Problem Statement

Real-world robotic applications typically require basic actions:
- Grasping an object, docking to a recharge station, etc.
- Typically uses FSM: slow to program, to debug, to tune.

We want to specify these actions by demonstrating them:
- Trajectory following replay (e.g. GMR) are typically reactive.
- Sequence segmentation demands a definition of step changes.
- Both approaches require many parameters/design choices.

Experiments

- 20 recorded trajectories (black dots), 3 x 5 different tested positions (blue)
- \( \theta_i = 0.05 \), 7 different tested values for \( \theta_i \)

Results

- For \( \theta_i \) comprised between 0.6-0.5 and 1.2-2, the success rate is high at 90%.
- A large \( \theta_i \) is better because training runs differ mostly at the beginning.
- In case of success, the alignment error is small and the duration constant.
- Most of the failures (60%) are linked to the controller stopping the robot indefinitely, due to a fixed or cyclic distribution on \( \theta_i \).
- With only 6 recorded trajectories, performances degrade gracefully: there are more failures but on successful runs mean error and duration are similar.

Example of Model Execution

<table>
<thead>
<tr>
<th>( \tau )</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \theta )</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- \( \tau \) : Step phase
- \( \theta \) : Robot obs.
- \( \tau \) : Latent cmd

Current and Future Work

- Test on search-and-rescue robot, PR2
- Observation model (Cauchy) and transition model (Log-normal)
- Sensor weighting (feature selection)
- Abstraction (similar subsequences, loops, branching)

Conclusion

- Real-time on laptop in Python/Cython, complexity: \( O(L \times N) \)
- Successful application to cube grasping
- Strong potential for other types of robots