EE 290-005 – Integrated Perception, Learning, and Control **Spring 2021**

Lecture 7: Learning and Predictive Control

Scribes: Michael H. Lim, Yiling You, Yaodong Yu

Presentation by Jay Monga, Tiffany Cappellari and Valmik Prabhu.

7.1 Combining Optimal Control and Learning for Visual Navigation in Novel Environments

7.1.1 Research Questions

The main research questions studied in this paper are the following:

- How can a robot autonomously navigate in an unknown environment?
- What if it is supposed to accomplish some tasks?
- What if there are obstacles that make planning nontrivial?

7.1.2 Related Work

7.1.2.1 Classical Robot Navigation

The classical robotics approaches adopt the *Map-then-Plan* idea, where we take the following approach:

- A map of the environment is first created using RGB / RGB-D images / LiDAR scans.
- Model-based control techniques are used for navigating that environment.

The estimation of 3D structure of the environment can be accomplished through the use of SLAM algorithms, as was discussed by Frank and Amay in week 4. Once a map of the environment has been made, classical planning approaches like optimal control can be leveraged. Plans generated through optimal control are smooth, dynamically feasible for the robot, and can be robust to disturbances.

Concerns with this method include:

- Techniques to estimate depth from RGB images are sensitive to lighting conditions and object textures.
- Algorithms that involve collecting lots of depth information to generate a 3D map can be computationally intensive, and maybe even unnecessary depending on the required navigation task.

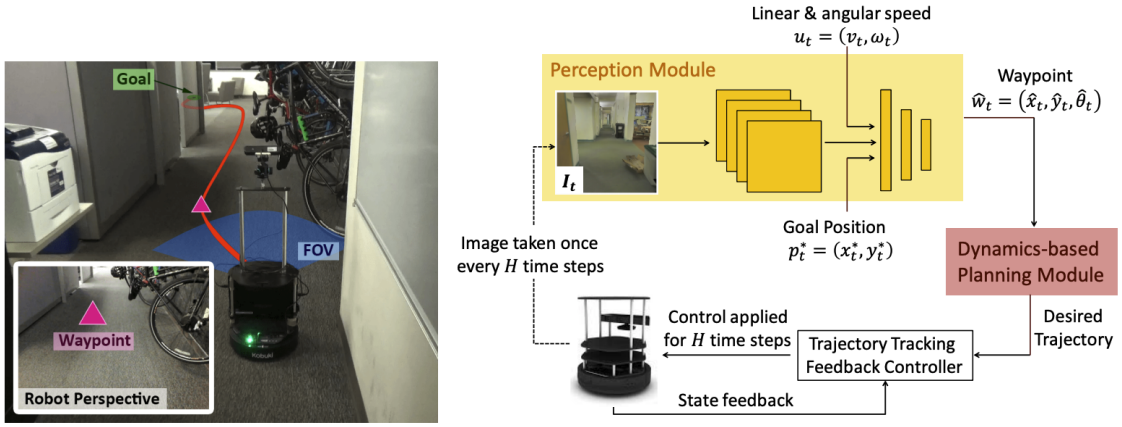

Figure 7.1: Overview of LB-WayPtNav framework, highlighting the interface between perception and planning/control modules.

7.1.2.2 End-to-End (E2E) Learning for Navigation

The End-to-End (E2E) learning involves learning to predict appropriate control signals or local plans for a robot *directly* from state and image data through the usage of neural net architectures. E2E learning approaches enjoy the following properties.

- Generalize well to unknown environments.
- Generate policies for obstacle avoidance (specifically in the context of locomotion).

Concerns with this method include:

- Many fully end-to-end approaches ignore the dynamics of the system they are acting on, and as a result will generate non-smooth, jerky trajectories that can be non-ideal for a navigation task and potentially damage the robot.
- End-to-end approaches are known to have high sample complexity.

7.1.3 Learning-Based Waypoint Approach to Navigation (LB-WayPtNav)

In this paper [1], the authors aim to combine the "the best of optimal control and learning:"

- **Optimal control** allows for robust tracking of smooth, dynamically feasible trajectories.
- **Learning approaches** can generalize, leverage statistical inference to predict unknown environment from "partial views" (i.e. RGB images), and do not need to generate costly maps.

