Playing Piano with a Robotic Hand

Yeon, Seong Ho
Massachusetts Institute of Technology
Cambridge, MA
syeon@mit.edu

***Abstract*—In this paper, I present a robotic framework to play a mechanical piano with an anthropomorphic robotic arm-hand based on an music sheet input. Detailed methods of implementing a full-stack pipeline including offline trajectory planner, dynamic controller and simulator, and sound generator are presented. In result, the presented framework manipulated the 23-degree-of-freedom robotic arm-hand model to played a classic song, Salut D'Amour, on drake environment. The limitations and potential future works are also discussed at last.**

I. Introduction

Playing a musical instrument is an extreme form of delicate manipulation maneuver of human. Specifically playing piano, manipulating ten end-effectors together, can be considered as a representative form of these manipulation activities. In order to demonstrate the advanced manipulation ability of mechatronic systems, varied research works have been presented [1]–[4]. However, strictly speaking, none of these works presented a comparable piano playing maneuver to human. These works, instead of using anthropomorphic arm-hand kinematic architecture, rather they utilized a mechatronics system specialized in pressing piano keys. While these approaches have demonstrated some interesting capabilities of mechatronics systems, thus, these works show limited proof in showing their capability to manipulate robot in an anthropomorphic manner.

Therefore, the goal of this project is to demonstrate piano playing maneuver with an anthropomorphic robotic arm-hand model with generic building blocks of robotic manipulations.

II. Method

A. System Overview

The Fig. 1 summarizes the overall system pipeline. With a music sheet input, offline trajectory planner analyzes the input music sheet and creates desired time-domain trajectories of all joints of the robotic arm. Based on this generated trajectory, dynamic simulation was performed tracking the given trajectory. Due to the high-degree-of-freedom(DoF) multibody plant (including the piano and the robot), dynamic simulation cannot be performed in real-time rate. Therefore, in order to simulate the resultant sound from the piano model, the resultant kinematic log of dynamic simulation was replayed in real-time rate while piano sound was generated based on the piano's mechcanical state. Details of each block are explained on the following subsections.

This is a final project report of the MIT 6.834 Robotic Manipulation course instructed by professor Russ Tedrake. The code for this project is in this repo https://github.com/seongho-yeon/playing-piano-with-a-robotic-hand

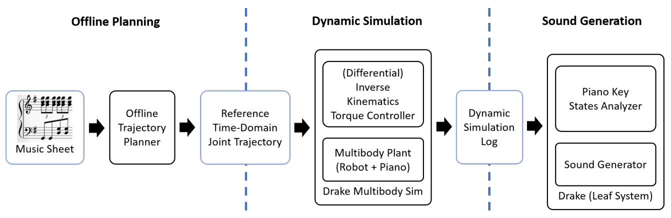

Offline Planning
Dynamic Simulation
Sound Generation
Music Sheet
Offline Trajectory Planner
Reference Time-Domain Joint Trajectory
(Differential) Inverse Kinematics Torque Controller
Multibody Plant (Robot + Piano)
Drake Multibody Sim
Dynamic Simulation Log
Piano Key States Analyzer
Sound Generator
Drake (Leaf System)

Fig. 1: System overview

B. Development of Multibody Plant

In this subsection, explanations of the constructed multibody plant are presented. The plant is consists of the 23-DoF robotic arm and hand and a 5-octave piano model. The 23-DoF robotic arm was constructed by welding the Allegro hand robot on an end-effector frame of the KUKA LBR-iiwa 7 manipulator arm [5], [6]. Fig. 2 shows the hand, arm, and constructed robot.

