Combining Optimal Control and Learning for Visual Navigation in Novel Environments

Bansal, Tolani et al.

Dylan Green
August 26, 2020
Introduction
So far...

- Fundamentals - Engineering Perspective (Cathy)
- Fundamentals - Optimization Perspective (Ben)
- Iterative LQR and Guided Policy Search (Betty)
- Sample Complexity of LQR (Joey)
- Model Predictive Control and Safe RL (Fred)
So far...

- Fundamentals - Engineering Perspective (Cathy)
- Fundamentals - Optimization Perspective (Ben)
- Iterative LQR and Guided Policy Search (Betty)
- Sample Complexity of LQR (Joey)
- Model Predictive Control and Safe RL (Fred)

Today...

- Robots 😊
Goal

- Autonomous, vision-based navigation in cluttered indoor environments
Goal

- Autonomous, vision-based navigation in cluttered indoor environments
- Factorized approach: learning is used to make high level navigational decisions, optimal control used to produce smooth trajectories
Goal

First Person View

Top View

Goal
Start
Door
Approach
System Diagram

Perception Module

Image taken once every $H$ time steps

Linear & Angular speed
$u_t = (v_t, \omega_t)$

Waypoint
$\hat{w}_t = (\hat{x}_t, \hat{y}_t, \hat{\theta}_t)$

Goal Position
$p_t^* = (x_t^*, y_t^*)$

Dynamics-based Planning Module

Control applied for $H$ time steps

Trajectory Tracking Feedback Controller

State feedback

Desired Trajectory
Problem Setup

- Notation and Terminology:
  - State: $z_t = (x_t, y_t, \phi_t)$
Problem Setup

- Notation and Terminology:
  - State: \( z_t = (x_t, y_t, \phi_t) \)
  - Position: \( p_t = (x_t, y_t) \)
Problem Setup

- Notation and Terminology:
  - State: $z_t = (x_t, y_t, \phi_t)$
  - Position: $p_t = (x_t, y_t)$
  - Goal position: $p^* = (x^*, y^*)$
Problem Setup

- Notation and Terminology:
  - State: $z_t = (x_t, y_t, \phi_t)$
  - Position: $p_t = (x_t, y_t)$
  - Goal position: $p^* = (x^*, y^*)$
  - Control input: $u_t = (v_t, w_t)$
Problem Setup

• Notation and Terminology:
  • State: \( z_t = (x_t, y_t, \phi_t) \)
  • Position: \( p_t = (x_t, y_t) \)
  • Goal position: \( p^* = (x^*, y^*) \)
  • Control input: \( u_t = (v_t, w_t) \)

• Assumed dynamics:
Problem Setup

- **Notation and Terminology:**
  - State: \( z_t = (x_t, y_t, \phi_t) \)
  - Position: \( p_t = (x_t, y_t) \)
  - Goal position: \( p^* = (x^*, y^*) \)
  - Control input: \( u_t = (v_t, w_t) \)

- **Assumed dynamics:**
  - \( \dot{x} = v \cos \phi, \quad \dot{y} = v \sin \phi, \quad \dot{\phi} = \omega \)
Problem Setup

- Notation and Terminology:
  - State: \( z_t = (x_t, y_t, \phi_t) \)
  - Position: \( p_t = (x_t, y_t) \)
  - Goal position: \( p^* = (x^*, y^*) \)
  - Control input: \( u_t = (v_t, w_t) \)

- Assumed dynamics:
  - \( \dot{x} = v \cos \phi, \quad \dot{y} = v \sin \phi, \quad \dot{\phi} = \omega \)
  - \( v \in [0, \bar{v}], \quad \omega \in [-\bar{\omega}, \bar{\omega}] \)
A CNN which takes as input

- A 224 × 224 pixel RGB image
- The target position $p_t^*$
- The robot's current linear and angular speed $u_t$

and outputs a waypoint

$$\hat{w}_t := (\hat{x}_t, \hat{y}_t, \hat{\phi}_t) = \psi(l_t, u_t, p_t^*)$$
Planning and Control Module

- Given a waypoint $\hat{w}_t$ and the current linear and angular speed $u_t$, the planning module designs a smooth trajectory (in terms of both position and speed) from the current position to the waypoint.
Planning and Control Module

• Given a waypoint $\hat{w}_t$ and the current linear and angular speed $u_t$, the planning module designs a smooth trajectory (in terms of both position and speed) from the current position to the waypoint.

• A spline-based planner provides desired state and control trajectories

$$\{z^*, u^*\}_{t:t+H} = \text{FitSpline}(\hat{w}_t, u_t).$$
Planning and Control Module

- Given a waypoint $\hat{w}_t$ and the current linear and angular speed $u_t$, the planning module designs a smooth trajectory (in terms of both position and speed) from the current position to the waypoint.

