REASONING ABOUT DISTANCE AND ORIENTATION" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Engineering in Electronics & Tele-communication Engineering. All information has been obtained and presented in accordance with academic rules and ethical conduct. It is hereby declared that, as required by these rules and conduct, all materials and results that are not original to this work have been properly cited and referenced.

Candidate Name – Deep Kumar Pal

Registration No. – 154080 of 2020-2021

Examination Roll No. – M4ETC22011

Thesis Title – Qualitative Spatial Reasoning about Distance and Orientation

Date: …….…….

Signature of Candidate

ACKNOWLEDGEMENT

I would like to express my earnest gratitude and sincere thanks to my thesis supervisor Prof. Amit Konar, Department of Electronics and Tele-communication Engineering, Jadavpur university, for giving me opportunity to work under him and inspiring me to explore the interesting fields of Artificial Intelligence and Robotics. I am indebted to him for his patient, guidance, critical and constructive views and untiring support that shaped my work. The past year has been a remarkable experience in terms of gaining knowledge and skill that I hope to carry on and develop further.

I am extremely grateful to Prof. Ananda Shankar Chowdhury who has acted as Head of the Department of Electronics and Tele-communication Engineering, Jadavpur University during my Master of Engineering course for their valuable guidance and support. I convey my regards and respect to all Professors and associates in the Department of Electronics and Tele-communication Engineering for their help and support. I am thankful to the Department of Electronics and Tele-communication Engineering for the assistance with materials, equipment and subjects for conducting extensive experiments in the laboratories.

I would like to extend my sincere respect and thanks to all seniors who have provided constant support and guidance throughout my work. I consider myself fortunate to have worked with such friendly and motivating seniors who helped me in cultivating my knowledge and improving my skills.

Finally, I am immensely indebted to my parents, family and friends for their continuous support and encouragement that made me believe in myself and provided the strength to work hard.

[Deep Kumar Pal]
Dept. of Electronics & Tele-communication Engineering
Jadavpur University

PREFACE

The chapter 1 introduces the spatial reasoning concept. There are the various axioms of spatial reasoning , topological relationship, fuzzy relationships required for the reasoning and finally the vision aspects required for the reasoning.

In chapter 2 methods for extracting spatial features of a binary 2-D image planar image is discussed. In this module the how grey values of images are passed through certain functions for determining the vertices, wire grid models and valid segments are discussed.

The Chapter 3 consists of the methods for representing position and orientation in space. The initial discussion starts from Cardinal method and ends with Granular point position calculus.

Finally in chapter 4 the reasoning is developed for spatial reasoning and using position and orientation.

The appendices contain the codes for implementation and some topics which were partly discussed in previous chapters.

CONTENTS

1

Introduction to Spatial Reasoning

1.1 Introduction

There exist many real-world problems, where the importance of space and time cannot be ignored. For instance, navigational planning of a mobile robot in a given workspace. The robot has to plan its trajectory from a pre-defined starting point to a given goal point. If the robot knows the map, it can easily plan its path so that it does not touch the obstacles in the map. If the robot has no prior knowledge about its trajectory, then it has to solely rely on the data it receives by its sonar and laser sensors or the images it grabs by a camera and processes these on-line. Representation of the space by some formalism and developing an efficient search algorithm for matching of the spatial data is required for robot's movement. Now, if the obstacles in the robot's world are dynamic then it requires information about both space and time. For example, the velocity and displacements of the obstacles at an instance is used to determine the speed and direction of the robot at next instance. Thus, there is a need for both spatial and temporal representation of information.

Spatial reasoning problems can be handled by many of the known AI techniques. If we can represent the navigational planning problem of a robot by a set of spatial constraints, we can solve it by a logic program or the constraint satisfaction techniques. Alternatively, if we can represent the spatial reasoning problem by predicate logic, we may employ the resolution theorem to solve it. One way of doing this is to define a set of spatial axioms by predicates and then describe a spatial reasoning problem as clauses of the spatial axioms.

The FOL based representation of a spatial reasoning problem sometimes is ambiguous and, as a consequence, the ambiguity propagates through the reasoning process as well. For example, suppose an object X *is not very close* to object Y in a scene. If we try to represent this in FOL then for each specific distance between two objects, we require one predicate. In fuzzy logic we need to define a membership function of '*Not-very-close*' versus distance, and can easily obtain the membership value of *Not-very-close*(X, Y) with known distance between X and Y. The membership values may later be used in fuzzy reasoning.

Reasoning with time is equally useful like reasoning in space. An occurrence of an event A at time t, and another event B at time t+1, causes the event C to occur at time t+2 can be represented by extending the First order logic to two alternative forms. First one is called the **situation calculus**, after John McCarthy, the father of AI. The other one is an extension by new temporal operators known as **propositional temporal logic**.

1.2 Axioms of Spatial Reasoning

Spatial reasoning deals with the problems of reasoning with space. It is used to understand how items fit together in a space. There are a few elementary axioms based on which such reasoning can be carried out. These axioms for spatial reasoning can be extended for specific applications. The **Axioms of Spatial Reasoning** are as follows –

Axiom 1: Consider the problems of two non-elastic objects O_i and O_j. Let the objects be infinitesimally small having 2D co-ordinates (x_i, y_i) and (x_j, y_j) respectively. Then it can be stated that, $\forall O_i, O_j$ we have $x_i \neq x_j$ or $y_i \neq y_j$.
Formally, $\forall O_i, O_j$ $Different(O_i, O_j) \rightarrow \neg(Eq(x_i, x_j) \wedge Eq(y_i, y_j))$

An extension of the above principle is that no two non-elastic objects, whatever may be their size, cannot occupy a common space. If S_i and S_j are the spaces occupied by O_i and O_j respectively.

Then, $S_i \cap S_j = \emptyset$.
$\Rightarrow \neg(S_i \cap S_j) = true$
$\Rightarrow \neg S_i \cup \neg S_j = true$

By combining the above two conditions and placing AND ($\wedge$) and OR ($\vee$) operators for intersection ($\cap$) and union ($\cup$) of surfaces or their negations (complements) it can be written as,

$\forall O_i, O_j$ $S_i(O_i) \wedge S_j(O_j) \wedge Different(O_i, O_j) \rightarrow \neg S_i(O_i) \vee \neg S_j(O_j)$
$\Rightarrow$ $\forall O_i, O_j$ $S_i(O_i) \wedge S_j(O_j) \wedge \neg Eq(O_i, O_j) \rightarrow \neg S_i(O_i) \vee \neg S_j(O_j)$

Further, $\forall O_i, O_j$ means $O_i, O_j \in S$, where S is the entire space that contains O_i and O_j as shown in figure below,

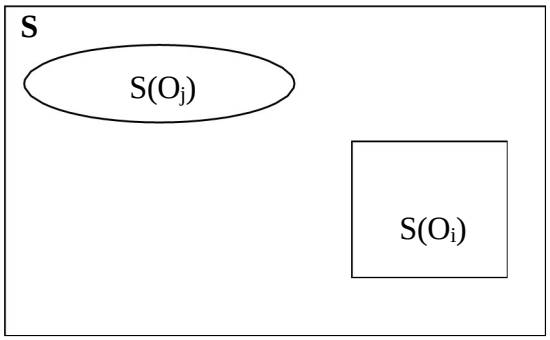


Fig. 1.1: Space S containing object O_i and O_j having 2-D surfaces $S(O_i)$ and $S(O_j)$

The above formulation can be extended for three dimensions as well.

Axiom 2: When an object O_i enters the space S, $S \cap S(O_i) \neq \emptyset$, which implies $S \wedge S(O_i)$ is true. Formally,

$$\forall O_i\ S(O_i) \wedge Enters(O_i, S) \rightarrow S \wedge S(O_i)$$

Similarly, when an object O_i leaves a space S, $S \cap S(O_i) = \emptyset$, which implies $\neg S \vee \neg S(O_i)$ is true. Formally,

$$\forall O_i\ S(O_i) \wedge Leaves(O_i, S) \rightarrow \neg S \vee \neg S(O_i)$$

Axiom 3: If $B(O_i)$ and $B(O_j)$ be the boundary lines (or surface) of the object O_i and O_j. When the objects touch each other, $B(O_i) \cap B(O_j) \neq \emptyset$ i.e., $B(O_i) \wedge B(O_j)$ is true. Formally,

$$\forall O_i, O_j\ Touch(O_i, O_j) \rightarrow B(O_i) \wedge B(O_j)$$

When object O_i does not touch object O_j in space S, $B(O_i) \cap B(O_j) = \emptyset$ i.e., $\neg B(O_i) \vee \neg B(O_j)$ is true. Formally,

$$\forall O_i, O_j\ Touch(O_i, O_j) \rightarrow \neg B(O_i) \vee \neg B(O_j)$$

When the intersection of the surface boundary of two objects is a non-null set, it means either one is partially or fully embedded within the other, or they touch each other. Further, when a two-dimensional surface touches another, the common points must form a 2-D line or a point. Similarly, when a 3-dimensional surface touches another, the common points must be a 3-D /2-D surface or a 3-D / 2-D line or a point. It is thus evident that two objects touch each other, when their intersection of surface forms a surface of at most their dimension. Formally,

$\forall O_i, \forall O_j$ Less-than-or-Equal-to**(**dim$(S(O_i) \wedge S(O_j))$, dim$(S(O_i)$**)** $\wedge$
Less-than-or-Equal-to**(**dim$(S(O_i) \wedge S(O_j))$, dim $(S(O_j)$**)** $\rightarrow$ Touch(O_i, O_j).

where 'dim' is a function that returns the dimension of the surface of its argument and dim **(**$S(O_i) \wedge S(O_j)$**)** represents the dimension of the two intersecting surfaces of objects O_i and O_j. The $\wedge$-operator between the predicates Less-than-or-Equal-to denotes logical AND operation.

Axiom 4: The shortest distance between two objects O_i and O_j can be obtained by drawing perpendiculars on surface of one object at regular interval of δ on the surface of very small

lengths. The perpendicular having the minimum length among the perpendiculars which hit the surface of other object is the shortest distance d_{ij} between boundaries of objects O_i and O_j.

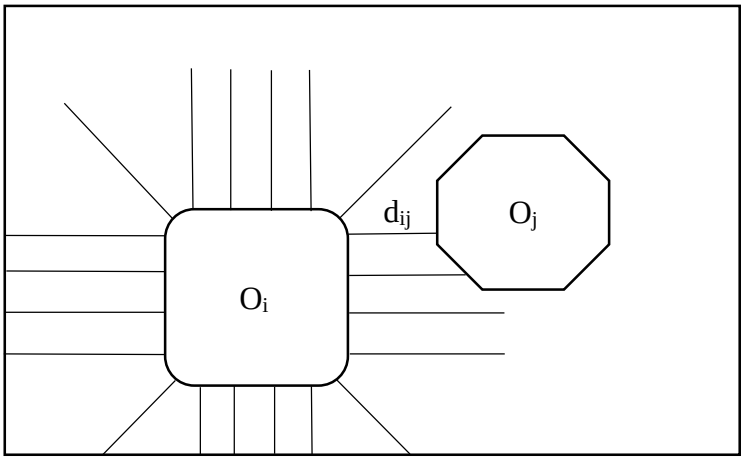


Fig. 1.2: Finding shortest distance between two boundaries B(O_i) and B(O_j)

For two scenarios if d_{ij_1} and d_{ij_2} denote the shortest distance between the objects O_i and O_j in scene 1 and 2 respectively, then if $d_{ij_2} < d_{ij_1}$, we can say the objects O_i and O_j are closer in moment 2 compared to that in moment 1. Formally,

$\forall O_i, O_j$ Exists(O_i, O_j, in-scene-1) $\wedge$ Shortest-distance(d_{ij_1}, O_i, O_j, in-scene-1) $\wedge$ Exists(O_i, O_j, in-scene-2) $\wedge$ Shortest-distance(d_{ij_2}, O_i, O_j, in-scene-2) $\wedge$ Smaller(d_{ij_2}, d_{ij_1}) $\rightarrow$ Closer(O_i, O_j, in-scene-2, wrt-scene = 1)

where the predicate Exists(O_i, O_j, in-scene-k) means O_i and O_j exists in scene k; Shortest-distance(d_{ij_k}, O_i, O_j, in-scene-k, wrt-scene = 1) denotes that d_{ij_k} is the shortest distance between O_i and O_j in scene k with respect to scene l.

The axioms of spatial reasoning presented above can be employed in many applications. One typical application is the path planning of a mobile robot. For example, the space S, where a triangular shaped mobile robot has to move from a given starting to goal point, without touching the obstacle O_1, O_2, O_3, O_4, …., O_7.

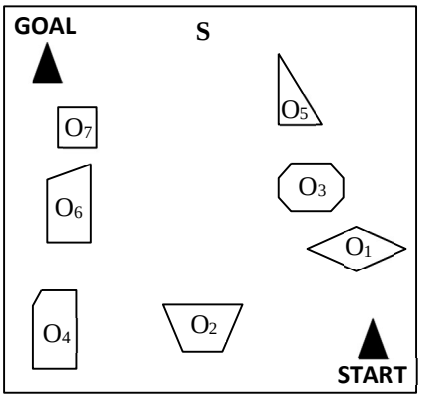


Fig. 1.3: Path planning of a robot R in space S

The robot R can sense the obstacles from a distance by sensors, located around the boundary of it. The CLP (constraint logic program) of this problem can be given as,

Move(R, Starting-position, Goal-position): -
Move S(R) in S
$\forall O_i \neg$Touch**(**S(R), S(O_i)**)**

Move(R, Goal-position, Goal-position)

The above program allows the robot R to wander around its environment, until it reaches the goal-position. The program ensures that during the robot's journey it does not hit an obstacle. Now to move the robot through a shortest path CLP defined uses the following nomenclature.

1. Next-position(R): It is a function that gives the next-position of a robot R.

2. S**(**Next-position(R)**)**: It is a function, representing the space to be occupied by the robot at the next-position of R.

The robot selects an arbitrary next position from its current position and then tests whether the next-position touches any object. If yes, it drops that next-position and selects an alternative one until a next-position is found, where the robot does not touch any obstacle. If more than one next-position is found, it would select that position, such that the sum of the distances between the current and the next-position, and between the next position and the goal is minimum. A pseudo-Pascal algorithm is presented below as,

Procedure Move-optimal(R, Starting-position, Goal-position)
Begin
Current-position(R) ← Starting-position(R);
While goal not reached **do**
Begin
Repeat
Find-next-position(R);
j ← 1;
If S**(**Next-position(R)**)** does not touch S(O_i) ∀i
Then do
Begin
Save Next-position(R) in A[j];
j ← j + 1;
End
Until all possible next positions are explored**;**
∀j Select the Next-position in A[j] with the minimum distance from the current position of R and the goal;
Call it A[k];
Current-position(R) ← A[k];
End while
End

The CLP takes care of two optimizing constraints, (i) Movement of the robot without touching the obstacles and (ii) Traversal of an optimal or near-optimal path from starting position to the goal position.

Move-optimal(R, Starting-position, Goal-position):-
Move S(R) in S,
$\forall i$ ¬Touch**(**S(R), S(O_i)**)**,
Distance**(**Next-position(R), Current-position(R)**)** + Distance**(**Next-position(R), Goal-position**)** is minimum $\forall$ feasible Next-position(R),
Current-position(R) ← Next-position(R),
Move-optimal(R, Current-position, Goal-position).

Move-optimal(R, Goal-position, Goal-position).

It is to be noted that Touch**(**S(R), S(O_i)**)** is not explicitly defined as it is presumed to be available in the system as a standard predicate, following axiom 3. Further the distance constraint can be redefined in the last program by axiom 4 as follows:

Closer**(**Next-position(R), Current-position(R), in-scene-k, wrt-scene ≠ k**)**,
Closer**(**Next-position(R), Goal-position, in-scene-k, wrt-scene ≠ k**)**

1.3 Topological Relationship Among the Regions

The representation of spatial relationship among regions can be described by graph. For simplicity it is assumed regions are all rectangular but it is applicable to non-rectangular region as well. The principle used to construct the spatial graph is presented below,

1. Start pointer from the topmost left corner.
2. i ← 1;
3. j ← 1;
4. Mark region as a_{ij} and scan horizontally right until pointer is in same region.
5. j ← j + 1;
6. Mark region a_{ij} if it is a new region. Connect region to $a_{i(j-1)}$ by right(R) link.
7. Repeat step 4 to step 6 until rightmost edge of the map.
8. Move down by a small say δ from topmost left and repeat step 7 until pointer is in same region.
9. i ← i + 1;
10. Connect a_{i1} to $a_{(i-1)1}$ by down(D) link.
11. Repeat step 8 until pointer reaches downmost edge of map.

The scanning principle given above is used to mark the regions given in map below and its corresponding graph.

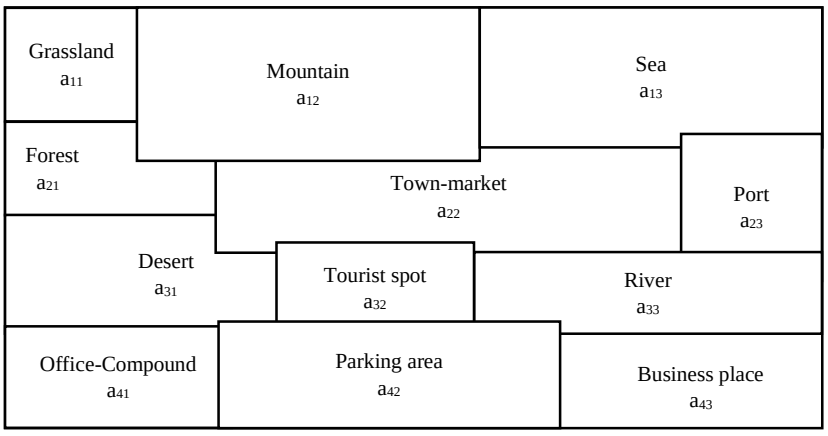


Fig. 1.4: Map of a region

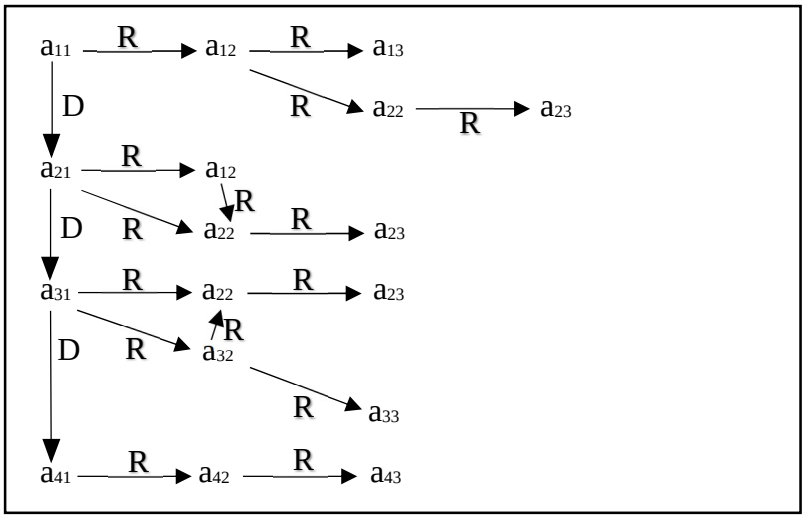


Fig. 1.5: Graph of the map

1.4 Spatial Relationships among Components of an Object

Many physical and geometric objects can be recognized from the spatial relationship among its components. For instance, let us define a chair as an object consisting of two planes "abcf" and "cdef" having an angle θ between them, where $\theta \leq 90^0+\alpha$ and where $0 \leq \alpha \leq 45^0$. Further, one of its planes is perpendicular to at least 3 legs (the 4th one being hidden in the image). So, it can be defined as:

Object(chair):- Angle-between(plane1, plane2, 90+α), Greater-than(α, 0), Less-than (α, 45), Parallel(line1, line2, line3), Perpendicular (line1, plane1)!

For actual realization of the small program presented above, one has to define equation of lines and planes; then one has to check the criteria listed in the logic program. It may be noted here that finding equation of a line in an image is not simple. One approach to handle this problem is to employ a stochastic filter, such as Kalman filtering.

A skeleton of a chair, which can be obtained after many elementary steps of image processing is presented in fig. 1.6. Now, the equation of the line segments is evaluated approximately from the set of 2-dimensional image points lying on the lines. This is done by employing a Kalman filter. It may be noted that the more the number of points presented to the filter, the better would be the accuracy of the equation of the 2-dimensional lines. These 2-D lines are then transformed to 3-D lines by another stage of Kalman filtering. Now, given the equation of the 3-D lines, one can easily evaluate the equation of the planes framed by the lines by using analytical geometry. Lastly, the constraints like the angles between the planes, etc. are checked by a logic program, as described above.

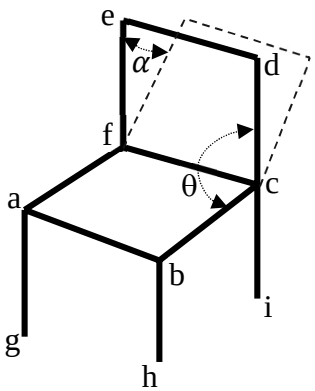


Fig. 1.6: Spatial relations among components of a skeleton chair

1.5 Fuzzy Spatial Relationships among Objects

Consider the objects A and B in fig. 1.7(a) and 1.7(b). B is left to A. It is to be noted that B and A have some overlap in (a) but there is no overlap in (b).

(a) (b)

Fig. 1.7: Object B is left to object A: (a) with overlap, (b) without overlap

Now consider fig. 1.8 where in both (a) & (b) B is down to A; but in (a) B is exactly down to A, whereas in (b) it is right shifted a little. To define these formally, we are required to represent the spatial relationships between the objects A and B by fuzzy logic.

(a) (b)

Fig. 1.8: Object B is down to object A: (a) exactly down, (b) down but right shifted

Let us first define spatial relations between points A and B. We consider four types of relations: right, left, above and below. Here the membership function considered as a square of sine or cosine angles θ (vide fig. 1.9), where θ denotes the angle between the positive X axis passing through point A and the line joining A and B. The membership functions for the primitive spatial relations are now given as follows:

$\mu_{right}(\theta) = \cos^2(\theta)$, when $-\pi/2 \leq \theta \leq \pi/2$

$= 0$, otherwise

$\mu_{left}(\theta) = \cos^2(\theta)$, when $-\pi < \theta < -\pi/2$ or $\pi/2 \leq \theta \leq \pi$

$= 0$, otherwise

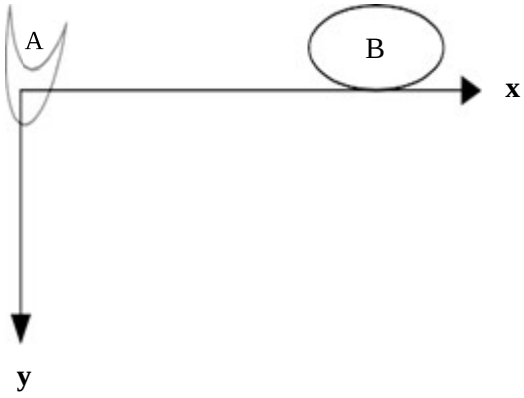
$\mu_{below}(\theta) = \sin^2(\theta)$, when $0 \leq \theta \leq \pi$

$= 0$, otherwise

$\mu_{above}(\theta) = \sin^2(\theta)$, when $-\pi \leq \theta \leq 0$

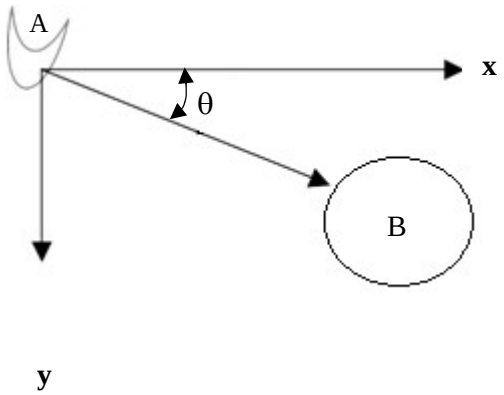
$= 0$, otherwise

For an example, the 'below membership function' $\mu_{below}(\theta)$ at a regular interval of $\theta = \pi/4$ in the graph $0 \leq \theta \leq \pi$. It is clear from the figure that when B is exactly below A (fig. 1.9(c)) $\mu_{below}(\theta = \pi/2) = 1$, which is logically appealing. Again when $\theta = \pi/4$ or $\theta = 3\pi/4$ (fig. 1.9(b) & 1.9(d)), the membership value of $\mu_{below}(\theta) = 1/2$; that too is logically meaningful. When $\theta = 0$ (fig. 1.9(a)) or π, $\mu_{below}(\theta) = 0$, signifying that B is not below A. The explanation of other membership functions like $\mu_{right}(\theta)$, $\mu_{left}(\theta)$, $\mu_{above}(\theta)$ can be given analogously.



