Optimization Problems in Visual Surveillance

M.Sc. Thesis Seminar

Ahmed Abdelkader Abdelrazek

August 13, 2013

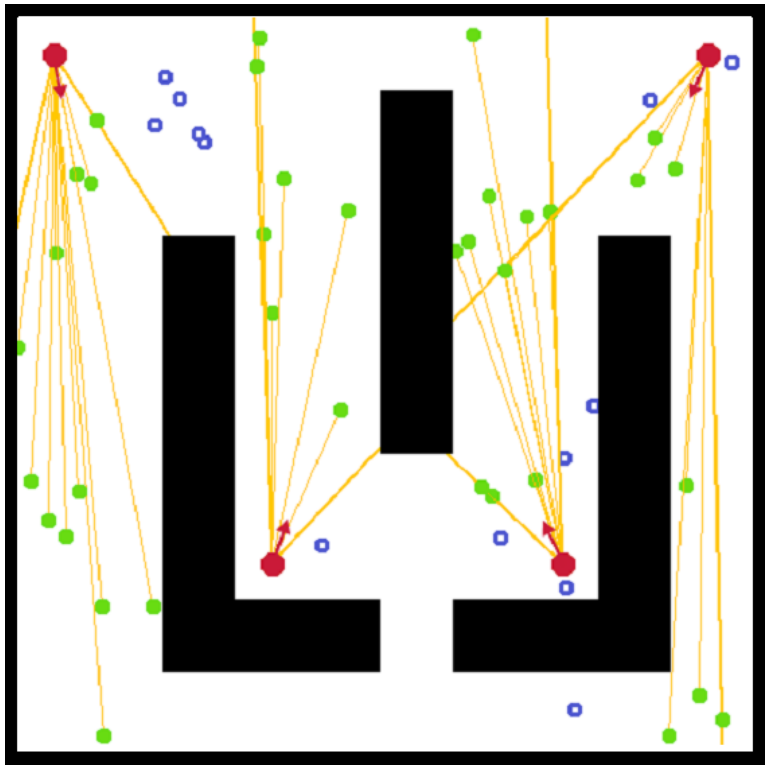
Visual Surveillance

Public Safety and more

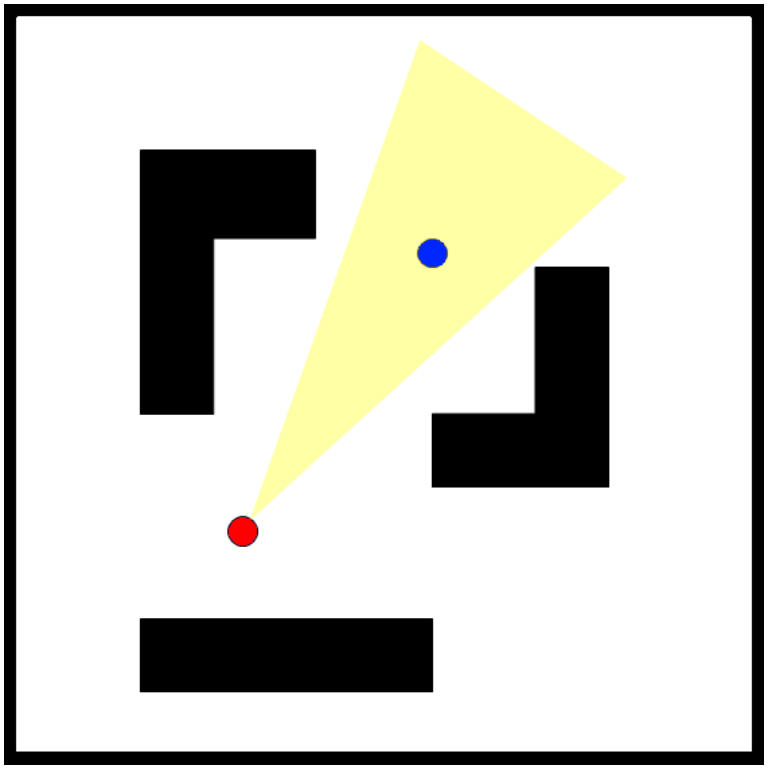
BCIVS DIWK41M

Suspicious?

Thesis Overview



Coverage Maximization
in Multi-Camera Surveillance

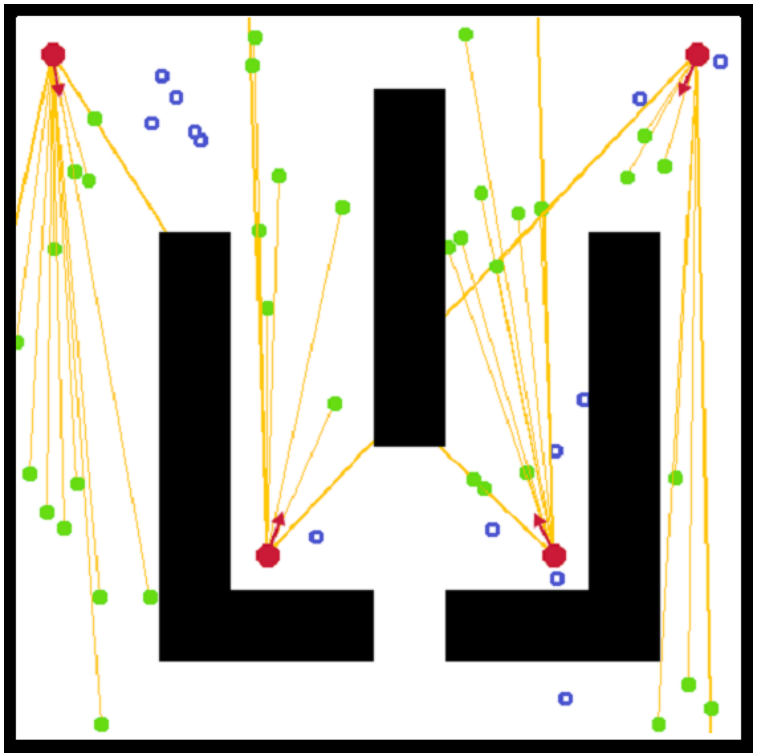


Visibility-Based
Pursuit-Evasion Games

Coverage Maximization in Multi-Camera Surveillance

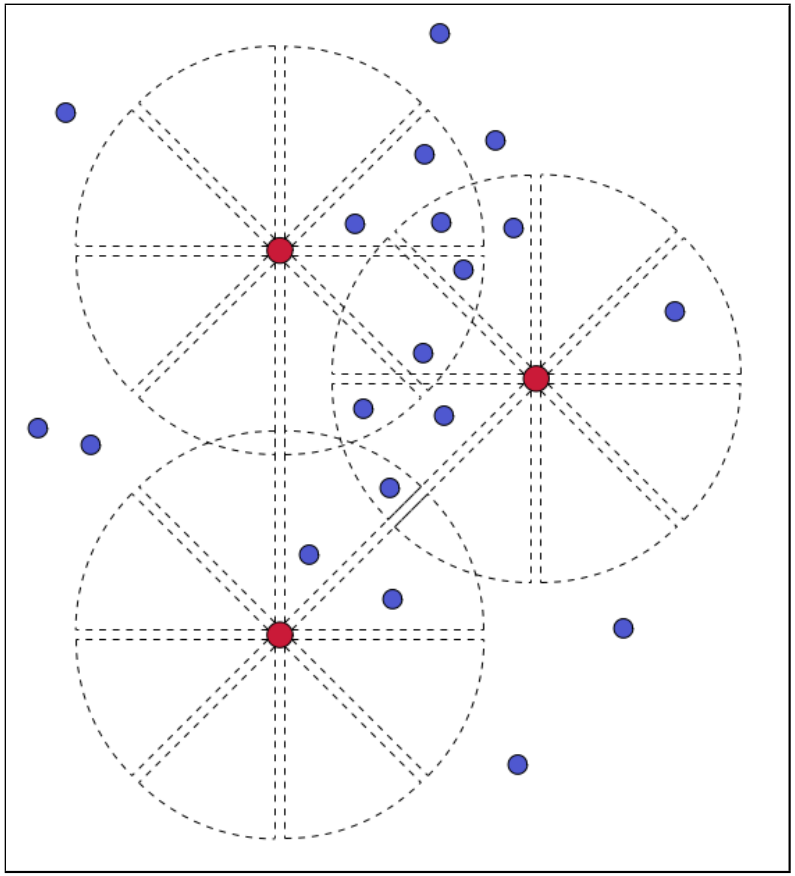
Problem Definition

- Goal: Maximize target coverage!
- Wide-angle top-view (Master)
- Panning Cameras (M Slaves)
- Unbounded sensing range
- Static targets (N targets)



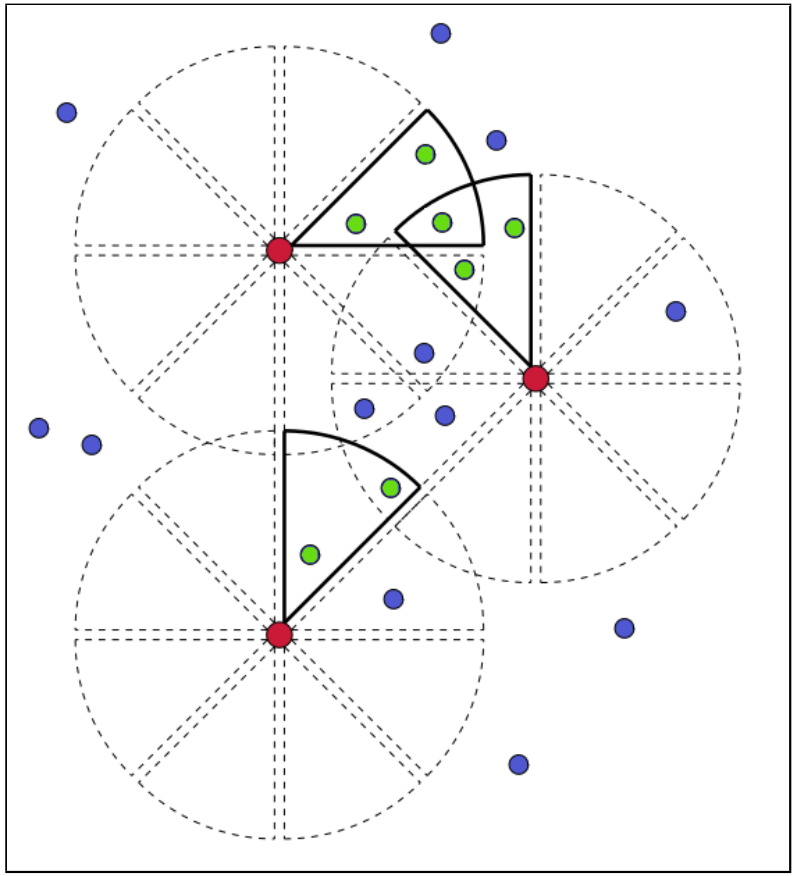
Coverage Maximization in Multi-Camera Surveillance