The hope is to combine the two approaches to leverage the robust performance of optimal control while navigating an unknown environment.

7.1.3.1 Problem Setup

- **Task**: The robot is given a goal position $p^* = (x^*, y^*)$ in reference to its own coordinate system, and the task is to navigate to the goal state while avoiding obstacles in an unknown map.
- **Assumptions**:
  - Perfect odometry information available
  - Static environment
  - The robot has a relatively simple known dynamics model (approximated with Dubin's car model)
- **Source of environmental perception**: A front-facing RGB camera.

7.1.4 Model-based Learning for Navigation

The LB-WayPtNav makes use of two submodules, perception and planning, to devise a closed-loop controller.

7.1.4.1 Perception Module

The perception model is implemented using a CNN with the following inputs and outputs:

- Inputs:
  - The 224×244 RGB image I_t from camera input
  - The goal position p_t^*
  - The current linear / angular velocities u_t
- Outputs:
  - The next waypoint $\hat{w}_t := (\hat{x}_t, \hat{y}_t, \hat{\theta}_t)$

The perception module is trained in simulation with maps taken from the Stanford Large-Scale 3D Indoor Dataset (S3DIS). During the training, knowledge of the map is assumed to generate an expert trajectory to a random goal configuration in the map, which is solved using MPC by optimizing for a trajectory that avoids obstacles and goes toward the goal. This supervised learning approach requires no human labeling and develops a perception module that can adequately generate waypoints for feasible trajectories to a target state.

7.1.4.2 Planning and Control Module

The planning and control module constitute the "optimal control" part of the system. After the perception module determines a waypoint, the control module performs the following to devise and track a trajectory that leads to the given waypoint:

- A third order spline is fitted to create a dynamically feasible trajectory from the robot current state to the waypoint, parameterized by a sequence of states and velocities.
- The nonlinear dynamics of the robot are linearized over the trajectory so that the LQR can determine the optimal feedback matrix and the optimal feedforward input.

- This procedure is done over a horizon of H timesteps and repeated after every horizon, until the robot reaches the goal position.

Together, these components form an explicit integration of perception and control, and an implicit integration of learning. See 7.1 for a diagram of the closed-loop system consists of the perception and planning modules.

7.1.5 Simulation Experiments

The LB-WayPtNav is tested in environments from scans of real-world buildings. The success metrics include:

- Success rate
- Average time to reach the goal
- Average acceleration
- Jerk along the robot trajectory

The results are compared to 2 different approaches: E2E Learning and Geometric Mapping and Planning. The main simulation results are summarized below.

- **Comparison with E2E learning**
  - LB-WayPtNav had the robot reach the goal with a 22% higher success rate, 40% faster, and with 50% less acceleration resulting in less power consumption while maintaining safety of the robot and successful navigation.
  - LB-WayPtNav only needs to learn "where to go" (i.e. the waypoints) while E2E needs to learn "how to go" there (i.e. the policy).
- **Comparison with Geometric Mapping and Planning**
  - LB-WayPtNav is a reactive planning framework so a memory-less version of Geometric Mapping and Planning was used for comparison. Memory-less mapping slightly outperformed LB-WayPtNav because of the perfect depth estimation in simulation.
  - Geometric Mapping and Planning with memory outperformed all other planners in a simulated environment, but as we see later, it quickly fails in realistic environments due to it not being as robust to different distributions of images caused by irregularities in lighting conditions.

Even though LB-WayPtNav is able to perform navigation tasks in novel environments, it can only do local reasoning and gets stuck in some situations: This is mainly due to the LB-WayPtNav not having a memory structure in the network. The most prominent failure modes are:

- When the robot is too close to an obstacle
- When the robot needs to "backtrack" from an earlier planned path

7.1.5.1 Hardware Experiments and Results

The LB-WayPtNav is also tested using Turtlebot 2 hardware testbeds and on-board odometry sensors. The main results indicate that LB-WayPtNav outperforms rest of the planners:

- LB-WayPtNav achieves 95% success rate and operates much better without need for extra information like depth sensors and without building explicit maps.
- Using LQR feedback control compensates for the crude approximations of the robot's physics from the simulation dynamics model and lets the robot closely track the desired trajectories.
- LB-WayPtNav can perform well under extreme lighting conditions due to aggressive data augmentation.
- LB-WayPtNav is agile and reactive, can adapt to changes in the environment.

