A Robot Learning to Play a Mobile Game Under Unknown Dynamics

Wonjun Yoon*  Sol-A Kim*  Jaesik Choi

Abstract—With the advance in robotic hardware and intelligent software, humanoid robot could play an important role in various fields including service for human assistance and heavy job for hazardous industry. Under unknown dynamics operating smart devices with a humanoid robot is a even more challenging task because a robot needs to learn both swipe actions and complex state transitions inside the smart devices in a long time horizon. Recent advances in task learning enable humanoid robots to conduct dexterous manipulation tasks such as grasping objects and assembling parts of furniture. In this paper, we explore another step further toward building a human-like robot by introducing an architecture which enables humanoid robots to learn operating smart devices requiring complex tasks. We implement our learning architecture in the Baxter Research Robot and experimentally demonstrate that the robot with our architecture could play a challenging mobile game, the 2048 game, as accurate as in a simulated environment.

I. INTRODUCTION

Manipulating objects with a humanoid robot is an important task since such human-like robots could play an important role in service for human assistance and heavy job for hazardous industry in the near future. However, designing and building a proper robotic manipulation task is not trivial since the dynamics of the robot and constraints of objects should be carefully considered. Thus, a successful robotics manipulation task is not easily expanded to general human-like tasks such as part assembly and smart device manipulation. Recent advances in task learning under unknown dynamics enable humanoid robots to conduct various dexterous manipulation tasks such as assembling parts of furniture [1] and toys [2].

Touch-enabled smart devices including smart phones, tablet computers, and Internet of Things (IoT) devices have widespread applications in everyday life. Compared to mechanical manipulations (e.g., device switches and buttons), touch-enabled smart devices are easier for developers to include rich functionality and for users to manipulate without much physical forces applied. Thus, a general-purpose humanoid robot may require to be able to learn (1) how to smoothly handle such smart devices and (2) how to execute a long sequence of actions to fulfill a non-physical task goal.

Manipulating touch-enabled smart devices has unique challenges in that such devices are usually fragile and require a long sequence of manipulation actions, i.e., a complex task planning. The manipulation becomes even harder especially under unknown dynamics of the robotics manipulator and unknown task constraints.

We solve this issue by introducing a new architecture which seamlessly satisfies the requirement of the local manipulations (swipes) and the long-term tasks. We present a general framework to learn to fulfill complex tasks on smart devices. As shown in Fig. 2, our architecture includes general tools for recognition, planning, and execution. One important aspect of our architecture is that we could learn (or train) individual components to fulfill the goal of complex tasks. In detail, our architecture includes (1) learning task actions with long time horizons by deep reinforcement learning [3] and (2) learning manipulation actions from Linear Quadratic Regulator (LQR) [4].

we experimentally demonstrate that the combined architecture can be used for learning non-trivial manipulation tasks for smart devices. Specifically, we demonstrate that the Baxter Research robot [5] equipped with our architecture solves a non-trivial mobile game application, the 2048 game [6], as shown in Fig. 1. In experiments, we show that the winning rate of our Baxter robot for the 2048 game is as high as the Deep Q Learning trained solely in a simulated environment.

In the following, Section II explains related work for learning manipulation tasks for humanoid robots. Section III explains existing models used in our work; Linear Quadratic Regulator (LQR) and Deep Reinforcement Learning (DRL). Section IV presents our architecture for the task learning. Section VI reports the experiments results followed by conclusion in Section VII.

*These authors contributed equally to this work.
School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology, Ulsan 44919, Republic of Korea {wjyoon26,johnksy,jaesik}@unist.ac.kr
II. RELATED WORK

Industrial robotic manipulators are used to verify the functionality of touch-enabled smart devices [9], [10], [11], [12]. Commercial robotic manipulators are successfully used to test and verify the functionality of smart devices [11], [12]. Most of commercial tests are designed to verify the functional characteristics of touches such as latency, sensed forces and durability with a specialized mobile application. Our work further extends this efforts to let a general-purpose humanoid robot manipulate smart devices for complex tasks in long time horizons.

