Playing Piano with the Shadow Dexterous Hand

Benjamin Scholz

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

09. July 2019

Outline

Outline (cont.)

Motivation

- Shadow Hand good fit for instruments
- challenging even for humans
- piano is versatile
- using MIDI feedback
- what is possible?

Figure: PR2 in front of the keyboard.

Related Work: Robots Designed to Play Piano

- fingers positioned directly over keys [1]
- movement restricted on one axis (rail/motor stage) [2] [3]

Figure: piano playing robot hands restricted on a motor stage [2]

- passive hand (no motors) [4]

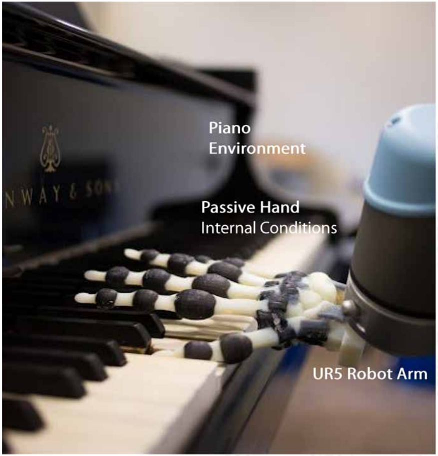

Figure: the passive hand playing piano [4]

Related Work: Playing Piano in Simulation

- record human motions, generate motions in simulation [5]

Figure: markers used to record human motions [5]

Figure: generated motions in simulation [5]

Related Work: Comparison to Human Performance

- compare human performance with robotic hand [6]

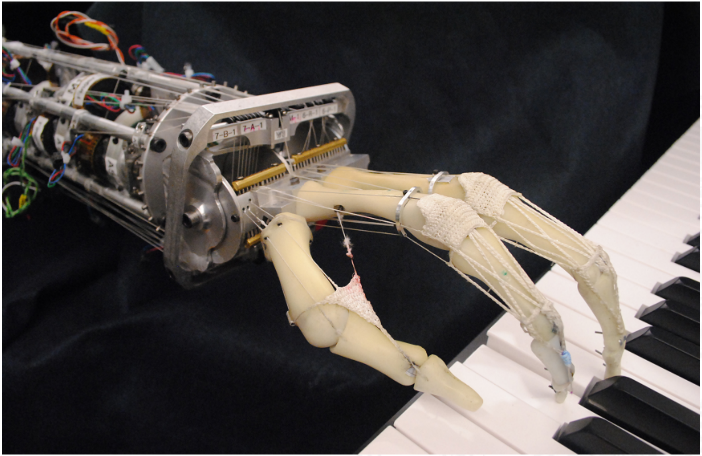

Figure: the ACT hand hitting a key [6]

Basics: MIDI

- **M**usical **I**nstrument **D**igital **I**nterface
- possibility to connect to PC
- trigger sound in software synthesizer
- relevant data:
  - key on/key off
  - velocity

Implementation: Keyboard Model & Localization

- keyboard: Roland PC-200 MK-II
- urdf model
- frame for every key

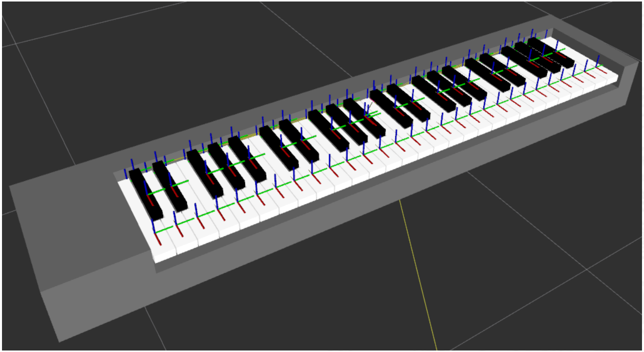

Figure: Model of the keyboard.

Implementation: Keyboard Model & Localization

- visual methods can be inaccurate due to calibration errors
- better solution: physical method
- mannequin mode on PR2 (`http://wiki.ros.org/pr2_mannequin_mode`)
- lead finger to push down key and record data:
  - which key: MIDI data
  - finger pose: tf (`http://wiki.ros.org/tf`)

Figure: Video of the localization process.