Problem Definition



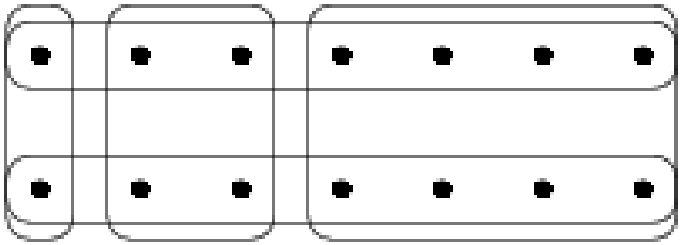
Input Configuration

$\longrightarrow$



Output Assignment

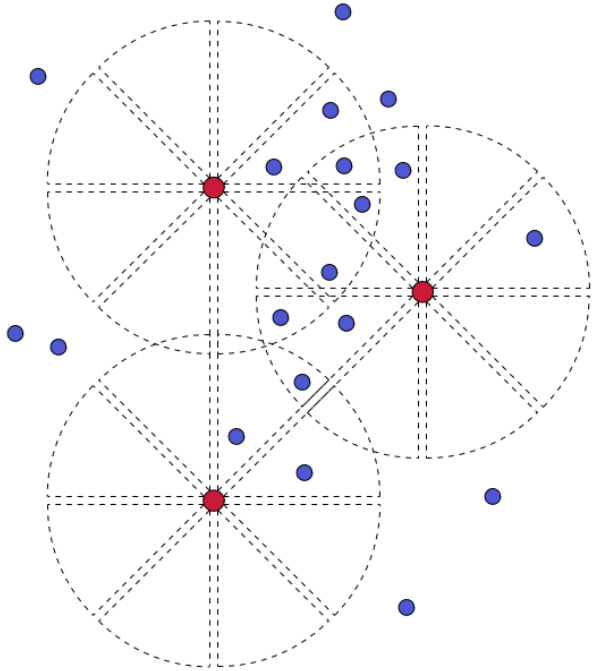
Greedy Set Cover



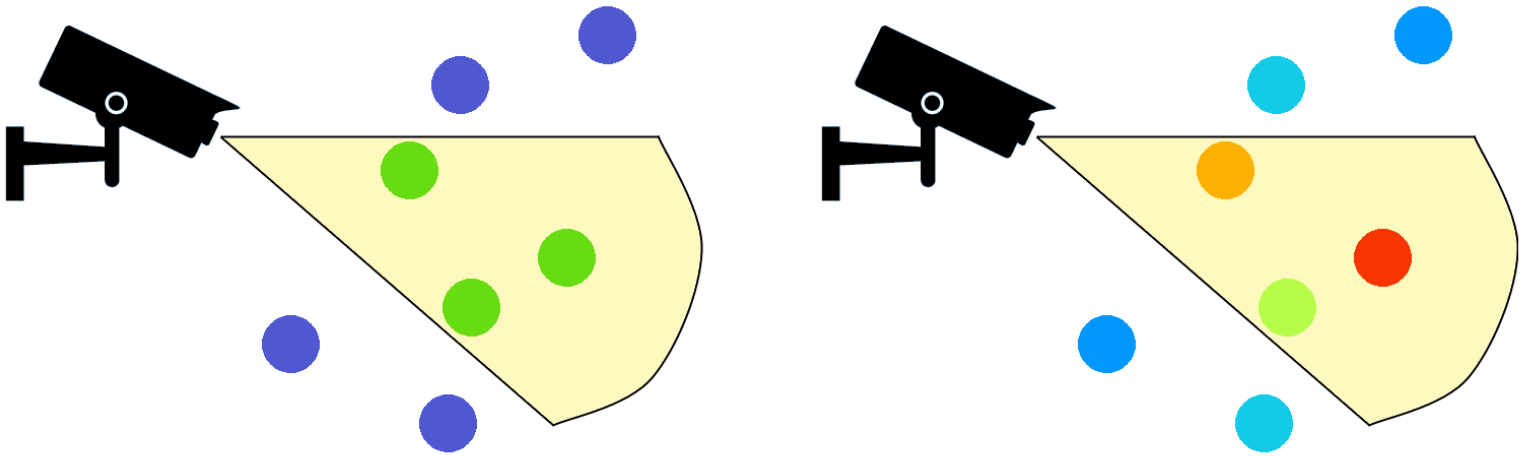
What is the maximum possible coverage using only 2 subsets?

Greedy Camera Assignment

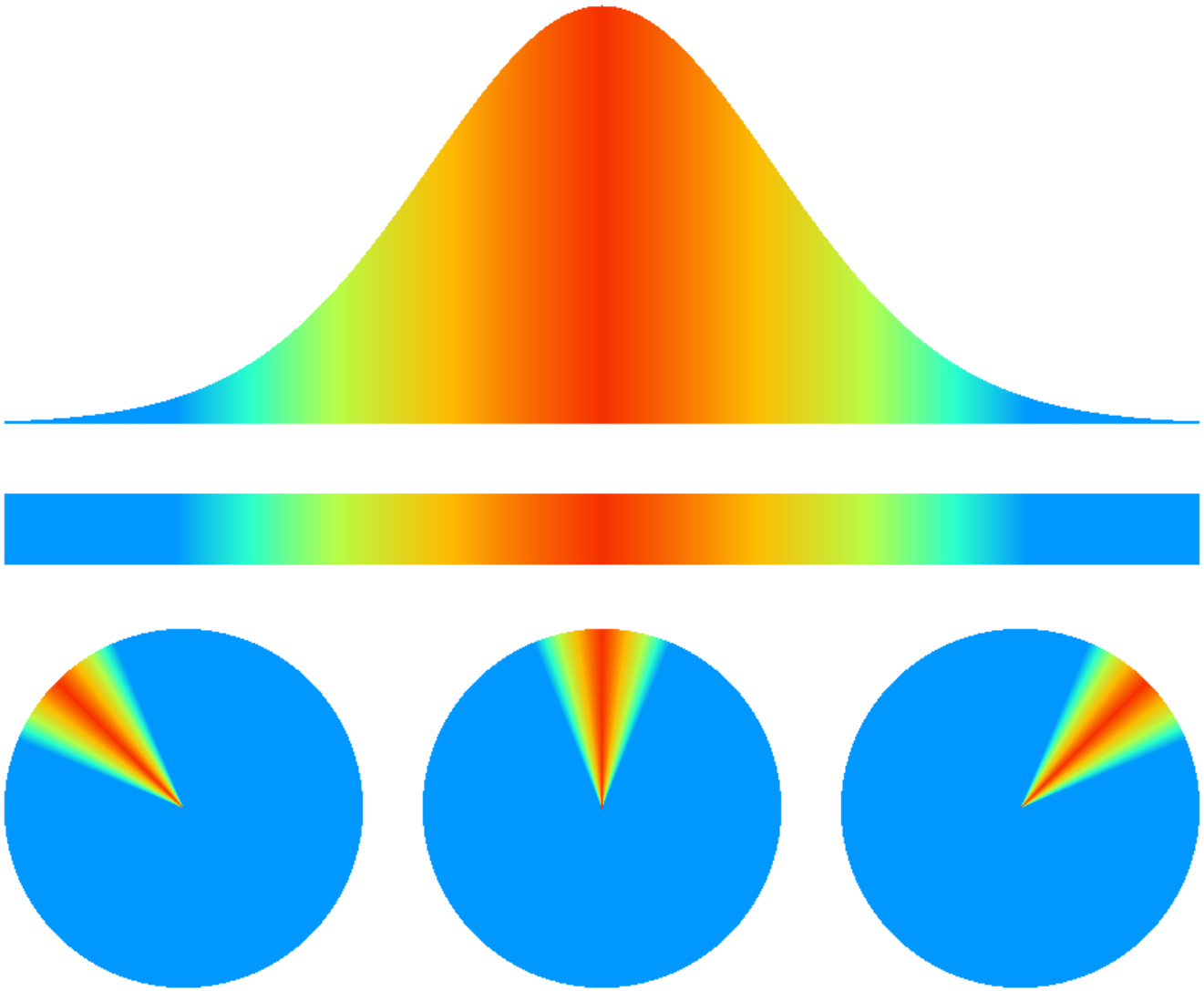
- Centralized Greedy Algorithm (CGA)
  - Repeat until done:
  - Assign the camera that covers the largest number of uncovered targets
- Centralized Force Algorithm (CFA)
  - Normalize the number of covered targets by the total number of targets that may be covered by the camera



Coverage as a Continuous Function



Coverage as a Continuous Function



Coverage as a Continuous Function

Lemma

Fix a positive real $w_{c,t}$ for each camera c and target t, such that $\frac{w_{min}}{w_{max}} > \frac{N-1}{N}$. Any weight function of the form

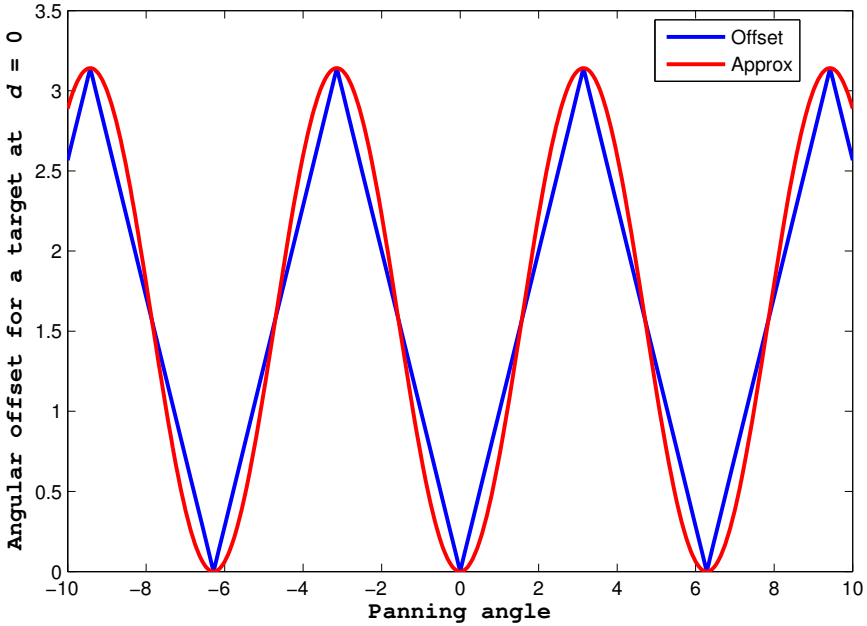
$$f_c(t) = \begin{cases} w_{c,t}, & c \ covers \ t, \\ 0, & otherwise, \end{cases}$$

describes an optimal solution maximizing $\sum_{c \in C} \sum_{t \in T} f_c(t)$.