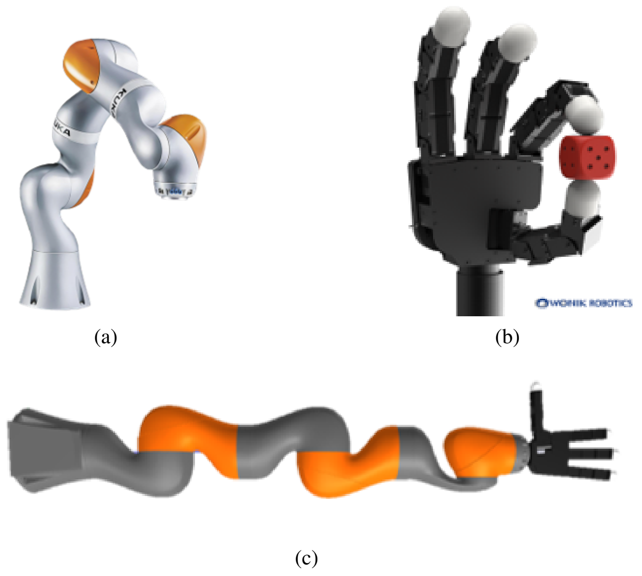


Fig. 2: Robot arm and hand models. (a) KUKA LBR-iiwa arm [6]. (b) Allegro hand robot [5]. (c) Constructed robot model on drake environment.

Each piano key was designed with a cube element and a virtual DoorHinge joint on *Drake* environment. The size of key was chosen with consideration of the size of robotic finger. This key was repeatedly placed for the 5-octave piano construction as shown in fig. 3.

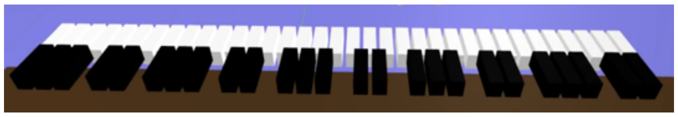


Fig. 3: Constructed piano model on *drake* environment

C. *Offline Trajectory Planner*

In this subsection, the detailed processing pipeline of the offline trajectory planner is explained. Fig. 4 shows an overview of the offline trajectory planner. With a music sheet input, the planner generates the desired joint trajectory of the robot to play the music on the piano.

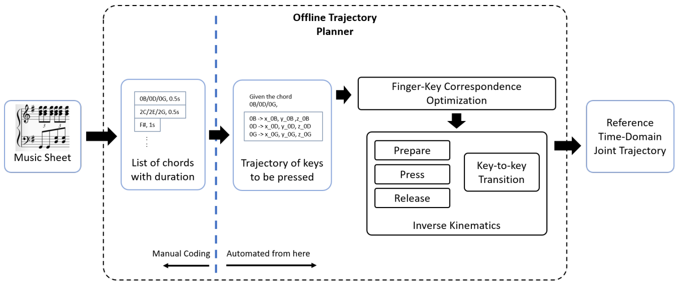


Fig. 4: Structure of the offline trajectory planner.

1) Interpreting Music Sheet: Proper coding of a given music sheet input is essential for a successful trajectory planning. A music sheet is consists of a "set" or a "trajectory" of discrete notes. Each note encodes its scale and syllable, or the resultant pitch of sound, and the duration of sound. In order to interpret these implications from the perspective of robotic trajectories, I transformed each note in to the class of a data structure. For each sound note data structure, four maneuver sub-sequences were defined: ready(prepare), press, release, and transition as shown in fig 5. This data structure contains four timing variables for these sub-sequences to encode the duration for each maneuver. Of course, this data structure also contains the pitch of sound as well as the physical positions of the corresponding key.

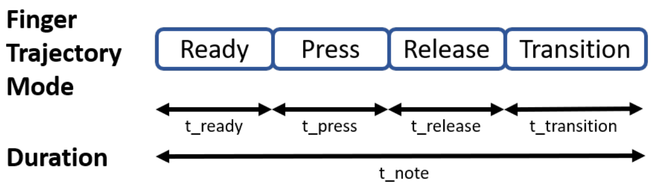


Fig. 5: Demonstration of four sub-sequence maneuvers within a single musical note.

Often a music sheet encodes a simultaneous notes being played together known as a chord. Piano is actually one of the best instrument to play these chords with multiple fingers. For coding a chord, a list of chord libraries were constructed containing the name of chord and its member notes.

To summarize, a given music sheet input generates a time-domain lists of either chords or notes encoding its abstracted duration of the trajectory modes, position of physical keys, and the pitch of sound.

