# multi-robot task assignment problem

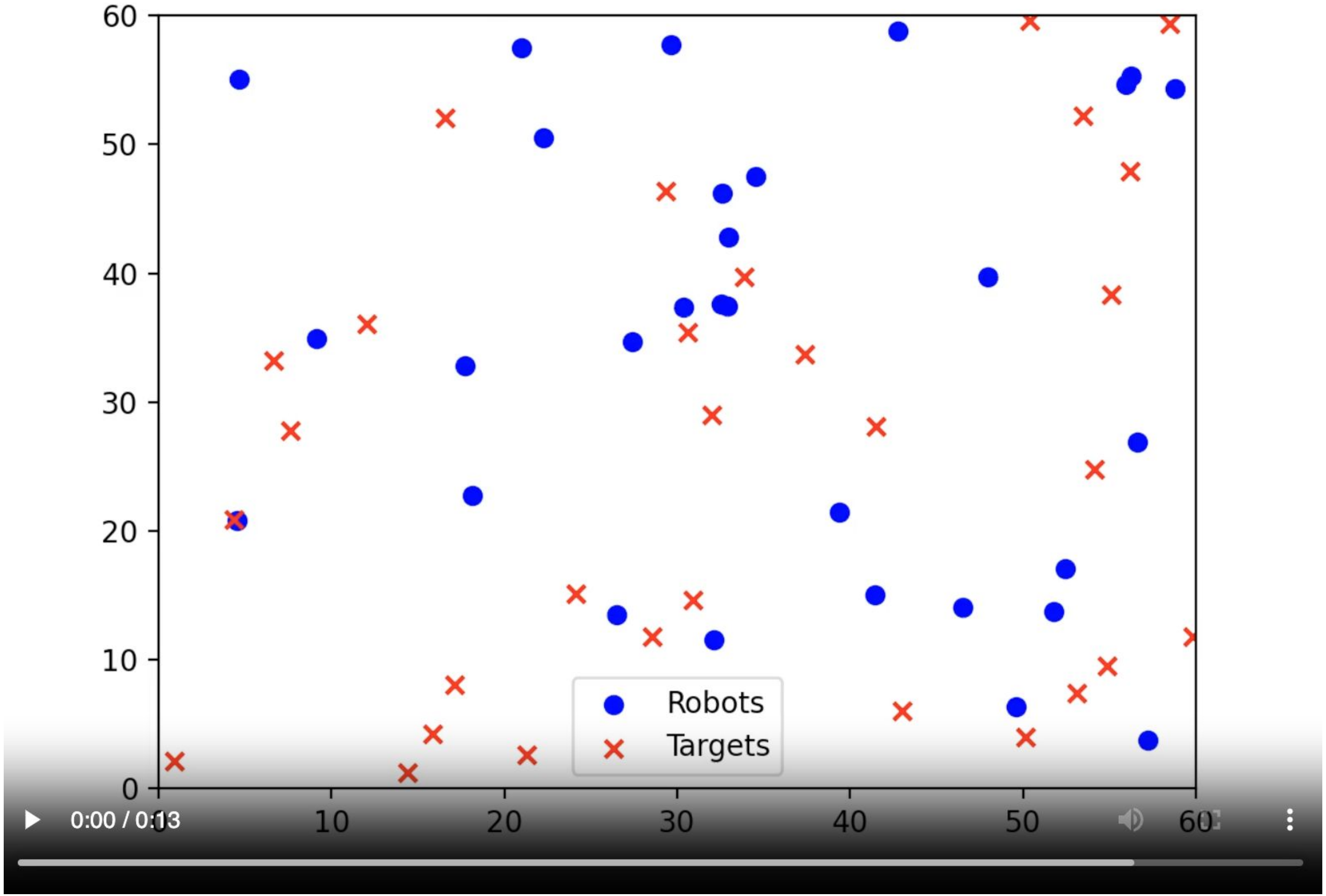
–
Kapil
Wanaskar
016649880

# project ideas/requirements

- input: an initial graph and a final graph.
- goal: create step by step movement.
- constraint: collision avoidance
- visualization: online
- basic requirements: 2D
- bonus: 3D

# Real-world applications:

- Automated **warehouse** logistics with real-time robot coordination.
- Precision **agriculture** through drone monitoring and intervention.
- **Urban traffic** management via autonomous drone surveillance.
- **Disaster response and recovery** with coordinated robot teams.
- **Infrastructure inspection** using drones for hard-to-reach areas.