$$f_c(t) = \left(\frac{N-1}{N} \right)^{\theta_c(t)/\theta_{max}}$$

Target Angle as a Function of Camera Angle

$$\theta_c(t) \longrightarrow \theta_c(d,t)$$

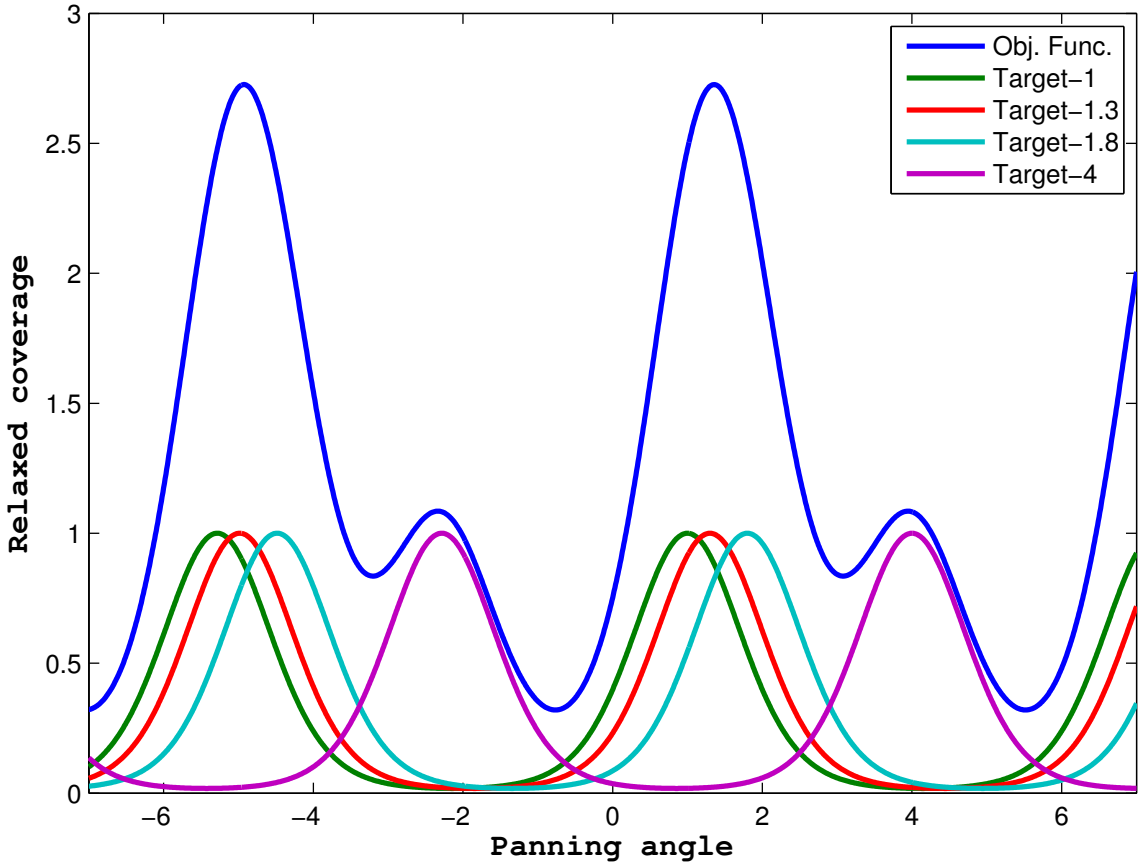


$$f_c(d,t) = e^{-\frac{\pi}{2}\left(\frac{1-\cos\theta_c(d,t)}{\theta_{max}}\right)}$$

Coverage by a Single Camera

Numerical Example

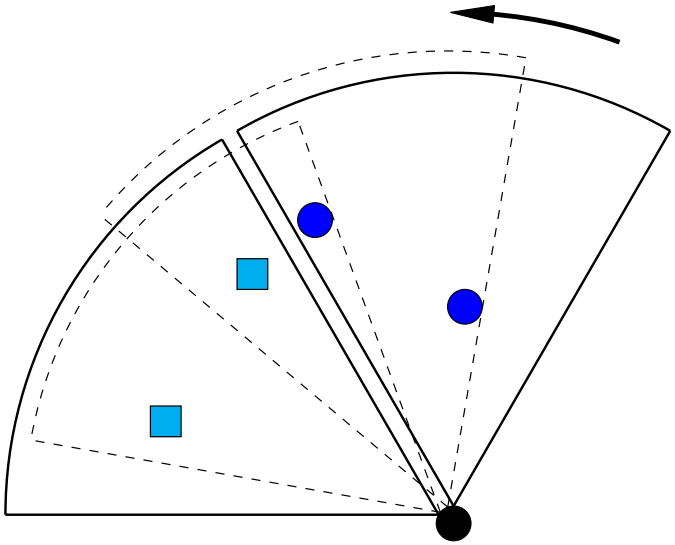
4 targets at angles [1, 1.3, 1.8, 4].



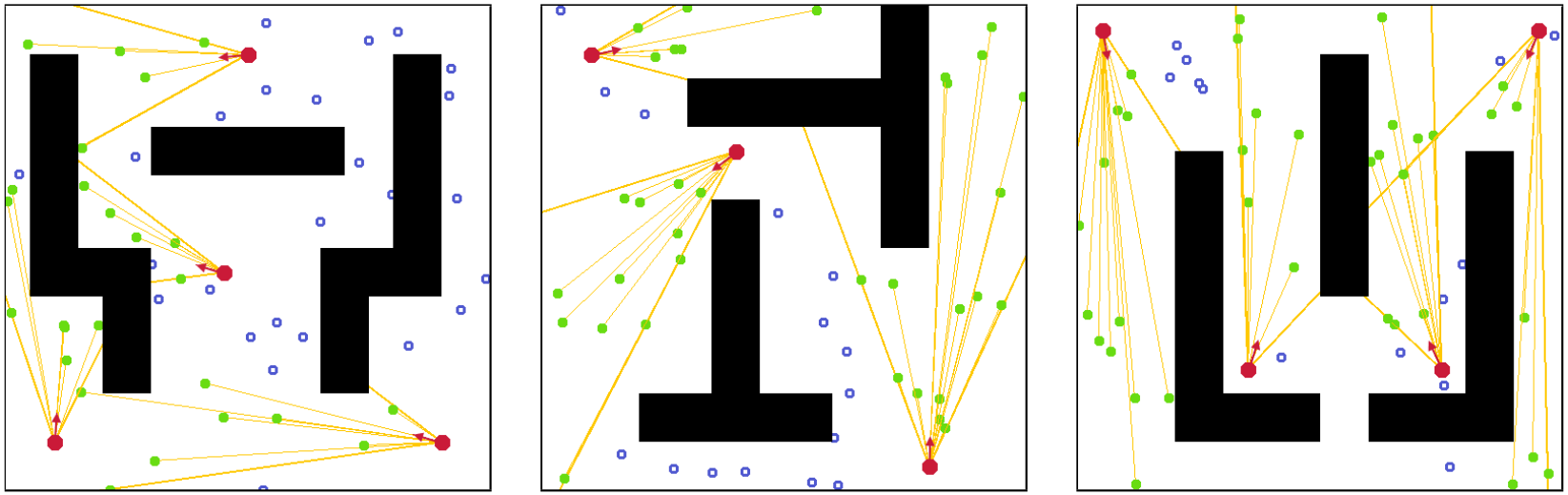
Coverage by Angular Sweeping

Observation (Coverage by Maximal Sets)

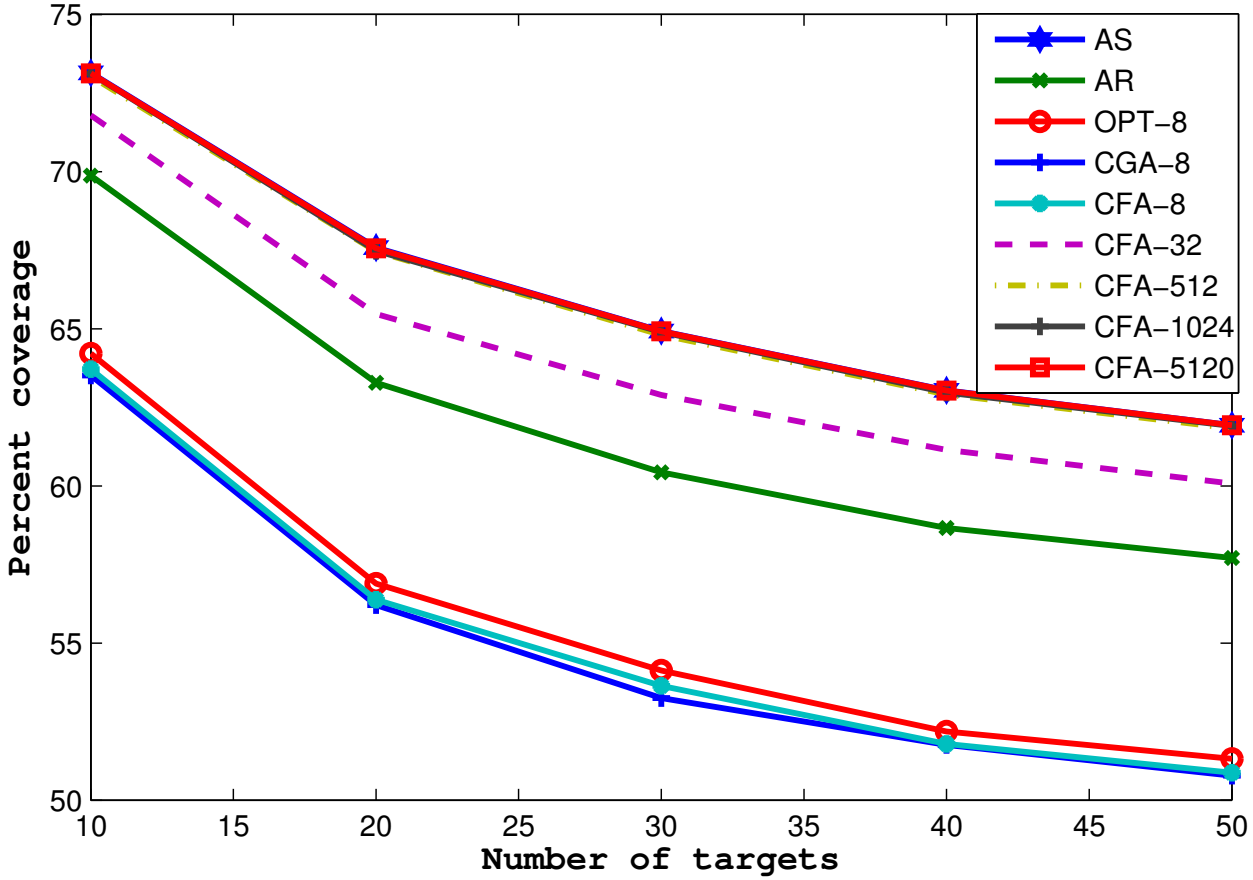
Any set of targets covered by a given camera can always be covered by panning the camera till one or more of the targets lie exactly on either side of the FOV.



