



**Modern University for technology
and Information
Faculty of Computers & Artificial
intelligence
Department of information system**



**Shortest path planning in polygonal world for an
Autonomous Mobile Robot
Graduation project 2024**

Submitted by:

Peter Hany Hosny shafek
Mahmoud Ahmed Mohammed Tawfik
Hamed Mohamed Hamed
Ahmed Emad Saeed Khllaf
Wesam Soltan Ahmed Soltan
Ahmed Tarek Saber Amin
Saleh Abdulhakim Saleh

Supervised by:

**Prof. Dr. Mahmoud Wahdan
TA. Hazem Omar
TA. Osama Hassan
TA. Ahmed Huzzain**

ACKNOWLEDGMENT

Our greatest in deep and gratitude to Modern University for Technology and Information (M.T.I) for the great role it played in forming our character and personalities, providing us with all means necessary for our educational development and providing us as well with a wonderful environment to learn in throughout our educational years.

This would not have been possible unless Prof. Dr. Olfat Kamel, President of the

M.T.I University has established such a learning environment, providing the most effective facilities for our job to be done.

It is a pleasure to thank those who maneuvered this Possible, Prof. Dr. Mohamed Taher ElMayah Dean of the faculty of Computers and Information & Artificial intelligence for his unlimited help, support, encouragement and guidance.

A Special thanks to our head of the Information System Department Prof. Dr. Hafez Abd El-Wahab are Honored and got the pleasure of working with Prof. Dr. Mahmod Wahdan , and our teaching assistants TA. Osama Hassan , TA. Hazem Omar and TA. Ahmed Huzzain.

As well as the Department staff, Prof. Dr. Alaa Abd El-Raheem and Prof. Dr. Mahmoud El-Shishtawy for their effort along our university studies.

A Special thanks to our head of the Computer Science Department Prof. Dr. Hanafy Isamil, as well as Prof. Dr. Hesham El Deeb for their effort along our university studies.

Special thanks to Prof. Dr. Emain Taha, head of Basic Science Department. As well as the department staff, Prof. Dr. Elsayed Bakkar for their effort teaching us the basic for our computer science degree, Prof. Dr.Rasha Mohammad, Prof.Dr.Marwa Refaey and Prof.Dr.Rania Ahmed Lecture in Basic science Department.

Special thanks to our families who always supported us during our academic life, and we could not do it without them with us.

TABLE OF CONTENTS

1.	<u>INTRODUCTION</u>	5
1.1	BACKGROUND.....	6
1.2	PROBLEM STATEMENT.....	8
1.2.1	Definition.....	8
1.2.2	Problem Description.....	8
2.	<u>POLYGONS</u>	11
2.1	POLYGONS AND THEIR ORIENTATION.....	12
2.2	POLYGONAL WORLD.....	19
2.3	Polygonal World and Path Class.....	20
2.3.1	Directed v-edge.....	21
2.3.2	Canonical Paths and Directed v-edges Sequences.....	23
2.3.3	Connectivity Graph.....	27
2.3.4	Path Class Representation.....	29
2.4	Finding the Best Path Class.....	32
2.5	Image of Point on Convex Polygon.....	32
2.5.1	Visibility from Point to Polygon.....	33
2.5.2	Type of an Image from a Point to a Convex Polygon...	37
2.5.3	The Image Type Algorithm.....	40
2.5.3.1	Proof of Correctness of the Algorithm.....	42
2.5.3.2	Analysis of the Worst-Case Time Complexity of the Algorithm.....	43
2.5.4	Finding an Image on a Non-convex Polygon.....	43
2.6	Path Classes and Sub-Polygons.....	45
2.7	Advantages of Path Class Representation Using Directed V-Edges Sequences.....	49

3.	<u>SHORTEST PATH PLANNING ALGORITHM</u>	51
3.1	CANONICAL PATHS AND TANGENT SEQUENCES.....	52
3.2	TANGENTS ON POLYGONS.....	57
3.2.1	Tangents from Point to Polygon.....	58
3.2.2	Tangents from Polygon to Point.....	61
3.2.3	Common Tangents between two Polygons.....	62
3.3	VISIBILITY.....	64
3.4	SHORTEST PATH FINDING ALOGRITHM.....	67
3.4.1	Comparison between two Paths.....	73
4.	CONCLUSIONS	73
	APPENDIX A. SOFTWARE DESIGN	00
	APPENDIX B. SOURCE CODE	00
	APPENDIX C. USER GIUDE	00
	LIST OF REFERENCES	00

Table of Figures

Figure 1.1 Canonical path.....	13
Figure 2.1 Interior and exterior angle of a simple polygon.....	17
Figure 2.2 Convex and concave simple polygons.....	15
Figure 2.3 Cross product of vectors.....	16
Figure 2.4 Intersecting segments	18
Figure 2.5 Interior and exterior of a simple polygon	20
Figure 2.6 Polygonal world.....	20
Figure 2.7 Generalized Voronoi diagram of polygonal world (1)	21
Figure 2.8 Generalized Voronoi diagram of polygonal world (2)	22
Figure 2.9 Defining directed v-edge for (ccw polygons)	23
Figure 2.10 Defining directed v-edges for (cw & ccw)	24
Figure 2.11 Paths and canonical paths.....	24
Figure 2.12 Interpretation of canonical path as directed v-edges sequence.....	26
Figure 2.13 Directed v-edges sequence $[\gamma_1]$	27
Figure 2.14 Directed v-edges sequence $[\gamma_2]$	28
Figure 2.15 Basic connectivity graph of a polygonal world (1).....	29
Figure 2.16 Basic connectivity graph of a polygonal world (2).....	30
Figure 2.17 Polygonal world (1).....	30
Figure 2.18 Augmented connectivity graph of a polygonal world.....	31
Figure 2.19 Polygonal world (II).....	32
Figure 2.20 Augmented connectivity graph of a polygonal world (II).....	32
Figure 2.21 Image of Point on convex polygon.....	33
Figure 2.22 Image on object.....	34
Figure 2.23 Images on world.....	34
Figure 2.24 Visibility from point p to convex polygon B.....	35
Figure 2.25 Classifications of vertex v_i of polygon B with respect to a segment $\overline{pv_i}$	36
Figure 2.26 Visibility from point p to convex polygon B.....	37

Figure 2.27 Images of point P lies on an edge of convex polygon B....	39
Figure 2.28 Images of point P lies on vertex v_i of convex polygon B...	40
Figure 2.29 Image type block diagram.....	41
Figure 2.30 Correctness of image type algorithm.....	43
Figure 2.31 Image of a point P on cw concave polygon B.....	44
Figure 2.32 Image of a point P on ccw concave polygon B.....	45
Figure 2.33 Problem1: initial orientation of a vehicle is different from the direction of a motion.....	47
Figure 2.34 Problem2: voronoi diagram of polygonal world consisting of two polygons (ccw polygon inside cw polygon boundary).....	48
Figure 2.35 solution of problem1: voronoi diagram of a sub polygonal world.....	48
Figure 2.36 Basic connectivity graph of a sub polygonal world.....	49
Figure 2.37 Augmented connectivity graph a sub polygonal world.....	49
Figure 2.38 Solution of problem2: world and augmented connectivity graph.....	50
Figure 3.1 Examples into paths and visibility.....	54
Figure 3.2 Tangent sequence $A^-B^-C^+$ and its canonical path.....	55
Figure 3.3 Tangents.....	59
Figure 3.4 Tangents from point to polygon.....	60
Figure 3.5 Tangents from polygon to point.....	62
Figure 3.6 Tangents between two polygons.....	63
Figure 3.7 Visibility.....	65
Figure 3.8 All tangents from s	66
Figure 3.9 All visible tangents.....	68
Figure 3.9 All visible tangents.....	70
Figure 3.10 Comparison of paths.....	75

CHAPTER.1

INTRODUCTION

1.1 BACKGROUND

There are two goals for planning autonomous vehicle navigation planning: shortest path and safe path. These goals are often in conflict. We will discuss the most dangerous path considering the path length being the most important factor in robot path planning.

The problem of planning the path of a robot around obstacles has attracted quite a bit of attention over the past several years. The two main types of path planning algorithm are graph searching and potential field. The graph searching techniques also fall into two main areas, those that determine a safe path by tracing around the obstacles and those that the free space, which is the complement of the space occupied by the obstacles.

The visibility graph [1] was an early technique for tracing around obstacles. By using the robot's configuration space, the problem of planning the path of the robot was reduced to planning the path of a point representing the robot. The visibility graph shows which vertices of the obstacles are visible from other vertices. A safe path is generated by piecing together several of these safe path segments. This algorithm is typical of the tracing based techniques.

Three typical free space searching techniques are: the generalized cones [2], the Voronoi diagram [3], and the subdivision of the free space [4]. In the first technique, the free space is represented by generalized cones whose sides are the face for the obstacles. A safe path is produced by piecing together the center lines of the cones the voronoi diagram in two dimensions is a collection of curves that are equi-distant from two or more obstacles. A safe path is found by moving along these curves. In three dimensions, the curves become surfaces that are equidistant from two or more obstacles. In the subdivision algorithms, the space is subdivided until a region is either completely filled with an obstacle or the region is empty. A safe path is then determined by moving from one free region to another to the goal. The A^* algorithm [5] is frequently employed for searching the subdivided free space.

Artificial potential field algorithm [6] uses repulsive potential fields around the obstacles to force away the robot, which is subjected this potential, and an attractive potential field around the goal to attract the robot. Another potential field algorithm [7] begins with a trial path through the space and uses repulsive potential fields around the obstacles to modify the path so that it moves toward the free space until a safe path is found. The difference to note here is that the potential field effects

the entire path simultaneously as opposed to the current robot position. This was shown to be less sensitive to local minima in the potential field than other potential field based algorithms.

The most effective of the free space algorithms, in terms path length optimization and thoroughness, is the A* technique [5]. However, it has a tendency to be slow planner due to the amount of space that is searched. Other artificial features are that it is easily extended to higher dimensions and obstacles are not restricted to nice, geometric shapes. If there was a way to increase the speed for this algorithm, it could be ideal for real-time path planning.

An unrelated technique that is quite fast is the vector based method [8]. In this technique, a trial vector is drawn from the starting position (s) to the goal (g). If the vector crosses an obstacle, a new intermediate goal (IG) is created and the previous goal is pushed on to a stack. The intermediate goal is obtained by drawing a new vector from the centroid (c) of the crossed obstacle through the midpoint (m) of the vector segment that crossed the obstacle. This vector is continued until it is clear of the obstacle plus a small extension for margin of safety. This point is used as an intermediate goal. The planning procedure is then repeated until there is a safe path to an intermediate goal. At this time, the starting point is moved to the intermediate goal, the intermediate goal is popped from the stack. This procedure continues until the final goal is reached, thus yielding a safe path from the original starting point. This brief survey is by no means complete, but it does show some of the typical algorithms. Techniques have also been presented for avoiding moving obstacles [9] and obstacles with uncertainty [10].

1.2 PROBLEM STATEMENT

1.2.1 Definitions

This subsection defines a list of terms and concepts used throughout this project.

Let \mathfrak{R} denote the set of real numbers. The environment for this task of this project is a two-dimensional plane \mathfrak{R}^2 on which a global Cartesian coordinate system is defined.

Let B_1, \dots, B_n be fixed objects (convex polygons) distributed in \mathfrak{R}^2 . These B_i 's are called *obstacles*.

A *world* W is a set of n convex polygonal obstacles,

$$W = \{B_0, B_1, \dots, B_n\}, \quad n > 0$$

where B_0 is the outermost polygonal boundary, B_1, \dots, B_n are polygonal obstacles inside the boundary, and no pair of polygons intersects or touches.

The *free space* $\text{free}(W)$ is the inside of B_0 minus the union of the n polygons contained in B_0 . In other words, the free space is the complement of the union of all polygons in W .

We consider path f to be directed curve with natural direction from $f(0)$ to $f(1)$. A *path* f in W is a continuous function

$$f: [0, 1] \rightarrow \text{free}(W)$$

with $f(0) \neq f(1)$. The two points $f(0)$ and $f(1)$ are called its *endpoints*, and the path *joins* them. If they are distinct, we usually denote $f(0)$ as a start S and $f(1)$ as a goal G .

• Problem Description

The purpose of this project is to find an algorithm to solve the shortest path planning problem for a point robot in a polygonal world. This is one of the most fundamental problems in robotics. In this project, actually we discuss the most dangerous path considering the path length being the most important factor in robot path planning.

In this project, a world W which consists of n convex polygons, and two points S and G in $\text{free}(W)$ are given. (Notice that W contains no CW polygons.) We want to find the shortest path in $\text{free}(W) \cup W$ connecting the two points (see Figure 1).

In order to find the (globally) shortest path, we first find the locally shortest path in each path class. Next we will find the globally shortest path among those locally shortest paths. Only “locally shortest paths” in path classes. A locally shortest path is called “canonical path” (see Figure 1). By this concept, we will obtain an efficient algorithm for this problem.

Canonical Paths:

1. **Locally Shortest Path:** A path within a specific "path class" (explained below) that is the shortest path for that particular class.
2. **Path Class:** A categorization of possible paths based on how they interact with the obstacles. Different path classes might involve the robot going around obstacles entirely, partially entering some obstacles, or completely traversing specific obstacles (depending on the project's specific definition).
3. **Canonical Path:** By finding the locally shortest path in each path class and then comparing them, the algorithm aims to identify the **globally shortest path** that connects S and G.

Overall Approach:

- **Classify Possible Paths:** Divide potential paths into different categories based on their interaction with obstacles (path classes).
- **Find Locally Shortest Paths:** Within each path class, identify the path with the minimum length that connects S and G.
- **Compare Locally Shortest Paths:** Compare the lengths of the locally shortest paths from each class.
- **Identify Globally Shortest Path:** The path with the minimum length among all the locally shortest paths is considered the globally shortest path for the robot to travel from S to G.

Benefits of Canonical Paths:

- This approach might lead to an **efficient algorithm** for finding the shortest path by focusing on specific path categories instead of exploring all possible paths blindly.

Additional Notes:

- The specific details of path classification and how the algorithm efficiently searches within each class are likely explained in the full project description.
- This project focuses on theoretical concepts rather than a physically realistic scenario where the robot can't pass through obstacles.

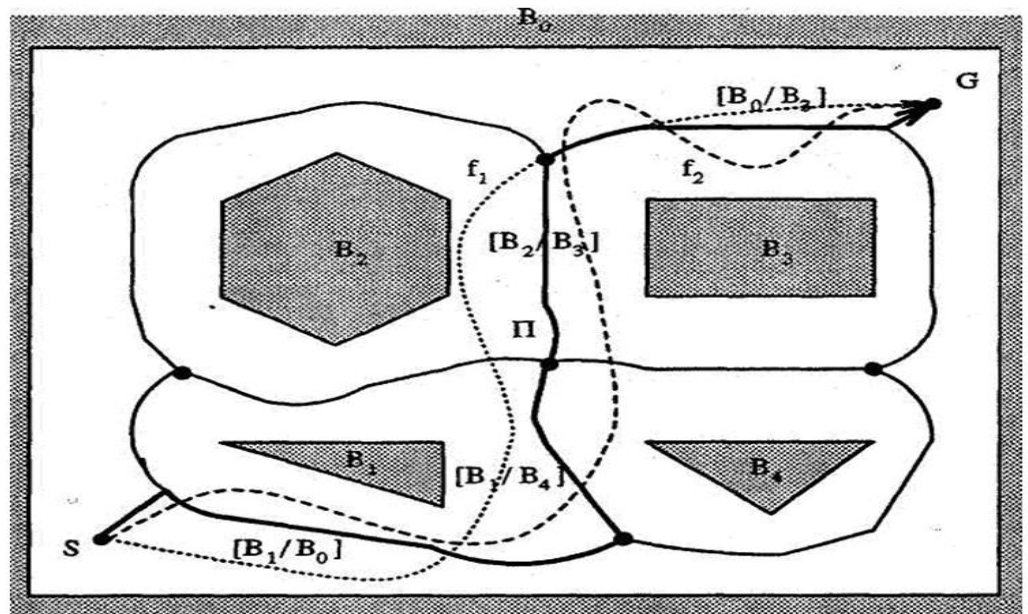


Figure 1.1 canonical path

Chapter.2

Project Plan

Project Plan

2.1. Introduction

This chapter is a discipline for saying how to complete a project within a certain timeframe, usually with defined stages, and with designated resources.

2.2. Project Task Description

The project includes the following tasks:

- Analysis and Data Collection
- System Design
- System Implementation
- Testing
- Documentation
- Demo

Final Presentation The description of the above tasks is written below:

1) Data Collection Task: Involves actions and methods performed on data that help describe facts, detect patterns, develop explanations, and test hypotheses. This includes data quality assurance, statistical data analysis, modelling, and interpretation of analysis results. o Data collection: Acquisition involves collecting or adding to the data holdings.

There are several methods of getting data:

- collecting new data.
- using your own previously collected data.
- reusing someone's other data.
- purchasing data.

2) System Design Task: System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

3) System Implementation Task: Describe the database environment where the software system and the database(s), if any, will be installed. Include a description of the diverse types of database and library environments (such as production, test, and training databases).

4) System Documentation: A set of documents provided on paper used as a quick reference or a guidebook to help the reader understand what is going on.

5) System Testing Task: The type of testing to check the behavior of a complete and fully integrated software product based on the software requirements.

6) Presentation: Representing the final work and everything that was done project by explaining what was learned and what was done by whom and how long it took to be done.





















7) Demo: A lost and found system demo is a freely distributed piece of an upcoming or recently released app. Demos are typically released by the system publisher to help consumers get a feel of the system before deciding whether to buy the full version.

2.3. Software Tools

Microsoft Project is a powerful project planning software developed by Microsoft. It helps users create project schedules, define tasks, allocate resources, and track progress. With features like Gantt charts, resource management, and collaboration tools, it supports efficient project management and ensures successful project completion. The task made by (Microsoft Project) Microsoft Project can:

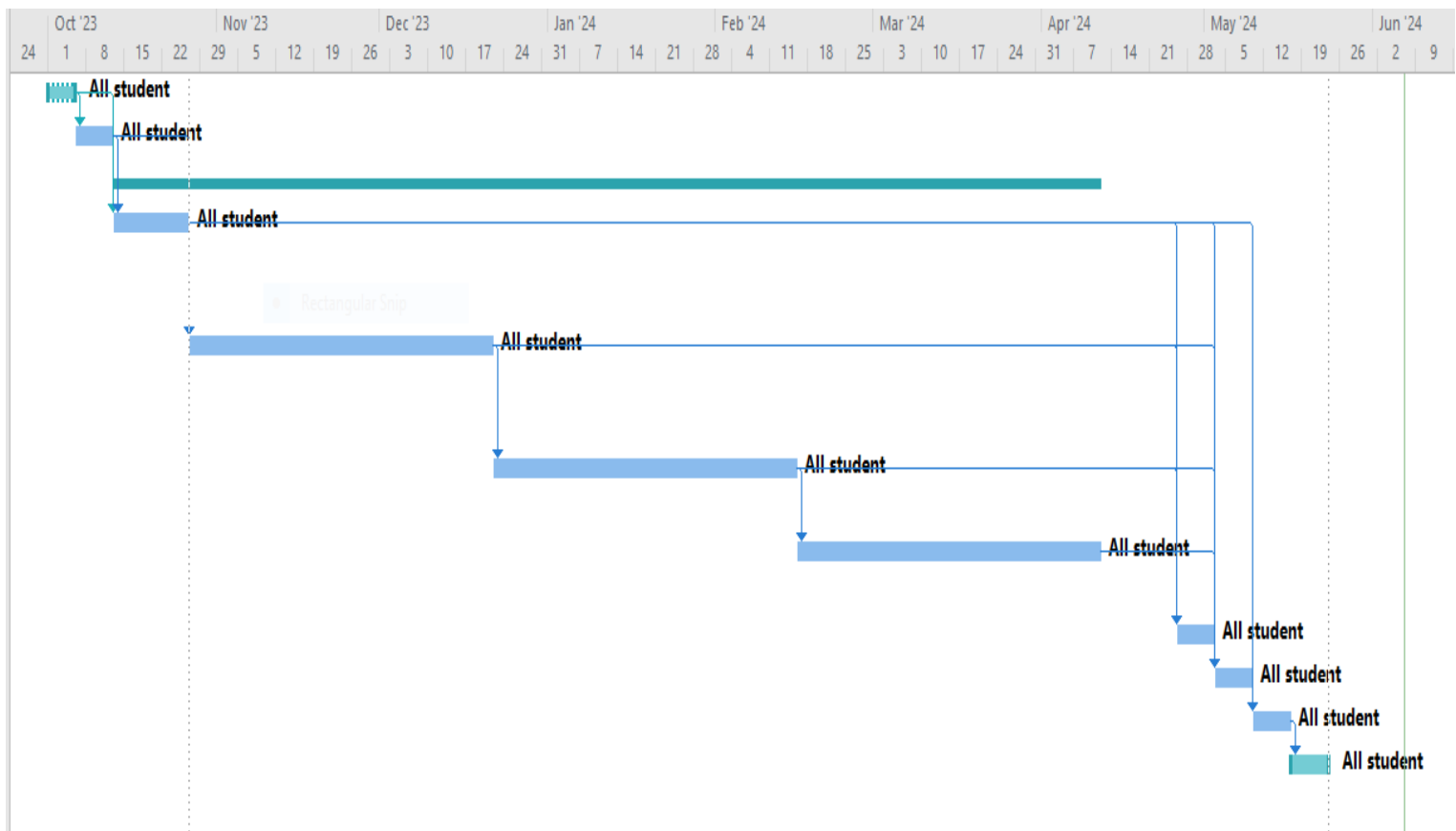
- Draw the action plan and represent it on both the Network Diagram and the Gantt chart.
- Allocate and organize resources for each activity.
- Follow-up progress of the project.
- Project budget management, and workload analysis.
- The possibility of programming work - as in all programs Microsoft Office package