- A spline-based planner provides desired state and control trajectories

$$\{ z^*, u^* \}_{t:t+H} = \text{FitSpline}(\hat{w}_t, u_t).$$

- An LQR-based feedback controller

$$\{ k, K \}_{t:t+H} = \text{LQR}(z^*_{t:t+H}, u^*_{t:t+H})$$

tracks the generated trajectory.
Expert Supervision

- A Model Predictive Control (MPC) scheme is used to generate expert supervision for training the perception module.
Expert Supervision

- A Model Predictive Control (MPC) scheme is used to generate expert supervision for training the perception module.
- Output is a sequence of dynamically feasible waypoints and corresponding spline trajectories.
Expert Supervision

- A Model Predictive Control (MPC) scheme is used to generate expert supervision for training the perception module.
- Output is a sequence of dynamically feasible waypoints and corresponding spline trajectories.
- Optimal trajectories can be found in the training phase as this process is done in simulation with perfect knowledge of the environment.
Cost Function

Cost function for trajectory is

\[ J(z, u) = \sum_{i=0}^{T} J_i(z_i, u_i) \]

where

\[ J_i(z_i, u_i) := \left( \max \left\{ 0, \lambda_1 - d^{obs}(x_i, y_i) \right\} \right)^3 + \lambda_2 \left( d^{goal}(x_i, y_i) \right)^2 \]

and

- \( d^{obs}(x_i, y_i) \) is the distance to the nearest obstacle at time \( i \)
- \( d^{goal}(x_i, y_i) \) is the minimum collision-free distance to the goal position
- \( \lambda_1 \) is the minimum allowable distance to an obstacle
MPC Problem

Given the cost function in the last slide, the MPC problem is

\[
\min_{z,u} J(z, u)
\]

subject to constraints

\[
x_{i+1} = x_i + \Delta T v_i \cos \phi_i, \quad y_{i+1} = y_i + \Delta T v_i \sin \phi_i, \quad \phi_{i+1} = \phi_i + \Delta T \omega_i
\]

\[
v_i \in [0, \bar{v}], \quad \omega_i \in [-\bar{\omega}, \bar{\omega}]
\]

\[
z_0 = (0, 0, 0), \quad u_0 = (0, 0)
\]
Starting from $i = 0$, we solve the optimization problem on the last slide in a receding horizon fashion. That is, for a timestep $i = t$ we solve

$$
\min_{\hat{w}_t} \sum_{i=t}^{t+H} J_i (z_i, u_i)
$$

subject to

$$
\{z, u\}_{t:t+H} = \text{FitSpline} (\hat{w}_t, u_t),
$$

$$
z_t, u_t \text{ - Given}
$$

where

- $\hat{w}_t = (\hat{x}_t, \hat{y}_t, \hat{\phi}_t)$ is the waypoint
- $\{z, u\}_{t:t+H}$ is the corresponding trajectory
Solved approximately using sampling-based approach. Specifically,

- Sample waypoints within ground-projected field-of-view.
Solved approximately using sampling-based approach. Specifically,

- Sample waypoints within ground-projected field-of-view.
- Apply optimal control sequence $u^*$ for time horizon $[t, t + H]$ to obtain state $z^*_{t+H}$.
Solved approximately using sampling-based approach. Specifically,

- Sample waypoints within ground-projected field-of-view.
- Apply optimal control sequence \( u^* \) for time horizon \([t, t + H]\)
  to obtain state \( z^*_{t+H} \)
- Compute cost of sequence \( \{z, u\}_{t:t+H} \)
Solved approximately using sampling-based approach. Specifically,

- Sample waypoints within ground-projected field-of-view.
- Apply optimal control sequence $u^*$ for time horizon $[t, t + H]$ to obtain state $z^*_{t+H}$
- Compute cost of sequence $\{z, u\}_{t:t+H}$
- Choose lowest cost sample $\hat{w}^*_t$
Solved approximately using sampling-based approach. Specifically,

- Sample waypoints within ground-projected field-of-view.
- Apply optimal control sequence $u^*$ for time horizon $[t, t + H]$ to obtain state $z^*_{t+H}$
- Compute cost of sequence $\{z, u\}_{t:t+H}$
- Choose lowest cost sample $\hat{w}^*_t$
- Repeat starting from time $t + H$
The solution to each instance of this optimization problem gives a training example comprised of:

- The image obtained at state $z_t^*, I_t$
- The relative goal position $p_t^*$
- The speed of the robot, $u_t^*$
- The optimal waypoint $\hat{w}_t^*$
**Algorithm 1** Model-based Navigation via Learned Waypoint Prediction