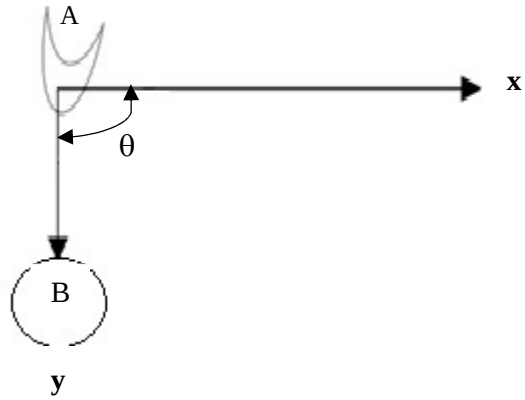
$\theta = 0$, $\mu_{below}(\theta) = 0$

(a)



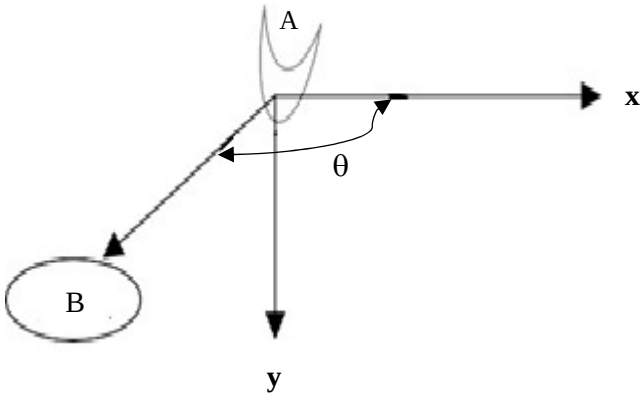
$\theta = \pi/4$, $\mu_{below}(\theta) = 1/2$

(b)



$\theta = \pi/2$, $\mu_{below}(\theta) = 1$

(c)



$\theta = 3\pi/4$, $\mu_{below}(\theta) = 1/2$

(d)

Fig. 1.9: Illustrating significance of the $\mu_{below}(\theta)$ function

So far, we discussed spatial relationship between two points by fuzzy measure. Now, we shall discuss the spatial relationships between two objects. Let A and B be two objects and $\{a_i, 1 \le i \le n\}$, $\{b_j, 1 \le j \le m\}$ be the set of points on the boundary A and B respectively. We first compute the angle θ_{ij} between each two points a_i and b_j . Since there is n no. of a_i points and m no. of b_j points, the total occurrence of θ_{ij} will be (m x n). Now, for each type spatial relation like b_j below a_i, we estimate $\mu_{below}(\theta_{ij})$. Since θ_{ij} has a large range of value $[0, \pi]$, we may find equal value of $\mu_{below}(\theta_{ij})$ for different values of θ_{ij}. A frequency count of $\mu_{below}(\theta_{ij})$ versus θ_{ij} is thus feasible. We give a generic name $f(\theta)$ to the frequency count. Since $f(\theta)$ can have the theoretical largest value (m . n), we divide $f(\theta)$ by (m . n) to normalize it. We call that normalized frequency $\underline{f}(\theta) = f(\theta) / (m . n)$. We now plot $\underline{f}(\theta)$ versus θ and find where it has the largest value. Now to find the spatial relationship between A and B, put the values of θ in $\mu_{below}(\theta)$ where $\underline{f}(\theta)$ is the highest.

In fig. 1.10 we illustrate the method of measurement of the possible θ_{ij}. Since "abcd" is a rectangle and "pqr" is a triangle, considering only the vertices, m . n = 3 . 4 = 12. We thus have 12 possible values of θ_{ij}. So, $\underline{f}(\theta) = f(\theta) / 12$. It is appearing clear that $f(\theta)$ will have the largest value at around 45 degrees (fig. 1.11); consequently below (θ = 45) gives the membership of "pqr" being below "abcd".

Fig. 1.10: Demonstrating $f(\theta) / (m . n)$ computation

Here in fig 1.10(a) and 1.10(b) computation of angles (w.r.t the horizontal axis) of the lines joining the vertices of the rectangle to the vertices p and q of the triangle have been illustrated. Similar computations have to be performed for the line joining the vertices of the rectangle to the vertex r of the triangle. All 8 angles have not been shown in the figure for clarity.

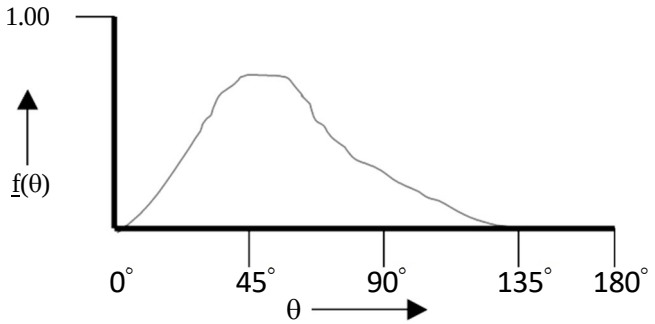


Fig. 1.11: Theoretical $\underline{f}(\theta)$ versus θ for example cited in fig. 1.10

1.6 Spatial Reasoning Through Vision

The currently existing scheme in mobile robotics include vision aids through camera which can graph two-dimensional images of objects. To trace the world map there is a strong demand for three-dimensional perception of the objects by the robot. This requires a transformation from 2-D images to 3-D models of the object seen by the camera. Kalman filtering can be applied for such modelling and transformation. Using Kalman filtering a 2-D point on an image can be mapped to the exact physical location of its 3-D point. Analogously, a 2-D line on an image can be mapped to a 3-D line and a 2-D plane to a 3-D plane. The most significant part of this technique concerns with minimal representation of 3-D line and 3-D plane. A linear recursive Kalman filter is then invoked to evaluate the parameters of a 3-D line (or plane) from multiple images of the same line (or plane) grabbed by a camera from different locations.

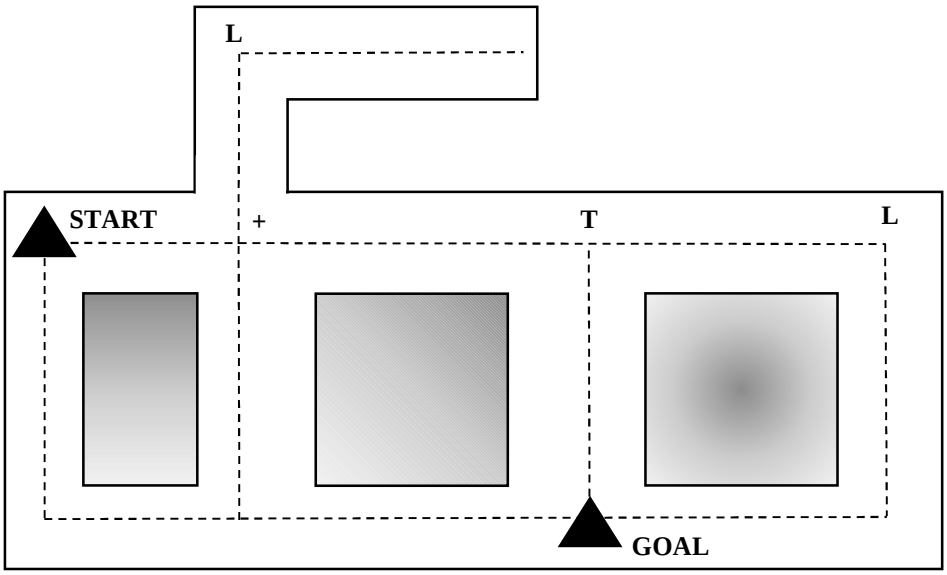


Fig. 1.12: A robot's workspace with starting and goal position

The robot uses the vision skill to study the surrounding for reasoning with space and time. It may study the discontinuities of the boundary of the obstacles and the starting and end boundary positions of the robot. A sequence of lines originated from one vertex of this starting position and terminal at one of goal position corresponds to one such possible path of the robot as given in fig. 1.12.
An alternative scheme for spatial reasoning requires representing the robots work space by a set of junctions shaped L, T, + and others. A robot has to determine the type of junctions while moving on a map and plan accordingly its tour to reach the desired goal position.

1.7 Summary

This chapter provided an overview of ideas and all axioms which are implemented in next chapters for the whole work. This chapter helps to get the reader acquainted with concepts of spatial reasoning for recognizing the various objects.

2
Visual Perception

2.1 Introduction

The phrase ‘Visual perception’ generally refers to construction of knowledge to understand and interpret 3-dimensional objects from the scenes that humans and machines perceive through their visual sensors. The human eye can receive a wide spectrum of light waves, and discriminate various levels of intensity of light reflected from an object. Many lower-class animals, however have a narrow spectrum of vision in the visible (400-700 nm) or infrared (>700 nm) wavelength. Modern robots are equipped with sonar (ultrasonic) or laser sensors and video cameras to capture the information around its world for constructing the model of the neighborhood. The knowledge about its neighboring world, derived from the sensory information, also called the ‘perception of the robot’, is useful for generating navigational plans and action (like arm movement) sequences by the robot.

In order to construct the models of visual perception, the camera is presumed to be the sensor.

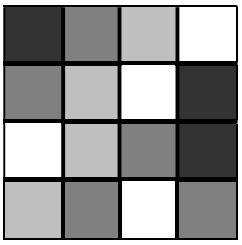
Though there exists some similarity between the human eye and a camera with respect to image formation (reflected light from an object grabbed by a camera). Another basic difference between the human eye and a camera lies in the dimensionality of the devices. While a human eye can feel the third dimension (the depth), a camera can extract only the 2-dimensional view of an object. Getting the third dimension requires integration of the multiple camera images taken from different directions referred to as the 3-D re-construction. There are many approaches to re-construct 3-D objects from their 2-D partial images. The 3-D reconstruction problem and its solution will be covered in this chapter by Kalman filtering, which dates back to the early 60's and remains a useful tool for signal recovery to date.

Another interesting and useful issue of image understanding is the 'recognition problem'. Kalman filtering can be used for recognizing and interpreting objects with planer surface boundaries. But it cannot identify objects having complex 3-D surfaces. Alternatively, the features of an object are extracted from its images and then a 'feature matching algorithm' may be employed to search the extracted features in the model feature-space of the objects.

Vision problems can be subdivided into three typical classes: low, medium and high-level vision. Though there exist no clear boundaries of these three levels, still the low and the medium levels include steps like pre-processing, enhancement and segmentation of images, while the high level corresponds to recognition and interpretation of scenes from their 2-D images.

2.2 Digital Images

The reflected light from the surface of an object is received by a camera and mapped onto a grid of cells in a Cartesian plane, called the image plane. The light received by the image plane is spatially sampled, quantized and encoded in binary codes. The encoded information thus obtained is represented by a two-dimensional array of sampled points, called pixels, and the encoded intensity level of each pixel is called its gray level. For instance, if 5 bits are used to represent the coding, we can have gray levels ranging from 0 to 31. Let's consider a (4 x 4) image, comprising of four gray levels. Let us define the darkest gray level to be 0 and the brightest to be 3 and the rest are in between.



3	2	1	0
2	1	0	3
0	1	2	3
1	2	0	2

Fig. 2.1: (a) An illustrative gray image and (b) it's encoding of gray levels in the range [0,3]

A digital image is a two-dimensional representation of pixels like that of fig. 2.1. The sampling and quantization process in a digital camera adds noise to the image. Elimination of noise from an image is a prerequisite step for understanding the image. In the next section, we discuss some elementary methods to eliminate noise from a digital image.

2.3 Low Level Vision

Low level vision pre-processes the image for subsequent operations. The most common type of pre-processing includes filtering the image from the noise. For filtering an image, it is to be first represented in frequency domain. An image is a two-dimensional array of pixels so two-dimensional Fourier transforms is used to get the frequency domain representation of the image. Thus, a filter can now be used to suppress the noise and one such filter is Gaussian filter. The digital filtering by Gaussian function attempts to smooth the frequency response of the image and is thus referred to as smoothing. Smoothing reduce the variations in the amplitude over a wide range of frequencies. Gaussian filter undoubtedly is a good scheme for smoothing. Besides gaussian filter, there exist masking techniques for smoothing. Two such masks, referred to as the 4-point and the 8-point masks. other smoothing algorithms, most common are mean and median filters. The **mean filter** takes the average of the gray levels of all 8- neighboring pixels to compute the changed gray level at pixel (x, y). The median filter, on the other hand, replaces the current gray level of pixel (x, y) by the median of its 8 neighborhood pixels. Experimental evidences show that mean filter **blurs** the image, but the median filter **enhances** the **sharpness** of the image.

2.4 Finding Edges or Boundaries in an Image

An **edge** is a contour of pixels that (artificially) separates two regions of different intensities. It can also be defined as a contour along which the brightness in the image changes abruptly. The idea of this module is to geta binary image and then to get the object boundaries. Here we have thought of an image with two grey levels only i.e., two types characters: asterisk (*) for region representing the object and blank ( ) for representing the background. Input the image file containing two grey levels in a character array, hence restore it in a same size integer array using the following rules,

1. If character is '*' then it corresponds to 1 (one) in the integer array.
2. Otherwise, it corresponds to 0 (zero) in the integer array.

The two-dimensional arrays in the program are taken as ca[M][N] for character array and a[M][N], b[M][N] for two identical integer arrays.

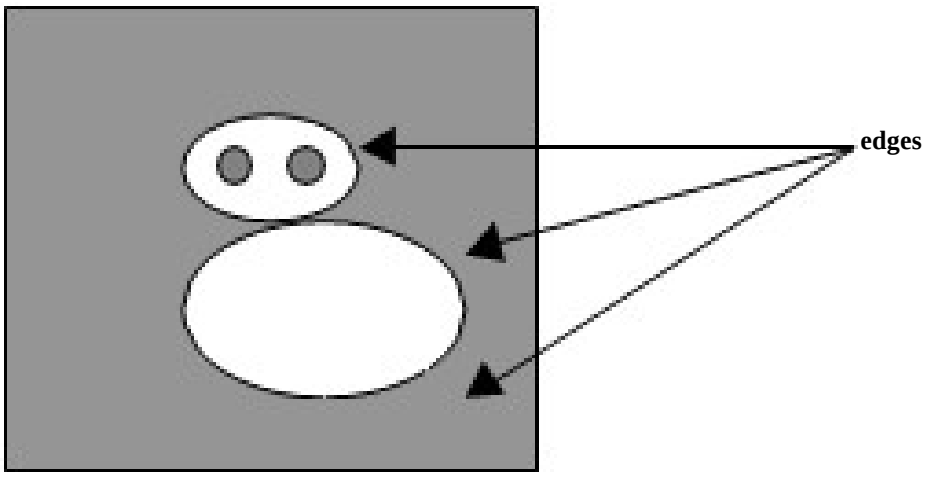


Fig. 2.2: Edges in a synthetic image

Applying Laplacian Operator: If $\emptyset$ be a scalar function then Laplacian of $\emptyset$ is given as $\nabla^2\emptyset$. For two-dimensional case it is expressed as $\nabla^2\emptyset = \frac{\partial^2\emptyset}{\partial x^2} + \frac{\partial^2\emptyset}{\partial y^2}$. The Laplacian is obtained such that $\frac{\partial^2\emptyset}{\partial x^2}$ represents the value of difference of two adjacent row elements done consecutively two times and $\frac{\partial^2\emptyset}{\partial y^2}$ represents the value of difference of two adjacent column elements done consecutively two times. If the matrix is of size M x N, then it becomes of size (M-1) x (N-1) after applying 1st order difference. All the elements in 1st row and 1st column are given value 0 (zero) to make it again a matrix of size M x N. In next step after applying 2nd order difference the matrix size is reduced to (M-2) x (N-2). At the end of application of Laplacian operator an M x N size matrix is obtained by appending 0 (zero) in 1st and Mth row and in 1st and Nth column. So, $\frac{\partial^2\emptyset}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial\emptyset}{\partial x}\right)$ and $\frac{\partial^2\emptyset}{\partial y^2} = \frac{\partial}{\partial y}\left(\frac{\partial\emptyset}{\partial y}\right)$ are obtained by performing the difference mentioned earlier. Other (M-2) x (N-2) size matrix will be used for tracing the object boundary.

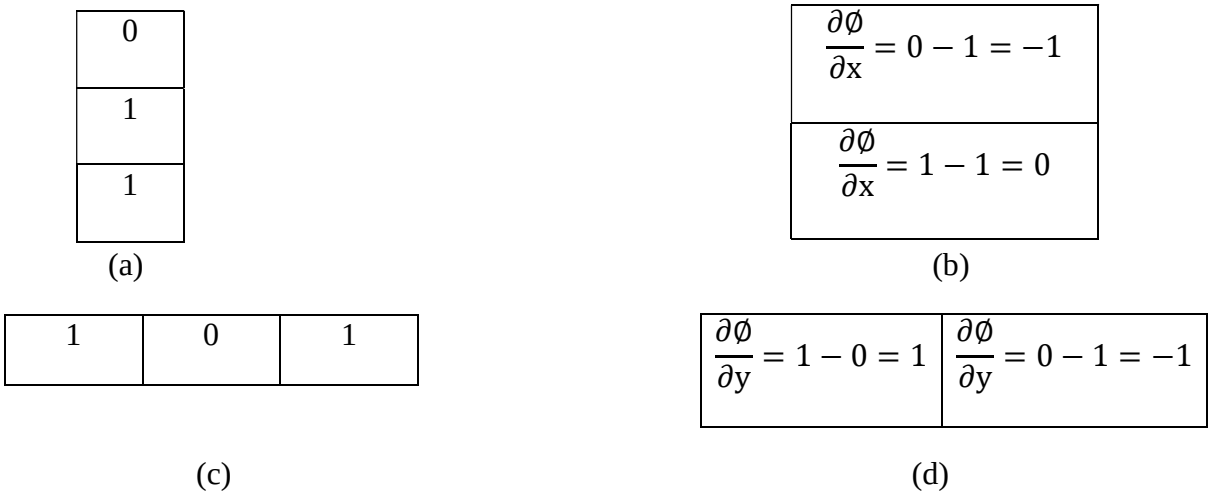


Fig. 2.3: Some examples of 1st order difference along row and column

Formally, the array indices of the matrix elements are represented by (m, n). It is required to found out whether it is a boundary element or not. The Laplacian of a matrix element can be directly obtained using the relation, $\nabla^2\emptyset_{mn} = f(m-1,n) + f(m+1,n) + f(m,n-1) + f(m,n+1) - 4f(m,n)$, where $f(m,n)$ represents the value of element in mth row and nth column in the matrix and $f(m-1,n), f(m+1,n), f(m,n-1), f(m,n+1)$ represents the values of adjacent matrix elements.

m-1, n-1	m-1, n	m-1, n+1
m, n-1	m, n	m, n+1
m+1, n-1	m+1, n	m+1, n+1

Fig. 2.4: Position of adjacent matrix elements

For realizing the resultant matrix after applying Laplacian two identical integer arrays a[M][N] and b[M][N] are taken. The value 1 (one) represents object and value 0 (zero) represents background region. Two other arrays p[M][N] and q[M][N] are taken to keep the intermediate results for rows i.e., $\frac{\partial^2 \emptyset}{\partial m^2}$ and for columns i.e., $\frac{\partial^2 \emptyset}{\partial n^2}$ respectively. The final result is obtained by adding the results $\frac{\partial^2 \emptyset}{\partial m^2}$ and $\frac{\partial^2 \emptyset}{\partial n^2}$. The final value is stored in a[M][N].

Procedure Apply-Laplacian(a[M][N])
Begin
∀i, j b[i][j] ← a[i][j], p[i][j] ← 0, q[i][j] ← 0;
For i = 1 to M
For j = 1 to N
Do begin
p[i+1][j] ← a[i+1][j] – a[i][j];
q[i][j+1] ← b[i][j+1] – b[i][j];
a[0][N] ← 0;
b[M][0] ← 0;
End
End for
End for
For i = 1 to M
For j = 1 to N
Do begin
a[i][j] ← p[i+1][j] – p[i][j];
b[i][j] ← q[i][j+1] – q[i][j];
End
End for
End for
For i = 1 to M
For j = 1 to N
a[i][j] ← a[i][j] + b[i][j];
End for
End for
Return a[M][N]
End

0	0	0	0	0	0
0	1	1	1	0	0
0	1	1	1	0	0
0	1	1	1	0	0
0	1	1	1	0	0
0	0	0	0	0	0

(a)

0	0	0	0	0	0
0	-2	-3	-2	0	0
0	-3	0	-3	0	0
0	-3	1	-3	0	0
0	-2	-3	-2	0	0
0	0	0	0	0	0

(b)

Fig. 2.5: (a) Matrix elements representing an object and its background (b) Values of the matrix after applying Laplacian Operator

Tracing the Edge of the Object: The values obtained after applying Laplacian can be used to detect the edges of the object. An image pixel can be considered as a part of edge if it itself is 1 and one or at most three of its adjacent pixels are 0. If all the four adjacent pixels is zero it is an individual particle object. Using this methodology if the value of Laplacian of a matrix element is either -1 or -2 or -3 then it is a boundary element.

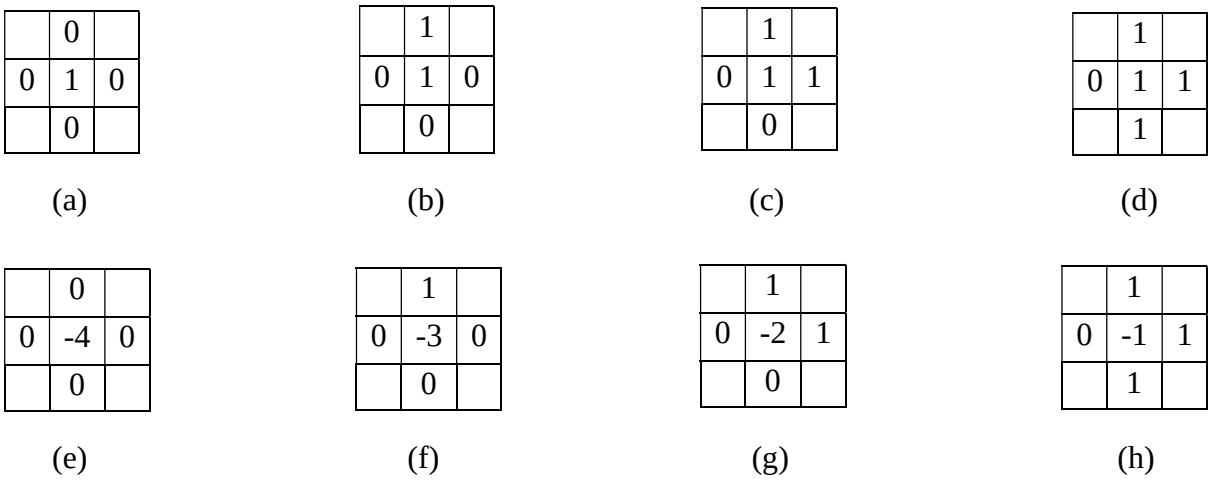


Fig. 2.6: (a), (b), (c), (d) Different scenarios for a matrix element
(e), (f), (g), (h) Corresponding Laplacian for matrix element

The values -1, -2, -3 will be taken as boundary elements and indicated as '*' correspondingly and kept in a character array. Thus, blue print of the object is obtained consisting of various edges of the object. The other characters are stored as ' ' (a blank) in the character array.

```
Procedure Edge-Tracing(a[M][N])
Begin
        For i = 1 to M
                For j = 1 to N
                        if(a[i][j] ≥ -3 & a[i][j] ≤ -1) ca[i][j] ← '*';
                        else ca[i][j] ← ' ';
                End for
        End for
        Return ca[M][N]
End
```

Fig. 2.7: (a) An image of a table (b) Extracted boundary of the table

2.5 Finding the Vertices of the Object

After boundary extraction there is need for obtaining the vertex position form the detected edges of the object.

Tracing Vertices from the Edges of Object: Vertices of the object are defined as the meet point of more than two edges or ending of an edge or two edges meeting with an angle of 180^0. Let the element considered is having indices m, n (m = row, n = column in the matrix). For checking whether a * is a vertex or not 8 (eight) neighbourhood elements are considered. The rules for vertex determination are defined below.

1. If the elements in position (m-1, n) and (m+1, n) are both '*' then element in position (m, n) is not a vertex.
2. If the elements in position (m, n-1) and (m, n+1) are both '*' then element in position (m, n) is not a vertex.
3. If the elements in position (m-1, n-1) and (m+1, n+1) are both '*' then element in position (m, n) is not a vertex.
4. If the elements in position (m-1, n+1) and (m+1, n-1) are both '*' then element in position (m, n) is not a vertex.
5. If only one of the above conditions are satisfied then element in (m, n) is not a vertex and it is kept as '*'. Otherwise, it is a vertex and it is updated to 'v' (denoting vertex) in (m, n) position of the matrix.

