2
Bug Algorithms

⇒ Bug1 and Bug2 algorithm are among the earliest and simplest sensor-based planners.

⇒ These algorithms assume:
- → Robot is a point ~~operating~~
- → Operating in a plane
- → with a contact sensor to detect obstacles.

⇒ When the robot has a finite range sensor, then the Tangent Bug Algorithm is a Bug derivative that can use the sensor information to find shorter path to the goal.

⇒ These algorithm require two behaviors:
- → Move on a straight line
- → Follow a boundary.

{ Curve-tracing technique based on the implicit function theorem }

⇒ there success is guaranteed, when possible.

1) Bug 1 and Bug 2

⇒ Robot is assumed to have perfect positioning (no positioning error).

⇒ Workspace is assumed to be bounded.

⇒ The robot can measure distance $d(x,y)$ between any two points x and y.

⇒ Let $B_r(x)$ denote a ball of radius r centered on x.

$$B_r(x) = \{ y \in \mathbb{R}^2 \mid d(x,y) < r \}$$

⇒ Workspace is bounded implies that $\forall x \in W$, there exists an $r < \infty$ such that $W \subset B_r(x)$.

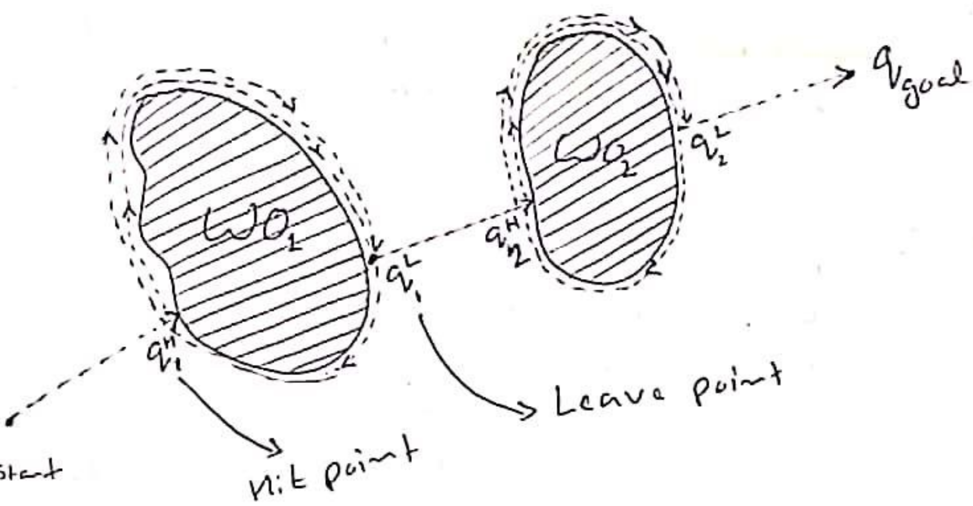
q_{start} ⇒ Starting point position

q_{goal} ⇒ Goal point position

⇒ Let $q_0^L = q_{start}$ &

⇒ m-line be the line segment that connects q_i^L to q_{goal}.

⇒



⇒ During motion to goal, the robot moves along the m-line toward q_{goal} until it either encounters the goal or an obstacle.

⇒ If robot encounters an obstacle, the q_1^H be the point where the robot first encounters an obstacle and call this point a hit point.

⇒ The robot then circumnavigate the obstacle until it returns to q_1^H.

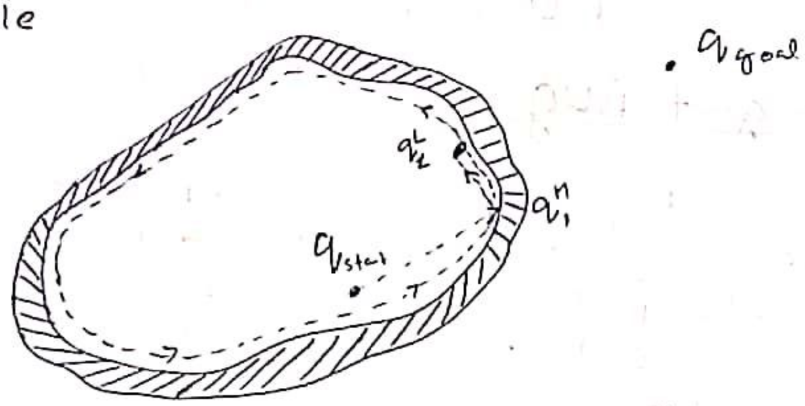
⇒ Then, the robot determines the closest point to the goal on the perimeter of the obstacle and traverses to this point.