7.1.6 Conclusion

- LB-WayPtNav combines learning and model-based control for goal-driven navigation in novel indoor environments:
  - Better at reaching unseen goals.
  - Can generalize from simulation to real-world robots.
- When objects are too different in simulation compared to real-world, it fails to predict good waypoints.
- It may not be optimal for long range tasks due to it being a reactive planner without memory structures.

7.1.7 Class Discussion Points

Dealing with Oscillatory Behavior of WayPtNav (by Prof. Tomlin)

- There is some oscillatory behavior of WayPtNav where it zigzags to the goal.
- Caused by the case when the robot has the goal outside of FOV, it turns and the goal is outside of FOV again, and keeps repeating.
- One possible solution is to have a persistent memory structure.
- But how can we get over this without adding too much memory architecture? Open architecture / training question.

Comparison with SLAM vs. WayPtNav

- Prof. Malik on failure modes of SLAM:
  - Failures of SLAM are due to having mismatched models (ex. Sunlight, irregularity, etc.).
  - Nature of noise is not "Gaussian", while SLAM assumes a rigid Gaussian noise structure.
- Prof. Sastry on reconstruction:
  - Reconstruction in SLAM is done in a very geometric way, a lot of constraints.

  - We are solving a problem we don't need to solve – For waypoint style navigation, we just need to know where to go next that is safe.

- Prof. Tomlin: WayPtNav does not focus so much on the environment (or reconstruction of it), so it can put more focus on the task at hand.
- Frank's question on quasi-static approaches: Where do quasi-static approaches fail? Where is more metric-type representation needed?
  - Prof. Malik: Two extremes exist:
    1. No maps, purely reactive systems – shouldn't work well. Any big environments, it wouldn't work either.
    2. Full metric maps? – topological maps are probably sufficient.
  - If robots are agile and are in tight corners, metric information from SLAM could be useful.
- Prof. Sastry on making SLAM competitive for different/adversarial lightings:
  - SLAM relies on point based features.
  - Extract higher level features and higher level landmarks.
- Prof. Ma: SLAM is good for mapping, not the navigation
  - Need higher level, robust features.

General Comment on Learning in the Context of Navigation (by Prof. Malik)

- Most navigation algorithms solve a "static" problem – stop-move-stop-move...
- Don't take dynamics of moving vehicle/robots into account.
- Need "quasi-static" approach which would make it more robust in real scenarios.

Certainty Equivalence Principle in Action

- Prof. Sastry: End-to-End (E2E) does both reconstruction and navigation at the same time:
  - Does it mix up too many things to do at once, or is it even more fundamental than that?
  - How does certainty equivalence outperform E2E?
- Prof. Tomlin: Separating out two tasks makes the planner perform better:
  - Don't have to learn dynamics, can optimize for planning.
  - When failures happen, a planner would still do what good control algorithms would do.
- Prof. Sastry: Certainty Equivalence is "comfortable." What about having some probabilistic modeling, and do stochastic control/probabilistic RL?
  - Prof. Malik: Sample complexity could be very high
    * Utilizing control part crushes the sample complexity.
    * Probabilistic viewpoint used to be popular before deep learning took over (ex. Using particle filters).
    * In deep learning, the modules are concatenated, and probabilistic parts are coming in purely through the sample distribution.

        - Prof. Sastry: Neural net somehow embodies all probabilistic aspects.
    - Prof. Tomlin: Does certainty equivalence hold in this case? It was trained using knowledge of the planning and control algorithms, which makes the method more supervised.

Bringing the Control Planner to Realistic Scenarios (Question raised by Prof. Ma)

- To bring this kind of systems to real world, imagine we need to deploy these kinds of robots to a shopping mall or a restaurant where there are moving people around or other independent/collaborative robots around. How will the perception, planning and control be?
- Valmik: Gather new data from the environment, which were not present.
- Prof. Ma: Adaptation requires three main tasks:
    - Improve through training
        - Amay: To use the same kind of supervision on the job as is done in this paper though, wouldn't you need to have a backend generating a map anyway as the robot acts?
        - Valmik: Yes potentially, but that map doesn't need to be all-encompassing, you can just use more localized 3D data.
    - Improve through adaptation
        - Kshitij: It also seems that in complicated environments like a shopping mall, etc., there is a need to model game-theoretic interactions because not only should the robot adaptively learn from mistakes, but other agents (perhaps humans) are modulating their actions with respect to each other and the robots.
    - Improve through mistakes
        - Michael: How do we know when mistakes, big or small, happen?
        - Prof. Ma: Most mistakes are conspicuous, but we still need to be careful.
- Prof. Tomlin: How do you set up databases of people moving?
    - Generating a database of simulating people can aid.
    - HumANav dataset.

