SEDS: a Framework to Generate Stable, Adaptive, Reactive, and Human-Like Robot Reaching Motions

Seyed Mohammad Khansari Zadeh and Aude Billard*

January 17, 2013

Abstract

This document provides a brief overview of the framework, called Stable Estimator of Dynamical Systems (SEDS), that is devised to push the level of adaptivity, reactivity, and robustness of robotic systems closer to human movements. SEDS can be seen as a universal framework to model a subclass of robot motions called *"reaching movements"*, i.e. movements in space stopping at a given target (also referred to as episodic motions, discrete motions, or point-to-point motions). These reaching movements can then be used as building blocks to form more advanced robot tasks. To achieve a high level of proficiency as described above, SEDS particularly generates control policies that: 1) *resemble* human motions, 2) *guarantee* the accomplishment of the task (if the target is reachable), and 3) can *instantly* adapt to dynamically changing environments.

This document is structured as follows: Section 1 describes the motivation behind the developed framework. Section 2 explains the main principles of our method, and (briefly) situates it among the relevant state of the art approaches. Section 3 reports on the experimental evaluation and comparison to alternative approaches, and Section 4 highlights the main contributions of our approach and concludes the document.

It should be noted that this document is mainly devoted to describe the general aspects of our method, avoiding a significant amount of mathematical details, derivations, and analysis. For a thorough description of the developed framework, please refer to (Khansari-Zadeh, 2012).

1 Motivations

FROM the moment you turn off the alarm clock in the morning till you turn it on again at night, you do numerous amounts of reaching movements without even noticing. For instance, imagine you are in a meeting. You are talking with your friend and at the same time reaching for an apple in the tray next to you. Whilst deep in your discussion and trying to reason out your argument, you start biting the apple. Your other friend — who is coincidentally a roboticist — does not enjoy the conversation, and instead is staring at your hand movement. She can clearly see that with every bite, your hand finds a different way to your mouth, though you may not even think about it. Even when the person next to you unintentionally bumps into your arm and you turn your head to see who has hit you, your hand can instantly adapt to these changes and reach your mouth effortlessly without pause.

Having robotic systems that exhibit the level of robustness and adaptability described above is essential, particularly if we envision to bring robots into our daily lives. Let us take another example. Imagine you are being served tea by a robot. As the robot is about to pour the boiling liquid in the cup you are holding, you sneeze. As a result of your sudden jolt, the cup is displaced and your hand is now in the way of the robot in place of the cup. It would be desirable that the robot to be able to react swiftly, and redirect its motion to the cup while avoiding your hand. This is one of the many examples that emphasize the necessity for adaptive robotic systems that

*S.M. Khansari-Zadeh and A. Billard are with the LASA Laboratory, School of Engineering, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. Email: `{mohammad.khansari,aude.billard}@epfl.ch` can react in the order of a second. In addition to the need of having adaptive robotic systems, it would be even more fascinating if we could achieve the described level of proficiency in an easy and intuitive way. In such a way, as it happens, that anybody could impart an instruction, education or command to a robot, even with no in-depth knowledge of engineering, of the robotic platform, or of the task at hand.

Devising a framework that can fulfil the combination of the above two requirements is nontrivial. For years classical control approaches have provided us with tools to perform high precision tasks with industrial robots. However, these approaches rely heavily on hardcoded motions with a limited ability to adapt autonomously to a dynamically changing environment. Besides, they require a large amount of engineering knowledge and experience to efficiently put them to use. Other approaches, such as global path planning, are comparatively easier to use and can generate feasible trajectories that can fulfil the constraints of the task at hand (Diankov & Kuffner, 2007; Burns & Brock, 2005; Toussaint, 2009). Despite recent efforts at reducing the computational costs of such global searches for a feasible path, however, these methods cannot offer the reactivity sought to adapt to dynamic environments.

Our main scope is to devise a generic framework that fulfils the two important requirements described above. This framework is particularly developed for a subclass of robot motions called *"reaching movements"*. Modeling of reaching movements provides basic components, the so-called motion primitives (Schaal, 1999), which can be seen as a basis from which more advanced robot tasks can be formed[1]. To avoid manually hardcoding robot motions, we exploit the power of machine learning techniques and take an *Imitation Learning (IL)* approach to build a generic model of robot reaching movements from a few examples provided by the expert. To achieve the required level of robustness and reactivity, we formulate the encoding of reaching movements as a control law that is driven from *nonlinear dynamical systems.*

When modeling robot motions with nonlinear Dynamical Systems (DS), ensuring stability of the learned DS (from a set of demonstrations of the task) is a key requirement to provide a useful control policy. Hence, the major part of our effort is concentrated on addressing the challenging problem of *"how to build a locally or globally stable estimate of nonlinear dynamical systems from a set of user demonstrations?"* The answer to this question is non-trivial and requires to bridge the gap between different fields such as "Control Theories" and "Machine Learning".

2 Approach

As outlined in the previous section, we aim at providing a generic framework to generate with ease control policies that exhibit a high level of proficiency and adaptivity. Specifically, our goal is to build an all-encompassing model of robot reaching movements that fulfils the following desiderata: 1) It should be estimated via imitation learning, 2) It should ensure accomplishment of the task as long as the target is reachable, 3) It should be robust to perturbations and adaptable to changes in a rather complex dynamic environments with several static and moving obstacles, and 4) It should perform the adaptation on-the-fly without any need to re-plan. The first criterion is essential as it provides an easy and intuitive means to program robots. Additionally, it allows generating motions that mimic human motions (which are more natural looking). The other three criteria are crucial since they increase the reliability of the approach, and are essential particularly when we have robotic systems that work in the close vicinity of humans. Our approach is founded on two main pillars: *Imitation Learning* and *Dynamical Systems*. Next, we describe how the field of robotics has evolved from the tedious hardcoding of robot motions to the appearance of learning-based approaches. Then we introduce imitation learning, a particular form of robot learning, and finally explain the emergence of dynamical system approaches to modeling robot motions.

