Reaching Benchmark Framework V.1.0 Documentation

Andre Lemme, Yaron Meirovitch, Mohammad Khansari

September 10, 2013

Contents

Chapter 1

Benchmark concepts

1.1 Introduction

Manual programming of robot motions often requires a large amount of engineering knowledge about both the task and the robot and it can become particularly non-intuitive when dealing with high Degrees of Freedom (DoF) humanoids. The advent of a new generation of humanoid robots that need to perform a wide variety of tasks in human daily lives stresses further more the importance of techniques to autonomously adapt to various situations and to be robust to various sources of perturbations/uncertainties. It is furthermore essential for social acceptance of humanoids to provide motion patterns that are more similar to human movements. In response to these concerns, many approaches have been introduced within the passed decade following different levels of modeling, from kinematic of the motion, biomechanic of the limb, planning and execution level, to neural modeling of cortical process.

This benchmark framework was developed to compare the state-of-the-art learning algorithms that cover model of the kinematic of human reaching motions. It is timely as an increasing variety of different approaches in generating human-like robot motions in the field calls for standardized and systematic comparisons. This benchmark provides the opportunity to understand and to compare methods through an open-source software framework that has been developed in the European project AMARSi (http://www.amarsi-project.eu/). This benchmark evaluates algorithms on a library of human motions based on various criteria such as "level of similarity to human motions", "accuracy in reaching the goal state", "adaptability to changes dynamic environments", "robustness to perturbations", etc.

The AMARSi Benchmark framework is a software package written in MATLAB that evaluates the performance of reaching motion generation methods against 11 different metrics. In this chapter we present the benchmarking framework and the methodology for systematically test and evaluate of the different models of motion generation.

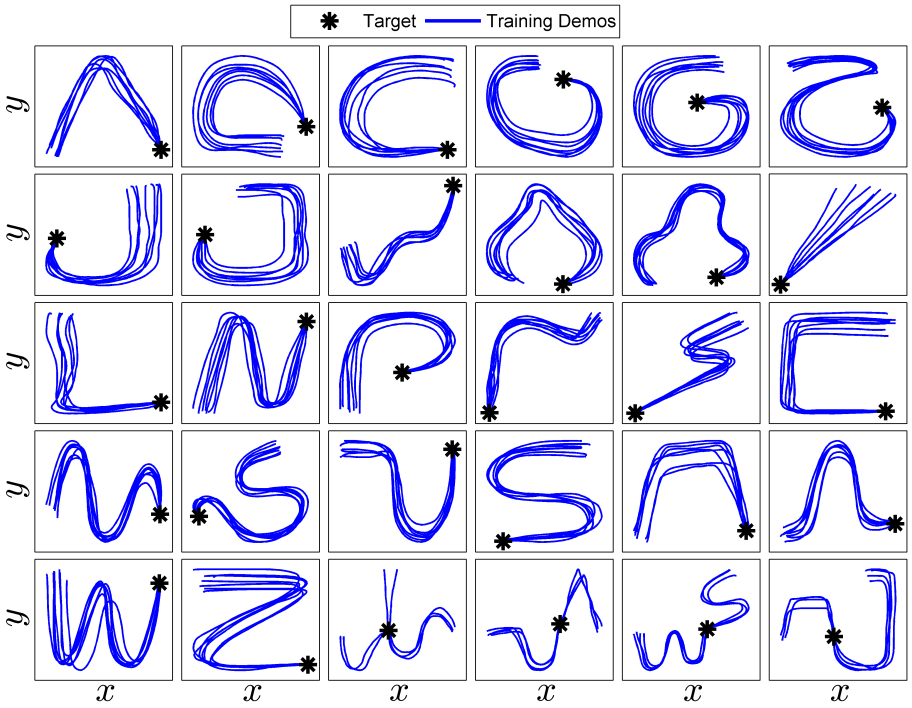

Figure 1.1: The library of LASA handwriting motions [13]. This library is composed of 30 two-dimensional point-to-point motions.

