Visible Intelligence

Daniel Chester

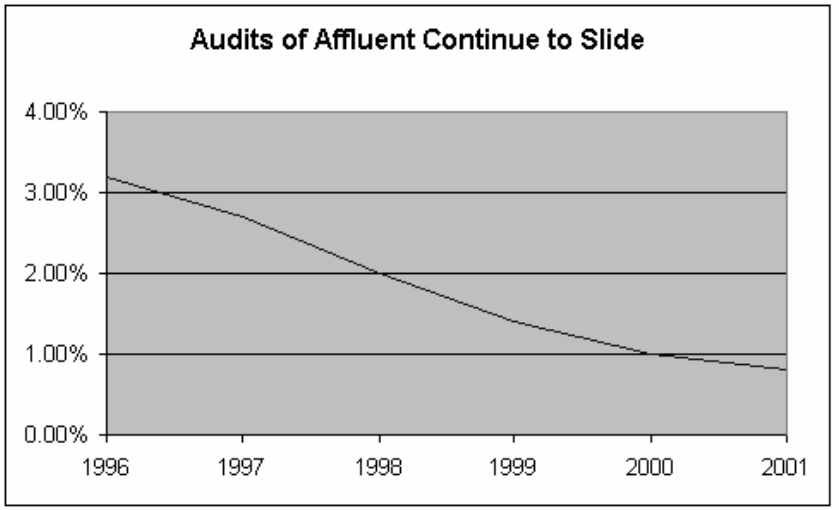
Visible Intelligence

- Extracting intelligence from images
  - Parsing charts and graphs
  - Detecting edges
- Intelligent actions
  - Real-time games
  - Emergent behaviors
  - Game AI generation

Parsing Charts and Graphs

The extraction of information from bar chart and line graph images requires the recognition and interpretation of lines, curves, geometric shapes, words and phrases used in titles and labels, and recognition of common conventions used in charts and graphs.

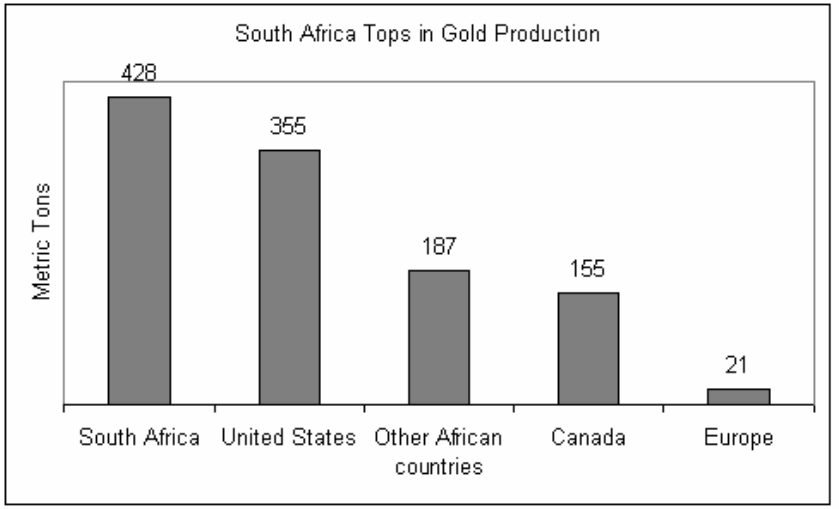
A Line Graph


Audits of Affluent Continue to Slide
4.00%
3.00%
2.00%
1.00%
0.00%
1996
1997
1998
1999
2000
2001

XML Output

```
<InformationGraphic Caption=
“Audits of Affluent Continue to Slide”>
<LineGraph>
<Xaxis>
<Label></Label>
</Xaxis>
...
<line>
<point>
<X>1996</X><Y>3.191</Y>
```

A Bar Chart


South Africa Tops in Gold Production
Metric Tons
428
355
187
155
21
South Africa
United States
Other African countries
Canada
Europe

XML Output

```
<InformationGraph Caption=
“South Africa Tops in Gold Production”>
<BarChart BarDirection=”vertical”>
<MeasurementAxis>
<Label>Metric Tons</Label>
...
<Bar>
<Color>128</Color>
<Annotation><Vaue>428</Value></Annotaton>
</Bar>
```

Edge Detectors That Preserve Fine Detail

Most edge detectors blur images and lose fine detail. We have a pair of edge detectors that preserve fine detail.