2.1 Robot Learning

Classical approaches to robot control rely on following pre-programmed motions with a limited level of adaptivity to changing environments. Manual programming of robot motions often requires a

[1]As an example, consider the standard "pick-and-place" task, which can be decomposed as follows: First reach to the item, then after grasping move to the target location, and finally return home after release.

large amount of engineering knowledge about both the task and the robot and, above that, it can become particularly non-intuitive when dealing with high Degrees of Freedom (DoF) robotic systems or fulfilling requirements of complex tasks. Emergence of a new generation of robots that need to perform a wide variety of tasks in human daily lives stresses further more the importance of seeking alternative techniques as manual programming cannot be a reasonable option anymore. In response to these concerns, learning-based approaches appear as a promising route to automate the tedious manual programming phase by having a robot actually learn how to perform a desired task.

Imitation Learning (also referred to programming by demonstrations, apprenticeship learning, or learning from demonstration) is one of the fundamental learning mechanisms in humans daily lives. Day in day out, we use IL so frequently that we seldom even notice. We employ IL for different purposes: from understanding a common social behavior in a small group to learning complex movements in sport games. In robotics, IL started attracting attention at the beginning of the 1980s (Lozano-Perez, 1983). Through the years, IL has been advocated as a powerful means to bootstrap robot learning (Kuniyoshi et al., 1989; Münch et al., 1994; Schaal, 1999). IL provides an intuitive way to transmit skills to robots without explicitly programming them. Moreover, it speeds up robot learning by reducing the search space of solutions (Billard et al., 2008). IL-based approaches have proved to be interesting alternative to classical control and planning methods in different applications such as locomotion (Ijspeert et al., 2002), control of acrobatic helicopters (Coates et al., 2008), reaching movements (Calinon et al., 2007), etc. A more detailed overview of IL-based approaches is provided in (Khansari-Zadeh, 2012).

In IL, the final goal is to perform a task as similar as possible to the examples provided by the teacher (also called expert or demonstrator). The notion of similarity is defined in terms of a *metric of imitation performance*. This metric provides a means to quantitatively express the user's intention during the demonstrations, and constitutes all features that remain invariant across them (Calinon, 2009). By using these features rather than only mimicking the teacher, IL-based approaches are able to generalize the task to unseen situations.

When modeling robot motions via IL, it should be noted that inferring solely based on the user examples may not necessarily be sufficient to derive a useful control policy. User demonstrations are usually noisy, and the estimated model may not be accurate enough to satisfy a task's hard constraints. In many situations specifying the minimum accuracy required to fulfil the task's constraint is non-trivial, and quite often satisfying that requirement is much harder (if not impossible). Let us take the example of figure skating. In this sport, only a few combinations of the whole body movements are acceptable and leads to successful performance of intricate and challenging forms such as spins, jumps, footwork, etc. Even the slimmest change in the angle between the skate's blade and the ice could affect the performer's stability and cause her falling. For these kinds of tasks, it might be much easier to impose a task's hard constraints during the learning process to control the range of possible values that the learning parameters can take. Similarly, in the approach that is adopted in our framework, the estimation of nonlinear DS through the direct use of existing techniques is not effective. These methods do not optimize under the constraint of making the system stable at the attractor, and thus, they are not guaranteed to result in motions that can reach the target. Hence we develop an alternative statistical-based technique to build an estimate of nonlinear DS under strict stability conditions.

2.2 Dynamical Systems

Classical approaches to modeling robot motions rely on decomposing the task execution into two separate processes: *planning* and *execution* (Brock & Khatib, 1999). The former is used as a means to generate a feasible path that can satisfy the task's requirements, and the latter is designed so that it follows the generated feasible path as closely as possible. Hence these approaches consider any deviation from the desired path (due to perturbations or changes in environment) as the tracking error, and various control theories have been developed to efficiently suppress this error in terms of some objective functions. Despite the great success of these approaches in providing powerful robotic systems, particularly in factories, they are ill-suited for robotic systems that are aimed to work in the close vicinity of humans, and thus alternative techniques must be sought.

In robotics, DS-based approaches to motion generation have been proven to be interesting alternatives to classical methods as they offer a natural means to integrate planning and execution into one single unit (Kelso, 1995; Schaal et al., 2000; Billard & Hayes, 1999; Selverston, 1980). For instance when modeling robot reaching motions with DS, all possible solutions to reach the target are embedded into one single model. Such a model represents a global map which specifies *instantly* the correct direction for reaching the target, considering the current state of the robot, the target, and all the other objects in the robot's working space. Clearly such models are more similar to human movements in that they can effortlessly adapt its motion to change in environments rather than stubbornly following the previous path. In other words, the main advantage of using DS-based formulation can be summarized as: "Modeling movements with DS allows having robotic systems that have inherent adaptivity to changes in a dynamic environment, and that can swiftly adopt a new path to reach the target". This advantage is the direct outcome of having a unified planning and execution unit.

For the reasons outlined above, we take a DS approach and models robot reaching movements with *autonomous nonlinear multi-dimensional DS*. The choice of "autonomous" (also called time-invariant) DS is essential as it allows instant adaptation to perturbations without any need to re-plan. Encoding motions with multi-dimensional DS is also advantageous since it allows capturing correlation across all dimensions, which may be crucial for accurate reproduction. As none of the existing statistical tools could build a stable estimate of the above DS, we present a novel statistical-based framework to build an estimate of the adopted DS formulation under strict global stability constraint.