The linear quadratic regulator (LQR) provides an optimal control solution where the system is a linear differential dynamic system and the cost function can be represented as a quadratic function. Under unknown dynamics, LQR is especially useful because LQR learns locally linear action model which is less dependent on dynamics. Also, the linear Gaussian transition model is simple and easy to control. Thus, LQR has widely used to learn manipulation actions of robots [13], [14], [15]. LQR does not require one to fully specify the dynamical constraints of the task and the robot manipulation. LQR learns a set of proper actions with minimal human (engineer) specifications. In experiments, we find a close-to-optimal actions by the iterative LQR [16], [17], [4] which has been successful to learn various non-linear robotic manipulations.

Reinforcement learning have been extensively used in high-level tasks such as playing games [3], planning to manage supply chains and controlling multi-agents in military systems. Reinforcement learning also holds the promise of enabling robots to learn motion skills for manipulating external objects. Recent advances have shown that it could learn various high-level tasks including playing tennis [18], stacking blocks [19], assembling parts using tools [4].

There has been long efforts to learn low-level (motion planning) actions to conduct high-level (complex) tasks which require to reason not only about the kinematic and geometric constraints but also intermingled constraints on state spaces [20], [21], [22], [23]. Our ultimate goal is to automate the procedure of learning low-level actions and high-level tasks simultaneously. In this paper, we present an integrate architecture to demonstrate the feasibility of our goal to learn motion planning and task planning, and then execute them together in an integrated humanoid platform.

III. BACKGROUND

A. Linear Quadratic Regulator Models

Linear quadratic regulator (LQR) solves feedback control problems where the system dynamics is composed of linear differential equations and the cost is represented as a
quadratic function, called the LQR problem. The following equations are state and cost functions with a finite-horizon, discrete-time LQR,

\[
x_{t+1} = A_t x_t + B u_t,
\]

\[
l(x, y) = \sum_{i=0}^{N} (x_t^T Q x_t + u_t^T R u_t),
\]

where \(x_t\) and \(u_t\) are states and user input (control) respectively, and \(Q, R\) and \(N\) are predefined model parameters (matrices) for the cost (or loss) function. Note that, we wish to find the optimal trajectory, \(\tau = (x_0, u_0, \cdots, x_N, u_N)\), which minimizes the loss function while satisfying the transition model as in Equation 1.

The optimal trajectory of the LQR problem is derived as,

\[
u_t = -F_t x_t
\]

\[
F_t = (R_t^T P_t B)^{-1} (B_t^T P_t A_t)
\]

where \(P_t\) is found iteratively backwards in time,

\[
P_{t-1} = A_t^T P_t A_t - (A_t^T P_t B_t (R_t + B_t^T P_t B_t)^{-1} (B_t P_t A_t) + Q,
\]

where \(P_t = Q\).

1) Iterative Linear Quadratic Regulator: Iterative Linear quadratic regulator (iLQR) is an iterative LQR method which finds the optimal trajectory by applying LQR repeatedly on the trajectory solved. In each iteration, iLQR adjusts the following cost function,

\[
l(x_{t+1}, u_{t+1}, \alpha_{t+1}) = (1 - \alpha) l(x_t, u_t) + \alpha (\|x_t - x_t^0\|^2_2 + \|u_t - u_t^0\|^2_2)
\]

where \(l\) is a quadratic cost function. If \(\alpha\) is close to one, the squared error term would converge to zero. That is, iLQR finds the optimal trajectory [17].

By the trajectory optimization, it finds a trajectory \(U^*\) which minimizes the sum of the cost function,

\[
l_0(x, U) = \sum_{i=0}^{N-1} l(x_t, u_t) + l_f(x_N)
\]

\[
l_i(x, U) = \sum_{i=1}^{N-1} l(x_t, u_t) + l_f(x_N).
\]

B. Deep Q-Learning

Deep Q-Learning incorporates deep neural networks in solving reinforcement learning (Q-learning) problem. A Deep Q-Learning method already has achieved the best performances in various tasks including game playing, called ‘Atari’ [3] by learning complex control policies.

1) Neural Networks for Reinforcement Learning: Typically the policy \(\pi\) and the value function \(Q^\pi(s, a)\) in the reinforcement learning is defined as follows.

\[
a = \pi(s)
\]

\[
Q^\pi(s, a) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s, a],
\]

where \(s\) and \(a\) are respectively state and action, and \(r_t\) is the reward at \(t\) and \(\gamma\) is the discount. Then, using the Bellman Equation and value iteration algorithms can solve above as,

\[
Q_{t+1}(s, a) = \mathbb{E}_r [r + \gamma \max_{a'} Q_t(s', a') | s, a]
\]

Here, the value function \(Q\) can be represented by deep Q network.