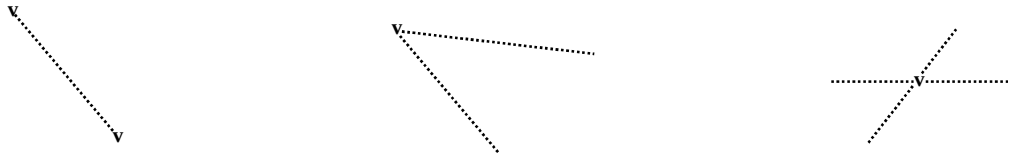


Fig. 2.8: Different examples of vertices

The output matrix having all the details of edges, vertices and background is stored in a different two-dimensional array of characters named ca[M][N].

```
Procedure Trace-Vertices(ca[M][N])
Begin
  For i = 1 to M
    For j = 1 to N
      If(ca[i][j] = '*')
        Do begin
          if (Two opposite adjacent elements of ca[i][j] = '*'
             & Other six neighbour elements of ca[i][j] ≠ '*')
             ac[i][j] ← '*';
          else ac[i][j] ← 'v';
        End
      End for
  End for
  Return ac[M][N]
End
```

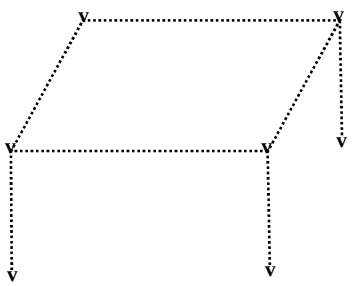


Fig. 2.9: Modified Boundary of an object including vertices

Finding Co-ordinates of Vertices and Indexing them: In previous section a character array is formed containing the vertices ‘v’ and valid edges as ‘*’. The co-ordinate of the vertices is now obtained in terms of (m, n) of the array position of the character array only when there is ac[m][n] = ‘v’. The co-ordinate along with an index for each different vertex is kept as linked list. So, each node of a linked list will contain three parameters as m, n and vertex index of a particular vertex. A linked list has two fields INFO and LINK. The LINK is connected to the next node that corresponds to the other vertex. The INFO field is divided in to three sub information fields INFO1, INFO2, INFO3 for m, n and vertex index respectively. Two procedures are used for implementing the linked list. The first procedure **Create-First-Node** creates the node which is used to point to the head of the linked list. The second procedure **Insert-Vertices** is used to insert new vertices to the linked list.

Procedure Create-First-Node( )
Begin
  node ← new; //gets a new node (an address of a linked list data type) from garbage//
  LINK(node) ← NULL;
  INFO1(node) ← 0;
  INFO2(node) ← 0;
  INFO3(node) ← 0;
  Return node
End

Procedure Insert-Vertices(m, n, index, node)
Begin
  temp ← new; //gets a new node (an address of a linked list data type) from garbage//
  LINK(temp) ← node;
  INFO1(temp) ← m;
  INFO2(temp) ← n;
  INFO3(temp) ← index;
  node ← temp;
  Return node
End

The creation of linked list for co-ordinates and indices requires a search procedure which finds the vertex from array ac[M][N] and creates the linked list.

Procedure Find-Vertices(ac[M][N])
Begin
  num ← 0;
  vertary ← Create-First-Node( );
  For ∀i, j : ac[i][j] = ‘v’
    vertary ← Insert-Vertices(i, j, num, vertary);
    num ← num + 1;
  End for
  Return vertary
End

Putting Vertices in an Array: Vertices in the linked list is put into an array. An integer array named av[i][j] is taken for this purpose. If i = M_1 is taken then it can be changed as per needs and it is the number of vertices. Now j is fixed to value 3 since there are two parameters for position namely m, n and one parameter for vertex index. The parameter m stands for y-coordinate (av[i][1] = m) and n stands for x-coordinate (av[i][2] = n) of the i^{th} vertex. The vertex indices are put according to the values of the input linked list (av[i][3] = vertex index). A temporary node "temp" is used to read the linked list store in "vertary" and map it into an array.

```
Procedure Vertices-in-Array(vertary)
Begin
  For i = 1 to M1
    For j = 1 to 3
      av[i][j] ← 0;
    End for
  End for
  i ← 1;
  temp ← vertary;
  av[i][1] ← INFO1(temp);
  av[i][2] ← INFO2(temp);
  av[i][3] ← INFO3(temp);
  While(LINK(temp) ≠ NULL)
    Begin
      i ← i + 1;
     temp ← LINK(temp);
      av[i][1] ← INFO1(temp);
      av[i][2] ← INFO2(temp);
      av[i][3] ← INFO3(temp);
    End
  End while
  Return av[M1][3]
End
```

The final model of the object after boundary extraction and vertex determination is called "Wire Grid Model".

2.6 Segmentation of Images

After the edges in an image are identified, the next major task is to segregate the image into modules of interest. The process of partitioning the image into modules of interest is informally called segmentation. The modules in the image are generally segmented based on the homogeneous features of the pixel regions and/ or the boundaries created by the connected edges in low level processing. The intermediate level requires combining the pieces of edges of contiguous regions to determine the object boundaries and then attaching a label to each of these boundaries. After segmenting an image into disjoint regions their shapes, spatial relationships and other characteristics can be described and labeled for subsequent interpretation. Typically, a region description includes size of the area, location of the center

of mass, minimum bounding rectangles, boundary contrast, shape classification number, chain code, position and type of vertices and the like.

In the previous section a wire grid model is obtained in the form of vertices and edges as ‘v’ and ‘*’ respectively. First vertex from the array of vertices is selected called the originating vertex as shown as v(0) in fig. 2.10(b). Now search is initiated on 8 directions along the ‘*’ from the originating vertex. If a ‘v’ character is having any neighbour of ‘*’ character it starts searching another ‘*’ character unless it gets a ‘v’ character or a ‘ ’ character. If it finds a ‘v’ moving along the ‘*’ then this second ‘v’ is called the searched vertex. If searching in a direction along ‘*’ is unable to find any searched vertex then search is backtracked to the originating vertex. For an originating vertex if all the 8 directions are searched then it is marked by replacing ‘v’ with ‘s’. Before reading all the vertices one by one and searching the related vertices a linked list is created where a node consists of the parameters of originating and the searched vertices. Parameters are co-ordinates of the vertices as (m, n) and the indices of the vertices. While searching around an originating vertex if it finds a searched vertex then it creates a node consisting of all parameters of both the vertices (originating and searched). After that this created node is inserted with the linked list. So finally, the output linked list will comprise all the valid segments i.e., the co-ordinates of two vertices and their indices together if only there is valid edge joining them together.

In the character array ac[M][N] where the first vertex is obtained i.e., when ac[m][n] = ‘v’ the searching in 8 directions starts. The originating vertex co-ordinate is (m, n) with an index as per the array of $av[M_1][3]$ such that $\forall l$ av[l][1] = m, av[l][2] = n, av[l][3] = index. The searching in 8 directions of the element taken as originating vertex is performed in the manner as shown in fig. 2.10(a).

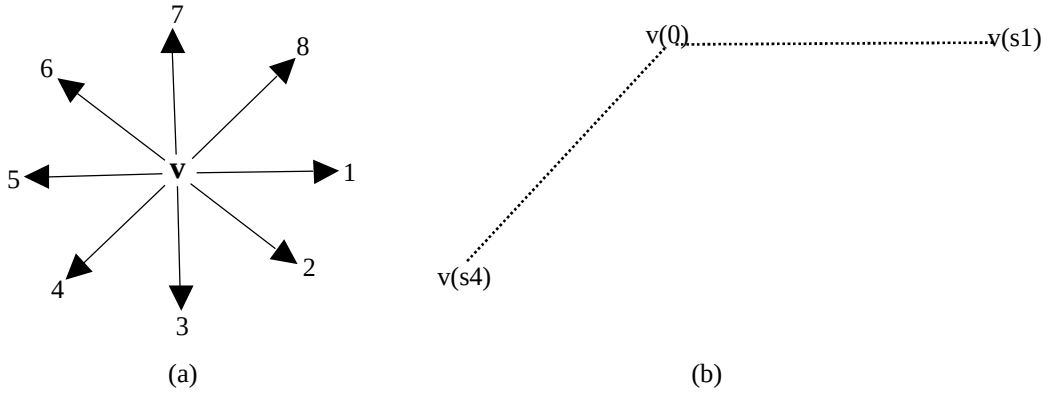


Fig. 2.10: (a) Arrows showing eight searching directions (b) A segment with originating and searched vertices

Two searched vertices are shown in direction 1 as v(s1) and in direction 4 as v(s4) searching along ‘*’. The search direction can be provided by the checking the increment and decrement in indices of the array ac[i][j] for all directions.

i-- , j--	i-- , j	i-- , j++
i , j--	i , j	i , j++
i++ , j--	i++ , j	i++ , j++

Fig. 2.11: Corresponding increment and decrement in array for a particular direction

Searching Direction	Increment on		Decrement on	
	i	j	i	j
1	No	Yes	No	No
2	Yes	Yes	No	No
3	Yes	No	No	No
4	Yes	No	No	Yes
5	No	No	No	Yes
6	No	No	Yes	Yes
7	No	No	Yes	No
8	No	Yes	Yes	No

Fig. 2.12: Table obtained by combining fig. 2.10(a) and fig. 2.11

The search as per above table in each 8 directions is valid until there is '*' character in that direction as per the wire grid model in the array ac[M][N] while plotted in M x N matrix form. Now as we search in 8 directions the search can provide up to maximum of 8 searched vertices for an originating vertex. All of the vertices are stored in form of pairs of originating and searched vertices along with their vertex indices. The vertex indices are obtained by comparing the vertex co-ordinate kept in array av[M1][3]. For example, if the co-ordinate of an originating vertex be (2, 4) and two searched vertices in direction 1 and 4 be (2, 10) and (10, 4) respectively. The indices of the vertices are 3, 0, and 1 respectively. So, there are two segments [(2, 4), (2, 10)] and [(2, 4), (10, 4)]. The representation of the indices is in the form as [3, 0] and [3, 1] respectively. All of this information is stored in a linked list as given in fig. 2.13.

Fig. 2.13: Linked showing a segment as pair of originating and searched vertices

There might be a case when in a different segment while searching the originating vertex of another segment may appear as searched vertex, thus there is a repetition of segment. To avoid this, the originating vertex after being searched in 8 directions is marked 's' in place of 'v' i.e., in array ac[M][N] which indicates searching is finished.

```
Procedure Search-Dir_1(ac[M][N])
Begin
  Do
    j ← j + 1;
  While(ac[i][j ] = '*')
  Return i, j
End

Procedure Search-Dir_2(ac[M][N])
Begin
  Do
    i ← i + 1;
    j ← j + 1;
  While(ac[i][j] = '*')
  Return i, j
End
```

```
Procedure Search-Dir_3(ac[M][N])
Begin
  Do
    i ← i + 1;
  While(ac[i][j] = '*')
  Return i, j
End

Procedure Search-Dir_4(ac[M][N])
Begin
  Do
    i ← i + 1;
    j ← j - 1;
  While(ac[i][j] = '*')
  Return i, j
End

Procedure Search-Dir_5(ac[M][N])
Begin
  Do
    j ← j - 1;
  While(ac[i][j] = '*')
  Return i, j
End

Procedure Search-Dir_6(ac[M][N])
Begin
  Do
    i ← i - 1;
    j ← j - 1;
  While(ac[i][j] = '*')
  Return i, j
End

Procedure Search-Dir_7(ac[M][N])
Begin
  Do
    i ← i - 1;
  While(ac[i][j] = '*')
  Return i, j
End
```

```
Procedure Search-Dir_8(ac[M][N])
```

```
Begin
  Do
    i ← i - 1;
    j ← j + 1;
  While(ac[i][j] = '*')
  Return i, j
End
```

The procedure Get-Indices returns the indices of vertices from av[M_1][3] and also sets two flag flag1 and flag2. The originating and searched vertex co-ordinates are (m, n) and is (i, j).

```
Procedure Get-Indices(av[M1][3], m, n, i, j)
Begin
  flag1 ← 0, flag2 ← 0, ind ← 100, ind1 ← 100;
  For l = 1 to M1
     if(m = av[l][1] & n = av[l][2])
       begin
         flag1 ← 1;
         ind ← av[l][3];
       end
     if(i = av[l][1] & j = av[l][2])
       begin
         flag2 ← 1;
         ind1 ← av[l][3];
       end
  End for
  Return flag1, flag2, ind, ind1
End
```

The procedures Create-Linked-List and Insert-Segment creates linked list for a segment as pair of originating and searched vertices.

```
Procedure Create-Linked-List( )
Begin
  node ← new;
  LINK(node) ← NULL;
  INFO_X(node) ← 0;
  INFO_Y(node) ← 0;
  INFO_X1(node) ← 0;
  INFO_Y1(node) ← 0;
  INFO_IND(node) ← 0;
  INFO_IND1(node) ← 0;
  Return node
End
```

```
Procedure Insert-Segment(m, n, i, j, av[M1][3], node)
```

```
Begin
  temp ← new;
  LINK(temp) ← node;
  INFO_X(temp) ← m;
  INFO_Y(temp) ← n;
  INFO_X1(temp) ← i;
  INFO_Y1(temp) ← j;
  flag1, flag2, ind, ind1 ← Get-Indices(av[M1][3], m, n, i, j);
  INFO_IND(temp) ← ind;
  INFO_IND1(temp) ← ind1;
  node ← temp;
  Return node
End
```

A procedure Find-Segments which takes ac[M][N] and av[M_1][3] as input and gives a linked list containing a valid segment as two vertices along with their vertex indices ind, ind1, with head as 'partary'.

```
Procedure Find-Segments(ac[M][N], av[M1][3])
Begin
  partary ← Create-Linked-List( );
  For m = 1 to M
    For n = 1 to N
      if(ac[m][n] = 'v')
        begin
          i ← m;
          j ← n;
          For k = 1 to 8
            i, j ← Procedure Search-Dir_k(ac[M][N]);
            if(ac[m][n] = 'v') then partary ← Insert-Segment(m, n, i, j, av[M1][3], partary);
          End for
        end
    End for
  End for
  Return partary
End
```

The valid segments co-ordinates as obtained in wire grid model is found out and kept in the linked list.

2.7 Minimal Representation of Geometric Primitives

For estimation of parameters of 2-D lines, 3-D points, 3-D lines and 3-D planes, they are represented with minimal parameters. Further the selected representation should be differentiable, so that the principles of Linear Kalman filtering can be applied.

Representation of Affine lines in R^2: A 2-D line can be represented by at least two independent parameters. The simplest form of representation of a 2-D line is given by the

following expressions.

In brief, the representation of a 2-D line is given by a vector (a, p), where the line passing through (0, 0) and (a, 1) is normal to the line under consideration, which also passes through the point (0, -p). This is illustrated in fig. 2.14.

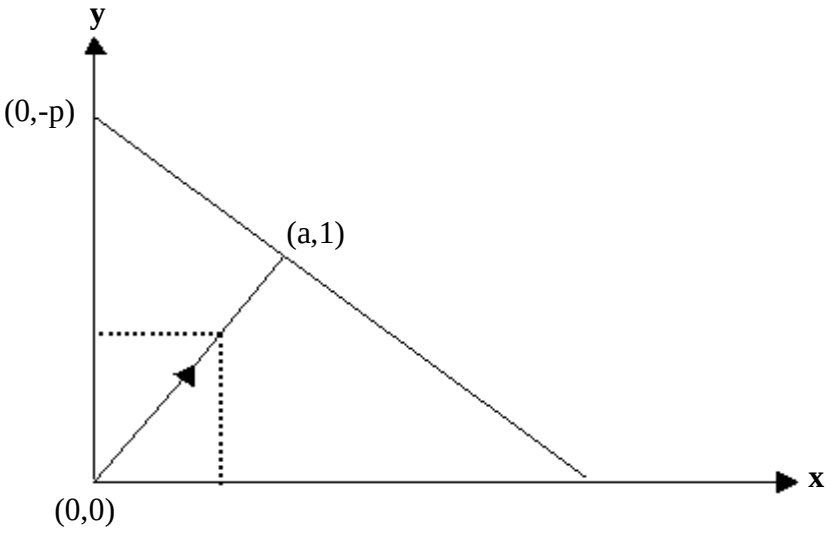


Fig. 2.14: 2D line representation by (a, p)

This representation is more preferable because the equation of line is linear in parameter (a, p) which is useful for Kalman filtering.

Representation of Affine lines in R^3: The 3-D affine line can be represented minimally by four parameters given by (a, b, p, q). Other minimal representations are possible but there exists scope of ambiguity in other representations. The representation is preferred by the following counts:

1. It imposes no constraints on the parameters (a, b, p, q).
2. Parametric representations of the lines remain linear, which are advantageous to Kalman filtering optimization.

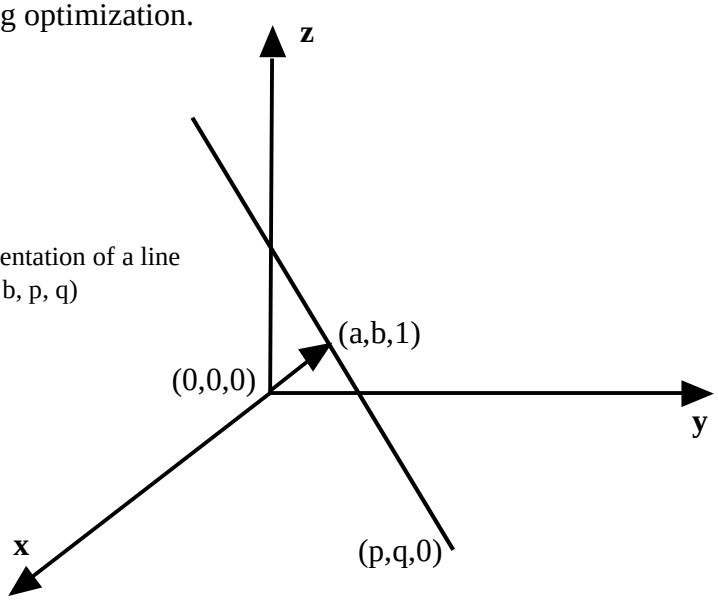


Fig. 2.15: A 3-D representation of a line by (a, b, p, q)

Representation of Affine planes in R^3: One way of representing 3-D planes is by a 3-D vector (a, b, p) such that points (x, y, z) of the plane are defined by the following equation.

a x + b y + z + p = 0

Here the vector $(a, b, 1)^T$ is the normal to the plane and the point $(0, 0, -p)^T$ is the point of intersection of the plane with the z-axis.

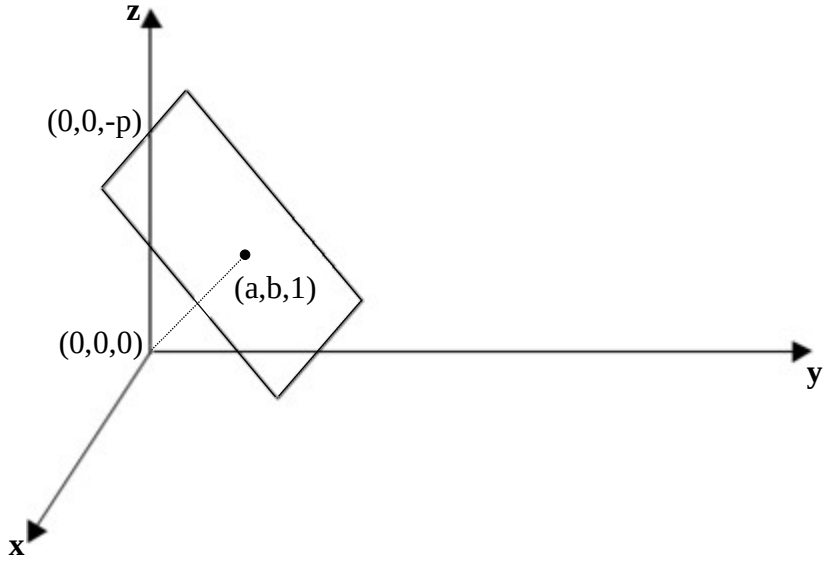


Fig. 2.16: A 3-D plane representation by (a, b, p, q)

The limitation of this notation is that planes parallel to the Z axis cannot be represented. More formally, we have three cases:

Case I: Planes not parallel to the Z axis is given as **a x + b y + z + p=0**
Case II: Planes not parallel to the X axis is given as **x + a y + b z + p = 0**
Case III: Planes not parallel to the Y axis is given as **b x + y + a z + p =0**

2.8 Kalman Filtering and its Application

A Kalman filter is a digital filter that attempts to minimize the measurement noise from estimating the unknown parameters, linearly related with a set of measurement variables. The most important significance of this filter is that it allows recursive formulation and thus improves accuracy of estimation up to users' desired level at the cost of new measurement inputs. Let

$f_i(\mathbf{x_i}, \mathbf{a}) = 0$ be a set of equations describing relationships among a parameter vector **a** and measurement variable vector $\mathbf{x_i}$,

$\mathbf{x_i}^* = \mathbf{x_i} + \mathbf{l_i}$, such that $E[\mathbf{l_i}] = 0$, $E[\mathbf{l_i}\,\mathbf{l_i}^T]$ = positive symmetric matrix $\mathbf{\Lambda_i}$, and $E[\mathbf{l_i}\,\mathbf{l_j}^T] = 0$,

$\mathbf{a_{i-1}^* = a + S_{i-1}}$, such that $E[\mathbf{S_{i-1}}] = 0$, $E[\mathbf{S_{i-1}\, S_{j-1}^T}]$ = positive symmetric matrix $\mathbf{S_{i-1}}$, $E[\mathbf{S_{i-1}\, S_{i-1}^T}] = 0$.

Expanding $f_i(\mathbf{x_i, a})$ by Taylor's series around $(\mathbf{x_i^*, a_{i-1}})$ we get,

$f_i(\mathbf{x_i, a}) = f_i(\mathbf{x_i^*, a_{i-1}^*}) + \mathbf{(\partial f_i / \partial x)}\,(\mathbf{x_i - x_i^*}) + \mathbf{(\partial f_i / \partial a)}\,(\mathbf{a - a_{i-1}^*}) = 0$

After some elementary algebra we get,

$\mathbf{y_i = M_i\, a + w_i}$, where $\mathbf{y_i} = -\, f_i(\mathbf{x_i^*, a_{i-1}}) + \mathbf{(\partial f_i / \partial a)}\,(\mathbf{a - a_{i-1}^*})$ is a new measurement vector of dimension (p_i x 1), $\mathbf{M_i} = \mathbf{(\partial f_i / \partial a)}$ and $\mathbf{w_i} = \mathbf{(\partial f_i / \partial x)}\,(\mathbf{x_i - x_i^*})$ is a measurement noise vector of dimension (p_i x 1).

We also want that $E[\mathbf{w_i}] = 0$ and define,

$\mathbf{W_i} = E[\mathbf{w_i\, w_i^T}] = \mathbf{(\partial f_i / \partial x)\, \Lambda_i\, (\partial f_i / \partial x)^T}$

Let $\mathbf{S_i} = E[(\mathbf{a_i - a_i^*})\,(\mathbf{a_i - a_i^*})^T]$. An attempt to minimize $\mathbf{S_i}$ yields the filter equations, given by:

$$\mathbf{a_i^* = a_{i-1}^* + K_i\,(y_i - M_i\, a_{i-1}^*)}$$

$$\mathbf{K_i = S_{i-1}\, M_i^T\,(W_i + M_i\, S_{i-1}\, M_i^T)^{-1}}$$

$$\mathbf{S_i = (I - K_i\, M_i)\, S_{i-1}}$$

Given $\mathbf{S_0}$ and $\mathbf{a_0}$, the Kalman filter recursively updates $\mathbf{a_i}$, $\mathbf{K_i}$, $\mathbf{S_i}$ until the error covariance matrix $\mathbf{S_i}$ becomes insignificantly small, or all the number of data points have been submitted. The $\mathbf{a_i}$ obtained after termination of the algorithm is the estimated value of the parameters.

The Kalman filter can be used for determining

1. Affine 2-D lines from a set of noisy 2-D points.
2. 3-D points from a set of noisy 2-D points.
3. Affine 3-D lines from noisy 2-D points.
4. 3-D planes from 3-D lines.

In the next section the formation of 2-D lines and 3-D lines from a set of noisy points by Kalman filtering is discussed.

Construction of 2-D Lines from Noisy 2-D Points: The filter equations are applied for the construction of affine 2-D lines from noisy 2-D points. Here, given the set of points $\mathbf{x_i^*} = (u_i^*, v_i^*)$, we have to estimate the parameters $\mathbf{a} = (a, p)^T$. The $f_i(\mathbf{x_i, a})$ in the present context is given by $f_i(\mathbf{x_i, a}) = a\, u_i + v_i + p = 0$.

Now, $\mathbf{y_i = M_i\, a + w_i}$, where $\mathbf{y_i} = -\, f_i(\mathbf{x_i^*, a_{i-1}}) + \mathbf{(\partial f_i / \partial a)}\,(\mathbf{a - a_{i-1}^*}) = \mathbf{-\, v_i}$ and $\mathbf{M_i} = \mathbf{(\partial f_i / \partial a)} = (u_i, 1)$

The measurement noise $\mathbf{w_i}$ is given by, $\mathbf{w_i} = \mathbf{(\partial f_i / \partial x)}\,(\mathbf{x_i - x_i^*})$, where $\mathbf{(\partial f_i / \partial x_i) = [a_{i-1}^*, 1]}$

The covariance matrix $\mathbf{W_i}$ is given by $\mathbf{W_i = (\partial f_i / \partial x_i)\, \Lambda_i\, (\partial f_i / \partial x_i)^T}$, where $\Lambda_i = 1$.

