RP1M: A Large-Scale Motion Dataset for Piano Playing with Bi-Manual Dexterous Robot Hands

Le Chen[1*], Yi Zhao[1,2*†], Jan Schneider[1], Quankai Gao[3], Juho Kannala[2,4], Bernhard Schölkopf[1], Joni Pajarinen[2], and Dieter Büchler[1]

***Abstract*—It has been a long-standing research goal to endow robot hands with human-level dexterity. Bi-manual robot piano playing constitutes a task that combines challenges from dynamic tasks, such as generating fast while precise motions, with slower but contact-rich manipulation problems. Although reinforcement learning based approaches have shown promising results in single-task performance, these methods struggle in a multi-song setting. Our work aims to close this gap and, thereby, enable imitation learning approaches for robot piano playing at scale. To this end, we introduce the *Robot Piano 1 Million* (RP1M) dataset, containing bi-manual robot piano playing motion data of more than one million trajectories. We formulate finger placements as an optimal transport problem, thus, enabling automatic annotation of vast amounts of unlabeled songs.**

I. Introduction

Empowering robots with human-level dexterity is notoriously challenging. Current robotics research on hand and arm motions focuses on manipulation and dynamic athletic tasks. Manipulation, such as grasping or reorienting [36], requires continuous application of acceptable forces at moderate speeds to various objects with distinct shapes and weight distributions. Environmental changes, like humidity or temperature, alter the already complex contact dynamics, which adds to the complexity of manipulation tasks. Dynamic tasks, like juggling [39] and table tennis [6] involve making and breaking contact, demanding high precision and tolerating less inaccuracy due to rarer contacts. High speeds in these tasks necessitate greater accelerations and introduce a precision-speed tradeoff.

Robot piano playing combines various aspects of dynamic and manipulation tasks: the agent is required to coordinate multiple fingers to precisely press keys for arbitrary songs, which is a high-dimensional and rich control task. Also, the finger motions have to be highly dynamic, especially for songs with fast rhythms. Experienced pianists can play arbitrary songs, but this level of generalization is extremely challenging for robots. In this work, we build the foundation to develop methods capable of achieving human-level bi-manual dexterity at the intersection of manipulation and dynamic tasks, making it possible to reach such generalization capabilities.

While reinforcement learning (RL) is a promising direction, traditional RL approaches often struggle to achieve excellent performance in multi-task settings [61]. The advent of scalable imitation learning techniques [12] enables representing complex and multi-modal distributions. Such large models are most effective when trained on massive datasets that combine the state evolution with the corresponding action trajectories. So far, creating large datasets for robot piano play is problematic due to the time-consuming fingering annotations. Fingering annotations map which finger is supposed to press a particular piano key at each time step. With fingering information, the reward is less sparse, making the training significantly more effective. These labels usually require expert human annotators [35], preventing the agent from leveraging the large amounts of unlabeled music pieces on the internet [22]. Besides, human-labeled fingering may be infeasible for robots with morphologies different from human hands, such as different numbers of fingers or distinct hand dimensions.

In this paper, we propose the *Robot Piano 1 Million* (RP1M) dataset. This dataset comprises the motion data of high-quality bi-manual robot piano play. In particular, we train RL agents for each of the 2k songs and roll out each policy 500 times with different random seeds. To enable the generation of RP1M, we introduce a method to learn the fingering automatically by formulating finger placement as an optimal transport (OT) problem [55, 38]. Intuitively, the fingers are placed in a way such that the correct keys are pressed while the overall moving distance of the fingers is minimized. Agents trained using our automatic fingering match the performance of agents trained with human-annotated fingering labels. Besides, our method is easy to implement with almost no extra training time. The automatic fingering also allows learning piano playing with different embodiments, such as robots with four fingers only.

II. Background

Task Setup The simulated piano-playing environment is built upon RoboPianist [61]. It includes a robot piano-playing setup, an RL-based agent for playing piano with simulated robot hands, and a multi-task learner. To avoid confusion, we refer to these components as *RoboPianist*, *RoboPianist-RL*, and *RoboPianist-MT*, respectively. The environment features a full-size keyboard with 88 keys driven by linear springs, two Shadow robot hands [49], and a pseudo sustain pedal.

Sheet music is represented by Musical Instrument Digital Interface (MIDI) transcription. Each time step in the MIDI file specifies which piano keys to press (active keys). The goal of a piano-playing agent is to press active keys and avoid inactive keys under *space* and *time* constraints. This requires

*Authors contributed equally and both can be considered as first authors.
†Work done during an internship at MPI-IS.
[1]Max Planck Institute for Intelligent Systems, Germany. firstname.lastname@tuebingen.mpg.de
[2]Aalto University, Finland. firstname.lastname@aalto.fi
[3]University of Southern California, USA. quankaig@usc.edu
[4]University of Oulu, Finland.