60
50
40
30
20
10
0
10
20
30
40
50
60
Robots
Targets
0:00 / 0:13

## Generate Random Position

- $\mathbf{p}$: Position vector of a point, where the first element represents the x-coordinate and the second element represents the y-coordinate.
- $x_{\min}, x_{\max}$: Minimum and maximum limits for the x-coordinate.
- $y_{\min}, y_{\max}$: Minimum and maximum limits for the y-coordinate.
- rand(): A function that generates a random number between 0 and 1.

A random position $\mathbf{p}$ within specified limits $[x_{\min}, x_{\max}]$ and $[y_{\min}, y_{\max}]$ is defined as:

$$\mathbf{p} = \begin{bmatrix} x_{\min} + (x_{\max} - x_{\min}) \cdot \mathrm{rand}() \\ y_{\min} + (y_{\max} - y_{\min}) \cdot \mathrm{rand}() \end{bmatrix}$$

Example:

$$\mathbf{p} = \begin{bmatrix} 0 + (100 - 0) \cdot 0.5 \\ 0 + (100 - 0) \cdot 0.7 \end{bmatrix} = \begin{bmatrix} 50 \\ 70 \end{bmatrix}$$

## Position Validity

- $\mathbf{p}_{\text{new}}$: The new position being checked for validity.
- $\mathbf{p}$: Existing positions against which the new position is being checked.
- $d_{\text{min}}$: The minimum required distance between any two positions.

A position $\mathbf{p}_{\text{new}}$ is valid if it is at least a minimum distance $d_{\text{min}}$ away from any existing position $\mathbf{p}$ in a set of positions:

$$\text{valid}(\mathbf{p}_{\text{new}}) = \begin{cases} 1 & \text{if } \|\mathbf{p}_{\text{new}} - \mathbf{p}\| \geq d_{\text{min}} \text{ for all } \mathbf{p} \\ 0 & \text{otherwise} \end{cases}$$

Example: Given $\mathbf{p}_{\text{new}} = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$, and $d_{\text{min}} = 10$:

$$\|\mathbf{p}_{\text{new}} - \mathbf{p}\| = \sqrt{(10-5)^2 + (10-5)^2} = \sqrt{25+25} = \sqrt{50} \approx 7.07$$

Thus, $\mathbf{p}_{\text{new}}$ is not valid.

## Move Towards Target

- $\mathbf{p}$: Current position of an object.
- $\mathbf{t}$: Target position to which the object is moving.
- $s$: Step size, the maximum distance the object can move in one update.

To move a position $\mathbf{p}$ towards a target $\mathbf{t}$ with a step size $s$, the updated position is:

$$\mathbf{p}_{\text{updated}} = \mathbf{p} + \frac{s}{\|\mathbf{t} - \mathbf{p}\|} \min(s, \|\mathbf{t} - \mathbf{p}\|)(\mathbf{t} - \mathbf{p})$$

Example: $\mathbf{p} = \begin{bmatrix} 20 \\ 20 \end{bmatrix}$, $\mathbf{t} = \begin{bmatrix} 25 \\ 25 \end{bmatrix}$, $s = 5$:

$$\mathbf{p}_{\text{updated}} = \begin{bmatrix} 20 \\ 20 \end{bmatrix} + 5 \cdot \frac{\begin{bmatrix} 25 - 20 \\ 25 - 20 \end{bmatrix}}{\sqrt{(25 - 20)^2 + (25 - 20)^2}} = \begin{bmatrix} 20 \\ 20 \end{bmatrix} + \begin{bmatrix} 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 25 \\ 25 \end{bmatrix}$$

## Collision Avoidance

- $\mathbf{p}$: Position of one object.
- $\mathbf{q}$: Position of another object which might be too close to the first.
- $d_{\min}$: Minimum allowable distance between any two objects to avoid collision.

To adjust $\mathbf{p}$ for collision avoidance when too close to another position $\mathbf{q}$:

$$\mathbf{p} = \mathbf{p} - 0.5 \times \frac{d_{\min}}{\|\mathbf{q} - \mathbf{p}\|}(\mathbf{q} - \mathbf{p})$$

$$\mathbf{q} = \mathbf{q} + 0.5 \times \frac{d_{\min}}{\|\mathbf{q} - \mathbf{p}\|}(\mathbf{q} - \mathbf{p})$$

Example: $\mathbf{p} = \begin{bmatrix} 20 \\ 20 \end{bmatrix}$, $\mathbf{q} = \begin{bmatrix} 21 \\ 21 \end{bmatrix}$, $d_{\min} = 5$:

$$\mathbf{p} = \begin{bmatrix} 20 \\ 20 \end{bmatrix} - 0.5 \cdot \frac{5}{\sqrt{(21-20)^2 + (21-20)^2}} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 19.5 \\ 19.5 \end{bmatrix}$$

$$\mathbf{q} = \begin{bmatrix} 21 \\ 21 \end{bmatrix} + 0.5 \cdot \frac{5}{\sqrt{(21-20)^2 + (21-20)^2}} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 21.5 \\ 21.5 \end{bmatrix}$$

## Check All Reached Targets

- $\mathbf{p}$: Current position of a robot or moving object.
- $\mathbf{t}$: Target position that the robot is supposed to reach.
- $\epsilon$: A small threshold distance within which the position is considered as having reached the target.