1.2 Dataset of handwriting motions

The demonstrations data for the handwriting motions were collected from pen input using a Tablet-PC. For each motion, the user was asked to draw 7 demonstrations of a desired pattern, by starting from different initial positions (but fairly close to each other) and ending to the same final point. These demonstrations may intersect each other. In total a library of 26 human handwriting motions were collected, in addition four motions were generated that more than one pattern is included (called Multi Models). Without loss of generality, the target (final) point is by definition set at $(0, 0)$ for all motions (shapes) in this library. This library was developed by LASA laboratory (`http://lasa.epfl.ch/`) and used in [14, 12, 13] as an evaluation benchmark to compare the performance of different regression techniques. All demonstrations of the different shapes are presented in Figure 1.1.

1.3 Benchmark Conditions

Four different benchmark conditions were designed, each condition test specific aspects of the trajectory generation methods. We describe each benchmark condition and their associated parameters and metrics that are used to quantify the performance of each method. In each benchmark, 150 samples for every parameter are drawn from specific probability distributions which will be given

in the following.

1.3.1 Discrete Push

In this benchmark condition, discrete perturbations are applied during motion generation. It emulates a hit against the end effector at a discrete time. A hit is a displacement of the end in one time step and evaluates the robustness of the movement generation method. These perturbations appear with varying timing, direction and amplitude. The target point remains fixed.

For each shape, we perform systematic tests for each combination of values of the parameters below:

Normalized start time: The normalized time at which the trajectory is perturbed[1]. The samples for start time are drawn from the following probability distribution function:

$$\mathcal{P}(x) = \begin{cases} 1 & \text{for } 0 \leq x \leq 1 \\ 0 & \text{for } x < 0 \text{ or } x > 1 \end{cases} \tag{1.1}$$

Amplitude: The amplitude of the perturbation (in millimeters). Let us define $l = 50mm$, corresponding to the length span of motion along both x and y axes. Then, we draw samples for amplitude from a normal PDF:

$$\mathcal{P}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1.2}$$

where $\mu = 0.1l$ and $\sigma = 0.05l$.

Vector: The direction of the perturbation (no unit). This vector is normalized and multiplied by the amplitude to get the actual perturbation. Vector is a two-dimensional variable and each of the two components are drawn from a uniform PDF.

$$\mathcal{P}(x) = \begin{cases} 0.5 & \text{for } -1 \leq x \leq 1 \\ 0 & \text{for } x < -1 \text{ or } x > 1 \end{cases} \tag{1.3}$$

1.3.2 Generalization

The generalization benchmark condition is a special variant of the discrete push benchmark condition. The difference is that the "start time" parameter is set to zero and remains fixed. The perturbation will now occurs at the beginning of the movement which results in a displacement of the starting position and shows the ability to start the movement from different initial conditions. For generalization benchmark condition, the parameters value are set as follows:

Amplitude: The amplitude of the perturbation (in millimeters). The samples for amplitude are drawn from a uniform PDF (see Eq. 1.2) with mean $\mu = 0.2l$ and standard deviation $\sigma = 0.1l$.

[1]Note that the normalized start time will be multiplied by the average duration of each motion to determine the actual start time of perturbations.

Vector: The direction of the perturbation (no unit), which is drawn from the pdf given in Eq. 1.3.

1.3.3 Continuous Push

In this benchmark condition, the movement generation is continuously perturbed during a specific amount of time. This emulates e.g. a teaching scenario in which a human tutor is correcting the movement for a time duration. As previously, perturbations appear at varying times, direction and amplitude. The target point remains fixed.

For each shape, systematic tests are performed for each combination of values of the parameters below:

Start time and vector are chosen as described in the *Discrete Push Benchmark* condition.

Amplitude: The amplitude of the perturbation (in millimeters/second). The samples for amplitude are drawn from a uniform PDF (see Eq. 1.2 with $\bar{v} = 20.7315$, mean $\mu = 0.5\bar{v}$ and standard deviation $\sigma = 0.25\bar{v}$. The parameter $\bar{v}$ corresponds to the mean speed of all demonstrations across all the 30 motions in the library.

Duration: The duration of the perturbation (in seconds) is drawn from the following PDF:

$$\mathcal{P}(x) = \begin{cases} 5 & \text{for } 0.1 \leq x \leq 0.3 \\ 0 & \text{for } x < 0.1 \text{ or } x > 0.3 \end{cases} \tag{1.4}$$

1.3.4 Moving Targets

This benchmark condition quantifies the ability of the trajectory generators to track and reach moving targets. The target movement starts at different times and lasts over a certain duration. Also the amplitude and direction of the target movement can be changed.