- known:
  - position of two keys
  - distance from keys to origin in model
- trigonometry to calculate position of piano's origin
- difference of key positions to compute orientation

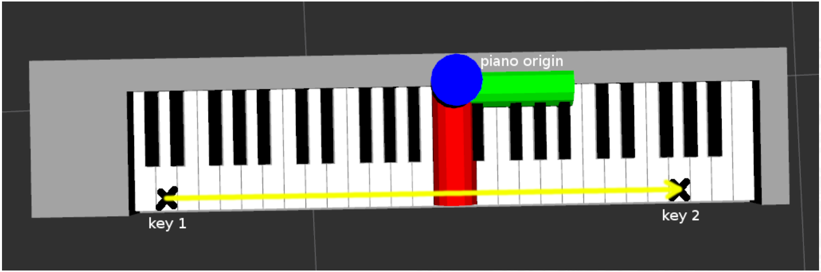

Figure: Using the difference of the keys position to determine the orientation.

Implementation: First Prototype

- use only index finger
- bio_ik [7] for inverse kinematics:
  - pose goal above key
  - goal in key, variable range for position along key
- plan and execute trajectories sequentially

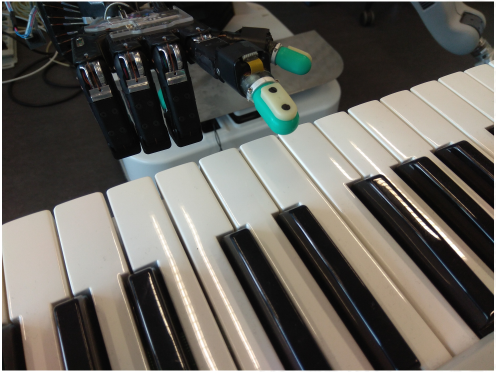

Figure: Playing piano with one finger.

Implementation: First Prototype

- planning between presses takes time
- solution: plan all trajectories beforehand, connect them
- arm movements between keys can be slow
- solution: adjust joint limits, use different planner

Implementation: Fast/Predictable Motions

- RRT-Connect -> randomization [8]
- arm movements not complex
- use point to point joint space motion planning instead (https://github.com/PilzDE/pilz_industrial_motion)

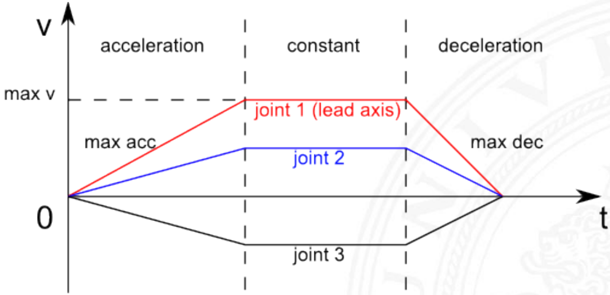

Figure: point to point motion planning. Retrieved from https://github.com/PilzDE/pilz_industrial_motion/blob/melodic-devel/pilz_trajectory_generation/doc/figure/ptp.png

Implementation: Fast/Predictable Motions

- blend 2 trajectories, so movement does not stop
- blending trajectory for connection:
  - $\mathrm{x}(\mathrm{s}) = \mathrm{x}_1(\mathrm{s}) + \alpha(\mathrm{s})(\mathrm{x}_2(\mathrm{s}) - \mathrm{x}_1(\mathrm{s}))$
- $\mathrm{x}_1, \mathrm{x}_2$: trajectories
- $\mathrm{s} \in [0, 1]$: normalized time parameter
- $\alpha(\mathrm{s}) \in [0, 1]$ for $\mathrm{s} \in [0, 1]$:
  polynomial function that ensures smooth increase, boundary conditions

Implementation: Fast/Predictable Motions

- blending radius

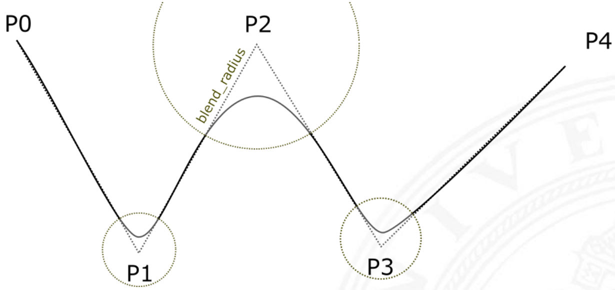

Figure: Blending radius to connect trajectories. Retrieved from `https://github.com/PilzDE/pilz_industrial_motion/blob/melodic-devel/pilz_trajectory_generation/doc/figure/blend_radius.png`

Implementation: Fast/Predictable Motions

- trajectory for pressing key is more fluent
- drawback: no control over timing
- solution: scale trajectory segments to fit note timing