2.3.1 Project Subsystems

		Task Mode	Task	Duration	Start	Finish	Predecessors	Resource Names
1			Defining Problem	1 wk	Sun 10/1/23	Thu 10/5/23		All student
2			Gathering Requirements	1 wk	Fri 10/6/23	Thu 10/12/23	1	All student
3			System Analysis & Design					
4			path Planing Problem , the most Important classical methods	2 wks	Fri 10/13/23	Thu 10/26/23	1,2	All student
5			polygons and Theory of a Free-Space Decomposition	8 wks	Fri 10/27/23	Thu 12/21/23	2	All student
6			Canonical Path & Tangent Sequences	8 wks	Fri 12/22/23	Thu 2/15/24	5	All student
7			Shortest Path Finding Algorithm	8 wks	Fri 2/16/24	Thu 4/11/24	6	All student
8			System Presentation	1 wk	Fri 4/26/24	Thu 5/2/24	4,5,6,7	All student
9			System Testing	1 wk	Fri 5/3/24	Thu 5/9/24	4,5,6,7	All student
10			System Documentation	1 wk	Fri 5/10/24	Thu 5/16/24	4	All student
11			System Demo	1 wk	Fri 5/17/24	Thu 5/23/24	10	All student

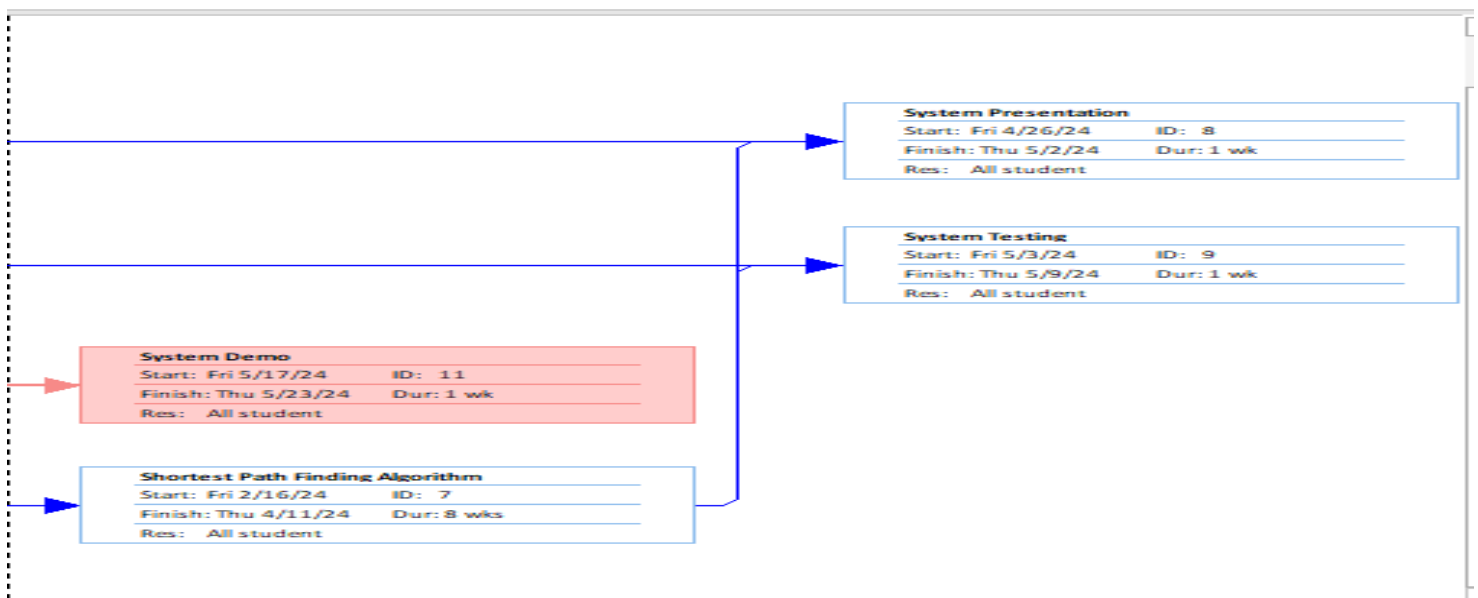
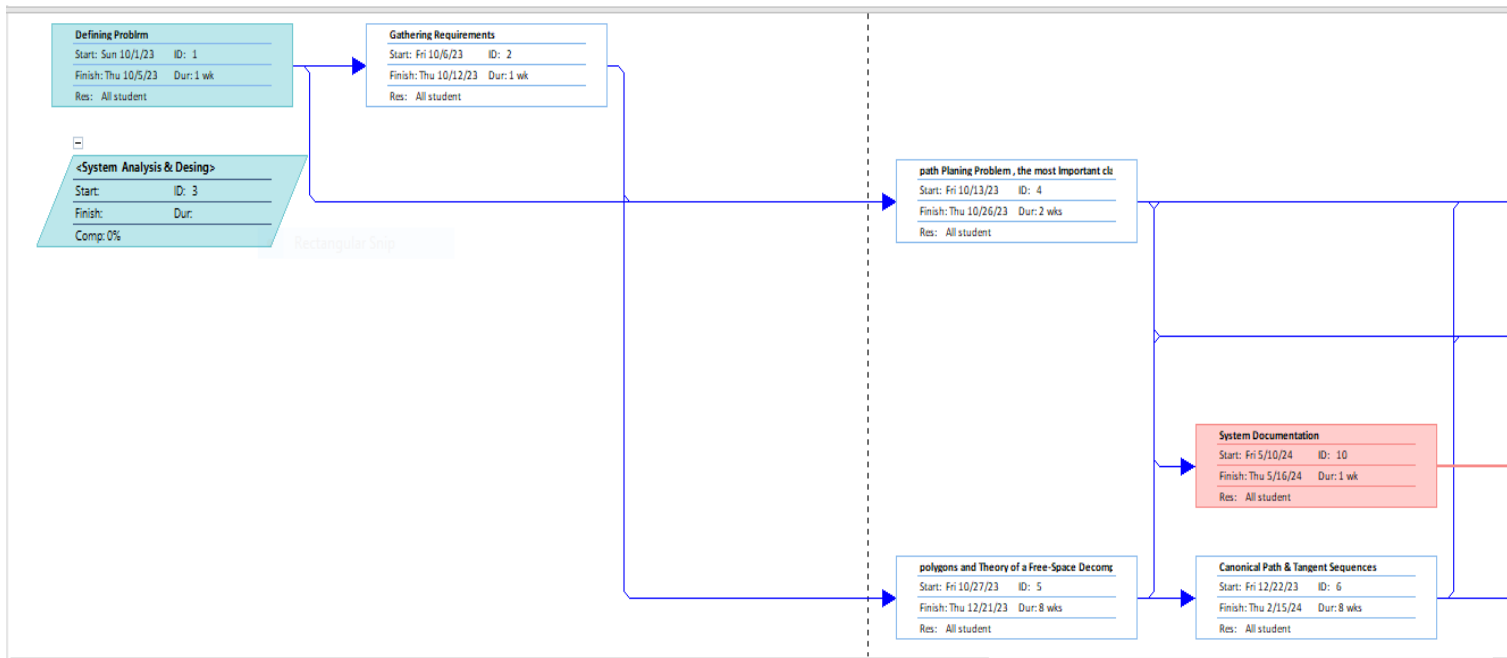
2.3.2 Gantt Chart

A Gantt chart commonly used in project management is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflect the start date, duration, and end date of the activity.



2.3.3. Network Diagrams

A Network Diagram is a visual representation of a project's schedule. Wellknown complements to network diagrams. A network diagram in project management is useful for planning and tracking the project from beginning to finish. It represents a project's critical path as well as the scope for the project



CHAPTER.3

POLYGONS

3.1 Polygons and their Orientation

Polygons, Vertices, Edges, and Navigation in a Simple Polygon

This text defines key concepts related to polygons and introduces functions for navigating between their vertices:

Polygons:

- A polygon (denoted by B) is a closed shape formed by connecting a sequence of n points (v_1 to v_n) in the plane, where $n \geq 3$.
- **Vertices (v_i):** The points that define the corners of the polygon.
- **Edges ($v_i v_{i+1}$):** Line segments connecting consecutive vertices.

Polygon Representation:

- $B = \{v_1, \dots, v_n\}$ (Equation II.1): A polygon is defined as a set of vertices in a specific order. The order matters because changing it can result in a different shape.

Types of Polygons (based on n):

- Triangle ($n = 3$)
- Quadrilateral ($n = 4$)
- n -gon (n sides)

Simple Polygons:

- A simple polygon is one whose path doesn't intersect itself. This means:
 - No consecutive edges are on the same line (i.e., any three consecutive vertices are not colinear).
 - No two edges intersect except at their common vertex (consecutive edges meet at a corner).
- Figure 2a shows simple quadrilaterals, while Figure 2b (not provided) is a non-simple one (likely with intersecting edges).

Next Vertex Function ($j(v_i)$):

- This function determines the next vertex in the sequence B after a given vertex v_i .
- $j(v_i) = v_{i+1}$ if $1 \leq i \leq n-1$ (i.e., for all vertices except the last one).
- $j(v_n) = v_1$ (the last vertex connects back to the first).
- In simpler terms, $j(v_i)$ gives you the vertex that comes after v_i when following the polygon's boundary in a counter-clockwise direction (assuming we follow a standard convention).
- Example: In Figure 2a (not provided), $j(v_1)$ is v_2 and $j(v_n)$ is v_1 .

Previous Vertex Function ($j^{-1}(v_i)$):

- This function (inverse of j) gives the vertex that comes before v_i in the sequence B .
- Proposition II.1 states that j is a bijection (one-to-one correspondence), so its inverse exists.
- The meaning of $j^{-1}(v_i)$ is the "previous vertex" of v_i .
- Example: In Figure 2a (not provided), $j^{-1}(v_1)$ is v_4 (assuming counter-clockwise traversal).

Interior Angles (β_i):

- The angle formed inside the polygon at vertex v_i .
- In any n -gon, the sum of all interior angles equals $180(n - 2)$ degrees.
- Example: A triangle's interior angles add up to 180 degrees.

Exterior Angles (δ_i):

- The angle formed outside the polygon at vertex v_i , formed by extending one edge and the next edge.
- Represented by δ_i (see Figure 2.1, not provided).

Direction ($\gamma(v_i, j(v_i))$):

- This represents the direction from vertex v_i to its next vertex $j(v_i)$.
- It likely refers to the angle or orientation in which you move from v_i to $j(v_i)$ while following the polygon's boundary.

This explanation clarifies the concepts of vertices, edges, simple polygons, and how to navigate between them using the j and j^{-1} functions. The introduction of angles and direction provides a foundation for further analysis of polygon properties and potential calculations involving rotations or movements within the polygon.

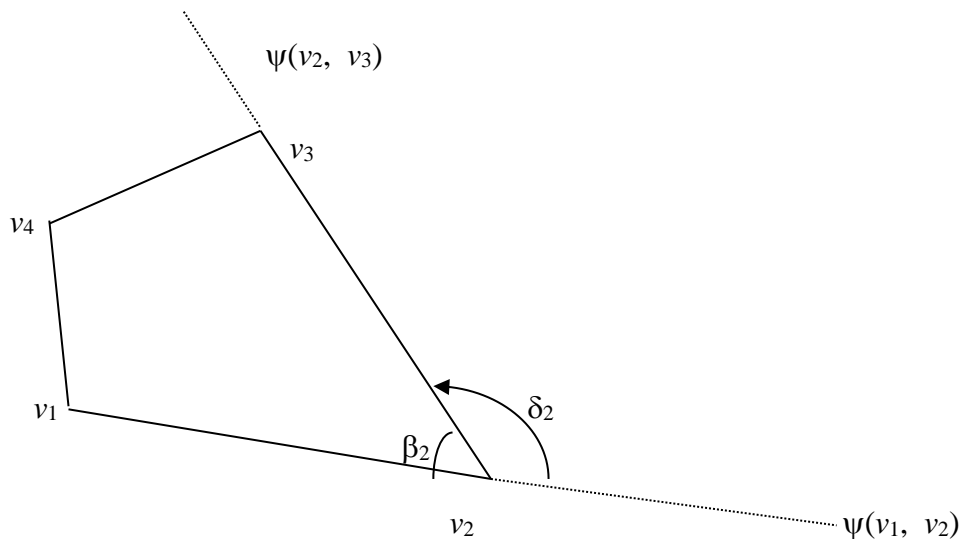


Figure 2.1 Interior and exterior angle of a simple polygon

Definition: Given two distinct points, $p_1 \equiv (x_1, y_1)$ and $p_2 \equiv (x_2, y_2)$. We define a direction function $\psi(p_1, p_2)$ as

$$\psi(p_1, p_2) \equiv \text{atan2}(y_2 - y_1, x_2 - x_1) \quad (\text{II.3})$$

The *exterior* angle, δ_i , at v_i is the angle between one side and the extension of the adjacent side related to v_i [11] (see Figure 3).

$$\delta_i = \Phi(\psi(v_i, \varphi(v_i)) - \psi(\varphi^{-1}(v_i), v_i)) \quad (\text{II.4})$$

Definition: A vertex v_i on a simple polygon is said to be a *convex* vertex if $\delta_i > 0$. If $\delta_i < 0$, a vertex v_i is said to be *concave* vertex.

For example, in Figure 4, in part (a), the vertex v_2 is convex because $\delta_2 > 0$. In part (b), the vertex v_3 is concave because $\delta_2 < 0$.

Definition: A simple polygon is a *convex* polygon if all of its vertices are convex (Figure 4(a)), otherwise it is *nonconvex* polygon (Figure 2.2(b)).

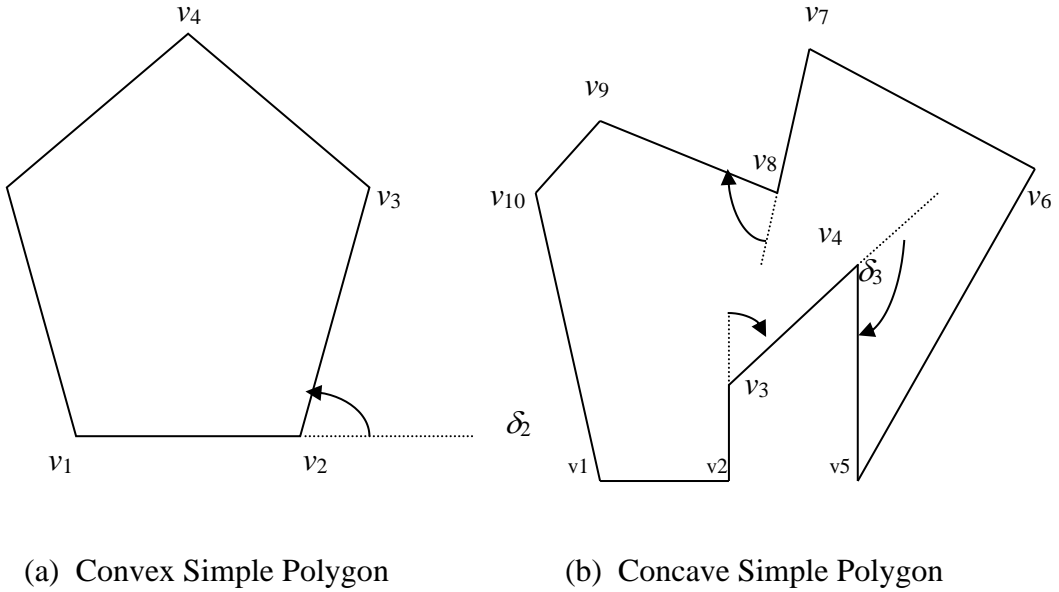


Figure 2.2 Convex and concave simple polygons

Now, we will define three important predicate, *ccw* (counterclockwise), *cw* (clockwise) and *col* (colinear). Consider vector $\vec{u} = (x_1, y_1)^T$ and $\vec{v} = (x_2, y_2)^T$, shown in Figure 2.3 (a). The *cross product* $\vec{u} \times \vec{v}$ can be interpreted as the signed area of the parallelogram formed by the points $(0, 0)$, u , v , and $u + v = (x_1 + x_2, y_1 + y_2)$. An equivalent, but more useful, definition gives the cross product as the determinant of a matrix.

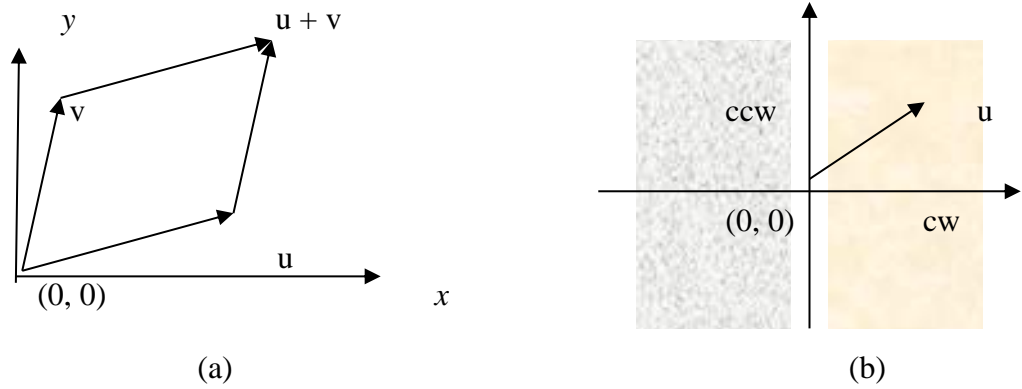


Figure 2.3 Cross product of vectors

$$\begin{aligned}\vec{u} \times \vec{v} &= \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} \\ &= x_1 y_2 - x_2 y_1 \\ &= -\vec{v} \times \vec{u} \quad (\text{II.5})\end{aligned}$$

If $\vec{u} \times \vec{v}$ is positive, then \vec{u} is clockwise from \vec{v} with respect to the origin $(0, 0)$; if this cross product is negative, then \vec{u} is counterclockwise from \vec{v} . Figure 5(b) shows the clockwise and counterclockwise regions relative to a vector \vec{u} . A boundary condition arises if the cross product is zero; in this case, the vectors are collinear, pointing in either the same or opposite directions.

The above discussion is very useful for all results related to the area of the polygon.

The *area* of a polygon whose vertices v_i have coordinates (x_i, y_i) , for $1 \leq i \leq n$, is the “signed” value of

$$\begin{aligned} \text{Area}(B) &= \frac{1}{2} (x_1 y_2 - x_2 y_1) + \dots + \frac{1}{2} (x_{n-1} y_n - x_n y_{n-1}) + \frac{1}{2} (x_n y_1 - x_1 y_n) \\ &= \frac{1}{2} \sum_{i=1}^n (x_i y_{i+1} - x_{i+1} y_i) \end{aligned}$$

In particular, for a triangle $B = \{v_1, v_2, v_3\} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$, the “signed” area is defined as

$$\begin{aligned} \Delta &= \frac{1}{2} \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix} \quad (\text{II.6}) \\ &= \frac{1}{2} ((x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)) \end{aligned}$$

Proposition III.1 *For any triangle B ,*

- (I) *If $\Delta > 0$, B is ccw and area of B is equal to Δ .*
- (II) *If $\Delta < 0$, B is cw and area of B is equal to $|\Delta|$.*
- (III) *If $\Delta = 0$, B is col and area of B is $= 0$.*

Definition: A convex polygon is a polygon whose ordered list of vertices produces a counterclockwise (ccw) boundary loop. A nonconvex polygon is a polygon whose ordered list of vertices produces a clockwise (cw) directed boundary loop.

Let us show how the orientation concept is useful in solving geometrical and robotics problems by giving an example. Any two distinct points in a plane define a *segment*. A *closed segment* includes the endpoints, but an *open segment* does not. Two segments are said to *intersect* if they share only one point (thus, if they are sharing a part or all of the segments, we do not claim

they are intersecting). The problem of testing whether two (open or closed) segments are intersecting is answered by the orientation function (Figure 2.4)

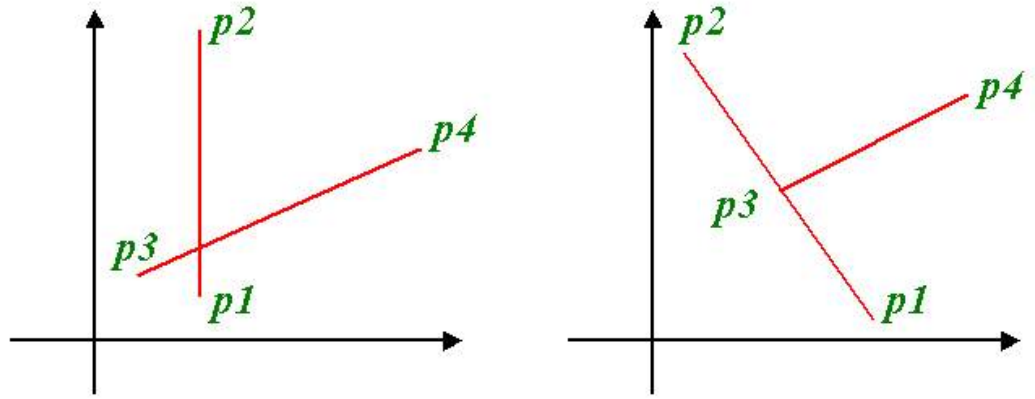


Figure 2.4 Intersecting Segments

Proposition II.2 A necessary and sufficient condition for two open segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$, to intersect each other is

$$[O(p_1, p_2, p_3) = -O(p_1, p_2, p_4) \neq 0] \wedge [O(p_3, p_4, p_1) = -O(p_3, p_4, p_2) \neq 0] \quad (\text{II.7})$$

Proposition II.3 A necessary and sufficient condition for two closed segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$, to intersect each other is

$$[O(p_1, p_2, p_3) \neq O(p_1, p_2, p_4)] \wedge [O(p_3, p_4, p_1) \neq O(p_3, p_4, p_2)] \quad (\text{II.8})$$

The computational costs of these tests are constant and inexpensive. If two open segments intersect, so do the closed segments. However, its converse is not true.

3.2 Polygonal World

Simple Polygon and Plane Partitioning:

- A simple polygon, as defined earlier, doesn't have any self-intersections and separates the plane into two distinct areas.
- This separation is based on the Jordan Curve Theorem (not explicitly mentioned here but implied).

Regions of a Simple Polygon (Figure 2.5):

- **Interior (int(B)):** The set of points **enclosed** by the polygon. This is the finite area "inside" the polygon's boundary.
- **Boundary (B):** The set of points that form the actual **edges** of the polygon.
- **Exterior (free(B)):** The set of points surrounding the polygon. This is the infinite area "outside" the polygon.