**Require:** \( p^* := (x^*, y^*) \)  \hspace{1cm} ▷ Goal location

1: \textbf{for} \( t = 0 \) to \( T \) \textbf{do}

\[
\begin{align*}
2: &\quad z_t := (x_t, y_t, \phi_t); \quad u_t := (v_t, \omega_t) \quad \text{▷ Measured robot pose, and linear and angular speed} \\
3: &\quad \textbf{Every} \ H \text{ steps} \quad \text{▷ Replan every} \ H \text{ steps} \\
4: &\quad p_t^* := (x_t^*, y_t^*) \quad \text{▷ Goal location in the robot’s coordinate frame} \\
5: &\quad \hat{w}_t = \psi(I_t, u_t, p_t^*) \quad \text{▷ Predict next waypoint} \\
6: &\quad \{z^*, u^*\}_{t:t+H} = \text{FitSpline}(\hat{w}_t, u_t) \quad \text{▷ Plan spline-based smooth trajectory} \\
7: &\quad \{k, K\}_{t:t+H} = \text{LQR}(z^*_{t:t+H}, u^*_{t:t+H}) \quad \text{▷ Tracking controller} \\
8: &\quad u_{t+1} = K_t(z_t - z_t^*) + k_t \quad \text{▷ Apply control} \\
9: &\textbf{end for}
\end{align*}
\]
Experiments
• End-to-End Learning
Alternative Approaches

• End-to-End Learning
  • A CNN which directly outputs velocity commands corresponding to the optimal trajectories output by the spine based planner.
Alternative Approaches

- **End-to-End Learning**
  - A CNN which directly outputs velocity commands corresponding to the optimal trajectories output by the spine-based planner.
  - Does not explicitly use any system knowledge at test time.
Alternative Approaches

- **End-to-End Learning**
  - A CNN which directly outputs velocity commands corresponding to the optimal trajectories output by the spine based planner.
  - Does not explicitly use any system knowledge at test time.

- **Geometric Mapping and Planning**
Alternative Approaches

- **End-to-End Learning**
  - A CNN which directly outputs velocity commands corresponding to the optimal trajectories output by the spine based planner.
  - Does not explicitly use any system knowledge at test time.

- **Geometric Mapping and Planning**
  - Learning free, purely geometric
Alternative Approaches

• **End-to-End Learning**
  - A CNN which directly outputs velocity commands corresponding to the optimal trajectories output by the spine based planner.
  - Does not explicitly use any system knowledge at test time.

• **Geometric Mapping and Planning**
  - Learning free, purely geometric
  - Simulation: ideal depth images are provided
Alternative Approaches

- **End-to-End Learning**
  - A CNN which directly outputs velocity commands corresponding to the optimal trajectories output by the spine based planner.
  - Does not explicitly use any system knowledge at test time.

- **Geometric Mapping and Planning**
  - Learning free, purely geometric
  - Simulation: ideal depth images are provided
  - Hardware: RGB-D images processed by RTAB-Map package
Alternative Approaches

• **End-to-End Learning**
  • A CNN which directly outputs velocity commands corresponding to the optimal trajectories output by the spine based planner.
  • Does not explicitly use any system knowledge at test time.

• **Geometric Mapping and Planning**
  • Learning free, purely geometric
  • Simulation: ideal depth images are provided
  • Hardware: RGB-D images processed by RTAB-Map package
  • Memory vs. Memoryless
Alternative Approaches

- **End-to-End Learning**
  - A CNN which directly outputs velocity commands corresponding to the optimal trajectories output by the spine based planner.
  - Does not explicitly use any system knowledge at test time.

- **Geometric Mapping and Planning**
  - Learning free, purely geometric
  - Simulation: ideal depth images are provided
  - Hardware: RGB-D images processed by RTAB-Map package
  - Memory vs. Memoryless
  - Uses the same spline-based planner
Simulation - Setup

- Experiments conducted in environments derived from 3D scans of real world buildings.
• Experiments conducted in environments derived from 3D scans of real world buildings.
• Scans from two buildings used for training
Simulation - Setup

- Experiments conducted in environments derived from 3D scans of real world buildings.
- Scans from two buildings used for training
- 185 test episodes from a third, held-out building used for testing
Simulation - Setup

- Experiments conducted in environments derived from 3D scans of real world buildings.
- Scans from two buildings used for training
- 185 test episodes from a third, held-out building used for testing
- Test episodes sampled to include scenarios such as: avoiding obstacles, leaving/entering rooms, using hallways, etc.