2.3 How does it work?

In our framework, robots are taught to perform a task by observing a set of demonstrations provided by a human teacher. Demonstrations to a robot may be performed in different ways: back-driving the robot, teleoperating it using motion sensors, etc. (see Fig. 1). The learning process consists of extracting the relevant information from the demonstrations and encoding this information into a motion model that can be used to reproduce the task.

The ability to generalize the task and to adapt it to a new situation is a critical feature that the encoded motion model should have. This concerns the problem of performing the task under different circumstances than those present during demonstrations, which is desirable mainly for two reasons: 1) The number of demonstrations can be kept small, and 2) Given appropriate adaptation, an acquired skill can be used to carry out a more complex task than the teacher is capable of demonstrating.

As discusses before, in our framework, we take a dynamical system-based approach to modeling robot motions as it is a powerful tool for robust control of robot motions from a small set of demonstrations. A robot motion that is modeled with DS is inherently robust to perturbations because it embeds all possible solutions to reach a target into one single function. Such a function represents a global navigation map which specifies on-the-fly the correct direction for reaching the target, considering the current position of the robot and the target.

In DS approaches, the control policy to drive a robotic platform is modeled with a first or higher order DS:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \tag{1}$$

where $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^d$ is a nonlinear autonomous (i.e. time invariant) function, and $\boldsymbol{\xi} \in \mathbb{R}^d$ is a state variable that can unambiguously describe a discrete motion of a robotic system (e.g. $\boldsymbol{\xi}$ could be a robot's joint angles, the position of an arm's end-effector in the Cartesian space, etc).

When controlled through a DS, a robot motion unfolds in time with no need to re-plan. In our framework, an estimate of the function $\boldsymbol{f}(\boldsymbol{\xi})$ is obtained from a few demonstrations of the task at hand. The estimated DS captures the invariant features in the user demonstrations, and can generate motions that resemble the user demonstrations. For instance, Fig. 2 shows an example of learning 20 human handwriting motions using our framework. For each motion, the user provides 3 demonstrations by drawing a pen on a Tablet-PC. These demonstrations are shown with red dots. By using our framework, a DS model is estimated for each of these motions. By using these models, we are now able to generate motion that follows the same behavior (i.e. follows the same shape, curvature, nonlinearity) from different point in space (see the solid curves in Fig. 2). In

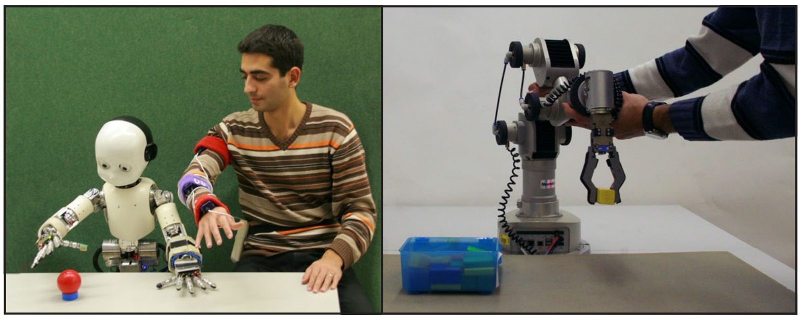


Figure 1: Demonstrating motions by teleoperating a robot using motion sensors (**left**) or by back-driving it (**right**).

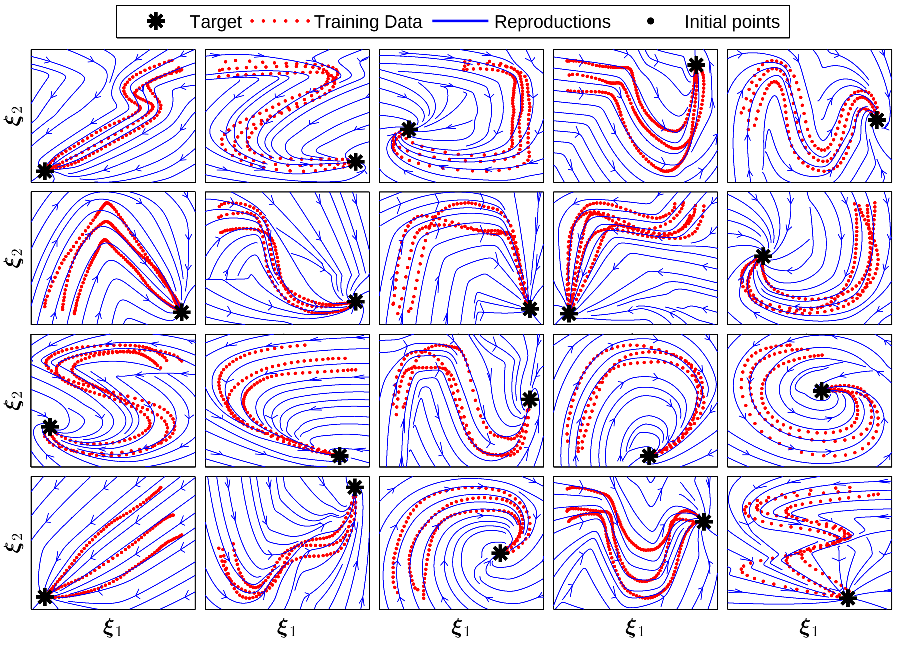
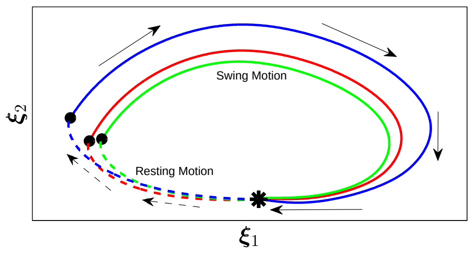


Figure 2: Learning a library of 20 human handwriting motion by using SEDS framework. This library is not fixed and the user can freely add a new style of writing, or modify one of the existing motions just by introducing a new set of sample trajectories.