Implementation: timing

- musical notes determine timing
- length between hitting two keys

Figure: Whole note to sixteenth note. Retrieved from `https://wsfcs.learning.powerschool.com/8142864827/5thgrade/cms_page/view/24925885`

Implementation: timing

- segment: lifting the finger, moving to next key and pressing it
- determine length of whole note:
  - largest value for $whole\_note = \dfrac{segment\_duration}{note\_divisor}$
- scale trajectory time parameterization to fit note length
- scale velocity and acceleration

Implementation: Pressing Keys

- notes can be hit on roughly the right time
- drawback: no control how keys are pressed
- solution: training data to learn pressing keys

Implementation: Pressing Keys

- MIDI velocity depends on how fast key is hit
- playing loud vs. quiet

Implementation: Pressing Keys

- collecting training data
- situation at start:
  - finger over key
  - joint 2 and joint 3 at 0 (extended finger)
- to move finger:
  - use trajectory controller with single target
  - varying: joint 2, joint 3, time_from_start
- recorded MIDI data: duration to signal, velocity

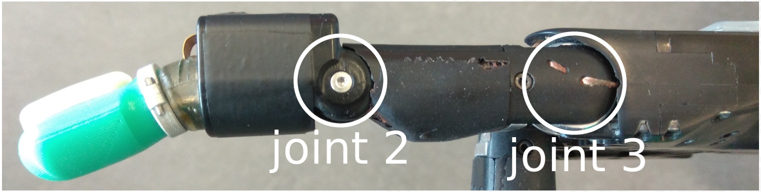

Figure: Joints used to press the keys.

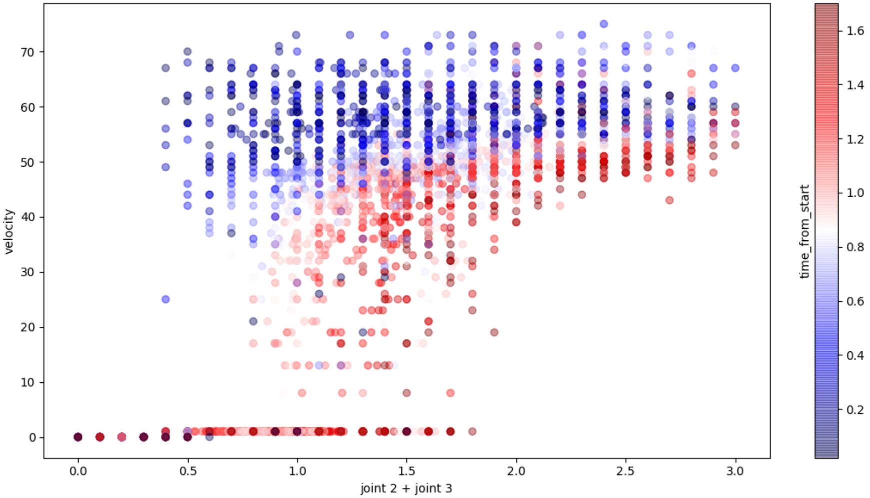

Figure: joint positions and *time_from_start* to velocity

- input: *MIDI velocity*
- output: *joint* 2, *joint* 3, *time_from_start*
- inverse problem/multivalued function

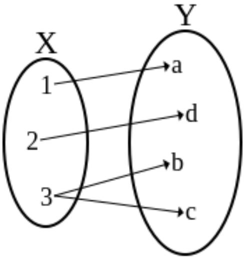

Figure: Example of multivalued function. Retrieved from `https://en.wikipedia.org/wiki/Multivalued_function#/media/File:Multivalued_function.svg`

Implementation: Pressing Keys

- conventional regression not able to represent function
- learns average of target value

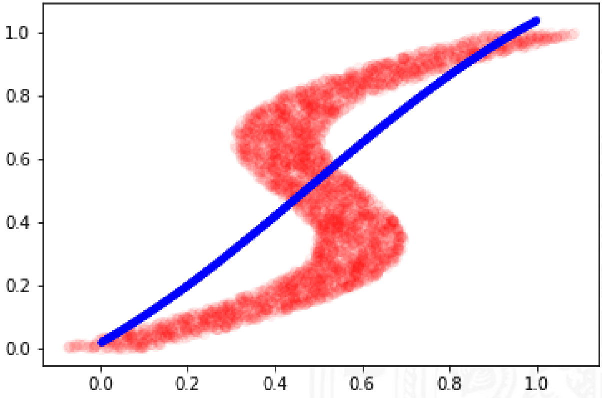

Figure: Attempt to use neural network regression to solve an inverse problem. Retrieved from `https://www.katnoria.com/mdn/`