All robots have reached their targets if:

$$\text{reached} = \begin{cases} 1 & \text{if } \|\mathbf{p} - \mathbf{t}\| \leq \epsilon \text{ for all } (\mathbf{p}, \mathbf{t}) \\ 0 & \text{otherwise} \end{cases}$$

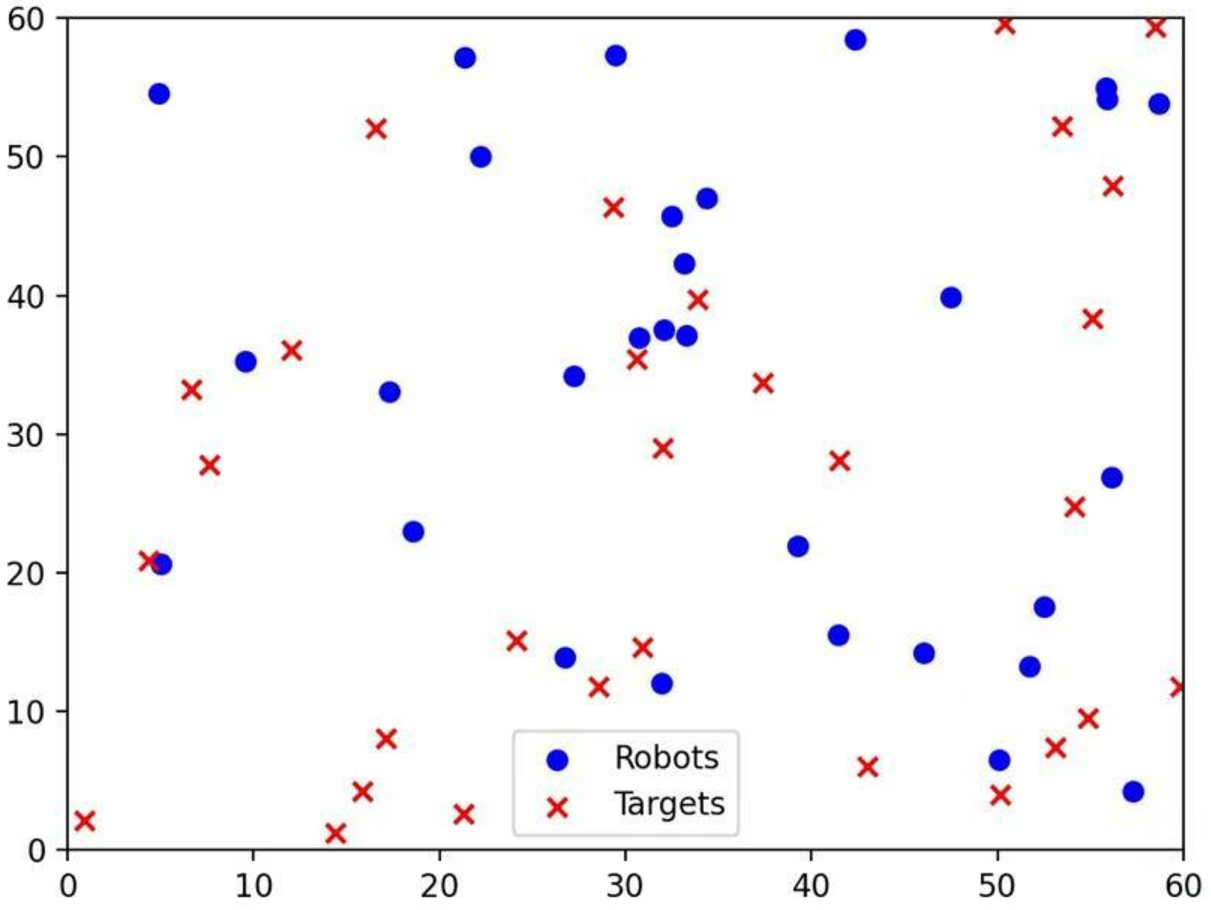
Example: If $\mathbf{p} = \begin{bmatrix} 30 \\ 30 \end{bmatrix}$ and $\mathbf{t} = \begin{bmatrix} 30.4 \\ 30.4 \end{bmatrix}$, and $\epsilon = 0.5$, then:

$$\|\mathbf{p} - \mathbf{t}\| = \sqrt{(30.4 - 30)^2 + (30.4 - 30)^2} = \sqrt{0.16 + 0.16} \approx 0.4$$

Thus, the target is reached.