The parameters are the same as in the *Continuous Perturbation Benchmark* only applied to the target point. Therefore, the probability distributions are the same as well.

1.4 Evaluation

In the benchmark evaluation, principles from motor control research are implemented in order to assess the behavioral plausibility of movements and, specifically, to evaluate the maintenance and distortion/violations of these human movement regularities in the movement reproductions.

1.4.1 Trajectory evaluation based on motor control research

Observations have been showing that various human movements are stereotypical [16, 3, 1, 6, 10, 9, 17, 4, 7]. One key observation is that point to point movements tend to be straight and their speed profiles bell-shaped. Such regularities appear in reaching movements with invariance to the direction and the end-points of the articulated trajectories. A theoretical account for this invariance was suggested by the minimum jerk model [10, 8] which predicts that hand trajectories are optimally smooth (see eqn. 1.9). The predictions of this model were successfully tested also for tasks involving via points [8] and curved movements (the constrained minimum jerk model [21]).

Another regularity is that hand movements tend to slow down when the shape of the trajectory becomes curved. This tendency was quantified by the two-thirds power law [15], which predicts that the hand's speed is proportional to the curvature of the path of the movement raised to the power of minus one third (see eqn. 1.10). Numerous studies have analyzed the persistency of this rule, mainly in drawing movements [22] but also for other modalities such as eye pursuits [23].

One of the most prominent features of human movement is encoded in its kinematics. Hence, we paid special attention to measuring the compliance of the reproduced kinematics with the demonstrated one by quantifying how much of the variance in the demonstrated kinematics is explained by the reproduced kinematics. Therefore we calculate the coefficient of determination (R^2) for explaining the demonstrated kinematics based on the reproduced kinematics. The scores of this metric are calculated from the standardized trajectories according to the formula,

$$R^2 = 1 - \frac{\sum_i \left(v_{demo}(i) - v_{repro}(i)\right)^2}{\sum_i \left(v_{demo}(i) - \bar{v}_{demo}\right)}, \tag{1.5}$$

where v is the standardized speed profile, which has been frequently used for the modeling of human movement (e.g. [20, 18, 2]). The R^2 metric compares the predictions of the model against those of the simple model of constant speed movement, where a value of 1 indicates perfect agreement and a value smaller than 0 means that the constant speed model is better than the reproductions at explaining the data of the demonstrations. In the evaluation process the R^2 metric is not only used for the speed profiles, but also e.g. scoring the reproduced movement path.

All trajectories are temporally scaled by a parameter τ ($d\tau \propto dt$) in order to have standardized durations. This allows to inspect motion kinematics independently of the total movement duration.

The geometric and kinematic accuracies of the simulated reproductions were separately inspected. To obtain only the shape relevant geometrical information, called "path", all trajectories are parametrized according to their Euclidean arc length parameter s such that for a trajectory r, $\left|\frac{dr}{ds}\right| = 1$. The compliance of the simulated path with that of the demonstration is measured by their RMSE measure (root of the mean squared error, eqn. 1.7 below).

1.4.2 Standardized Scores and Ranking

In the benchmark, normalized scores are chosen that can be used for comparing both among benchmarked methods and across shapes in each method. Scores are standardized to reduce the sensitivity of the scoring system to noise and digitization. All scores whose optimal values may potentially be attained by a good reproduction are thresholded near that value. For example, all reaching positions that were smaller than 1 mm far from the target positions were counted as optimal. However doing so for the mean squared jerk is artificial and meaningless since an optimal attainable jerk thresholds is unknown and only parabolas attain the optimal value of zero mean squared jerk [19]. When applied, the thresholds are given below for each measurement.

1.4.3 Measures on the Kinematic level

The following measures are specialized to evaluate the performance on the velocity or the speed profiles.

Velocity Accuracy

The reproduced and demonstrated velocity vector profiles are compared using the RMSE measure above (see 1.4.4) with the standardized trajectory parameter τ (see sec. 1.4.5).
Scope: *global*.
Variable: *kinematic*.