Classification of Simple Polygons (ccw vs. cw):

- The text introduces a classification system for simple polygons based on how their "free side" (exterior) is defined:
 - **Counter-clockwise (ccw) polygon:**
 - The exterior (free(B)) of the polygon is the area to the **right** of the boundary when following the polygon's vertices in a counter-clockwise direction.
 - **Clockwise (cw) polygon:**
 - The exterior (free(B)) of the polygon is the area to the **left** of the boundary when following the polygon's vertices in a clockwise direction.

Key Points:

- This classification system is crucial for various geometric calculations and algorithms that interact with polygons. It helps determine which side of the polygon is considered "outside" based on the direction you traverse its boundary.

Additional Notes:

- The text doesn't explicitly mention, but the "left" and "right" definitions for exterior might be based on a specific convention where the polygon is drawn on a plane with a positive x-axis to the right and a positive y-axis upwards.
- In some contexts, the "interior" might also be referred to as the "hole" of the polygon.

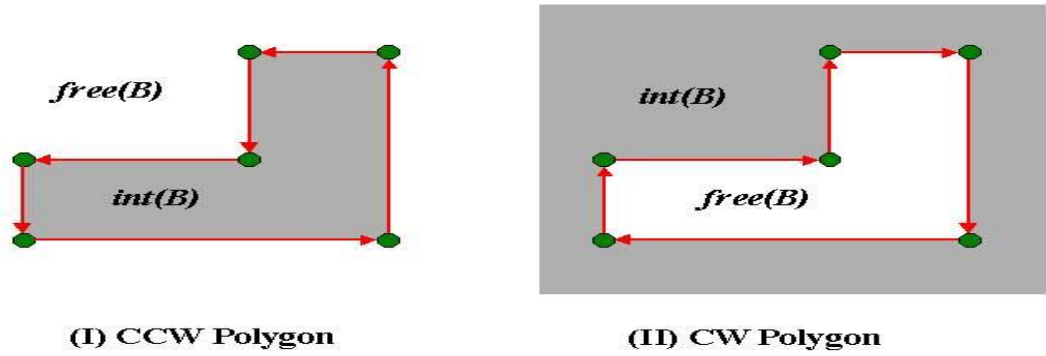


Figure 2.5 Interior and exterior of a simple polygon

2.3 Polygonal World and Path Class

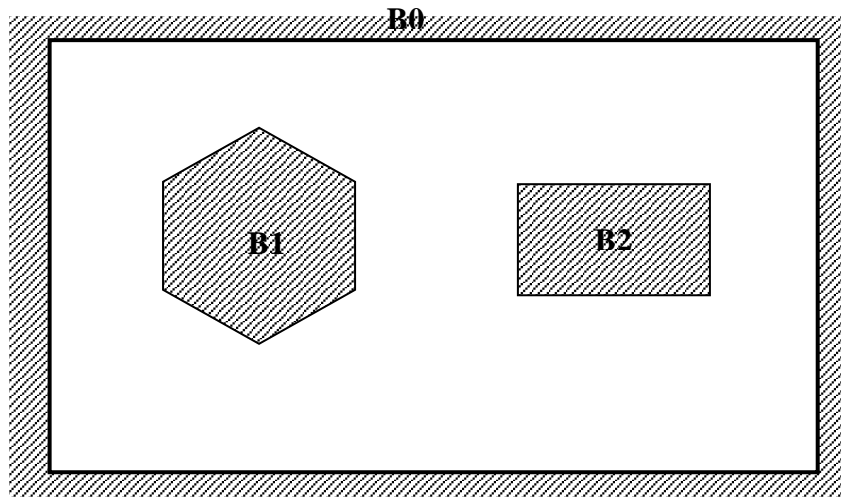


Figure 2.6: Polygonal world

Consider a world W which consists of a finite number of polygons, i.e.,

$$W = \{B_0, B_1, \dots, B_n\}, n > 0$$

Where B_0 is a *cw* polygon, and *ccw* polygons B_1, \dots, B_n are considered to be obstacles for the robot (see Figure 2.6).

For a point $p \in \text{free}(W)$, the distance $d(p, B_i)$ from p to a polygon B_i is defined in Eq5.2. The Voronoi region $V(B_i)$ of a polygon B_i in W is defined as

$$V(B_i) = \{p \in \text{free}(W) \mid (\text{for all } j) [(i \neq j \wedge 1 \leq j \leq n) \rightarrow [d(p, B_i) < d(p, B_j)]]\} \quad (4.1)$$

For instance, Eq.4.1 means that any point within $\text{free}(W)$ has its image on the two polygons. The Voronoi diagram of world W consisting of three polygons is shown in (figure 4.7), and the Voronoi boundaries of W consists of line segments and parabolic arcs. Note that the intersection point of three or more Voronoi boundary segments is called a *v-node*. A Voronoi boundary segment(s) between two *v-nodes* is called a *v-edge*. For example, there are two *v-nodes* and three *v-edges* as shown in Figure 2.7.

Each undirected *v-edge* \mathfrak{f} is the boundary of two Voronoi regions, $V(B_i)$ and $V(B_j)$. We denote an undirected *v-edge* \mathfrak{f} by $\mathfrak{f} = [B_i : B_j]$,

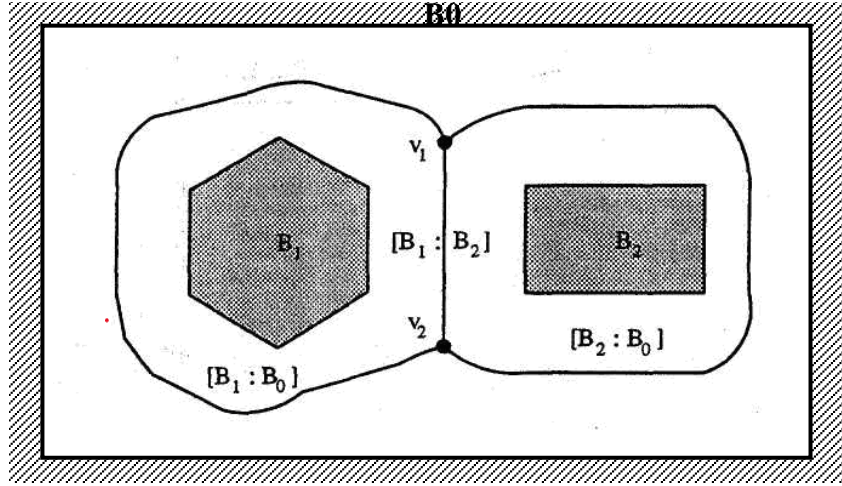


Figure 2.7 : Generalized Voronoi diagram of polygonal world (1)

Where $[B_i : B_j]$ and $[B_j : B_i]$ are considered the same. For example, in Figure 2.7, the undirected *v-edge* between the two *v-nodes* v_1 and v_2 is $\mathfrak{f} = [B_1 : B_2]$ or $\mathfrak{f} = [B_2 : B_1]$. Also, there are three undirected *v-edges* $[B_1 : B_0]$, $[B_1 : B_2]$, and $[B_2 : B_0]$. Another example is shown in (figure 2.8). In which, a world W consists of five polygons B_0 , B_1 , B_2 , B_3 and B_4 . There are five *v-nodes* and eight undirected *v-edges* $[B_1 : B_0]$, $[B_1 : B_2]$, $[B_2 : B_0]$, $[B_2 : B_3]$, $[B_1 : B_4]$, $[B_4 : B_0]$, $[B_3 : B_4]$, and $[B_3 : B_0]$.

3.3.1 Directed v-edge

Each undirected v-edge is the boundary of two Voronoi regions, $V(B_i)$ and $V(B_j)$. In this case,

$$[B_i : B_j] \neq [B_j : B_i].$$

Now, we consider the *directed* v-edge. Once the directed v-edge is given, the concepts of left and right images take on meaning. This will aid in using the world data to capture the spatial relationship between the objects in the world. We have two types of directed boundaries:

- Directed boundaries of two polygons that are the same (*ccw*):

There are two opposite directions on an undirected v-edge $[B_i : B_j]$. One direction goes *ccw* with B_i and *cw* with B_j . The other direction goes *cw* with B_i and *ccw* with B_j (Figure 2.9).

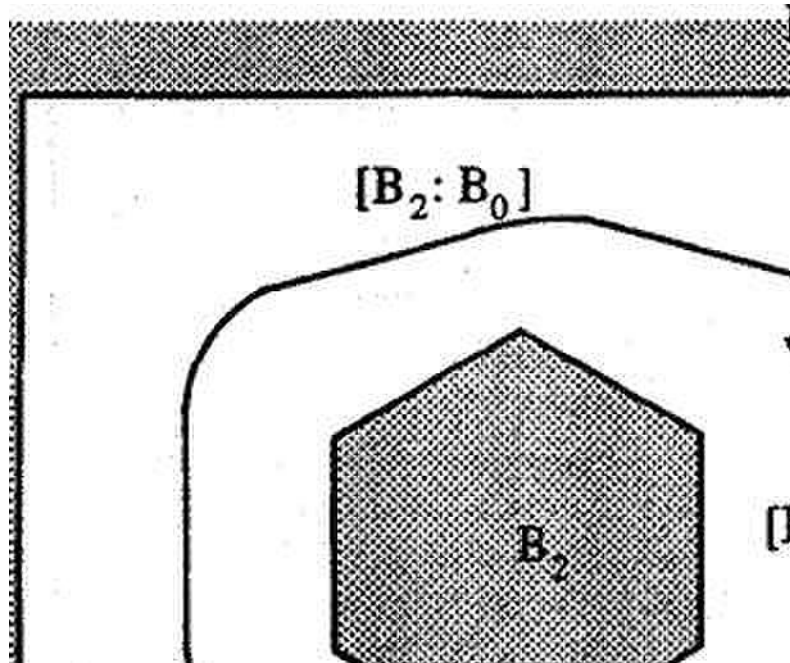


Figure 2.8: Generalized Voronoi diagram of polygonal world (2)

- Directed boundaries of two polygons that are different (*ccw* and *cw*):

There are two opposite directions on an undirected v-edge $[Bi : Bj]$. One direction goes *ccw* with Bi and *cw* with Bj . The other direction goes *cw* with Bi and *ccw* with Bj (Figure 2.10).

Now, we denote *directed* v-edge ξ by

$$\xi = [B_i / B_j],$$

where Bi and Bj refer to the left and right polygons respectively. It is clear that $[B_i/B_j]$ and $[B_j/B_i]$ are not the same.

NOTE : From the above we see that we don't have to be exhausted by the costly calculation of the Voronoi Diagram for our polygonal world, but it is enough to represent the $\text{free}(W)$ in the form of v-edges (right and left obstacles) and v-nodes.

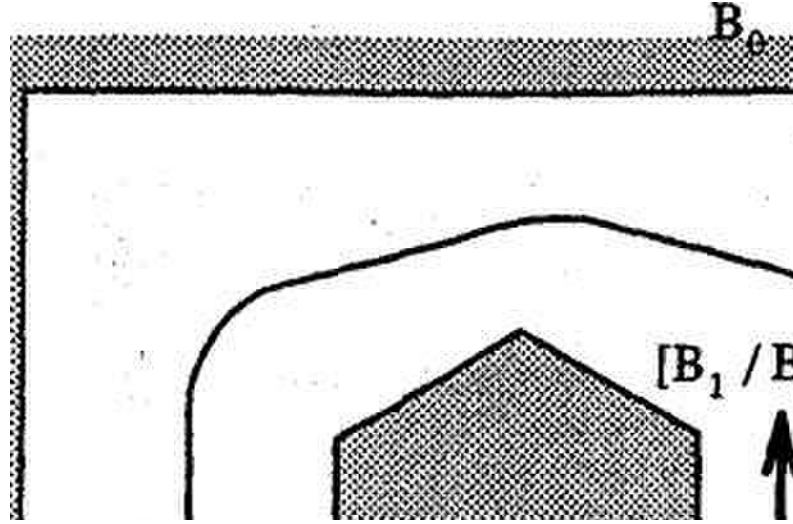


Figure 2.9 : Defining directed v-edge for the same directed boundaries
(ccw polygons)

Although the assignment of left and right is arbitrary, it is fixed for all times once set. For consistency in this work, left and right polygons will be the first and second terms in directed v-edges, respectively. The following is the result of the previous discussion of directed v-edge.

Lemma 4.1 *In a polygonal world W , where W is encircled by an outermost *cw* polygonal boundary and has n ($n \geq 1$) *ccw* polygonal obstacles inside the boundary, a directed v-edge always consists of two different polygons.*

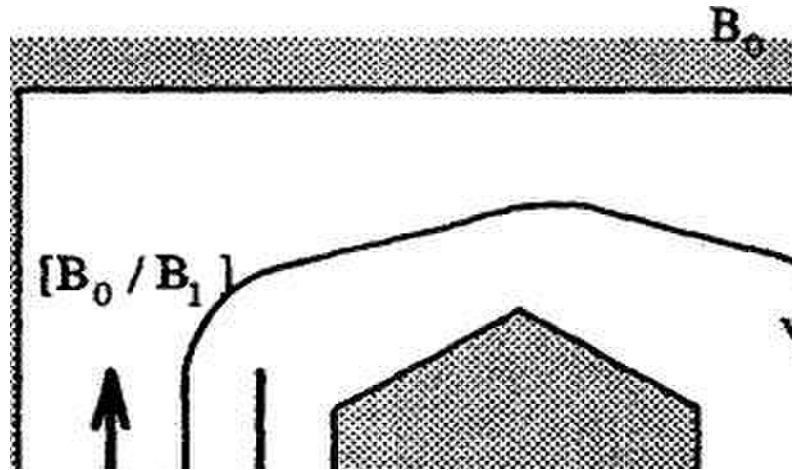


Figure 2.10: Defining directed v -edge for different directed boundaries
(cw and ccw)

3.3.2 Canonical Paths and Directed v -edges Sequences

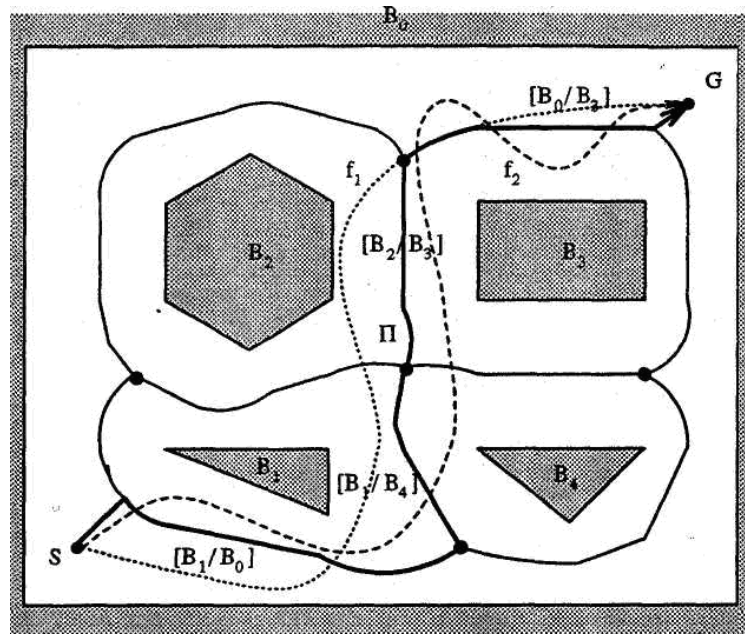


Figure 2.11: Paths and canonical paths

A robot can work only in the free space, $\text{free}(W)$. A *path* f in a world W is a continuous function

$$F : [0,1] \rightarrow \text{free}(W) \quad (4.2)$$

Consider the problem of finding a path from a start configuration, S , to a goal configuration, G in a polygonal world W (Figure 2.11), where *ccw* polygons B_1, B_2, B_3 and B_4 are considered as obstacles for robot in this world and a world has one *cw* polygon B_0 . It is desired to connect the start configuration, S , to the goal configuration, G , using a continuous, smooth path. There are infinitely many distinct paths connecting S and G . However, actually, we need to compare only paths which satisfy a special property.

Definition: A path Π is called a *canonical path* if there exists a sequence of directed v -edges such that : $\Pi = s_s \xi_1 \dots \xi_k s_g \quad k \geq 1$ (4.3)

Where

- the right hand side of Eq. 4.3 is the concatenation of $k + 2$ sub paths,
- the sub path s_s is the shortest path from S to ξ_1 ,
- $\xi_1 \dots \xi_k$ is the sequence of directed v -edges, and
- the sub path s_g is the shortest path from ξ_k to G .

For example, in figure 2.12, $\Pi = s_s[B_1/B_0][B_1/B_4][B_2/B_3][B_0/B_3]s_g$.

The following is the result of the previous discussion of the canonical path.

Lemma 4.2 ; For a given W, S , and G , a canonical path Π is the only one among all the paths in a homotopy class which satisfies the following conditions:

- the sub path connecting S to first directed v -edge is the shortest one,
- sequential pieces from one directed v -edge to the next, and
- the sub path connecting the last directed v -edge to G is the shortest one.

Proposition 4.1: For a given W, S , and G , for paths f_1 and f_2 in a homotopy class, if $f_1 \rightarrow \Pi_1$ and $f_2 \rightarrow \Pi_2$ then $\Pi_1 \equiv \Pi_2$.

Proof. Assume that the hypothesis is true. Since f_1 and Π_1 are homotopic, there is a continuous function H which transforms f_1 into Π_1 . Also, there is a continuous function H which transforms f_2 into Π_2 . By Lemma 4.2, there is only one canonical path Π among all paths in a homotopy class. It follows that $\Pi_1 \equiv \Pi_2$.

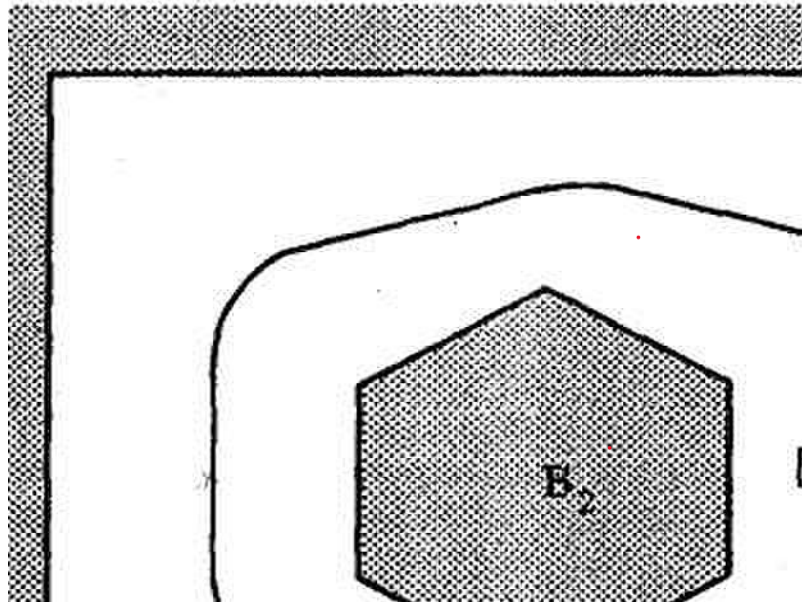


Figure 2.12: Interpretation of canonical path as directed v-edges sequence

Definition: A *directed v-edges sequence* γ is a finite sequence of directed v-edges such that no subsequence of $[B_i/B_j] [B_j/B_i]$ is a part of it.

By definition, if Π is a canonical path, then $\Pi = s_s \gamma s_g$ (Figure 2.12), where

γ is $\xi_1 \dots \xi_k$.

Several examples of directed v-edges sequences are illustrated in Figures 2.13 and 2.14. For example, the directed v-edges sequences for the above figures are as follows:

$$\gamma_1 = [B_1/B_0][B_1/B_4][B_2/B_3][B_0/B_3] \quad (\text{Figure 2.13})$$

$$\gamma_2 = [B_1/B_0][B_4/B_0][B_3/B_0] \quad (\text{Figure 2.14})$$

Proposition 4.2: In a homotopy class, for all paths f_1 and f_2 , $\gamma_1 = \gamma_2$ is the necessary and sufficient condition to make $f_1 = f_2$.

Proof.

First prove the sufficiency. Assume $\gamma_1 = \gamma_2$. If $\gamma_1 = \gamma_2$, each path has a sequence of the same directed v-edges. Furthermore, in a homotopy class, both paths have the same left and right polygons. Each path is a concatenation of pieces. These pieces connect

the start configuration to the first directed v-edge in γ . the sequential pieces from one directed v-edge to the next, and the last directed v-edge to the goal configuration. We can easily construct H to transform f_1 into f_2 piece by piece without running over any obstacles. The transformation, H , is the composition of the sequences of the transformations shown. Hence, the paths are homotopic.

To prove the necessity, assume $f_1 \approx f_2$. We are given a path f_1 . Consider a directed v-edges sequence γ_1 of f_1 . Since f_1 and f_2 are homotopic, there is a continuous function H which transforms f_1 into f_2 . Since $H(s, t)$ is a continuous function, each directed v-edge ξ , which has left and right polygons, continuously concatenates with the next ξ over s as t moves when transforming f_1 into f_2 . However, there is no way in which f_2 can eliminate, insert or repeat any ξ other than in the monotopic sequence of f_1 . $H(s, t)$ can neither destroy existing nor create any new ξ , because $H(s, t) \in \text{free}(W)$ and $H(s, t)$ is continuous. Therefore $\gamma_1 = \gamma_2$.

From above, we can conclude that:

- A directed v-edges sequence γ is unique for paths which are not homotopic.
- A directed v-edges sequence γ is a symbolic representation.

