Using Mental Rotation in Primate-inspired Robotic Navigation

Michael J. Pettinati and Ronald C. Arkin
School of Interactive Computing
Georgia Institute of Technology

This research is supported by the Office of Naval Research under grant #00014-11-1-0593
Underlying Question

What role does mental rotation play in navigation, if any, and under what circumstances can it prove useful in support of robotic navigation?
Introduction to Mental Rotation

![Graph](image)

- Displacement from Aligned Orientation (angle)
- Time to Respond
- 180deg
Considerations

• With our navigation algorithm, we are exploring mental rotations using the visual analog approach [Paivio 90, Khooshabeh 90].

• Dealing with scene itself can be complex and the mechanism of mental rotation can be fragile [Aretz & Wickens 92, Bläsing, de Castro Campos, Schack, & Brugger 12]

• To deal with this complexity humans often segment the object (scene) into parts and rotate these independently [Khooshabeh & Hegarty 10]
Sequential Process for Mental Rotations

[Johnson 90]

1. Form (encode) a mental representation of an object (scene)
2. Rotate the object mentally until axial orientation allows the comparison to the standard
3. Make the comparison
4. Make a judgment
5. Report a decision
Navigation Algorithm

Differs from visual servoing [Franz 96, Moller 08, Sato 12] as higher-level, abstract representation is used

More biologically plausible than previous approach, which projected information about scene into ground plane. This approach works directly with scene.
Scenes Rigorously Tested

Scene 1

Scene 2
Segmentations of Goal Views

Noisy result due to limited smoothing of input RGBD image.

Normal Estimation

Correspondence

• Bootstrap
  – Human user matches “key” segments that result from the segmentation of the goal image and the segmentation of the view from the start location.

• Feedforward
  – The vision algorithm uses information about how it has instructed the robotic agent to move as well as the previous positions of “key” segments to determine matching segments.
Visualizing at Aligned Orientation

Current View

Goal View
Visualizing at Aligned Orientation (cont.)

Compute Average Normal Vector of each "Key" Segment
Deciding how to Move

• Current implementation has the agent orienting itself (when not at aligned orientation) and reassessing direction in order to avoid losing track of the “key” segments being used to navigate because of limited horizontal field of view of the Kinect.

• Agent moves quickly left/right when possible to avoid being turned away from the segments later in the navigation when they can be more easily lost from view.
Segments Used to Navigate

Scene 1

Scene 2
Test Cases

Scene

(-0.75, 1.75), 15deg right

(-0.25, 2.0), 20deg left

(0.0, 2.5)

(0.5, 1.5), 10deg right

(0.5, 2.0)

Goal

First Scene

Second Scene
## Results

### First Scene

<table>
<thead>
<tr>
<th>Location</th>
<th>Success</th>
<th>Direction Updated on Average</th>
<th>Avg. Rotational Offset from Goal Orientation</th>
<th>Avg. Depth from Goal</th>
<th>Avg. Horizontal Displacement from Goal</th>
<th>Avg. Dist. from Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location 1</td>
<td>70%</td>
<td>5.8</td>
<td>$7^\circ\pm4.74^\circ$</td>
<td>13.08cm±8.25cm</td>
<td>16.49cm±8.25cm</td>
<td>21.96cm±8.25cm</td>
</tr>
<tr>
<td>Location 2</td>
<td>70%</td>
<td>7.4</td>
<td>$1.38^\circ\pm7.48^\circ$</td>
<td>12.21cm±10.09cm</td>
<td>-6.63cm±15.44cm</td>
<td>22.65cm±4.47cm</td>
</tr>
<tr>
<td>Location 3</td>
<td>90%</td>
<td>6.3</td>
<td>$6.8^\circ\pm4.66^\circ$</td>
<td>8.65cm±10.83cm</td>
<td>-4.18cm±7.83cm</td>
<td>13.48cm±9.44cm</td>
</tr>
</tbody>
</table>

### Second Scene

<table>
<thead>
<tr>
<th>Location</th>
<th>Success</th>
<th>Direction Updated on Average</th>
<th>Avg. Rotational Offset from Goal Orientation</th>
<th>Avg. Depth from Goal</th>
<th>Avg. Horizontal Displacement from Goal</th>
<th>Avg. Dist. from Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location 1</td>
<td>80%</td>
<td>3.7</td>
<td>$-1.5^\circ\pm6.09^\circ$</td>
<td>4.34cm±11.74cm</td>
<td>34.19cm±10.19cm</td>
<td>36.6cm±10.19cm</td>
</tr>
<tr>
<td>Location 2</td>
<td>90%</td>
<td>6.4</td>
<td>$-1.4^\circ\pm5.41^\circ$</td>
<td>16.89cm±14.03cm</td>
<td>-30.47cm±12.45cm</td>
<td>37.73cm±11.93cm</td>
</tr>
<tr>
<td>Location 3</td>
<td>100%</td>
<td>5.3</td>
<td>$-0.4^\circ\pm4.65^\circ$</td>
<td>13.26cm±10.03cm</td>
<td>9.7cm±15.48cm</td>
<td>21.09cm±12.86cm</td>
</tr>
</tbody>
</table>
Results Overview

• 83.33% success rate
  – Success rate rises to 90% from 76.67% when transition between Scene 1 and Scene 2.

• 70% of all failures attributable to the feedforward matching step.

• Improvement when agent was moved closer to scene (kept within 4m of scene).
  – Improvement comes even when testing on more complex scene.
  – Kinect provides reliable depth up to 4m.
Allowing for Waypoints

Start

Select Next Waypoint

Goal Segmentation → Compute Goal Normals

Capture Current View → Current Segmentation → Compute Current Normals

Not at Goal

Send Motion Vector → Assess Goal Direction → Visualize at Aligned Orientation (Mentally Rotate Scene)

Not At Goal

At Goal

Bootstrap/FeedForward Match

Align Avg. Normals (Determine Relative Orientation)

Stop
Introduction of Waypoints
Navigating Using Multiple Segments
Future Work

• Enhance bootstrap step to better fit advice giving context
• Segmentation that is not strictly planar/object recognition
  • Will require changes to mental rotation step.
  • Will make algorithm better fit notion of advice giving.