Speed Accuracy

the speed profiles of the demonstrated and reproduced trajectories are compared using the R^2 measure as in eqn. 1.5. As in sec. 1.4.5, calculations are carried out based on the standardized trajectory parameter τ.
Scope: *global*.
Variable: *kinematic*.
Optimality threshold: $R^2_{speed-accuracy} \geq 0.8$ of explained variance.

Target Velocity Error

This is a task relevant measure, because for reaching task stopping at the target is necessary for save human robot interaction. Therefor we check if the last velocity sent by the model is close to zero taking the L2-norm of this vector.
Scope: *local*.
Variable: *kinematic*.

Movement Duration

The trajectories are not reparameterized and the duration of the reproduction and the corresponding demonstration is compared. Let us denote the demonstration and its reproduction final time by t_d^f and t_r^f. Then the movement duration error is computed according to:

$$\varepsilon_{movement-duration}(t_d^f, t_r^f) = |1.0 - \frac{t_r^f}{t_d^f}| \tag{1.6}$$

Scope: *global*.
Variable: *kinematic*.
Optimality threshold: $\varepsilon_{movement-duration}(t_d^f, t_r^f) \leq 0.1$.

1.4.4 Measures on the Geometric level

The reproduced and demonstrated trajectories are first reparamterized to have only geometric information regarding the shape of the movement and will be called "path" in the following. To obtain their paths, all trajectories are parametrized according to their Euclidean arc length parameter s such that for a trajectory r, $\left|\frac{dr}{ds}\right| = 1$.

Path Accuracy

The reproduced and demonstrated paths, $r_{re}(s)$. $r_{demo}(s)$, are compared by measuring their RMSE,

$$RMSE_{path-accuracy}(r_{de}, r_{re}) = \sqrt{\frac{1}{T}\int_0^T |r_{de}(s) - r_{re}(s)|^2 \, ds} \quad (1.7)$$

Scope: *global*.
Variable: *geometrical*.

Target Position Error

Primary scope: *local*.
Variable: *geometrical*
Optimality threshold: $Err_{reachingtargeterror} \geq 1$ mm.

1.4.5 Measures using Geometric and Kinematic

In the evaluation we only have one measurement that evaluates the geometrical and also the kinematic information of the reproduction versus the demonstration which are normalized only in movement duration.

Trajectory Accuracy

The trajectories normalized in movement duration, $r_{re}(\tau)$, $r_{demo}(\tau)$, are compared using the following R^2 measure,

$$R^2_{trajectory-accuracy}(r_{de}, r_{re}) = 1 - \frac{\int_0^{\mathrm{T}} |r_{de}(\tau) - r_{re}(\tau)|^2 d\tau}{\int_0^{\mathrm{T}} |r_{de}(\tau) - \mathbb{E}[r_{de}]|^2 d\tau} \quad (1.8)$$

Scope: *global*.
Variable: *geometrical* and *kinematic*.
Optimality threshold: $R^2_{trajectory-accuracy} \geq 0.95$.

Minimum Jerk Trajectories

The minimum jerk model predicts that human trajectories minimize the following functional,

$$I(r) = \int_0^T \left| \frac{d^3}{dt^3} r \right|^2 dt \quad (1.9)$$

subject to boundary conditions

where $r(t)$ is any end-effector's trajectory. We therefore used the root of the Mean Squared Derivative (MSD) measure of a trajectory $r(t)$,

$$RMSD_n(r) = \sqrt{\frac{1}{T} \int_0^T \left| \frac{d^n}{dt^n} r \right|^2 dt},$$

where for the mean squared jerk $n = 3$.
Scope: *global*.
Variable: *spatiotemporal*.

The 2/3 power law

The 2/3 power law predicts the speed profile of a trajectory, $\frac{ds}{dt}$, based on its curvature $\kappa(t)$,

$$\frac{ds}{dt} = \alpha \kappa(t)^\beta, \quad (1.10)$$

where α and β are segment-wise constants. The parameter β is usually close to a value of $-\frac{1}{3}$, mostly for drawing movements, and α is a velocity gain factor (see [5] for the use of this law for complex 3D tasks).

The question of movement control and specifically segmentation is not addressed in this benchmark. However since the compliance with such power laws is evaluated only over segments of movements, we exhaustively evaluated various segment sizes in the reproductions against those of the demonstrations. To avoid explicit segmentation, sliding windows are used for each demonstration, evaluating the following linear prediction [22],

$$\log \frac{ds}{dt} = \log \alpha + \beta \log \kappa(t),$$

with the R^2 metric.
Scope: *segmental*.
Variable: *spatiotemporal*.
Optimality threshold: $\Delta R^2_{power-law} \geq 0\%$ (i.e. as good as demonstrations with respect to compliance with the power law).