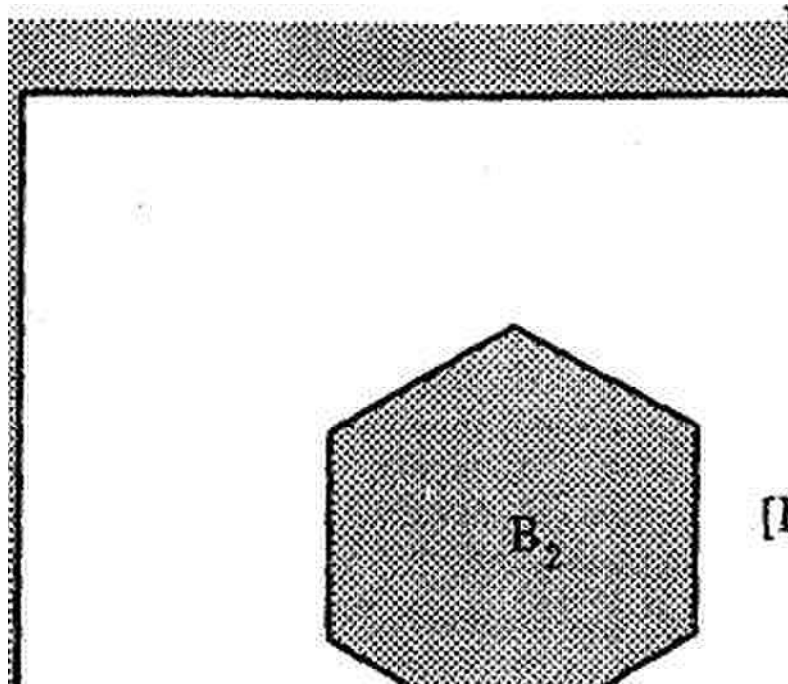


Figure 2.13 : Directed v-edges sequence $[\gamma_1]$

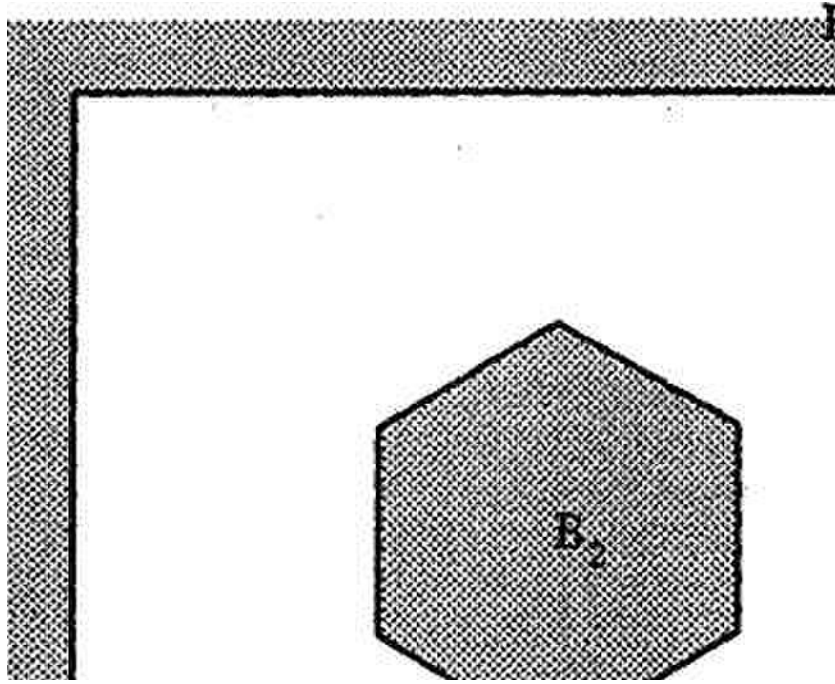


Figure 2.14 : Directed v-edges sequence $[\gamma_2]$

3.3.3 Connectivity Graph

We make the following observations about the world in Figure 2.7. Three Voronoi boundary segments intersect in one node (v-node). There is one line segment between two v-nodes (v-edge). Each v-node operates in both directions, and no v-node has a v-edge to itself.

Definition: A basic connectivity graph $G = (V, E)$ consists of V , a non-empty set of v-nodes, and E , a set of unordered pairs of distinct elements of V called undirected v-edges. Consequently this figure can be modeled using a basic connectivity graph, consisting of vertices which represent v-nodes, and undirected edges, which represent undirected v-edges, where each edge connects two distinct vertices.

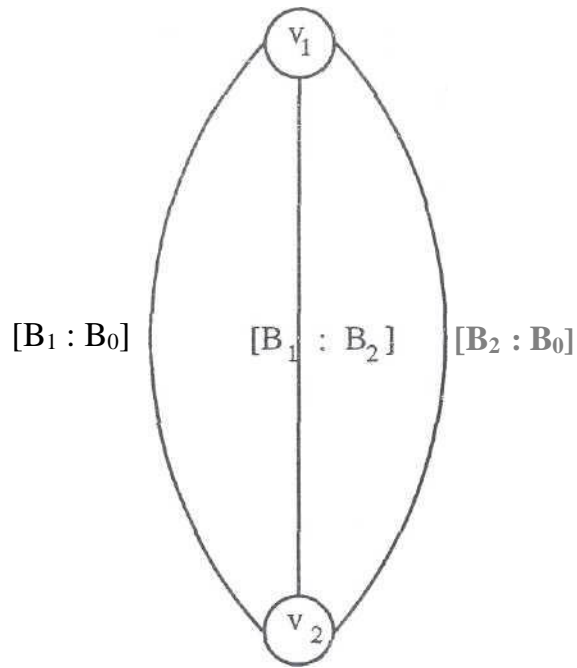


Figure 2.15 Basic connectivity graph of a polygonal world (1)

The basic connectivity graphs generated by the world in figures 2.7 and 2.8 are shown in figures 2.15 and 2.16.

Now we will explain how to represent a path class (see subsection 3.4).

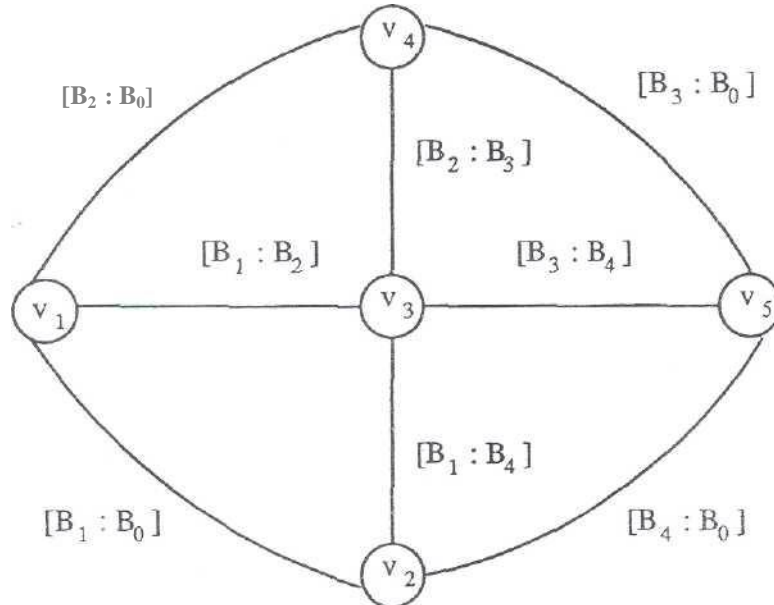


Figure 2.16 : Basic connectivity graph of a polygonal world (2)

3.3.4 Path Class Representation

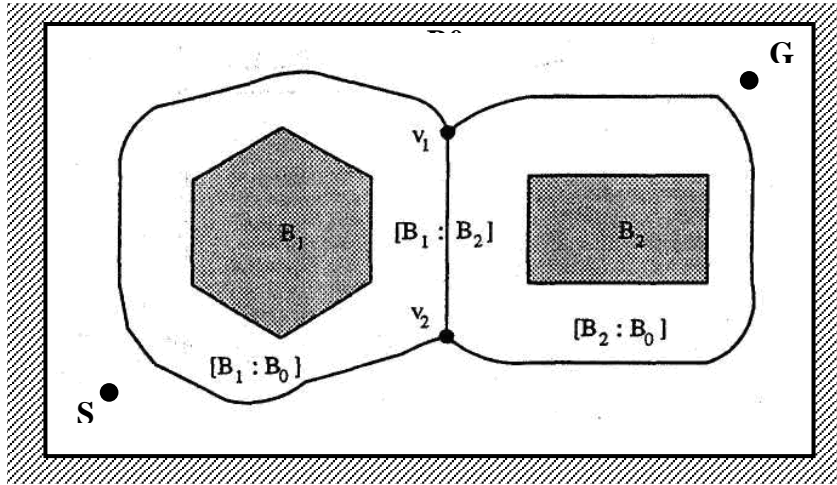


Figure 2.17 : Polygonal world (1)

Consider the problem of finding a path from a start configuration S , to a goal configuration G , in a polygonal world W (Figure 2.17). It is desired to connect the start configuration S , to the goal configuration G , using a continuous, smooth path. In Figure 2.17, there are four different path classes. Consider the problem of how to symbolically represent each path class. A method based on directed v -edges is presented. Given start and goal configurations, we add two new nodes, S and G , to the basic connectivity graph to obtain an augmented connectivity graph. The augmented connectivity graph generated by the world in Figure 2.17 is shown in Figure 2.18. In which, there are four different path classes. In its most general form, a path classes, Π , is symbolically represented by a sequence of directed v -edges. For instance, four different path classes in Figure 2.18 are represented by:

$$\pi_1 = [B_0/B_1] [B_0/B_2]$$

$$\pi_2 = [B_0/B_1] [B_2/B_1] [B_2/B_0]$$

$$\pi_3 = [B_1/B_0] [B_1/B_2] [B_0/B_2]$$

$$\pi_4 = [B_1/B_0] [B_2/B_0]$$

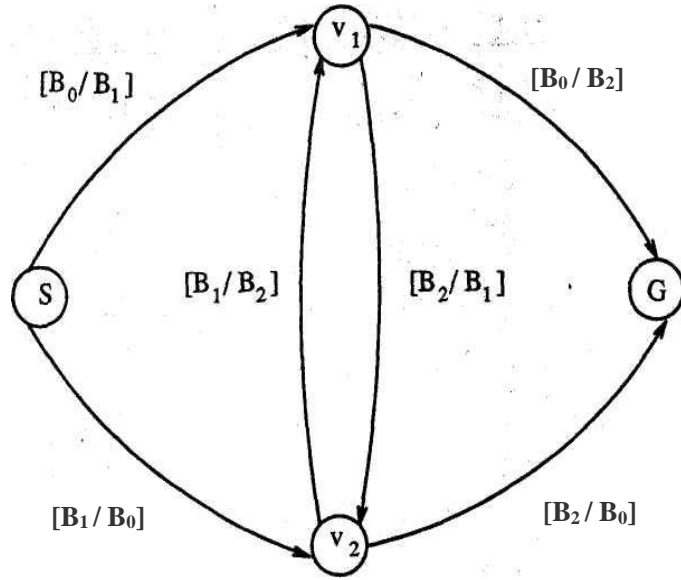


Figure 2.18 : Augmented connectivity graph of a polygonal world

Another example is shown in Figure 2.16. The augmented connectivity graph generated by the world in Figure 2.19 is shown in Figure 2.20. In which, there are twelve different path classes which connect S with G :

- $\pi1 = [B0/B1] [B0/B2] [B0/B3]$
- $\pi2 = [B0/B1] [B0/B2] [B3/B2] [B3/B4] [B3/B0]$
- $\pi3 = [B0/B1] [B0/B2] [B3/B2] [B4/B1] [B4/B0] [B3/B0]$
- $\pi4 = [B0/B1] [B2/B1] [B2/B3] [B0/B3]$
- $\pi5 = [B0/B1] [B2/B1] [B3/B4] [B3/B0]$
- $\pi6 = [B0/B1] [B2/B1] [B4/B1] [B4/B0] [B3/B0]$
- $\pi7 = [B1/B0] [B1/B4] [B1/B2] [B0/B2] [B0/B3]$
- $\pi8 = [B1/B0] [B1/B4] [B2/B3] [B0/B3]$
- $\pi9 = [B1/B0] [B1/B4] [B3/B4] [B3/B0]$
- $\pi10 = [B1/B0] [B4/B0] [B3/B0]$
- $\pi11 = [B1/B0] [B4/B0] [B4/B3] [B2/B3] [B0/B3]$
- $\pi12 = [B1/B0] [B4/B0] [B4/B3] [B1/B2] [B0/B2] [B0/B3]$

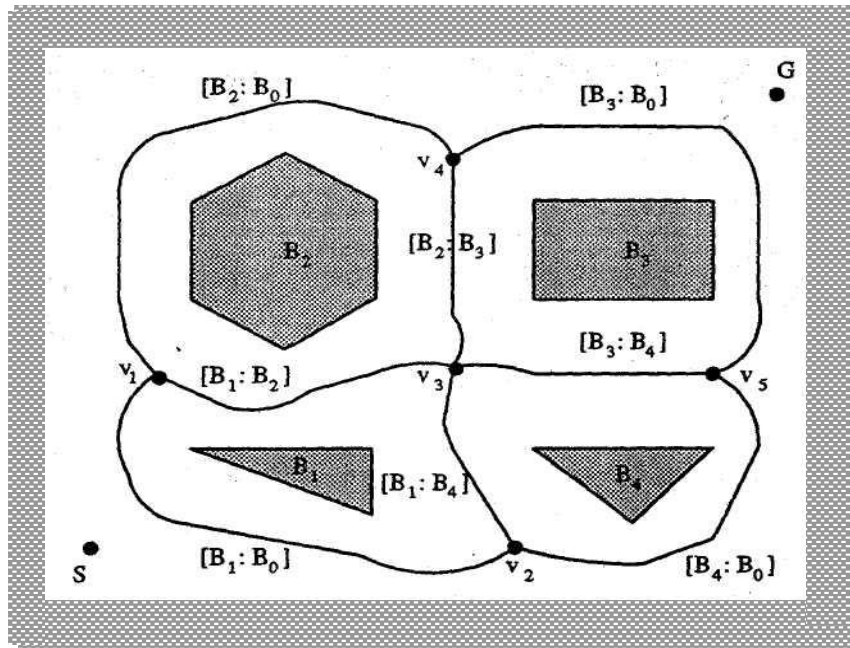


Figure 2.19: Polygonal world (II)

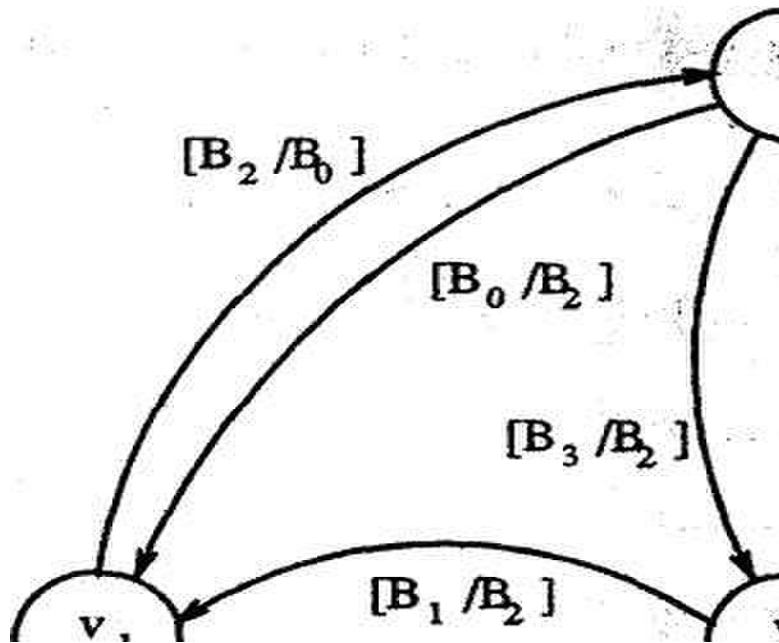


Figure 2.20 : Augmented connectivity graph of a polygonal world (II)

3.4 Finding the Best Path Class

In this section we outline how to find the best path class. Finding the best path class from start configuration S , to goal configuration G , in the world is transformed into the minimum cost path finding problem from S to G in the augmented connectivity graph. The augmented connectivity graph uses a weighted edge whose value depends upon the mission-based cost function associated with the v-edge. For instance, a cost for the edge is defined as the energy (or time) for the vehicle to make a motion from one v-node V_i to another v-node V_j . This cost not only reflects the distance, but the turns needed to make the motion. It may also reflect the safety (i.e., if the region is narrow, the cost is high). Then by applying any of the graph searching techniques (discussed in appendix B), e.g. Dijkstra's algorithm can be perfectly applied to this global motion planning problem. The best path class in terms of a sequence of directed v-edges is hence obtained. The computation time is

$O((n + m)\log n)$ using a priority queue, where n (is v-nodes (vertices) in a graph) and m (is the number of v-edge in the graph) in the augmented connected graph respectively.

Once the path class is found, it is passed to the next stage of the Motion planning process, Local Motion Planning which ensures that the vehicle will follow the path class to reach the goal. The choice of the mission type ultimately affects which steering function (which is out of our scope in this work) is used to guide the vehicle. Simulation results for the proposed approach are given in chapter 6.

2.5 Image of Point on Convex Polygon

We assume a global two-dimensional Cartesian coordinate system in a plane E^2 . Given two distinct points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ in E^2 , The Euclidean distance $d(p_1, p_2)$ between them is defined as:

$$d(p_1, p_2) \equiv \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (5.1)$$

Assume that there is an object O in a plane. An object might be a point, a line, an open line segment, a polygon, or other set of points. The shortest distance $d(p, o)$ between a point p and an object O is defined as follows:

$$d(p, o) \equiv \min_{p_1 \in o} d(p, p_1) \quad (5.2)$$

Eq. 2.22 generalizes the function d defined by Eq. 2.21

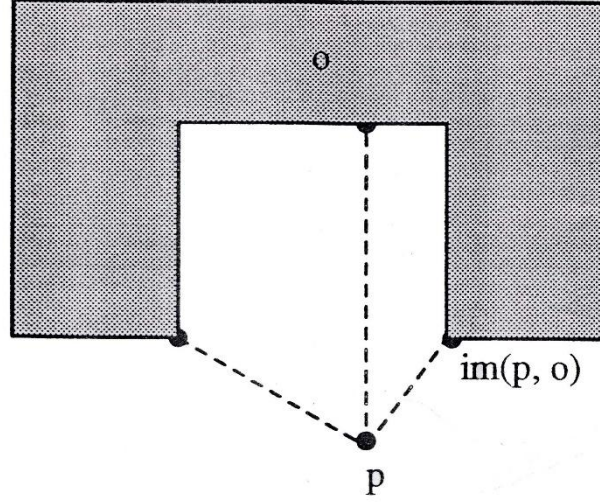


Figure 2.21 Image on object

Definition: A point p_1 in O which satisfies $d(p, p_1) = d(p, o)$ is said to be an *image* of p on o and is denoted by $im(p, o)$ (Figure 2.23).

If a world W has more than one object, an image $im(p, W)$ is defined as the image $im(p, o_i)$ such that $d(p, o_i)$ is the minimum over all objects in W (figure 2.24).

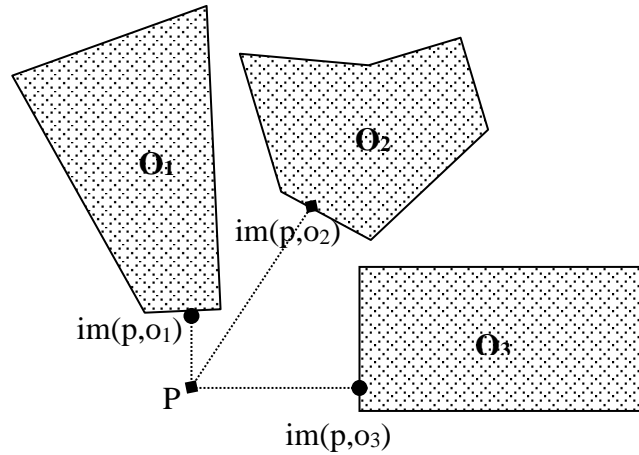


Figure 2.22: Images on world

2.5.1 Visibility from Point to Polygon

Assume that we are given a convex polygon $B = (v_1, \dots, v_n)$ and a point $p \in \text{free}(B)$. The significant notion for our purpose is the following classification of each vertex v_i of B with respect to the segment $\overline{pv_i}$. Each vertex of B is said to be *visible*, *invisible*, *cw-tangential*, or *ccw-tangential* (Figure 2.23).

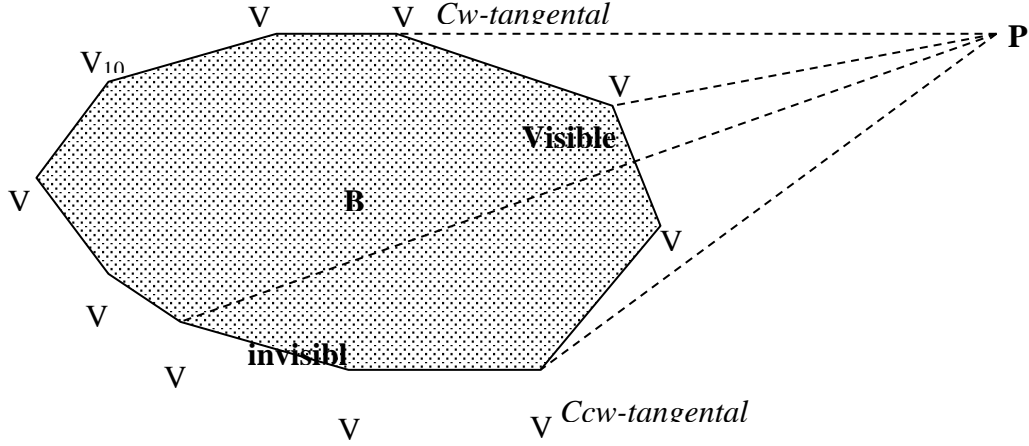


Figure 2.23 : Visibility from point p to convex polygon B .

Definition: Let B be a convex polygon, and let a point $p \in \text{free}(B)$.

1. A vertex v_i is *tangential* from point p if the two vertices adjacent to v_i lie on the same side of the line containing $\overline{pv_i}$.
2. A vertex v_i is *visible* if the segment $\overline{pv_i}$ does not intersect the interior of B and the two vertices adjacent to v_i lie on opposite sides of the line containing $\overline{pv_i}$.
3. A vertex v_i is *invisible* if the segment $\overline{pv_i}$ intersects the interior of B .

Let $\text{cw-tangential}(p, v_i, B)$ denote that vertex v_i of B is clockwise tangential with respect to the segment $\overline{pv_i}$, $\text{ccw-tangential}(p, v_i, B)$ denote that vertex v_i of B is counterclockwise tangential with respect to the segment $\overline{pv_i}$, $\text{visible}(p, v_i, B)$ denote that vertex v_i of B is visible with respect to the segment $\overline{pv_i}$, and $\text{invisible}(p, v_i, B)$ denote that vertex v_i of B is invisible with respect to the segment $\overline{pv_i}$. It is now easy to establish the following lemma.