Algorithm 1: Bug1 Algorithm

Input: A point robot with a tactile sensor

Output: A path to the q_{goal} or a conclusion no such path exists

```
while Forever do
    repeat
        From q^L_{i-1}, move toward q_goal.
    until q_goal is reached or an obstacle is encountered at q_i^H
    if Goal is reached then
        Exit
    end if
    repeat
        Follow the obstacle boundary
    until q_goal is reached or q_i^H is re-encountered.
    Determine the point q_i^L on the perimeter that has the shortest distance to the goal.
    Go to q_i^L
    if the robot were to move toward the goal then
        Conclude q_goal is not reachable and exit
    end if
end while
```



{ Bug1 algorithm will report goal is unreachable }

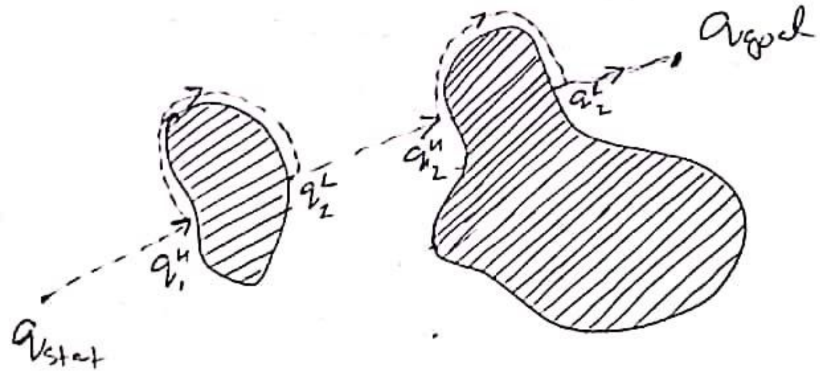
⇒ Bug2 algorithm is similar to Bug1 algorithm except m-line connect q_{start} and q_{goal}, and thus remain fixed.

⇒ BUG2 algorithm is also called greedy, since it opts for the first promising option that is found.

⇒ Bug1 algorithm performs an exhaustive search to find the optimal leave point.

Bug1 → Performance is good when obstacle is complex

Bug2 → Performance is good when obstacle is simple



BUG2 Algorithm

2) Tangent BUG

⇒ Tangent Bug serves as an improvement to the BUG2 algorithm in that it determines a shorter path to the goal using a range sensor with 360 degree infinite orientation resolution.

⇒ We model this range sensor with the raw distance function $f: \mathbb{R}^2 \times S^1 \rightarrow \mathbb{R}$

⇒ The value $f(x, \theta)$ is the distance to the closest obstacle along the ray from x at an angle θ.

$$\rho(x,\theta) = \min_{\lambda \in [0,\infty]} d(x, x + \lambda[\cos\theta, \sin\theta]^T),$$

such that $x + \lambda[\cos\theta, \sin\theta] \in \bigcup_i WO_i$

⇒ Since real sensors have limited range, we define the saturated raw distance function, denoted $\rho_R : \mathbb{R}^2 \times S^1 \to \mathbb{R}$, which takes on the same values as ρ when the obstacle is within sensing range, and has a value of infinity when the ray lengths are greater than the sensing range, R.

$$\rho_R(x,\theta) = \begin{cases} \rho(x,\theta), & \text{if } \rho(x,\theta) < R \\ \infty, & \text{otherwise} \end{cases}$$

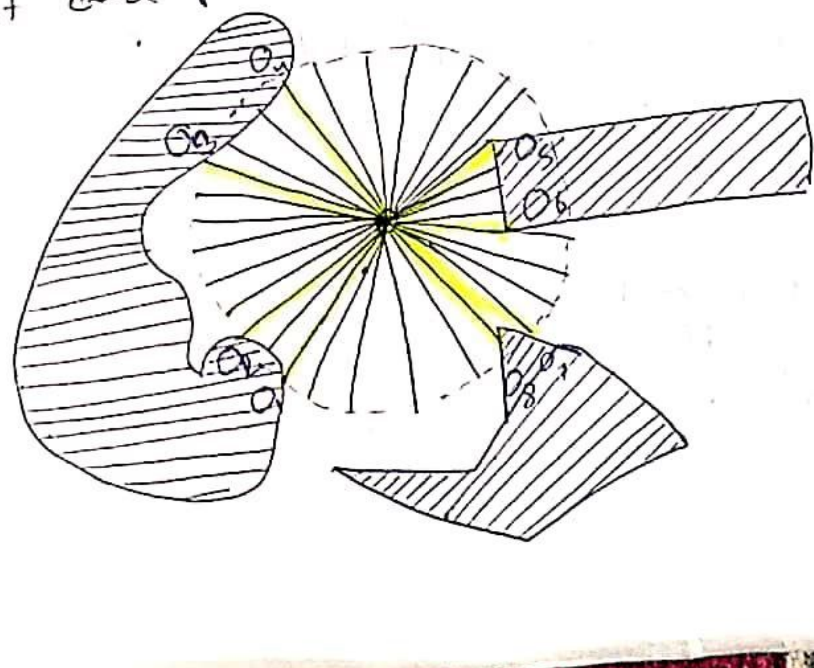
⇒ The Tangent Bug planner assumes that the robot can detect discontinuities in ρ_R.