Sample Maps



Results: Coverage Rate

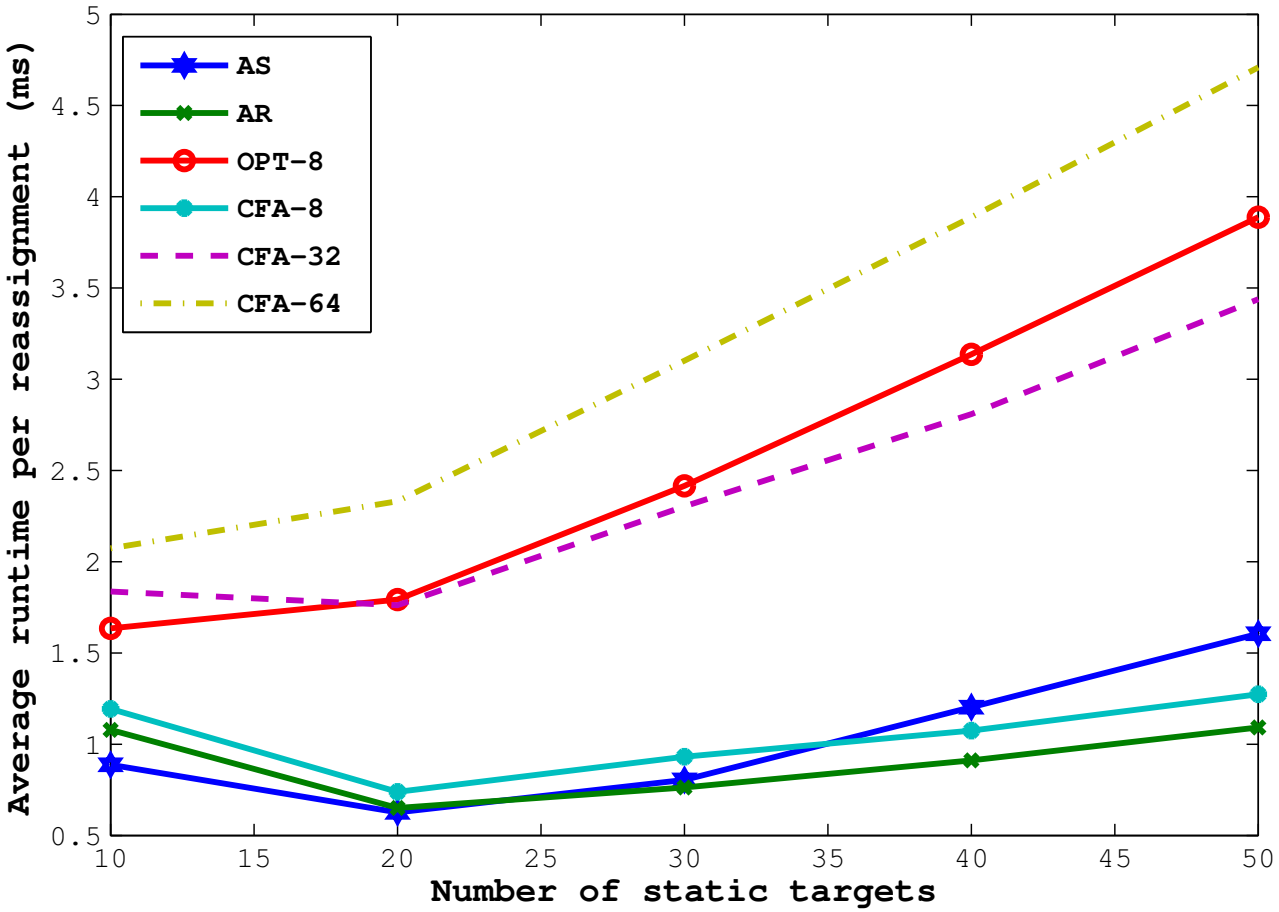


Results: Coverage Rate

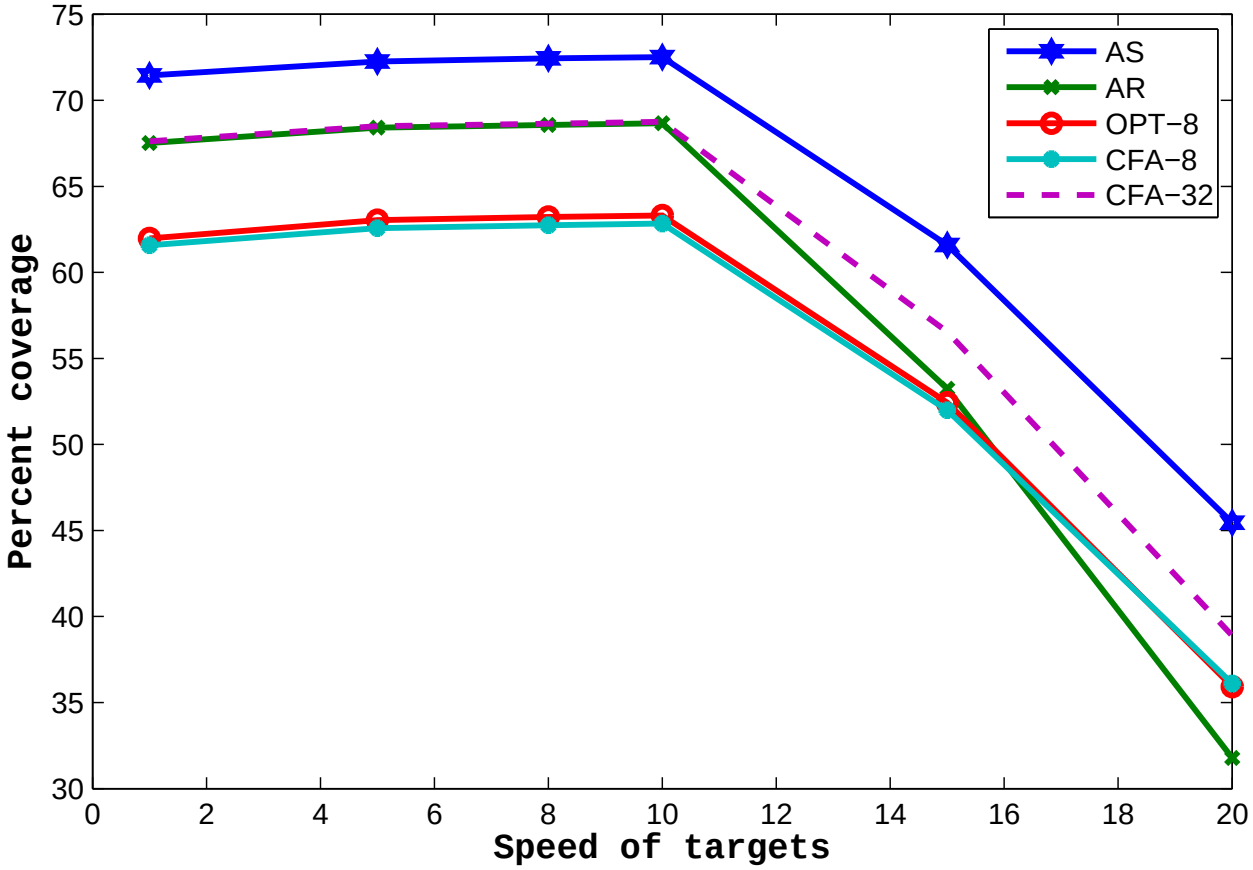
A closer look

N	10	20	30	40	50
CFA-32	71.79	65.47	62.89	61.14	60.08
CFA-512	73.03	67.46	64.8	62.91	61.83
CFA-1024	73.11	67.48	64.90	62.97	61.90
CFA-5120	73.13	67.56	64.92	63.04	61.93
AS	73.14	67.58	64.93	63.04	61.93

Results: Runtime



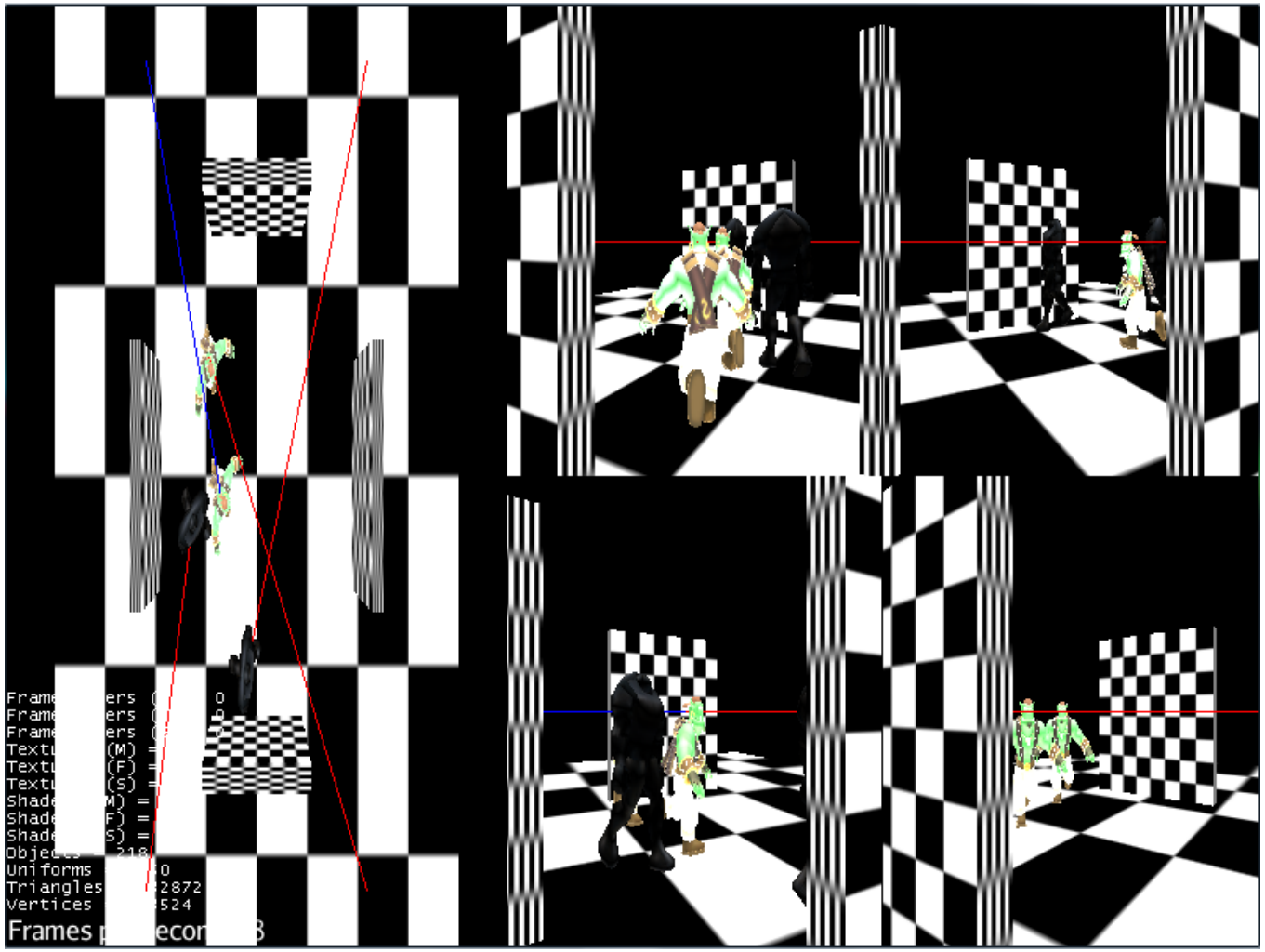
Results: Dynamic Coverage



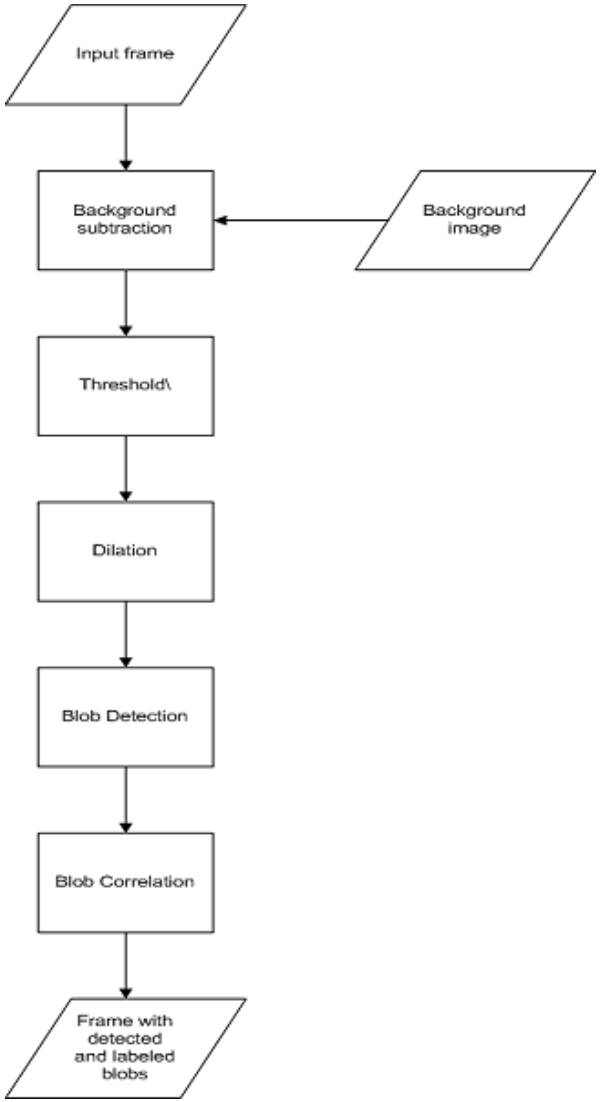
Experimental Testbeds

- 3D and 2D Simulators.
- Toy Experiment - 7th floor, EE Building.
- Prototype System - Preparatory Building.
- Full System - Administration Building.