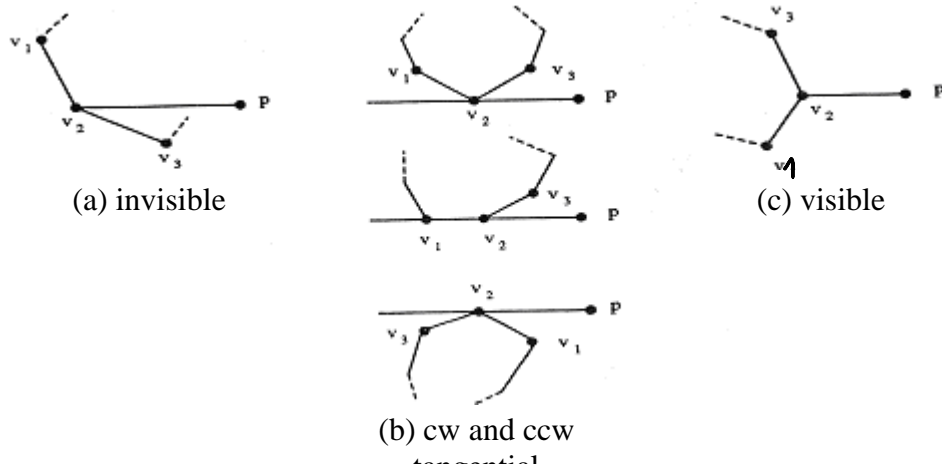


Figure 2.24: Classifications of vertex v_i of polygon B with respect to a segment $\overline{pv_i}$

Lemma 5.2 Given a convex polygon B and a point $p \in \text{free}(B)$, the vertex v_i is one of the following four types:

$$\text{cw-tangential}(p, v_i, B) \equiv \sim \text{ccw}(p, v_i, \Phi^{-1}(v_i)) \ \& \ \sim \text{ccw}(p, v_i, \Phi(v_i)) \quad (5.3)$$

$$\text{ccw-tangential}(p, v_i, B) \equiv \sim \text{ccw}(p, v_i, \Phi^{-1}(v_i)) \ \& \ \sim \text{cw}(p, v_i, \Phi(v_i)) \quad (5.4)$$

$$\text{visible}(p, v_i, B) \equiv \text{ccw}(p, v_i, \Phi^{-1}(v_i)) \ \& \ \text{cw}(p, v_i, \Phi(v_i)) \quad (5.5)$$

$$\text{invisible}(p, v_i, B) \equiv \text{cw}(p, v_i, \Phi^{-1}(v_i)) \ \& \ \text{ccw}(p, v_i, \Phi(v_i)) \quad (5.6)$$

Proof.

For the first part (Eq.5.3), v_i is cw-tangential if the two vertices adjacent to v_i , $\Phi^{-1}(v_i)$ and $\Phi(v_i)$ lie on the same side of the line containing $\overline{pv_i}$, and we have the following three cases.

1. **case 1:** $cw(p, vi, \Phi^{-1}(vi))$ & $cw(p, vi, \Phi(vi))$

If \overline{pvi} and $vi \overline{\Phi^{-1}(vi)}$ make a right turn at vi , $\overline{p\Phi^{-1}(vi)}$ is clockwise from \overline{pvi} , And \overline{pvi} and $vi \overline{\Phi(vi)}$ make a right turn at vi , $\overline{p\Phi(vi)}$ is clockwise from \overline{pvi} then vi is *cw-tangential*.

2. **case 2:** $col(p, vi, \Phi^{-1}(vi))$ & $cw(p, vi, \Phi(vi))$

If p, vi and $\Phi^{-1}(vi)$ are collinear and \overline{pvi} and $vi \overline{\Phi(vi)}$ make a right turn at vi , $\overline{p\Phi(vi)}$ is clockwise from \overline{pvi} , then vi is *cw-tangential*.

3. **case 3:** $(p, vi, \Phi^{-1}(vi))$ & $cw(p, vi, \Phi(vi))$

If \overline{pvi} and $vi \overline{\Phi^{-1}(vi)}$ make a right turn at vi , $\overline{p\Phi^{-1}(vi)}$ is clockwise from \overline{pvi} , And \overline{pvi} and $\Phi(vi)$ are collinear, then vi is *cw-tangential*.

This gives a proof of (Eq.4.3), in other word vi , is *cw-tangential* from p (figures 2.25, 2.26)

The second part (Eq.5.4) is proven similarly. (Eq.4.3 = $\Pi = s_s \xi_1 \dots \xi_k s_g$ $k \geq 1$)

For the third part (Eq.5.5), since the two vertices adjacent to vi lie on the opposite side of the line containing \overline{pvi} , and \overline{pvi} doesn't intersect the interior of B , therefore \overline{pvi} and $vi \overline{\Phi^{-1}(vi)}$ make a left turn at vi , $\overline{p\Phi^{-1}(vi)}$ is counter clockwise from \overline{pvi} , and \overline{pvi} and $vi \overline{\Phi(vi)}$ make a right turn at vi , $\overline{p\Phi(vi)}$ is a clockwise from \overline{pvi} , this gives a proof of equation (Eq.5.5) (see figures 5.11, 5.12)

For the last part (Eq.5.6), since \overline{pvi} intersects the interior of B , therefore \overline{pvi} and $vi \overline{\Phi^{-1}(vi)}$ make a right turn at vi , $\overline{p\Phi^{-1}(vi)}$ is clockwise from \overline{pvi} this gives a proof of (Eq.5.6) (figures 5.11, 5.12)

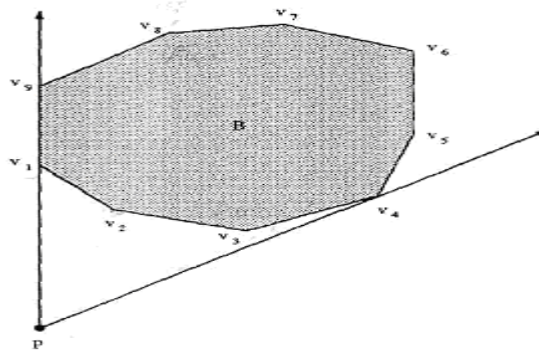


Figure 2.25 Visibility from point p to convex polygon B

For example, in Figure 2.27, vertex v_1 is *cw-tangential*, vertex v_2 is *visible*, vertex v_4 is *ccw-tangential* and vertex v_7 is *invisible*

3.5.2 Type of an Image from a Point to a Convex Polygon

Let B denote a convex polygon with n vertices. Let a point $p \in \text{free}(B)$. The image of p may be on an edge or a vertex of convex polygon. If an image of p is on an edge, the image moves when p moves slightly. However, if the image of p is on a vertex, it does not move when p moves slightly. The following theorem determines the image occurs either on an edge or on a vertex.

Theorem 5.1: let $B = \{v_1, \dots, v_n\}$ be a convex polygon, and let p be a point in $\text{free}(B)$ and define θ , θ_1 and θ_2 by:

$$\begin{aligned}\theta &= \psi(v_i, \Phi(v_i)) + \pi/2 \\ \theta_1 &= \psi(p, v_i) \\ \theta_2 &= \psi(v_i, \Phi(v_i))\end{aligned}$$

Let vertex v_i be *cw-tangential* from point p . there exists a vertex v_j ($i = j$ or $i \neq j$) such that the image of p on B is of one of the following two types.

$$(I) \text{ If } (\theta_1 > \theta) \wedge (\theta_2 < \theta) \quad (5.7)$$

then the image lies on an edge $\overline{v_i \Phi(v_i)}$ of polygon B .

$$(II) \text{ If } (\theta_1 \leq \theta) \wedge (\theta_2 \leq \theta) \quad (5.8)$$

then the image of p is a vertex v of polygon B .

Proof.

Consider two straight lines, one joining p with v_i and the other joining p and $\Phi(v_i)$. The orientations of these two lines are θ_1 and θ_2 respectively. Also, denote by α is the orientation of $\overline{v_i \Phi(v_i)}$ from v_i to $\Phi(v_i)$ and is $\theta = \alpha + \frac{\pi}{2}$ the perpendicular from p to $\overline{v_i \Phi(v_i)}$.

For the first part of the proof (Eq. 5.7), let p_{im} be the intersection of two lines whose orientations are α and θ (Figure 2.28). Assume that the hypothesis of

Eq.5.7 is true. Since $\theta_1 > \theta$, then $\overline{pp_{im}}$ and $\overline{p_{im}v_i}$ make a *left turn* (see appendix A) at p_{im} . Also, $\theta_2 < \theta$, then $\overline{pp_{im}}$ and $\overline{p_{im}\phi(v_i)}$ make a right turn at p_{im} . It follows that p_{im} is visible from p by lemma 5.2. This means that v_i and $\phi(v_i)$ are on opposite sides of p_{im} . Therefore, p_{im} lies on the boundary of B . In other words, p_{im} lies on an edge of B .

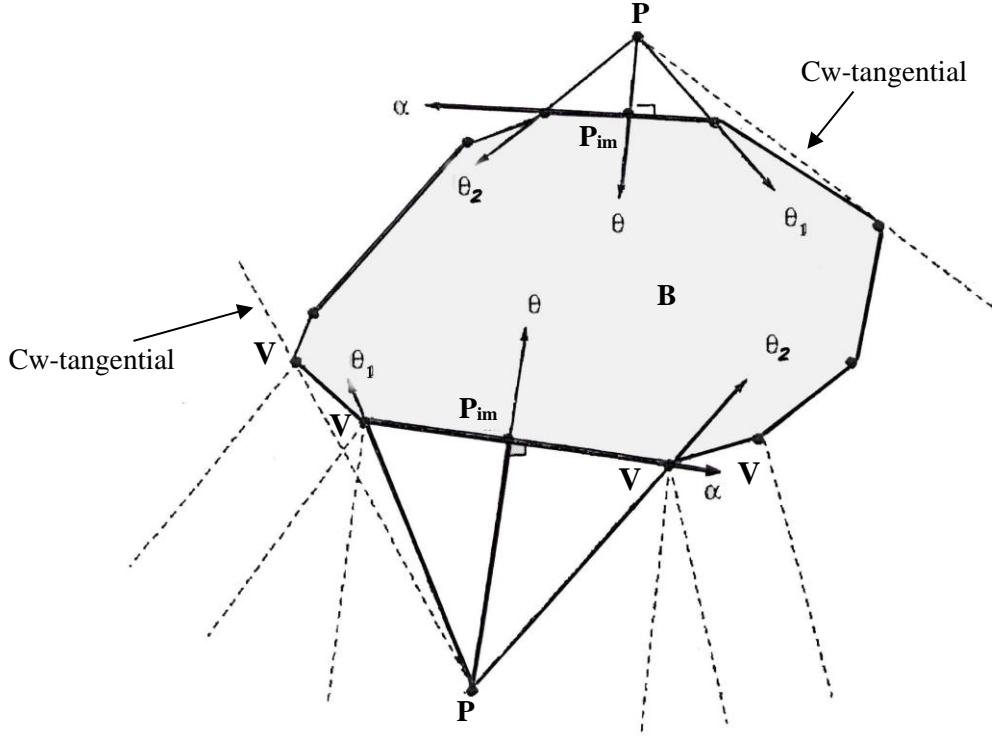


Figure 2.26: Images of point p lies on an edge of convex polygon B

For the second part (Eq. 5.8), assume that the hypothesis of Eq. 5.8 is true; we have the following three cases (see Figure 2.12).

- Case 1: $\theta_1 < \theta$ AND $\theta_2 < \theta$

Since $\theta > \theta_1$, and $\theta > \theta_2$ - Therefore the image of p does not lie on the edge $\overline{v_i\phi(v_i)}$. But $\theta_1 < \theta$, since $\phi(v_i)$ is the next vertex to v_i . Then v_i is a closed point from p . Therefore, the image of p is a vertex v_i .

- Case 2: $\theta_1 = \theta$ AND $\theta_2 < \theta$

Since $\theta_1 = \theta$ and $\theta_2 < \theta$, then the image of p does not lie on the edge $v_i\phi(v_i)$. But $\theta_1 < \theta_2$ since $\phi(v_i)$ is the next vertex to v_i . Then v_i is a closed point from p . Therefore, the image of p is a vertex v_i .

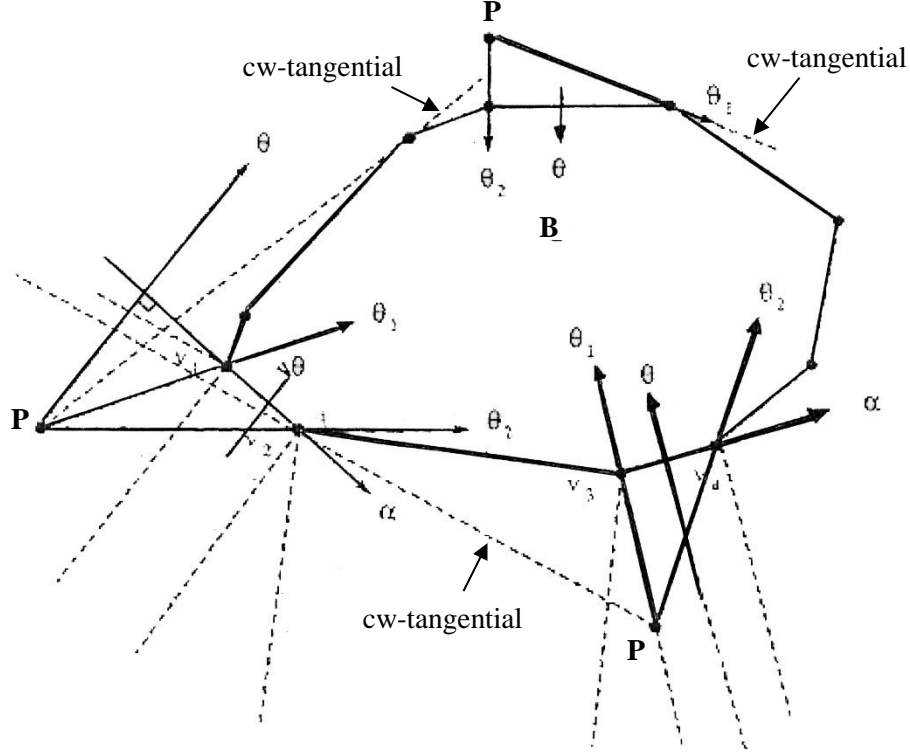


Figure 2.27: Image of point p lies on vertex v_i of convex polygon B

- Case 3: $\theta_1 < \theta$ AND $\theta_2 = \theta$

Since $\theta_1 < \theta$ and $\theta_2 = \theta$, then the image of p does not lie on the edge But $v_i\phi(v_i)$ $\theta_1 < \theta_2$, since $\phi(v_i)$ is the next vertex to v_i . Then $\phi(v_i)$ is a closed point from p .

Therefore, the image of p is a vertex $\phi(v_i)$.

This gives a proof of Eq. 5.8. in other words, p_{im} occurs on a vertex of B .

Since there are no vertices in the interior of a convex polygon B , then by Theorem 5.1 we obtain the following corollary.

Corollary 5.1: For any point $P \in \text{free}(B)$ and a convex polygon B , there exists only one image from p to a convex polygon B .

3.5.3 The Image Type Algorithm

Now, we describe the construction of an algorithm for image type. The block diagram for finding image type is shown in Figure 2.29. The inputs are a convex polygon B and a point $p \in \text{free}(B)$. The outputs are an image type (vertex type or edge type), a vertex v_i , the orientation from p to its image, and the closed distance from p to the image. For a vertex type image, vertex v_i is the image of p on B , but for an edge type image, the image of p on B lies on an edge $\overline{v_i \phi(v_i)}$. In pseudo-code the method is as follows:

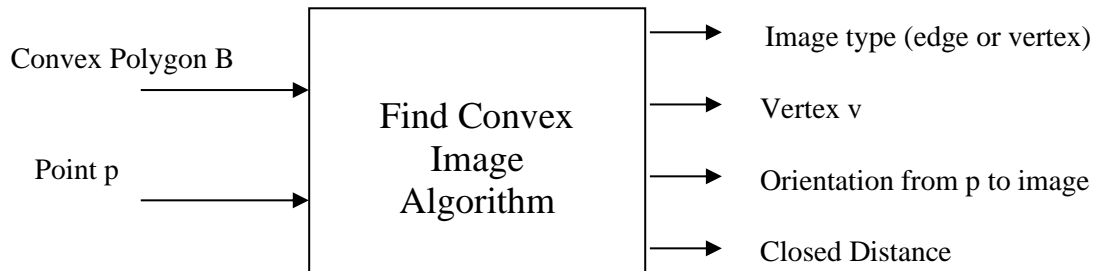


Figure 2.28 : Image type block diagram

Convex Image(p, B)

Input: *point* $p (\in \text{free}(B))$
 Convex polygon $B = (v_1, \dots, v_n)$

Output: *image* image type (vertex-type or edge-type)
 vertex v
 orient (the orientation from p to image)
 closed (the distance from p to image)

BEGIN

```

v = first vertex(B)
***Find clockwise tangential***
While ( $ccw(p, v, \Phi^{-1}(v))$  or  $ccw(p, v, \Phi(v))$ )
   $v = \Phi(v)$ 
Loop
***Find image type***
Do Until (image reached)
   $\theta = 00) + \pi/2$ 
  
```

```

 $\theta_1 = \psi(p, v)$ 
 $\theta_2 = \psi(P, \Phi(v))$ 
if  $((\theta_1 \leq \theta) \wedge (\theta_2 \leq \theta))$  then
    image.type = VERTEX
    image.posi = v
    image.orient =  $\theta_1$ 
    image.closed = Computer_Education_Distance(p, v)
else if  $((\theta_1 > \theta) \wedge (\theta_2 < \theta))$  then
    image.type = EDGE
    image.posi = v
    image.orient =  $\theta$ 
    image.closed = Compute_Dist(p, v)
else
     $v = \Phi(v)$ 
end if
Loop
END

```

The algorithm simply loops until the *image* is reached. First, the algorithm loops until a cw-tangential vertex is reached.

Through the second loop, we check the condition for vertex image type. If the condition is not satisfied, the condition for edge image type is checked. Also, if it is not satisfied, we take the next vertex in ccw direction. We continue in this process until one of the above two conditions satisfies.

The subroutine "Compute_Euclidean_Distance" computes the distance between two points ($d(p_1, p_2) \equiv \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$). The subroutine "Compute_Dist" computes the closest distance from point p to its image which lies on an edge. The subroutine for Compute_Dist is as shown below:

Compute_Dist (p, v)

Input: point p ($\in \text{free}(B)$) V first vertex of edge where the image on it

output: closed the closet distance from p to *image*

begin

$area = \text{Compute_Area_Triangle}(p, v, \Phi(v))$

$dist = \text{Compute_Euclidean_Distance}(v, \Phi(v))$

$closed = 2 \times area / dist$

return $closed$

end

The subroutine `Compute_Area_Triangle` computes the area of triangle (see Appendix A).

3.5.3.1 Proof of Correctness of the Algorithm

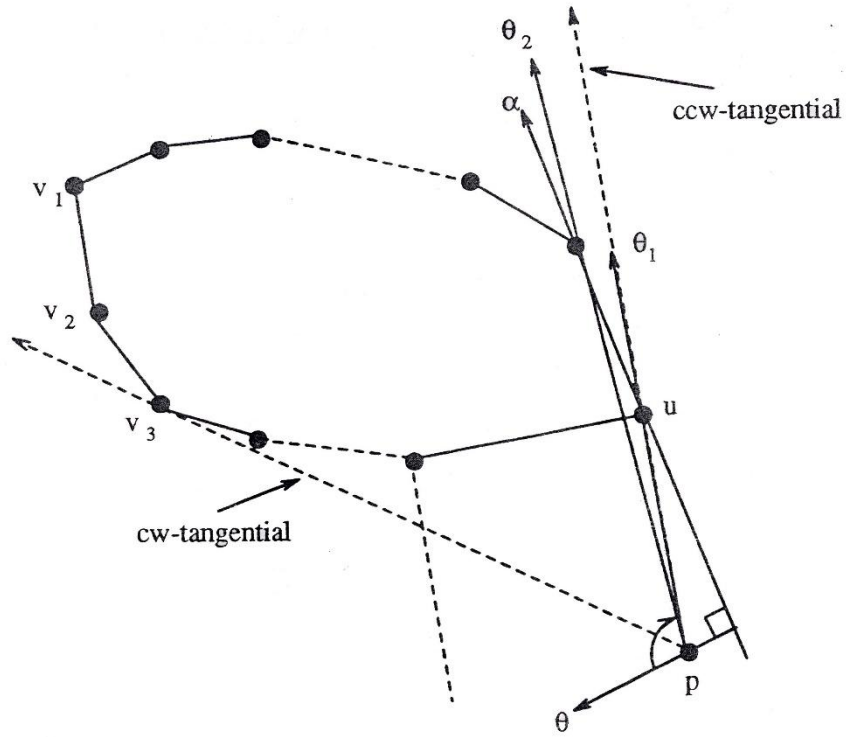


Figure 2.30: Correctness of image type algorithm

To prove the correctness of the above algorithm, we want to show that the algorithm always returns an *image* structure when the Do-loop is executed. In other words, the Do-loop is never executed forever.

Assume that v_1 is the starting vertex of polygon B (Figure 2.30). Since v_3 is *cw-tangential*, the while-loop returns $v = v_3$. It follows that, at the beginning of the Do-loop, v will be checked to determine the image type.

If the two if conditions are not satisfied, we take the next vertex, $v = \Phi(v)$. In the worst-case, we continue in this process until vertex $v = v_n$. Vertex v is *ccw-tangential*, but the first if condition will be satisfied ($\theta_1 < \theta \wedge \theta_2 < \theta$). It follows that the algorithm returns the image type of point p as vertex type and vertex v . This proves that the Do-loop is always terminated.

3.5.3.2 Analysis of the Worst-Case Time Complexity of the Algorithm

The operations each takes $O(1)$ time, except. The while loop will be taken $O(n)$ time in the worst-case. The Do-until-loop takes $O(n)$ time in the worst-case. The overall running time of the algorithm is $O(n)$.