Implementation: Pressing Keys

- solution: mixture density networks (MDN) [9]
- use neural network to learn parameters of Gaussian mixture model
- learn the underlying probability distribution
- parameters:
  - number Gaussians
  - number of hidden nodes
- tensorflow [10] with keras-mdn-layer (`https://github.com/cpmpercussion/keras-mdn-layer`)

Implementation: Pressing Keys

- input:
  - velocity
- output for each Gaussian:
  - 3 means (π_i)
  - 3 variances (μ_i)
  - 1 weight (σ_i)
- output from sampling from Gaussian mixture model:
  - *joint* 2
  - *joint* 3
  - *time_from_start*

Implementation: Pressing Keys

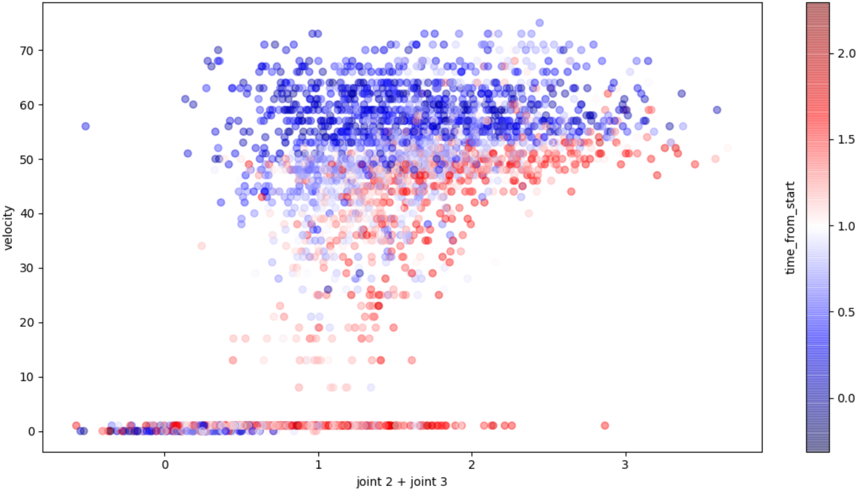

Figure: predicted joint positions and *time_from_start* from velocity

Implementation: Pressing Keys

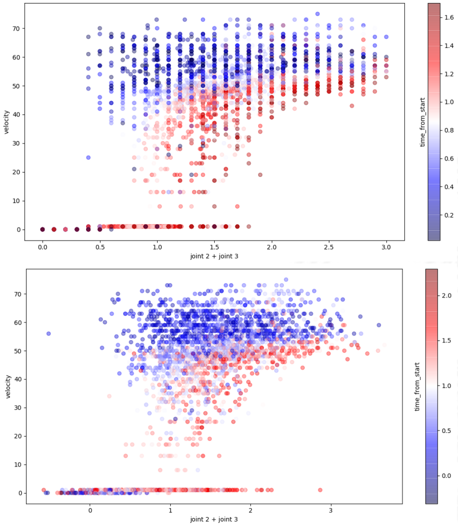

Figure: Comparison of recorded data (top) and predicted values (bottom).

Implementation: Pressing Keys

- problem: variety in velocities messes with timing
- hitting keys takes differently long
- solution: learn duration to signal

Implementation: Pressing Keys

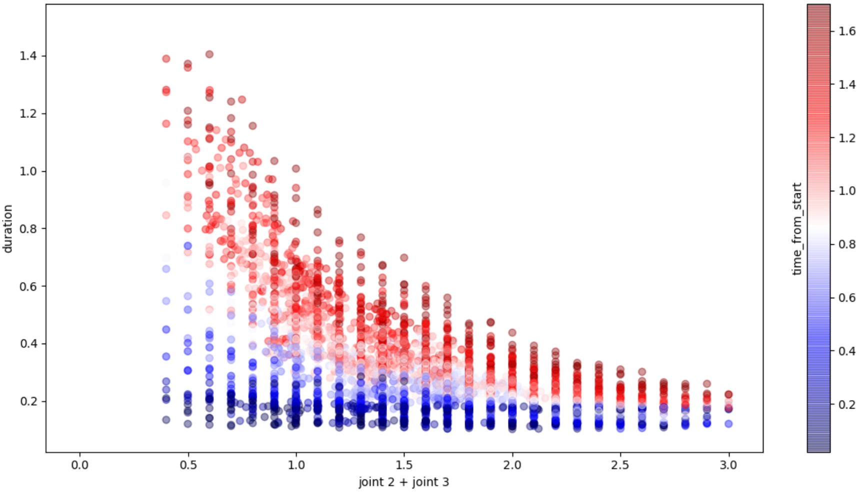

Figure: joint positions and *time_from_start* to duration

Implementation: Pressing Keys

- predict duration from sending command till hitting the key
- neural network regression
- input:
  - joint 2
  - joint 3
  - time_from_start
- output:
  - duration until signal
- using TensorFlow [10] with keras
  https://github.com/keras-team/keras