A procedure can be developed which can take input of co-ordinates of image points and provides the state estimate A(2x1), along with the covariance error S(2x2) as output.

Procedure Construct 2-D-Lines (2D-image-points: u, v)
Begin
Initialize the Initial Covariance matrix S and the state estimate:
$S_0 \leftarrow$ Very large initial value;
$a_0 \leftarrow$ Arbitrary value preferably $[0\ 0\ 0]^T$;
For(no-of-points = 1 to n)
Compute the perspective matrix from the input camera parameters;
Compute the measurement vector **y**(1x1), the linear transformation **M**(2x1) obtained after linearizing measurement equation from input parameters at each iteration;
Initialize all the matrices involved in matrix multiplication;
Compute gain **K** using previous **S** and **M** values ;
Compute covariance matrix **S** recursively using its value at previous iteration;
Compute the state estimate **a** recursively using its value at previous iteration;
End for
End

Construction of 3-D Points Using 2-D Image Points: The 3-D object points are mapped onto an image plane by using the principle of perspective projection. Let the 3-D object point be P having co-ordinates $(x, y, z)^T$, which is mapped onto the image plane at point $(U, V, S)^T$. Let T be the perspective projection matrix. Then,

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where t_{ij} is the (i, j)th element of the perspective projection matrix. Let $u = U/S$ and $v = V/S$. Now, after elementary simplification let assume that $t_i = (t_{i1}\ t_{i2}\ t_{i3}\ t_{i4})^T$ and P is $(x, y, z)^T$ also assume that $a = (t_1\ t_{14}\ t_2\ t_{24}\ t_3\ t_{34})^T$. For a match of an image point I with an associated scene point P, we now have the following relationships between P, u and v.

$P^T t_1 + t_{14} - u(P^T t_3 + 1) = 0$ and $P^T t_2 + t_{24} - v(P^T t_3 + 1) = 0$

Now, suppose we have $\mathbf{x_i} = (u_i, v_i)^T$ and we have to evaluate $\mathbf{a} = (x, y, z)^T$. The measurement equation is given by,

$f_i(\mathbf{x_i, a}) = 0$ yields $(\mathbf{t_1^i} - u_i\mathbf{t_3^i})\ \mathbf{a} + t_{14}^i - u_i\ \mathbf{t_{34}^i} = 0$ and $(\mathbf{t_2^i} - v_i\mathbf{t_3^i})\ \mathbf{a} + t_{24}^i - v_i\ \mathbf{t_{34}^i} = 0$ where, $\mathbf{t_j^i}$ comes from perspective matrix $\mathbf{T_i}$ from camera **i**.
Further, $\mathbf{y_i} = \mathbf{M_i} + \mathbf{w_i}$, where

$$\mathbf{y_i} = \begin{bmatrix} \mathbf{t_{34}^i}\ \mathbf{u_i^*} - \mathbf{t_{34}^i} \\ \mathbf{t_{34}^i}\ \mathbf{v_i^*} - \mathbf{t_{24}^i} \end{bmatrix}, \mathbf{M_i} = \begin{bmatrix} -(\mathbf{u_i}\ \mathbf{t_3^i} - \mathbf{t_1^i})^T \\ -(\mathbf{v_i}\ \mathbf{t_3^i} - \mathbf{t_2^i})^T \end{bmatrix} \text{ and } \frac{\partial \mathbf{f_i}}{\partial \mathbf{x_i}} = \begin{bmatrix} -\mathbf{t_3^i a_{i-1}^*} - t_{34}^i & 0 \\ 0 & -\mathbf{t_3^i a_{i-1}^*} - t_{34}^i \end{bmatrix}$$

A procedure can be developed which takes co-ordinates of the image points along with the six camera parameters determined by its position (x_0, y_0, z_0) and orientation (A, B, C) w.r.t global co-ordinate system as input and provides the state estimate **a** (3 x 1) along with the co-variance error **S** (3 x 3) associated with the estimate.

Procedure 3-D-Point-Construction(2-D image points: u, v; Camera parameters: x_0, y_0, z_0, A, B, C)
Begin
For(no. of points: = 1 to n)
Initialize the Initial Covariance matrix **S** and the state estimate
$\mathbf{S_0} \leftarrow$ Very large initial value;
$\mathbf{a_0} \leftarrow$ Arbitrary Value preferably $[0\ 0\ 0]^T$;
For (j: =1 to no. of iterations)
Compute the perspective matrix from the input camera Parameters;
Compute the measurement vector **y** (2 X 1), the linear transformation **M** (2 x 3) and **W** (2 x 2) the weight matrix obtained after linearizing measurement equation from the input parameters at each iteration;
Initialize all the matrices involved in matrix multiplication;
Compute gain **K** using previous **S** and **M** values;
Compute covariance matrix **S** recursively using its value at previous iteration;
Compute the state estimate '**a**' recursively using its value at previous iteration; j←j + 1;
End for
End for
End

2.9 Summary

Vision systems can be functionally classified into three main levels, namely, low, medium and high-level vision. AI is mostly used in the high-level vision. The high-level vision mainly deals with recognition and interpretation of 3-D objects from their 2-D images. There exist many approaches to interpret a scene from more than one image. Kalman filtering is one of such techniques. Its main advantage is that it employs a recursive algorithm and thus can update an estimator from input data in an incremental fashion. The vertices of points in a 2-D image can be first mapped to their 3-D locations by supplying the same 2-D points from multiple images. The spatial relationships among the planes are then analyzed to determine the 3-D planer object.

3

Spatial Representation

3.1 Introduction

For using the know-how of space and do operations like reasoning with the knowledge it is required to have a proper presentation. There are two usual options to represent spatial knowledge: the qualitative and the quantitative (numerical) methods. The distinct aspects of space that are represented by different kinds of spatial relations. The one-dimensional temporal reasoning will be discussed briefly as it had huge impact on qualitative spatial reasoning. Topology characterization are by default qualitative whereas orientation and distance knowledge can also be determined quantitatively. Positional knowledge consists of both distance and orientation. The discussed aspects of space are mostly dependent on one another, especially if orientated objects with extension are present.

Table 3.1: Thirteen Relationships

Relation	Symbol	Inverse Symbol	Example
X equal Y	=	=	XXX YYY
X before Y	<	>	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYY

Temporal Logic: Temporal logic is composed of time intervals. The requirements for the algorithm are: It should permit imprecision (instead of absolute values relative relations are derived) and uncertainty of knowledge (restriction between two instants of times may exist although the connection between two instants may be unknown). Also, the algorithm should differ in its frame of reasoning (years, days, milliseconds etc.) and it should allow continuity (assist default reasoning). Intervals of time are used because time itself can't be decomposed further. All the known events in time might be decomposed up into two or more separate events thus creating an interval of time. In table 3.1 the thirteen possible relationships between intervals are noted down. Convenience allows the collapse of the three during relations (d, s, f) into one relationship called dur and the three containment relations (di, si, fi) into a relationship called con. The time intervals forms nodes of a network, while the arcs between those nodes are marked with one or more of those thirteen relationships (allowing unpredictability for disjunction of relations). An algorithm to calculate the transitive closure of such a network can be done using a "transitivity table" or "composition table". Introduction of reference intervals to group collection of intervals decreases the need of space requirements of the representation without hugely affecting the deductive power of his system.

Studies have been done to augment temporal logic into spatial dimensions like using the temporal distinctions for the two axes of a Cartesian coordinate system. These methods lack analytical reasoning because humans don't observe the world into two axes nor performs operations on each of them. It is known that there is (although both are one dimensional) significant contrast between time and a one-dimensional space. The most specific one is, that time always progress forward whereas space has no permanent direction. Thus, spatial representations immensely differ from temporal logic.

Topology: Topological dissimilarities between various spatial entities are a basic feature of spatial knowledge. The dissimilarities are mostly qualitative, so there is always an interest for qualitative spatial reasoning. Topological representations most of the time depict relationships between spatial regions which are subsets of some topological space. Most methods that formalize topological features of spatial regions utilize a single primitive relation (the binary connectedness relation) to create many other relations. In topology, formal definitions like neighborhood, interior, exterior, boundary etc. can be used to define relations like disjoint,

meet, overlap, contain etc. for two regions.

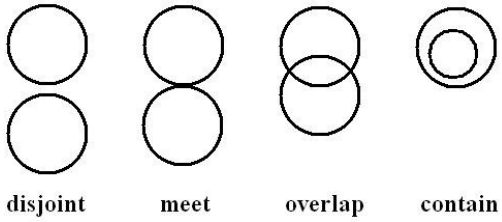


Fig. 3.1: Examples for topological relations

Orientation: Orientation is also a very important aspect for spatial representation and reasoning. Relative orientation of spatial entities is a ternary relationship depending on the referent, the relatum and the frame of reference (reference system) which can be specified either by a third object (origin) or by a pre-defined vector. The orientation is decided by the position in which the referent is situated in terms of the frame of reference. Three reference systems can be distinguished:

1. In the **Intrinsic Reference System**, the orientation is decided upon by a built-in feature of the relatum.
2. The **Absolute Reference System** (also called extrinsic reference system) uses external framework like global coordinate systems to obtain the orientation of relatum.
3. A third object known as origin serves in the **Relative Reference System** (also called deictic reference system) so that the orientation is obtained from the "point of view" of the third object in the reference system.

Most spatial representations use points in a two-dimensional space to represent spatial objects since developing spatial orientations between extended objects is much more difficult.

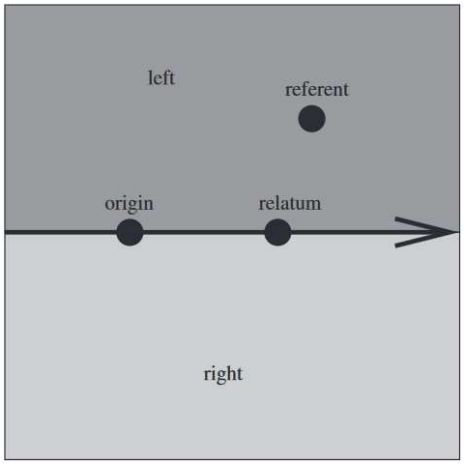


Fig. 3.2: The left/right decision making in a relative reference system

Distance: The scalar quantity distance combined with topology and orientation, one of the most important feature sets of space. For human reasoning qualitative explanations of distance are used. There are absolute distance relations that mentions distances between two objects like "close to", "far" or "very far". There also are relative distance relations that compares the distance between objects with the distance to a third object like "closer than" or "further than".

Qualitatively expressed absolute distances depend on the relative size of space (for example plates on a desk, in a school or in a town). Various distance relations not only depend on distance but also on orientation. If an entity A is far from B and another entity C is also far from B, we cannot say anything definite regarding the distance between A and C. If the entities are on an imaginary line in the sequence of A B C, A would be very far from C, but if they were on the imaginary line in the order of A C B then A could be close to C. It is thus useful to use distance in fusion with orientation which is called positional information.

Positional information: The expression "positional information" refers to information system that combine the orientation and distance representation i.e., the position of an object is characterised by a combination of a qualitative orientation and a qualitative distance. The total number of possible relations (atomic relations) is the product of number of possible relations of the orientation representation with the number of relations of the distance representation. This might not be accurate for all cases, for example if in a relation where objects are on the same position. The term "positional information" can also be used for combinations of orientation information with topological information.

Orientated objects with extension: All real-world objects have an extended portion. If spatial representations could make use of this information enhancement can be attained for some applications. The extension is usually mentioned by size and shape of the objects. The shape of an object could change the frame of reference of a relation in an intrinsic reference system to determine the orientation of this object. The size can have impact on how humans describe spatial relations because they prefer large, salient objects as reference objects for example humans rather say "The post-box is in front of the house" than "The house is behind the post-box". Difficulties with extended objects arise if they overlap because two orientations could then refer to the same configuration.

Shape: Shape is something difficult to express qualitatively. Using topology, it can be determined whether an object has holes or whether it is one single object or not. For finer grained differentiation shape primitives could be applied. Other ways extract the boundary of an object using a chain of different types of boundary segments or curvature parameters. Adding to that, shape could be mentioned in terms of polygons that qualitatively define each vertex corner whether it is convex or concave, whether it is obtuse or right-angled or acute and with a qualitative description of the direction of the corner. Shape representations can make statements only about the boundary or about the internal body of an object.

Motion: Motion is a spatio-temporal phenomenon that can be quite easily expressed qualitatively. In this method, the motion of an object is viewed from a deictic point of view with a fixed frame rate. The two constituents i.e., orientation and distance, that establish a vector, are used to describe the movement of the object from one frame to the other. If the object is standing and fixed at one position then the orientation and distance are 0. If two successive qualitative motion vectors are equal an index is increased by one. A motion could be in form (distance, orientation). For example (close, forward)^{5}, (close, left)^{9}, (0, 0)^{6}, (far, forward)^{5}. Another way of representation is one with an intrinsic frame of reference given by the orientation of the previous vector (the first vector is always "forward"). Now the frame rate is only used for halted object, in other cases a new qualitative value, the velocity, is used. A motion would now look like this: (short-dist, forward, slow), (medium-dist, left, slow), (0, 0)^{6}, (far, right, fast).

An abstract representation is a propositional one which identifies a set of movement shapes like "straight-line", "left-turn", "right-turn", "u-turn", "loop", etc. and can have relations between them that describe distinctions in magnitude, orientation and velocity.

3.2 Structure of Spatial Reasoning

Reasoning should be done on back of effective algorithms. Brute-force "generate-and-test" method is always present but most often unsuitable due to its complexity. Reasoning can accomplish many tasks. Most significantly it can deduce knowledge which is implicit in the knowledge base and make it consequently explicit. In this way the knowledge base is expanded. Reasoning can also reply to queries that are given with only incomplete knowledge and with a precise condition. Different kinds of reliability can be supported using reasoning techniques that streamline the use of the knowledge base. In general reasoning is used to gain and process new knowledge.

Algebraic Structure: In this topic some basic reasoning methods of spatial representations are introduced. It is assumed that the relations between the objects that are being handled are ternary. Ternary relations are taken over three objects: the first argument is the origin, the second one is the relatum and the third one the referent. The relation expresses the position of the referent with respect to the frame of reference (or reference system) determined by the origin and the relatum. In qualitative representations each representation has a fixed number of atomic relations corresponding to the fixed number of spatial alignments that are characterised by the representation. The relation in terms of the three objects corresponds to the spatial configuration of those objects. Special instances occurs when two of the three objects are at the same position. In order for the algebra to categorize those cases they are introduced the set of atomic relations. In broad sense, the three objects split the space and the set of atomic relations is given by the divisions.

General Relation: A (general) relation is any subset of the set of all atomic relations. The analysis of such a relation R is given as: $(\forall X, Y, Z)\ (R(X, Y, Z) \Leftrightarrow \vee_{r \in R}\ r(X, Y, Z))$.
The atomic relations are constructed to be Jointly Exhaustive and Pairwise Disjoint (JEPD) i.e., for any three given objects X, Y, Z, there exists one and only one atomic relation r such that r(X, Y, Z).

Unary Operations: The ternary relations have three arguments and thus there are 3! = 6 possible ways for the arrangement of those arguments. Following the expression shown in table 3.2 is used for those transformations.

Table 3.2.: Transformations

Term	Symbol	Permutation
identical	I_D	$(\forall X, Y, Z)\ (R(X, Y, Z) \Longrightarrow\ I_D\ (R(X, Y, Z)))$
inversion	I_{NV}	$(\forall X, Y, Z)\ (R(X, Y, Z) \Longrightarrow\ I_{NV}\ (R(X, Y, Z)))$
short cut	S_C	$(\forall X, Y, Z)\ (R(X, Y, Z) \Longrightarrow\ S_C\ (R(X, Y, Z)))$
inverse short cut	S_{CI}	$(\forall X, Y, Z)\ (R(X, Y, Z) \Longrightarrow\ S_{CI}\ (R(X, Y, Z)))$
homing	H_M	$(\forall X, Y, Z)\ (R(X, Y, Z) \Longrightarrow\ H_M\ (R(X, Y, Z)))$
inverse homing	H_{MI}	$(\forall X, Y, Z)\ (R(X, Y, Z) \Longrightarrow\ H_{MI}\ (R(X, Y, Z)))$

Binary Operations: The **composition** $R_1 \otimes R_2$ of two R_1 and R_2 is the relation R_j such that: $(\forall X, Y, Z, W)\ (R_1(X, Y, Z) \wedge R_2(X, Z, W) \Longrightarrow R_j(X, Y, Z))$

Given four objects X, Y, Z, W and two atomic relations r_1 and r_2 the conjunction $r_1(X, Y, Z) \wedge r_2(X, Z, W)$ is trivially inconsistent if either of the following holds:

$(\forall X, Y, Z, W)\ ((r_1(X, Y, Z) \Longrightarrow (Z = X)) \wedge (r_2(X, Z, W) \Longrightarrow (Z \neq X)))$

$(\forall X, Y, Z, W)\ ((r_1(X, Y, Z) \Longrightarrow (Z \neq X)) \wedge (r_2(X, Z, W) \Longrightarrow (Z = X)))$

For atomic relations for which the above holds the result of the composition is empty: $r_1 \otimes r_2 = \emptyset$

In general, a **compositional inference** is a abstraction from **two relational** inference of the form $R_1(a, b)$ and $R_2(a, b)$ to a relational inference of the form $R_3(a, c)$, involving only a and c. The logic of compositional inferences in many occurrences does not depend on the specific elements present but only on the logical properties on the relations. In that case the composition of pairs of relations can be abstracted by table look up when required. Given the set of n atomic relations, one can store in a $n \times n$ composition table the relationships between x and z for a pair of relations $R_1(x, y)$ and $R_2(y, z)$. In general, each entry can be a disjunction of the base relations. For the **ternary relations** those tables have to be made available for the following two cases:

$(\forall X, Y, Z, W)\ ((r_1(X, Y, Z) \Longrightarrow (Z = X)) \wedge (r_2(X, Z, W) \Longrightarrow (Z = X)))$ and

$(\forall X, Y, Z, W)\ ((r_1(X, Y, Z) \Longrightarrow (Z \neq X)) \wedge (r_2(X, Z, W) \Longrightarrow (Z \neq X)))$

For a representation using a fixed set of binary relations, the lucidity of the compositional inference makes it a good choice of effective reasoning.

The **intersection** of two relations R_1 and R_2 is the relation R which consists of the set theory intersection of the atomic relation sets from R_1 and R_2 : $R = R_1 \cap R_2$.

Constraint Based Reasoning: The constraint satisfaction problem is an abstract formulation of many difficult problems in Artificial Intelligence. Given that qualitative representations can be expressed in form of relations, general constraint satisfaction methods can be applied for various types of inference. Constraint satisfaction techniques play a major role in Computer Science and in Artificial Intelligence. In particular many difficult problems involving search from areas such as robot navigation, temporal reasoning, graph algorithms and machine design and manufacturing can be recognized as special cases of the **constraint satisfaction problem (CSP)**. Constraint Satisfaction Problems (CSP) generally consist of:

1. a set of variables $X = \{X_1, \ldots, X_n\}$,
2. a discrete domain for each variable $\{D_1, \ldots, D_n\}$ so that each variable X_i has a finite set D_i of possible values,
3. a set $\{R_k\}$ constraints, defined over some subset of the variable domains, $R_j \subseteq D_{i1} \times \ldots \times D_{ij}$, and showing the mutually compatible values for a variable subset $\{X_{i1}, \ldots, X_{ij}\}$. Those constraints are restricting the values the variables can simultaneously take.

The hurdle is to assign values to the variables such that all constraints are satisfied. Alternatives of the hurdle are to find all such assignments, the best one, if there exists any at all, etc.

The constraints are confined to be unary or binary because those are the operations defined for the spatial representation. Furthermore, it is presumed that all variable domains have the same cardinality.

The elementary approach to find a reasonable assignment is a backtracking algorithm that corresponds to an unknowledgeable systematic search. After initialising all variables applicable to a set of constraints if any of them is not valid then the algorithm backtracks to the most recently initialised variable that still has untried values available. The complexity of the algorithm is exponential and thus makes it useless for realistic input sizes. This inefficiency arises because the same computations are done multiple times.

The possible solutions to this problem are the alteration of the search space, the use of heuristics to define the search or the use of the particular problem structure to align the search. The latter one is of particular interest for spatial reasoning as will be discussed later.

Consistency improvement: The aim of modifying the search space is to reduce useless computation without eliminating any of the solutions of the original space. In other words, we are finding for a smaller but equivalent search space. This can be attained either before the search (by increasing the steadiness of the network through constraint propagation) or, in hybrid algorithms, during the search. One way of decreasing the number of repeated computations is constraint propagation. There those values are eliminated from a domain that do not satisfy the corresponding unary predicates, as well as those values for which no matching value can be found in the adjoining domains such that the corresponding binary predicates are satisfied. The former process achieves node consistency, the latter arc consistency. Expressed differently it can be said that this process of constraint relaxation is triggered by unsatisfied constraints. This idea of local consistency can be generalized to any number of variables. A set of variables is k-consistent if for each set of $k - 1$ variables with satisfying values, it is possible to find a value for the kth variable such that all constraints among the k variables are satisfied. A set of variables is strong k-consistent if it is j-consistent for all $j \leq k$. There is special interest for strong 3-consistency, which is equivalent to arc consistency plus path consistency. A network is path consistent if any value pair allowed in R_{ij} is also allowed by any other path from i to j. The process of achieving consistency in a network of constraints is called constraint propagation. Several authors give different observations about constraint propagation, whose equivalence is not prominent.

One way of defining is that constraint propagation is a method of deriving stronger (i.e., more restrictive) constraints by analysing sets of variables and their related constraints. The value elimination consistency procedures given above assume a very common extensional form of constraints as sets of satisfying value pairs. Whenever a value of the domain of a variable involved in more than one constraint is deleted, a satisfying value pair might have to be deleted from one or more of the other constraints, thus making them more limiting.

Another way of defining constraint propagation is as a process of making implicit constraints explicit. Implicit constraints are defined as constraints which are not recorded directly in compatible value pairs, but implied by them. This, however, can be seen as a side effect of the consistency procedures. The universal constraint, that permits any value pair, holds implicitly between two variables which are not explicitly linked together. If the domains of the two variables X_i and X_j are limited to a few values, then the implicit constraint R_{ij} can be made explicit as the set of conjugation of the two domains.

Note that the global full constraint propagation is same as finding the minimal graph of the CSP, where each value allowed by any explicit constraint belongs to at least one problem

solution. Full constraint propagation is thus as hard as the CSP itself (in general NP-complete). The local constraint propagation techniques used to achieve node-, arc- and path-consistency have polynomial complexities and can be used as pre-processors that substantially reduce the need to backtrack during the search for a global solution. Unfortunately, arc- and path consistency do not eliminate in general the need to backtrack during the search because constraints are propagated only locally.

3.3 Qualitative Spatial Calculi

Qualitativeness can be best described in contrast with its parallel Quantitativeness. **Quantity** is a specified or uncertain number or amount. It is easily obtained measurable, countable, or comparable property of a thing. **Quality** is the abstract character of something. It is inherent or determining characteristic.

The most crucial words for quantity are number and measurable, because spatial quantities are measured which suggests that a number is assigned to represent a magnitude. Usually, the measurement is provided by a simple comparison. The measure of the quantity is compared to a standard quantity, the measure of which is arbitrarily chosen to have the value 1.

The term quality is more tough to explain. The qualitative representations can be described by establishing a correlation between the abstract entities in the representation and the actual magnitudes. Quantitative knowledge is obtained whenever a standardized scale is used for representing the measured values relative to a standard value. The use of a scale is also the context in which issues of granularity and resolution are valuable, since a scale defines a smallest unit of possible feature below which we will not able to say anything specific about a quantity.

Qualitative representation provides techniques for representing only those features that are unique or essential, whereas a quantitative representation permits to represent all those values that can be expressed with respect to a predefined standard quantity. Although qualitative reasoning allows inferences to be made in absence of complete knowledge but it is not a probabilistic or fuzzy approach because it does not differentiate between certain quantities. The goal of qualitative spatial representations is to come up with a general vocabulary for performing efficient spatial inferences and deductions. Furthermore, the semantics of the vocabulary may not be unique, the discrete values for its definition over the continuous domains should permit computationally efficient spatial inference and less vague spatial descriptions (i.e., with some level of precision).