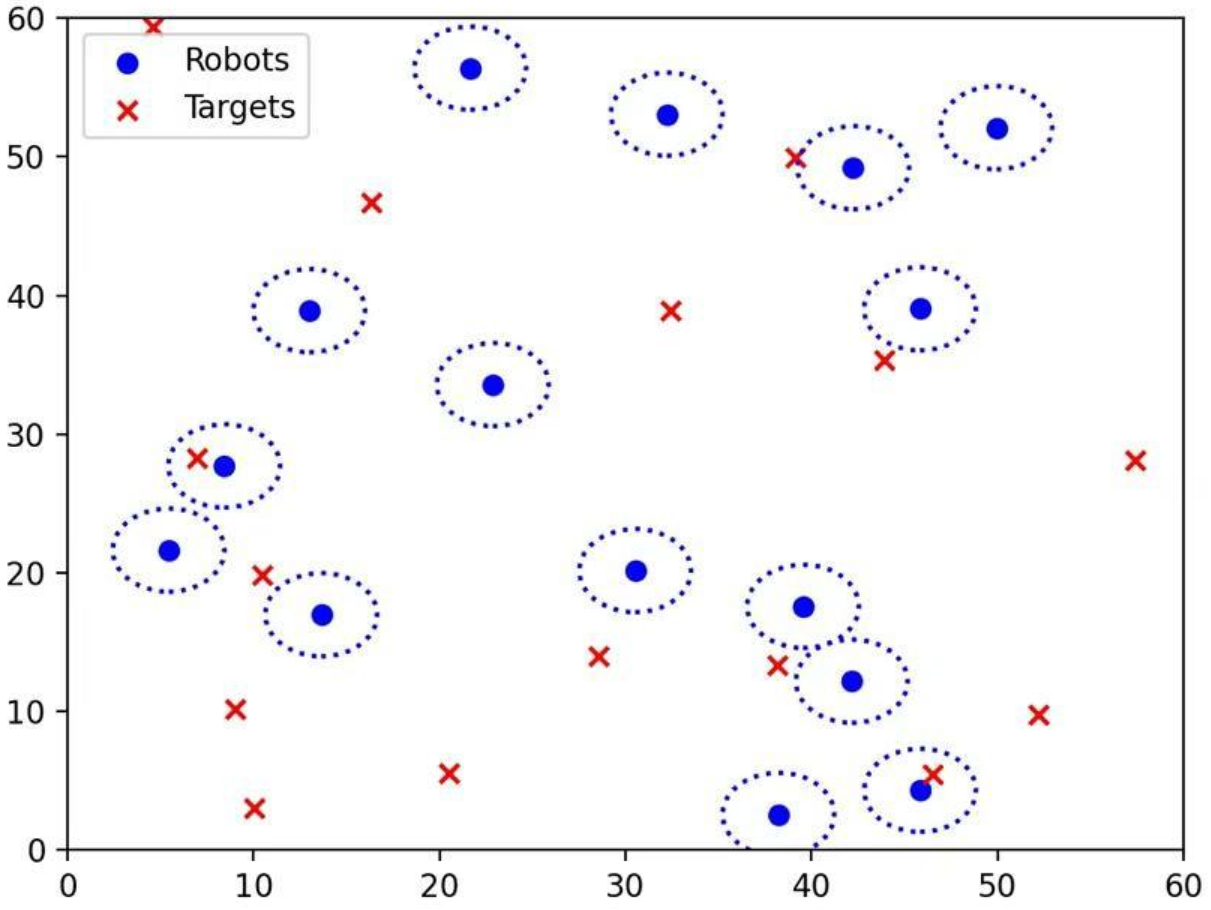
2D: Robots as Single point

Video [Link]