2.5.4 Finding an Image on a Non-convex Polygon

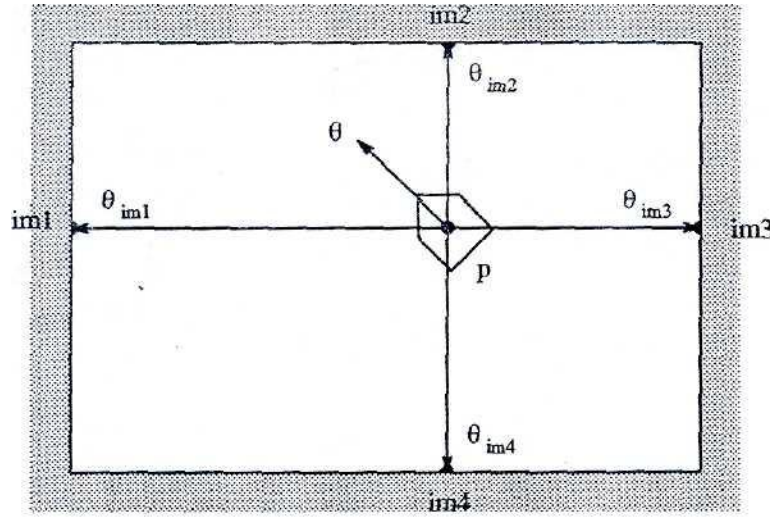


Figure 2.31: Image of a point p on cw concave polygon B

Suppose we have an outermost non-convex *cw* polygonal boundary (Figure 2.31) or non-convex *ccw*-polygon obstacle inside the boundary (Figure 2.32). Let a point $p \in \text{free}(B)$. In the case of an outermost non-convex *cw*-polygon, there is more than one image. The image always lies on an edge of B . In the case of non-convex *ccw* polygon, there may be one or more images depending upon the position of the robot. The image may be one of the vertices of B or it may lie on an edge of B . We have the following observations. First, the image may be behind the vehicle. For instance, in Figure 2.31, p_{im3} and p_{im4} are behind the vehicle. In this case, this can not be an image.

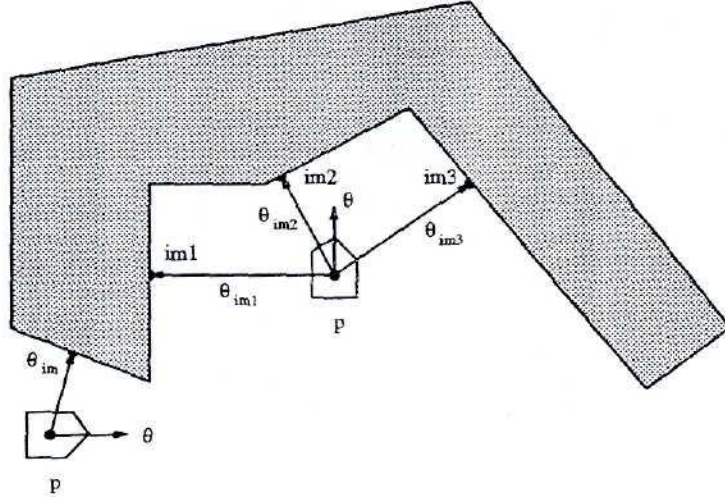


Figure 2.32: Image of a point p on ccw concave polygon B

The following remark illustrates how we can know whether the image is behind a vehicle. Let θ denote a vehicle's heading (the direction from p) and let $\psi(p, p_{im})$ denote the direction from p to p_{im}

Remark 5.1: given a non convex polygon B and a point $p \in \text{free}(B)$

(a) if

$$| \Phi(\theta - \psi(p, p_{im})) | \leq \Pi/2 \quad (5.9)$$

then the image of p is usable.

(b) if

$$| \Phi(\theta - \psi(p, p_{im})) | > \Pi/2 \quad (5.10)$$

then the image of p is behind the vehicle.

The second observation, for the usable image (Eq. 5.9) the following remark illustrates how we can know if the image is in the right, left or front of the vehicle.

Remark 5.2: the real image is of one of the following three types.

(a) if

$$\Phi(\theta - \psi(p, p_{im})) > 0 \quad (5.11)$$

Then the image of p is on the right of the vehicle.

(b) if

$$\Phi (\theta - \psi (p, p_{im})) < 0 \quad (5.12)$$

Then the image of p is on the left of the vehicle.

(c) if

$$\Phi (\theta - \psi (p, p_{im})) = 0 \quad (5.13)$$

Then the image of p is on the front of the vehicle.

To summarize: in the case of a non-convex polygon, we conclude that

1. We need another algorithm to find the image(s).
2. We need another data structure for the image. In this case, we may have one or more images. Therefore, we need an array of image structures. The size of this array is the maximum numbers of images.
3. If the initial orientation, θ , of the vehicle is in the opposite direction to the desired motion of the vehicle, then we cannot reject the, image which lies behind the robot.

According to above, the division of the boundary cw-polygon into set of sub polygons will let us use the same algorithm for convex polygons and the same data structure for the image.

2.6 Path Classes and Sub-Polygons

The objective of path classes using polygonal world is to provide useful information for local motion planning. The directed v-edges sequences, of a world W which consists of a finite number of polygons n is independent of the position of the vehicle inside the $free(W)$. For example, in Figure 2.17, suppose the path class $\pi = [B_1 / B_0] [B_2 / B_0]$ and the start configuration of the vehicle are given. Also, we know that any point within $free(W)$ has its left and right images on the two boundary polygons. We proved previously that there is only one image of a point (that lies in free space) to a convex polygon and more than one image for a non convex polygon. When representing the path class in a polygonal world, we have the following disadvantages:

1. In Figure 2.33, B_1 and B_2 are *ccw* convex polygons and a B_0 is *cw* non convex polygon. When the vehicle starts to navigate through the given path Π , left image is im_3 and its right images are im_1 and im_2 . Since the start orientation of the vehicle is θ , the right images are im_1 and im_2 , but im_2 is behind the vehicle.
2. We can not construct the connectivity graph if a world W consists of two polygons B_0 and B_1 , where W is encircled by an outermost *cw* polygonal boundary B_0 , and has one *ccw* polygonal obstacle B_1 inside the boundary (Figure 2.34), since every v-node of the connectivity graph is the common intersection of three or more Voronoi boundary segments.

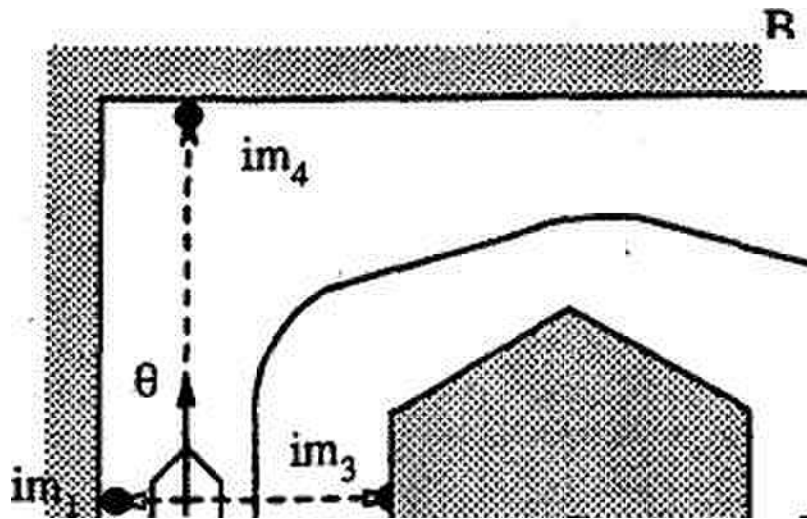


Figure 2.33: Problem 1: initial orientation of a vehicle is different from the direction of a motion

Due to the above shortcomings, we need more information when we represent the path classes in order to simplify local motion planning. The use of the sub-polygons (refer to sec. 5.1) will solve the above problems and give more information for the local motion planning task.

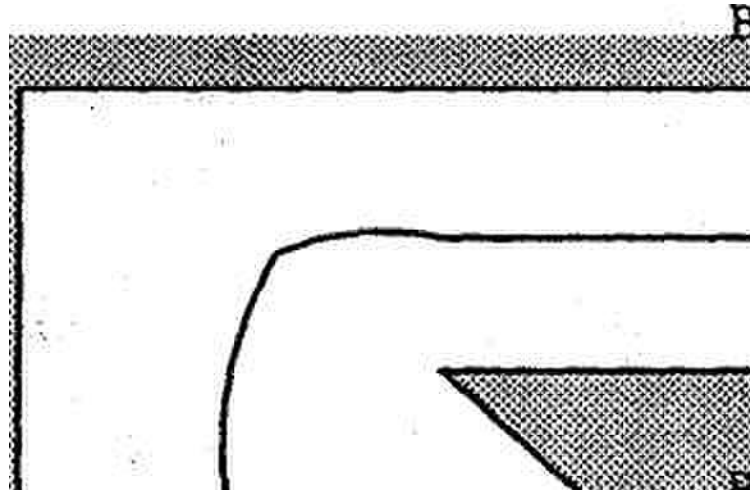


Figure 2.34 : Problem 2: Voronoi diagram of polygonal world consisting of two polygons (ccw polygon inside cw polygon boundary)

Consider the same world W in Figure 2.33 The non convex polygon B_0 can be broken into four sub-polygons B_{00} , B_{01} , B_{02} , and B_{03} (Figure 2.35). The basic connectivity graph generated that world is shown in Figure 2.36. There are six v-nodes (v_1, \dots, v_6) and seven undirected v-edges:

$$[B_1 : B_{00}], [B_1 : B_{01}], [B_1 : B_{03}], [B_2 : B_{01}], [B_2 : B_{02}], [B_2 : B_{03}], [B_1 : B_2].$$

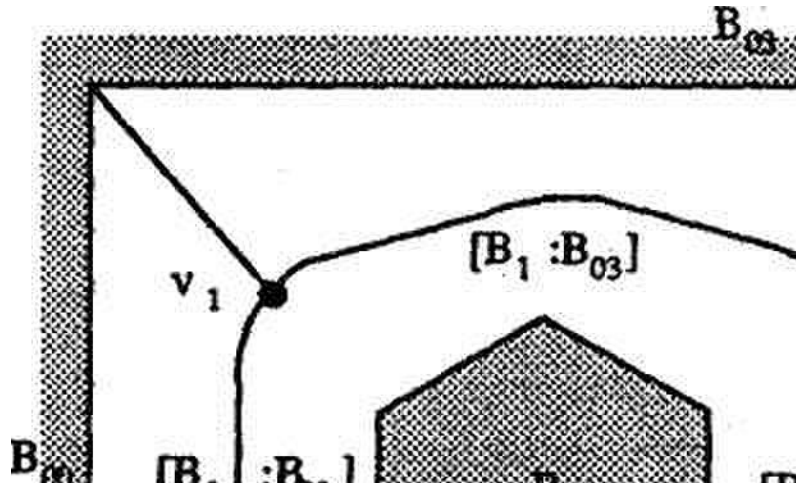


Figure 2.35: Solution of problem 1: Voronoi diagram of a sub polygonal world

Now, assume that a start configuration, S , and a goal configuration, G , are given in $\text{free}(W)$ (Figure 2.35). The augmented connectivity graph generated by this world is shown in Figure 2.37. While there are four different path classes represented by a directed v-edges sequences as follows:

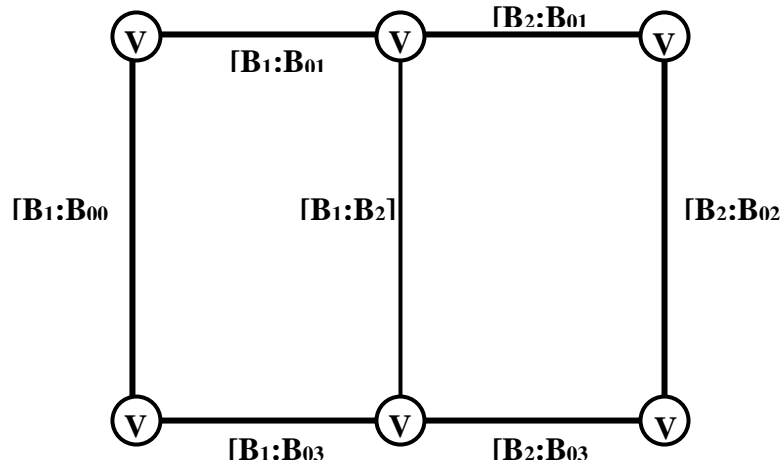


Figure 2.36: Basic connectivity graph of a sub polygonal world

$$\pi_1 = [B_{00}/B_1] [B_{03}/B_1] [B_{03}/B_2] [B_{02}/B_2]$$

$$\pi_2 = [B_{00}/B_1] [B_{03}/B_1] [B_2/B_1] [B_2/B_{01}] [B_2/B_{02}]$$

$$\pi_3 = [B_1/B_{00}] [B_1/B_{01}] [B_1/B_2] [B_{03}/B_2] [B_{02}/B_2]$$

$$\pi_4 = [B_1/B_{00}] [B_1/B_{01}] [B_2/B_{01}] [B_2/B_{02}]$$

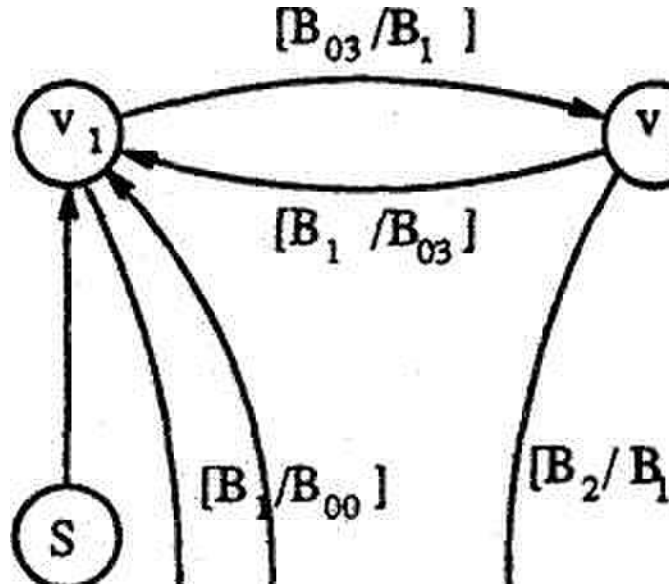


Figure 2.37: Augmented connectivity graph of a sub polygonal world

As a result, the use of sub-polygons solves the problem when the start orientation of the vehicle is different from the direction of the motion. In other words, path classes represented by sub-polygons possess more information for local motion planning than do those represented by polygons.

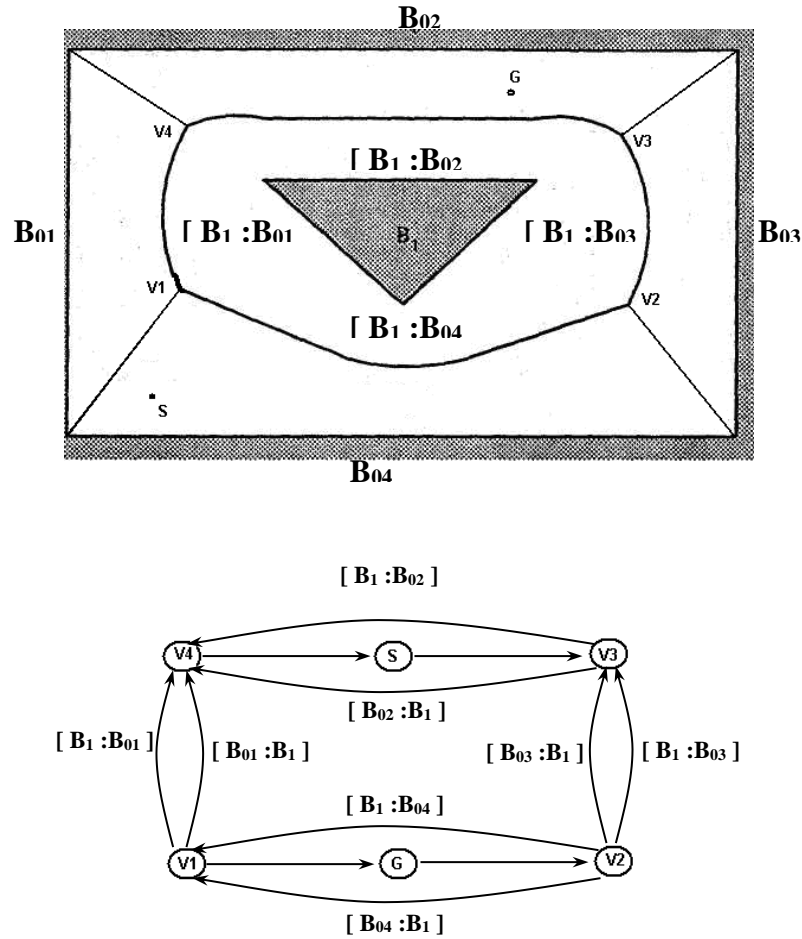


Figure 2.38: Solution of problem 2: world and augmented connectivity graph

2.7 Advantages of Path Class Representation Using Directed V-Edges Sequences

There are several advantages. They include:

1. A unique representation of a path class. In other words, this representation is unambiguous since a directed v-edge is defined by the "closest" two obstacle features (like vertices

sequence, obstacle shape and any other properties of the obstacle which will be used by the robot sensors through its navigation).

1. It is exact free space decomposition, so that if a path exists, the local motion planning should be able to find it.
2. It simplifies the planning of collision-free paths for a robot among obstacles once the directed v-edge sequence, in which the robot is located, is identified.
3. The local motion planning problem is expected to be simpler if a path class representing by directed v-edge sequence is given.
4. This PCMP algorithm of solving the motion planning problems is exactly the same as the human mind, solve any transition mission in any known environment.

CHAPTER.4

**SHORTEST PATH PLANNING
ALGORITHM**

4.1 CANONICAL PATHS AND TANGENT SEUENCES

This text introduces the concept of canonical paths and tangent sequences in the context of path planning within a world of convex polygons. Here's a breakdown:

Canonical Paths:

- Informally referred to as the **shortest paths** (locally) within a specific class of paths.
- The goal is to identify these "canonical paths" symbolically using a concept called a **tangent sequence**.

World Definition (W):

- Represented by the symbol W , it's a set of n convex polygons ($n > 0$).
 - Convex polygons have all interior angles less than 180 degrees and no inward corners.

Oriented Polygons (B_i^+ , B_i^-):

- Each polygon B_i in the world W can be represented in two orientations:
 - B_i^+ (counter-clockwise): This signifies following the polygon's boundary in a counter-clockwise direction.
 - B_i^- (clockwise): This signifies following the polygon's boundary in a clockwise direction.
- This concept of orientation is crucial for path planning algorithms.

Set of All Oriented Polygons (Z):

- Represented by the symbol Z , it's the collection of all possible oriented polygons derived from the original set W .
- Formula: $Z = \{B_1^+, B_1^-, \dots, B_n^+, B_n^-\}$ (contains $2n$ elements, one for each polygon in both orientations).

Tangent Sequence (σ):

- Defined as a sequence of oriented polygons.
- A key restriction: The sequence cannot contain subsequences like $B+B^-$ or $B-B^+$ where B belongs to the original set W (prevents going around a single polygon in both directions).
- Empty sequence: denoted by ϵ .

Summary:

This explanation sets the stage for understanding how canonical paths (shortest paths within a class) can be identified and represented using tangent sequences, which define valid sequences of oriented polygons for navigating within a world of convex obstacles. By analyzing tangent sequences, the algorithm can potentially determine the most efficient path for the robot to travel.

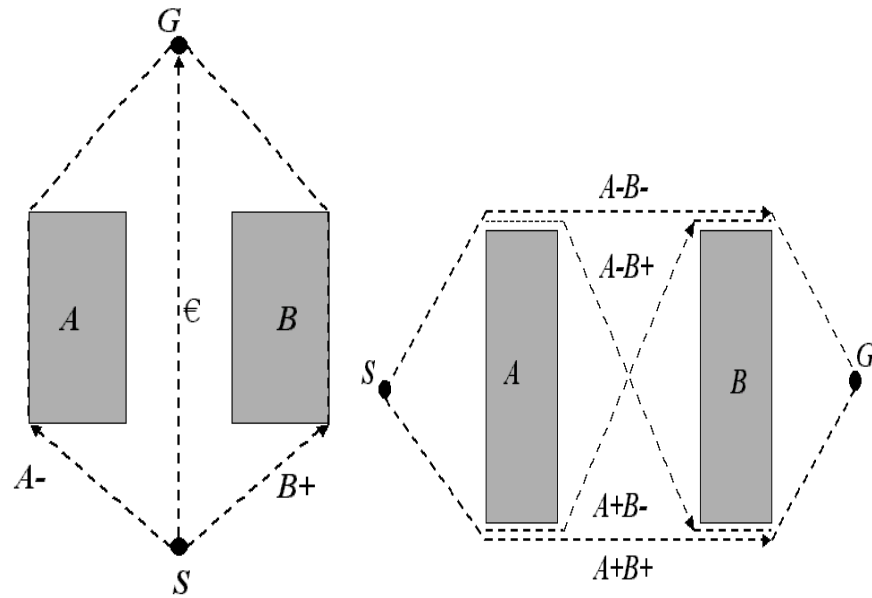


Figure 3.1. Examples of canonical paths

Interpreting Tangent Sequences into Paths and Visibility

This text builds upon the previous explanation of canonical paths and tangent sequences, introducing how these sequences translate to actual paths and the concept of visibility in the context of path planning.

Interpreting Tangent Sequences ($I(\sigma)$):

- The text defines a function $I(\sigma)$ that takes a tangent sequence (σ) and interprets it as a path (π).

- The specific details of this interpretation process are likely explained later, but it essentially translates the sequence of oriented polygons into a valid path for the robot to follow within the world.

Canonical Paths (π):

- A path (π) is considered **canonical** if there exists a corresponding tangent sequence (σ) such that $\pi = I(\sigma)$.
- In simpler terms, a path is considered "shortest within its class" if it can be represented by a valid tangent sequence.

Visibility in Free Space ($\text{free}(W)$):

- The text introduces the concept of visibility between two points (p_1 and p_2) within the free space ($\text{free}(W)$).
- Two points are considered **visible** if there are no obstacles (polygons in W) that block a straight line segment connecting those points.
- This concept is crucial for path planning algorithms, as the robot needs to consider visibility between points when navigating the environment.