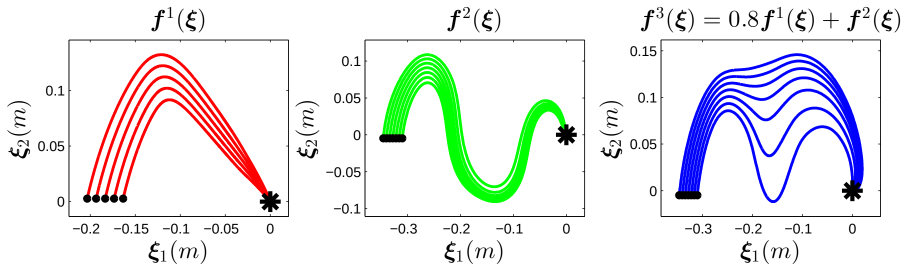
other words, we are now able to give the robot the ability to follow a specific behavior (e.g. writing a letter 'z') just with a few examples. Additionally, the user can also teach the robot a new style of writing the letter 'z', by just providing a new set of sample trajectories.

Another feature of our framework is its modularity. In fact, each DS model codes a specific motion (behavior), which is called a movement primitive (also known as motor primitive). These motion primitives can be seen as a building block that can be used to generate more complex or new motions through sequencing or superimposition of the primitives. This modularity of DS-based movement primitives is essential as it allows controlling a wide repertoire of movements from a (small) set of basic motions (Schaal et al., 2003; Wolpert & Kawato, 1998). Figure 3 shows two examples of exploiting this modularity of movement primitives to generate new motions.

To summarize, a conceptual workflow describing how our framework works is depicted in Fig. 4. As it is illustrated, first the user provides a set of demonstrations describing how robot should perform a task. Then a DS-based control policy is build using the 'Learning Core' of our framework. This DS model can be used to generate robot commands to properly execute the desired task based on the situation in the robot's workspace. The 'Execution Core' of our framework is able to adapt in realtime a new trajectory if any change happen in the workspace (e.g. the target point or other objects are replaced). For example in the image shown in the block 'robot' in Fig. 4, the robot is required to put a (transparent) glass in front of the user in a cluttered environment. As the robot



(a) This graph shows an example of a tennis swing which is composed of a swing and a resting phase. The motion in each phase is encoded as a basic movement primitive, and the complete tennis swing motion is thus obtained by sequencing these two primitives. The trajectories generated by the swing and resting models are shown in solid and dashed lines, respectively. The direction of the motion is indicated by arrows. Only three examples of generated trajectories are shown here (plotted as red, green, and blue lines).



(b) In this example $\boldsymbol{f}^1(\boldsymbol{\xi})$ and $\boldsymbol{f}^2(\boldsymbol{\xi})$ are two basic movement primitives that represent an angle and sine-shaped motions, respectively. The new movement primitive $\boldsymbol{f}^3(\boldsymbol{\xi})$ that includes a mixture of both behaviors is obtained through a linear superposition of $\boldsymbol{f}^1(\boldsymbol{\xi})$ and $\boldsymbol{f}^2(\boldsymbol{\xi})$.

Figure 3: Illustration of two examples exploiting modularity of DS models to generate **(a)** a more complex motion and **(b)** a new movement primitive. In this figure, the black star and circles indicate the target and initial points, respectively.

approaches, the person intentionally moves the red glass in a way that crosses the robot trajectory. Despite this quick change, the robot is able to robustly handle this perturbation and successfully put the transparent glass in front of the user without hitting other objects (more information on this experiment is provided in Section 3.3). In case it is necessary, our approach also supports online learning which allows the user to refine the behavior of the robot in an interactive manner.

3 Robot Experiments

We evaluated the performance of our method in various experiments and on a wide variety of robotic platforms. In this document, we only report on four of these experiments: 1) The pick-and-place experiment, 2) Playing mini-golf, 3) Working on a cluttered office environment, 4) Dodging a fast moving box. For a complete list of all validation experiments, please refer to (Khansari-Zadeh, 2012).

3.1 Pick-and-place experiment

In the first experiment, we teach the 6-DoF industrial Katana-T arm how to put small blocks into a container (see Fig. 5). We use the Cartesian coordinates system to represent the motions (i.e. $\boldsymbol{\xi} = [x\ y\ z]^T$ and $\dot{\boldsymbol{\xi}} = [\dot{x}\ \dot{y}\ \dot{z}]^T$). In order to have human-like motions, the learned model should be able to generate trajectories with both similar position and velocity profiles to the demonstrations. In this experiment, the task was shown to the robot six times. In each demonstration, we recorded the robot's joints angles at 20Hz by directly reading them from each joint's encoder. Forward kinematics was used to compute the Cartesian position and velocity of the end-effector. This data was then used to model the task. Figure 5a illustrates the obtained results for generated trajectories starting from different points in the task space. The position of objects is detected at the rate of 50 fps using two high-speed Mikrotron MK-1311 cameras. The direction of motion is