3D Simulator

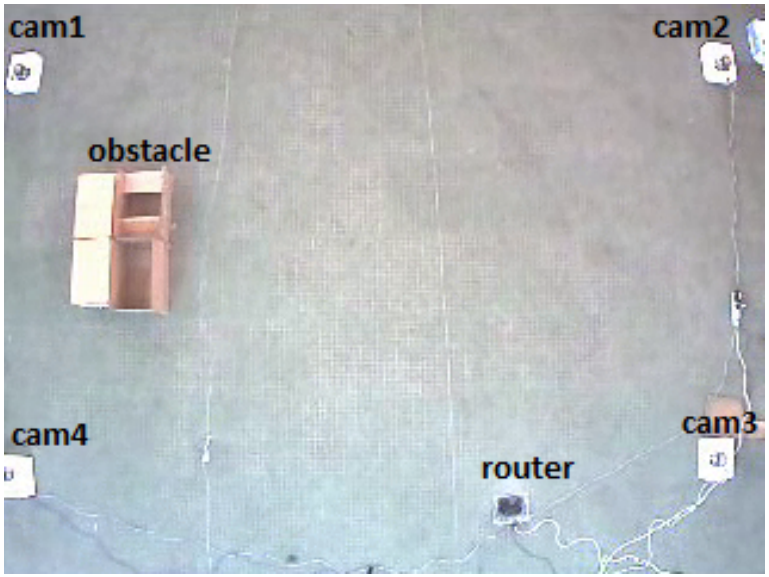


Computer Vision Methods

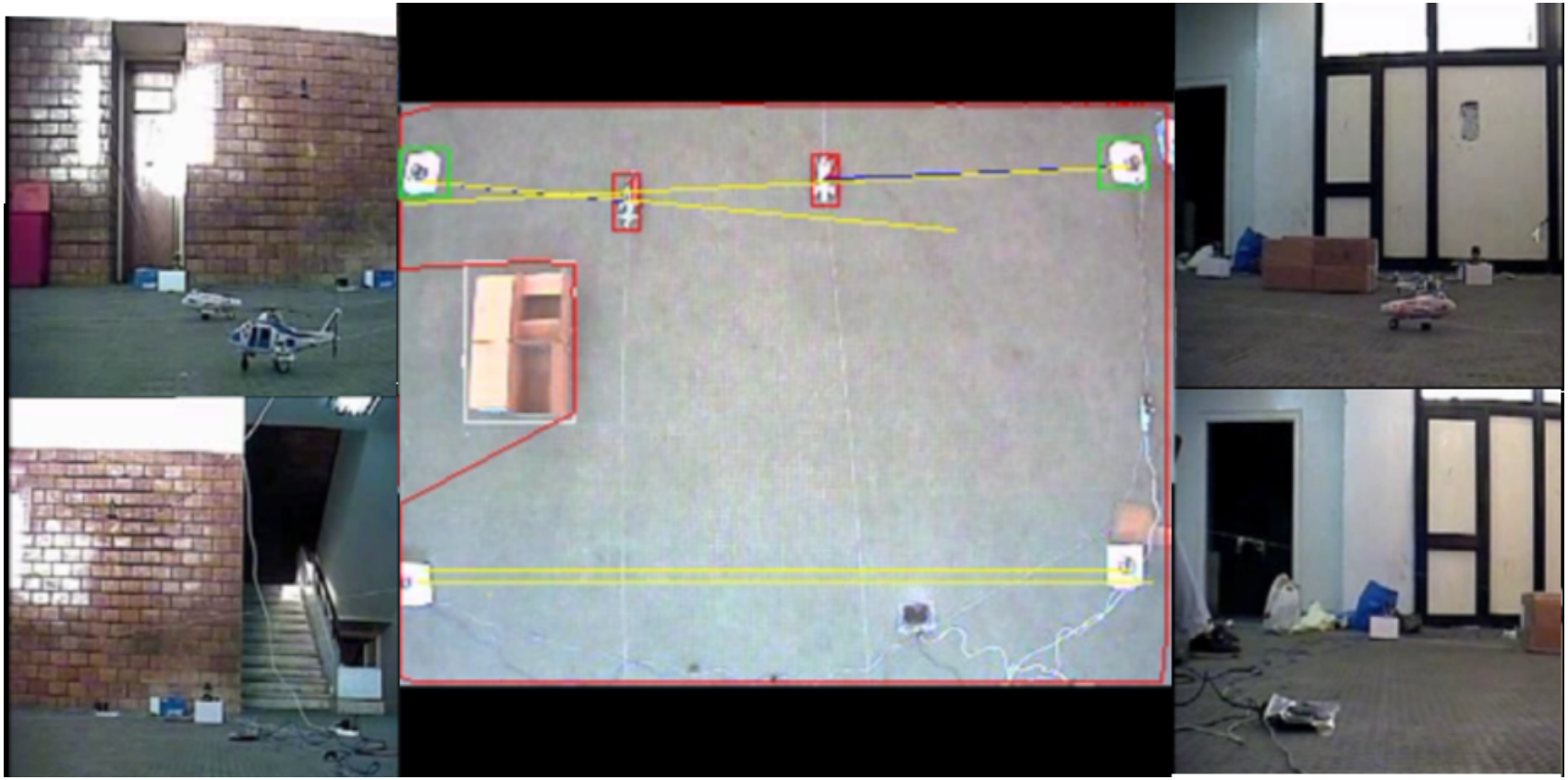


Video Tracking

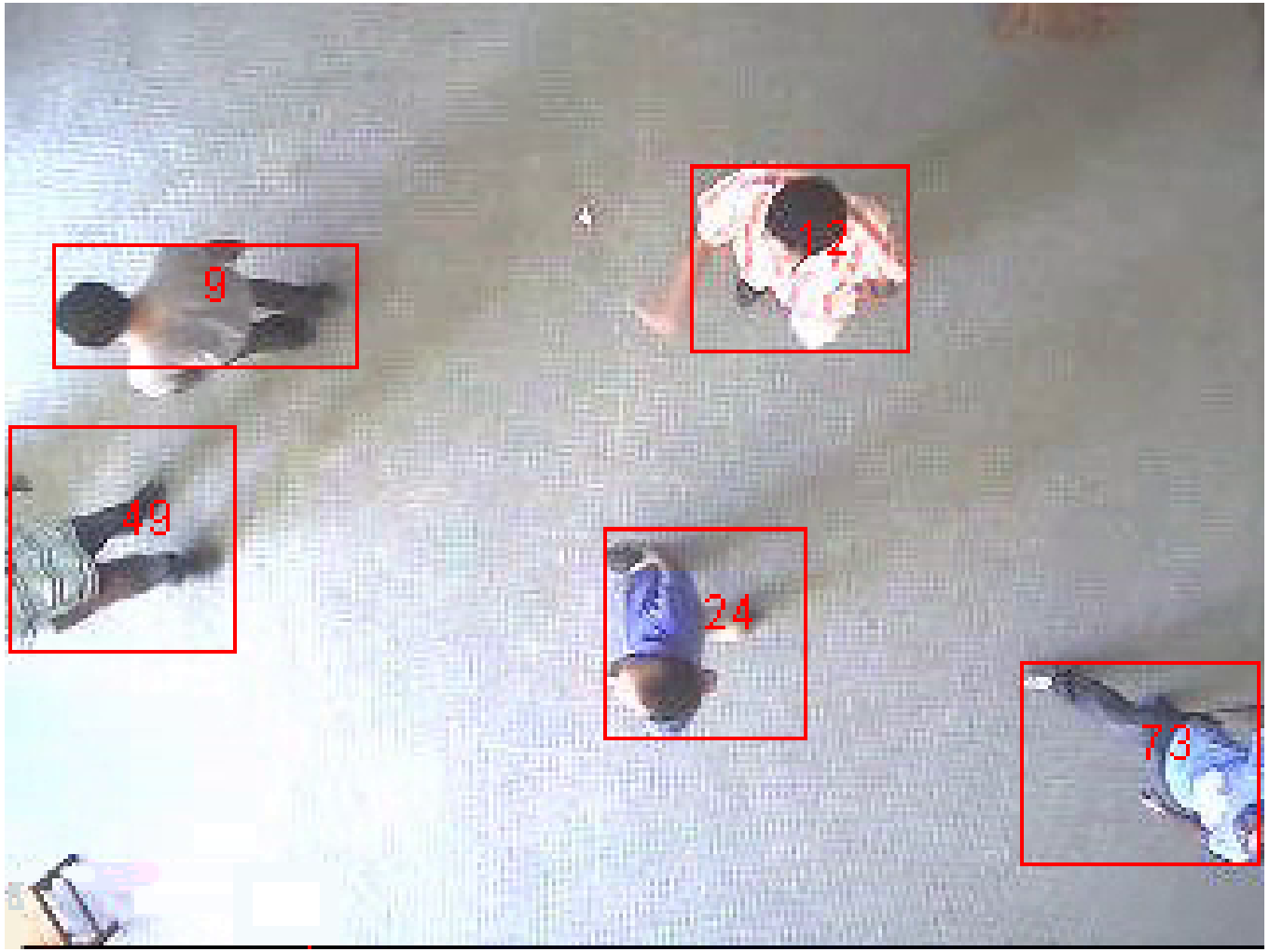
Toy Experiment - 7th floor, EE Building



Toy Experiment - Sample Frame



First Attempt at Human Tracking

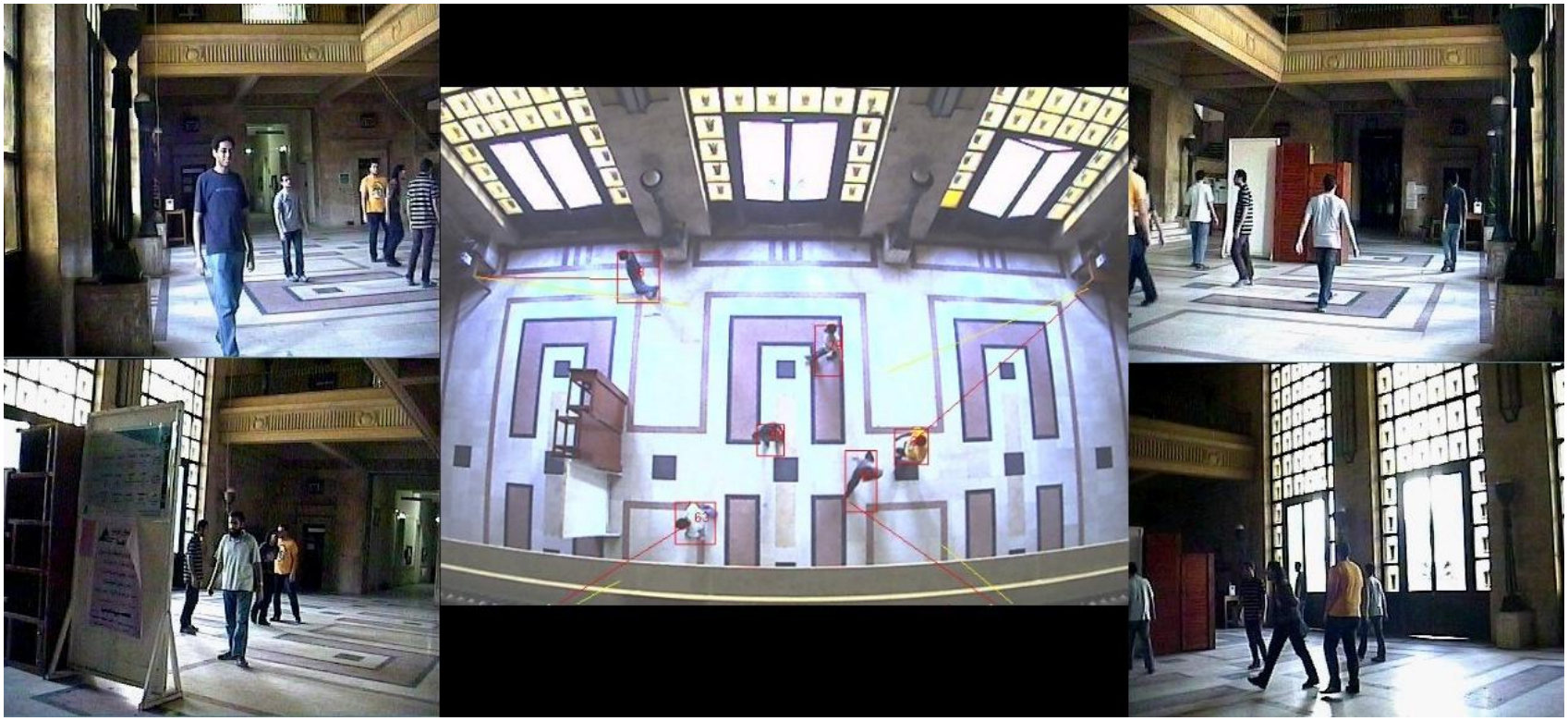


Prototype System - Preparatory Building

Prototype System - Sample Frame