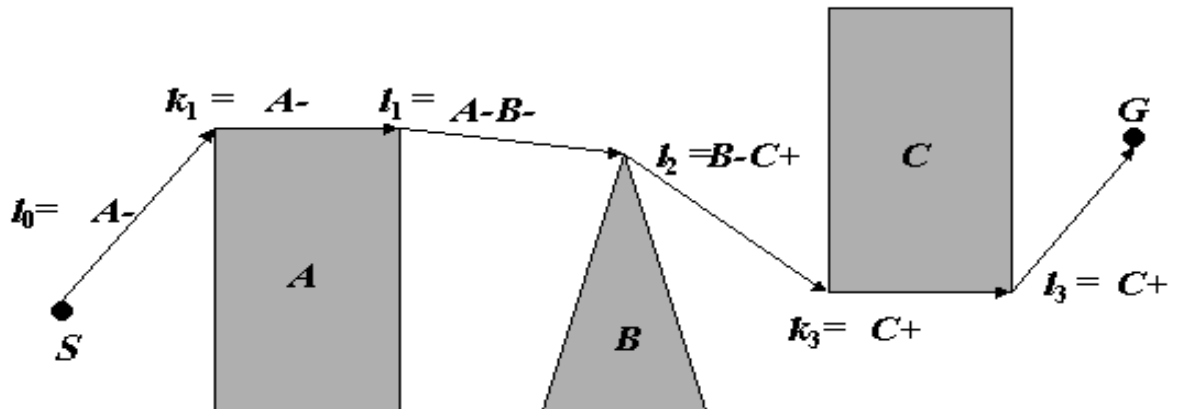


Figure 3.2. Tangent sequences $A^-B^-C^+$ and Its canonical path

$$\left\{ \begin{array}{ll} SG & \text{if visible (S, G)} \\ \text{Undefined SG} & \text{if invisible (S, G)} \end{array} \right. \quad (\text{III.3})$$

Where visible (S, G) means that S and G are visible in W:

Visibility for Start and Goal (visible (S, G)):

- This notation signifies that the starting point (S) and the goal point (G) of the robot's path are visible within the world (W). In simpler terms, there are no obstacles blocking a straight line connecting S and G.

Formal Interpretation of Tangent Sequence (I(σ)):

- The formula $I(\sigma) = l_0 k_1 l_1 \dots k_q l_q$ defines how a tangent sequence (σ) is interpreted as a path.
- Here's a breakdown of the formula:
 - $\sigma = B_{\mu_1 i_1} l_1 \dots B_{\mu_q i_q}$ (represents the tangent sequence with oriented polygons and their signs)
 - $q \geq 1$ (ensures there's at least one polygon in the sequence)
 - $\mu_1, \dots, \mu_q \in \{+1, -1\}$ (signs indicating polygon orientations)
- The right side of the equation represents the concatenation (combination) of smaller sub-paths:
 - l_0, l_1, \dots, l_q (these represent sub-paths based on tangent lines)
 - k_1, \dots, k_q (minimum portions of polygon boundaries)
- likely [Figure 3.2](#)

Details of Sub-Paths and Visibility:

- l_0 : This is a **μ_1 -tangent sequence** from the starting point (S) to the first polygon (B_{i_1}) in the sequence. It considers the visibility between S and B_{i_1} .
 - If this tangent is not visible (blocked by an obstacle), the entire path $I(\sigma)$ is undefined.
- l_j (for $1 \leq j \leq q-1$): These represent **$\mu_j \mu_{j+1}$ common tangents** between consecutive polygons (B_{i_j} to $B_{i_{j+1}}$). These tangents connect points on the boundaries of adjacent polygons while considering visibility.
 - Similar to l_0 , if any of these common tangents is not visible, the path $I(\sigma)$ is undefined.
- l_q : This is a **μ_q -tangent sequence** from the last polygon (B_{i_q}) in the sequence to the goal point (G). It considers the visibility between B_{i_q} and G.

- Again, if this final tangent is not visible, the path $I(\sigma)$ is undefined.

Minimum Portions of Boundaries (k_j):

- For each j ($1 \leq j \leq q$), k_j represents the **minimum portion** (possibly empty) of the **μ_{ij} -oriented boundary** of polygon B_{ij} .
- This portion connects the point of intersection with the previous tangent (l_{j-1}) and the point of intersection with the next tangent (l_{j+1}) on the polygon's boundary.
- Essentially, k_j defines the specific part of the polygon's boundary that the robot needs to traverse along the path.

Proposition III.1 *For any given W , S and G , a canonical path is the shortest one among all paths in a homotopy class.*

Proof. Let σ be a tangent sequence such that $\pi = I(\sigma)$. If $\sigma = \epsilon$, the straight segment SG is the shortest. Let $\sigma \neq \epsilon$. Consider a normal at each osculating point (each endpoint) of every tangent,

l_i ($0 \leq i \leq q$), in π . The canonical path π and all paths which are homotopic to π intersects each of these normals exactly one time. If π intersects any of these normals other than the polygon boundary, each path segment becomes longer than l_i s and k_i s in the canonical path.

Corollary III.1 *For given W , S and G , the shortest path is a canonical path.*

As the previous analysis shows, a locally shortest path (or canonical path) consists of a sequence of four kinds of path segments:

- A tangent from S to a polygon.
- Common tangents between two polygons.
- A tangent from a polygon to G .
- Portions of oriented polygon boundaries

Therefore, the shortest path finding problem is essentially a search problem in a world using tangents and oriented boundaries. Since there are only a finite number of tangents in the given world, this problem is converted into a graph search problem.

4.2 TANGENTS ON POLYGONS

This section dives into the details of finding tangents and common tangents between a ray (directed line) and a convex polygon.

Tangents on a Convex Polygon (B):

- The text defines a tangent on a convex polygon (B) using set theory notation:
 - L: The ray (directed line) in the free space surrounding the polygon ($\text{free}(B)$) or on the polygon's boundary (B).
 - $[(L \cap B) \neq \emptyset]$: Ensures the line intersects the polygon (B) at least once.
 - $[(L \cap \text{int}(B)) = \emptyset]$: Ensures the line doesn't intersect the interior ($\text{int}(B)$) of the polygon.

Osculating Point:

- Based on the definition, a tangent line (L) must touch the polygon (B) at either:
 - A vertex (v) of the polygon.
 - An edge ($v\phi(v)$) of the polygon (where $\phi(v)$ refers to the next vertex in the counter-clockwise order).
- This point of contact between the tangent and the polygon is called the **osculating point**.

Orientation of Tangents (Plus/Minus):

- The text introduces the concept of tangent orientation based on the position of the polygon's interior:
 - **Plus Tangent (Counter-clockwise)**: When the polygon's interior ($\text{int}(B)$) is on the left side of the tangent line (L).
 - **Minus Tangent (Clockwise)**: When the polygon's interior ($\text{int}(B)$) is on the right side of the tangent line (L).
- Figure 10 (not provided) likely illustrates these concepts visually.

Additional Notes:

- This section focuses on convex polygons, where all interior angles are less than 180 degrees, and there are no inward corners.
- The concept of tangent orientation will likely be crucial for later discussions on finding common tangents between two polygons.

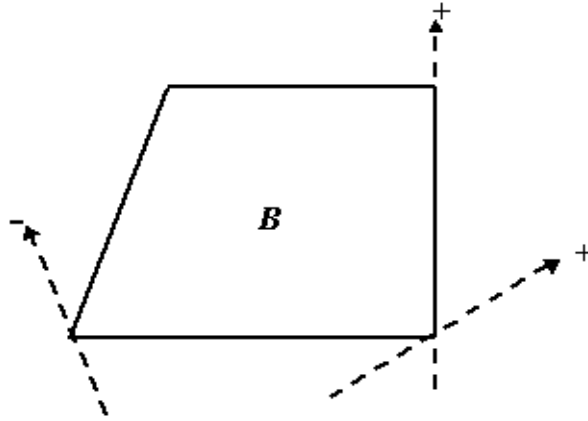


Figure 3.3. Tangents

4.2.1 Tangents from point to polygon

Assume that we are given a polygon B , a point $p \in \text{free}(B)$, and an orientation μ of tangents ($\mu = +1$ or -1).

Let $\text{tangnet}_{01}(p, B, \mu, v)$ denotes a condition for the ray \overrightarrow{pv} being a μ tangent on B .

Proposition III.2

(I) $\text{Tangnet}_{01}(p, B, +1, v) =$

$$[ccw(p, v, \phi^{-1}(v)) \wedge ccw(p, v, \phi(v))] \vee [col(p, v, \phi(v)) \wedge ccw(p, v, \phi^{-1}(v)) \wedge ccw(p, v, \phi^2(v))] \quad (\text{III.5})$$

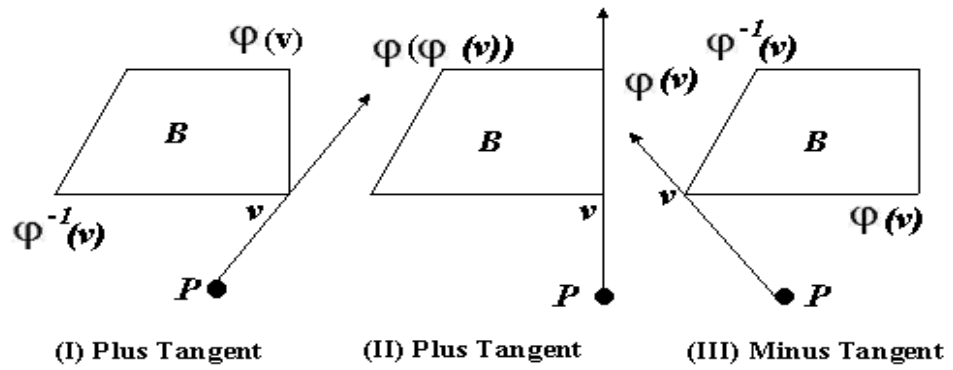


Figure 3.4 Tangents from point to polygon

First Condition: $[ccw(p, v, \phi^{-1}(v)) \wedge ccw(p, v, \phi(v))]$

This condition uses the concept of "ccw" (counter-clockwise). It might check if point p , vertex v , and its neighbors ($\phi^{-1}(v)$ and $\phi(v)$) form a counter-clockwise sequence.

If this condition is true, the tangent line likely touches vertex v and goes between $\phi^{-1}(v)$ and $\phi(v)$.

Second Condition: $[col(p, v, \phi(v)) \wedge ccw(p, v, \phi^{-1}(v)) \wedge ccw(p, v, \phi^2(v))]$

This condition uses "col" (collinear) and "ccw". It might check if point p , vertex v , and its next vertex ($\phi(v)$) are collinear (on the same line).

Additionally, it might check if point p , vertex $\phi^{-1}(v)$, and the vertex two positions ahead ($\phi^2(v)$) form a counter-clockwise sequence.

If this condition is true, the tangent line likely coincides with the edge between v and $\phi(v)$ and goes between $\phi^{-1}(v)$ and $\phi^2(v)$.

Understanding $\phi(v)$:

- $\phi(v)$ refers to the next vertex of polygon B in the counter-clockwise direction from vertex v .

(II)

$$\text{Tangnet}_{01}(p, B, -1, v) =$$

$$[cw(p, v, \phi^{-1}(v)) \wedge cw(p, v, \phi(v))] \vee$$

$$[col(p, v, \phi^{-1}(v)) \wedge cw(p, v, \phi(v)) \wedge cw(p, \phi^{-1}(v), \phi^2(v))]$$

(III.6)

Two conditions are presented, each likely defining the tangent line under different geometric circumstances:

First Condition: $[cw(p, v, \phi^{-1}(v)) \wedge cw(p, v, \phi(v))]$

This condition uses the concept of "cw" (clockwise). It might check if point p , vertex v , and its neighbors ($\phi^{-1}(v)$ and $\phi(v)$) form a clockwise sequence.

If this condition is true, the tangent line likely touches vertex v and goes between $\phi^{-1}(v)$ and $\phi(v)$.

Second Condition: $[col(p, v, \phi^{-1}(v)) \wedge cw(p, v, \phi(v)) \wedge cw(p, \phi^{-1}(v), \phi^{-2}(v))]$

This condition uses "col" (collinear) and "cw". It might check if point p , vertex v , and its previous vertex ($\phi^{-1}(v)$) are collinear (on the same line).

Additionally, it might check if point p , vertex $\phi(v)$, and the vertex two positions before ($\phi^{-2}(v)$) form a clockwise sequence.

If this condition is true, the tangent line likely coincides with the edge between v and $\phi^{-1}(v)$ and goes between $\phi^{-2}(v)$ and $\phi(v)$.

Understanding $\phi(v)$:

- 1 $\phi(v)$ refers to the next vertex of polygon B in the counter-clockwise direction from vertex v .

Proposition III.3 we consider a problem of finding the osculating point (or the first osculating point if there are two) v on B such that the \overline{pv} is a μ tangent on B (" μ tangent " means plus tangent if $\mu = +1$ and a minus tangent if $\mu = -1$). This point (vertex) v is said to be a landing point in this problem (see Fig. 11). For any p , B , and μ , if $p \in \text{free}(B)$, there exists one and only one μ tangent from p to B .

Plus-tangent-from-point-to-polygon (p, B)

begin

$v := \text{first_vertex}(B)$

doforever

if $ccw(p, v, \phi^{-1}(v))$

then if $ccw(p, v, \phi(v))$

then return (v)

else $v := \phi(v)$

else $v := \phi^{-1}(v)$

end.

In order to give a first idea, given a point p and a polygon B , an algorithm of finding the plus landing point (landing vertex) v is shown above. We assume the position of p is "general" so that only one vertex is on the resultant tangent on B . Here first_vertex is defined as a function, which returns the first vertex of a polygon. We move v towards the right until it gets at the correct landing point through the next or previous function. Its computation time is $O(|B|)$. We generalize this program so that it can handle the cases where $\mu = \pm 1$ and the case where an edge (not only a vertex) is on the resultant tangent.

Tangent-from-point-to-polygon (p, B, μ)

begin

$v := \text{first_vertex}(B)$

doforever

if $O(p, v, \varphi^{-\mu}(v)) = \mu$

then if $O(p, v, \varphi^{\mu}(v)) \neq -\mu$

then **return** (v)

else $v := \varphi^{\mu}(v)$

else $v := \varphi^{-\mu}(v)$

end.

4.2.2 Tangents from Polygon to Point

Now, given a polygon B , a point $p \in \text{free}(B)$, and an orientation μ , we consider a problem of finding the osculating point (or the second osculating point if there are two) v on B such that the ray vp is the μ tangent on B . This point (vertex) v is said to be a leaving point in this problem (see Figure 3.5).

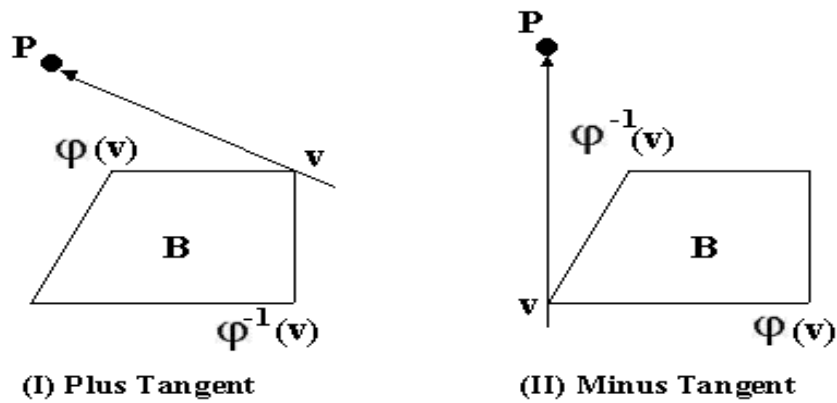


Figure 3.5 Tangents from polygon to point

Let $\text{Tangnet}_{10}(p, B, \mu, v)$ denotes a condition for the ray \overrightarrow{vp} being a μ tangent on B .

Proposition III.4 For any point p , convex polygon B , and μ ,
if $p \in \text{free}(B)$ and if v is a vertex on B ,

$$\text{Tangnet}_{10}(p, B, \mu, v) = \text{Tangnet}_{01}(p, B, -\mu, v) \quad (\text{III.7})$$

Due to the last Proposition, we will use the point-to-polygon algorithm in the previous section to find polygon-to-point tangents instead of creating a new algorithm.

Proposition III.5 For any p, B , and μ , if $p \in \text{free}(B)$, there exists one and only one μ tangent from p to B .

4.2.3 Common Tangents Between Two Polygons

Assume that we are given two convex polygons B_1, B_2 whose insides and boundaries are not intersecting. We are going to find rays which are tangents on both polygons. Because a ray can have an independent orientation on each polygon, there are four distinct orientations in all, $B_1^+ B_2^+$, $B_1^+ B_2^-$, $B_1^- B_2^+$, and $B_1^- B_2^-$ (see Figure13). If the direction of a tangent is reversed, both orientations are also flipped. For instance, when $B_1^+ B_2^+$ is considered a common tangent from B_2 to B_1 by changing its direction, that tangent becomes the $B_2^- B_1^-$ tangent. We consider a problem of finding the osculating point (or the first osculating point if there are two) v on B such that the ray pv is a μ tangent on B .

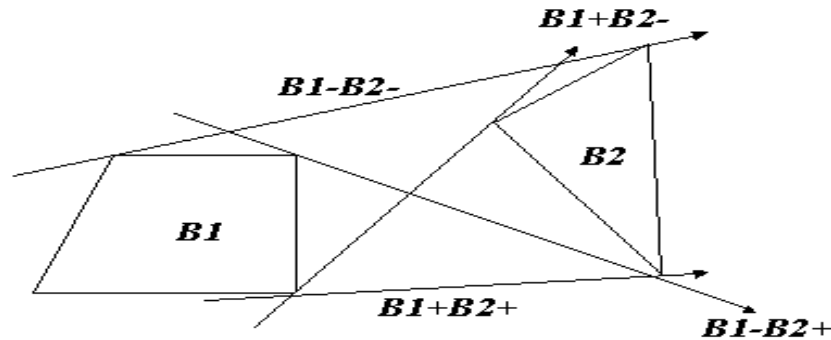


Figure 3.6 Tangents between two polygons

Let v_1, v_2 be vertices on polygons B_1 and B_2 respectively. If the ray $\overrightarrow{v_1 v_2}$ is a μ_1 tangent on B_1 and a μ_2 tangent on B_2 , we write $\text{Tangent}_{11}(B_1, v_1, \mu_1, B_2, v_2, \mu_2)$. We call the tangent a $\mu_1 \mu_2$ common tangent on B_1 and B_2 .

Proposition III.6 *For any polygons B_1, B_2 , vertices v_1, v_2 , and orientations μ_1 and μ_2 , if v_1 is in B_1 and v_2 is in B_2 ,*

$$\begin{aligned} \text{Tangent}_{11}(B_1, v_1, \mu_1, B_2, v_2, \mu_2) = \\ \text{Tangent}_{01}(v_1, B_2, \mu_2, v_2) \wedge \text{Tangent}_{10}(v_2, B_1, \mu_1, v_1). \end{aligned} \quad (\text{III.8})$$

The proposition states that the common tangent between B_1 and B_2 (Tangent_{11}) can be found by combining two simpler functions:

$\text{Tangent}_{01}(v_1, B_2, \mu_2, v_2)$: This function, as discussed earlier, finds a tangent line from vertex v_1 of B_1 towards polygon B_2 with orientation μ_2 , touching it at vertex v_2 .

$\text{Tangent}_{10}(v_2, B_1, \mu_1, v_1)$: This function (likely defined elsewhere) might find a tangent line from vertex v_2 of B_2 towards polygon B_1 with orientation μ_1 , touching it at vertex v_1 .

Proposition III.7 *For any polygons B_1, B_2 , vertices v_1, v_2 , and orientations μ_1 and μ_2 , if v_1 is in B_1 , v_2 is in B_2 , and the insides and boundaries of these polygons are not intersecting, there exists one and only one $\mu_1 \mu_2$ tangent on these polygons.*

The following algorithm finds a common tangent between two polygons.

This takes $O(|B_1| + |B_2|)$, since it searches the solution on B_1 and B_2 alternatively.

Tangent-between-two-polygons (B_1, μ_1, B_2, μ_2)

begin

$v_1 := \text{first_vertex}(B_1)$

$v_2 := \text{first_vertex}(B_2)$

do

```

    flag1 := flag2 := false
    if  $O(v_1, v_2, \phi^{\mu_2}_2(v_2)) = \mu_2$ 
        then if  $O(v_1, v_2, \phi^{\mu_2}_2(v_2)) \neq -\mu_2$ 
            then flag2 := true
            else  $v_2 := \phi^{\mu_2}_2(v_2)$ 
        else  $v_2 := \phi^{\mu_2}_2(v_2)$ 
    if  $O(v_2, v_1, \phi^{\mu_1}_1(v_1)) = -\mu_1$ 
        then if  $O(v_2, v_1, \phi^{\mu_1}_1(v_1)) \neq -\mu_1$ 
            then flag1 := true
            else if  $v_1 := \phi^{\mu_1}_1(v_1)$ 
        else  $v_1 := \phi^{\mu_1}_1(v_1)$ 
    until flag1 ^ flag2
    return (v1, v2)
end.

```

4.3 VISIBILITY

In a world W with n polygons, there is a possibility that some tangents are “occluded” by one or more polygons. Let p_1 and p_2 be two points in the free-space of W . The two points are said to be visible if there is no polygon $B \in W$ such that the segment p_1p_2 has an intersection with B ; otherwise they are said to be invisible (see Figure 14). Given a start S and goal G for a shortest path planning problem, we need to obtain three visible tangent sets: (i) The set of all visible tangents from S to polygons, (ii) the set of all visible from polygons to G , and (iii) the set of all visible common tangents among polygons. The visibility test is the most time-consuming subtask in the shortest path planning task.