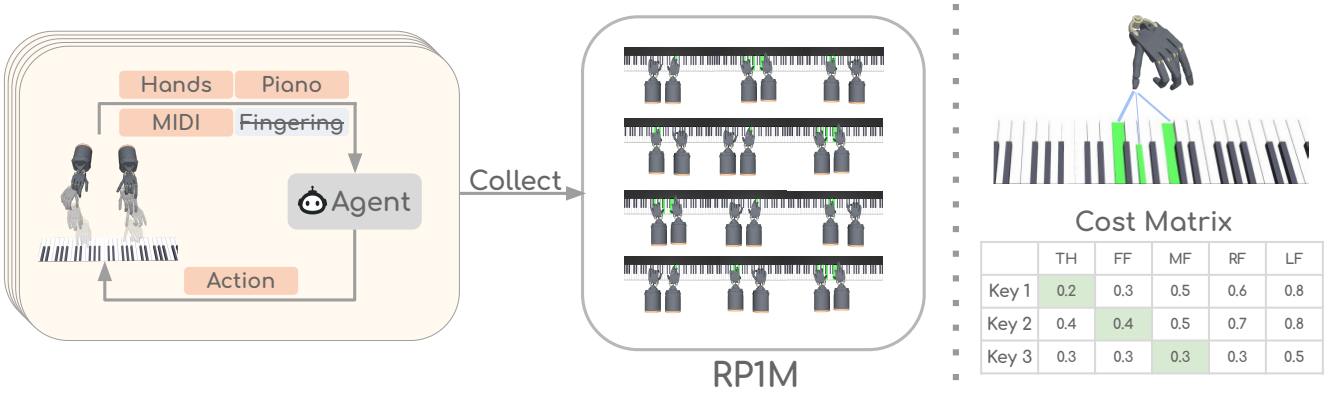

Fig. 1. Overview of RP1M. (**Left**) RP1M is a large-scale motion dataset for piano playing with bi-manual dexterous robot hands. The dataset includes $\sim$*1M* expert trajectories collected by $\sim$*2k* RL specialist agents. (**Right**) To collect a diverse motion dataset of playing sheet music, we lift the requirement of human-annotated fingering by formulating the finger placement as an optimal transport problem such that the robot hands play piano in an energy-efficient way.

the agent to coordinate its fingers and place them properly in a highly dynamic scenario such that target keys can be pressed accurately and timely at not only the current time step but also the future time steps. The original RoboPianist uses MIDI files from the PIG dataset [35] which includes *human fingering* information annotated by experts. However, as mentioned earlier, this limits the agent to only play human-labeled music pieces, and the human annotation may not be suitable for robots due to the different morphologies.

The observation includes the state of the two robot hands, fingertip positions, piano sustain state, piano key states, and a goal vector, resulting in an 1144-dimensional observation space. The goal includes 10-step active keys and 10-step target sustain states obtained from the MIDI file, represented by a binary vector. RoboPianst further includes 10-step human-labeled fingering in the observation space but we remove this observation in our method since we do not need human-labeled fingering. For the action space, we remove the DoFs that do not exist in the human hand or are used in most songs, resulting in a 39-dimensional action space, consisting of the joint positions of the robot hands, the positions of forearms, and a sustain pedal. We evaluate the performance of the trained agent with an average F1 score calculated by $F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$. For piano playing, recall and precision measure the agent's performance on pressing the active keys and avoiding inactive keys respectively [61].

Playing Piano with RL We use RL to train specialist agents per song to control the bi-manual dexterous robot hands to play the piano. We frame the piano playing task as a finite Markov Decision Process (MDP). At time step t, the agent $\pi_\theta(a_t|s_t)$, parameterized by θ, receives state s_t and takes action a_t to interact with the environment and receives new state s_{t+1} and reward r_t. The state and action spaces are described above and the reward r_t gives an immediate evaluation of the agent's behavior. We will introduce reward terms used for training in Section III-A. The agent's goal is to maximize the expected cumulative rewards over an episode of length H, defined as $\mathcal{J} = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{H} \gamma^t r_t(s_t, a_t)\right]$, where γ is a discount factor ranging from 0 to 1.

III. Large-Scale Motion Dataset Collection

In this section, we describe our RP1M dataset in detail. We first introduce how to train a specialist piano-playing agent without human fingering labels. Removing the requirement of human fingering labels allows the agent to play any sheet music available on the Internet. We then analyze the performance of our specialist RL agent as well as the learned fingering. Lastly, we introduce our collected large-scale motion dataset, RP1M, which includes $\sim$1M expert trajectories for robot piano playing, covering $\sim$2k pieces of music.

A. Piano Playing without Human Fingering Labels

To mitigate the hard exploration problem posed by the sparse rewards, RoboPianist-RL adds dense reward signals by using human fingering labels. Fingering informs the agent of the "ground-truth" fingertip positions, and the agent minimizes the Euclidean distance between the current fingertip positions and the "ground-truth" positions. We now discuss our OT-based method to lift the requirement of human fingering.

Although fingering is highly personalized, generally speaking, it helps pianists to press keys timely and efficiently. Motivated by this, apart from maximizing the key pressing rewards, we also aim to minimize the moving distances of fingers. Specifically, at time step t, for the i-th key k^i to press, we use the j-th finger f^j to press this key such that the overall moving cost is minimized. We define the minimized cumulative moving distance as $d_t^{\text{OT}} \in \mathbb{R}^+$, which is given by

$$
\begin{aligned}
d_t^{\text{OT}} &= \min_{w_t} \sum_{(i,j)\in K_t \times F} w_t(k^i, f^j) \cdot c_t(k^i, f^j), \\
\text{s.t.,} \quad i) \; & \sum_{j\in F} w_t(k^i, f^j) = 1, \quad \text{for} \quad i \in K_t, \\
ii) \; & \sum_{i\in K_t} w_t(k^i, f^j) \le 1, \quad \text{for} \quad j \in F, \\
iii) \; & w_t(k^i, f^j) \in \{0, 1\}, \quad \text{for} \quad (i,j) \in K_t \times F.
\end{aligned} \tag{1}
$$

K_t represents the set of keys to press at time step t and F represents the fingers of the robot hands. $c_t(k^i, f^j)$ represents the cost of moving finger f^j to piano key k^i at time step t calculated by their Euclidean distance. $w_t(k^i, f^j)$ is a boolean weight. In our case, it enforces that each key in K_t will be

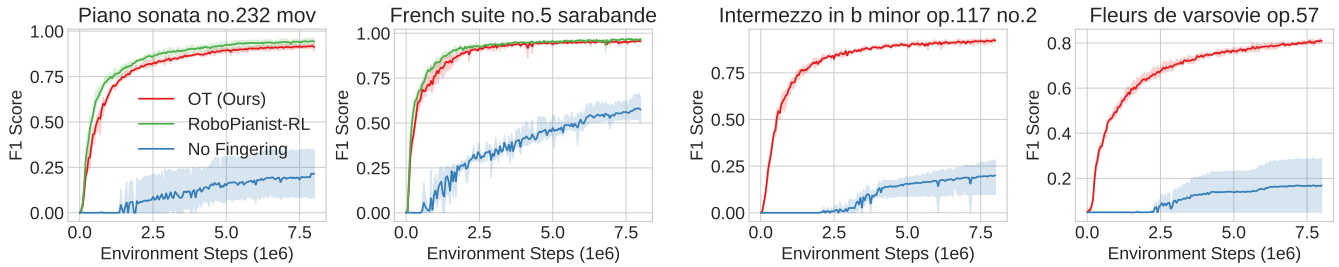

Fig. 2. Comparison of the RL performance with our OT fingering, human-annotated fingering, and no fingering. Our method matches the performance of RoboPianist-RL, which is trained with human fingering. We also outperforms the baseline without any fingering information by a large margin. The plots show the mean over 3 random seeds and the shaded areas represent the 95% confidence interval.