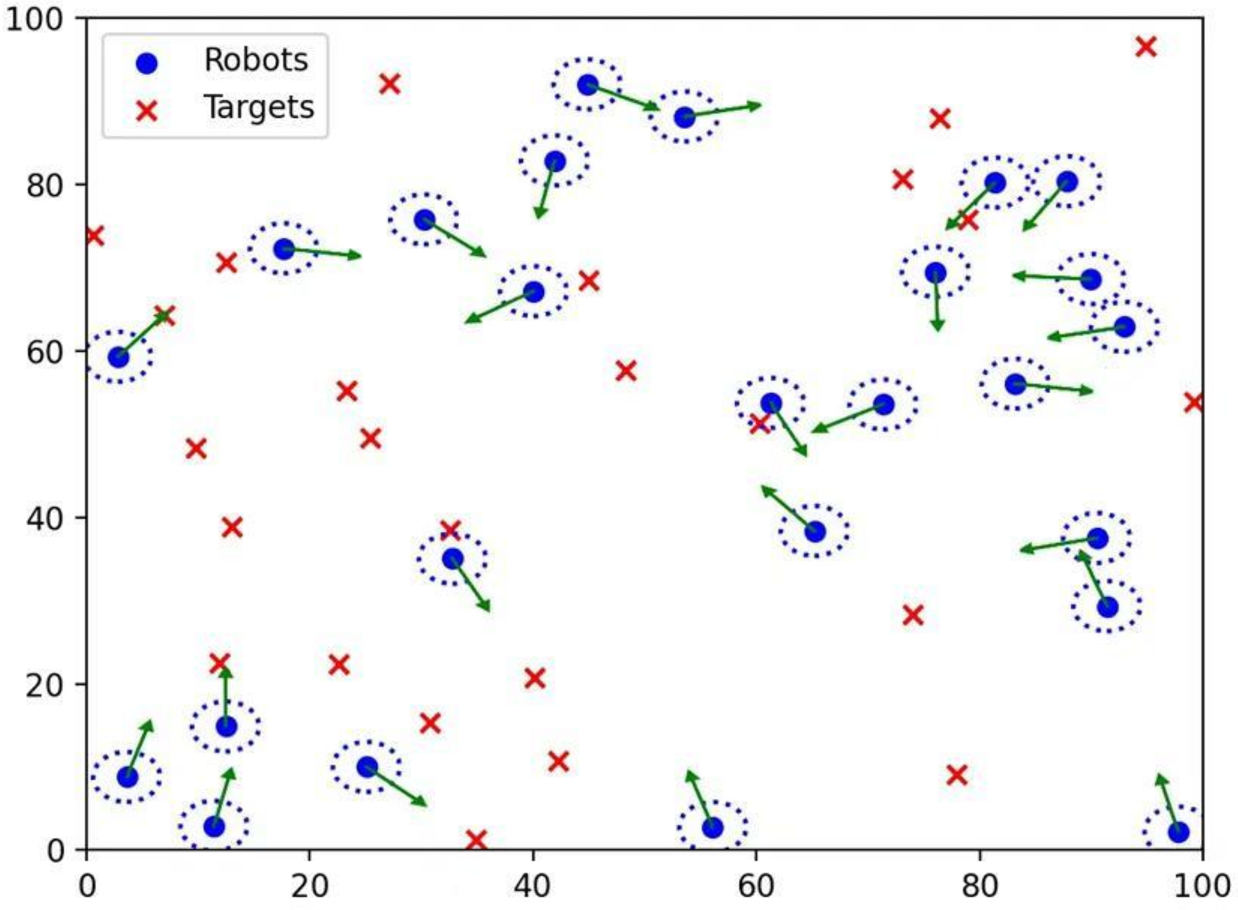
2D: Robots with Circular area

Video Link



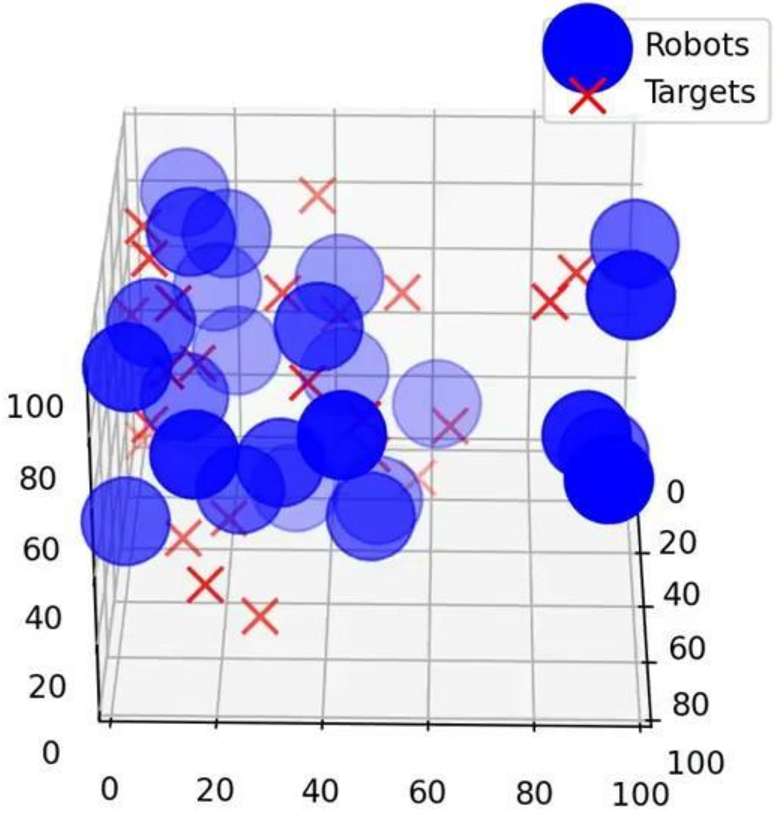
2D: Robots with Vector showing current direction