For each demonstration and window size, the R^2 scores are averaged across all sliding windows. The score profile of a demonstration is thus,

$$S_{demo_i}(W_n) = \mathbb{E}[R^2],$$

calculated over all sliding windows of length W_n, where W_n is the duration of the n-samples windows. And then the worse (minimal) R^2 is calculated for each window size across all demonstrations,

$$S(W_n) = \min_i \{S_{demo_i}(W_n)\},$$

This score is a lower bound for the R^2 values across all demonstrations, indicating for what window sizes the power law is evident in all demonstrations. The set of scores $\{S(W_n)\}$ is referred to as the "groundtruth" compliance of the task's demonstrations with the power law.

The same procedure is carried out to obtain the score of each reproduction, $S_{repro_i}(W_n)$. The deviation of each reproduction from the groundtruth score is calculated for each window size and then averaged over the sizes for which the groundtruth competence is high ($R^2 > 0.5$). For each reproduction, the resulting score $F(\text{repro}_i)$ is relative,

$$F(\text{repro}_i) = \mathbb{E}\left[\frac{S_{repro_i}(W_n) - S(W_n)}{S(W_n)}\right],$$

calculated over all window durations W_n.

1.4.6 Properties of the implemented model

Besides the performance of the movement generation module to generate good trajectories, we are also interested of the resource management, which can give some insights to the complexity in the movement generation method.

Processing Time

Processing time provides an estimation of the computational complexity of the motion generator algorithm. It corresponds to the amount of time (in millisecond) for the algorithm to provides the next desired state based on the current state of the motion.

Model Size

For a motion generator algorithm, matlab model '.mat' file size provides a rough estimation of the number of parameters required to model a motion. For each motion in the library, participant are required to provide a (trained) model as a MATLAB binary file (i.e. '.mat' format). We will use the size of this file in bytes as an indication of the required number of parameters without going into mathematical details. Thus, it is essential to only save essential parameters in the '.mat' file of each model.

Chapter 2

Technical manual

2.1 First steps in using the Benchmark framework

This MATLAB benchmark framework was developed to compare different methods for generating reaching movements and extract their specificities, strengths and weaknesses. It allows each user to configure different perturbations which can occur during a movement execution and prepare their models for the given task before a baseline parameter set is used to create comparable results.

To participate in the Benchmark, you do not need to provide us with your learning algorithm. You only need to send a piece of code (i.e. integration in a common interface class) that can be used at the execution level and also your (trained) model of the motions. Your code at the execution level should provide the next state given the current situation (i.e. current time, goal position, and current position and velocity). In case if it is necessary, you could create a content-obscured version of your code by using the MATLAB "pcode" function.

In the following the class interface is introduced and it is described how to prepare the functions which are needed for the specific movement generation algorithm. In section 2.3 the basic usage of the graphical user interface is described.

2.2 Interface class

In preparation of using the benchmark a movement generation algorithm needs to be wrapped in a common interface. The proposed interface is specified in a MATLAB class definition. This class is only for using the movement representation of the users model and does not provide functionality for learning from data. The idea is that the benchmark user has already MATLAB models which are trained with a dataset used in the benchmark. If that is the case only three steps are needed and the benchmark framework can be used immediately:

1. write a class definition, which inherits the interface class 'MovementGenerator' (example can be seen Code 2.1)

2. initialize the new class with the learned model

3. store the wrapped model in the 'YOUR_ MODEL _ FOLDER' (see sec. 2.4)

2.2.1 Properties

The default properties:

t this gives number of past time steps

dt used delta t. This is used for integrating the trajectory

dim dimensionality of the movement task (2D/3D)

learnedModel will hold the original movement representation

curr_position is used for feedback of the current position of the End Effector

curr_velocity is used for feedback of the current velocity of the End Effector

target_position is used for feedback of the current position of the target end point.

If additional properties are needed, one can add more in the inherited class. However the feedback will only be provided in position, velocity and current time step.