pressed by only *one* finger in F, and each finger presses *at most* one key. The constrained optimization problem in Eq. (1) is an optimal transport problem. Intuitively, it tries to find the best "transport" strategy such that the overall cost of moving (a subset of) fingers F to keys K_t is minimized. We solve this optimization problem with a modified Jonker-Volgenant algorithm [13] from SciPy [56] and use the optimal combinations (i^*, j^*) as the fingering for the agent.

We define a reward r_t^{OT} based on d_t^{OT} to encourage the agent to move the fingers close to the keys K_t, which is defined as:

$$r_t^{\text{OT}} = \begin{cases} \exp(c \cdot (d_t^{\text{OT}} - 0.01)^2) & \text{if } d_t^{\text{OT}} \geq 0.01, \\ 1.0 & \text{if } d_t^{\text{OT}} < 0.01. \end{cases} \quad (2)$$

c is a constant scale value and we use the same value as Tassa et al. [53]. The overall reward function is defined as:

$$r_t = r_t^{\text{OT}} + r_t^{\text{Press}} + r_t^{\text{Sustain}} + \alpha_1 \cdot r_t^{\text{Collision}} + \alpha_2 \cdot r_t^{\text{Energy}} \quad (3)$$

r^{Press} and r_t^{Sustain} represent the reward for correctly pressing the target keys and the sustain pedal. $r_t^{\text{Collision}}$ encourages the agent to avoid collision between forearms and r_t^{Energy} prefers energy-saving behaviors. α_1 and α_2 are coefficient terms, and $\alpha_1 = 0.5$ and $\alpha_2 = 5 \cdot 10^{-3}$ are adopted. Our method is compatible with any RL methods, and we use DroQ [18] here.

B. Analysis of Specialist RL Agents

The performance of specialist RL agents decides the quality of our dataset. In this section, we investigate how our specialist RL agents perform. We are interested in i) how the proposed OT-based finger placement helps learning, ii) how the fingering discovered by the agent itself compares to human fingering, and iii) how our method transfers to other embodiments.

Results In Fig. 2, we compare our method with RoboPianist-RL both with and without human fingering. We use the same DroQ algorithm with the same hyperparameters for all experiments. RoboPianist-RL includes human fingering in its inputs, and the fingering information is also used in the reward function to force the agent to follow this fingering. Our method, marked as *OT*, removes the fingering from the observation space and uses OT-based finger placement to guide the agent to discover its own fingering. We also include a baseline, called *No Fingering*, that removes the fingering entirely. The first two columns of Fig. 2 show that our method without human-annotated fingering matches RoboPianst-RL's performance on two different songs. Our method outperforms the baseline without human fingering by a large margin, showing that the proposed OT-based finger placement boosts agent learning. The proposed method works well even on challenging songs. We test our method on *Flight of the Bumblebee* and achieve 0.79 F1 score after 3M training steps.

Analysis of the Learned Fingering We now compare the fingering discovered by the agent itself and the human annotations. In Fig. 3, we visualize the sample trajectory of playing *French Suite No.5 Sarabande* and the corresponding fingering. We found that although the agent achieves strong performance for this song (the second plot in Fig. 2), our agent discovers different fingering compared to humans. For example, for the right hand, humans mainly use the middle and ring fingers, while our agent uses the thumb and first finger. Furthermore, in some cases, human annotations are not suitable for the robot hand due to different morphologies. For example, in the second time step of Fig. 3, the human uses the first finger and ring finger. However, due to the mechanical limitation of the robot hand, it can not press keys that far apart with these two fingers, thus mimicking human fingering will miss one key. Instead, our agent discovered to use the thumb and little finger, which satisfies the hardware limitation and accurately presses the target keys.

Cross Emboidments Labs usually have different robot platforms, thus having a method that works for different embodiments is highly desirable. We test our method on a different embodiment. To simplify the experiment, we disable the little finger of the Shadow robot hand and obtain a four-finger robot hand, which has a similar morphology to Allegro [2] and LEAP Hand [50]. We evaluate the modified robot hand on the song French Suite No.5 Sarabande, where our method achieves a 0.95 F1 score, similar to the 0.96 achieved with the original robot hands. In the bottom row of Fig. 3, we visualize the learned fingering with four-finger hands. The agent discovers different fingering compared to humans and the original hands but still accurately presses active keys, meaning our method is compatible with different embodiments.

C. RP1M Dataset

To facilitate the research on dexterous robot hands, we collect and release a large-scale motion dataset for piano playing. Our dataset includes ∼1M expert trajectories covering ∼2k musical pieces. For each musical piece, we train an individual DroQ agent with the method introduced in Section III-A for 8 million