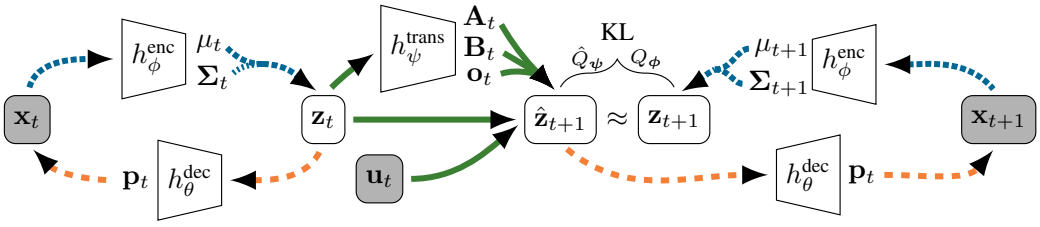

Figure 7.2: The information flow in the E2C model. From left to right, we encode and decode an image x_t with the networks h^{enc_ϕ} and h_θ^{dec}, where we use the latent code z_t for the transition step. The h_ψ^{trans} network computes the local matrices A_t, B_t, o_t with which we can predict $\hat{z}^{t+1}$ from z_t and u_t. Similarity of the predicted $\hat{z}^{t+1}$ to the encoding z_{t+1} is enforced by a KL divergence on their distributions and reconstruction is again performed by h_θ^{dec}.

7.2 Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images

This paper [3] studies how to model the learning and control of non-linear dynamical systems from *raw pixel images*. By applying the learned deep generative models, the proposed Embed to Control (E2C) framework is able to generate image trajectories from a latent space in which the dynamics is constrained to be *locally linear*.

7.2.1 Stochastic Optimal Control in the Latent Space

E2C considers the control of unknown dynamical systems of the form:

$$s_{t+1} = f(s_t, u_t) + \xi, \quad \xi \sim \mathcal{N}(0, \Sigma_\xi), \tag{7.1}$$

where t denotes the time step, $s_t \in \mathbb{R}^{n_s}$ the system state, $u_t \in \mathbb{R}^{n_u}$ the applied control and ξ the system noise. The goal of E2C is to infer a low-dimensional latent state space model in which optimal control can be performed, i.e.,

$$\begin{aligned} z_t &= m(x_t) + \omega, \quad \omega \sim \mathcal{N}(0, \Sigma_\omega), \\ z_{t+1} &= f^{\text{lat}}(z_t, u_t). \end{aligned} \tag{7.2}$$

Thus, E2C aims to learn a function m, a mapping from high-dimensional images x_t to low-dimensional vectors $z_t \in \mathbb{R}^{n_z}$ with $n_z \ll n_x$, such that the control problem can be solved using z_t instead of x_t.

7.2.1.1 Problem Assumptions

1. Assumes known dynamics model.
2. Low-dimensional state representation can be achieved – learn the dynamics in the *latent* space.
3. Consider quadratic cost and linearized dynamics (iLQG [2]).

7.2.1.2 Main Features of the Framework

Some highlights of the framework include:

1. Latent space representation of nonlinear dynamics (assume Gaussian noise).
2. Linearized/quadraticized control and cost.
3. Learn model f^{lat} or the linearized dynamics of it.

In order to learn the latent space, a good framework should satisfy the following properties:

1. Must sufficiently capture x_t for reconstruction (observability).
2. Must enable prediction of z_{t+1}.
3. This prediction must be locally linearizable for any magnitude of control.

This means that in general, our system should satisfy observability, predictability, and linearizability. Furthermore, it implies that we need some level of bijection between the latent state and the image state.