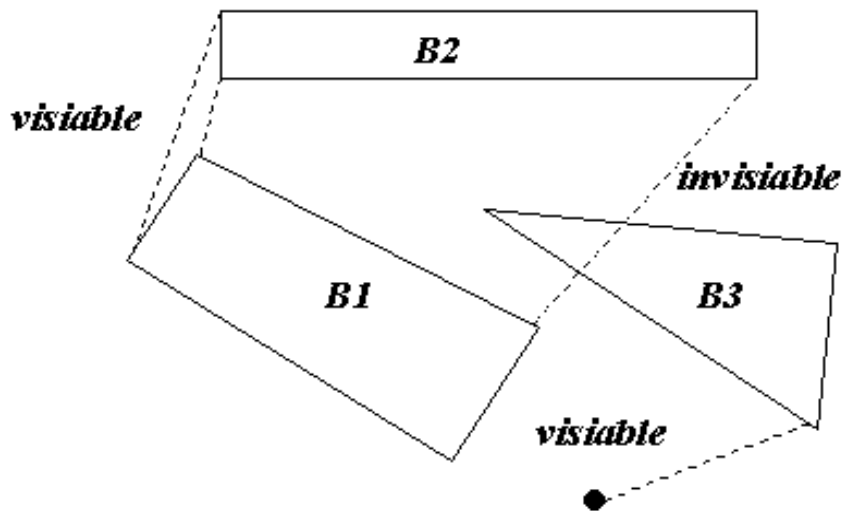


Figure 3.7 Visibility

First consider the problem of finding all visible tangents from one point S to all oriented polygons in W . we compute all visible and invisible tangents B^+ and B^- for each polygon B from S without visibility tests. We obtain the landing vertices v^+ , v^- , and their directions ψ^+ , and ψ^- for each polygon B . This computation takes $O(mn)$ time, where m is the maximum number of vertices in a polygon and n the number of polygons in W (see Figure 3.8)

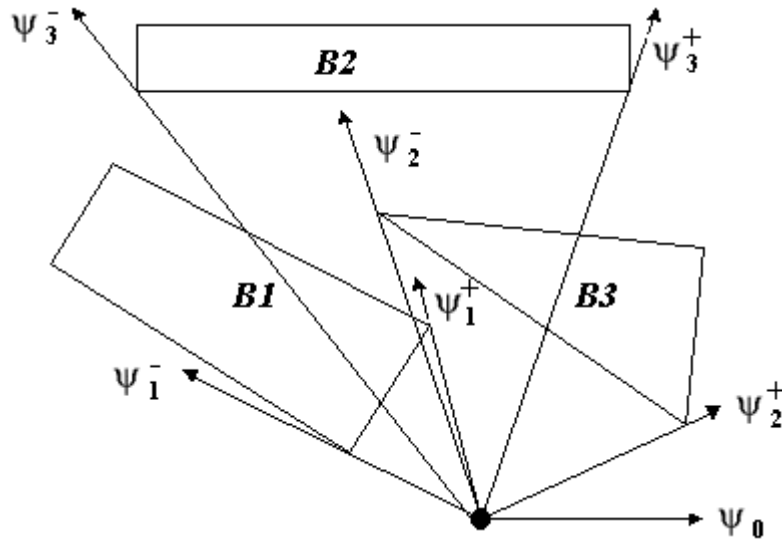


Figure 3.8 All tangents from S

After this preparatory step, we make an ordered list of tangents by its tangent directions ψ^u . The visibility of a tangent from S depends on the “distance” between the polygon and S . we sweep a ray with a direction ψ from S keeping the information of the current list of polygons intersecting the ray as well as the order about closeness. As the ray sweeps, the list is updated at the direction of each tangents. Each updating operation is an insertion or a deletion of a polygon. A tangent is visible only when there is no polygon which is more closely located than the current

one. This computation takes $O(n \log n)$ time. (This algorithm assumes that there is a direction ψ such that the ray (S, ψ_0) does not intersect any polygons. If there is no such ψ_0 , we need to slightly modify the algorithm.)

Visible_Tangents (W, S, G)

begin

1. Make a list L of tangents $B^\mu = (v^\mu, \psi^\mu)$ from S
For each polygon $B \in W$ and for $\mu = \pm 1$ (visible or invisible).
2. Find ψ_0 such that the ray (S, ψ_0) does not intersect any polygons.
3. Sort L by their directions ψ^μ based on ψ_0 .
4. Initialize a priority queue P .
5. For each tangent B^μ in L do
6. **If** $\mu = +1$
7. **then** Insert $B = (v^+, \psi^+, v^-, \psi^-)$ in P .
8. **if** the B is placed in the “closest” position in P ,
9. then Mark the B^+ as visible.
10. **Else** Mark the B^+ as invisible.
11. **Else** Find the corresponding B in P .
12. **if** the B is placed in the “closest” position in P .
13. **then** Mark the B^- as visible.
14. **Else** Mark B^- as invisible.
15. Delete the B from P .

end.

The “closeness” test actually should not be done by the simple length of each tangent, but the “orientation test” with the “diameter” v^+v^- of each polygon. The whole computation takes $O(mn + n \log n) = O(n(m + \log n))$.

If there is no ψ_0 such that the ray (S, ψ_0) intersects no polygons, we rotate the ray one complete turn to initialize P correctly, and the main processing is done at second complete turn. This two complete sweep task does not increase the computational complexity.

The visibility testing for tangents from polygon to a point G is done in a similar manner. The visibility testing for tangents among polygons will be done in this way:
(i) All tangents among polygons are obtained without visibility test. This computation takes $O(mn^2)$ time. (ii) Take one polygon B at one time. We will do a similar sweeping work to find visible common tangents. This task takes $O(n^2 \log n)$

time. By applying all these algorithms to the example world, we obtain a result as shown in Figure 3.9

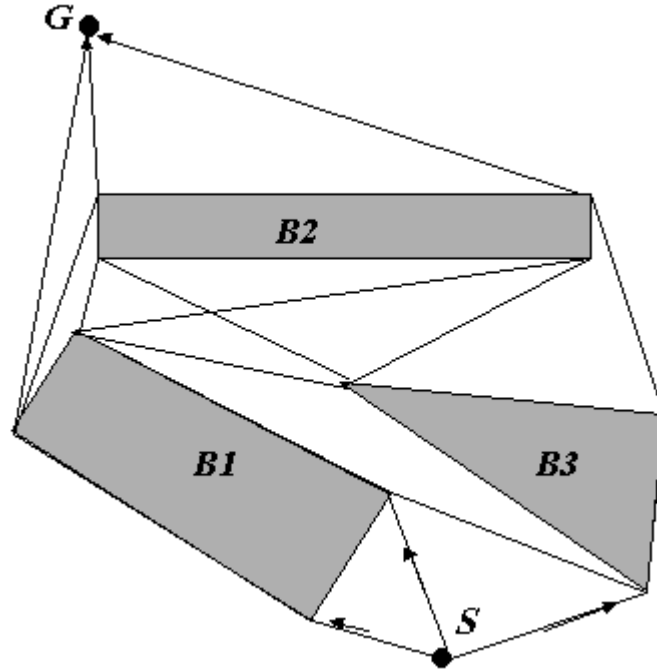


Figure 3.9 All visible tangents

4.4. SHORTEST PATH FINDING ALGORITHM

We will describe the top-level algorithm in this section. The sub problems of finding tangents and common tangents are discussed in earlier sections. Since a canonical path is represented by a tangent sequence, we consider a “graph” consisting of the set of “nodes”,

$$\{S, B_1^+, B_1^-, \dots, B_n^+, B_n^-, G\},$$

(III.9)

and the set of “oriented edges” which is the set of tangents. This observation justifies the use a variation of the Dijkstra’s algorithm for graph search in this theory. However, this algorithm is not as simple as the basic Dijkstra’s algorithm, because a oriented polygon (as a “node”) is not a point, but a polygon. For instance, the fact that a landing vertex to and a leaving vertex of a oriented polygon may not be the

same makes this planning task difficult. Before discussing the algorithm itself, we show a behavior of the algorithm for a world. The exact coordinates of two points and three convex polygons (See Figure 3.8).

$$\begin{aligned}
S &= (0, 0), \\
B_1 &= \{v_{11}, v_{12}, v_{13}, v_{14}\} = \{(-15, 7), (-3, 1), (-1, 5), (-7, 8)\}, \\
B_2 &= \{v_{21}, v_{22}, v_{23}\} = \{(-3, 8), (6, 1), (9, 6)\}, \\
B_3 &= \{v_{31}, v_{32}, v_{33}, v_{34}\} = \{(-10, 9), (8, 9), (8, 11), (-10, 11)\}, \\
G &= (-5, 16).
\end{aligned}$$

As the preprocessing, all visible tangents this world are first computed (see Figure 3.8). This problem is actually discussed in the previous section of this chapter.

For each oriented polygon z (S and G are also considered as special oriented polygons), we prepare the following data structure in order to run a generalized-Dijkstra's algorithm.

$$(z.mark, z.cost, z.land, z.prev, z.leav, z.area)$$

(III.10)

The meaning of each member is:

$z.mark$: =1 if $z.cost$ is confirmed as the smallest one, =0 otherwise.

$z.cost$: the smallest cost to z known so far.

$z.land$: the landing osculating vertex on z .

$z.prev$: the previous oriented polygon or point.

$z.leav$: the leaving vertex of the previous oriented polygon.

$z.area$: the partial area of the current partial to z .

Let us give an example behavior of the algorithm. We start from S to enlarge the domain of "marked" oriented polygon until G itself is marked. A star in the "mark" slot means that this oriented polygon was newly marked. The $z.area$ slot is not explicitly shown here.

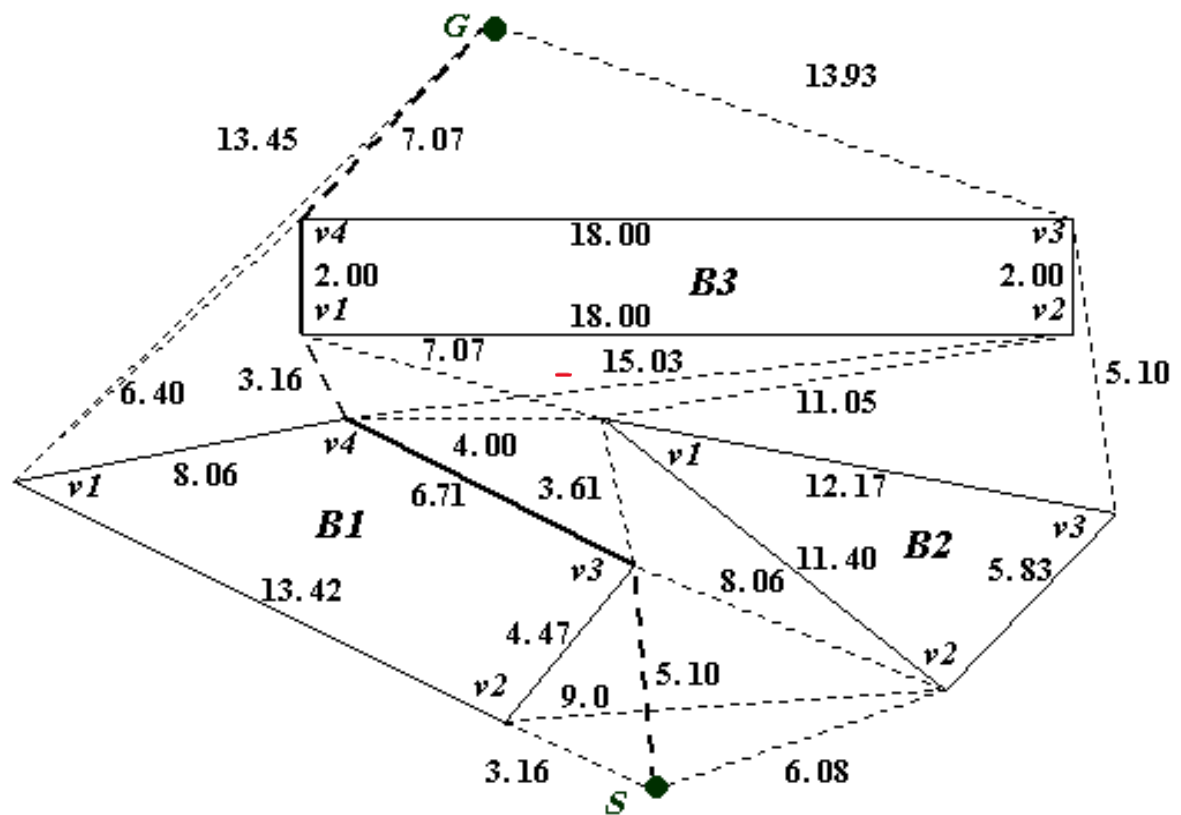


Figure 3.9 All visible tangents

STEP 0

z	mark	cost	land	Prev	leave
S	1*	0.00	S	-	-
B1+	0	∞	-	-	-
B1-	0	∞	-	-	-
B2+	0	∞	-	-	-
B2-	0	∞	-	-	-
B3+	0	∞	-	-	-
B3-	0	∞	-	-	-
G	0	∞	-	-	-

STEP 1

z	mark	cost	land	Prev	leave
S	1	0.00	S	-	-
B1+	0	5.10	V13	S	S
B1-	1*	3.16	V12	S	S
B2+	0	6.08	V22	S	S
B2-	0	∞	-	-	-
B3+	0	∞	-	-	-
B3-	0	∞	-	-	-
G	0	∞	-	-	-

STEP 2

z	mark	cost	land	Prev	leave
S	1	0.00	S	-	-
B1+	1*	5.10	V13	S	S
B1-	1	3.16	V12	S	S
B2+	0	6.08	V22	S	S
B2-	0	28.64	V21	B1-	V14
B3+	0	39.67	V32	B1-	V14
B3-	0	22.98	V34	B1-	V11
G	0	30.03	G	B1-	V11

STEP 3

z	mark	cost	land	Prev	leave
S	1	0.00	S	-	-
B1+	1	5.10	V13	S	S
B1-	1	3.16	V12	S	S
B2+	1*	6.08	V22	S	S
B2-	0	8.71	V21	B1+	V13
B3+	0	19.76	V32	B1-	V14
B3-	0	14.97	V31	B1+	V14
G	0	30.03	G	B1-	V11

STEP 4

<i>z</i>	mark	cost	land	Prev	leave
S	1	0.00	S	-	-
<i>B1+</i>	1	5.10	V13	S	S
<i>B1-</i>	1	3.16	V12	S	S
<i>B2+</i>	1	6.08	V22	S	S
<i>B2-</i>	1*	8.71	V21	B1+	V13
<i>B3+</i>	0	17.01	V33	B2+	V23
<i>B3-</i>	0	14.97	V31	B1+	V14
G	0	30.03	G	B1-	V11

STEP 5

<i>z</i>	mark	cost	land	Prev	leave
S	1	0.00	S	-	-
<i>B1+</i>	1	5.10	V13	S	S
<i>B1-</i>	1	3.16	V12	S	S
<i>B2+</i>	1	6.08	V22	S	S
<i>B2-</i>	1	8.71	V21	B1+	V13
<i>B3+</i>	0	17.01	V33	B2+	V23
<i>B3-</i>	1*	14.97	V31	B1+	V14
G	0	30.03	G	B1-	V11

STEP 6

<i>z</i>	mark	cost	land	Prev	leave
S	1	0.00	S	-	-
<i>B1+</i>	1	5.10	V13	S	S
<i>B1-</i>	1	3.16	V12	S	S
<i>B2+</i>	1	6.08	V22	S	S
<i>B2-</i>	1	8.71	V21	B1+	V13
<i>B3+</i>	1*	17.01	V33	B2+	V23
<i>B3-</i>	1	14.97	V31	B1+	V14
G	0	30.03	G	B1-	V11

STEP 7

z	mark	cost	land	Prev	leave
S	1	0.00	S	-	-
B1+	1	5.10	V13	S	S
B1-	1	3.16	V12	S	S
B2+	1	6.08	V22	S	S
B2-	1	8.71	V21	B1+	V13
B3+	1	17.01	V33	B2+	V23
B3-	1	14.97	V31	B1+	V14
G	1*	24.04	G	B3-	V34

By this result, the shortest path obtained is

$$G \leftarrow B3^- (v34) \leftarrow B3^- (v31) \leftarrow B1^+ (v14) \leftarrow B1^+ (v13) \leftarrow S$$

(III.11)

And its tangent sequence is B1+ B3-. Now a formal algorithm is described here (the variable δ in line 14 will be described in the next sub section).

Find_Shortest_Path (W, S, G)

begin

1. Find all visible tangents from S
2. Find all visible tangents from each oriented polygon
3. **Forall** $z \in Z \cup \{S, G\}$ **do**
4. Z.mark :=0;
5. z.cost :=0;
6. z :=S;
7. z..cost :=0;
8. z.land :=0;
9. z.area :=0;
10. **doforever**
11. z.mark :=1;
12. if z = G then return
13. forall z' visible from z with leaving vertex v_0 on z and landing vertex v_1 on z' do
14. $w := z.cost + \lg (z(z.land, v_0)) + d (v_0, v_1);$
15. $a_0 := z.area + D (z.land, v_0) + D (v_0, v_1);$
16. $a_1 := z'.area + D (z'.area, v_1) - a_0;$
17. **if** ($z'.cost = \infty$) **V**

```

(sign (O(z')) = sign (a1) ^ w + lg (z' (v1, z'.land)) < z'.cost) V (sign (O (z')) ≠ sign
(a1) ^ w - lg (z' (z'.land, v1)) < z'.cost)
      then
18.          z'.cost := w;
19.          z'.land := v1;
20.          z'.prev := z;
21.          z'.leave := v0;
22.          z'.area := a0;
23.          z := the unmarked oriented polygon with the minimum z.cost;
      end.

```

Here $lg(z(u, u'))$ the z-orientation boundary length of z polygon from u to u' assuming u and u' vertices of a z-polygon.

3.4.1 Comparison between Two Paths

The complex test in line 17 in the previous algorithm is related to the fact that there might be two paths reaching a same oriented polygon z' but at distinct landing vertices. Specifically, consider the situation depicted in Figure III.11. In this situation, the cost to z has been just established and there is a visible tangent v0 v1 from z to z', where a path to z' was already found with a landing vertex v'.land. Therefore, we must evaluate which one (the newly found one through z or the old one) is better. First task is to evaluate the new cost w to z' (at the new landing vertex v1) and the new partial area a0.

$$w = z.cost + lg(z(z.land, v0)) + d(v0, v1) \quad (III.12)$$

$$a0 = z.area + D(z.land, v0) + D(v0, v1) \quad (III.13)$$

In case (I) (Figure 3.10), the current cost z'.cost and $w + lg(z'(v1, v'.land))$ must be compared. However, in case (II), cost z'.cost and $w - lg(z'(v'.land, v1))$ should be compared. If the former one is smaller in both cases, the new path is better, and we must replace the old one by the new one.

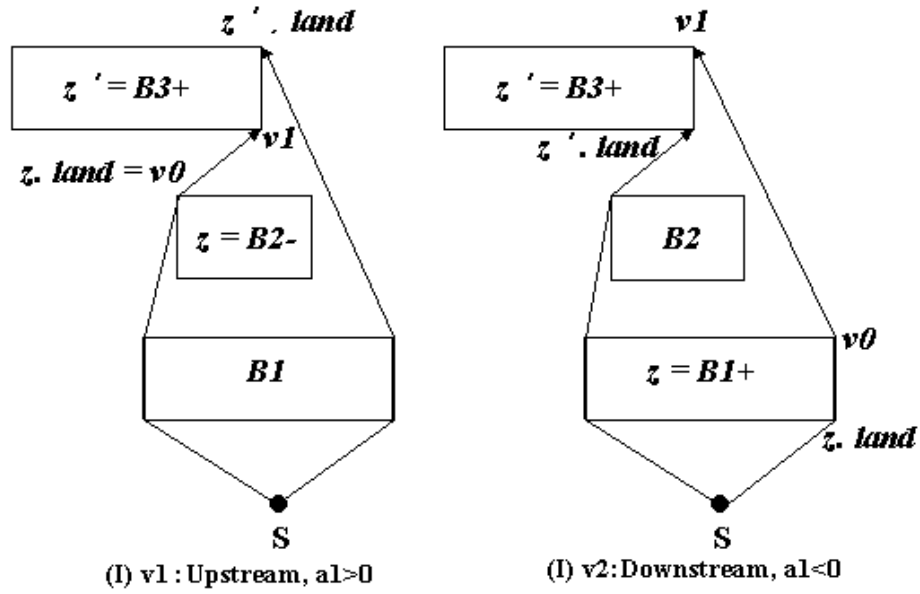


Figure 3.10. Comparison of paths

The next question is “which case the current situation falls into?” the best way is to compute the orientation of the polygon formed by two paths and the line segment $v1, z'.land$ by computing its area. We compute the signed area

$$A1 = z'.area + D(z'.land, v1) - a0.$$

(III.13)

Through testing of the sign of $a1$, we can correctly compare the old one and new path as shown in the previous program. We need a similar one like Figure III.11 with $O(z') = -1$, but it is omitted here.

CHAPTER.5

CONCLUSION

This project addressed new algorithm in finding the shortest path for point robot in a world of convex polygons (which represent the projection of the obstacles in the 2-D). This shortest path algorithm was very successful on all of the problems attempted. This algorithm has been widely used because it does determine the existence of a free path and because it finds the shortest path if one is available.

Significant features of this method are:

1. A variation of the Dijkstra's algorithm is used.
2. A directed polygon is the unit of the searching process instead of an edge. Since the number of directed polygons is $2n$, this greatly reduces the computational complexity of this task as opposed to the known results.
3. As a part of item 2, an algorithm for comparing all paths reaching to the same directed polygon is established.

REFERENCES

- [1] T. Lozano-Perez and M.A. Wasely. An Algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560-570, Oct. 1979.
- [2] R.A. Brooks. Solving the find-path problem by good representation of free space. *IEEE transactions on systems, Man, and Cybernetics*, SMC-13(2):190-197, Mar./Apr. 1983.
- [3] B.R. Donald. *Motion Planning With Six Degrees of Freedom*. Technical Report AIM-791, MIT Artificial Intelligence Laboratory, 1984.
- [4] R.A. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for find path with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(2):225-233, Mar./Apr. 1985.
- [5] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transaction of Systems Science and Cybernetics*, SSC 4(2): 100-107, July 1968.
- [6] O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90-98, Spring 1986.
- [7] C.W. Warren, J.C. Danos, and B.W. Mooring. An approach to manipulator path planning. *The International Journal of Robotics Research*, 8(5):87-95, October 1989.
- [8] C.W. Warren. A vector based approach to path planning. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1021-1026, Sacramento, CA, 1991.
- [9] M. Erdmann and T. Lozano-Perez. On multiple moving obstacles. In proceedings of the *The 1986 IEEE International Conference on Robotics and Automation*, pages 1419-1424, San Francisco, CA 1986.
- [10] V.J. Lumelsky. Continuous motion planning in unknown environment for a 3D Cartesian robot arm. In *Proceedings of the The 1986 IEEE International Conference on Robotics and Automation*, pages 1050-1055, San Francisco, CA, 1986.
- [11] D. Zwilling, *Statndard Mathematical Tables and Formulae*, CRC Press 1996