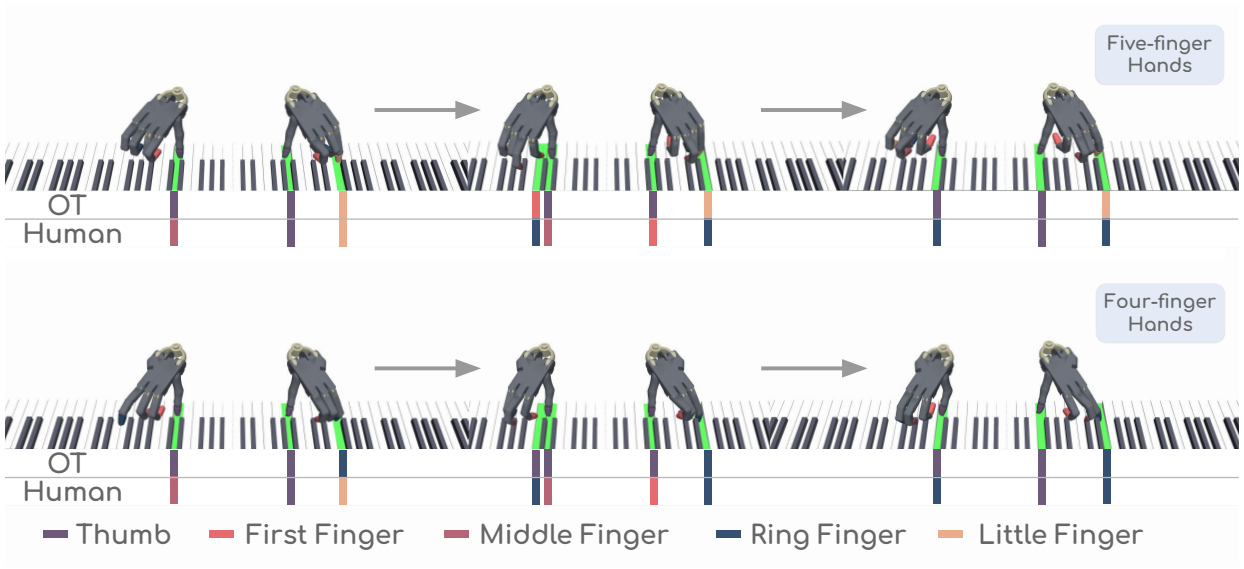

Fig. 3. Comparison of fingering discovered by the agent itself and human annotations.

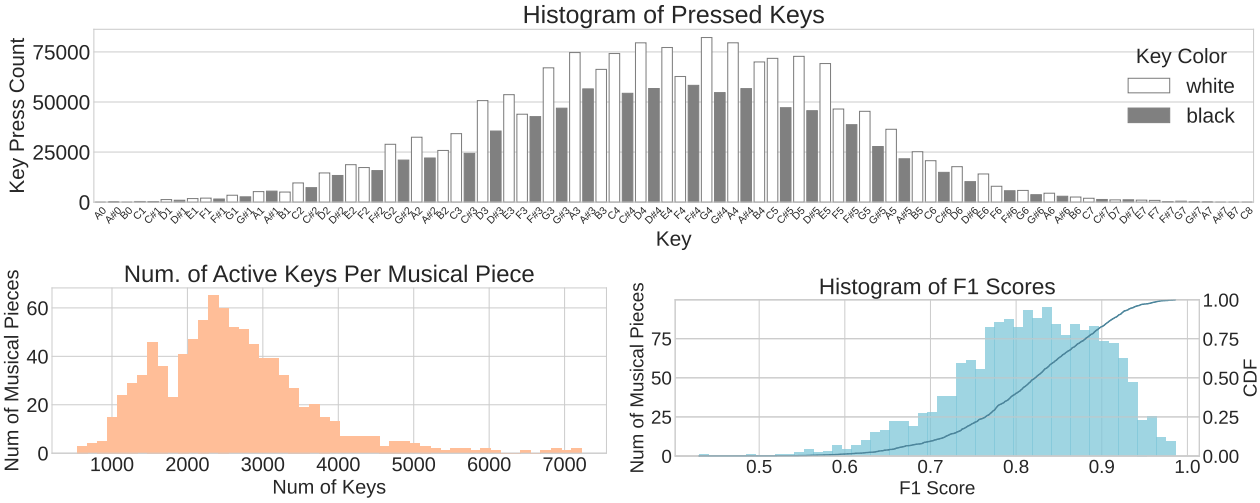

Fig. 4. Statistics of our RP1M dataset. (**Top**) Histogram of pressed keys in our RP1M dataset. (**Bottom Left**) Distribution of the number of keys to press in one musical piece. (**Bottom Right**) Distribution of F1 scores in our dataset.

environment steps and collect 500 expert trajectories with the trained agent. We chunk each sheet music every 550 time steps, corresponding to 27.5 seconds, so that each run has the same episode length. The sheet music used for training is from the PIG dataset [35] and a subset (1788 pieces) of the GiantMIDI-Piano dataset [22].

In Fig. 4, we show the statistics of our collected motion dataset. The top plot shows the histogram of the pressed keys. We found that keys close to the center are more frequently pressed than keys at the corner. Also, white keys, taking 65.7%, are more likely to be pressed than black keys. In the bottom left plot, we show the distribution of the number of keys to press in each musical piece. It roughly follows a Gaussian distribution, and 90.70% musical pieces in our dataset include 1000-4000 active keys. We also include the distribution of F1 scores of trained agents used for collecting data. We found most agents (79.00%) achieve F1 scores larger than 0.75, and 99.89% of the agents' F1 scores are larger than 0.5. The distribution of F1 scores reflects the quality of the collected dataset. We empirically found agents with F1 score ≥ 0.75 are capable of playing sheet music reasonably well with only minor errors. Agents with ≤ 0.5 F1 scores usually have notable errors due to the difficulty of songs or the mechanical limitations of the Shadow robot hand. We also include the F1 scores for each piece in our dataset.

IV. Conclusion

In this paper, we propose a large-scale motion dataset named RP1M for piano playing with bi-manual dexterous robot hands. RP1M includes 1M expert trajectories for playing 2k musical pieces. To collect such a diverse dataset for piano playing, we lift the need for human-annotated fingering in the previous method by introducing a novel automatic fingering annotation approach based on optimal transport. On single songs, our method matches the baselines with human-annotated fingering and can be adopted across different embodiments. We believe the RP1M dataset, with its scale and quality, forms a solid step towards empowering robots with human-level dexterity.

APPENDIX

A. Related Work

Dexterous Robot Hands The research of dexterous robot hands aims to replicate the dexterity of human hands with robots. Many previous works [47, 5, 17, 4, 32, 14, 8, 23] use planning to compute a trajectory followed by a controller, thus require an accurate model of the robot hand. Closed-loop approaches have been developed by incorporating sensor feedback [26]. These methods also require an accurate model of the robot hand, which can be difficult to obtain in practice, especially considering the large number of active contacts between the hand and objects.

