Multi-Robot Coverage with Interval Analysis-based Obstacle Avoidance

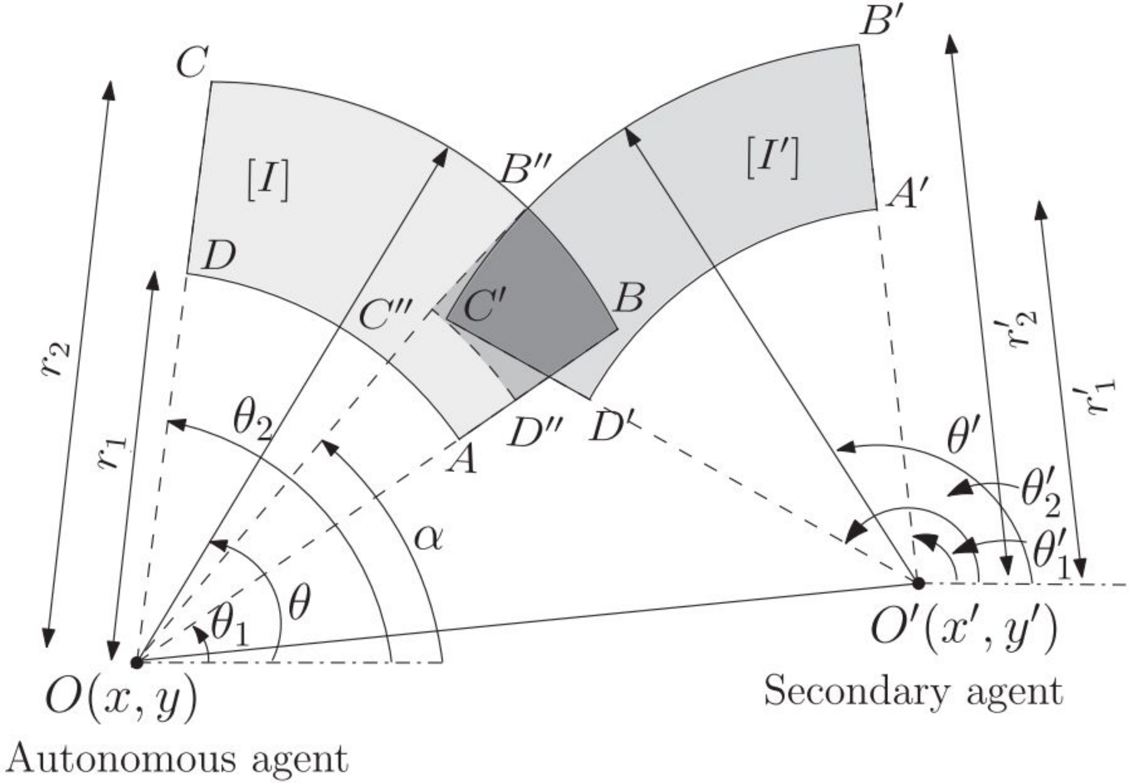
Kshitij M Bhat, Vivek Yogi and Leena Vachhani, Systems and Control Group, IIT Bombay.

Abstract

The goal of a Coverage Path Planning (CPP) Problem is to find an optimal path which covers an Area Of Interest (AOI) while ensuring collision avoidance with obstacles in the area. The use of multiple agents for this task leads to increase in robustness and scalability of the coverage. This work studies the multi-agent coverage problem for an enclosed environment with numerous static obstacles. Interval Analysis (IA) is used for the parallel collision avoidance between the agents as well as with the obstacles within the dynamic environment. This technique would enable the use of hardware and sensor-efficient low-end robots with only local sensing capabilities. Random Walks (RW) would lead to the decentralized solution without the use of localization and mapping.We compare the results of systematic coverage with the random walk coverage with IA-based Collision Avoidance.