Implementation: Pressing Keys

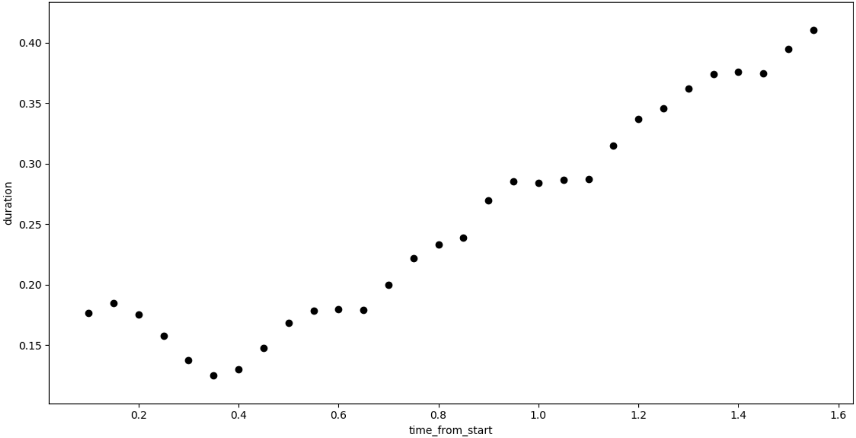

Figure: *time_from_start* to duration until signal with static joint positions

Implementation: Pressing Keys

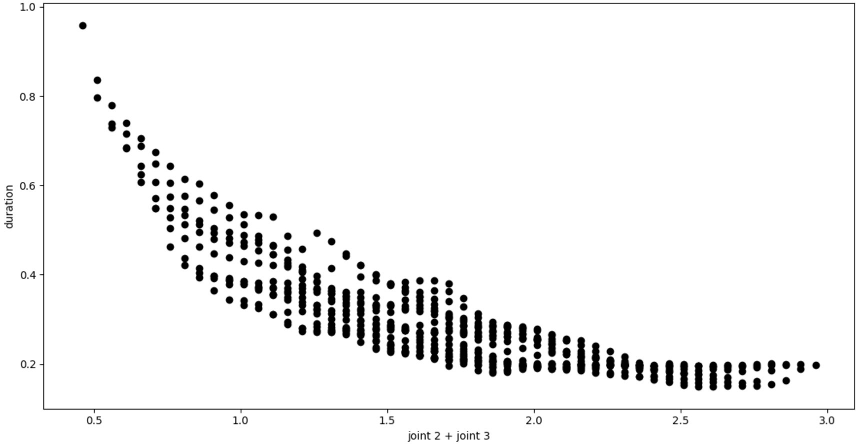

Figure: joint $2 +$ joint 3 to duration until signal with static joint positions

Implementation: Thumb

- thumb has different joints
- same method to learn, just 1 joint used
- harder to reach keys

Implementation: Thumb

Figure: Video of thumb pressing keys.

Implementation: Thumb

- alternative hand pose with better reach
- also useful to play white and black keys

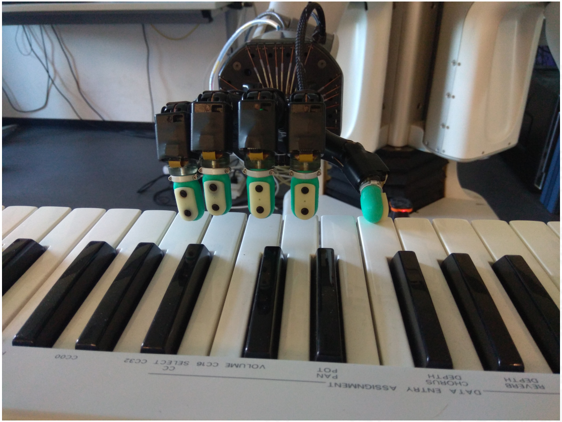

Figure: Alternative pose for hand.