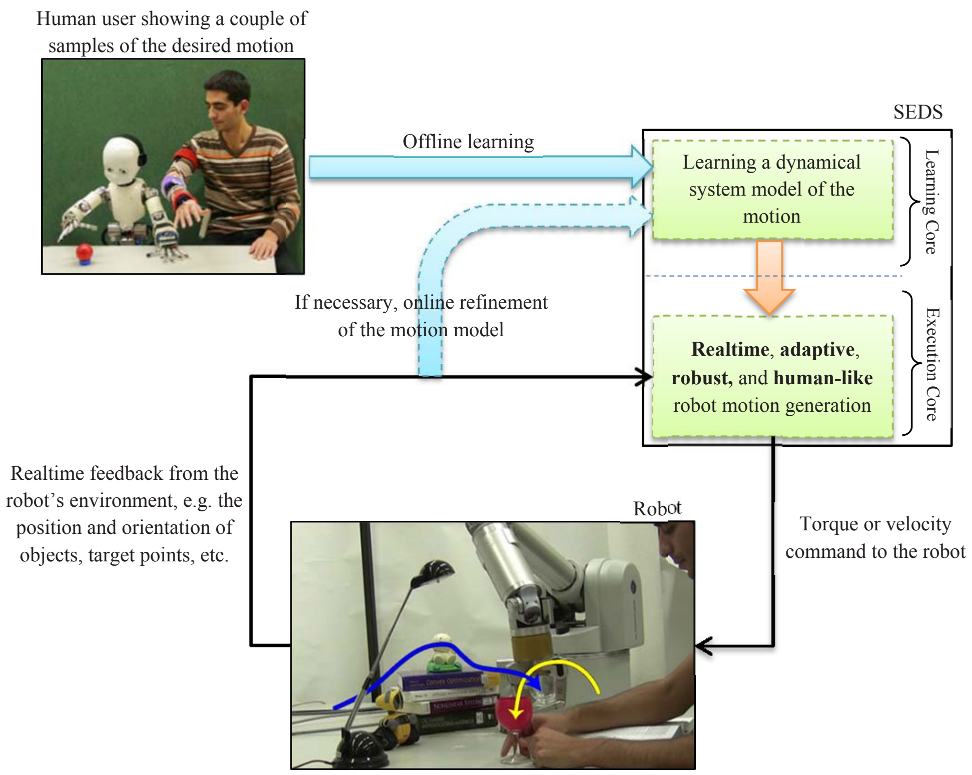
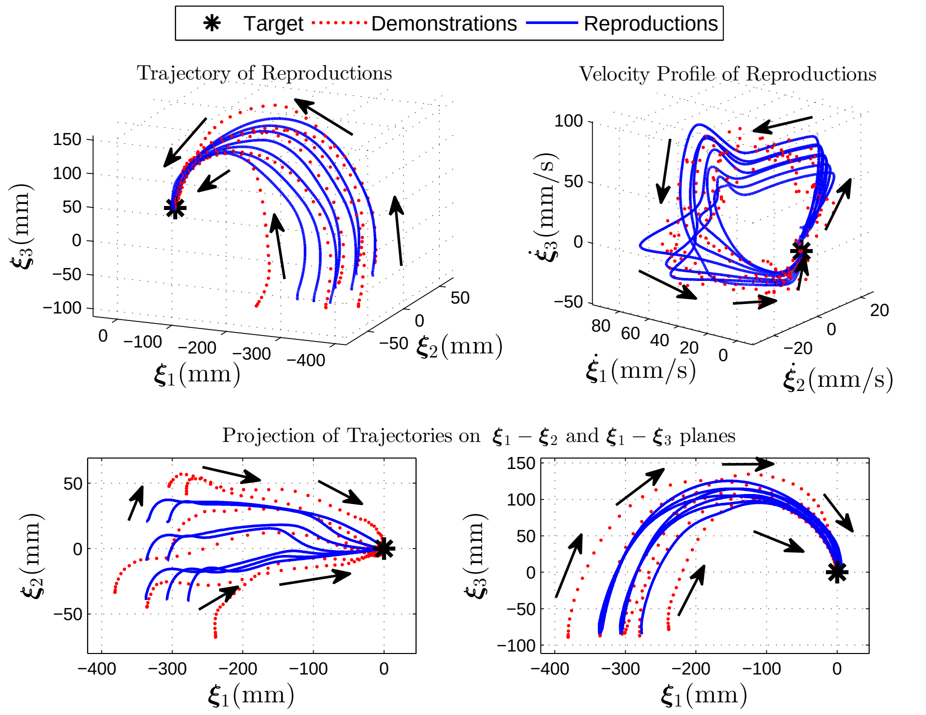

Human user showing a couple of samples of the desired motion
Offline learning
SEDS
Learning a dynamical system model of the motion
Learning Core
If necessary, online refinement of the motion model
Realtime, adaptive, robust, and human-like robot motion generation
Execution Core
Realtime feedback from the robot's environment, e.g. the position and orientation of objects, target points, etc.
Robot
Torque or velocity command to the robot

Figure 4: A conceptual workflow describing how the SEDS framework works. For further information please refer to Section 2.3.

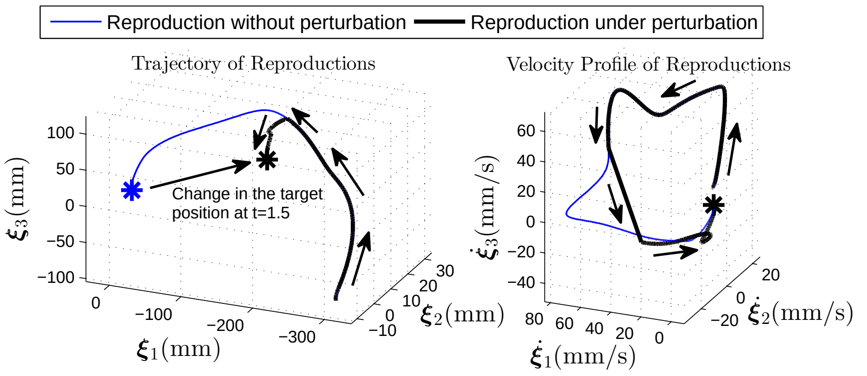
indicated by arrows. All reproduced trajectories are able to follow the same dynamics (i.e. having similar position and velocity profile) as the demonstrations.