Full System - Administration Building

Full System - Sample Frame



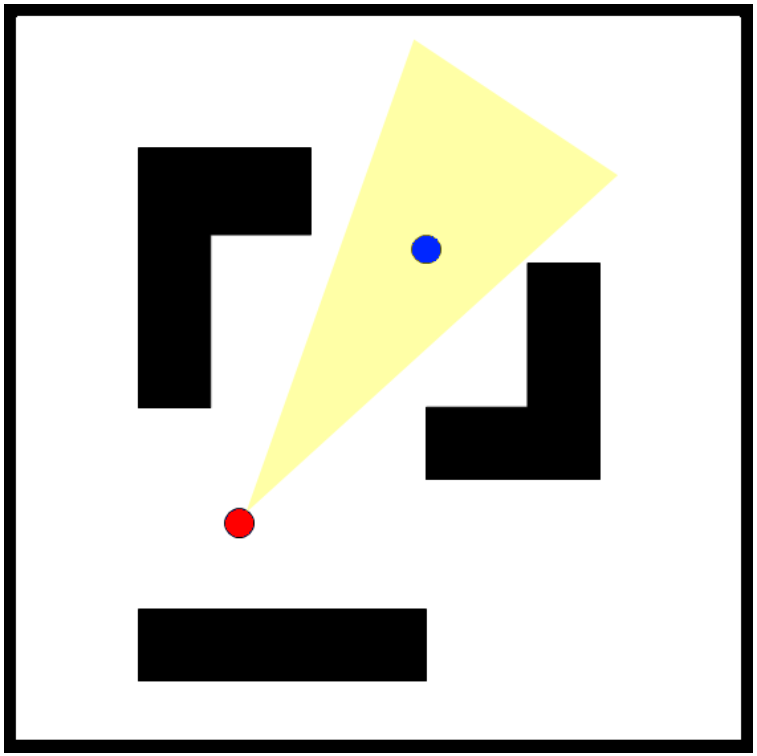
Multi-Camera Surveillance Tests

Sample Videos

Visibility-based Pursuit-Evasion

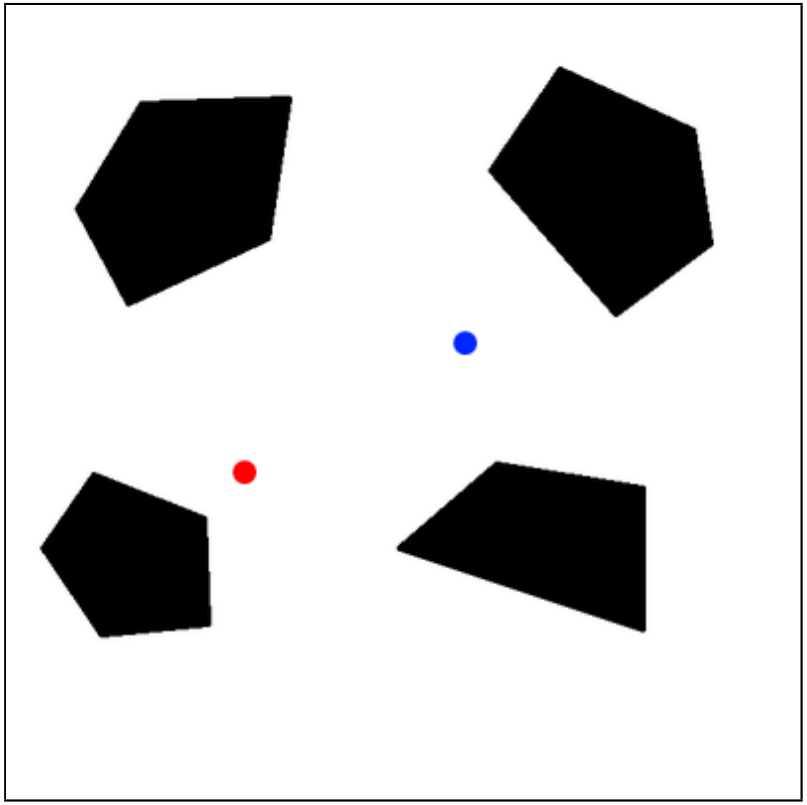
Problem Definition

- Goal: Keep evader in sight!
- Omnidirectional vision
- Move in turns, evader goes first
- Bounded speeds
- Holonomic motion
- Arbitrarily shaped obstacles
- Complete information