Implementation: Thumb

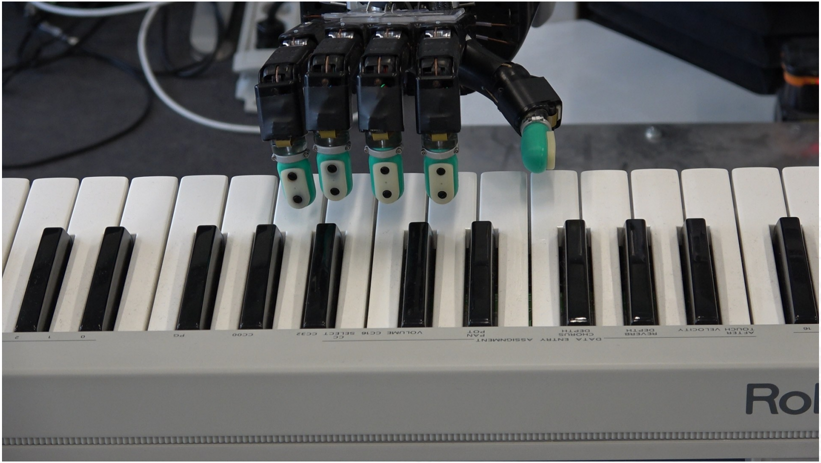

Figure: Video of thumb pressing keys with alternative hand pose.

Implementation: Pipeline

- putting everything together
- interface to press keys:
  - keys
  - fingers
  - velocities
  - press duration
  - note timing (whole note, 1/2 note etc.)

Implementation: Pipeline

- keys and fingers:
  - bio_ik to get arm pose with fingers over keys
  - create trajectory to move fingers above keys (no pressing yet)
- velocities:
  - predict presses for given velocities
  - add presses and lifts to previously created trajectories
- press duration:
  - add pause after pressing
- note timing:
  - use prediction of press durations and segment lenghts for scaling

Implementation: Staccato and Legato

- Staccato: short duration, detached from following note
- Legato: smoothly connected

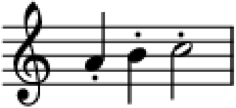

Figure: Staccato notes. (Retrieved from https://en.wikipedia.org/wiki/Staccato)

Figure: Legato notes. (Retrieved from https://en.wikipedia.org/wiki/Legato)

Implementation: Staccato and Legato

Figure: Video of explanation of staccato and legato. (Retrieved from `https://www.youtube.com/watch?v=N3XDpc2WBeI`)

Implementation: Staccato and Legato

- legato requires next finger to press, while previous finger is still lifting
- use bio_ik to keep both fingers over keys
  - specific goal type to be more flexible with orientation
- use press duration parameter to keep finger down longer

Implementation: Staccato and Legato

Figure: Video of playing staccato.

Implementation: Staccato and Legato

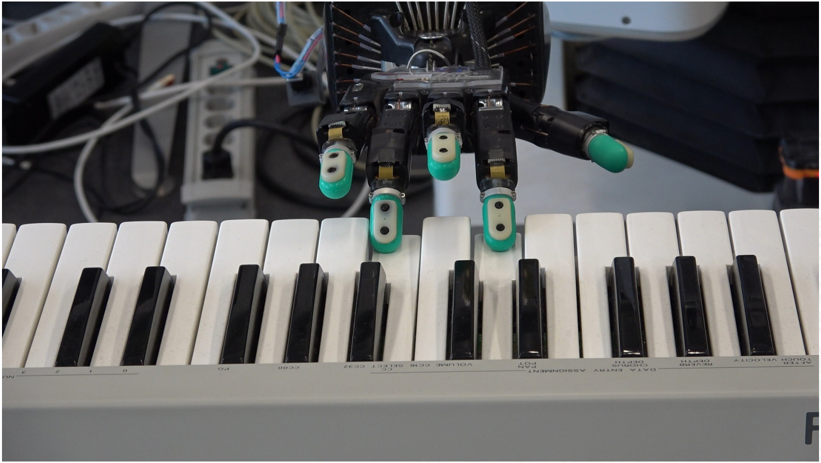

Figure: Video of playing legato.