Immediate Adaptation: Fig. 5b shows the robustness of the model to the change in the environment. In this graph, the original trajectory is plotted in thin blue line. The thick black line represents the generated trajectory for the case where the target is displaced at $t = 1.5$ second. Having defined the motion as an autonomous DS, the adaptation to the new target's position can be done instantly.

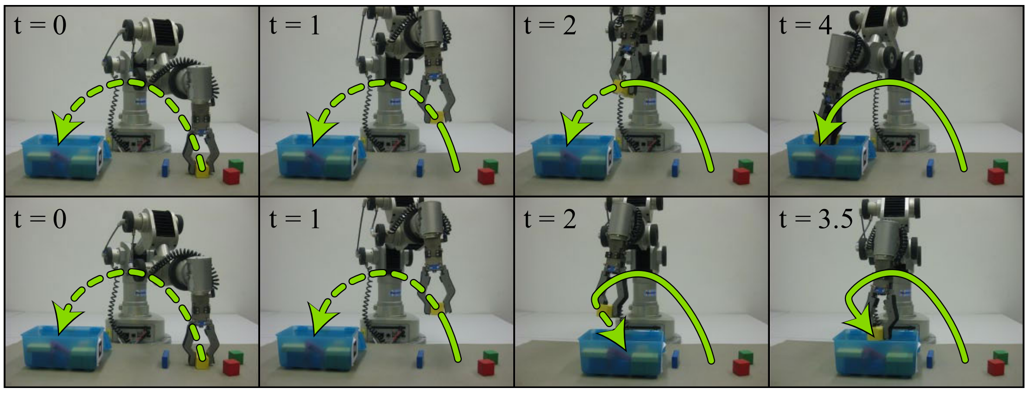
Increasing Accuracy of Generalization: While convergence to the target is always ensured in our approach, due to the lack of information for points far from demonstrations, the model may reproduce some trajectories that are not consistent with the usual way of doing the task. For example, consider Fig. 6a, i.e. when the robot starts the motion from the left-side of the target, it first turns around the container and then approaches the target from its right-side. This behavior may not be optimal as one expects the robot to follow the shortest path to the target and to reach it from the same side as the one it started from. However, such a result is inevitable since the information given by the teacher is incomplete, and thus the inference for points far from the demonstrations is not reliable. In order to improve the task execution, it is necessary to provide the robot with more demonstrations (information) over regions not covered before. By showing the robot more demonstrations and re-training the model with the new data, the robot is able to successfully accomplish the task (see Fig. 6b).



(a) The ability of the model to reproduce similar trajectories starting from different points in the space.

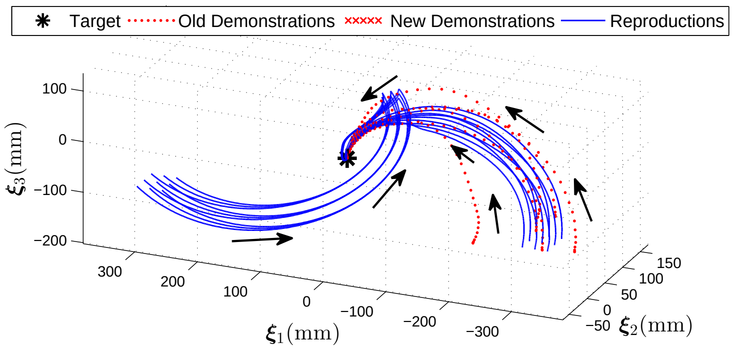


(b) The ability of the model to adapt its trajectory on-the-fly to a change in the target's position

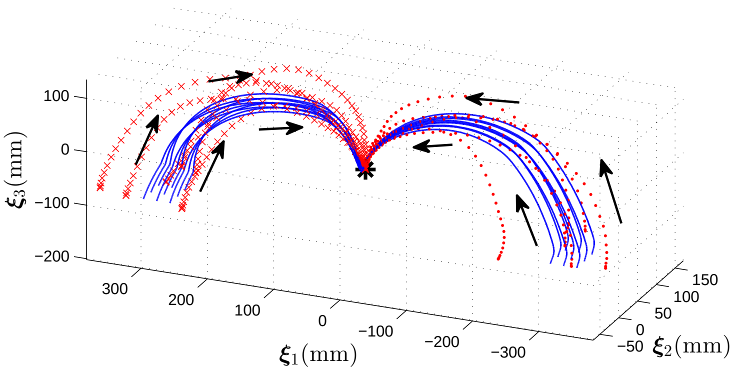


(c) Stills showing the reproduction of the task by the robot in the absence and presence of perturbation.

Figure 5: The Katana-T arm performing the experiment of putting small blocks into a container. Please see the text for further information.



(a) Generalization based on the original model.



(b) Generalization after retraining the model with the new and old datasets.

Figure 6: Improving the task execution by adding more data for regions far from the demonstrations.

3.2 The Minigolf Experiment

In the second experiment, we validated our framework in the context of playing minigolf using 6-DoF Katnata-T robot. In the minigolf task considered in this experiment, the player only gets one chance to sink the ball. For this experiment, we collected a set of demonstrations by passively moving the robot arm to strike the ball (see Fig. 7a). In total, seven successful demonstrations were collected and used to learn the task (see Fig. 7b). The location of the ball was detected at the rate of 80 fps using two high-speed Mikrotron MK-1311 cameras.

Figure 7c illustrates the reproductions of the motion after using our framework to learn the task from demonstrations. Figures 7d to 7f represents the velocity profile of the reproductions versus demonstrations along the axes x, y, and z respectively. Figure 7g shows the sequence of the motion for one of the reproductions.