A mathematical definition for a qualitative abstraction can be represented as:
Let x be a quantitative variable, such that $x \in R$, and $R \subseteq \Re$. If the entire domain R is portioned into a finite set of mutually disjoint subdomains $\{Q_1, Q_2, \ldots, Q_m\}$, i.e., $\bigcup_{i=1}^{m} Q_i = R$, and furthermore, all numerical values lying within Q_i are treated as being equivalent and named symbolically by Label (Q_i), then the qualitative variable [x] corresponding to x is defined as follows:

$$[x] \in X, X \subseteq \bigcup_{i=1}^{m} \text{Label}\,(Q_i), \text{where Label}(Q_i) \text{ is called a primitive qualitative value.}$$

Properties of qualitative representations: The valuable properties of qualitative representations are:

1. Qualitative representations take only limited distinctions as a necessary measure to identify objects, events, situations, etc. in a given context (recognition task) as opposed to those representation which fully construct a situation (reconstruction task).
2. All information about the physical world in general, and space in particular, is composed of comparisons between magnitudes. The representations that define such comparisons, qualitative representations provide the relative arrangement of magnitudes but not absolute information about magnitudes.
3. The exploration for distinctive features that characterizes the qualitative approach has a distinctive side effect. It configures the domain according to a particular viewpoint used. Some of the qualitative distinctions being made are practically closer to each other than others. This configuration can be observed in the set of relations used to represent the domain.
4. Qualitative representations are down played in the sense that they might correspond to many real-life situations. The justification that it still can be productively used to solve spatial problems is that those problems are always embedded in a particular context. The context, which for simplicity is deserved, should limit the relative information enough to allow spatial reasoning by making it possible to find a unique order along a description of dimension. In another way it can be a representation that can count on being used with some particular context that does not need to possess too much specific information itself.
5. Qualitative representations can tackle vague knowledge by using coarse granularity levels, which can avoid to provide specific values on a given dimension. With other words, the inherent down play by the representation absorbs the vagueness of the existing knowledge.
6. In qualitative representations of space, the structural similarity between the representing and the represented world averts from violating constraints corresponding to basic properties of the represented world, which in propositional systems would have to be restored through revision mechanisms at huge cost.
7. Unlike quantitative representations, which require a scale to be made constant before measurements can be done, qualitative representations are independent of constant granularities. The qualitative distinctions made may relate to finer or coarser differences in the represented world, depending on the granularity of the acquired knowledge and the actual context.
8. The knowledge content of qualitative relations varies. Some might represent a large range of quantitative values of the same quality, while others may single out a unique distinguishing value.
9. While the segregating power of single qualitative relations is kept intentionally low, the combination of several relations can lead to a relatively fine distinction. If each relation has a set of possible values, the intersections of those sets correspond to elements that satisfy all constraints simultaneously.

3.4 Qualitative Spatial Representation

Quantitative representations generally maintain the spatial information in a common global or local coordinate system. The agent in consideration might have a local coordinate system derived from the pre-defined orientation of the agent or alternatively in the global coordinate

system where all entities and objects use the same common system which can, for example oriented in a street map or the cardinal directions. Agents can provide their spatial knowledge by mapping their local coordinate systems to a global one. That is only possible only if all agents have knowledge about the global coordinate system. The required global coordinates can then be shared.

Quantitative representations cannot properly handle indeterminate or inexact knowledge. For a quantitative representation the exact position and the size of all objects should be known. Furthermore, detecting certain spatial configurations is quite hard but it is often required to trigger certain tasks or behaviours. In the numerical approach reasoning is done with numerical or geometrical methods like computing a tangent to the object or closest points of two objects to each other.

The qualitative approach presents spatial knowledge without perfect numerical values. It uses a finite set of vocabulary that describes a finite number of possible arrangements. In contrast to the quantitative approach this approach is closer to how humans represent spatial knowledge. In natural language the human says something like "A is above B" or "A is close to B" to express spatial information. This knowledge is usually sufficient to identify an object or follow a route. It is easy to represent indefinite or vague knowledge with qualitative methods. For example, one could say "A is right or behind B". Furthermore, the rules are easy to define like "Move back if A is close to B".

Qualitative spatial information is not imprecise although no numerical values are used. This is because segregation is only made when necessary and they depend on the level of granularity that was selected. Both, the quantitative approach and the qualitative approach, have their own advantage because both have uses where they are appropriate. Robots that interact with humans usually gains upper hand by using a qualitative representation. If available, exact coordinate-based system should be used. The side-by-side use and interchange of both representation approaches is often helpful, for example for a car navigation system by giving directions that provide the way in an unknown city using its street map and data from the Global Positioning System.

Points versus Areas: Qualitative spatial reasoning can be constructed using spatially extended objects or abstract points. It is better to use intervals of time in place of points because every event in time (for example "finding a letter") can be decomposed into a time interval ("looking at an alphabet" → "realizing that it is the one that is searched for"). This is also true for space since every real-life point can be magnified to an area (at least as one stays above nano-sized structures). Another ground for selecting intervals of time is the problem of open or closed intervals that occurs if one models intervals of time with points of time. Considering a circumstance where a light is switched off there is a point in time where the light is neither on or off assuming open intervals. With closed intervals there would be a point in time where the light is both - on and off. These problems are very classically handled with this model of Temporal logic.

In two-dimensional space intervals are equivalent to the aspect’s topology and shape. It is decided that it is needed to consider areas and not points if one wants to reason about topology or shape. But problems appear in two-dimensional space because there are lots of possible classes of shapes which cannot be managed equally well. So, it is more convenient to use

points, especially for important aspects of space like orientation and distance. One rationale is, that the properties related to points hold for the entire spatial domain. Another reason is that shapes can be represented using points at different levels of abstraction. It is advantageous to be flexible with respect to the spatial entities and their resolution i.e., in one context one might be purely interested in 0-dimensional points such as points on a map while other applications might be interested in 1-dimensional information like the width of a road or the length of a street. 2-dimensional projections (e.g., area of a pond) or 3-dimensional shapes of objects might be of interest in other circumstances.

Orientation: Orientation is a ternary relationship depending on the referent, the relatum and the frame of reference which can be described either by a third object (origin) or by a given direction. If the frame of reference is provided the orientation can be expressed using binary relationships. For 2-dimensional space orientation can be illustrated as a 1-dimensional feature which is determined by an oriented line. The oriented line is defined by a dedicated set of two points. An orientation is denoted by a directed line *ab* through the two points from *a* towards *b* in fig. 3.3 and *ba* denotes the opposite vector. The relative orientation in 2-dimensional space is then provided by two oriented lines (which are represented by two dedicated sets of points). The two dedicated sets of points can share one point without loss of generality because the feature orientation is independent of location and vice versa. Thus, it can be used to describe the orientation of a line *bc* relative to the orientation line *ab*. This way the ternary relationship is again established. Three special cases arise if the locations of *a* or *c* or both are identical with the location of *b*. In the first special case *(a=b)* no orientation information can be inferred. In the second case *(c=b)* and the third case *(a=b=c)* orientation information is absent but location information of *c* is still available. The point *c* is called the referent (also primary object or located object), *b* the relatum (also reference object) and *a* is called origin (also parent object).

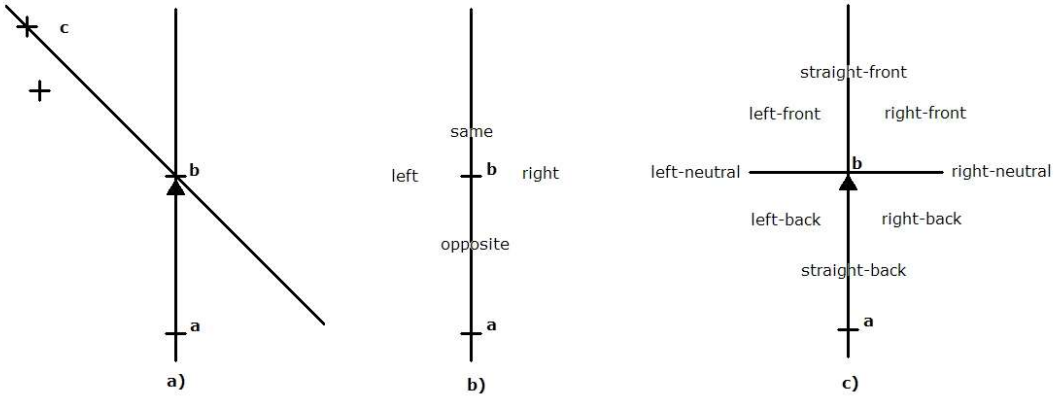


Fig. 3.3: (a)Location of *c* wrt. the location of *b* and the orientation *ab* (b)Orientation relations wrt. to the location *b* and the orientation *ab* (c)Left/Right and Front/Back dichotomies in an orientation system

There are different approaches which use multiple methods to determine the qualitative orientation values. A simple contrast can be established by defining four qualitatively different orientation values labelled *same*, *opposite*, *left* and *right*. If the point *c* is on the line *ab* on the other side of *b* than *a* then the orientation is *same* and if it is on the line *ab* on the other side of *a* than *b* then the orientation is *opposite*. The orientation is *left* if the point *c* is located on the left half plane of the oriented line *ab* and it is *right* if *c* is located on the right half plane of the oriented line *ab*. It is noted that unlike in the case of linear dimensions, increasing quantitative

orientation returns back to position of initial orientation. In this perception, orientation is a circular phenomenon. A considerable increment in information can be gained by introducing a front/ back division. In cardinal direction representation the eight different orientation labels *straight-front, right-front, right-neutral, right-back, straight-back, left-back, left-neutral* and *left-front* are created as shown in fig. 3.3(c). It is also known as "projection-based" method or "cardinal algebra". The reasoning with the cardinal algebra is NP-complete. Another way to augment the number of orientation values is by "cone-based reasoning". Here nine different relations are created: *north, north-east, east, south-east, south, south-west, west, north-west* and *equal* (fig. 3.4). In this method the plane is split into eight slices of 45^0 by four lines. All eight segments have equal scope unlike in the cardinal algebra, where the *-front* and *-back* relations signify an infinite number of angles (as in the cone-based approach) while the *straight-* and *neutral-* relations signify a single angle.

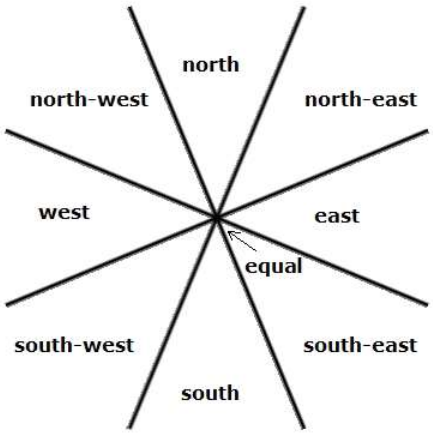


Fig. 3.4: Cone-based Representation

Distance: Distance in contrast to topology and orientation is a scalar entity. It can be quantified in terms of absolute distance relations or relative distance relations. Absolute distance relations mean the distance between two points and is measured by dividing the real line into a different number of sectors depending on the granular level. They can be represented quantitatively or qualitatively and relies upon on a uniform global scale. Absolute distance relations *name* distances whereas relative distance relations *compare* the distance between two points with the distance to a third point. Next to the obvious predicates <, =, > more relations for relative distances can be defined if needed (e.g., "much longer", "a little bit more", "much shorter").

A general framework for representing qualitative distances at different levels of coarseness is based on the space around the relatum RO, which is partitioned according to a number of totally ordered distance distinctions $Q = \{q_0, q_1, q_2, \ldots, q_n\}$, where q_0 is the distance closest to the relatum and q_n is the one farthest away (which can go to infinity). Distance relations are assembled in distance systems D defined as: $D = (Q, A, \mathfrak{I})$, where:

1. Q is the totally ordered set of distance relations,
2. A is an acceptance function defined as A: Q × O → I, such that, given a reference object (relatum) RO and a set of objects O, A (q_i, RO) returns the geometric interval $\delta_i \in I$ corresponding to the distance relation q_i,
3. $\mathfrak{I}$ is an algebraic structure with operations and order relations defined over a set of

intervals I, where $\mathfrak{I}$ defines the structure relations between different interval.

Each distance relation can be associated to a specified area surrounding a reference object (relatum) which will be circular in isotropic space (that is space which possess equal cost of moving in all directions).

Homogeneous distance systems are systems in which all distance relations have the same structure relations i.e., the measure of the geometric intervals δ_i follows a repetitive pattern. The general type of distance systems where this is not the situation is called heterogeneous. More imposing properties of the structure of these intervals include monotonicity (each interval is greater or equal than its previous) and range imposition (any given interval is greater than the entire range from the origin to the previous interval).

The frame of reference is significant for distance systems, too. In the intrinsic reference system, the distance is measured by some inherent characteristics of the relatum, like its topology, size or shape (e.g., 50 meters can be considered far away from a small house but they seem to be close if standing next to a skyscraper). The distance is obtained by some external factor in the extrinsic reference system, like the arrangement of objects, the traveling time or the costs involved. The deictic reference system uses an external point of view to obtain the distance, like how the objects are visually perceived by an observer.

Conceptual Neighbours: For higher-level reasoning and knowledge abstraction the conceptual neighbourhood feature provides several advantages. In a representation two relations are conceptual neighbours, if there exists an operation in the domain that causes a direct transition from one relation to the other. Those operations can be either spatial movement or deformations for the physical space. In the *cone-based approach* described above each relation except *equal* has three conceptual neighbours. The conceptual neighbours of *south*, for example, are *south-east, south-west* and *equal* because it is possible to make a direct move there without the need to traverse any other relation. The relation *east* on the other hand is not a conceptual neighbour of *south* because there is no direct transition from *south* to *east*. Possible ways either have to cross *south-east, equal* or even detour by traversing through *south-west*. But this only holds for the cone-based representation presented here. Other representations have other conceptual neighbours and for more coarse ones there might be a direct way from *north* to *east*.

The conceptual neighbours for a distance relation q_x in the distance representation above are obviously those two distance relations in the totally ordered set Q that are next to q_x i.e., q_{x-1} and q_{x+1} .Thus there is only one neighbour for the first (q_0) and last (q_n).

A great benefit of conceptual neighbourhood structures is, that they naturally reflect the composition of the represented real world with their operations. This makes it easy to implement reasoning principles which are strongly biased toward the operations in the represented domain. Conceptual neighbourhoods allow to only consider operations which are possible in the specific domain which can limit the problem space and thus achieve nice computational gains.

3.5 Qualitative Spatial Reasoning

In order to use the reasoning abilities discussed earlier the unary and binary operations need to be defined for the spatial representation. In this segment these operations are illustrated by using the cardinal direction representation discussed in the orientation section **(**fig. 3.3(c)**)**.

The first discussion will be on the unary operation of a spatial alignment where *c*, the referent is at the *left-back* of *ab*, where *a* is the origin and *b* is the relatum. Now there might be requirement of exchanging the origin and the relatum. It is achieved by using the **inverse** (I_{NV}) operation (table 3.2). The point *b* is now the origin where *a* is the relatum. The outcome of the operation is unclear, as can be seen in fig. 3.5.

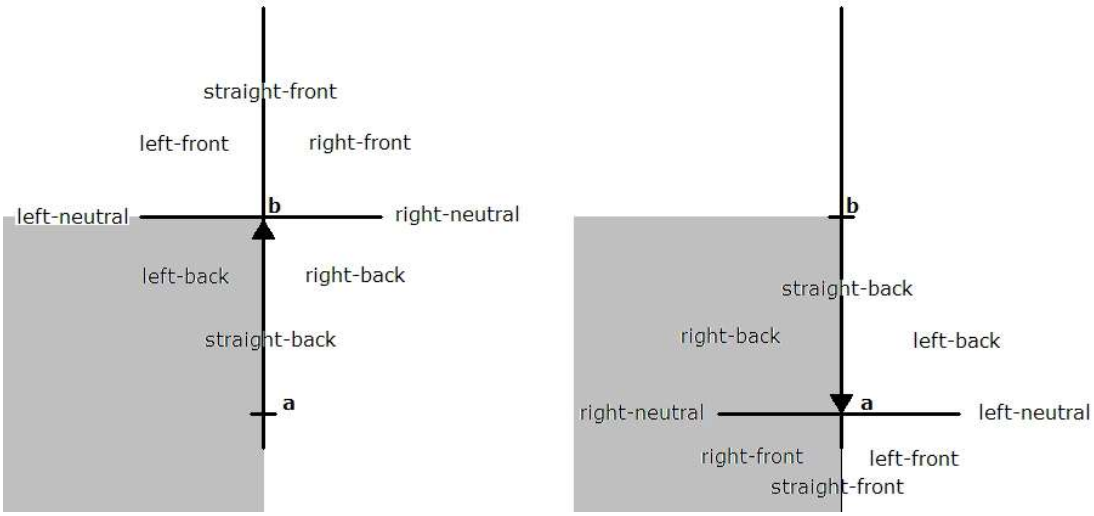


Fig 3.5: The possible locations of *c* (grey) before and after the unary I_{NV} operation

The space is now segregated differently because the front/ back division now divides the space through *a*. Of course, the labels switched, too. For example, what was *right* is now *left* and vice versa because the direction has been rotated by 180^0. Because of the qualitative character of the representation, it cannot be said whether the point *c* is *right-back, right-neutral* or *right-front* of *ba*. The outcome of this operation is thus a disjunction of all three possible atomic relations.

The binary **composition** operation can have uncertain results as well. Let's consider four points *a, b, c* and *d* and two spatial relations like one over *a, b* and *c* and the other over *b, c* and *d* (always origin, relatum and referent in this sequence). The composition of these two relations provides the location where the point *d* is in relation to *a* and *b* in fig. 3.6. The possible location of *d* for a predefined position of *c* is shaded grey. In both given relations the referent *c* and *d* are located in the *right-front*. It is obvious that the composition result, which provides the location of *d* (grey) with *a* as origin and *b* as relatum, consist of the atomic relations *right-front, right-neutral* and *right-back*. The outcome of this composition is thus a disjunction of all three possible atomic relations.

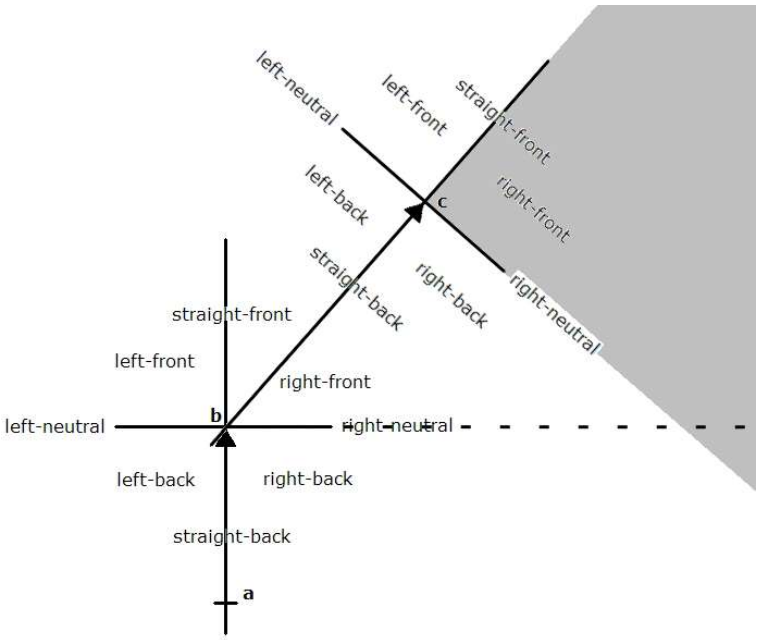


Fig. 3.6: Composition - the possible locations of *d* are shown in grey

3.6 Region Connection Calculus

In this reasoning topological relations are used as a starting point for qualitative spatial representation. It was co-developed by Randell, Cui and Cohn. RCC-8 is a subset of eight relations from RCC. Those eight base relations are: "A is disconnected from B", "A is externally connected to B", "A partially overlaps B", "A is equal to B", "A is a tangential proper part of B", "A is a non-tangential proper part of B" and the converse of the latter two relations where A and B are spatial regions. RCC-8 also holds all possible unions of these base relations. Most other relations in RCC are refinements of the RCC-8 base relations. RCC is a fully axiomatized first-order theory for representing topological relations. All entities in space are regarded as spatial regions. RCC-8 semantics of base relations can be expressed using propositional logics rather than first-order logic required in RCC.

3.7 The Dipole Calculus

It uses directed line segments called dipole that are constructed by a pair of points - a start point (s_A for a dipole A) and an end point (e_A for a dipole A). The dipoles are used to represent two-dimensional extended objects in space with an intrinsic direction as shown in fig. 3.7(a). The local direction of the dipoles is even simpler as the relation expressed in qualitative spatial representation. In the dipole representation a point can only be *left (l)*, *right (r)* and *on the straight line (o)* of the dipole in question. It merges the atomic relations *same* and *opposite* of fig. 3.3(b) into one. With the *left* and *right* relations between a dipole and the start and end points of another dipole, 24 JEPD atomic relations can be formed provided that no more than two points are allowed to be on a line (this is called general position) with the exception that two dipoles may share one point.

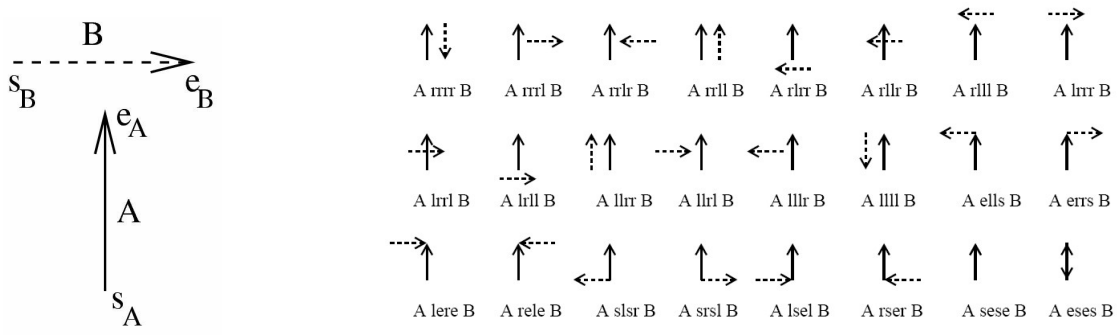


Fig. 3.7: (a) Dipole (b) The 24 atomic relations of the dipole calculus

Given two dipoles A and B with their start points s_A, s_B and their end points e_A, e_B, the atomic relations between them can be described as follows:
$AR_1s_B \wedge AR_2e_B \wedge BR_3s_A \wedge BR_4e_A$ with $R_1, R_2, R_3, R_4 \in \{r, l, s, e\}$
If R_1 is *s* or *e* (meaning that s_B is on the same point as s_A or e_A), R_2 can only be *r* or *l* and vice versa (R_1 and R_2 exchanged), since dipoles share maximal one point. This also holds for R_3 and R_4. A short form of the term above can be written as: A $R_1R_2R_3R_4$ B as given in fig. 3.7(b).

A difficulty of the Dipole Calculus is, that it deduces intrinsic objects although the intrinsic character of objects may not exist or not needed for an application's sensor.

3.8 Ternary Point Configuration Calculus

The Ternary Point Configuration Calculus (TPCC) is a qualitative spatial reasoning method that uses ternary relations of points developed by Reinhard Moratz. The distinctions in TPCC are less rigorous than in the method described in dipole calculus and thus TPCC permits more useful differentiations for realistic application scenarios.

Flip-Flop Partition: TPCC have point-like objects on a 2D-plane. A relative reference system is given by an origin and a relatum. The origin and the relatum forms the reference axis. The spatial relation between the reference system and the referent is then obtained qualitatively by naming the segment of the partition in which the referent lies.

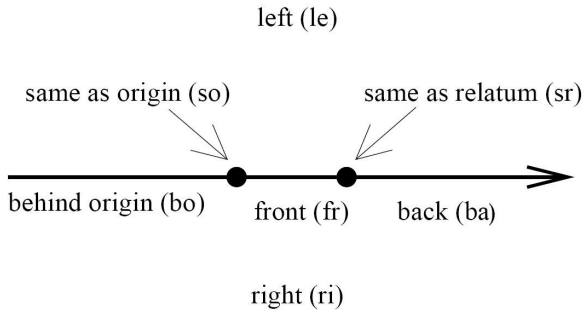


Fig. 3.8: The flip-flop partition

In this method the reference axis segments the 2D-plane into two parts - left and right. The spatial relation between the reference system and the referent is described qualitatively by naming the part of the segment in which the referent lies. The referent can not only be on the left or the right but it can also be on the reference axis itself. Considering all cases, the position is categorised by five configurations. The referent can either be behind the relatum (back - ba), at the same position as the relatum (same as relatum - sr), in front of the relatum (front - fr), at the same point at the origin (same as origin - so) or behind the origin (behind origin - bo). shows this partition. The partition is shown in fig. 3.8.

Points A, B and C can be examples for origin, relatum and referent. Two additional configurations are also possible in which origin and relatum have exactly the same location. In the first case the origin and relatum are at the same point and the referent at some other (double point - dou). In second case all three points can be at the same location (triple point - tri). Infix notation is used to describe configurations. The reference system consisting of origin and relatum are written before the relation symbol and the referent is written after it.

The region of acceptance for front and back need similar extensions like left and right. In the flip-flop calculus front and back only have acceptance regions on a line. The flip-flop calculus can be extended by partitioning the 2D-plane with a cross. Therefore, the left and right side is respectively divided into a front and back.