Barbie Image

Using Sobel Detector

Note blurred tablecloth and clothing

Our Darkside Detector

Fine detail is clearer

Games Played in Real Time

- While traditional games are turn-based, there is growing interest in games that run in continuous, real time.
  - MMORPG (Massively Multiplayer Online Role Playing Game)
  - LARP (Live Action Role Playing)
  - War games
- Traditional AI game playing assumes discrete, turn-based game play.
- New techniques are needed for continuous, real-time game play.

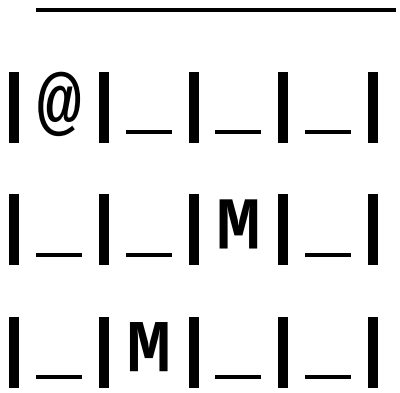
Emergient Behaviors

- Presently, the AI components in games are hand-written solutions to planning problems, e.g., finite state machines, behavior trees, hierarchical task networks.
- Since all possible situations can't be anticipated, such AI is bound to exhibit stupid behavior, such as not going around obstacles created by an opponent, or missing an easy tactical maneuver.
- What is needed is a general AI component that dynamically finds solutions to new problem situations.
- Operating in real time puts additional restrictions on the AI component compared to traditional AI in games.

Emergient Behavior Example

Suppose that @ and Ms are opponents. Each can move to an adjacent location. If that location is occupied, the occupant is captured. Neither @ nor the Ms want to be captured.

It is easy to program the Ms to move toward @, avoid capture, and capture @ if they get the chance. But they will probably stalemate when they could have forced a capture.



One M could possibly sacrifice himself so that the other M can capture @.