Generalization Ability: Figure 8 illustrates the generalization ability of the model to different positions of the golf ball and the hole. In Fig. 8a, we changed the position of the ball along the y-axis from 0 to -0.18 m. We also changed the position of the hole so that the vector connecting the center of the ball and the hole always remained along the x-axis. In all cases, the robot successfully hit the ball with the correct speed at the target. Figure 8b demonstrates the adaptation of the robot motion to three different positions of the hole. Though the initial configuration of the robot's arm and the ball positions were fixed, the robot took three different paths to hit the ball in the correct direction.

Adaptation to Changes in Dynamic Environments: Similar to all autonomous DS, the proposed model is inherently robust to external perturbations and can provide instant adaptation to changes in the environment. Figures 9a and 9b illustrate the model's behavior in a dynamic environment. In these experiments, during the robot's arm movement, the ball (Fig. 9a) and the hole (Fig. 9b) were displaced along the negative direction of the y-axis. At each time step, the robot successfully adapted its trajectory to the new position of the ball/hole until it hit the

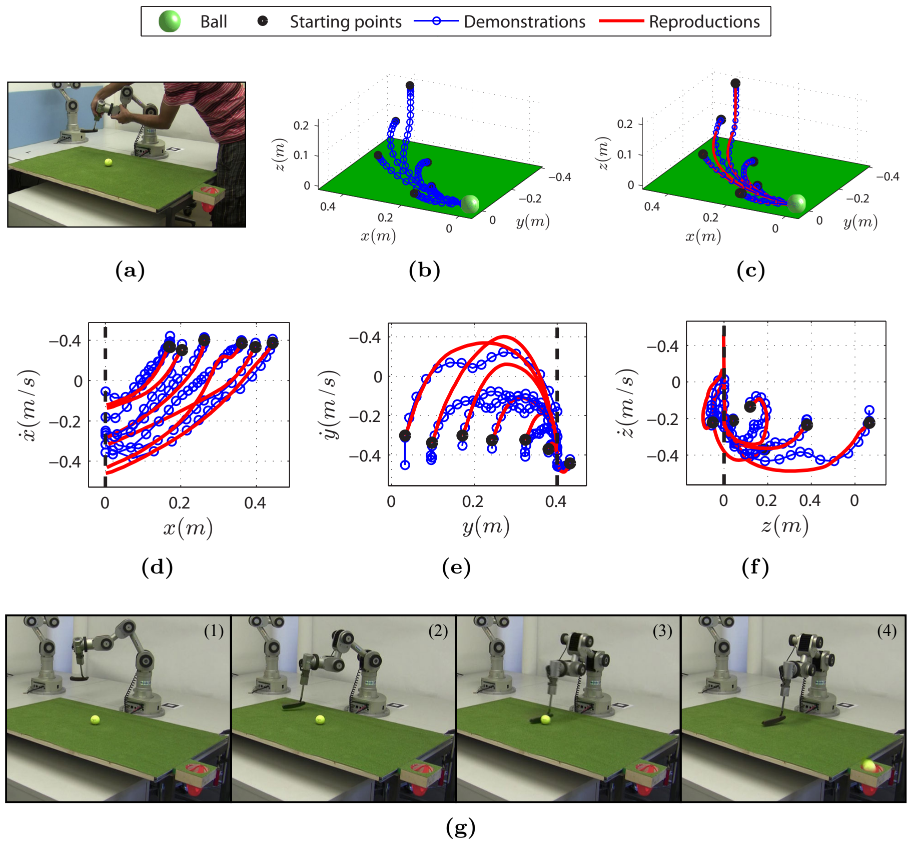


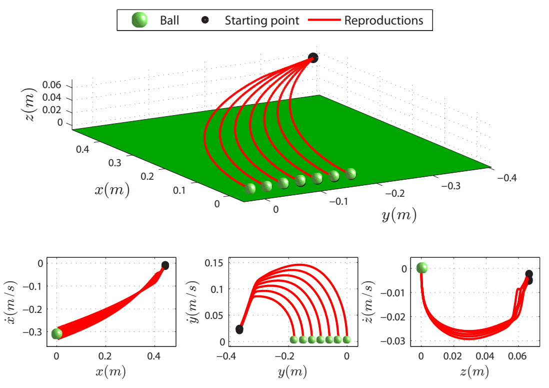
Figure 7: **(a)** Kinesthetic demonstration of the putting motion to the 6-DoF Katana-T robot. **(b)** Illustration of the collected successful demonstrations. **(c)** Reproductions of the motion from the model learned with the extended version of SEDS. **(d)-(f)** Evaluation of the model's accuracy in estimating the desired velocity profile. The thick dashed lines locate the position of the ball. **(g)** Illustration of one of the generated motions sequences.

ball. In both examples, the robot successfully managed to hit the ball in the correct direction as the adaptation to the perturbation was done on-the-fly, i.e. without any re-planning. Note that despite the inherent robustness of stable autonomous DS to perturbations, there is an upper bound for the maximum value of perturbations that can be handled. This upper bound is due to the robot's hardware limitations, which affect the maximum acceleration and velocity that the robot can achieve. Thus, if the robot faces a large perturbation when it is close to the ball, it might not be able to react swiftly and hit the ball with the correct hitting direction and speed.

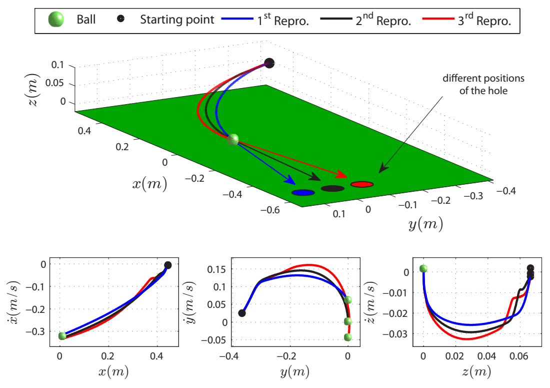
Experiments on advance fields: We have also verified our framework on more challenging fields using the 7-DoF Barrett WAM arm as shown in Fig. 10. For brevity of this document, we do not report on this experiment here. Please refer to (Khansari-Zadeh, 2012) for further information.