7.2.2 Architecture

The architecture of E2C consists of three modules, encoding, decoding, and transition. The overall architecture is shown in Figure 7.2:

1. Encoding module: learn latent embedding, from x_t to z_t
2. Decoding module: decompose the latent space representation and reproduce the raw image, from z_t to x_t
3. Transition module: predict the dynamics using learned linearized dynamics models, from z_t to z_{t+1}

Encoding Module Q_ϕ: To infer the latent state, we additionally assume the Variational Gaussian posterior distribution assumption, such that we can approximately sample from the latent space using Gaussian distributions. Thus, we assume the following approximate Gaussian form

$$Q_\phi(Z|X) = \mathcal{N}(\mu_t, \text{diag}(\sigma_t^2)), \tag{7.3}$$

where the encoding network parameters are given by

$$\mu_t = W_\mu h_\phi^{\text{enc}}(x_t) + b_\mu \tag{7.4}$$
$$\log \sigma_t = W_\sigma h_\phi^{\text{enc}}(x_t) + b_\sigma. \tag{7.5}$$

Decoding Module P_θ: The transition model also assumes Gaussian transition. As in Figure 7.2, A-matrix is assumed to be a perturbed matrix, an assumption that makes sense as it is discrete time transition, which should usually have an identity term. The transition model with the Gaussian transition assumption gives rise the following equation for the density model of the next state prediction $\hat{z}_{t+1}$ (for now, we assume that the A_t, B_t, o_t are known):

$$\hat{Q}_\psi(\hat{Z}|Z, u) = \mathcal{N}(A_t\mu_t + B_t u_t + o_t, C_t). \tag{7.6}$$

where the uncertainty is drawn from estimation error as well as measurement noise:

$$C_t = A_t \Sigma_t A_t^\top + H_t. \tag{7.7}$$

This update is inspired by the Kalman filter uncertainty update.

The generative model to reconstruct the image, from both z_t and $\hat{z}_{t+1}$ utilizes the Bernoulli distribution, which lets us generate black and white images,

$$P_\theta(X|Z) = \text{Bernoulli}(p_t), \tag{7.8}$$

where p_t is the output of the decoding neural net:

$$p_t = W_p h_\theta^{\text{dec}}(z_t) + b_p. \tag{7.9}$$

Note this seems to only work for Boolean states (black-and-white images). (*Remark:* what about greyscale? Looks pretty limiting.)

Transition Module $\hat{Q}_\psi$: Up until now, we assumed that we know the A_t, B_t, o_t parameters. In order to learn these parameters, we learn the dynamics models from the neural network h_ψ^{trans}:

$$\text{vec}[A_t] = W_A h_\psi^{\text{trans}}(z_t) + b_A, \quad A_t = (I + v_t r_t^\top), \tag{7.10}$$

$$\text{vec}[B_t] = W_B h_\psi^{\text{trans}}(z_t) + b_B \tag{7.11}$$

$$o_t = W_o, h_\psi^{\text{trans}}(z_t) + b_o. \tag{7.12}$$

7.2.3 About training

To train all the deep learning modules, E2C uses stochastic gradient descent (SGD) on a single loss function for all three neural nets at once (in Figure 7.2). This loss term includes:

1. KL on predicted model vs. actual model – (i.e., enforce that the transition in latent space matches up with the encoding)
2. Loss term from Variational Autoencoder for two time steps $t, t+1$ (evidence lowerbound, ELBO)

The loss function is given as the following:

$$\mathcal{L}(D) = \sum_{(x_t, u_t, x_{t+1}) \in D} \mathcal{L}^{\text{bound}}(x_t, u_t, x_{t+1}) + \lambda \text{KL}(\hat{Q}_\psi(\hat{Z}|\mu_t, u_t)||Q_\phi(Z|x_{t+1}), \tag{7.13}$$

$$\mathcal{L}^{\text{bound}}(x_t, u_t, x_{t+1}) = \mathbb{E}_{\substack{z_t \sim Q_\phi \\ \hat{z}_{t+1} \sim \hat{Q}_\psi}} \left[-\log P_\theta(x_t|z_t) - \log P_\theta(x_{t+1}|\hat{z}_{t+1}) + \text{KL}(Q_\phi||P(Z)) \right]. \tag{7.14}$$

7.2.4 Experimental setup

This paper studies 4 visual tasks, where in each of the tasks, we are given a third person view image of the robot performing these tasks:

1. An agent in plane with obstacles.
2. Inverted pendulum swing-up task.
3. Balancing a cart-pole system.
4. Control of a 3-link robotic arm.