Due to the difficulty of actually modeling the dynamics of the dexterous robot hand, recent methods resort to learning-based approaches, especially RL, which has achieved huge success in both robotics [25, 1, 36] and computer graphics [37]. To ease the training of dexterous robot hands with a large number of degrees of freedom (DoFs), demonstrations are commonly used [24, 43, 42, 20]. Due to the advance of both RL algorithms and simulation, recent work shows impressive results on dexterous hand manipulation tasks without human demonstrations. Furthermore, the policy trained in the simulator can further be deployed on real dexterous robot hands via sim-to-real transfer [9, 60, 10, 59, 3, 41].

Piano Playing with Robots Piano playing with robot hands has been investigated for decades. It is a challenging task since bi-manual robot hands should precisely press the right keys at the right time, especially considering its high-dimensional action space. Previous methods require specific robot designs [21, 28, 54, 19, 7, 63] or trajectory pre-programming [27, 62]. Recent methods enable piano playing with dexterous hands through planning [48] or RL [58] but are limited to simple music pieces. RoboPianist [61] introduces a benchmark for robot piano playing and demonstrates strong RL performance, but requires human fingering labels and performs worse in multi-task learning.

Human fingering informs the agent of the correspondence between fingers and pressed keys at each time step. These labels require expert annotators and are, therefore, expensive to acquire in practice. Several approaches learn fingering from human-annotated data with different machine learning methods [35, 45, 46]. Moryossef et al. [33] extract fingering from videos to acquire fingering labels cheaply. Ramoneda et al. [44] propose to treat piano fingering as a sequential decision-making problem and use RL to calculate fingering but without considering the model of robot hands. Shi et al. [51] automatically acquires fingering via dynamic programming, but the solution is limited to simple tasks. In our paper, we do not introduce a separate fingering model, instead, similar to human pianists, fingering is *discovered automatically* while playing the piano.

Datasets for Dexterous Robot Hands Most large-scale datasets of dexterous robot hands focus on grasping various objects. To get suitable grasp positions, some methods utilize planners [29, 57, 34], while others use learned grasping policies [59], or track grasping motions of humans and imitate these motions on a robot hand [30]. Compared to the abundance of datasets for grasping, there exist relatively few datasets for object manipulation with dexterous robot hands. The D4RL benchmark [16] provides small sets of expert trajectories for four such tasks, consisting of human demonstrations and rollouts of trained policies. Zhao et al. [64] provide a small object manipulation dataset that utilizes a low-cost bi-manual platform with simple parallel grippers. Chen et al. [11] collect offline datasets for two simulated bi-manual manipulation tasks with dexterous hands. Table I summarizes the characteristics of these existing datasets. To the best of our knowledge, our RP1M dataset is the first large-scale dataset of dynamic, bi-manual manipulation with dexterous robot hands.

B. Benchmarking

The analysis in the previous section highlighted the diversity of highly dynamic piano-playing motions in the RP1M dataset. In this section, we assess the multi-task imitation learning performance of several widely used methods on our benchmark. To be specific, the objective is to train a single multi-task policy capable of playing various music pieces on the piano. We train the policy on a portion of the RP1M dataset and evaluate its in-distribution performance (F1 scores on songs included in the training data) and its generalization ability (F1 scores on songs not present in the training data).

Baselines We evaluated Behavior Cloning (BC) [40], Implicit Behavioral Cloning (IBC) [15], BC with a Recurrent Neural Network policy (BC-RNN) [31], and Diffusion Policy [12]. BC directly learns a policy by using supervised learning on observation-action pairs from expert demonstrations. IBC learns an implicit policy as an energy-based model conditioned on observation and action. BC-RNN uses an RNN as the policy network to encode a history of observations. Diffusion Policy learns to model the action distribution by inverting a process that gradually adds noise to a sampled action sequence. We used a CNN-based Diffusion Policy with DDIM [52] as the sampler. We use the same code and hyperparameters as Chi et al. [12].

Experiment Setup We first train the policies with 3 different sizes of expert data: 50, 150, and 300 songs. We then evaluate the trained policies on 3 different groups of music pieces. (1) In-distribution songs: music pieces that overlap with the training sets. It shows the multitasking performance of the trained policies. (2) Easy out-of-distribution (OOD) songs: simple music pieces that do not overlap with the training songs. Those pieces are easy to play, with only slow motions and short horizons. (3) Hard out-of-distribution songs: difficult music pieces that do not overlap with the training songs. They contain more diverse motions and longer horizons. The out-of-distribution evaluation assesses the zero-shot generalization ability of the trained policies. We report the average F1 scores of each group of music pieces for policies trained with each baseline method.

Discussion As shown in Table II, most baselines including BC (256), IBC, and BC-RNN have worse performance. This

TABLE I
EXISTING DATASETS ON DEXTEROUS OR BI-MANUAL ROBOTIC MANIPULATION.

Dataset	Task	Dexterous hands	Bi-manual	Dynamic tasks	Demonstrations
DexGraspNet [57]	grasping	✓			1.3M
RealDex [30]	grasping	✓			2.6K
UniDexGrasp [59]	grasping	✓			1.1M
ALOHA [64]	manipulation		✓		825
Bi-DexHands [11]	manipulation	✓	✓	partially	∼20K
D4RL [16] (Adroit)	manipulation	✓			30K
RP1M (ours)	piano	✓	✓	✓	1M

TABLE II
COMPARISON RESULTS OF MULTI-TASK IMITATION LEARNING.