Representation in TPCC: The TPCC calculus is acquired from the cardinal direction calculus and presents finer differentiations than the cardinal direction calculus. Its 2D-plane is segmented equally into eight slices by adding another cross which is rotated 45^0. Additionally, the distance between the relatum to the referent is compared to the distance from the relatum to the origin to provide a separation between the two ranges.

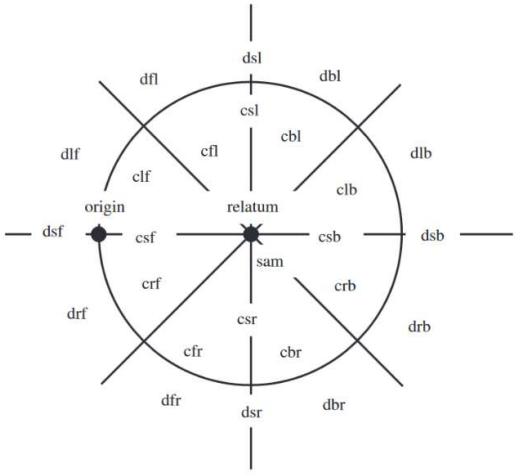


Fig. 3.9: The reference system used by the TPCC

The letters f, b, l, r, s, d, c in fig. 3.9 means front, back, left, right, straight, distant, close. The TPCC has 8 different orientations and 4 precise orientations. With 2 distances and 3 special cases there are (8 + 4) x 2 + 3 = 27 possible configurations. The configuration in which the referent is at the same position as the relatum is called *sam* (for “same location”). The two special configurations in which the origin and the relatum have the same location are *dou* and

tri and are also base relations of this calculus. The formal definition of the TPCC relations is described by geometric configurations on the basis of a cartesian co-ordinate system represented by R^2. If the position for origin A, relatum B and referent C be $(x_A, y_A), (x_B, y_B)$ and (x_C, y_C) respectively, then two of the special cases are defined as:

A, B dou C $:= (x_A = x_B \neq x_C) \wedge (y_A = y_B \neq y_C)$
A, B tri C $:= (x_A = x_B = x_C) \wedge (y_A = y_B = y_C)$

For A≠B a relative radius $r_{A,B,C}$ and a relative angle $\emptyset_{A,B,C}$ are defined:

$$r_{A,B,C} := \frac{\sqrt{(x_c - x_B)^2 + (y_c - y_B)^2}}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}} \text{ and } \emptyset_{A,B,C} := \tan^{-1}\left(\frac{y_c - y_B}{x_c - x_B}\right) - \tan^{-1}\left(\frac{y_B - y_A}{x_B - x_A}\right)$$

The spatial relations are defined as:

A, B sam C $:= r_{A,B,C} = 0$
A, B csb C $:= (0 < r_{A,B,C} < 1) \wedge (\emptyset_{A,B,C} = 0)$
A, B dsb C $:= (r_{A,B,C} \geq 1) \wedge (\emptyset_{A,B,C} = 0)$
A, B clb C $:= (0 < r_{A,B,C} < 1) \wedge \left(0 < \emptyset_{A,B,C} \leq \frac{\pi}{4}\right)$
A, B dlb C $:= (r_{A,B,C} \geq 1) \wedge \left(0 < \emptyset_{A,B,C} \leq \frac{\pi}{4}\right)$
A, B cbl C $:= (0 < r_{A,B,C} < 1) \wedge \left(\frac{\pi}{4} < \emptyset_{A,B,C} < \frac{\pi}{2}\right)$
A, B dbl C $:= (r_{A,B,C} \geq 1) \wedge \left(\frac{\pi}{4} < \emptyset_{A,B,C} < \frac{\pi}{2}\right)$
A, B csl C $:= (0 < r_{A,B,C} < 1) \wedge \left(\emptyset_{A,B,C} = \frac{\pi}{2}\right)$
A, B dsl $:= (r_{A,B,C} \geq 1) \wedge \left(\emptyset_{A,B,C} = \frac{\pi}{2}\right)$
A, B cfl C $:= (0 < r_{A,B,C} < 1) \wedge \left(\frac{\pi}{2} < \emptyset_{A,B,C} < \frac{3\pi}{4}\right)$
A, B dfl C $:= (r_{A,B,C} \geq 1) \wedge \left(\frac{\pi}{2} < \emptyset_{A,B,C} < \frac{3\pi}{4}\right)$
A, B clf C $:= (0 < r_{A,B,C} < 1) \wedge \left(\frac{3\pi}{4} \leq \emptyset_{A,B,C} < \pi\right)$
A, B dlf C $:= (r_{A,B,C} \geq 1) \wedge \left(\frac{3\pi}{4} \leq \emptyset_{A,B,C} < \pi\right)$
A, B csf C $:= (0 < r_{A,B,C} < 1) \wedge \left(\emptyset_{A,B,C} = \pi\right)$
A, B dsf C $:= (r_{A,B,C} \geq 1) \wedge \left(\emptyset_{A,B,C} = \pi\right)$
A, B crf C $:= (0 < r_{A,B,C} < 1) \wedge \left(\pi < \emptyset_{A,B,C} < \frac{5\pi}{4}\right)$
A, B drf C $:= (r_{A,B,C} \geq 1) \wedge \left(\pi < \emptyset_{A,B,C} < \frac{5\pi}{4}\right)$
A, B cfr C $:= (0 < r_{A,B,C} < 1) \wedge \left(\frac{5\pi}{4} < \emptyset_{A,B,C} < \frac{3\pi}{2}\right)$
A, B dfr C $:= (r_{A,B,C} \geq 1) \wedge \left(\frac{5\pi}{4} < \emptyset_{A,B,C} < \frac{3\pi}{2}\right)$
A, B csr C $:= (0 < r_{A,B,C} < 1) \wedge \left(\emptyset_{A,B,C} = \frac{3\pi}{2}\right)$

$$\text{A, B dsr C} := (r_{A,B,C} \geq 1) \wedge \left(\emptyset_{A,B,C} = \frac{3\pi}{2}\right)$$

$$\text{A, B cbr C} := (0 < r_{A,B,C} < 1) \wedge \left(\frac{3\pi}{2} < \emptyset_{A,B,C} < \frac{7\pi}{4}\right)$$

$$\text{A, B dbr C} := (r_{A,B,C} \geq 1) \wedge \left(\frac{3\pi}{2} < \emptyset_{A,B,C} < \frac{7\pi}{4}\right)$$

$$\text{A, B crb C} := (0 < r_{A,B,C} < 1) \wedge \left(\frac{7\pi}{4} \leq \emptyset_{A,B,C} < 2\pi\right)$$

$$\text{A, B drb C} := (r_{A,B,C} \geq 1) \wedge \left(\frac{7\pi}{4} \leq \emptyset_{A,B,C} < 2\pi\right)$$

The notation for set configurations is to write the relations of the sets in parentheses, separated by commas - e.g. A, B clf C ∧ A, B cfl C would be described by A, B (clf,cfl) C.

Reasoning with TPCC: The unary and binary operations are performed in TPCC as well. In order to maintain the transformation and composition tables small a visual representation for the TPCC relations is presented as shown in fig. 3.10. Segments corresponding to a relation are given as filled segments, sets of base relations have several segments filled. This representation is easier to express into its semantic content compared with a representation that uses the mathematical relations and conditions.

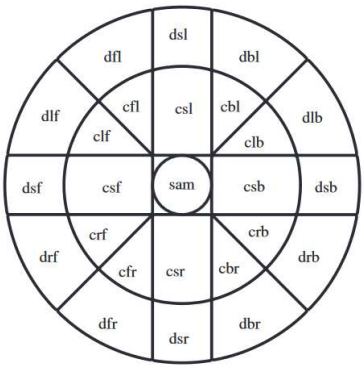


Fig. 3.10: Iconic representation for TPCC-relations

3.9 Distance / Orientation-interval Propagation

The Distance/orientation-interval propagation (DOI) suggests an approach to model the typically ambiguous sensor data about orientations and distances. This approach generates orientation and distance intervals to build global knowledge. The Distance/orientation-interval propagation (DOI) can be used as a calculus for mobile robot indoor exploration. Therefore, it reflects the ambiguous sensor data robots usually provide. Furthermore, the data provided by the sensors may be not only ambiguous but also incomplete. This creates serious issues regarding the integration of local spatial knowledge into survey knowledge which is, for example, required for robot explorations in unknown environments. Navigation tasks needs calculi that handle orientation and distance information as pure topological information is not sufficient. The DOI uses a relative reference system which makes use of continuous interval

borders for modelling ambiguousness. Therefore, it can be used as an extension to the qualitative approaches. The advantage of qualitative approaches can be combined with metric measurements by using DOI. Qualitative calculi can represent imprecise spatial knowledge while metric representations are good at distinguishing different spatial entities.

DOI Definition: The Distance/orientation-interval propagation is based on continuous distance orientation intervals. Thus, again a point-like object on a 2D-plane is considered. The DOI uses this point together with a reference direction as anchor and it has four additional parameters: $r^{min}, r^{max}, \emptyset^{min}$ and $\emptyset^{max}$. A DOI *d* is a set of polar vectors $(r^i, \emptyset^j)$ with:
$d = \{(r^i, \emptyset^j) \mid r^{min} \leq r^i \leq r^{max} \wedge \emptyset^{min} \leq \emptyset^j \leq \emptyset^{max}\}$
There is a special case which represents the spatial arrangement where the goal location can be the same as the reference point. This case is represented by the following values:
$\emptyset^{max} = \pi, \emptyset^{min} = -\pi$ and $r^{min} = 0$

In all other cases, $\emptyset^{max} - \emptyset^{min} \leq \pi$ holds and for convenience it is assumed that $-2\pi \leq \emptyset^{min} \leq \pi$ and $-\pi \leq \emptyset^{max} \leq \pi$.

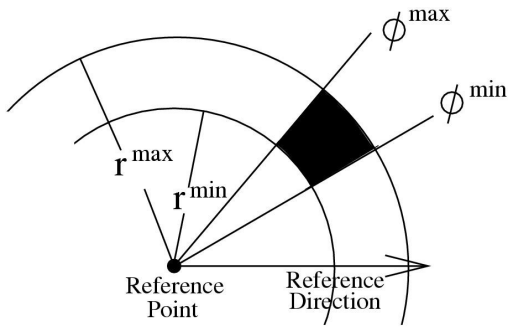


Fig. 3.11: A DOI and its parameters

DOI Composition: The composition between two DOIs is the basic step for propagation along paths. Two DOIs (d_1 and d_2) can be composited into a third(d_3). For the three DOIs given as: $d_1 = (r_1^{min}, r_1^{max}, \emptyset_1^{min}, \emptyset_1^{max})$, $d_2 = (r_2^{min}, r_2^{max}, \emptyset_2^{min}, \emptyset_2^{max})$, $d_3 = (r_3^{min}, r_3^{max}, \emptyset_3^{min}, \emptyset_3^{max})$ it holds

$$d_3 = \Big(\min_{r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l} \Sigma^r(r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l), \max_{r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l} \Sigma^r(r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l), \min_{r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l} \Sigma^{\emptyset}(r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l), \max_{r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l} \Sigma^{\emptyset}(r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l) \Big)$$

with $r_1^{min} \leq r_1^i \leq r_1^{max}, \emptyset_1^{min} \leq \emptyset_1^j \leq \emptyset_1^{max}, r_2^{min} \leq r_2^i \leq r_2^{max}, \emptyset_1^{min} \leq \emptyset_2^j \leq \emptyset_1^{max}$.
The functions Σ^r and $\Sigma^{\emptyset}$ are defined as:

$$\Sigma^r(r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l) = \sqrt{\left(r_1^i \sin\emptyset_1^j + r_2^k \sin(\emptyset_1^j + \emptyset_2^l)\right)^2 + \left(r_1^i \cos\emptyset_1^j + r_2^k \cos(\emptyset_1^j + \emptyset_2^l)\right)^2}$$ and

$$\Sigma^{\emptyset}(r_1^i, r_2^k, \emptyset_1^j, \emptyset_2^l) = \tan^{-1}\left(\frac{r_1^i \sin\emptyset_1^j + r_2^k \sin(\emptyset_1^j + \emptyset_2^l)}{r_1^i \cos\emptyset_1^j + r_2^k \cos(\emptyset_1^j + \emptyset_2^l)}\right)$$

The values for the minimum and maximum of d_3 thus the cases for which r_3^m and $\emptyset_3^n$ have their minimum or maximum and can be listed by geometric analysis. The composition is only an approximation in form of an upper bound of the area consisting of the vectors $(r_3^m, \emptyset_3^n)$ which

can be directly composed by vectors $(r_1^i, \emptyset_1^j)$ and $(r_2^k, \emptyset_2^l)$ from d_1 and d_2 respectively. A typical spatial layout of these areas is shown in fig. 3.12.

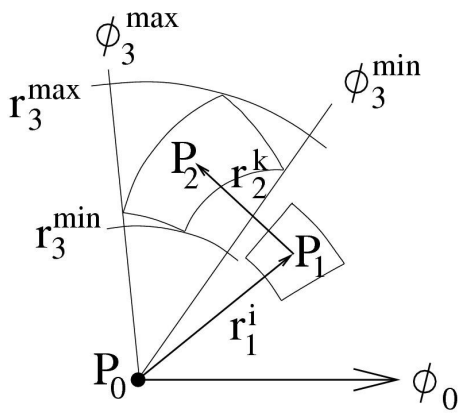


Fig. 3.12: Resulting DOI

A DOI can be observed as a difference between two vectors expressed in polar coordinates relative to the reference direction. A function Δ maps two points and a reference direction to the corresponding difference vector. The function $\emptyset$ maps two points and a reference direction from the first point to the second. The relation between points can now be expressed in imprecise relative position and the respective DOIs:

$\Delta(\emptyset_0, P_0, P_1) \in d_1 \wedge \Delta(\emptyset(P_0, P_1), P_1, P_2) \in d_2 \Longrightarrow \Delta(\emptyset_0, P_0, P_2) \in d_1 \diamond d_2$

Now the DOI composition can be used to generate local, relative spatial knowledge along a path. There is an anchor point P_0, a reference direction $\emptyset_0$ and a sequence of points which determine the path segments P_1, P_2, …. P_i, …., P_n. Each Point P_i on the path has an associated DOI d_i. A stepwise composition recursively beginning with the end of the path yields the relative position of the end point with respect to the anchor point P_0 and the reference direction $\emptyset_0$: $\Delta(\emptyset_0, P_0, P_n) \in d_1 \diamond (d_2(\dots(d_{n-1} \diamond d_n)\dots))$

3.10 Granular Point Position Calculus

Two points define a relative reference system in two-dimensional space. The Granular Point Position Calculus (GPPC) partitions the space in several orientations and distances. The special cases for the origin A = (x_A, y_A), the relatum B = (x_B, y_B) and the referent C = (x_C, y_C) are:

A, B dou C $:= (x_A = x_B) \wedge (y_A = y_B) \wedge ((x_C \neq x_A) \vee (y_C \neq y_A))$

A, B tri C $:= (x_A = x_B = x_C) \wedge (y_A = y_B = y_C)$

The relative radius for the other cases is defined as, $r_{A,B,C} := \frac{\sqrt{(x_c - x_B)^2 + (y_c - y_B)^2}}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}}$

A, B sam C $:= r_{A,B,C} = 0$

For A≠B≠C the relative angle is defined as, $\emptyset_{A,B,C} := \tan^{-1}\left(\frac{y_c - y_B}{x_c - x_B}\right) - \tan^{-1}\left(\frac{y_B - y_A}{x_B - x_A}\right)$

GPPC allows different levels of granularity. A $GPPC_m$ has $(4m - 1)(8m) + 3$ base relations which are defined as follows:

$$0 \leq j \leq 8m - 2 \wedge j \bmod 2 = 0 \rightarrow \emptyset_{A,B,C} = \frac{j\pi}{4m}$$

$$1 \leq j \leq 8m - 1 \wedge j \bmod 2 = 1 \rightarrow \frac{(j-1)\pi}{4m} < \emptyset_{A,B,C} < \frac{(j+1)\pi}{4m}$$

$$1 \leq i \leq 2m-1 \wedge i \bmod 2 = 1 \rightarrow \frac{i-1}{2m} < r_{A,B,C} < \frac{i+1}{2m}$$

$$2 \leq i \leq 2m \wedge i \bmod 2 = 0 \rightarrow r_{A,B,C} = \frac{i}{2m}$$

$$2m+1 \leq i \leq 4m-3 \wedge i \bmod 2 = 1 \rightarrow \frac{m}{2m-\frac{i-1}{2}} < r_{A,B,C} < \frac{m}{2m-\frac{i+1}{2}}$$

$$2m+2 \leq i \leq 4m-2 \wedge i \bmod 2 = 0 \rightarrow r_{A,B,C} = \frac{m}{2m-\frac{i}{2}}$$

$$i = 4m-1 \rightarrow m < r_{A,B,C}$$

Fig. 3.13 shows an example configuration of an GPPC with the granularity level m of three.

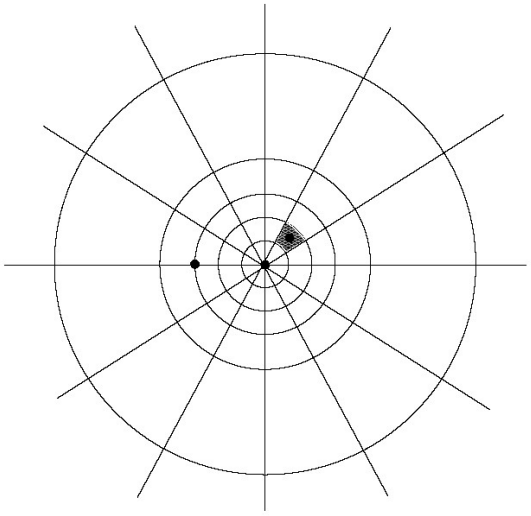


Fig. 3.13: Example configuration in GPPC

The DOI calculus described in above is used to calculate the composition table for the GPPC. This is possible because the flat segments and their borders are condensed obtaining a quasi-partition.

3.11 Summary

Qualitative representations of space can be used in various application areas in which spatial knowledge plays a part. It is in particular used in those systems which are characterized by uncertainty and incompleteness. This coarse knowledge often comes handy in early stages of design projects, in which verbally expressed spatial requirements and rough sketches are very common. Examples for such application areas include computer aided systems for architectural design and urban planning, computer vision, visual (programming) languages, natural language processing (input and output), document analysis, spatial and geographical information systems, qualitative simulations of physical processes and of course navigation, robot navigation and robot-control.

4

Qualitative Spatial Reasoning about Point Positions

4.1 Introduction

The method that is being discussed in this chapter can be used in robotics and other applications that works on positional information. The current robotic navigation and communication systems can work with two-dimensional space and produce satisfactory results. This is possible because the motion most often happens on the plane and the sensors as well as the maps are two-dimensional. Another reason is humans most often does movement this way if there is not a requirement to do otherwise. Finally two-dimensional reasoning is comparatively easier than three-dimensional reasoning, while it is simply not possible to do three-dimensional navigation with one dimensional calculus. The space is assumed to be isometric and homogeneous. Isometric signify that the cost of movement is the same in all directions while a homogeneous space is one that has the same properties at all positions. These restrictions are taken because otherwise it would be really difficult to develop a satisfactory representation.

The most important spatial aspects for distinguishing objects are shape, topology and position.

Representation of the shape of an object is independent from its position and vice versa for both absolute and relative reference system. Thus, it does not influence the calculus discussed here. It is further assumed that all considered spatial entities are separate from each other. Reasons for that are, that sensors may have problems to distinguish objects that touching each other, that most objects actually are separated from each other and if they are touching each other most of the time this fact is ignored. Therefore, the most important aspect of space for robotics is the position, comprising of orientation information and distance information. It is more suitable to use points for representing orientation and distance. Thus, it is called Qualitative Spatial Reasoning about Point Positions.

4.2 Representation

It has a situation-based level of granularity. Depending on the situation a very fine-grained representation could be chosen or a quite coarse one which would require less memory and it will be time consuming. Very little changes need to be done in order to change the granularity.

The reasoning is defined over ternary points: the origin, the relatum and the referent. The three points are needed for the qualitative reasoning, because it will use a relative frame of reference. The qualitative distance is however independent of the origin as discussed in the next section.

Distance for Spatial Reasoning: Hernández distance will be used for the spatial reasoning.

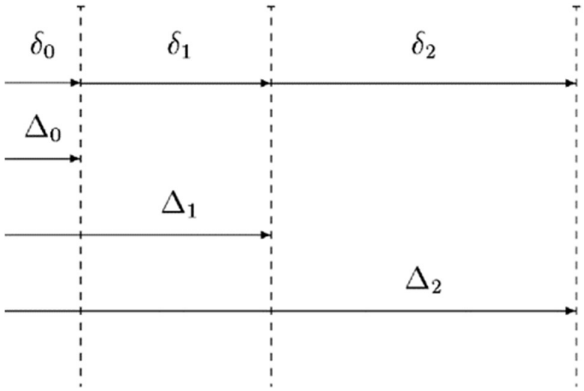


Figure 4.1: Distance from relatum Δ_x and distance range δ_x

The distance system D is defined over a totally ordered set of distance relations Q, an acceptance function A and an algebraic structure $\Im$: D = (Q, A, $\Im$). Q = {q_0, q_1, q_2, …, q_n,}, given a level of granularity with $n + 1$ distance segregations. The width of an acceptance area corresponding to a distance symbol q_i is denoted with δ_i whereas Δ_i denotes the maximum distance that a relatum can have to the referent for still falling into the acceptance are of the distance symbol q_i in fig. 4.1.

The reasoning utilizes an absolute distance representation. The surrounding area of the relatum is segmented into a number of circular acceptance areas using a global scale. In the indoor scenario, the performance radius is quite low compared to the distance of objects within the sensing range. It is thus permitted to use a monotone distance system.

The absolute representation is selected because the quantitative data input allows an easy way to relate distance representation (opposed to orientation). It is advantageous to keep the absolute distance information, because it is very hard to find out a metric distance for the robot using only relative distance relations. Absolute distances also provide better calculation results and finds easier usage in maps and other applications. A drawback is that it is far away from human nature.

For the distance part of the reasoning the origin is not required. The distance is only defined between the relatum and the referent. That concludes the fact that all unary operations except

I_D and S_C (from table 3.2) lose all distance information. The final representation only has orientation information and any distances may be possible for an orientation. But this does not necessarily have disadvantage. During the constrain propagation phase the distance information will be restored if it was available earlier in the initial set of relations.

In the absolute distance system for length L of the first interval δ_0 has to be specified. It's value strongly depends on the number of distance partitions *m* as well as on the distance system chosen. The following values seem to be quite reasonable and are thus used later on:

$$L = 10 \text{ cm}$$
$$m = 24$$
$$\delta_{i+1} = \delta_i \cdot 1.25$$

The last distance interval q_{24} has a size of δ_{24} = 21.2 m ($1.25^{24} \cdot 10$ cm) and the maximum distance would be at Δ_{24} = 105.5 m. The representation can access distances greater than the maximum distance and consider it in the last distance interval. The size of δ_{24} and Δ_{24} is thus infinite. Besides that, exception all distance intervals δ_i bigger than ten (i > 10) than have about 20% of the size of the total distance Δ_i.

Orientation for Spatial Reasoning: The reasoning employs a relative frame of reference for orientation. The advantages of relative positions are that no knowledge of intrinsic characteristics of objects is needed and that no global coordinate system is required which is often unavailable in indoor scenarios.

The space around the relatum is segmented according to a number of ordered orientation distinctions $R = \{o_0, o_1, o_2, \ldots, o_m\}$. The *acceptance function* defines acceptance conditions areas around the relatum which correspond to the orientation relations (o_j).

Depended on the level of granularity required, different numbers of orientation relations are possible. The number of orientations *m* is always even. This way the origin is always located on the border between the acceptance areas of the orientation relations and $o_{\frac{m}{2}-1}$ and $o_{\frac{m}{2}}$.

These special cases in which objects are located directly on the border between two acceptance areas are very unlikely to happen in real life scenarios. It is thus decided that in such cases all bordering acceptance areas are to be considered as possible locations for the referent termed as quasi-partition. The acceptance function for an angle ∅ are defined by the following formula:

$$\emptyset \text{ in } o_j \rightarrow \frac{j\pi}{m} \leq \emptyset \leq \frac{(j+1)\pi}{m}$$

Position for Spatial Reasoning: The distance and orientation are put together by defining the Cartesian product of the set of distance relations Q and the set of orientation relations R: $S = Q \times R$. Thus, the representation has a set S of m.n atomic relations $f_{i,j}$:

$$S = \begin{bmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,j} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,j} \\ \vdots & \vdots & \ddots & \vdots \\ f_{i,0} & f_{i,1} & \ldots & f_{i,j} \end{bmatrix}$$

The reasoning is defined over three points: the origin A with its coordinates $A = (x_A, y_A)$, the relatum $B = (x_B, y_B)$ and the referent $C = (x_C, y_C)$.