2) Finger-Key Correspondence Planner: After generating an abstract time-domain trajectories of piano keys, finger-key correspondence planner processes this information and generates correspondence map of the robotic fingers and piano keys to be pressed. Fig. 6 visualizes a scenario to describe why finger-key correspondence planning is important such that there can be many combination to press a given chord with a robotic hand.

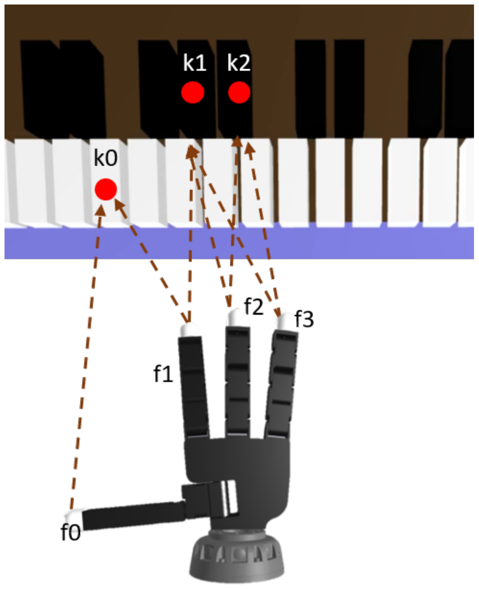


Fig. 6: Finger-key correspondence scenario demonstration. There exists multiple scenarios in pressing the given keys k0, k1, and k2.

The detailed algorithm of the planner is shown in the following.

$$
\begin{aligned}
C^* = \quad & \underset{C}{\arg\min} \quad \sum_i \sum_j |c_{ij}(p^{k_i} - p^{f_j})| \\
\text{s.t.} \quad & c_{ij} \in C, \\
& c_{ij} \in \mathbb{N}, \\
& \forall i, \sum_j c_{ij} = 1, \\
& \forall i < (N_{key} - 1), \sum_j c_{ij} < \sum_j c_{(i+1)j}
\end{aligned}
$$

To elaborate, the given algorithm is a simple heuristic optimization where the correspondence is chosen by minimizing the sum of corresponding finger-key distance. This approach is largely inspired from the point-cloud key point registration approach in iterative closest point (ICP) algorithm [7].

3) Key Poses Generation using Inverse Kinematics: With proper abstraction of the input music sheet and the finger-key correspondence planning, abstracted and discrete trajectories of end-effectors' positions of the individual robotic fingers are decided. Based on this information, inverse kinematics problems for each mode of the finger-ready(prepare), press, release, and transition-shown in fig. 5 were formulated. To describe the obvious purpose of the inverse kinematics problems, for press-key mode, the inverse kinematics problem should generate a state of the robot such that the finger only presses the desired piano keys. For press and release modes, the inverse kinematics would generate "reasonable position" to be prepared or released for key-pressing while not pressing any keys. For key-to-key transition mode, inverse kinematics problem interpolate between previous release-mode and next-prepare mode without colliding any keys of the piano. These inverse kinematics problems were solved based on the *InverseKinematics* module of *drake* environment. For all of the problems, the cost was set as a quadratic cost between the state of robot in the previous maneuver and the current state of the robot which is the solution of the problem. Fig. 7 shows detailed constraints of the inverse kinematics problems.

Finger Maneuver	**Constraints**
Press Keys	- Finger-Key position - Finger-Key orientation - Palm-Key clearance
Prepare & Release	- Z-clearance of finger positions (from target keys) - Horizontal slacks of finger positions - Palm-Key clearance
Key-to-key Transition	- Interpolation of palm pose - Palm-Key clearance - Limiting maximal change of palm orientation

Fig. 7: Constraints of the inverse kinematics problems.

These formulations were implemented with the member functions of the InverseKinematics module in *drake* environment: *AddPositionConstraint*, *AddOrientationConstraint*, and *AddAngleBetweenVectorsConstraint*.