Experiments: Velocity

- start with target velocity of 1
- increase target velocity by 5 up to 70
- human for comparison

Experiments: Velocity

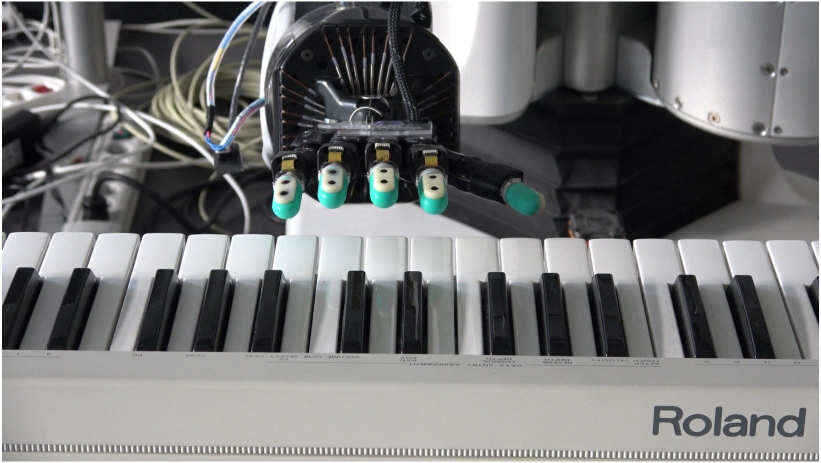

Figure: Video of the velocity experiment.

Experiments: Velocity

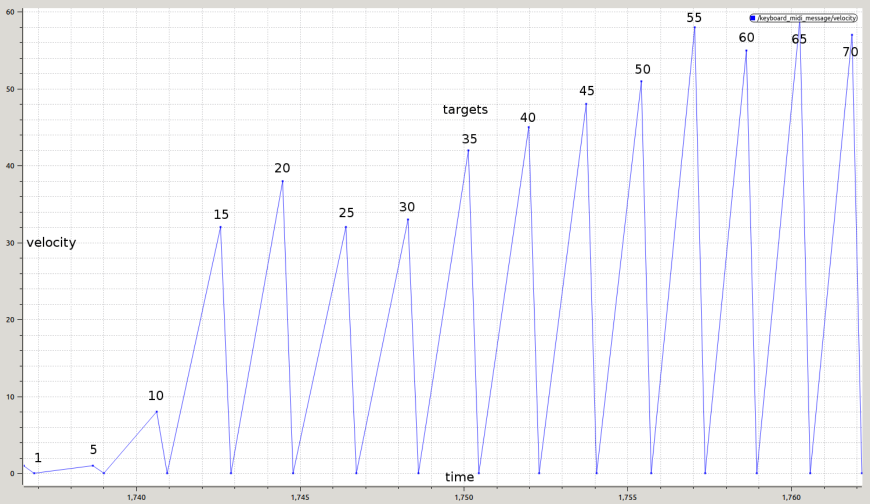

Figure: Result of the robot pressing keys.

Experiments: Velocity

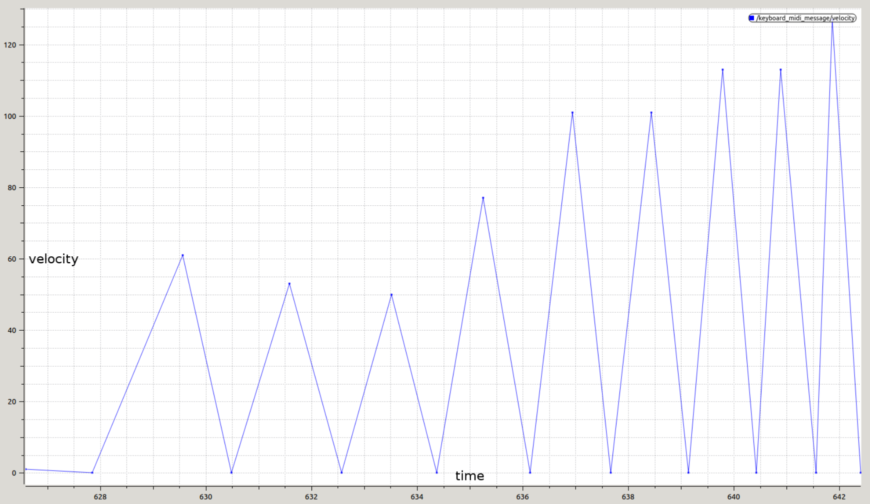

Figure: Result of the human pressing keys.

Experiments: Timing

- play keys with and without scaling
- compare results

Experiments: Timing

Figure: Video of hitting keys without trajectory scaling.

Experiments: Timing

Figure: Video of hitting keys with trajectory scaling.