Code: https://github.com/KshitijBhat/IA-Coverage

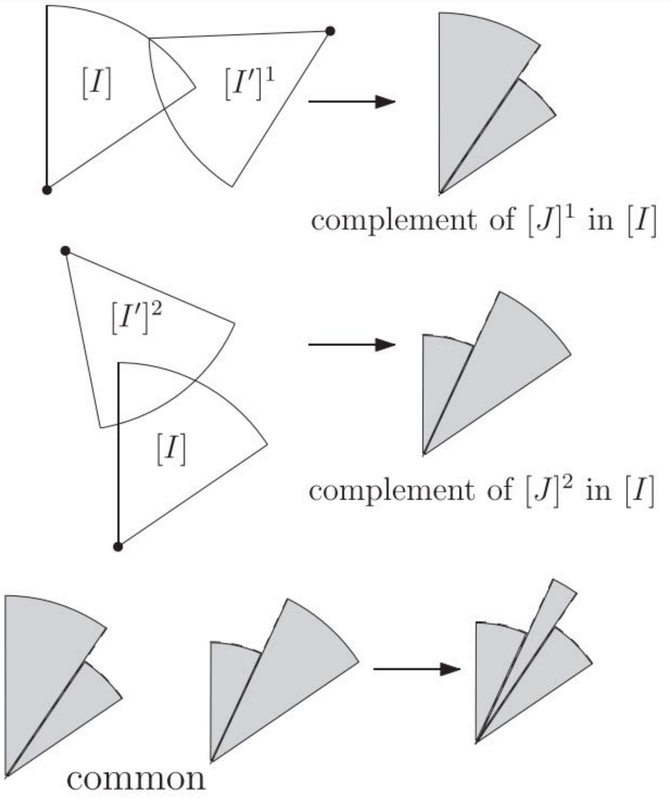
Interval Analysis



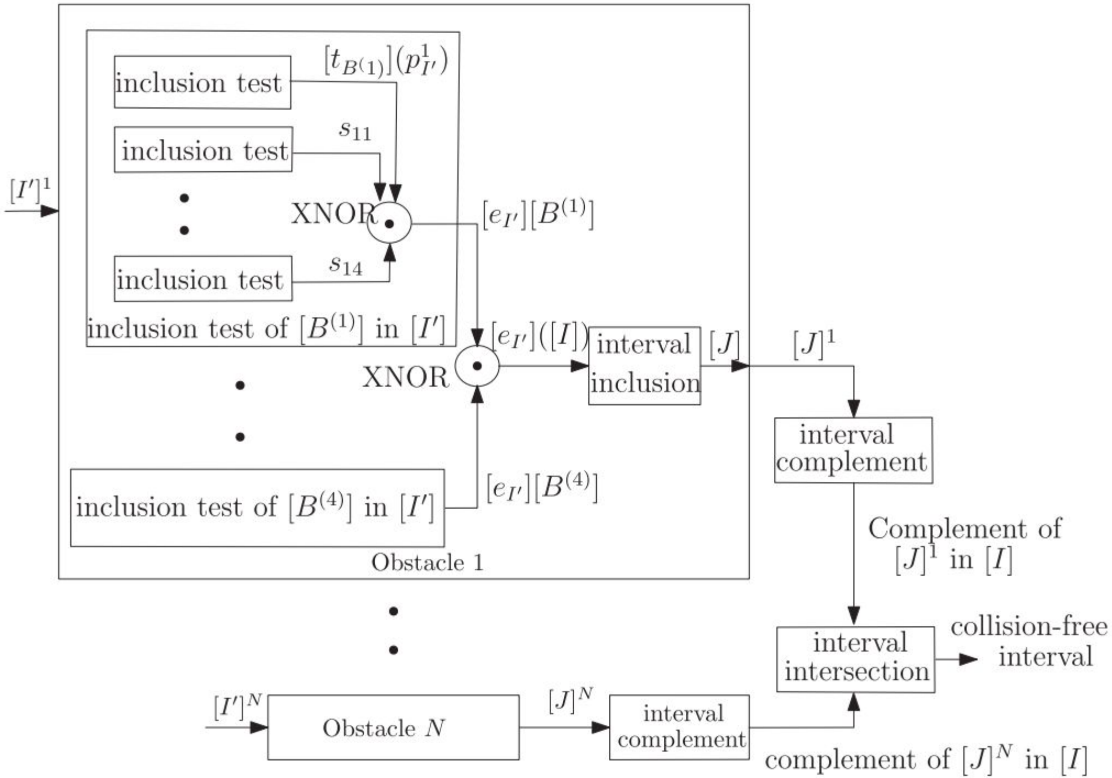
Collision Avoidance Algorithm

Algorithm 1 Interval Inclusion.