⇒ For a fixed $x \in \mathbb{R}^2$, an interval of continuity is defined to be a connected set of points

$x + \rho(x,\theta)[\cos\theta, \sin\theta]^T$

finite

⇒ Let end point be denoted by O_i.



⇒ Just like the other Bugs, Tangent Bug iterates between two behaviours:

- motion-to-goal
- Boundary following.

⇒ First the robot moves in a straight line toward the goal until it senses an obstacle R units away and directly between it and the goal.

⇒ When the robot initially senses an obstacle, the circle of radius R becomes tangent to the obstacle.

⇒ Immediately after this, tangent point splits into two O_i's which are the endpoints of the interval.

⇒ The robot then moves toward the O_i that ~~maximizes~~ minimizes a heuristic distance to the goal.

$$\left(d(x, O_i) + d(O_i, q_{goal})\right)$$

{Can be more complicated when factoring in available info with regard to the obstacles}

⇒ When the robot switches to Boundary-following, it finds a point M on the sensed portion of the obstacle which has the shortest distance on the obstacle to the goal.

⇒ If Sensor range is 0, then M is the same as the hit point from the Bug1 and Bug2 Algorithm.

followed obstacle

Sensed Obstacle is called followed obstacle

blocking Obstacle

Closest obstacle within Sensor range that intersects the segment. $(1-\lambda)x + \lambda q_{goal} \ \forall \lambda \in [0,1]$

⇒ Initially, the blocking Obstacle and the followed obstacle are the Same.

⇒ Now the robot moves in the Same direction as if it were in the motion-to-goal behavior.

⇒ While undergoing this motion, the planner also updates two values: $d_{followed}$ and d_{reach}

d_{reach}: Distance between the goal and the closest point on the followed obstacle that is within line of sight of the robot

$d_{followed}$: Shortest distance between the boundary which had been sensed and the goal.

⇒ Let Λ be all the points within line of Sight of x with range R that are on the followed obstacle WO_f.

$$\Lambda = \{ y \in \partial WO_f : \lambda x + (1-\lambda) y \in Q_{free} \ \forall \lambda \in [0,1] \}$$

$$d_{reach} = \min_{c \in \Lambda} d(q_{goal}, c)$$

⇒ When $d_{reach} < d_{followed}$, the robot terminates the boundary-following behavior.

⇒ Let T be the point where a circle centered at x of radius R intersects the segment that connects x and q_{goal}.

Tangent Bug Algorithm

Input: A point robot with range sensor.
Output: A path to the q_{goal} or a conclusion no such path exists.

```
while True do
   repeat
      Continuously move toward the point n ∈ {T, O_i}
      which minimizes d(x, n) + d(n, q_goal)
   Until
      ▪ the goal is encountered or
      ▪ The direction that minimizes d(x, n)
   begins to increase d(x, q_goal) i.e. the robot
      detects a "local minimum" of d(·, q_goal)

   Chose a boundary following direction which
   continues in the same direction as the most
   recent motion-to-goal direction.
   repeat
      Continuously update d_reach, d_followed & {O_i}
      Continuously move toward n ∈ {O_i} that
      is in the chosen boundary direction.
   Until
      ▪ The goal is reached
      ▪ The robot completes a cycle around the obstacle
        in which case the goal cannot be achieved.
      ▪ d_reach < d_followed
end while
```

3) Implementation

⇒ If the obstacle were flat, such as a long wall in a corridor, then following the obstacle is trivial.

↳ Simply move parallel to the obstacle

⇒ Driving toward a point is simple gradient descent.