IV. LEARNING MANIPULATIONS FOR SMART DEVICES

Manipulation of smart devices is an elaborate and complex task which typically needs adequate pressure, swipe velocity, and direction. Learning the controller must be harder than any other nonlinear systems. In our work, typical manipulation tasks for smart device swipe action learning is handled.

2) Our architecture of the manipulation: Finding the optimal trajectory, from nonlinear dynamics, is not an easy task. Thus, we change the problem into a linear Gaussian model [16]. The controller for swipe action is locally trained with the linear dynamics by using iLQR.

First, we initialize the five trajectory points for each direction; left, right, backward, and forward as plotted in Fig. 3.

3) Action and State Space for Baxter Research Robot: Our Baxter research robot has seven joint angles in each arm. However, for convenience and intuition, we utilize the inverse kinematics to construct the dynamic manipulation with a set of states and actions in 3-dimensional workspace.

- State \(X_t = [x_t, y_t, z_t]\)
- Action \(U_t = [\Delta x_t, \Delta y_t, \Delta z_t]\)

4) Learning LQR for actual trajectories: Considering the executable velocities of our Baxter robot, we discretize the time step into 250 in 3 seconds. As the iteration number becomes larger, the optimized trajectory by LQR fits to the intended trajectory.

\[
l = Q(X - X_N) + \sum_{i=1}^{N-1} Q(X - X_i),
\]

\[
l = RU.
\]

V. LEARNING TO PLAY A MOBILE GAME FOR A HUMANOID ROBOT

A. Executing a mobile game by the humanoid robot

Learning a smart phone game with the humanoid robot does not mean that it is necessary to learn the game with its arm. Since executing the game by the robot is relatively slower than executing it automatically, we have tried to find the optimal policy in simulation then, transfer it to the robot.

We chose the 2048 game [6] because of its simplicity of manipulating the smart phone and its complexity of winning the game. The input state is a bit string converted from a set of the positions of tiles and each value. That is the state is now with the 12 x 16 size since the number of possible values from 0 to 2048 (from \(2^{10}\) to \(2^{11}\)) in 16 possible positions [Fig. 2.b.1]. The output includes four swiping actions: up, down,
Fig. 3: Expected trajectory (red line) is just linked line of input trajectory points (red triangle). After learning by iLQR, the optimized trajectory(green line) is almost follow the expected trajectory (blue line). White the line is a set of points touching the surface (blue surface) which refers to the surface of the smart device with optimized trajectory.

left, and right. The rewards are given when a tile marked with a goal number comes out.

We have provide fours heuristic algorithms for solving the 2048 game as explained in Section VI-3. However, since the rewards from the heuristic algorithm are much lower than the ultimate reward for winning the game, the agent will not exactly follow the heuristics. Instead, such heuristics help the learning algorithm converge faster to get to the goal. The learning procedure consists of two steps: recognizing digits by neural networks and executing the actions from the input by the recognition.

1) Recognizing Digits by Neural Networks: We design a simple neural network to recognize digits in the smartphone since the digits recognition task in the smartphone resemble the digit recognition in the MNIST database [24] where the deep learning methods achieve the state of the art performance.

After the image segmentation in [Fig. 2-(a.3)], extracted tiles with the size of 32 by 32 pixels each enter into the 1-layered DBN. To fit the model, we gather the 14,034 tiles from the camera attached to the arm of the Baxter robot by randomly moving the robot for data augmentation. This configuration makes a similar situation where the robot move as [Fig. 2-(c) to (a)]. We also normalize data, and generated augmentation data by rotating and moving slightly [25]. Now, the augmented data is with the 84,204 tiles. Such augmented data let our recognition accuracy as high as 98.9%.

2) Executing the Actions for Robot: Learning the actions for the robot starts from defining its input state. Since a set of input digits recognized by neural network is discrete number from 0 to 11, we change the set into a bit string which indicates a specific value in a specific position as 1 or 0. The bitmap is size of 16 x 12, in ascending order from 0 to 2048 for 16 positions, and the same values are adjacently mapped as shown in [Fig. 2-(b.1)]. We use three layers of fully connected neural network, each of them consists of 500 hidden units [Fig. 4]. The learning step is composed of three steps: Forward, Action, and Backward.