Metrics: success rate, average time to reach goal, average acceleration and jerk
Simulation - Setup

- Experiments conducted in environments derived from 3D scans of real world buildings.
- Scans from two buildings used for training
- 185 test episodes from a third, held-out building used for testing
- Test episodes sampled to include scenarios such as: avoiding obstacles, leaving/entering rooms, using hallways, etc.
- Metrics: success rate, average time to reach goal, average acceleration and jerk
## Simulation - Quantitative Results

<table>
<thead>
<tr>
<th>Agent</th>
<th>Input</th>
<th>Success (%)</th>
<th>Time taken (s)</th>
<th>Acceleration ($m/s^2$)</th>
<th>Jerk ($m/s^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert</td>
<td>Full map</td>
<td>100</td>
<td>10.78 ±2.64</td>
<td>0.11 ±0.03</td>
<td>0.36 ±0.14</td>
</tr>
<tr>
<td>LB-WayPtNav (our)</td>
<td>RGB</td>
<td>80.65</td>
<td>11.52 ±3.00</td>
<td>0.10 ±0.04</td>
<td>0.39 ±0.16</td>
</tr>
<tr>
<td>End To End</td>
<td>RGB</td>
<td>58.06</td>
<td>19.16 ±10.45</td>
<td>0.23 ±0.02</td>
<td>8.07 ±0.94</td>
</tr>
<tr>
<td>Mapping (memoryless)</td>
<td>Depth</td>
<td>86.56</td>
<td>10.96 ±2.74</td>
<td>0.11 ±0.03</td>
<td>0.36 ±0.14</td>
</tr>
<tr>
<td>Mapping</td>
<td>Depth + Spatial Memory</td>
<td>97.85</td>
<td>10.95 ±2.75</td>
<td>0.11 ±0.03</td>
<td>0.36 ±0.14</td>
</tr>
</tbody>
</table>
Simulation - Qualitative Results

![Graph showing simulation results with 'Proposed' and 'End To End' lines.](image-url)
Simulation - Qualitative Results

Task 1
Task 2
Task 3

LB-WayPtNav

E2E
• Network trained in simulation is deployed directly on hardware testbed with no additional training.
• Network trained in simulation is deployed directly on hardware testbed with no additional training.
• Testing took place in two different buildings, neither of which is in the training dataset.
● Network trained in simulation is deployed directly on hardware testbed with no additional training.
● Testing took place in two different buildings, neither of which is in the training dataset.
● On-board odometry is used for state measurement.
Hardware - Setup

- Network trained in simulation is deployed directly on hardware testbed with no additional training.
- Testing took place in two different buildings, neither of which is in the training dataset.
- On-board odometry is used for state measurement.
- 4 different experiments, each repeated 5 times for each method.
Video
## Hardware - Quantitative Results

<table>
<thead>
<tr>
<th>Agent</th>
<th>Input</th>
<th>Success (%)</th>
<th>Time taken (s)</th>
<th>Acceleration ( m/s^2 )</th>
<th>Jerk ( m/s^3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB-WayPtNav (our)</td>
<td>RGB</td>
<td>95</td>
<td>22.93 ±2.38</td>
<td>0.09 ±0.01</td>
<td>3.01 ±0.38</td>
</tr>
<tr>
<td>End To End</td>
<td>RGB</td>
<td>50</td>
<td>33.88 ±3.01</td>
<td>0.19 ±0.01</td>
<td>6.12 ±0.18</td>
</tr>
<tr>
<td>Mapping (memoryless)</td>
<td>RGB-D</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Mapping</td>
<td>RGB-D + Spatial Memory</td>
<td>40</td>
<td>22.13 ±0.54</td>
<td>0.11 ±0.01</td>
<td>3.44 ±0.21</td>
</tr>
</tbody>
</table>
• Combining learning and optimal control gets the best of both worlds; semantic understanding of navigational cues and smooth, robust trajectories
Summary of Results

- Combining learning and optimal control gets the best of both worlds; semantic understanding of navigational cues and smooth, robust trajectories
- More reliable and efficient at reaching goals than comparable methods
Summary of Results

• Combining learning and optimal control gets the best of both worlds; semantic understanding of navigational cues and smooth, robust trajectories

• More reliable and efficient at reaching goals than comparable methods

• Can be directly transferred from simulation to hardware in previously unseen environments
Dynamic Environments

Video
Questions/Discussion
Appendix
The choice of $\hat{\phi}$ in $\hat{w}$ provides an additional degree of freedom, allowing the robot to select collision free trajectories.
Training vs. Test Areas

(a) Training

(b) Test
Data Augmentation

(a) Undistorted image  
(b) Adding superpixels  
(c) Adding Gaussian blur  
(d) Adding motion blur  
(e) Image sharpening  
(f) Adding Gaussian noise  
(g) Changing brightness  
(h) Dropping pixels  
(i) Changing saturation  
(j) Changing contrast  
(k) Changing field-of-view  
(l) Changing camera tilt