Video [Link]



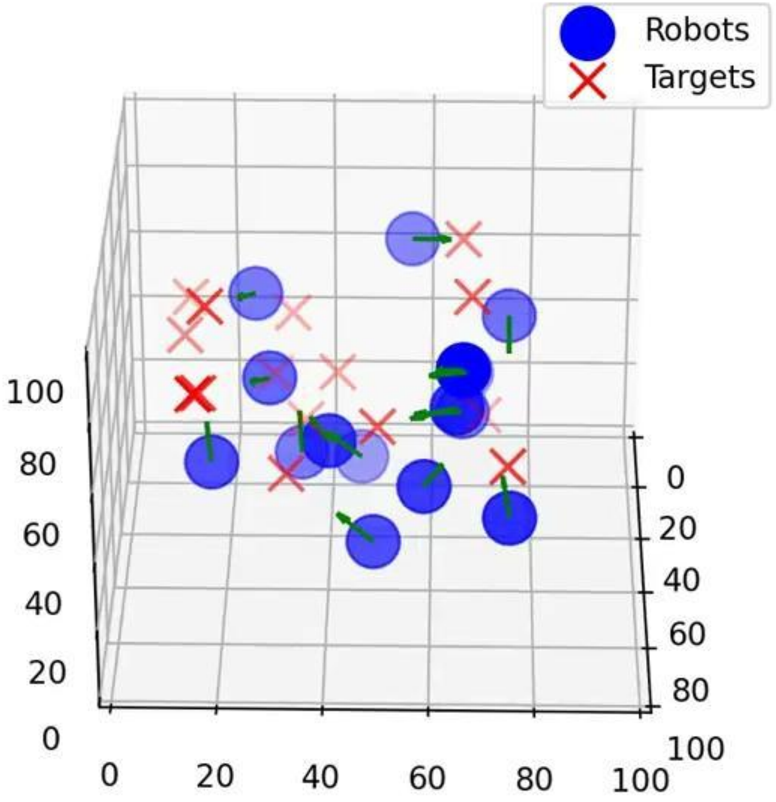
**3D**: Robots as sphere

Video [Link]