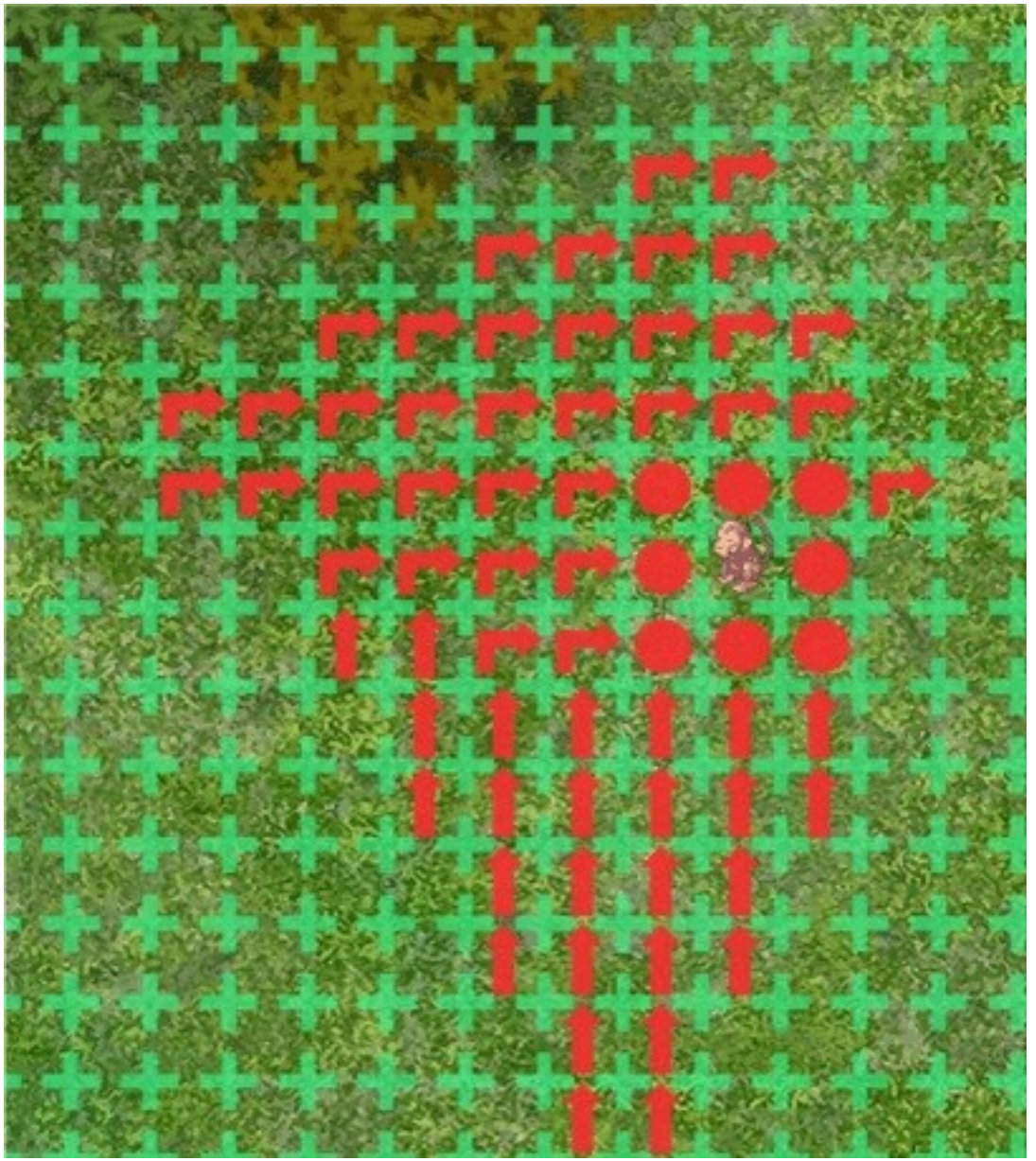
Game AI Generation

- Simplification – Break game into smaller sub-games.
- Simulation – Generate portion of state space in vicinity of terminal states.
- Analysis – Generate tabular representation of near-optimal policy for players using game theory. (Exact optimal policies generally don't exist for recursive games.)
- Synthesis – Generate a program that generalizes the tabular policy representation. We are exploring the use of Inductive Logic Programming for this purpose.

Example: Monkey-catching Orc

- Monkey moves one space orthogonally.
- Orc moves two spaces orthogonally, either in his forward direction, or first turning right and then moving two spaces.
- If the orc is in the same space as the monkey, or he is in an adjacent space (including diagonally), he captures the monkey.
- Each state of the game can be uniquely represented by keeping the monkey at the center of an x,y coordinate system with the orc placed in a relative position such that the orc is facing in the +y direction.

Orc Policy (Turn-based Version)



Orc's Game AI

```
pursuer-policy (x,y):
    if |x| <= 1 and |y| <= 1 then
        stop
    else if y > 1 or (y = 1 and x > 3) then
        move forward 2 spaces
    else
        turn right and move 2 spaces
```

[<x,y> is the location of the monkey relative to the orc being at <0,0>.]

Some Observations

- If the game starts in one of the spaces marked in red, the orc can always catch the monkey.
- If the game starts in one of the spaces not marked in red, the monkey can escape capture forever.
- If the orc and the monkey move simultaneously, the set of starting states where the orc can capture the monkey with a policy consisting of pure strategies is even smaller.
- We are exploring the use of mixed strategies (orc makes probabilistic choices of his move so the monkey can't predict what he is going to do) to see if the range of capture can be increased.