These set of inverse kinematics problems generate key points of each maneuver mode. These key points were then interpolated linearly on joint-space to generate continuous time-domain trajectory of the whole joint states of the robot.

D. Multibody Contact Dynamics Simulation

In this subsection, an implemented structure of multibody contact dynamics simulation is briefly explained. In this work, the dynamic simulation phase is simply a data-processing pipeline to extract resultant keyboard states from the desired trajectory of the robot. Thus, this phase was designed in a conservative manner with standardized control block.

The dynamic simulation was performed with a time-step of 10 μs. In order to make the simulation simpler and run faster, instead of compensating gravity on the controller of the robot, the gravity in the simulation plant was nullified. For simulating 45 seconds of music sequence, it took 84 minutes to complete.

1) Joint Impedance Controller: A simple joint impedance controller is implemented to follow desired joint trajectory input as shown in equation 1.

$$\tau = K(q_{des} - q) + D(\dot{q_{des}} - \dot{q}) \tag{1}$$

Differential inverse kinematics controller utilizing the jacobians of each link were also implemented with intention to use them for task-space spatial impedance control. However, because playing piano incorporates multiple contacts on fingers, tuning of the differential inverse kinematics controller, specifically applying task-space force on fingertip, was not straight-forward. As a result, for stability of the controller, only the simple joint impedance controller was utilized in the final result.

2) Contact Mode Switching: In general, having tactile information for regulating contact behavior is always desirable when the sensors are available. This is because contact events imply a kinematic mode changes resulting the system becoming hybrid dynamical system [8]. For instance, in case of bipedal walker robots or quadrupeds robot, the roles of sensing contact and feedback was critical for their successes [8], [9]. The information from contact sensors often used to 1) regulate force/torque gains of actuators, and 2) adjust desired trajectories of joints to mitigate the impact.

However, for playing piano application, I evaluated that importance of regulating accurate sensing of contact force is less critical compared to other bipedal walker robots or quadruped robots. Also with realistic perspective that there exists issues in technical stability and difficulty in constructing tactile sensor on the robotic hand and incorporating them in controller on *drake* environment, I decided to not utilize tactile sensor in this implementation. Instead, I generated an abstracted contact state trajectory for each finger so that the controller can utilized this predicted contact state in regulating the control gain of them as shown in fig. 8.

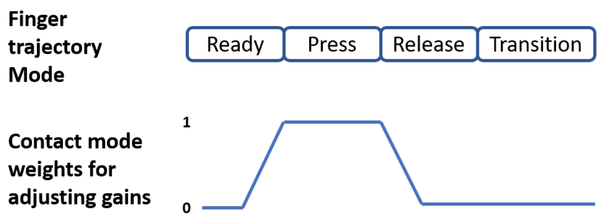


Fig. 8: Contact mode switching scheme of the robotic finger

Based on this contact mode weights of each finger based on its trajectory mode, torque gains on the joints of each finger get adjusted for piano key pressing maneuver.

E. Sound Generator

In this subsection, a structure of the sound generator is explained. Sound generator utilizes the simulation log from dynamic simulation to replay the music based on the piano's mechanical state.

1) System Diagram: Simplified system diagram of the sound generator block is shown in fig. 9. Dynamic simulation log encoding kinematic states of the robot and the piano was replayed in real-time rate for visual reconstruction of the log through *MultibodyPositionToGeometryPose* and *SceneGraph* module as well as piano sound generation on *drake* environment. Piano sound was generated with *PyGame* environment [10].

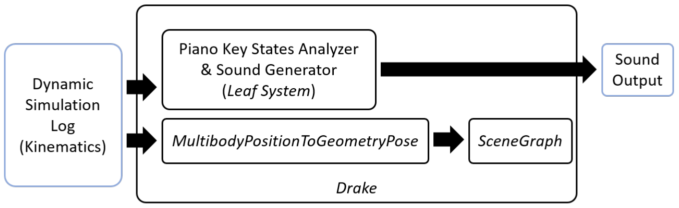

Dynamic Simulation Log (Kinematics)
Piano Key States Analyzer & Sound Generator (Leaf System)
MultibodyPositionToGeometryPose
SceneGraph
Drake
Sound Output