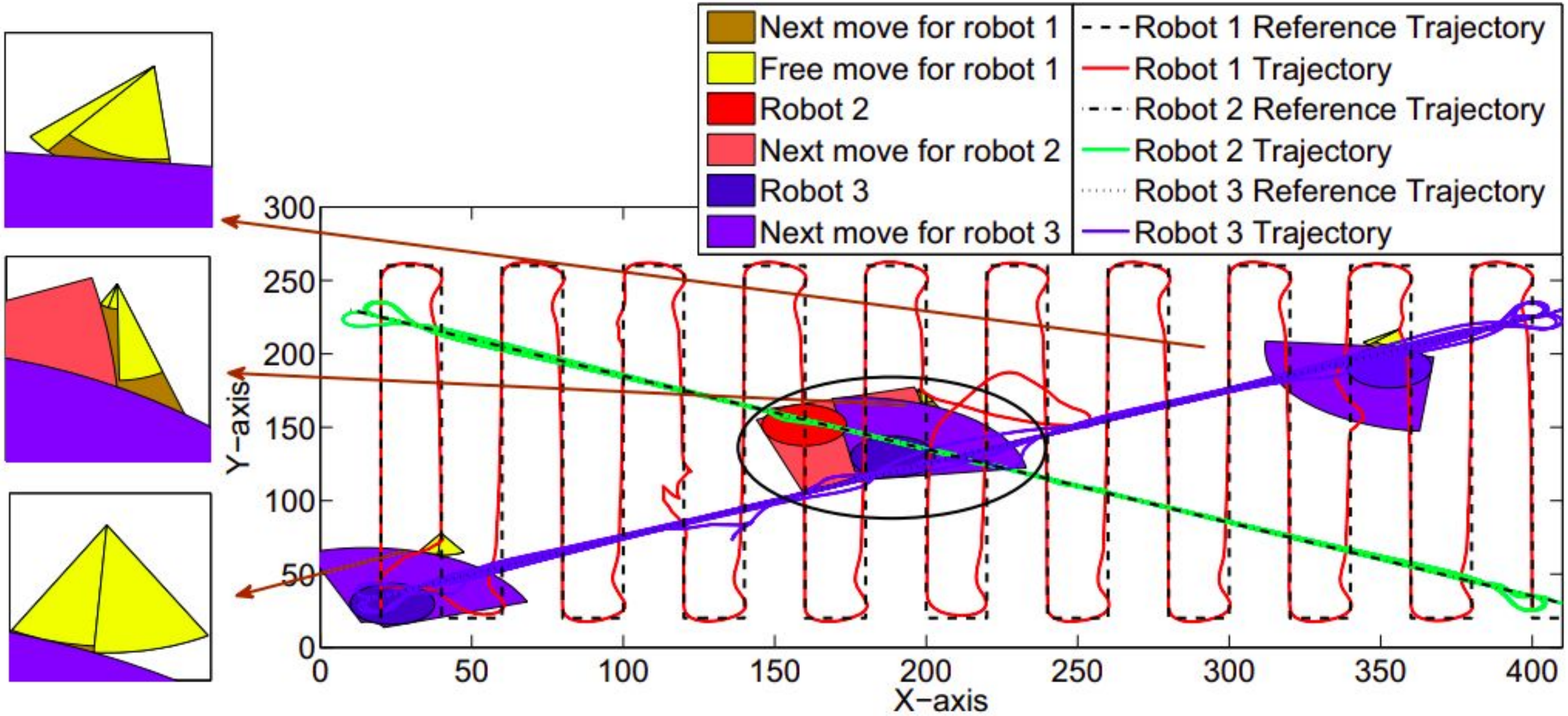
```
function INTERVAL_INCLUSION([I], [I′], N_r, N_θ)
    if [e_I′]([I]) = 0 then
        [J] = [0, 0][0, 0];
    else
        N ← N_r;  n ← 0
        for i = 1 to 2 do
            bisect [I] in [L] and [R]
            while n < N do
                bisect [L] in [LL] and [LR] and [R] in
[RL] and [RR]
                if [e_I′]([R]) = 0 then
                    [L] ← [LL];  [R] ← [LR]
                else if [e_I′]([L]) = 0 then
                    [L] ← [RL];  [R] ← [RR]
                else
                    if [e_I′]([LL]) ≠ 0 then
                        [L] ← [LL]
                    else
                        [L] ← [LR]
                    end if
                    if [e_I′]([RR]) ≠ 0 then
                        [R] ← [RR]
                    else
                        [R] ← [RL]
                    end if
                end if
                n ← n + 1
            end while
            [J_i] = [L̲_i, R̄_i];  N ← N_θ;  n ← 0
        end for
    end if
end function
```



Obstacle Avoidance Framework



Validation



Next move for robot 1
Free move for robot 1
Robot 2
Next move for robot 2
Robot 3
Next move for robot 3
Robot 1 Reference Trajectory
Robot 1 Trajectory
Robot 2 Reference Trajectory
Robot 2 Trajectory
Robot 3 Reference Trajectory
Robot 3 Trajectory
Y−axis
X−axis
0
50
100
150
200
250
300
350
400

Multi-Robot Collision Avoidance