The three special cases that can occur in all ternary calculi are defined:

$$A, B \text{ dou } C := x_A = x_B \wedge y_A = y_B \wedge (x_C \neq x_A \vee y_C \neq y_A)$$

$$A, B \text{ tri } C := x_A = x_B = x_C \wedge y_A = y_B = y_C$$

The relative radius for the other cases is defined as:

$$r_{A,B,C} := \frac{\sqrt{(x_c - x_B)^2 + (y_c - y_B)^2}}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}}$$

$$A, B \text{ sam } C := r_{A,B,C} = 0$$

For $A \neq B \neq C$ the relative angle is defined as:

$$\emptyset_{A,B,C} := \tan^{-1}\left(\frac{y_c - y_B}{x_c - x_B}\right) - \tan^{-1}\left(\frac{y_B - y_A}{x_B - x_A}\right)$$

The acceptance areas for the reasoning representation are defined according to the distance and orientation relations above:

$$C \text{ in } f_{i,j} \rightarrow \Delta_{i-1} \leq r_{A,B,C} \leq \Delta_i \wedge \frac{j\pi}{m} \leq \emptyset_{A,B,C} \leq \frac{(j+1)\pi}{m}$$

For the first and the last distances the following definitions are made:

$$\Delta_{-1} = 0 \text{ and } \Delta_n = \text{infinite}$$

In fig. 4.2 a representation with 416 (excluding the three special cases) base relations is shown. For display advancement only 13 distances are used rather than the suggested 24, but the distance system follows the $\delta_{i+1} = \delta_i \times 1.25$ formula. 32 orientations are differentiated. The origin is shown on the left (it's distance to relatum and referent is absent), the relatum is in the middle while the referent is the object that is to be located. In fig. 4.3 an example for possible locations of the referent is marked gray. The referent object has equal probabilities of being present in one of the 22 indicated base relations. The given reasoning relation in fig. 4.3 thus consists of a disjunction of those 22 base relations.

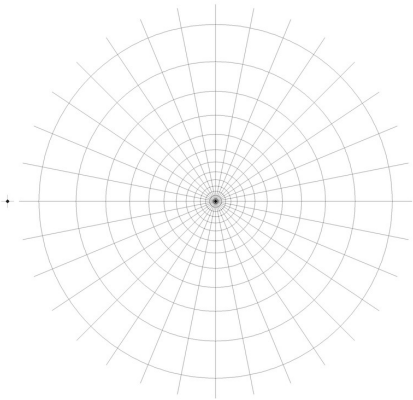


Fig. 4.2: An example of reasoning with 32 orientation distinctions and 13 distances

4.3 Reasoning

The unary operations and the binary composition are required for constraint-based reasoning. But those operation are not trivial in the given representation.

Composition: The reasoning has the ability to make use of the Distance/orientation-interval propagation. The DOI provides easy access to composition results. The atomic relations of the reasoning are quite close to that of DOIs so there is no problem to use the DOI composition for the atomic relations. This is done mostly on demand and not prior due to the requirement of huge composition table. The composition result of a general reasoning relation is thus the union of the composition results of the atomic relations.

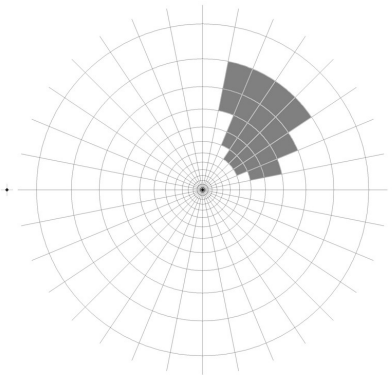


Fig. 4.3: Disjunction of 22 base relations indicating possible locations of the referent

Unary operations: The unary operations are harder to develop. The difficulty is that the distance of the origin from the relatum and the referent is unknown which leads to quite uncertain results. The unary operations are performed as follows (the identity operation I_D is ignored):

1. Inversion (I_{NV}): For performing the inversion operation the composition is used. First a temporary Spatial Reasoning about Point Positions (SPP) is obtained. It represents the possible reference of the origin A to the relatum B, where A is always at the border in the back and it has an unknown distance. The SPP thus looks like in figure 4.4. Now the composition of this temporary SPP with the original has the inversion as a result.

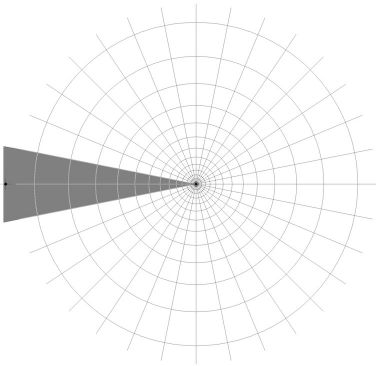


Fig. 4.4: The temporary SPP for the I_{NV} operation

2. Short Cut (S_C): The short cut reorders the SPP (ABC) to (ACB). That means, the distance information is preserved (since the distance BC is the same as CB). But the orientation is ambiguous after that operation because of the unknown distance of the origin A. The result of $S_C(o_i)$ is for $i < \frac{m}{2}$: all orientations in the range from $\frac{m}{2}$ to $\frac{m}{2} + i$. For $i \geq \frac{m}{2}$: all orientations in the range from $\frac{m}{2} - 1$ to $i - \frac{m}{2}$. Figure 4.5 shows the original SPP on the left and the result of the S_C on the right.

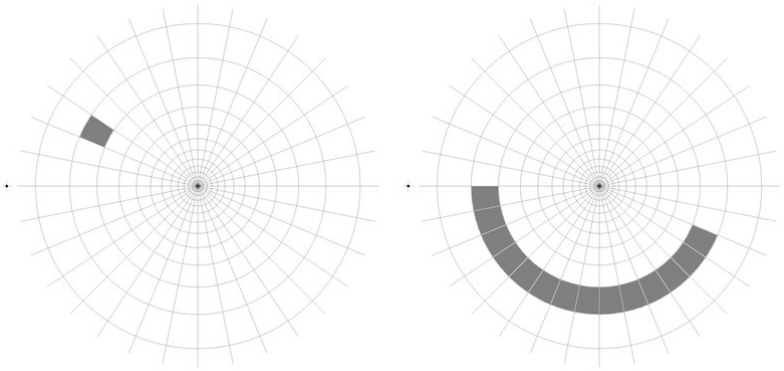


Fig. 4.5: The short cut S_C operation

3. S_{CI}, H_M and H_{MI}: The remaining unary operations can be performed using the two above operations:

$$S_{CI}(ABC) \rightarrow I_{NV}(S_C(ABC))$$
$$H_M(ABC) \rightarrow S_C(I_{NV}(ABC))$$
$$H_{MI}(ABC) \rightarrow S_C(S_{CI}(ABC))$$

Conceptual neighbourhood of the SPP: The definition of the conceptual neighbourhood of the SPP is quite simple. Movement is only possible between the acceptance areas that share an edge since quasi partition is used. The conceptual neighbours of an SPP $f_{i,j}$ are thus $f_{i,(j+1) \bmod m}$, $f_{i,(j-1+m) \bmod m}$, $f_{i-1,j}$ and $f_{i+1,j}$. The first two conceptual neighbours are those with the same distance but neighbouring orientations. It is necessary to use the modular operation since the orientation is circular - the "last" basic orientation relation is next to the "first" one. The latter two conceptual neighbours are those with the same orientation but with smaller and bigger distance. Those are only existent, of course, if $i > 0$ respectively $r < n$. See fig. 4.6 for an example. The dark grey atomic relations are the SPP relation and the light grey relations are its conceptual neighbourhood. The geometric neighbourhoods discussed in the next section have similarities with the conceptual neighbourhood, but the idea is based over different algorithm and for different applications so that they are only partly comparable.

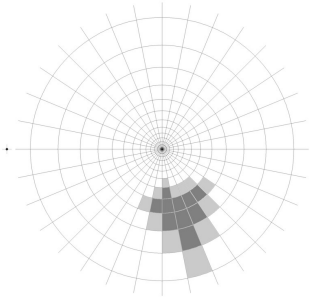


Fig. 4.6: Conceptual neighbourhood of an SPP

Improvement for the Composition using Contour Tracing: Complexity issues are a significant issue not only for Artificial Intelligence but also for qualitative spatial reasoning. A problem in the composition arises if the SPP has disjunction of more than a few basic relations. The number of DOIs that required to be computed rapidly grows. Many of those computations are not required since they often lead to results (basic SPP relations) that already have been found discovered to be possible locations of the referent. A good way to avoid these tedious computations is to calculate only the composition of those atomic relations that are border of atomic relations describing the position of the referent. An efficient way of finding those relations will be discussed in this section. The approach is called contour tracing, but it is also known as border or boundary following. In order to describe the actual algorithm some definitions are given below.

1. Geometrical Neighbourhood: The geometrical neighbourhoods that are being studied are defined on a two-dimensional plane that is divided with square segments. The resulting square segments are considered to have either a value of 0 or 1. Although the qualitative representation basic relations are not squares because two of their curved edges, this approach is still applicable. Contour tracing is a computer graphics problem where the squares are being called pixels which can have the colour white for the value 0 and black for the value 1. That way it is also easier to understand the following terms and algorithm.

2. Moore Neighbourhood: The Moore neighbourhood of a pixel P, is the set of 8 pixels which either share a vertex or edge with that pixel. These pixels are called P1, P2, P3, P4, P5, P6, P7 and P8 beginning with the top left one and encircling clockwise around the pixel P (see fig. 4.7).

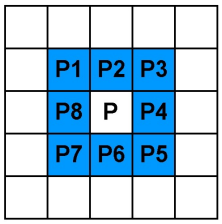


Fig. 4.7: The Moore Neighbourhood

3. 4-neighbourhood: The 4-neighbourhood is a subset of the 8-neighbourhood. It consists only the pixels that share an edge with the pixel P. Those are the pixels P2, P4, P6 and P8 of the Moore Neighbourhood.

4. Border Pixels: Along the lines of neighbourhoods the border pixels can be defined as 8-border pixels or 4-border pixels. A black pixel is an 8-border pixel if it shares a vertex or edge with at least one white pixel. Using the neighbourhood definitions from above we can say that a black pixel is an 8-border pixel if at least one of the pixels of its Moore Neighbourhood are white. 4-border pixels are thus black pixels which have at least one white pixel in their 4-neighbourhood. Those pixels share an edge with one or more white pixels.

5. Connectivity: A connected component is a set of black pixels B, such that for every pair of pixels p_i and p_j in B, there exists a sequence of pixels $p_i, \ldots, p_j$ such that:

    a) all pixels in the sequence are in the set B (they are black) and
    b) b) every 2 pixels that are adjacent in the sequence are neighbours.

A component is 4-connected if 4-neighborhood used in the second condition and it is 8-connected if the Moore Neighbourhood is used.

6. Pavlidis Contour Tracing Algorithm: It is possible to determine 4-connected borders as well as 8-connected borders using this algorithm. It is important to keep track of the direction in which exploration started in the pixel which is currently active. The first task that has to be done using this algorithm is to find a starting black pixel. The only restriction to this pixel is, that its left neighbour (P8 in the Moor Neighbourhood model) is white. From now on the only important pixels are the ones in front of the current pixel (left, ahead and right front: P1, P2 and P3). Four different cases have to be considered now:

    a) If P1 is black it is declared as our new current pixel and the current direction is changed by turning left (-90^0). P1 is also added to the set of border pixels B (see fig. 4.8).
    b) If the first failed (P1 is white) and P2 black the new current pixel is P2 which is also added to the set of border pixels B. The current direction is not being changed(see fig. 4.9).
    c) If the first two cases failed (P1 and P2 white) but P3 is black the new current pixel is P3. P3 is added to the set of border pixels B and the direction is not being changed(see fig. 4.9).
    d) If first three cases failed (all three pixels P1, P2 and P3 are white) the current direction is changed by turning right (90^0).

    These above computations are repeated until one of the following exit conditions is met:

    a) The algorithm will terminate after turning right three times **on the same pixel**
    b) Or, after reaching the start pixel.

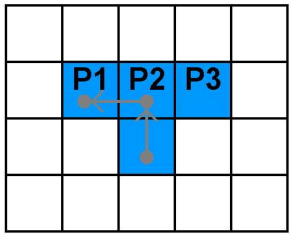


Fig. 4.8: Pavlidis: check P1

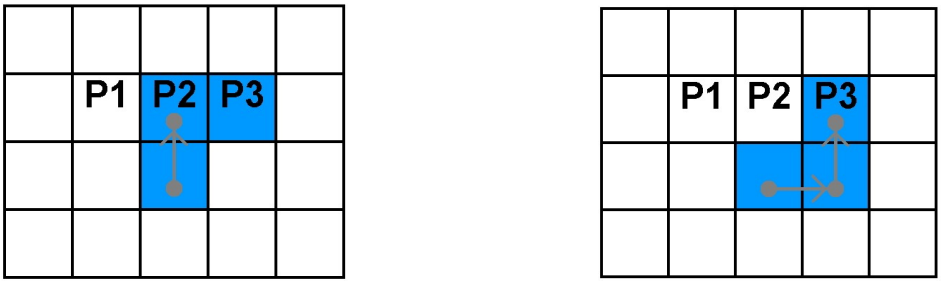


Fig. 4.9: Pavlidis: check P2(left) and check P3(right)

Using Pavlidis algorithm in SPP: For the reasoning the 8-connected border is needed because it corresponds to the conceptual neighbourhood of the calculus. Figure 4.10 shows a random SPP on the left and on the right the border atomic relations of the random SPP are marked dark while the inner atomic relations are light grey.

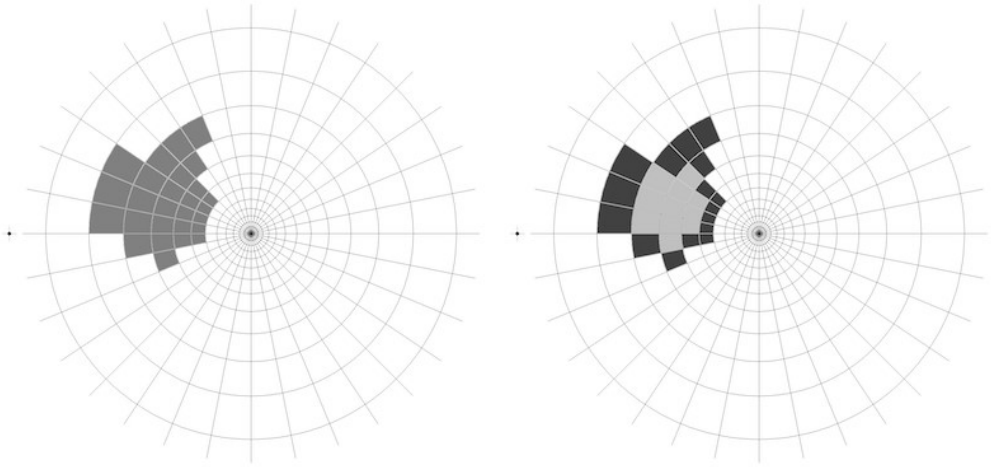


Fig. 4.10: 8-connected border of an SPP

The result of a composition that used at least one SPP with border segments it will probably not be a solid but will have holes in its representation that need to be filled (i.e., atomic relations that are needed to be added). For filling the representation Pavlidis algorithm is used. This time not only the border relations are stored but also those outside atomic relations that are neighbours of the border segments and that do not contain the referent. Those outside segments then enclose the actual SPP relation. Now all atomic relations, that are neighbours of relations that are activated if they are not one of those enclosing segments calculated above. This way an efficient filling algorithm for the SPP is implemented.

During the constraint propagation it is only required to fill the SPP when doing intersections. For all other operations it is sufficient to use the border relations. It thus might be a good practise not to use the contour tracing algorithm after every operation but maybe only after every second or third. At the end of the inference process the resulting SPP should be filled.

4.4 Summary

The SPP implemented uses a Distance/orientation-interval propagation implementation. The representation of the basic relations can be done by setting bits in a byte array. The number of orientations m is can be expressed as “ROT_SLICES” (for rotation) and the number of distances n is defined as “TRANS_SLICES” (for translation). SPP is also denoted by "Pieces". The different input and output functions come pairwise request the bit-number of the atomic relation.

5

Conclusion

In this thesis a reasoning about qualitative positional information with help of distance and orientation is explored. Chapter 1 provides a basic detail on the topic of spatial reasoning. Chapter 2 gives an approach to visualise and represent objects located in space. The basic terms and definition of reasoning have been defined in Chapter 3. Representation and reasoning techniques as well as the properties of qualitative spatial reasoning are presented there. Some of the approaches have been outlined in Chapter 3. The Ternary Point Configuration Calculus (TPCC) is of special interest. The Distance/orientation-interval propagation is important because it is needed for the reasoning process. The reasoning methods has been discussed in Chapter 4. The absolute distance system and the relative orientation are outlined and merged into the ternary position representation. Unary and binary operations are needed for the reasoning process. The algorithms for both operations are presented. The composition makes use of the Distance/orientation-interval propagation while the unary inversion (I_{NV}) operation makes use of the composition. The algorithm for the unary short cut (S_C) operation is presented so that the other operations can be calculated using I_{NV} and S_C. An important improvement for the calculation of the composition is the use of a contour trancing algorithm is also discussed.

Appendix A

DOI Composition Formula

The resulting DOI d_3 of a composition of the DOIs d_1 and d_2 has its minimum and maximum radius and angle to be computed:

$\min_{r_1^i, r_2^k, \phi_1^j, \phi_2^l} r_{\sum} \left(r_1^i, r_2^k, \phi_1^j, \phi_2^l\right)$
$\max_{r_1^i, r_2^k, \phi_1^j, \phi_2^l} r_{\sum} \left(r_1^i, r_2^k, \phi_1^j, \phi_2^l\right)$
$\min_{r_1^i, r_2^k, \phi_1^j, \phi_2^l} \phi_{\sum} \left(r_1^i, r_2^k, \phi_1^j, \phi_2^l\right)$
$\max_{r_1^i, r_2^k, \phi_1^j, \phi_2^l} \phi_{\sum} \left(r_1^i, r_2^k, \phi_1^j, \phi_2^l\right)$

For $\min_{r_1^i, r_2^k, \phi_1^j, \phi_2^l} r_{\sum} \left(r_1^i, r_2^k, \phi_1^j, \phi_2^l\right)$ 12 geometric cases have to be considered $r_3^1, \ldots, r_3^{12}$.
The $\min_{r_1^i, r_2^k, \phi_1^j, \phi_2^l} r_{\sum} \left(r_1^i, r_2^k, \phi_1^j, \phi_2^l\right) = \min\left(r_3^1, \ldots, r_3^{12}\right)$

$$
\begin{aligned}
r_3^1 &= r_{\sum}\left(r_1^{min}, r_2^{min}, 0, \phi_2^{min}\right)\\
r_3^2 &= r_{\sum}\left(r_1^{min}, r_2^{min}, 0, \phi_2^{max}\right)\\
r_3^3 &= r_{\sum}\left(r_1^{min}, r_2^{max}, 0, \phi_2^{min}\right)\\
r_3^4 &= r_{\sum}\left(r_1^{min}, r_2^{max}, 0, \phi_2^{max}\right)\\
r_3^5 &= r_{\sum}\left(r_1^{max}, r_2^{min}, 0, \phi_2^{min}\right)\\
r_3^6 &= r_{\sum}\left(r_1^{max}, r_2^{min}, 0, \phi_2^{max}\right)\\
r_3^7 &= r_1^{min} - r_2^{max} \Leftarrow \left(\phi_2 min \leq -\pi \leq \phi_2 max \wedge r_1^{min} > r_2^{max}\right)\\
r_3^8 &= r_2^{min} - r_1^{max} \Leftarrow \left(\phi_2 min \leq -\pi \leq \phi_2 max \wedge r_2^{min} > r_1^{max}\right)\\
r_3^9 &= r_{\sum}\left(r_1^{min}, r_1^{min}\cos\left(\pi - \phi_2^{max}\right), 0, \phi_2^{max}\right) \Leftarrow\\
&\quad \left(\phi_2 max > \frac{\pi}{2} \wedge r_2^{min} < r_1^{min}\cos\left(\pi - \phi_2^{max}\right) < r_2^{max}\right)\\
r_3^{10} &= r_{\sum}\left(r_1^{min}, r_1^{min}\cos\left(\pi + \phi_2^{min}\right), 0, \phi_2^{min}\right) \Leftarrow\\
&\quad \left(\phi_2 min < -\frac{\pi}{2} \wedge r_2^{min} < r_1^{min}\cos\left(\pi + \phi_2^{min}\right) < r_2^{max}\right)\\
r_3^{11} &= r_{\sum}\left(-\cos\phi_2^{max} r_2^{min}, r_2^{min}, 0, \phi_2^{max}\right) \Leftarrow \cos\phi_2 max r_2^{min} + r_1^{min} < 0 < \cos\phi_2^{max} r_2^{min} + r_1^{max}\\
r_3^{12} &= r_{\sum}\left(-\cos\phi_2^{min} r_2^{min}, r_2^{min}, 0, \phi_2^{min}\right) \Leftarrow \cos\phi_2 min r_2^{min} + r_1^{min} < 0 < \cos\phi_2^{min} r_2^{min} + r_1^{max}
\end{aligned}
$$

For $\max_{r_1^i, r_2^k, \phi_1^j, \phi_2^l} r_{\sum}\left(r_1^i, r_2^k, \phi_1^j, \phi_2^l\right)$ the geometric analysis shows seven distinct cases over which the maximum $\max\left(r_3^{13}, \ldots, r_3^{19}\right)$ has to be computed:

$$
\begin{aligned}
r_3^{13} &= r_{\sum}\left(r_1^{max}, r_2^{min}, 0, \phi_2^{min}\right)\\
r_3^{14} &= r_{\sum}\left(r_1^{max}, r_2^{min}, 0, \phi_2^{max}\right)\\
r_3^{15} &= r_{\sum}\left(r_1^{max}, r_2^{max}, 0, \phi_2^{min}\right)\\
r_3^{16} &= r_{\sum}\left(r_1^{max}, r_2^{max}, 0, \phi_2^{max}\right)\\
r_3^{17} &= r_{\sum}\left(r_1^{min}, r_2^{max}, 0, \phi_2^{min}\right)\\
r_3^{18} &= r_{\sum}\left(r_1^{min}, r_2^{max}, 0, \phi_2^{max}\right)\\
r_3^{19} &= r_{\sum}\left(r_1^{min}, r_2^{max}, 0, 0\right) \Leftarrow \phi_2^{min} < 0 < \phi_2^{max}
\end{aligned}
$$

The algorithm for $\min_{r_1^i, r_2^k, \phi_1^j, \phi_2^l} \phi_{\sum}\left(r_1^i, r_2^k, \phi_1^j, \phi_2^l\right)$ and $\max_{r_1^i, r_2^k, \phi_1^j, \phi_2^l} \phi_{\sum}\left(r_1^i, r_2^k, \phi_1^j, \phi_2^l\right)$ have been debugged. First the cases are calculated:

$$
\begin{aligned}
\phi_3^1 &= \phi_{\sum}\left(r_1^{min}, r_2^{min}, \phi_1^{min}, \phi_2^{min}\right) \\
\phi_3^2 &= \phi_{\sum}\left(r_1^{min}, r_2^{max}, \phi_1^{min}, \phi_2^{min}\right) \\
\phi_3^3 &= \phi_{\sum}\left(r_1^{max}, r_2^{min}, \phi_1^{min}, \phi_2^{min}\right) \\
\phi_3^4 &= \phi_{\sum}\left(r_1^{max}, r_2^{max}, \phi_1^{min}, \phi_2^{min}\right) \\
\phi_3^5 &= \phi_{\sum}\left(r_1^{min}, r_2^{max}, \phi_1^{min}, \phi_2^{max}\right) \\
\phi_3^6 &= \phi_{\sum}\left(r_1^{min}, r_2^{max}, \phi_1^{min}, -\frac{\pi}{2} - \sin^{-1}\frac{r_2^{max}}{r_1^{min}}\right) \Leftarrow \phi_2^{min} < -\frac{\pi}{2} - \sin^{-1}\frac{r_2^{m}ax}{r_1^{min}} < \phi_2^{max} \\
\phi_3^7 &= \phi_{\sum}\left(r_1^{min}, r_2^{min}, \phi_1^{max}, \phi_2^{max}\right) \\
\phi_3^8 &= \phi_{\sum}\left(r_1^{min}, r_2^{max}, \phi_1^{max}, \phi_2^{max}\right) \\
\phi_3^9 &= \phi_{\sum}\left(r_1^{max}, r_2^{min}, \phi_1^{max}, \phi_2^{max}\right) \\
\phi_3^{10} &= \phi_{\sum}\left(r_1^{max}, r_2^{max}, \phi_1^{max}, \phi_2^{max}\right) \\
\phi_3^{11} &= \phi_{\sum}\left(r_1^{min}, r_2^{max}, \phi_1^{max}, \phi_2^{min}\right) \\
\phi_3^{12} &= \phi_{\sum}\left(r_1^{min}, r_2^{max}, \phi_1^{max}, \frac{\pi}{2} + \sin^{-1}\frac{r_2^{max}}{r_1^{min}}\right) \Leftarrow \phi_2^{min} < \frac{\pi}{2} + \sin^{-1}\frac{r_2^{max}}{r_1^{min}} < \phi_2^{max} \\
\phi_3^{13} &= \phi_{\sum}\left(r_1^{max}, r_2^{min}, \phi_1^{max}, \phi_2^{min}\right) \\
\phi_3^{14} &= \phi_{\sum}\left(r_1^{max}, r_2^{min}, \phi_1^{min}, \phi_2^{max}\right)
\end{aligned}
$$

The values of ϕ_3^{1-14} are modified to be in the range of $-2\pi \leq \phi_3^n \leq 0$. After that ϕ_3^{1-14} are sorted by their value. The difference between the neighboring cases are calculated, with the biggest and the smallest value being neighbors, too, since the values are angles on a circle! The values that have the biggest difference between each other are taken into account now - they are the new ϕ_3^{min} and ϕ_3^{max} - with the smaller value being min and the bigger being max.

