Dynamic Optimized LEC Algorithm (DO-LEC): Efficient Largest Empty Circle Computation for Dynamic Obstacles

Overview

The **Dynamic Optimized LEC Algorithm (DO-LEC)** efficiently computes the Largest Empty Circle (LEC) that avoids all Voronoi sites and dynamically moving obstacles within a bounded region. This algorithm is implemented in the function `compute_dynamic_optimized_lec` and is the first practical, scalable solution for LEC computation in the presence of dynamic obstacles.

Key Features

- **Interactive GUI:** Add, select, and manage Voronoi sites and obstacles via mouse clicks and control buttons.
- **Dynamic Obstacles:** Define obstacle paths and animate their movement at adjustable speeds.
- **Visualization Options:** Toggle overlays for Voronoi edges, convex hull, and LEC.
- **Optimized LEC Computation:** Efficiently computes the largest empty circle avoiding all sites and current obstacle positions.
- **Statistics Panel:** Real-time display of site and obstacle counts, and animation time.
- **Clear and Reset:** Easily clear all data or reset animation time.

Algorithm: Dynamic Optimized LEC (DO-LEC)

Input

- Set of Voronoi sites $S = \{s_1, s_2, \ldots, s_n\}$.
- Set of dynamic obstacles $O = \{o_1, o_2, \ldots, o_m\}$, each with a path, radius, and current position.
- Bounding rectangle for the workspace.

Output

- Center and radius of the largest empty circle avoiding all sites, obstacles, and boundaries.

Steps

1. **Candidate Generation ($O(n \log n)$):**
   - Compute Delaunay triangulation; use circumcenters of triangles as LEC center candidates.
   - Add boundary points (corners, midpoints).
   - Compute intersections of perpendicular bisectors between sites and the bounding box.

2. **Candidate Evaluation ($O(nk)$):**
    - For each candidate, compute the minimum distance to all sites, all obstacles (accounting for their radii), and the boundary.
    - The candidate with the largest such minimum is selected as the LEC center.
    - Early termination and duplicate removal are used for efficiency.

Time Complexity Summary

Algorithm/Function	**Time Complexity**	**Description/Remarks**
`compute_voronoi` (scipy)	$O(n \log n)$	Fortune's algorithm via `scipy.spatial.Voronoi` for planar diagrams.
`compute_convex_hull`	$O(n \log n)$	Graham scan; optimal for 2D convex hull.
`get_current_obstacle_positions`	$O(m)$	Linear in the number of obstacles; path interpolation per obstacle.
`compute_dynamic_optimized_lec`	$O(n \log n + nk)$	Delaunay triangulation (incremental or sampled), boundary/bisector checks; k is a small constant (e.g., 8). Much faster than previous $O(n^2)$ or $O(n^3)$ approaches.
`draw`	$O(n + m)$	Iterates through all sites and obstacles for plotting; dominated by subroutine complexities.
`on_canvas_click`	$O(1)$	Constant time to add a site or obstacle waypoint.
`update_stats`	$O(1)$	Simple string formatting and label update.
`clear_all`/`reset_time`	$O(1)$	Resets state variables; no iteration over data.

Where:

- n: Number of sites
- m: Number of obstacles
- k: Number of nearest neighbors (constant, e.g., 8)

Core Efficiency and Pioneering Aspects

- **Scalability:** The optimized LEC computation is significantly more scalable and practical for large numbers of sites compared to brute-force methods ($O(n^2)$ or $O(n^3)$).
- **Real-Time Dynamic Obstacle Support:** DO-LEC recomputes the LEC as obstacles move, making it the first efficient, interactive method for LEC under dynamic constraints—a field not previously addressed in depth in the literature.
- **Practical Performance:** For dozens to hundreds of sites and obstacles, all algorithms perform efficiently in practice.
- **Extensibility:** For very large datasets, spatial partitioning (e.g., k-d trees) can further accelerate nearest-neighbor queries in LEC computation.
- **Novelty:** No prior published algorithm efficiently computes the LEC in the presence of dynamic, moving obstacles.
- **Optimality:** By leveraging geometric duality (Voronoi/Delaunay) and spatial pruning, DO-LEC achieves near-optimal performance for real-world use cases.