Largest Empty Circle Algorithm for Stationary Obstacles Using Voronoi Diagrams

Theoretical Description

The **Largest Empty Circle (LEC)** problem, in the presence of **stationary obstacles**, extends the classical LEC problem defined on a set of points $P = \{p_1, p_2, \ldots, p_n\}$ by considering additional fixed obstacles that the empty circle must avoid.

Formally, given:

- A set of points P in the plane (called *sites*).
- A set of stationary obstacles $O = \{o_1, o_2, \ldots, o_m\}$, each with known geometry (e.g., disks with fixed radius or polygonal shapes).

The goal is to find the *largest circle* centered inside the **convex hull** of P such that:

- The circle contains no points of P in its interior.
- The circle does not intersect any obstacle in O.

The Voronoi diagram of the sites P plays a central role in solving the LEC problem because:

- The centers of candidate largest empty circles lie on the **Voronoi vertices** or on the intersection of **Voronoi edges** with the convex hull boundary.
- Voronoi vertices are equidistant to three or more sites and correspond to local maxima of the minimum distance to the sites.
- The presence of stationary obstacles imposes additional constraints, restricting feasible circle centers to locations that maintain clearance from obstacles.

Thus, the LEC with stationary obstacles is found by searching among Voronoi vertices and Voronoi edge–convex hull intersections that also satisfy obstacle clearance constraints.

Algorithm

Input

- Sites $P = \{p_1, \ldots, p_n\}$.
- Stationary obstacles $O = \{o_1, \ldots, o_m\}$ with known geometry.

Output

- Center c^* and radius r^* of the largest empty circle avoiding all sites and obstacles.

Steps

1. Compute the **Voronoi diagram** $V(P)$ of the sites P.
2. Compute the **convex hull** $CH(P)$ of the sites.
3. Initialize an empty candidate set C.
4. For each **Voronoi vertex** v:
   - Check if v lies inside $CH(P)$.
   - Check if a circle centered at v with radius equal to the distance to the nearest site avoids all obstacles.
   - If yes, add v to C.
5. For each **Voronoi edge** e:
   - Compute intersections of e with edges of $CH(P)$.
   - For each intersection point x, check obstacle clearance as above.
   - If feasible, add x to C.
6. For each candidate center $c \in C$:
   - Compute radius $r_c = \min\left(\min_{p \in P} \|c - p\|, \min_{o \in O} \text{clearance}(c, o)\right)$, where $\text{clearance}(c, o)$ is the minimum distance from c to the boundary of obstacle o.
7. Select the candidate c^* with the maximum radius r^*.
8. Return (c^*, r^*).

Pseudocode

```
function LargestEmptyCircleWithObstacles(P, O):
    V = VoronoiDiagram(P)
    CH = ConvexHull(P)
    candidates = []

    for vertex v in V.vertices:
        if v inside CH and circle centered at v avoids obstacles O:
            candidates.append(v)

    for edge e in V.edges:
        intersections = intersect(e, CH.edges)
        for point x in intersections:
            if circle centered at x avoids obstacles O:
                candidates.append(x)

    max_radius = 0
    best_center = None

    for c in candidates:
        r_sites = min_distance(c, P)
        r_obs = min_clearance(c, O)
        r = min(r_sites, r_obs)
        if r > max_radius:
            max_radius = r
```

```
            best_center = c

    return best_center, max_radius
```

Complexity Analysis

- **Voronoi diagram construction:** $O(n \log n)$ time using Fortune's algorithm or other efficient methods.
- **Convex hull computation:** $O(n \log n)$ time (e.g., Graham scan).
- **Candidate identification:** There are $O(n)$ Voronoi vertices and edges. Checking each vertex and edge intersection against the convex hull can be done in $O(nh)$, where h is the hull size.
- **Obstacle clearance checks:** For each candidate, clearance must be verified against all obstacles. If m is the number of obstacles, naive checking costs $O(m)$ per candidate, leading to $O(nm)$ in total.
- **Overall complexity:** Dominated by $O(n \log n + nm)$ in the worst case. Spatial data structures (e.g., bounding volume hierarchies) can accelerate obstacle clearance checks in practice.

Remarks

- The presence of stationary obstacles restricts feasible circle centers, requiring additional geometric checks beyond classical LEC.
- The algorithm relies on the geometric duality between Voronoi diagrams and Delaunay triangulations to limit candidate centers.
- Efficient spatial indexing and pruning are recommended to handle large obstacle sets.
- This approach is widely used in robotics, facility location, and motion planning where obstacles are static and known.