Forward : Every previous two pairs (state and action) is saved in our memory to fill up the input for our network. Then, using the previous two state and action pairs and current state, the policy is calculated for all action values. An action with the maximum reward is selected as the best action. We use $\varepsilon$-greedy algorithm where the probability of choosing an random action decreases from 1.0 to 0.05 as age increases in each 2,000 learning iteration.

Action : Run the best action from our policy and get the output as the next state. Especially, in our 2048 game, the computer inserts randomly a 2 or 4 tile in an empty space (adversarial moves).

Backward : Give the reward computed from the last action and a set of states $(s_t, a_t, s_{t+1})$. The network will be trained by random sampling method which selects samples from our entire experience history. The batch size is 16 and the total experience size is 30,000. Each backward step, we append the current state, action, reward and the next state, $(s_t, a_t, r_t, s_{t+1})$. When the experience is full, one of the existing entry is replaced by the new entry.

VI. Experimental Results

3) Details of the 2048 game: The 2048 game [6] is an variant of a preceding game the Threes [26]. Given a 4 by 4 grid, the player can move the grid in 4 directions (up, down, right and left). Each action makes the adjacent tiles merged when they have the same value as shown [Fig. 5].
Fig. 5: The 2048 game board is with a 4 by 4 grid. The right grid is generated by taking a right action from the left grid. Two values of 4 tiles in the last row are merged into 8. A tile with a value 2 is generated at the left-bottom corner.

| Target Value | Random Moves | Deep Q (Simulator) | Deep Q (Baxter) | difference(%) |
|--------------|--------------|--------------------|----------------|---------------|
| 128          | 53.98%       | 92.81%             | 90.74%         | -2.07%        |
| 256          | 7.09%        | 54.35%             | 54.00%         | -0.35%        |

The merged tile now has the sum of values. After every action, computer places 2 or 4 in each empty tile with the probability of 0.9 and 0.1 respectively.

4) Heuristics of the 2048 game: The four typical heuristics in the 2048 game are presented as follows.

- **Monotonicity**: This heuristic measures whether the values of the tiles are strictly increasing or decreasing along both the left/right and up/down directions. It tries to make the board well organized, making possible to merge smaller tiles gradually, then larger tiles.

- **Smoothness**: It is from an idea that the adjacent tiles need to have the same values to be merged. It calculates the value difference between its neighbors in four directions and tries to minimize the difference.

- **Free tiles**: Giving penalty when there are too few free tiles left.

- **Maximum value**: Calculating the maximum value among the whole tiles.

5) Results and Discussions: As shown in TABLE ref tbl: result, our Baxter robot reaches 128 with winning rate more than 90 percent which is comparable to Deep Q learning model in a simulated environment. The result is significantly better than the performance of random move which is just 53.98%. The winning rate of the game for reaching 256 is now 54.00% which is comparable to the Deep Q learning model in a simulated environment. Note that the performance of random moves is just 7.09%.

In our work, there are 2 factors which may incur errors beyond learning game policy model. We have a small error for DBN digit recognition model which is 1.1% and unknown error from the iLQR controller. In TABLE [1] it reflects the errors (difference) rate incurred from two potential erroneous factors. The difference of simulation and Baxter executions with respect to the number of iterations per game is about 7%. TABLE [1] reflect the accumulated erroneous moves caused by the recognition errors and control errors.

Table II: Comparisons of the number of iterations per game

| Target Value | Deep Q (Simulator) | Deep Q (Baxter) | difference(%) |
|--------------|--------------------|----------------|---------------|
| 128          | 86                 | 90             | 4.65%         |
| 256          | 146                | 157            | 7.53%         |

VII. CONCLUSION AND FUTURE WORK

Deep learning may boost the performance of robots in various aspects in learn to recognize and make plans. In this paper, we present an architecture which seamlessly combines the recognition, planning and execution. We demonstrate how deep learning can improve the performance of recognition and planning in a humanoid robot. In the experiments with the Baxter Research robot, we show that the Baxter research robot equipped with our architecture could achieve the high winning rate of a complex mobile game, the 2048 game, as the Deep Q learning algorithm in simulation.

ACKNOWLEDGMENT

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, ICT & Future Planning (MSIP) (NRF-2014R1A1A1002662), the NRF grant funded by the MSIP (NRF-2014M2A8A2074096). Authors thank Phuong Hoang and Janghoon Ju for their constructive comments.