If the difference between the newly calculated ϕ_3^{min} and ϕ_3^{max} is greater than 180° the special case with $\phi^{max} = \pi$, $\phi^{min} = -\pi$ and $r^{min} = 0$ is set because the goal location can be at the reference point now! The same holds if the following conditions are met : $\phi_2^{min} \leq -\pi \leq \phi_2^{max}$ or $r_1^{min} \leq r_2^{min} \leq r_1^{max}$ or $r_1^{min} \leq r_2^{min} \leq r_1^{max}$.

Appendix B

Topology definition

Definition: topology, topological space: Let U be a non-empty set, the universe. A topology on U is a family T of subsets of U that satisfies the following axioms:

1. U and $\emptyset$ belong to T,
2. the union of any number of sets in T belongs to T,
3. the intersection of any two sets of T belongs to T.

A topological space is a pair $[U, T]$. The members of T are called open sets.

In a topological space $[U, T]$, a subset X of U is called a closed set if its complement X^c is an open set, i.e. if X^c belongs to T. By applying the DeMorgan laws, we obtain the properties of closed sets:

1. U and $\emptyset$ are closed sets,
2. the intersection of any number of closed sets is a closed set,
3. the union of any two closed sets is a closed set.

If the particular topology T on a set U is clear or not important, the U can be referred to as the topological space. Closely related to the concept of an open set is that of a neighborhood.

Definition: neighborhood, neighborhood system. Let U be a topological space and $p \in U$ be a point in U.

- $N \subset U$ is said to be a neighborhood of p if there is an open subset $O \subset U$ such that $p \in O \subset N$.

- The family of all neighborhoods of p is called the neighborhood system of p, denoted as N_p.

A neighborhood system N_p has the property that every finite intersection of members of N_p belongs to N_p. Based on the notion of neighborhood it is possible to define certain points and areas of a region.

Def interior, exterior, boundary, closure. Let u be a topological space, $X \subset U$ be a subset of U and $p \in U$ be a point in U.

- p is said to be an interior point of X if there is a neighborhood N of p contained in X. The set of all interior points of X is called the interior of X, denoted $i(X)$.
- p is said to be an exterior point of X of there is a neighborhood N of p that contains no point of X. The set of all exterior points of X is called the exterior of X, denoted $e(X)$.
- p is said to be a boundary point of X if every neighborhood N of p contains at least one point in X and one point is not in X. The set of all boundary points of X is called the boundary of X, denoted $b(X)$.
- The closure of X, denoted $c(X)$, is the smallest closed set which contains X.

The closure of a set is equivalent to the union of its interior and its boundary. Every open set is equivalent to its interior, every closed set is equivalent to its closure.

Def regular open, regular closed. Let x be a subset of a topological space U.

- X is said to be regular open if X is equivalent to the interior of its closure, i.e. $X = i\,(c\,(X))$.
- X is said to be regular closed if X is equivalent to the closure of its interior, i.e. $X = c\,(i\,(X))$.

Two sets of a topological space are called separated if the closure of one set is disjoint from the other set, and vice-versa. A subset of a topological space is internally connected if it cannot be written as a union of two separated sets.

Topological spaces can be categorized according to how points or closed sets can be separated by open sets. Different possibilities are given by the separation axioms T_i. A topological space U that satisfies axiom T_i is called a T_i space. Three of these separation axioms which are important for this work are the following:

T_1 : Given any two distinct points $p, q \in U$, each point belongs to an open set which does not contain the other point. T_2 : Given any two distinct points $p, q \in U$, there exist disjoint open sets $O_p, O_q \subseteq U$ containing p and q respectively. T_3 If X is a closed subset of U and p is a point not in X, there exist disjoint open sets $O_X, O_p \subseteq U$ containing X and p respectively.

A connected space is a topological space which cannot be partitioned into two disjoint open sets, a topological space is regular, if it satisfies axioms T_2 and T_3.

Appendix C

```
#include<stdio.h>
#include <string.h>
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#define M 15
#define N 15
#define M1 7
#define M2 M1+1

//STRUCTURE DEFINED FOR THE LINKED LIST FOR VALID SEGMENTS.
struct node
{
    int x;
    int y;
    int x1;
    int yl;
    int num;
    int numl;
    int flag;
    node* next;
}

int ax2[4];
int a[M][N], b[M][N], p[M][N], q[M][N], at[M1][M1], at1[M2][M2];
char name[10], name1[10], ca[M][N], c, reply, ac[M][N], ac1[M][N];
int m, n, i, j, k, l, x, y;



//NODES FOR LINKED LIST DECLARATION
node* vertary;
node* partary;
node* temp;

//THE MAIN MODULE STARTS HERE
void main()
{
    int load(int i, int j);
    void INITIALISE();
    void APPLY_LAPLACIAN();
    void EDGE_TRACING();
    void TRACE_VERTICES();
}
```

```
void INITIALISE()
{
    for(m=0;m<M;m++)
    {
        for(n=0;n<N;n++)
        {
            ca[M][N] = ' ';
            ac[M][N] = ' ';
            ac1[M][N] = ' ';
        }
    }
    count = 0;
    count1 = 0;
    for(m=0;m<=M-1;m++)
    {
        count++;
        for(n=0;n<=N-1;n++)
        {
            a[m][n] = 0;
            b[m][n] = 0;
            p[m][n] = 0;
            q[m][n] = 0;
            at[m][n] = 0;
            count1++;
            if(count<7 && count1<7) at[m][n] = 0;
        }
    }
    for(i=0;i<7;i++)
    {
        for(j=0;j<7;j++)
        {
            cout<<"at[]="<<at[i][j]<<endl;
        }
    }
}

void APPLY_LAPLACIAN()
{
    for(m=0;m<M-1;m++)
    {
        for(n=0;n<=N-1;n++)
        {
            p[m+1][n] = a[m+1][n] - a[m][n];
            a[0][n] = 0;
        }
    }
    for(m=1;m<M-1;m++)
    {
        for(n=0;n<=N-1;n++)
        {
            a[m][n] = p[m+1][n] - p[m][n];
        }
    }
```

```
    for(n=0;n<=N-1;n++) a[M-1][n] = 0;
    for(n=0;n<N-1;n++)
    {
        for(m=0;m<=M-1;m++)
        {
            q[m][n+1] = b[m][n+1] - b[m][n];
            b[m][0] = 0;
        }
    }
    for(n=1;n<N-1;n++)
    {
        for(m=0;m<=M-1;m++)
        {
            b[m][n] = q[m][n+1] - q[m][n];
        }
    }
    for(m=0;m<=M-1;m++) b[m][N-1)=0;
    for(m=0;m<=M-1;m++)
    {
        for(n=0;n<=N-1;n++)
        {
            a[m][n] = a[m][n] + b[m][n];
            p[m][n] = a[m][n];
        }
    }
}

void EDGE_TRACING()
{
    for(m=0;m<M:m++)
    {
        for(n=0;n<N;n++)
        {
            if(p[m][n]<=-1) ca[m][n] = '*';
            else ca[m][n] = ' ';
        }
    }
}

void TRACE_VERTICES()
{
    for(m=1;m<M-1;m++)
    {
        for(n=1;n<N-1;n++)
        {
            if(ca[m][n]==' ')
            ac[m][n] = ca[m][n];
            else
            {
                if((ca[m][n]=='*' && ca[m-1][n]=='*' && ca[m+1][n]=='*') ||
                   (ca[m][n]=='*' && ca[m][n-1]=='*' && ca[m][n+1]=='*') ||
                   (ca[m][n]=='*' && ca[m-1][n-1]=='*' && ca[m+1][n+1]=='*') ||
                   (ca[m][n]=='*' && ca[m-1][n+1]=='*' && ca[m+1][n-1]=='*'))
                ac[m][n] = '*';
```

```
                else ac[m][n] = 'v';
            }
            if(ca[m][n]=='*' && ca[m][n-1]=='*' && ca[m+1][n-1]=='*' &&
               ca[m+1][n]=='*' && ca[m-1][n]=='*' && ca[m][n]!='v' &&
               ca[m][n-1]=!'v' && ca[m+1][n-1]!='v' && ca[m+1][n]!='v' &&
               ca[m-1][n]!='V' && ca[m][n-1]!='v' && ca[m+1][n-1]!='v')
            ac[m][n] = 'v';
            if(ca[m][n]=='*' && ca[m][n+1]=='*' && ca[m+1][n+1]=='*' &&
               ca[m+1][n]=='*' && ca[m-1][n]=='*' && ca[m][n]!='v' &&
               ca[m][n+1]=!'v' && ca[m+1][n+1]!='v' && ca[m+1][n]!='v' &&
               ca[m-1][n]!='V' && ca[m][n-1]!='v' && ca[m+1][n-1]!='v')
            ac[m][n] = 'v';
            if(ac[m][n]=='v' && ac[m-1][n]=='v' && ac[m+1][n]==' ')
            ac[m-1][n] = '*';
            if(ac[m][n]=='v' && ac[m-1][n]==' ' && ac[m+1][n]=='v')
            ac[m-1][n] = '*';
        }
    }
}
```

The original image:

The modelled image:

```
      * * * * * *
    * * * * * * *
* * * * * *     *
*             *   *
*             *
*             *
```

The binary form of the image:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 0
0 0 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0 1 0
0 1 0 0 0 0 1 0 1 0
0 1 0 0 0 0 1 0 0 0
0 1 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0
```

The Laplacian of the binary image:

```
0 0 0 0 0 0 0 0 0 0
0 0 0-2-3-3-3-3-2 0
0 0-2 0 0 0 0-1-1 0
0-2-1-1-1-1-1 0-3 0
0-2 0 0 0 0-2 0-2 0
0-2 0 0 0 0-2 0 0 0
0-2 0 0 0 0-2 0 0 0
0-2 0 0 0 0 0 0 0 0
```

The Wire Grid Model:

The Wire Grid Model with vertices:

The SPP is implemented in C. It uses a Distance/orientation-interval propagation implementa -tion which also has been written in C. The representation of the basic relations is done by setting bits in a byte array. The number of orientations m is defined as “ROT_SLICES” (for rotation) and the number of distances n is defined as “TRANS_SLICES” (for translation). SPP are named "Pieces". The various input and output functions come pairwise - the ones with an A at the end request the orientation and the distance index of the atomic relation whereas the ones with an B at the end request the bit-number of the atomic relation.

The BYTE_NUM(x) (x/8) and BIT_NUM(x) (x%8) directives are used to access the actual bit in the array. Some short exemplary functions are shown to demonstrate the implementation.

The procedures to set a specific atomic relation to an value (either 0 or 1) look like this:

```
void setA(Pieces *pcs, int rot, int trans, int value){
    setB(pcs,rot+trans*ROT_SLICES,value);
}

void setB(Pieces *pcs, int which, int value){
    if(value==0){
        pcs->pPcs[BYTE_NUM(which)] =
            ~(1 << BIT_NUM(which)) & pcs->pPcs[BYTE_NUM(which)];
    }else{
        pcs->pPcs[BYTE_NUM(which)] =
            1 << BIT_NUM(which) | pcs->pPcs[BYTE_NUM(which)];
    }
}
```

The get functions:

```
int getA(Pieces *pcs, int rot, int trans){
    return getB(pcs, rot+trans*ROT_SLICES);
}

int getB(Pieces *pcs, int which){
```

```
    return (((1 << BIT_NUM(which)) & pcs->pPcs[BYTE_NUM(which)]) > 0);
}
```

The union of two SPPs:

```
void pieceORpiece(Pieces *rtn, Pieces *pcs1, Pieces *pcs2){
    int i;

    // setting the last unused bits in the last byte to 0
    pcs1->pPcs[BYTES_USED-1] &= ~(255<<BIT_NUM(PIECES));
    pcs2->pPcs[BYTES_USED-1] &= ~(255<<BIT_NUM(PIECES));

    for(i=0; i<BYTES_USED; i++){
        rtn->pPcs[i] = (pcs1->pPcs[i] | pcs2->pPcs[i]);
    }
}
```

The composition using DOI:

```
void composition(Pieces *rtn, Pieces *pcs1, Pieces *pcs2){
    DoiTyp doi1, doi2, comp;
    Pieces tmp;
    Searchy search1,search2;
    int x,y;

    tmp.pPcs = getPiecesArray();

    search1.pPiece = pcs1;
    search1.number = -1;    // initializing the serach
    search2.pPiece = pcs2;
    // searching through the first SPP for atomic relations
    while((x = getNextTrue(&search1))>= 0){
        search2.number = -1; // initializing the serach
        // searching through the second SPP for atomic relations
        while((y = getNextTrue(&search2))>= 0){
            singleDoiB(&doi1,x); // generating a DOI with
                                 // the atomic relation x
            singleDoiB(&doi2,y); // generating a DOI with
                                 // the atomic relation y
```

```
            // the composition
            composition_exact(&doi1,&doi2,&comp,"");
            clearPiece(&tmp);
            // now a FPSS is being generated out of
            // the DOI composition result
            doiToAnder(&tmp,&comp);
            // the union of this atomic composition result
            // with the former composition results
            pieceORpiece(rtn,rtn,&tmp);
        }
    }
    free(tmp.pPcs);
}
```

The unary operation Short cut Sc:

```
void scCalc(Pieces *rtn, Pieces *in){
    Searchy search;
    int x,oldRot,oldTrans,i;

    search.pPiece = in;
    search.number = -1;
    while((x = getNextTrue(&search))>= 0){
        oldRot      = getRot(x);
        oldTrans    = getTrans(x);

        if(oldRot < ROT_SLICES/2){
            for(i = ROT_SLICES/2; i <= ROT_SLICES/2+oldRot; i++){
                setA(rtn,i,oldTrans,1);
            }
        }else{
            for(i = oldRot-ROT_SLICES/2; i < ROT_SLICES/2; i++){
                setA(rtn,i,oldTrans,1);
}}}}
```

Test if two SPPs are equal:

```
int testEqual(Pieces *pcs1, Pieces *pcs2){
```

```
    // set the unused bits at the end to 0
    pcs1->pPcs[BYTES_USED-1] &= ~(255<<BIT_NUM(PIECES));
    pcs2->pPcs[BYTES_USED-1] &= ~(255<<BIT_NUM(PIECES));

    return (memcmp(pcs1->pPcs,pcs2->pPcs,BYTES_USED)==0);
}
```

A program that is successfully demonstrating the Sc operation, Pavlidis algorithm and the the composition:

```
int main(int argc, char* argv[])
{
    // the SPPs
    Pieces a,b,c,d,e,f;
    // testing if the defines are set correctly
    pretest();

    // allocate memory for the byte arrays
    initPieces(&a);
    initPieces(&b);
    initPieces(&c);
    initPieces(&d);
    initPieces(&e);
    initPieces(&f);

    // initialize an SPP by spezifing atomic relations
    // using their orientation and distance index
    setA(&a,6,3,1);
    setA(&a,6,4,1);
    setA(&a,5,3,1);

    // initialize an SPP using a metric coordinate system
    setPoint(&b,6,39,1);

    // initialize an SPP by spezifing atomic relations
    // using their bit number in the array
    setB(&c,58,1);
    setB(&c,59,1);

    // ASCII output of the SPPs
```

```
printf(" SPP a \n");
printPieces(&a);
printf(" SPP b \n");
printPieces(&b);
printf(" SPP c \n");
printPieces(&c);

// calculate the short cut of b - store the result in d
unary(&d,&b,2);

// print the Sc of b
printf("\n\n result of SC b \n");
printPieces(&d);

// set all atomic relation to 0
clearPiece(&d);

// composition of a and b - store the result in d
composition(&d,&a,&b);

// print the result of the composition
printf("\n\n composition result 1\n");
printPieces(&d);

// composition of a and d - store the result in e
composition(&e,&a,&d);

// do contour tracing and set the SPP to the contour
calcBoundaryPavlidi(&d);
setPcsLikeBoundary(&d);

// output of the contour
printf("\n\n contour of composition \n");
printPieces(&d);

// composition with the contour
composition(&f,&a,&d);

// output to compare the compusition results with
// and without contour
```

```
    printf("\n\n composition 2 (calculated without contour)\n");
    printPieces(&e);
    printf("\n\n composition 2 (calculated out contour)\n");
    printPieces(&f);

    // using a function to test for equality
    printf("\n e and f equal ? %d \n",testEqual(&e,&f));

    // do contour tracing and set the SPP to the contour
    calcBoundaryPavlidi(&f);
    setPcsLikeBoundary(&f);

    // print the contour
    printf("\n\n contour of composition \n");
    printPieces(&f);

    // freeing the memory
    free(a.pPcs);
    free(b.pPcs);
    free(c.pPcs);
    free(d.pPcs);
    free(e.pPcs);
    free(f.pPcs);
}
```

This is the output of the program above (it uses 18 orientation distinctions and 20 distances).

```
SPP a
         TRANS
        00        10        20
        |    .    |    .    |
  000 : 00000000000000000000
R 001 : 00000000000000000000
O 002 : 00000000000000000000
T 003 : 00000000000000000000
  004 : 00000000000000000000
  005 : 00010000000000000000
  006 : 00011000000000000000
  007 : 00000000000000000000
  008 : 00000000000000000000
  009 : 00000000000000000000
        |    .    |    .    |
  010 : 00000000000000000000
  011 : 00000000000000000000
  012 : 00000000000000000000
  013 : 00000000000000000000
  014 : 00000000000000000000
  015 : 00000000000000000000
  016 : 00000000000000000000
  017 : 00000000000000000000
SPP b
         TRANS
        00        10        20
        |    .    |    .    |
  000 : 00000000000000000000
R 001 : 00000000000000000000
O 002 : 00000000000000000000
T 003 : 00000000000000000000
  004 : 00000000000000010000
  005 : 00000000000000000000
  006 : 00000000000000000000
  007 : 00000000000000000000
  008 : 00000000000000000000
  009 : 00000000000000000000
        |    .    |    .    |
  010 : 00000000000000000000
  011 : 00000000000000000000
  012 : 00000000000000000000
  013 : 00000000000000000000
  014 : 00000000000000000000
  015 : 00000000000000000000
  016 : 00000000000000000000
  017 : 00000000000000000000
SPP c
         TRANS
        00        10        20
        |    .    |    .    |
  000 : 00000000000000000000
R 001 : 00000000000000000000
O 002 : 00000000000000000000
T 003 : 00000000000000000000
  004 : 00010000000000000000
  005 : 00010000000000000000
  006 : 00000000000000000000
  007 : 00000000000000000000
  008 : 00000000000000000000
  009 : 00000000000000000000
        |    .    |    .    |
  010 : 00000000000000000000
  011 : 00000000000000000000
  012 : 00000000000000000000
  013 : 00000000000000000000
  014 : 00000000000000000000
  015 : 00000000000000000000
  016 : 00000000000000000000
  017 : 00000000000000000000
```

```
result of SC b
         TRANS
        00        10        20
        |    .    |    .    |
  000 : 00000000000000000000
R 001 : 00000000000000000000
O 002 : 00000000000000000000
T 003 : 00000000000000000000
  004 : 00000000000000000000
  005 : 00000000000000000000
  006 : 00000000000000000000
  007 : 00000000000000000000
  008 : 00000000000000000000
  009 : 00000000000000010000
        |    .    |    .    |
  010 : 00000000000000010000
  011 : 00000000000000010000
  012 : 00000000000000010000
  013 : 00000000000000010000
  014 : 00000000000000000000
  015 : 00000000000000000000
  016 : 00000000000000000000
  017 : 00000000000000000000


composition result 1
         TRANS
        00        10        20
        |    .    |    .    |
  000 : 00000000000000000000
R 001 : 00000000000000000000
O 002 : 00000000000000000000
T 003 : 00000000000000000000
  004 : 00000000000000000000
  005 : 00000000000000000000
  006 : 00000000000000000000
  007 : 00000000000000000000
  008 : 00000000000000111100
  009 : 00000000000000111100
        |    .    |    .    |
  010 : 00000000000000111100
  011 : 00000000000000111100
  012 : 00000000000000000000
  013 : 00000000000000000000
  014 : 00000000000000000000
  015 : 00000000000000000000
  016 : 00000000000000000000
  017 : 00000000000000000000


contour of composition
         TRANS
        00        10        20
        |    .    |    .    |
  000 : 00000000000000000000
R 001 : 00000000000000000000
O 002 : 00000000000000000000
T 003 : 00000000000000000000
  004 : 00000000000000000000
  005 : 00000000000000000000
  006 : 00000000000000000000
  007 : 00000000000000000000
  008 : 00000000000000111100
  009 : 00000000000000100100
        |    .    |    .    |
  010 : 00000000000000100100
  011 : 00000000000000111100
  012 : 00000000000000000000
  013 : 00000000000000000000
  014 : 00000000000000000000
  015 : 00000000000000000000
  016 : 00000000000000000000
  017 : 00000000000000000000
```

```
composition 2 (calculated without contour)
         TRANS
        00        10        20
        |    .    |    .    |
  000 : 00000000011111111000
R 001 : 00000000001111111000
O 002 : 00000000001111000000
T 003 : 00000000000000000000
  004 : 00000000000000000000
  005 : 00000000000000000000
  006 : 00000000000000000000
  007 : 00000000000000000000
  008 : 00000000000000000000
  009 : 00000000000000000000
        |    .    |    .    |
  010 : 00000000000000000000
  011 : 00000000000000000000
  012 : 00000000001111110000
  013 : 00000000011111110000
  014 : 00000000011111110000
  015 : 00000000011111110000
  016 : 00000000011111111000
  017 : 00000000011111111000


composition 2 (calculated out contour)
         TRANS
        00        10        20
        |    .    |    .    |
  000 : 00000000011111111000
R 001 : 00000000001111111000
O 002 : 00000000001111000000
T 003 : 00000000000000000000
  004 : 00000000000000000000
  005 : 00000000000000000000
  006 : 00000000000000000000
  007 : 00000000000000000000
  008 : 00000000000000000000
  009 : 00000000000000000000
        |    .    |    .    |
  010 : 00000000000000000000
  011 : 00000000000000000000
  012 : 00000000001111110000
  013 : 00000000011111110000
  014 : 00000000011111110000
  015 : 00000000011111110000
  016 : 00000000011111111000
  017 : 00000000011111111000

e and f equal ? 1


contour of composition
         TRANS
        00        10        20
        |    .    |    .    |
  000 : 00000000010000001000
R 001 : 00000000001000111000
O 002 : 00000000001111000000
T 003 : 00000000000000000000
  004 : 00000000000000000000
  005 : 00000000000000000000
  006 : 00000000000000000000
  007 : 00000000000000000000
  008 : 00000000000000000000
  009 : 00000000000000000000
        |    .    |    .    |
  010 : 00000000000000000000
  011 : 00000000000000000000
  012 : 00000000001111110000
  013 : 00000000010000010000
  014 : 00000000010000010000
  015 : 00000000010000010000
  016 : 00000000010000001000
  017 : 00000000010000001000
```

Appendix D

Bibliography

1. Robotics: Control, Sensing, Vision and Intelligence - Gonjaledz and Fu, Mc. Graw Hill Publication.
2. Advanced Analytical Geometry – J.G. Chakraborty, P.R. Ghosh, U.N. Dhar and Sons Pvt. Ltd.
3. Artificial Intelligence – Thomas Dean, James Allen and Yiannis Aloimonds, Adeson Wesley Publication.
4. Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain – Amit Konar, CRC Press
5. On the Utilization of Spatial Structures for Cognitively Plausible and Efficient Reasoning - Christian Freksa and Kai Zimmermann, IEEE Conference.
6. Qualitative Spatial Reasoning Using Orientation, Distance, and Path Knowledge - Kai Zimmermann and Christian Freksa, IJCAI Workshop on Spatial and Temporal Reasoning.
7. Qualitative spatial reasoning about relative point position - Reinhard Moratz and Marco Ragni, Elsevier.
8. Spatial Reasoning for Robot Exploration - Reinhard Moratz and Jan Oliver Wallgrun
9. Algorithms for Graphics and Image Processing – Theo Pavlidis, Computer Science Press.