Task	#music	BC			IBC	BC-RNN	Diffusion Policy
		256	1024	4096			
In-Dist.	50	0.086	0.621	0.200	0.120	0.174	0.706
	150	0.124	0.245	0.176	0.088	0.121	0.578
	300	0.084	0.249	0.102	0.083	0.025	0.596
Easy OOD	50	0.06	0.278	0.109	0.128	0.204	0.446
	150	0.085	0.257	0.172	0.058	0.050	0.465
	300	0.056	0.236	0.112	0.056	0.039	0.486
Hard OOD	50	0.141	0.244	0.217	0.183	0.221	0.303
	150	0.155	0.275	0.229	0.148	0.105	0.432
	300	0.145	0.253	0.181	0.144	0.081	0.440

is because of the limited model capacity. We increase the model capacity of the BC baseline by increasing the hidden dimension of a 3-layer MLP from 256 to 1024 and we observe clear improvement in both in-distribution evaluation and OOD evaluation. However, when further increasing the model size to a 6-layer MLP with 4096 hidden dimensions, a performance drop is observed, meaning directly increasing the model capacity of MLP causes issues for model training. Among the baselines we have evaluated, Diffusion Policy performs the best in all cases, demonstrating that it is a strong baseline for the piano-playing task. Furthermore, when increasing the size of the training set, the OOD performance gradually increases. However, for in-distribution evaluation, Diffusion Policy still has a lower F1 score than our RL specialist and we observed a performance drop when increasing the dataset size, which indicates the Diffusion Policy has issues fitting our dataset well. This can be caused by multiple reasons, e.g., the limited model capacity or improper hyperparameters but we leave it as future work.

C. Limitations

Our paper has limitations in several aspects. Firstly, although our method lifts the requirement of human-annotated fingering, enabling RL training on diverse songs, our method still fails to achieve strong performance on challenging songs due to fast rhythms and mechanical limitations of the robot hands. This could be solved by proposing a better RL method and improving the hardware design of the robot hands. Secondly, our dataset only includes proprioceptive observations. However, humans play piano with multi-modal inputs, including vision, tactile sensing, and auditory information. Enabling the agent to play the piano from such rich input sources is an intriguing direction. Lastly, although we demonstrate better zero-shot generalization performance than RoboPianist-MT [61], there is still a gap between our best multi-task agent and RL specialists, which requires future investigation. We believe our RP1M dataset is a fundamental step toward empowering robot hands with human-level dexterity.

D. RP1M Dataset Collection Details

1) Reward formulation: In Equation (3) , we give the overall reward function used in our paper. We now give details of each term. r_t^{Press} indicates whether the active keys are correctly pressed and inactive keys are not pressed. We use the same implementation as [61], given as: $r_t^{\text{Press}} = 0.5 \cdot (\frac{1}{K} \sum_t^K g(||k_s^i - 1||_2)) + 0.5 \cdot (1 - \mathbf{1}_{\text{fp}})$. K is the number of active keys, k_t^i is the normalized key states with range [0, 1], where 0 means the i-th key is not pressed and 1 means the key is pressed. g is tolerance from Tassa et al. [53], which is similar to the one used in Equation (2). $\mathbf{1}_{\text{fp}}$ indicates whether the inactive keys are pressed, which encourages the agent to avoid pressing keys that should not be pressed. r_t^{Sustain} encourages the agent to press the pseudo sustain pedal at the right time, given as $r_t^{\text{Sustain}} = g(s_t - s_t^{\text{target}})$. s_t and s_t^{target} are the state of current and target sustain pedal respectively. $r_t^{\text{Collision}}$ penalizes the agent from collision, defined as $r_t^{\text{Collision}} = 1 - \mathbf{1}_{\text{collision}}$, where $\mathbf{1}_{\text{collision}}$ is 1 if collision happens and 0 otherwise. r_t^{Energy} prioritizes energy-saving behavior. It is defined as $r_t^{\text{Energy}} = |\tau_{\text{joints}}|^\intercal |\mathbf{v}_{\text{joints}}|$. τ_{joints} and $\mathbf{v}_{\text{joints}}$ are joint torques and joint velocities respectively.

2) Training details:

a) Observation Space: Our 1144-dimensional observation space includes the proprioceptive state of dexterous robot hands and the piano as well as L-step goal states obtained from the MIDI file. In our case, we include the current goal and 10-step future goals in the observation space (L=11). At each time step, an 89-dimensional binary vector is used to represent the goal, where 88 dimensions are for key states and the last dimension

is for the sustain pedal. The dimension of each component in the observation space is given in Table III.

TABLE III
OBSERVATION SPACE.

Observations	Dim
Piano goal state	L · 88
Sustain goal state	L · 1
Piano key joints	88
Piano sustain state	1
Fingertip position	30
Hand state	46

b) Training Algorithm & Hyperparameters: Although our proposed method is compatible with any reinforcement learning method, we choose the DroQ [18] as Zakka et al. [61] for fair comparison. DroQ is a model-free RL method, which uses Dropout and Layer normalization in the Q function to improve sample efficiency. We list the main hyperparameters used in our RL training in IV.

TABLE IV
HYPERPARAMETERS USED IN OUR RL AGENT.

Hyperparameter	Value
Training steps	8M
Episode length	550
Action repeat	1
Warm-up steps	5k
Buffer size	1M
Batch size	256
Update interval	2
Piano environment	
Lookahead steps	10
Gravity compensation	True
Control timestep	0.05
Stretch factor	1.25
Trim slience	True
Agent	
MLPs	[256, 256, 256]
Num. Q	2
Activation	GeLU
Dropout Rate	0.01
EMA momentum	0.05
Discount factor	0.88
Learnable temperature	True
Optimization	
Optimizer	Adam
Learning rate	3e-4
β_1	0.9
β_2	0.999
eps	1e-8

3) Computational resources: We train our RL agents on the cluster equipped with AMD MI250X GPUs, 64 cores AMD EPYC "Trento" CPUs, and 64 GBs DDR4 memory. Each agent takes 21 hours to train. The overall data collection cost is roughly 21 hours * 2089 agents = 43,869 GPU hours.