2.2.2 Functions

Only two functions need to be implemented by the user. The 'init' function and the 'step' function.

The "step" function is meant to produce velocity commands to the simulation, where xd is the velocity vector. We suggest that additional functions which are needed for the original learnedModel will be located into the lib folder of the corresponding participant.

2.3 GUI

The concept of this benchmark graphical user interface (GUI) is to configure a full set of parameters for the different benchmark conditions (described in Sec. 1.3) first and then press the "start" button to start the benchmark process. At the end of the evaluation the results are stored in a 'mat' file located in the corresponding result path which can also be specified in the GUI. In the following the basic functions and settings are described.

2.3.1 How to start the Benchmark

After starting MATLAB go to the Benchmark root path and condition 'startBenchmark'. This script will close all figures you may have open and also will clear the workspace. Additionally it includes all necessary folders in the program path of MATLAB. When the script successfully finished the GUI is visible on the screen as shown in Fig 2.1 without any plotted results. The appearance might varying a little if a different operating system is used e.g. mac, windows, or linux.

Code 2.1 Simple class definition example for a Movement Generator which can be used in the benchmark framework

```
classdef  DUMMY_mg < MovementGenerator
    properties
        %properties in superclass
    end
    methods
        function p = DUMMY_mg()  % constructor
            p@MovementGenerator();
        end

        % This function is wrapping the learned model
        function init(p,model,dim)
            p.init@MovementGenerator(dim);
            p.learnedModel = model;
        end

        % this function is called frequently in the benchmark
        % (see Simulation.m). The output of this function is
        % the desired velocity. The next state in the Simulation.m
        % is computing by x_next = xd*dt + x_current
        function xd = step(p)
            %Your code
        end
    end
end
```

2.3.2 General layout

The main window is divided into three main planes (see Fig. 2.1). The first in the top left allows to set general information like the path to the dataset and also to the trained models. It also has a drop down selecting object (see Fig. 2.1 green background) in which a benchmark condition can be selected. When a benchmark condition is selected the corresponding parameter settings are displayed in the second main plane on the top right (see Fig. 2.2). The third plane below the first two is used to display results of the evaluation for each iteration.

The results are structured as follows. In the first row two plots are displayed, which show first the movement trajectories from the demonstration (blue) the reproduction (red) and matched version (green) of the reproduction. The matched version is the result of using the Procrustes algorithm [11] which rotates and scales the reproduction such that a minimum error between the demonstration and reproduction exist. The second plot in that row shows the speed profile of the demonstrations(blue) and reproductions(red).

The next plots display the results per iteration. One iteration represent one set of parameters used for the benchmark conditions. The labels on the y-axis describe the measure displayed in the plot:

computationTime see Sec. 1.4.6 (Processing Time)