REFERENCES

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” CoRR, vol. abs/1504.00702, 2015.
[2] S. Levine, N. Wagener, and P. Abbeel, “Learning contact-rich manipulation skills with guided policy search,” in IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015, pp. 156–163.
[3] Y. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, “Playing atari with deep reinforcement learning,” CoRR, vol. abs/1312.5602, 2013.
[4] W. Han, S. Levine, and P. Abbeel, “Learning compound multi-step controllers under unknown dynamics,” in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 6435–6442.
[5] E. Guizzo and E. Ackerman, “The rise of the robot worker,” IEEE Spectrum, vol. 49, no. 10, pp. 34–41, 2012.
[6] S. Rodriguez, “Maker of hit puzzle game ’2048’ says he created it over a weekend,” The Los Angeles Times, March 11 2014. [Online]. Available: http://www.latimes.com/.
[7] M. Teutsch, P. Trantelle, and J. Beyrer, “Adaptive real-time image smoothing using local binary patterns and gaussian filters,” in IEEE International Conference on Image Processing, ICIP 2013, Melbourne, Australia, September 15-18, 2013, pp. 1120–1124.
[8] J. Flusser, T. Suk, S. Farokhi, and C. H. IV, “Recognition of images degraded by gaussian blur,” in Computer Analysis of Images and Patterns - 16th International Conference, CAIP 2015, Valletta, Malta, September 2-4, 2015 Proceedings, Part I, 2015, pp. 88–99.
[9] T. Simonite, “Intel robot puts touch screens through their paces,” MIT Technology Review.
[10] K. Finley, “Robot with long finger wants to touch your iphone apps,” August 1 2013. [Online]. Available: http://www.wired.com/2013/08/tapster/.
[11] R. Electronics, (2015) Agile testing for electronic devices. [Online]. Available: http://www.pkcelectronics.com/.
[12] OptoFidelity, (2015) Phone validation by optofidelity human simulator robot. [Online]. Available: http://www.optofidelity.com/
[13] D. H. Nguyen, “A sub-optimal consensus design for multi-agent systems based on hierarchical LQR,” *Automatica*, vol. 55, pp. 88–94, 2015.

[14] A. Mosebach and J. Lunze, “LQR design of synchronizing controllers for multi-agent systems,” *Automatisierungstechnik*, vol. 63, no. 6, pp. 403–412, 2015.

[15] A. M. Benomair, F. A. Bashir, and M. O. Tokhi, “Optimal control based lqr-feedback linearisation for magnetic levitation using improved spiral dynamic algorithm,” 2015, pp. 558–562.

[16] H. Zhang, J. Gong, Y. Jiang, G. Xiong, and H. Chen, “An iterative linear quadratic regulator based trajectory tracking controller for wheeled mobile robot,” *Journal of Zhejiang University - Science C*, vol. 13, no. 8, pp. 593–600, 2012.

[17] Y. Tassa, T. Erez, and E. Todorov, “Synthesis and stabilization of complex behaviors through online trajectory optimization,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4906–4913.

[18] J. Kober, E. Oztop, and J. Peters, “Reinforcement learning to adjust robot movements to new situations,” in *Robotics: Science and Systems VI (RSS)*, 2010.

[19] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, “Learning to control a low-cost manipulator using data-efficient reinforcement learning,” in *Robotics: Science and Systems VII (RSS)*, 2011.

[20] L. P. Kaelbling and T. Lozano-Pérez, “Integrated task and motion planning in belief space,” *I. J. Robotic Res.*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[21] G. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez, “Constructing symbolic representations for high-level planning,” in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, 2014, pp. 1932–1938.

[22] L. P. Kaelbling and T. Lozano-Pérez, “Hierarchical task and motion planning in the now,” in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1470–1477.

[23] J. Choi and E. Amir, “Combining planning and motion planning,” in *2009 IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 238–244.

[24] L. Deng, “The MNIST database of handwritten digit images for machine learning research [best of the web],” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012.

[25] Z. Gan, R. Henao, D. E. Carlson, and L. Carin, “Learning deep sigmoid belief networks with data augmentation,” in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, 2015.

[26] B. Kuchera. (2015) Why it took a year to make, and then break down, an amazing puzzle game. [Online; posted 06-February-2013]. [Online]. Available: http://www.polygon.com/2014/2/6/5386200/why-it-took-a-year-to-make-and-then-break-down-an-amazing-puzzle-game