E. MuJoCo XLA Implementation

To speed up training, we re-implement the RoboPianist environment with MuJoCo XLA (MJX), which supports simulation in parallel with GPUs. MJX has a slow performance with complex scenes with many contacts. To improve the simulation performance, we made the following modifications:

- We disable most of the contacts but only keep the contacts between fingers and piano keys as well as the contact between forearms.
- Primitive contact types are used whenever possible.
- The dimensionality of the contact space is set to 3.
- The maximal contact points are set to 20.
- We use Newton solver with iterations=2 and ls_iterations=6.

After the above modifications, with 1024 parallel environments, the total steps per second is 159,376.

We use PPO implementation implemented with Jax to fully utilize the paralleled simulation. The PPO with MJX implementation is much faster than the DroQ implementation, which only takes 2 hours and 7 minutes for 40M environment steps on the Twinkle Twinkle Little Star song while as a comparison, DroQ needs roughly 21 hours for 8M environment steps. However, the PPO implementation fails to achieve a comparable F1 score as the DroQ implementation as shown in Fig. 5. Therefore, we use the DroQ implement with the CPU version of the RoboPianist environment.

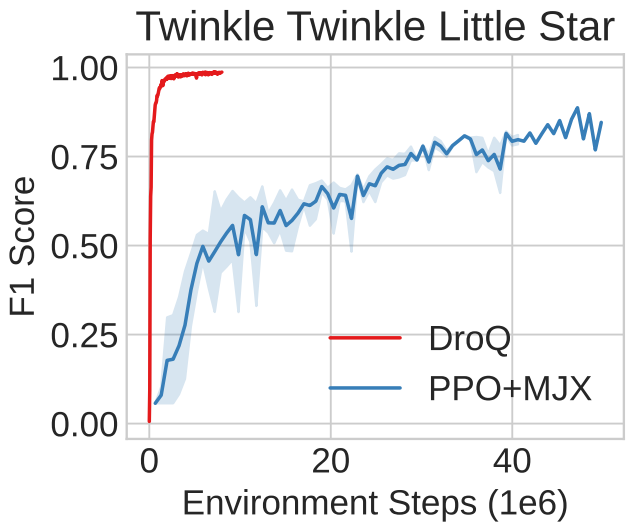

Fig. 5. Comparison of the RL performance between DroQ and PPO with the MJX implementation of the RoboPianist environment. PPO+MJX is faster to run but has a worse performance than DroQ. We use DroQ with the CPU-version RoboPianist environment when training our RL agents.

F. Discussion on Real-world Applications

Since we can access all observations provided by the simulator, in this paper, we assume full observation is accessible. In

real-world scenarios, full observation is typically unattainable, hence, one needs to resort to sensor measurements such as tactile sensing, hand states, and pixel inputs. Including these sensor measurements in our dataset would be interesting, which we plan to explore in future work. Even then, a significant gap between simulation and real setup will remain due to the unmatched dynamics of the keyboard and robot. Potential approaches to address the sim-to-real gap include data augmentations [36, 25] and teacher-student training, where a policy derived from our dataset serves as the teacher. Our dataset lays the groundwork for bridging the sim-to-real gap by enabling high-performance multi-task agents in simulation.

ACKNOWLEDGMENTS

We thank the support of the Max Planck Institute for Intelligent Systems, Tübingen (Germany). We acknowledge CSC – IT Center for Science, Finland, for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through CSC. Yi Zhao, Juho Kannala, and Joni Pajarinen acknowledge funding by the Research Council of Finland (345521 353138, 327911). Yi Zhao thanks Yuxin Hou for inspiring this project and Wenyan Yang for the discussion of Optimal Transport.

REFERENCES

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving Rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[2] Allegro. https://www.wonikrobotics.com/research-robot-hand.

[3] Arthur Allshire, Mayank MittaI, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg. Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11802–11809. IEEE, 2022.

[4] Yunfei Bai and C Karen Liu. Dexterous manipulation using both palm and fingers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1560–1565. IEEE, 2014.

[5] Antonio Bicchi and Raffaele Sorrentino. Dexterous manipulation through rolling. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 1, pages 452–457. IEEE, 1995.

[6] Dieter Büchler, Simon Guist, Roberto Calandra, Vincent Berenz, Bernhard Schölkopf, and Jan Peters. Learning to play table tennis from scratch using muscular robots. *IEEE Transactions on Robotics*, 38(6):3850–3860, 2022.

[7] Ruben Castro Ornelas. Robotic finger hardware and controls design for dynamic piano playing, 2022.

[8] Nikhil Chavan-Dafle and Alberto Rodriguez. Sampling-based planning of in-hand manipulation with external pushes. In *Robotics Research: The 18th International Symposium ISRR*, pages 523–539. Springer, 2020.

[9] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022.

[10] Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

[11] Yuanpei Chen, Yiran Geng, Fangwei Zhong, Jiaming Ji, Jiechuang Jiang, Zongqing Lu, Hao Dong, and Yaodong Yang. Bi-DexHands: Towards human-level bimanual dexterous manipulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[12] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[13] David F Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696, 2016.

[14] Nikhil Chavan Dafle, Alberto Rodriguez, Robert Paolini, Bowei Tang, Siddhartha S Srinivasa, Michael Erdmann, Matthew T Mason, Ivan Lundberg, Harald Staab, and Thomas Fuhlbrigge. Extrinsic dexterity: In-hand manipulation with external forces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1578–1585. IEEE, 2014.

[15] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[16] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[17] Li Han and Jeffrey C Trinkle. Dextrous manipulation by rolling and finger gaiting. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 730–735. IEEE, 1998.

[18] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout Q-functions for doubly efficient reinforcement learning. *arXiv preprint arXiv:2110.02034*, 2021.

[19] JAE Hughes, P Maiolino, and Fumiya Iida. An anthropomorphic soft skeleton hand exploiting conditional models for piano playing. *Science Robotics*, 3(25):eaau3098, 2018.

[20] Rae Jeong, Jost Tobias Springenberg, Jackie Kay, Daniel Zheng, Yuxiang Zhou, Alexandre Galashov, Nicolas Heess, and Francesco Nori. Learning dexterous manipulation from suboptimal experts. *arXiv preprint*

arXiv:2010.08587, 2020.

