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

Output Assignment
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
Lemma

Fix a positive real \( w_{c,t} \) for each camera \( c \) and target \( t \), such that
\[
\frac{w_{\text{min}}}{w_{\text{max}}} > \frac{N-1}{N}.
\]
Any weight function of the form
\[
f_c(t) = \begin{cases} w_{c,t}, & \text{c covers } t, \\ 0, & \text{otherwise,} \end{cases}
\]
describes an optimal solution maximizing \( \sum_{c \in C} \sum_{t \in T} f_c(t) \).

\[
f_c(t) = \left( \frac{N - 1}{N} \right)^{\theta_c(t)/\theta_{\text{max}}}
\]
Target Angle as a Function of Camera Angle

\[ \theta_c(t) \rightarrow \theta_c(d, t) \]

\[ f_c(d, t) = e^{-\frac{\pi}{2d} \left( \frac{1 - \cos \theta_c(d, t)}{\theta_{\text{max}}} \right)} \]
Coverage by a Single Camera

Numerical Example

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

The graph shows the percent coverage against the number of targets for different algorithms. The algorithms include AS, AR, OPT-8, CGA-8, CFA-8, CFA-32, CFA-1024, CFA-5120. The graph indicates that as the number of targets increases, the percent coverage decreases for all algorithms.
## Results: Coverage Rate

A closer look

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFA-32</td>
<td>71.79</td>
<td>65.47</td>
<td>62.89</td>
<td>61.14</td>
<td>60.08</td>
</tr>
<tr>
<td>CFA-512</td>
<td>73.03</td>
<td>67.46</td>
<td>64.80</td>
<td>62.91</td>
<td>61.83</td>
</tr>
<tr>
<td>CFA-1024</td>
<td>73.11</td>
<td>67.48</td>
<td>64.90</td>
<td>62.97</td>
<td>61.90</td>
</tr>
<tr>
<td>CFA-5120</td>
<td>73.13</td>
<td>67.56</td>
<td>64.92</td>
<td>63.04</td>
<td>61.93</td>
</tr>
<tr>
<td>AS</td>
<td>73.14</td>
<td>67.58</td>
<td>64.93</td>
<td>63.04</td>
<td>61.93</td>
</tr>
</tbody>
</table>
Results: Runtime

![Graph showing runtime as a function of number of static targets for different algorithms and camera array configurations.](chart.png)

- **AS**
- **AR**
- **OPT−8**
- **CFA−8**
- **CFA−32**
- **CFA−64**

The graph plots the average runtime per reassignment (in milliseconds) against the number of static targets. The runtime increases with the number of targets for all algorithms and configurations.
Results: Dynamic Coverage

The diagram illustrates the percent coverage as a function of speed of targets for various algorithms:
- AS
- AR
- OPT−8
- CFA−8
- CFA−32

The x-axis represents the speed of targets, ranging from 0 to 20. The y-axis represents the percent coverage, ranging from 30 to 75.

The graph shows how each algorithm performs under different speeds of targets, with OPT−8 generally maintaining higher coverage compared to the others.
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

- Input frame
- Background subtraction
- Threshold
- Dilation
- Blob Detection
- Blob Correlation
- Frame with detected and labeled blobs
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

Decision Map
Motion Model
Von Neumann Neighborhood


How To Play
A Minimal Example
Definition (*Bad function*)

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

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

forall $(p, e) \in \text{grid} \times \text{grid}$ do

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

// Induction loop

1 while $M' \neq M$ do

2 $M' = M$

3 forall $(p, e) \in \text{grid} \times \text{grid}$ do

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

5 $M[p, e] = 1$

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 \(O(\kappa^2 N^3)\).
Lemma (Synchronized Neighborhoods)

\[ \neg \text{Bad}(p, e, i) \land \text{Bad}(p, e, i + 1) \implies \exists (p^*, e^*) \in \mathcal{N}(p) \times \mathcal{N}(e) \text{ s.t.} \]
\[ \neg \text{Bad}(p^*, e^*, i - 1) \land \text{Bad}(p^*, e^*, i) \]

Lemma

The induction loop in Algorithm 1 is only \( O(\kappa^4 N^2) \).
Decision Maps
More Interesting Decision Maps
Coverage Maximization
Pursuit Evasion
Deciding The Winner
Extensions & Applications

Runtime

Grid Size (N)

Runtime (s)

1.42574e-06 N^2 - 0.0806369 N + 644.381

Ahmed Abdelkader Abdelrazek
Optimization Problems in Visual Surveillance
August 13, 2013 41 / 50
Guaranteed Navigation for Winners

Definition (Lose function)

\[
\text{Lose}(a, b) = (\text{Pursuer}(a) \land \text{Bad}(a, b)) \lor \\
(\text{Evader}(a) \land \neg \text{Bad}(b, a))
\]

Algorithm 2: Generic trajectory planning for winners.

Input: \text{Bad}(., .), current state (player, opponent).

1 begin
2 \quad \mathcal{N}^* = \{
3 \quad \text{foreach } n \in \mathcal{N}(\text{player}) \text{ do }
4 \quad \quad \text{if } \neg \text{Lose}(n, n') \ \forall n' \in \mathcal{N}(\text{opponent}) \text{ then }
5 \quad \quad \quad \mathcal{N}^* = \mathcal{N}^* \cup n
6 \quad \text{Move to any neighbor in } \mathcal{N}^*.
1 end
Simulated Tracks
Optimal Trajectory Planning

Definition (\(J\) function)

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

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

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)

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

\[
\text{Bad}_d(p, e, i + 1) = 0 \quad \forall (p, e) \exists p' \in \mathcal{N}(p) \text{ s.t.} \\
\forall e' \in \mathcal{N}(e) \text{ Bad}_d(p', e', i) = 0 \\
\forall i < d
\]
More Interesting Simulated Tracks
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!