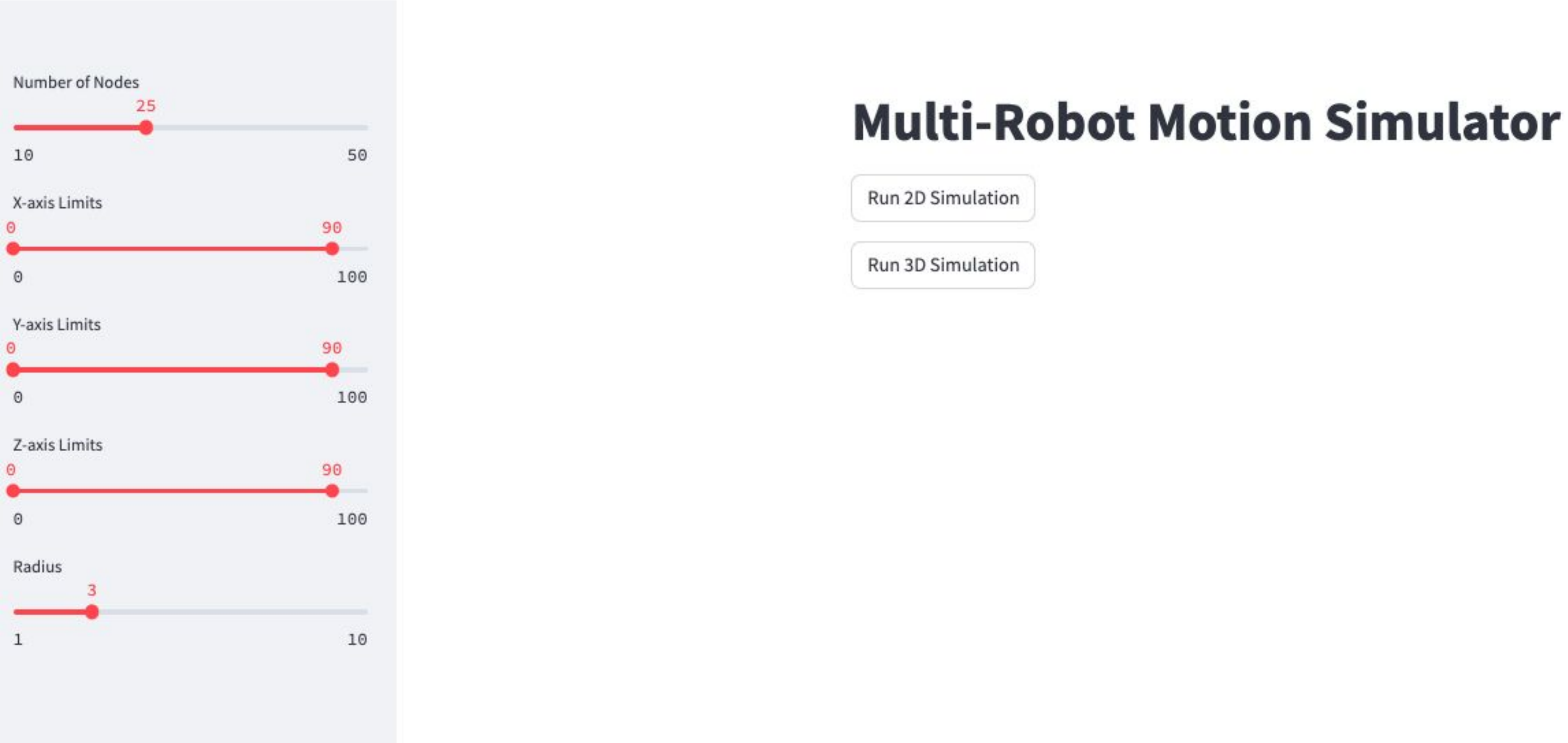


**3D**: Spherical Robots with Vector showing current direction

Video Link



# Demo:


Number of Nodes
25
10
50
X-axis Limits
0
90
0
100
Y-axis Limits
0
90
0
100
Z-axis Limits
0
90
0
100
Radius
3
1
10
Multi-Robot Motion Simulator
Run 2D Simulation
Run 3D Simulation