[21] Ichiro Kato, Sadamu Ohteru, Katsuhiko Shirai, Toshiaki Matsushima, Seinosuke Narita, Shigeki Sugano, Tetsunori Kobayashi, and Eizo Fujisawa. The robot musician 'wabot-2'(waseda robot-2). *Robotics*, 3(2):143–155, 1987.

[22] Qiuqiang Kong, Bochen Li, Jitong Chen, and Yuxuan Wang. GiantMIDI-Piano: A large-scale midi dataset for classical piano music. *arXiv preprint arXiv:2010.07061*, 2020.

[23] Vikash Kumar, Yuval Tassa, Tom Erez, and Emanuel Todorov. Real-time behaviour synthesis for dynamic hand-manipulation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6808–6815. IEEE, 2014.

[24] Vikash Kumar, Abhishek Gupta, Emanuel Todorov, and Sergey Levine. Learning dexterous manipulation policies from experience and imitation. *arXiv preprint arXiv:1611.05095*, 2016.

[25] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47): eabc5986, 2020.

[26] Miao Li, Yasemin Bekiroglu, Danica Kragic, and Aude Billard. Learning of grasp adaptation through experience and tactile sensing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3339–3346. Ieee, 2014.

[27] Yen-Fang Li and Li-Lan Chuang. Controller design for music playing robot—applied to the anthropomorphic piano robot. In *2013 IEEE 10th International Conference on Power Electronics and Drive Systems (PEDS)*, pages 968–973. IEEE, 2013.

[28] Jen-Chang Lin, Hsing-Hsin Huang, Yen-Fang Li, Jen-Chao Tai, and Li-Wei Liu. Electronic piano playing robot. In *2010 International Symposium on Computer, Communication, Control and Automation (3CA)*, volume 2, pages 353–356. IEEE, 2010.

[29] Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. Deep differentiable grasp planner for high-dof grippers. *arXiv preprint arXiv:2002.01530*, 2020.

[30] Yumeng Liu, Yaxun Yang, Youzhuo Wang, Xiaofei Wu, Jiamin Wang, Yichen Yao, Sören Schwertfeger, Sibei Yang, Wenping Wang, Jingyi Yu, et al. RealDex: Towards human-like grasping for robotic dexterous hand. *arXiv preprint arXiv:2402.13853*, 2024.

[31] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1678–1690. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

[32] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 137–144, 2012.

[33] Amit Moryossef, Yanai Elazar, and Yoav Goldberg. At your fingertips: Extracting piano fingering instructions from videos. *arXiv preprint arXiv:2303.03745*, 2023.

[34] Luis Felipe Casas Murrilo, Ninad Khargonkar, Balakrishnan Prabhakaran, and Yu Xiang. MultiGripperGrasp: A dataset for robotic grasping from parallel jaw grippers to dexterous hands. *arXiv preprint arXiv:2403.09841*, 2024.

[35] Eita Nakamura, Yasuyuki Saito, and Kazuyoshi Yoshii. Statistical learning and estimation of piano fingering. *Information Sciences*, 517:68–85, 2020.

[36] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafał Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[37] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.

[38] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[39] Kai Ploeger, Michael Lutter, and Jan Peters. High acceleration reinforcement learning for real-world juggling with binary rewards. In *Conference on Robot Learning*, pages 642–653. PMLR, 2021.

[40] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[41] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. DexPoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *Conference on Robot Learning*, pages 594–605. PMLR, 2023.

[42] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation. in 2021 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7865–7871.

[43] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[44] Pedro Ramoneda, Marius Miron, and Xavier Serra. Piano fingering with reinforcement learning. *arXiv preprint arXiv:2111.08009*, 2021.

[45] Pedro Ramoneda, Dasaem Jeong, Eita Nakamura, Xavier Serra, and Marius Miron. Automatic piano fingering from partially annotated scores using autoregressive neural

networks. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 6502–6510, 2022.

[46] David A Randolph, Barbara Di Eugenio, and Justin Badgerow. Modeling piano fingering decisions with conditional random fields. 2023.

[47] Daniela Rus. In-hand dexterous manipulation of piecewise-smooth 3-d objects. *The International Journal of Robotics Research*, 18(4):355–381, 1999.

[48] Benjamin Scholz. Playing piano with a shadow dexterous hand, 2019.

[49] ShadowRobot. ShadowRobot Dexterous Hand. https://www.shadowrobot.com/products/dexterous-hand/, 2005.

[50] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023.

[51] Wenjing Shi, Yihui Li, Yisheng Guan, Xiaohan Chen, Shengtian Yang, and Senyu Mo. Optimized fingering planning for automatic piano playing using dual-arm robot system. In *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 933–938. IEEE, 2022.

[52] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[53] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[54] Anthony Topper, Thomas Maloney, Scott Barton, and Xiangnan Kong. Piano-playing robotic arm. *Worcester MA*, pages 01609–2280, 2019.

[55] Cédric Villani et al. *Optimal transport: Old and new*, volume 338. Springer, 2009.

[56] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3): 261–272, 2020.

[57] Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. DexGraspNet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11359–11366. IEEE, 2023.

[58] Huazhe Xu, Yuping Luo, Shaoxiong Wang, Trevor Darrell, and Roberto Calandra. Towards learning to play piano with dexterous hands and touch. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10410–10416. IEEE, 2022.

[59] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. UniDexGrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4737–4746, 2023.

[60] Zeshi Yang, Kangkang Yin, and Libin Liu. Learning to use chopsticks in diverse gripping styles. 2022.

[61] Kevin Zakka, Philipp Wu, Laura Smith, Nimrod Gileadi, Taylor Howell, Xue Bin Peng, Sumeet Singh, Yuval Tassa, Pete Florence, Andy Zeng, et al. RoboPianist: Dexterous piano playing with deep reinforcement learning. In *7th Annual Conference on Robot Learning*, 2023.

[62] Ada Zhang, Mark Malhotra, and Yoky Matsuoka. Musical piano performance by the ACT hand. In *2011 IEEE international conference on robotics and automation*, pages 3536–3541. IEEE, 2011.

[63] 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.

[64] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.