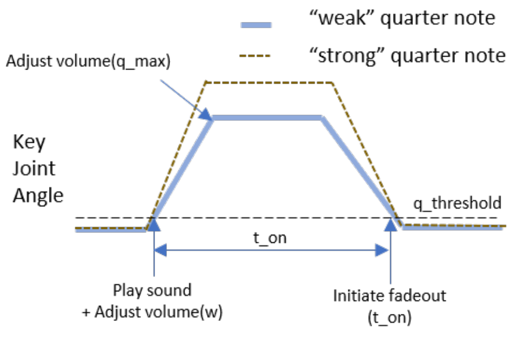
Fig. 9: Structure diagram of the sound generator block.

2) Regulating Piano Sound: As shown in fig. 9, the sound of piano was generated based on "kinematics" of the piano. In other words, contact force information was not available in this setup. In order to recreate sound in a "causal" manner in real-time while not incorporating much of a system "delay", I formulated an heuristic music regulation algorithm based on the mechanical state of each key.

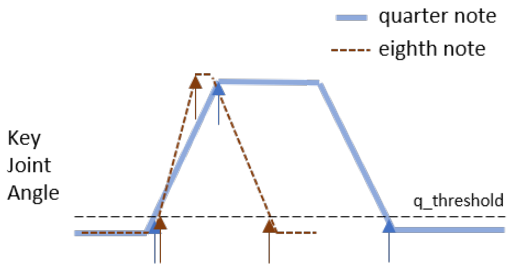
In regulating sound, there exists three control points of sound when the key is pressed by finger. First control happens when the key position q exceeds the $q_{threshold}$. At this event, a corresponding sound of the pressed key is played with adjusted volume proportional to the velocity of the key $\dot{q}$. When key reaches its apex, second adjustment of the volume is executed based on the maximum joint position q_{max}. At last, when the joint position q gets lower than $q_{threshold}$, it starts to fade out with the duration proportional to its duration of being pressed. Fig. 10 shows the example of regulated sound generation with different note lengths and different strength of notes being pressed. This heuristic was largely tuned with trial and error approach. This approach has a similar insight with this work [3].

III. RESULTS

The piano playing robotic hand system described in Section II was successfully implemented and tested. For the qualitative evaluation of the system, a classic music song, Salut D'Amour by Edward Elgar, was chosen [11]. The succesful result of this project was presented on Youtube link https://youtu.be/WzJJ4c6AqnE [12]. The fig. 11 shows partial maneuvers in playing the song.



(a)



(b)

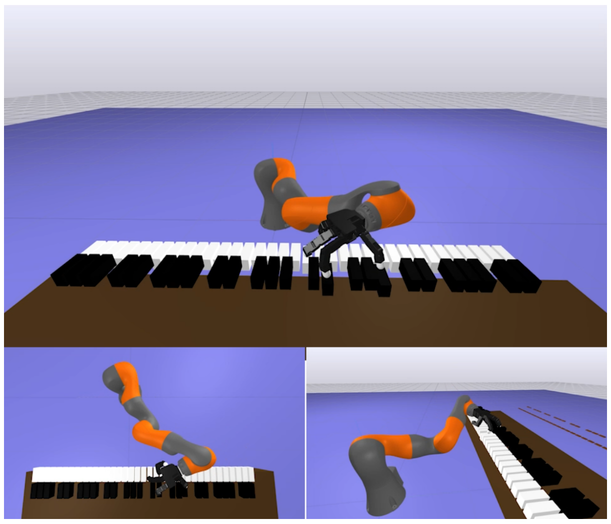
Fig. 10: Demonstration of sound regulation heuristic algorithms based on its scenarios. (a) Sound generator playing the two notes with different intensity. (b) Sound generator playing the two notes with different duration.