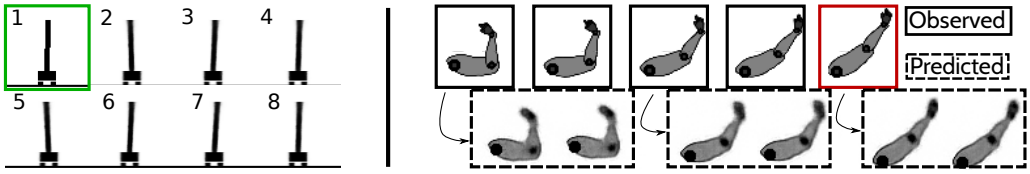

Figure 7.3: Left: Trajectory from the cart-pole domain. Only the first image (green) is "real", all other images are "dreamed up" by our model. Notice discretization artifacts present in the real image. Right: Exemplary observed (with history image omitted) and predicted images (including the history image) for a trajectory in the visual robot arm domain with the goal marked in red.

Two network types for the model are utilized to learn each of the modules:

1. Standard fully connected neural networks with up to 3 layers.
2. A deep convoluted network for the encoder in combination with an up-convolutional network as the decoder.

Two trajectory optimization algorithms are used to solve the control problem:

1. Iterative linear quadratic regulation (iLQR) for plane and inverted pendulum.
2. Approximate inference control (AICO) for cart-pole and arm.

Planning is performed in latent state without access to any observations except the current state. We present some visualization results of the proposed method in Figure 7.3.

7.2.5 Conclusion

1. E2C is a founding framework to tackle stochastic optimal control via latent space embedding when we are only given access to high-dimensional image streams.
2. E2C extracts a latent dynamics model which is constrained to be locally linear in its state transitions.
3. Based on experiments, E2C can find embeddings in which control can be performed with ease, reaching performance close to optimal control on the real system model.

7.2.6 Critiques and Suggestions

1. The experiments are differentially flat, so obtaining performance close to optimal control should not be too challenging using conventional algorithms.
2. Black-and-White images in static third person view are not the types of sensor data where this could be useful – what about navigation and manipulation in 3D?
3. Learning with sensor data might be more useful in soft robotics with noisy sensor and complicated / nonlinear dynamics.

7.2.7 Class Discussion Points

Some Noticeable Problems of This Approach (by Prof. Ma)

- Fundamental problem: model (complexity) selection
  - Using KL divergence, distributions blindly can lead to comparing degenerate distribution.
  - For low dimensional systems, it's problematic – no one talks about this, people rather use this as a heuristic.
  - How do we choose the latent dimension? No explanation given in this paper.
- Practical problem: is predicting image necessary?
  - We only need to predict some of the hidden states/object level, not exactly the image.
  - Valmik: The fact that they used the image is not quite as necessary?
    - Basically, it is only doing state estimator and dynamics learning in one.

Embedding of High Dimensional Features to Latent Space (by Prof. Tomlin)

- Any notable embedding work in context of navigation and manipulation in 3D? (i.e. outward looking images)
  - Valmik: The E2C paper is cited a lot in E2E navigation papers, though not many explicit connections/extensions.
    - E2E training of deep visuomotor policies by Prof. Pieter Abbeel: use some concepts, but for simple dynamics systems.
  - Prof. Ma: Not much guarantee in this process, what kind of problems can this solve?
    - Even for linear subspace selection like PCA, choosing dimensionality is already complex and unclear.
    - For driving, people have very much abandoned this approach.

Connections to Non-Linear System Identification (by Prof. Ma)

- Can this problem be thought of as non-linear SysId with images as observables?
- Prof. Tomlin: Different way of solving system identification problem - http://fenn.freeshell.org/Science.pdf.
- Local linear controls are rather promising in navigation.
- Where does it not work well?
  - E.g. Games, walking.
  - Valmik: Systems with cusps/bifurcations/sensitive to approximations.

References

[1] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining optimal control and learning for visual navigation in novel environments. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 420–429. PMLR, 30 Oct–01 Nov 2020.

[2] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 300–306. IEEE, 2005.

[3] Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv preprint arXiv:1506.07365*, 2015.