Experiments: Timing

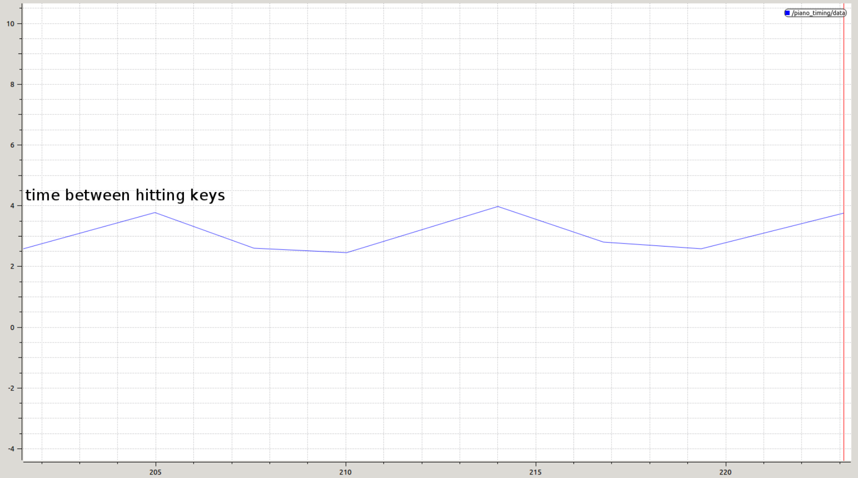

Figure: duration between hitting keys without trajectory scaling

Experiments: Timing

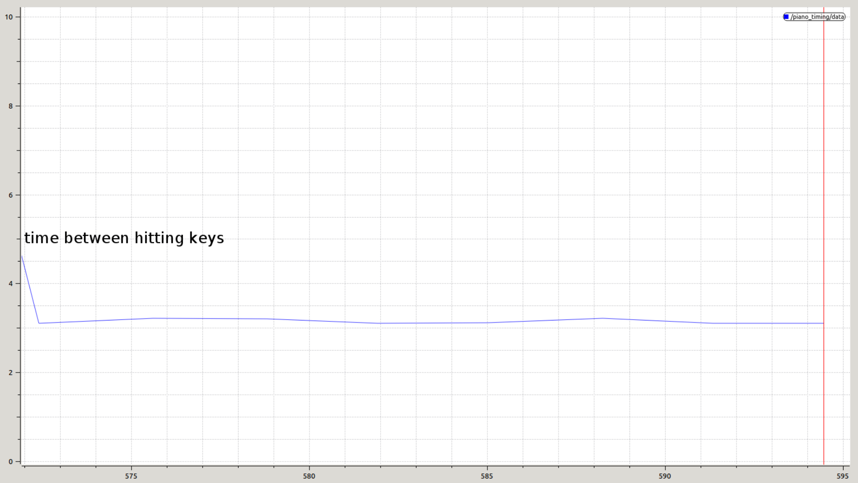

Figure: duration between hitting keys with trajectory scaling

Future Work

Figure: Video of failing to hit correct keys.

Future Work

- increase accuracy:
  - use BioTac sensors
  - use MIDI feedback
- integration
  - mix both finger poses
  - play black keys with alternative pose
  - legato with thumb
- play with increased speed

References

[1] S. Sugano and I. Kato, “Wabot-2: Autonomous robot with dexterous finger-arm–finger-arm coordination control in keyboard performance,” in *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 90–97, IEEE, 1987.

[2] J.-C. Lin, H.-H. Huang, Y.-F. Li, J.-C. Tai, and L.-W. Liu, “Electronic piano playing robot,” in *2010 International Symposium on Computer, Communication, Control and Automation (3CA)*, vol. 2, pp. 353–356, IEEE, 2010.

[3] D. Zhang, J. Lei, B. Li, D. Lau, and C. Cameron, “Design and analysis of a piano playing robot,” in *2009 International Conference on Information and Automation*, pp. 757–761, IEEE, 2009.

References (cont.)

[4] J. A. E. Hughes, P. Maiolino, and F. Iida, “An anthropomorphic soft skeleton hand exploiting conditional models for piano playing,” *Science Robotics*, vol. 3, no. 25, 2018.

[5] K. Yamamoto, E. Ueda, T. Suenaga, K. Takemura, J. Takamatsu, and T. Ogasawara, “Generating natural hand motion in playing a piano,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3513–3518, IEEE, 2010.

[6] A. Zhang, M. Malhotra, and Y. Matsuoka, “Musical piano performance by the act hand,” in *2011 IEEE International Conference on Robotics and Automation*, pp. 3536–3541, IEEE, 2011.

References (cont.)

[7] P. S. Ruppel, "Performance optimization and implementation of evolutionary inverse kinematics in ROS," Master's thesis, Universität Hamburg, Germany, 2017.

[8] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, pp. 995–1001 vol.2, April 2000.

[9] C. M. Bishop, "Mixture density networks." 1994.

References (cont.)

[10] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015.
Software available from tensorflow.org.