IV. DISCUSSION

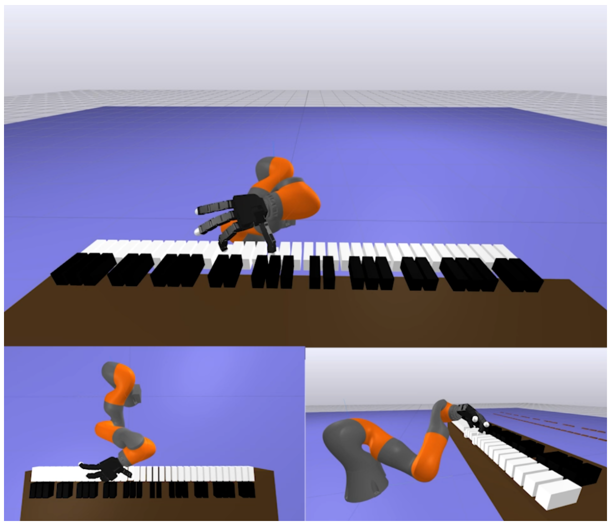
In this report, I present a robotic framework enabling a robotic hand to process and play the music with mechanical piano. While this project provides interesting full-stack pipeline to process a music sheet into a robotic framework, the goal of this project is to show its feasibility as a prototype within limited duration of development. While many heuristic algorithms were effectively developed to process data for the systematic demonstration, it also implies that the presented system has a clear limitations in many aspects. To list a few, the music interpretation framework has a clear limitation in coding a advanced music sheet with complex and emotional finger maneuvers, as well as the dynamic controller is merely a simplest joint trajectory follower without the ability to compensate its inertia resulting its expected errors in fast maneuvers.

For the potential future works, this project can be extended to a real-world robotic arm and hand. In order to extend the project, several processing blocks need to be matured significantly.To list a few,tilization of tactile sensor on the fingers would also be required. As discussed above, advanced torque controller would be essential to deploy this framework in real world. Also, incorporating vision-based processing would be beneficial for using piano state feedback. Expanding this problem with RL-based controller would also provide some interesting questions as well.

To summarize, through this project, I intended to demon-



(a)



(b)

Fig. 11: Partial capture of the result demonstration.

strate that the robotic system can demonstrate diverse interesting maneuver with even generic and basic building blocks of robotics when they are integrated in a proper and effective manner. Successful demonstration of playing piano with a 23-DoF robotic hand successfully achieved my goal.

References

[1] Yen-Fang Li and Ci-Yi Lai. Development on an intelligent music score input system—applied for the piano robot. In *2016 Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 67–71. IEEE, 2016.

[2] Dan Zhang, Jianhe Lei, Beizhi Li, Daniel Lau, and Craig Cameron. Design and analysis of a piano playing robot. In *2009 International Conference on Information and Automation*, pages 757–761. IEEE, 2009.

[3] Yen-Fang Li and Chun-Wei Huang. Force control for the fingers of the piano playing robot—a gain switched approach. In *2015 IEEE 11th international conference on power electronics and drive systems*, pages 265–270. IEEE, 2015.

[4] Vinay Jaju, Amit Sukhpal, Prapti Shinde, Aditya Shroff, and Archana B Patankar. Piano playing robot. In *2016 International Conference on Internet of Things and Applications (IOTA)*, pages 223–226. IEEE, 2016.

[5] Wonik Robotics. Allegro hand specifications. https://www.wonikrobotics.com/research-robot-hand.

[6] KUKA. Lbr iiwa 7 specifications. https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa.

[7] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987.

[8] Eric R Westervelt, Jessy W Grizzle, and Daniel E Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE transactions on automatic control*, 48(1):42–56, 2003.

[9] Dong Jin Hyun, Sangok Seok, Jongwoo Lee, and Sangbae Kim. High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah. *The International Journal of Robotics Research*, 33(11):1417–1445, 2014.

[10] Pete Shinners. Pygame. http://pygame.org/, 2011.

[11] Edward Elgar. Wikipedia page of salut d'amour. https://en.wikipedia.org/wiki/Salut_d%27Amour, 2011.

[12] Seong Ho Yeon. Playing piano with a robotic hand. https://youtu.be/WzJJ4c6AqnE, 2011.