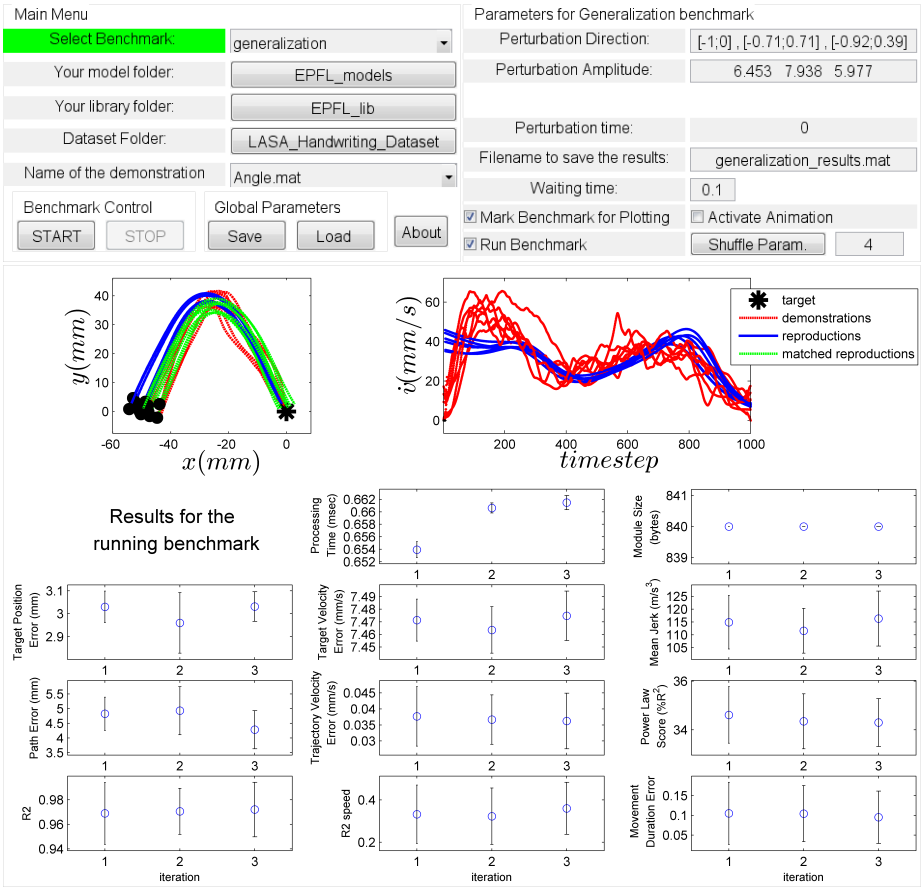

Figure 2.1: Benchmark framework GUI after starting the evaluation.

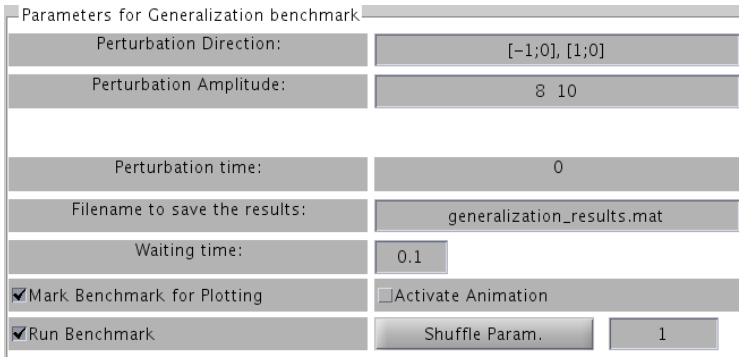

Figure 2.2: Close up view to the parameter settings of one Benchmark condition. For each parameter two samples are displayed.

modelSize see Sec. 1.4.6 (Model Size)

targetErrorPos see Sec. 1.4.4 (Target Position Error)

targetErrorVel see Sec. 1.4.5 (Target Velocity Error)

meanJerk see eq. 1.9 (Mean Jerk of the movement)

trajErrorPos see eq. 1.7 (Trajectory Position Error)

trajErrorVel see Sec. 1.4.5 (Velocity Accuracy)

PL_ R2 see eq. 1.10 (Power-Law)

R2 see eq. 1.5 applyed to the path.

R2_ speed see eq. 1.5

normalizedFinalTime see Sec. 1.4.5 (Movement Duration)

2.3.3 Setting parameters

All parameters described in Sec. 1.3 can be specified. For each parameter a discrete list of numbers are allowed following the MATLAB array convention i.e. in the discrete benchmark the parameter perturbation direction which is a vector can be specified like this: [1; 2], [1 0]... The value will be processed one after the other. Additionally each benchmark condition can be excluded from evaluation and marked for plotting.

Note: By checking the check box labeled "Run Benchmark" you will not start the benchmark process. All benchmark conditions with this check box checked will be processed after pressing the "Start" button in the main menu.

2.4 FAQ

Where should I save my wrapped movement generation files?

All modules should be saved in one folder located here:

```
BENCHMARK_ROOT filesep DATASET_PATH  filesep YOUR_MODEL_FOLDER
```

each module should correspond to one of the shapes in the dataset.
An example: one shape dataset from the "LASA_ Handwriting_ Dataset" is located here:

```
BENCHMARK_ROOT filesep LASA_Handwriting_Dataset ...
 filesep DataSet filesep 'Angle.mat'
```

The corresponding learned module (in the benchmark interface class structure) should be located here:

```
BENCHMARK_ROOT filesep LASA_Handwriting_Dataset ...
 filesep YOUR_MODEL_FOLDER filesep 'Angle.mat'
```

Each shape can be selected in the GUI in the main menu "Name of the demonstration" or all together by choosing 'all' in the section.

Where should I save my utility m-files that I need?