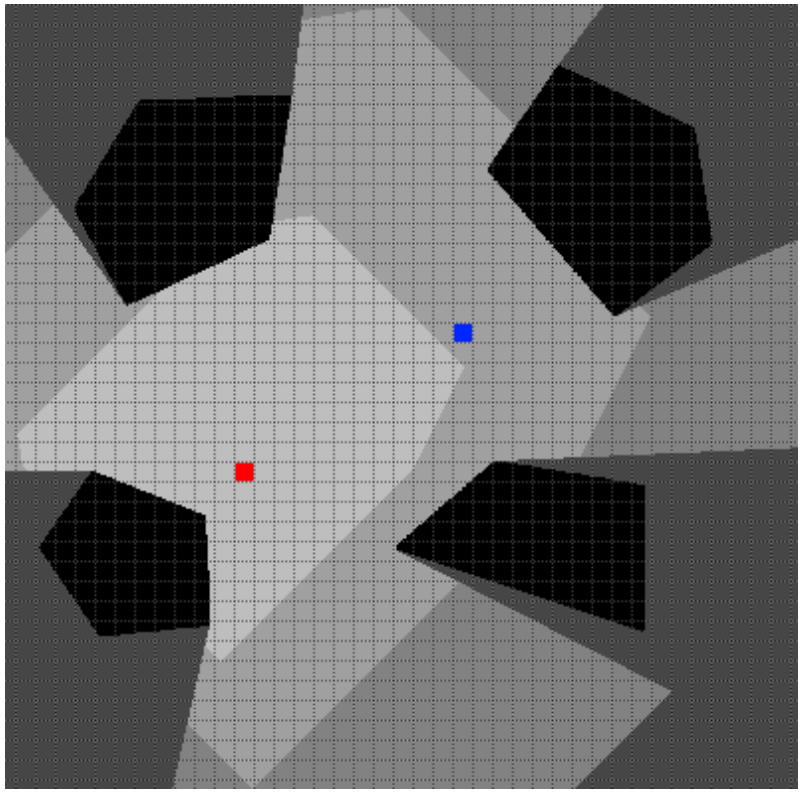
Visibility-based Pursuit-Evasion

Problem Definition



Input Map

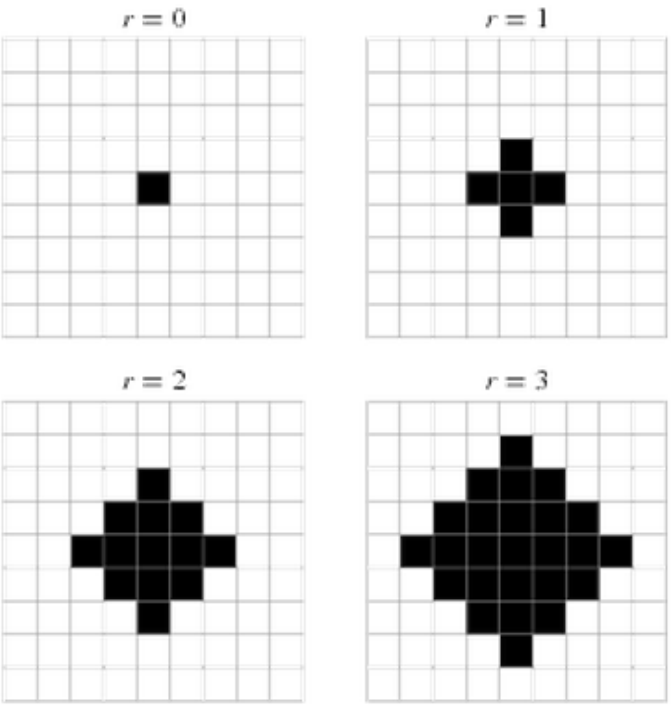
$\longrightarrow$



Decision Map

Motion Model

Von Neumann Neighborhood

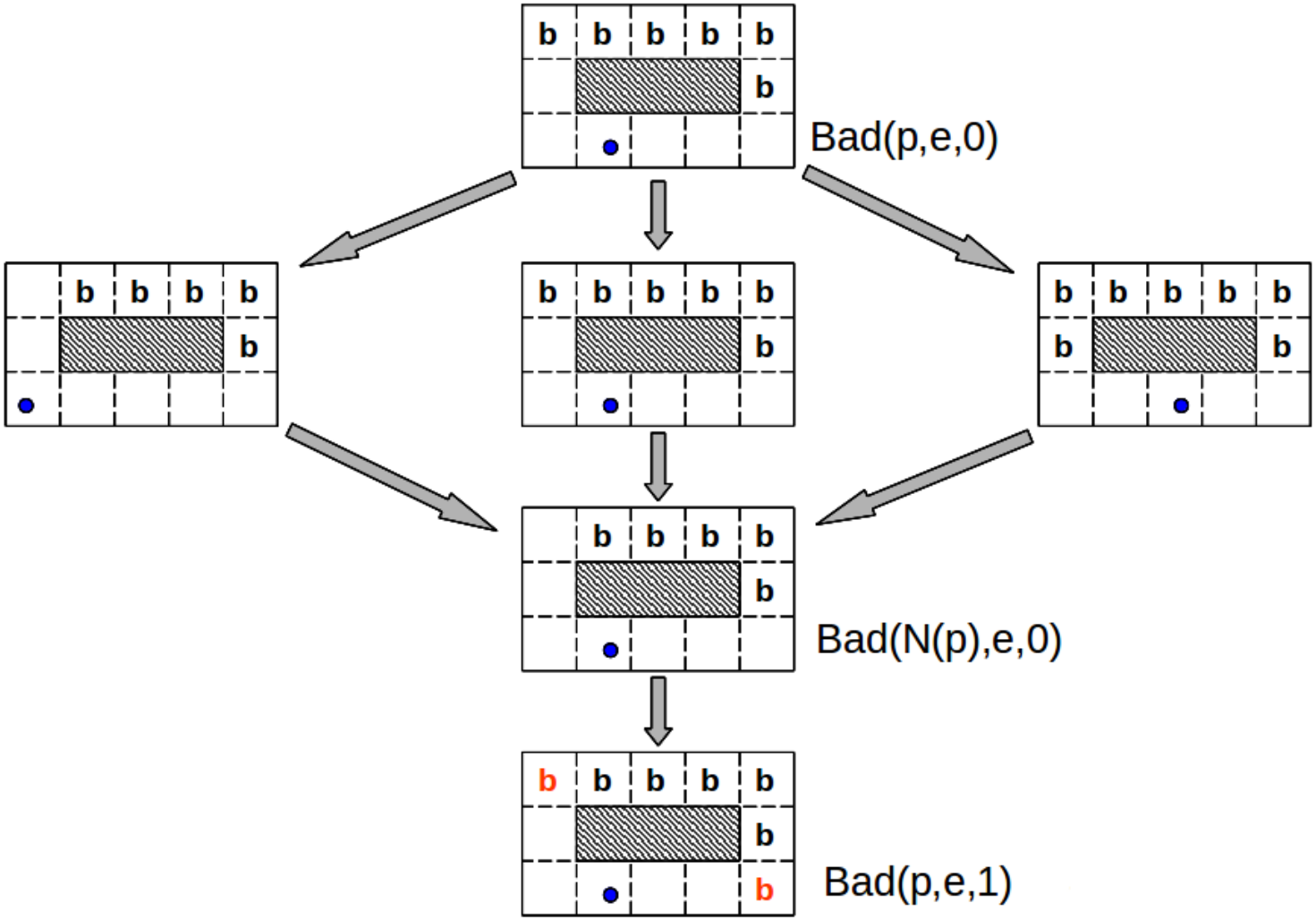


Weisstein, Eric W. "von Neumann Neighborhood." From MathWorld–A Wolfram Web Resource.

http://mathworld.wolfram.com/vonNeumannNeighborhood.html

How To Play

A Minimal Example



Deciding The Winner

Definition (*Bad function*)

$$Bad(p, e, 0) = \begin{cases} 0, & e \text{ is visible to } p, \\ 1, & \text{otherwise}, \end{cases}$$

$$Bad(p, e, i+1) = 1 \quad \forall (p, e)\, \exists e' \in \mathcal{N}(e) \text{ s.t.} \quad \forall p' \in \mathcal{N}(p)\; Bad(p', e', i) = 1$$

Deciding The Winner

Algorithm 1: Decides the game for a given map.

begin

Discretize the map into a uniform *grid* of N cells.

Initialize M and M' to 0.

foreach $(p, e) \in grid \times grid$ **do**

  if e *not visible to* p **then** $M[p, e] = 1$

`// Induction loop`

while $M' \neq M$ **do**

  $M' = M$

  foreach $(p, e) \in grid \times grid$ **do**

    if $\exists e' \in \mathcal{N}(e)\ s.t. \forall p' \in \mathcal{N}(p)\ M'[p', e'] = 1$ **then**

    $M[p, e] = 1$

return M

Correctness and Efficiency

Definition (*Neighborhood Size*)

$$\kappa = max(|\mathcal{N}(p)|, |\mathcal{N}(e)|)$$

Theorem (*Visibility Induction*)

Algorithm 1 decides the discretized game for a general environment in $\mathcal{O}(\kappa^2 N^3)$.

Optimizations

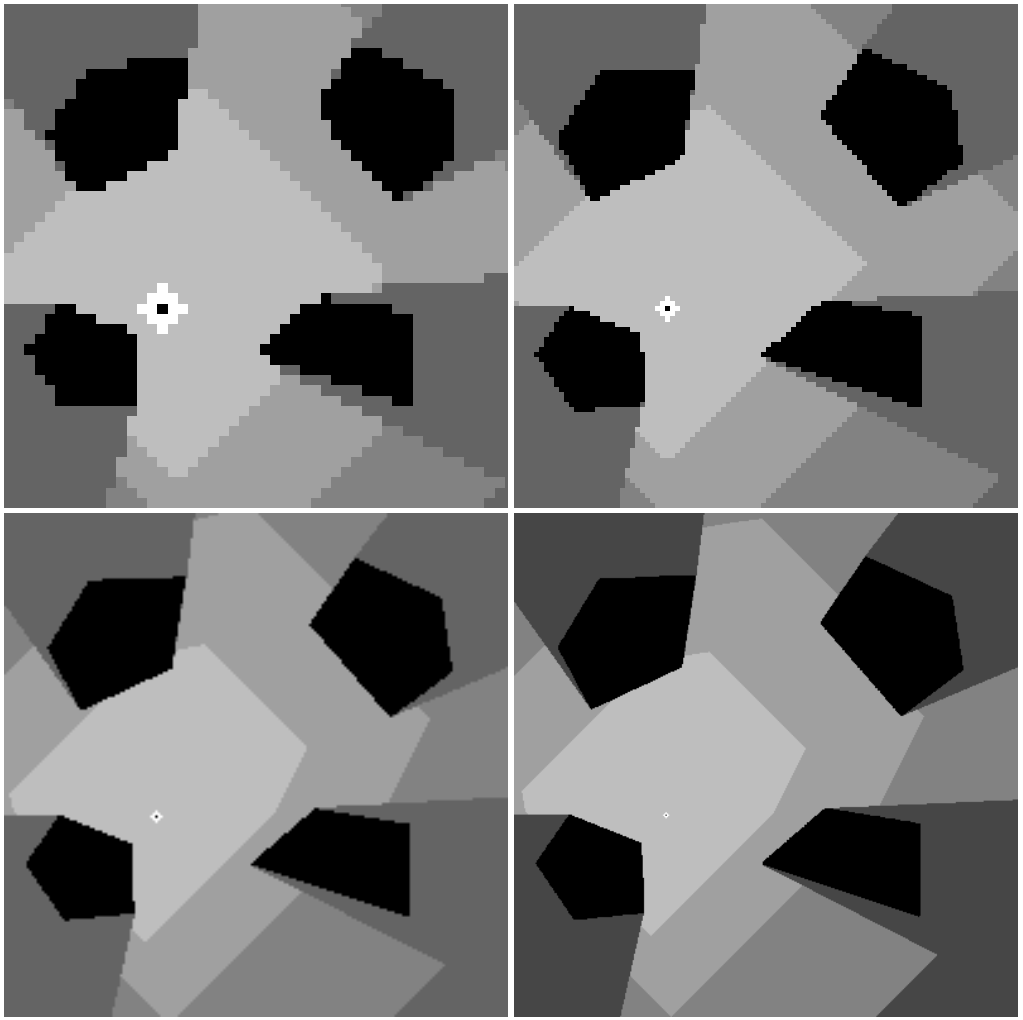
Lemma (*Synchronized Neighborhoods*)

$$\neg Bad(p, e, i) \wedge Bad(p, e, i+1) \implies \exists (p^*, e^*) \in \mathcal{N}(p) \times \mathcal{N}(e) \; s.t. \; \neg Bad(p^*, e^*, i-1) \wedge Bad(p^*, e^*, i)$$

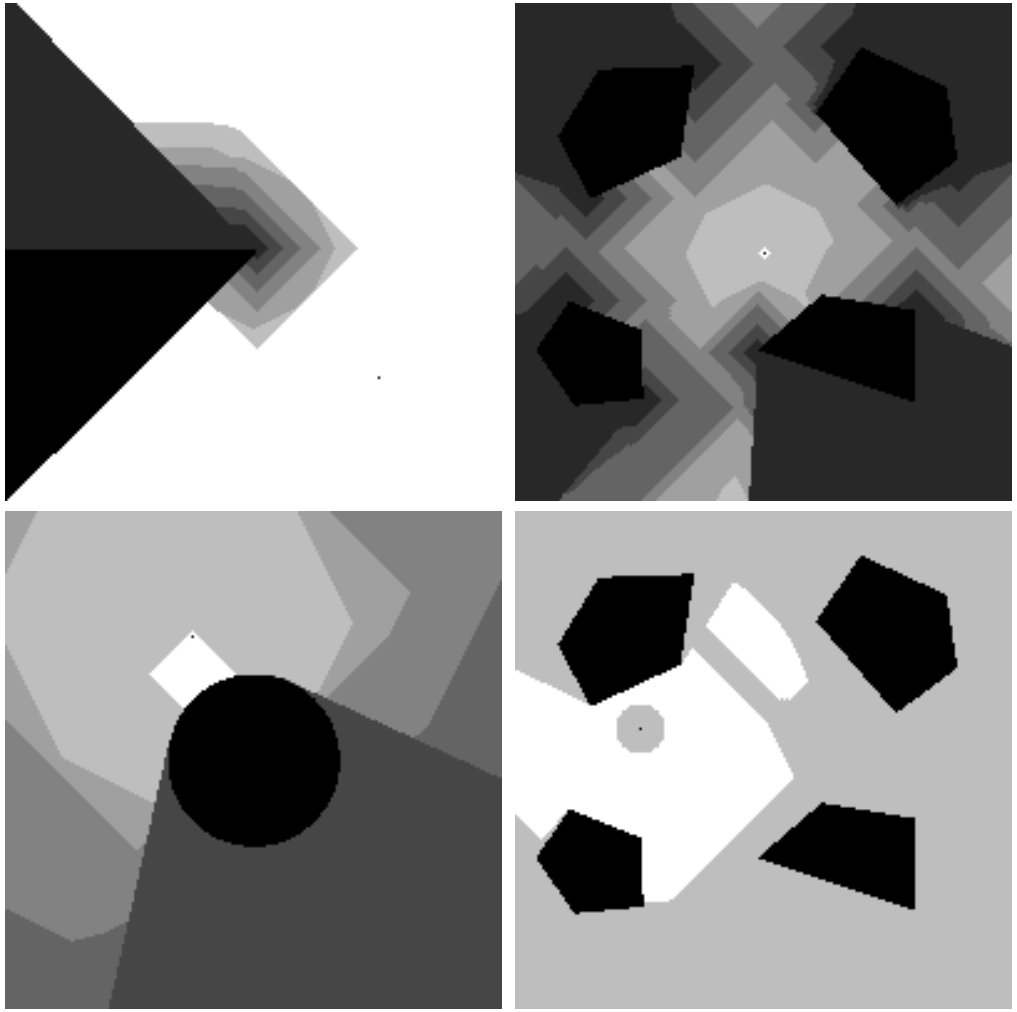
Lemma

The induction loop in Algorithm 1 is only $\mathcal{O}(\kappa^4 N^2)$.