# Demo:

Number of Nodes

25

10 50

X-axis Limits

0 90

0 100

Y-axis Limits

0 90

0 100

Z-axis Limits

0 90

0 100

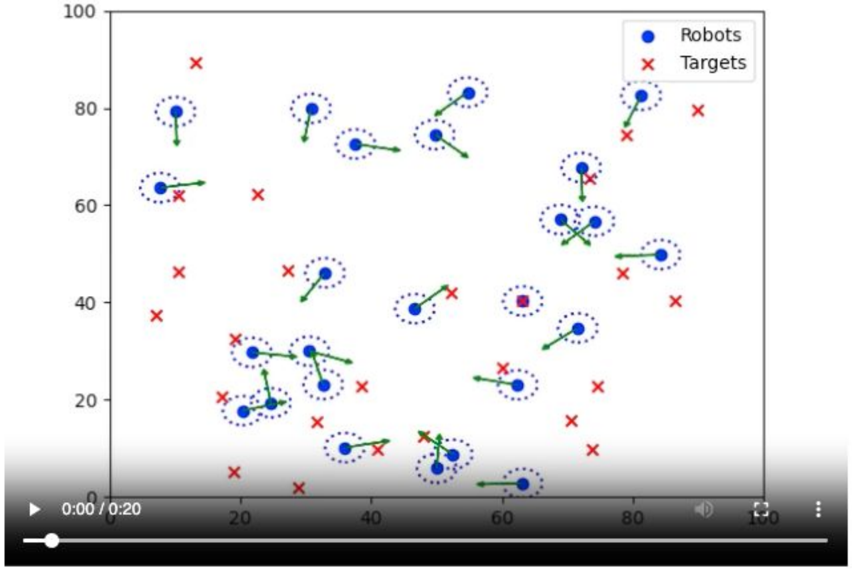
Radius

3

1 10

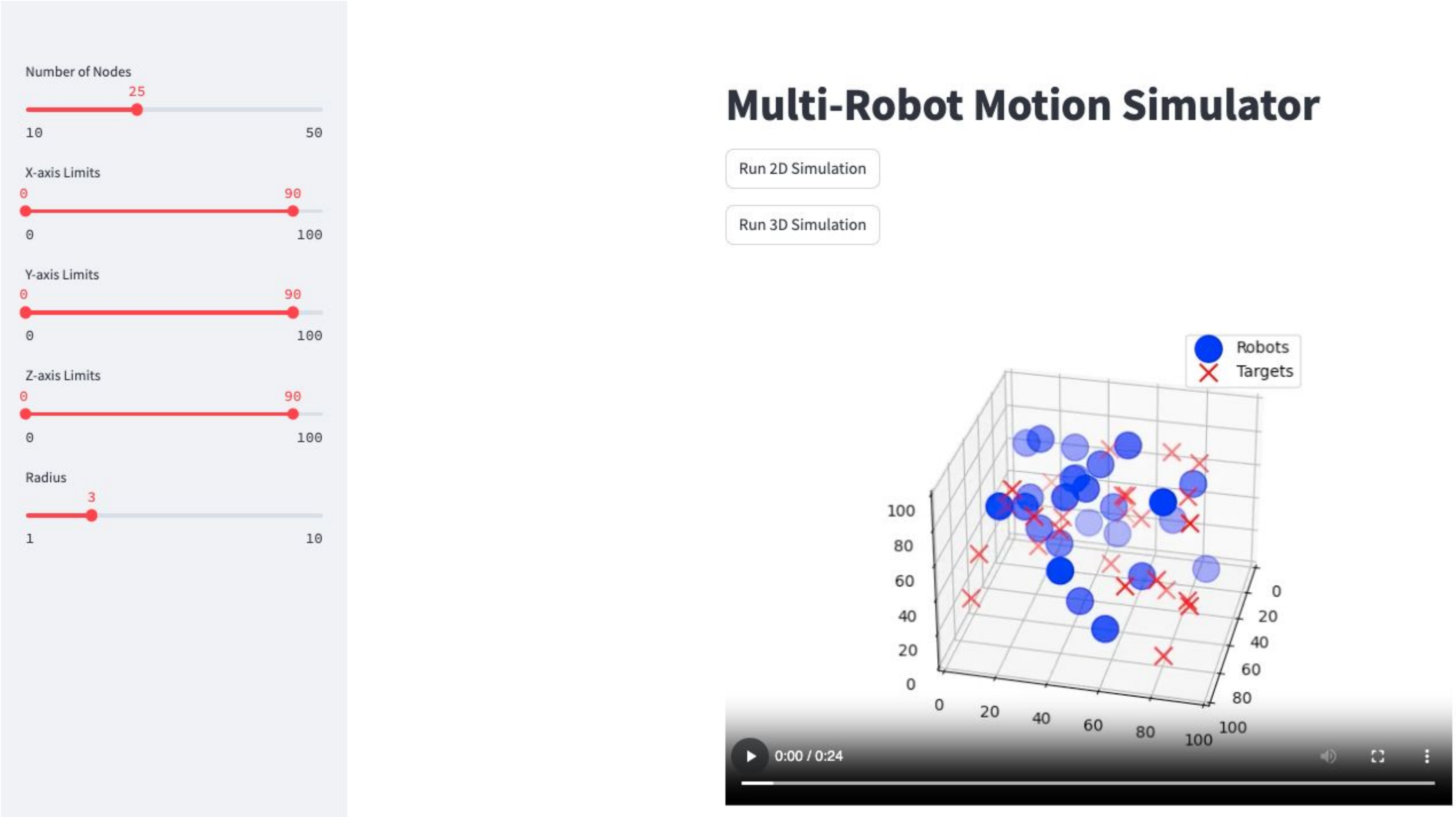
## Multi-Robot Motion Simulator

Run 2D Simulation




Run 3D Simulation

# Demo:


Number of Nodes
25
10
50
X-axis Limits
0
90
0
100
Y-axis Limits
0
90
0
100
Z-axis Limits
0
90
0
100
Radius
3
1
10
Multi-Robot Motion Simulator
Run 2D Simulation
Run 3D Simulation
Robots
Targets
0:00 / 0:24

# Conclusion:

- Effective multi-robot collision avoidance implemented.
- Demonstrated 2D and 3D simulation capabilities.
- Utilized Python and Streamlit for interactive visuals.
- Robots successfully reached designated targets.
- Potential for broader robotic navigation applications.

# Future Work:

- Integrate real-time data for dynamic simulation adjustments.
- Explore more complex collision avoidance algorithms.
- Implement machine learning for autonomous decision-making.
- Develop interfaces for multi-user operational control.
- Extend to heterogeneous robot systems with varied capabilities.