We suggest to locate such files into a library folder of you choice which can be located here:

```
BENCHMARK_ROOT filesep YOUR_LIB_FOLDER
```

Your adapted interface class can also be located here:

```
BENCHMARK_ROOT filesep 'Class'
```

Bibliography

[1] W. Abend, E. Bizzi, and P. Morasso. Human arm trajectory formation. *Exp Brain Res*, 105:331–348, 1982.

[2] D. Bennequin, R. Fuchs, A. Berthoz, and T. Flash. Movement timing and invariance arise from several geometries. *PLoS computational biology*, 5(7):e1000426, 2009.

[3] N. Bernstein. *The Co-ordination and Regulation of Movements*. Pergamon Press, Oxford, 1967.

[4] E. Bizzi, M.C. Tresch, P. Saltiel, and A. d Avella. New perspectives on spinal motor systems. *Nature Reviews Neuroscience*, 1(2):101–108, 2000.

[5] Dominik Endres, Yaron Meirovitch, Tamar Flash, and Martin A Giese. Segmenting sign language into motor primitives with bayesian binning. *Frontiers in computational neuroscience*, 7, 2013.

[6] T. Flash. Organizing principles underlying the formation of hand trajectories. *Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, MA*, 1983.

[7] Tamar Flash and Binyamin Hochner. Motor primitives in vertebrates and invertebrates. *Current Opinion in Neurobiology*, 15(6):660 – 666, 2005. Motor sytems / Neurobiology of behaviour.

[8] Tamar Flash and N. Hogan. The coordination of arm movements - an experimentally confirmed mathematical-model. *J Neurosci*, 5(7):1688–1703, 1985.

[9] C. M. Harris and D. M. Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394:780–784, 1998.

[10] N. Hogan. An organizing principle for a class of voluntary movements. *Journal of Neuroscience*, 4(11):2745, 1984.

[11] R Hurely, J and B Cattell, R. The procrustes program: producing direct notation to test a hypothesised factor structure. *Behav. Sci.*, 7:258–262, 1962.

[12] Seyed Mohammad Khansari-Zadeh and A. Billard. Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2676–2683, 2010.

[13] Seyed Mohammad Khansari-Zadeh and A. Billard. Learning stable non-linear dynamical systems with Gaussian mixture models. *IEEE Trans. on Robotics*, 27(5):943–957, 2011.

[14] Seyed Mohammad Khansari-Zadeh and Aude Billard. BM: An iterative algorithm to learn stable non-linear dynamical systems with Gaussian mixture models. In *Proceeding of the International Conference on Robotics and Automation (ICRA)*, pages 2381–2388, 2010.

[15] F Lacquaniti, C Terzuolo, and P Viviani. The law relating kinematic and figural aspects of drawing movements. *Acta Psychologica*, 54:115–130, 1983.

[16] K. Lashley. The problem of serial order in psychology. *Cerebral mechanisms in behavior. New York: Wiley*, 1951.

[17] Bizzi E Mussa-Ivaladi FA. Motor learning through the combination of primitives. *Philosophical Transactions of the Royal Society of London Series B-Biological Sciences*, 355:1755–1759, 2000.

[18] Frank E Pollick, Uri Maoz, Amir A Handzel, Peter J Giblin, Guillermo Sapiro, and Tamar Flash. Three-dimensional arm movements at constant equi-affine speed. *Cortex*, 45(3):325–339, 2009.

[19] F. Polyakov, E. Stark, R. Drori, M. Abeles, and T. Flash. Parabolic movement primitives and cortical states: merging optimality with geometric invariance. *Biological Cybernetics*, 100(2):159–184, 2009.

[20] Dagmar Sternad and Stefan Schaal. Segmentation of endpoint trajectories does not imply segmented control. *Experimental Brain Research*, 124(1):118–136, 1999.

[21] E. Todorov and M. I. Jordan. Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. *J. Neurophysiol*, 80:696–714, 1998.

[22] P. Viviani and M. Cenzato. Segmentation and coupling in complex movements. *Journal Experimental Psychology Humam Perception and Performence*, 11(6):828–845, 1985.

[23] P. Viviani and C. deSperati. The relationsheep between curvature and velocity in two dimensional smooth persuit eye movement. *The Journal of Neuroscience*, 17:3932–3945, 1997.