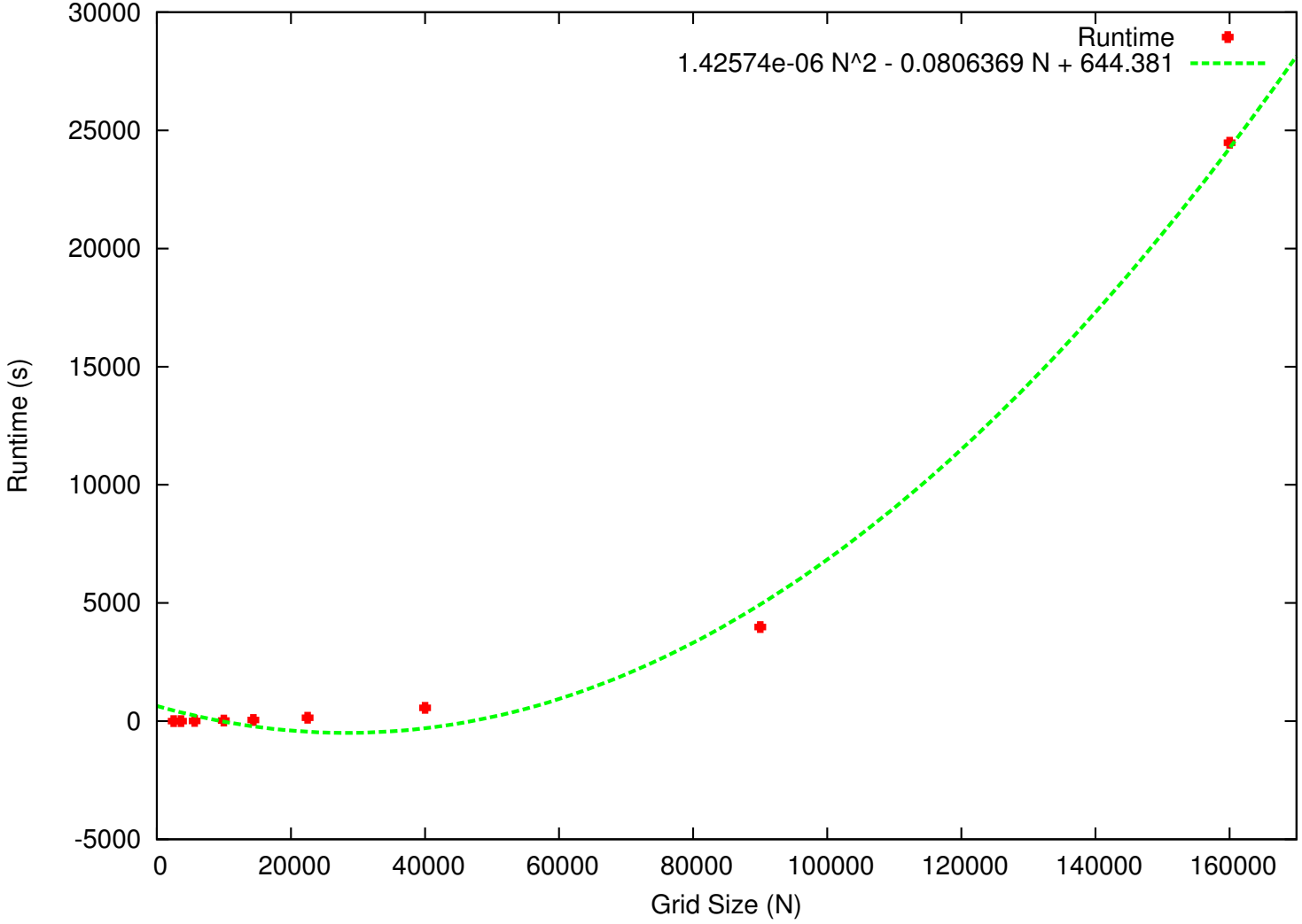
Decision Maps



More Interesting Decision Maps



Runtime


Runtime
1.42574e-06 N^2 - 0.0806369 N + 644.381
Runtime (s)
Grid Size (N)
30000
25000
20000
15000
10000
5000
0
-5000
0
20000
40000
60000
80000
100000
120000
140000
160000

Guaranteed Navigation for Winners

Definition (*Lose function*)

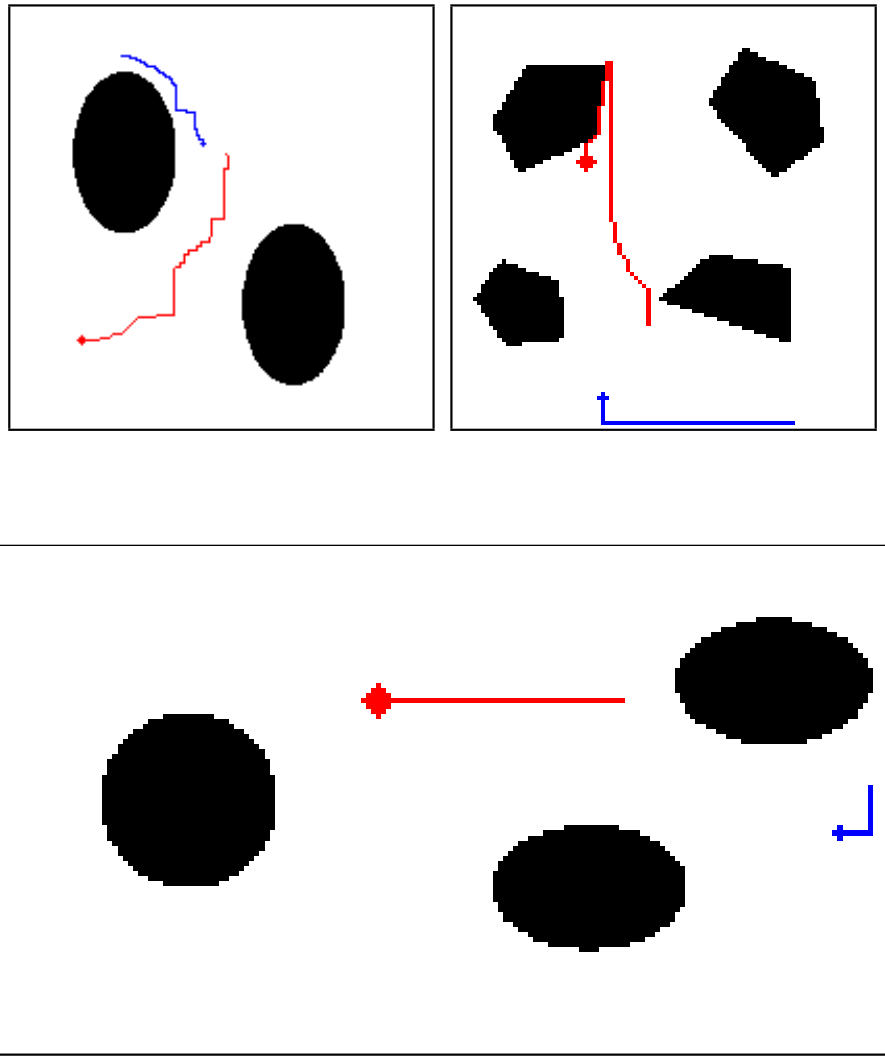
$$Lose(a,b) = (Pursuer(a) \wedge Bad(a,b)) \vee (Evader(a) \wedge \neg Bad(b,a))$$

Algorithm 2: Generic trajectory planning for winners.

Input: $Bad(.,.)$, current state $(player, opponent)$.

```
begin
    N* = {}
    foreach n ∈ N(player) do
        if ¬Lose(n, n') ∀n' ∈ N(opponent) then
            N* = N* ∪ n
    Move to any neighbor in N*.
```

Simulated Tracks



Optimal Trajectory Planning

Definition (*J function*)

$$J(p, e, 0) = \begin{cases} \infty, & e \text{ is visible to } p, \\ 0, & \text{otherwise}, \end{cases}$$

$$J(p, e, i+1) = \min\left(J(p, e, i), 1 + \min_{e' \in \mathcal{N}(e)} \max_{p' \in \mathcal{N}(p)} J(p', e', i)\right)$$

Theorem (Time-Optimal Escape Trajectories)

$J(p, e)$ gives the time left before visibility is broken, assuming both players move optimally.

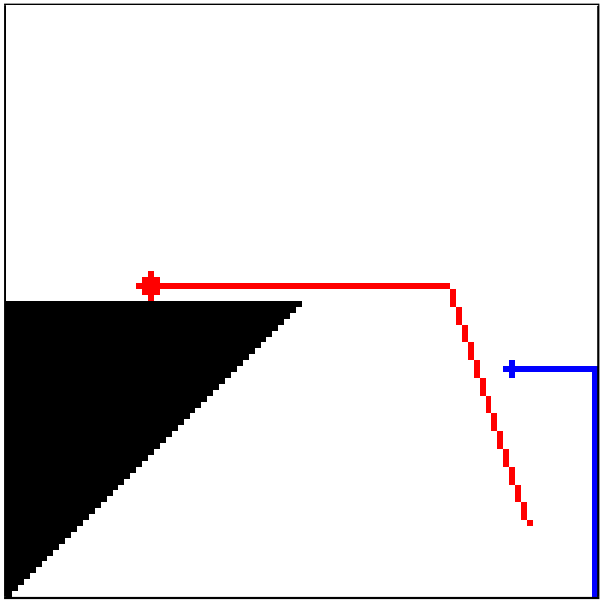
Regaining Lost Visibility

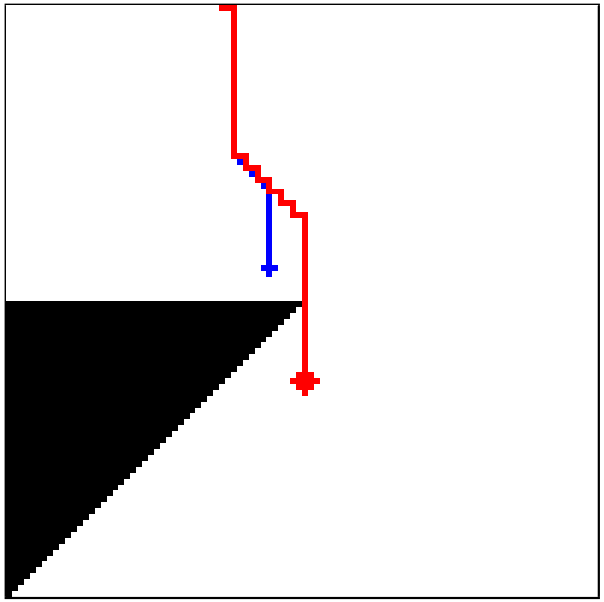
Definition (*Bad_d function*)

$$Bad_d(p, e, 0) = \begin{cases} 0, & e \text{ is visible to } p, \\ 1, & \text{otherwise,} \end{cases}$$

$$\begin{aligned} Bad_d(p, e, i+1) = 0 \quad & \forall (p, e)\, \exists p' \in \mathcal{N}(p) \text{ s.t.} \\ & \forall e' \in \mathcal{N}(e)\; Bad_d(p', e', i) = 0 \\ & \forall i < d \end{aligned}$$

More Interesting Simulated Tracks





Pursuit-Evasion Simulations

Sample Videos

Future Work

Multi-Camera Surveillance

- Large scale experimental realization
- Motion modeling and target path prediction
- Quality of coverage trade-offs e.g. zooming
- High level applications e.g. face and car plate recognition

Visibility-Based Pursuit-Evasion

- Large scale realization
- Error analysis
- Online decision making
- Adaptation to changing environments
- More players

Questions?

abdelkader@alexu.edu.eg

Thank you!