3.3 Working in a Cluttered Office Environment

In this experiment we evaluate our method in the presence of several obstacles including a desk lamp, a pile of books, a Wall-E toy, a pencil sharpener, a book, a (red) glass, and a desk. The task consists of having the 7-DoF Barret WAM arm place a (transparent) glass on the desk, and in front of the person (see Fig. 11). The position and orientation of all the objects except the glass are pre-set. In order to have a more realistic experiment, at each trial we add a error vector $\boldsymbol{\epsilon}$ to the predefined position of each obstacle $\boldsymbol{\xi}^{o,k}$ to account for uncertainty in the environment, i.e. $\hat{\boldsymbol{\xi}}^{o,k} = \boldsymbol{\xi}^{o,k} + \boldsymbol{\epsilon}^k$. The value of each component of the error vector $\boldsymbol{\epsilon}^k$ is drawn from a Gaussian distribution with $\mathcal{N}(0, 0025)$. The position of the glass is actively tracked through the stereo camera described above. The maximum tracking error in sensing the glass position is ± 0.05m. The orientation of the glass is not measured, though it may change during each trial.

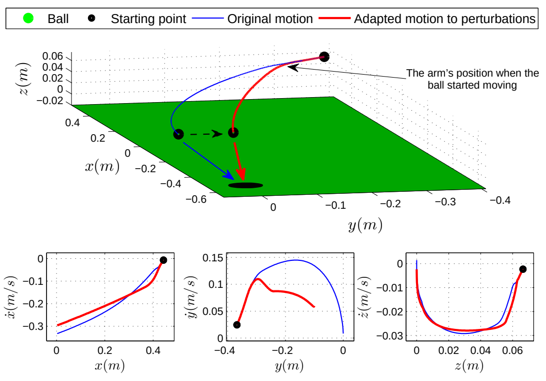


(a) Generalization to different ball positions.

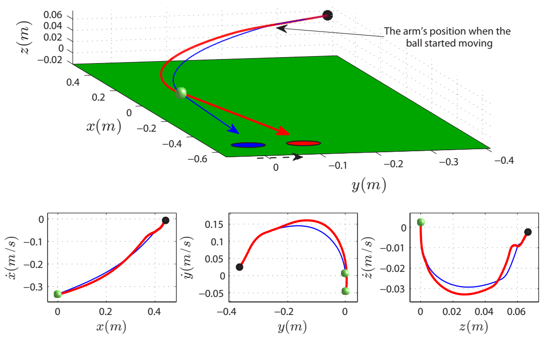


(b) Generalization to different hole positions.

Figure 8: Evaluation of the model's performance in generalization.



(a) Robustness to changes in the ball position.



(b) Robustness to changes in the hole position.

Figure 9: Performance evaluation of the model in a dynamic environment. In this example the ball **(a)** and the hole **(b)** are pushed along the negative direction of the y-axis, as the robot approaches.

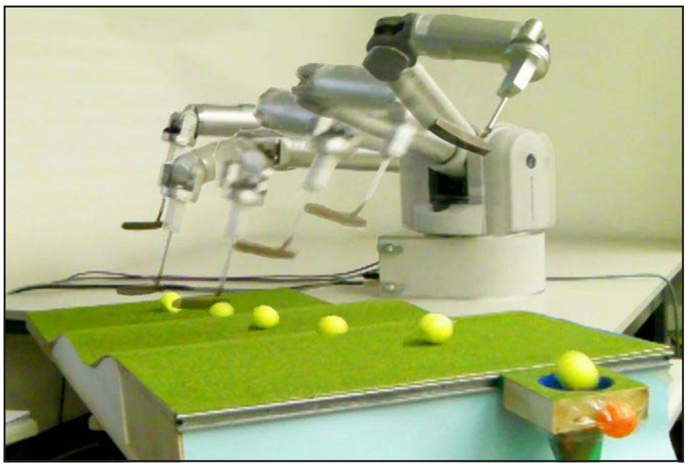
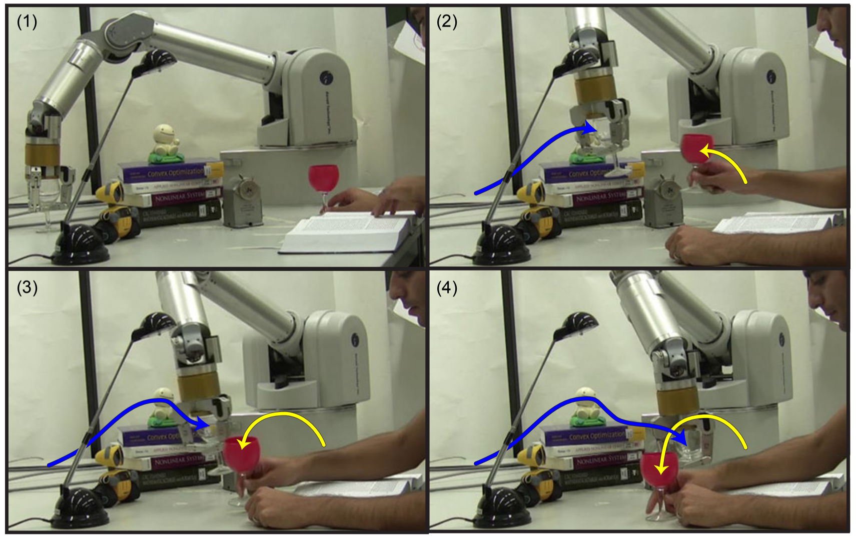
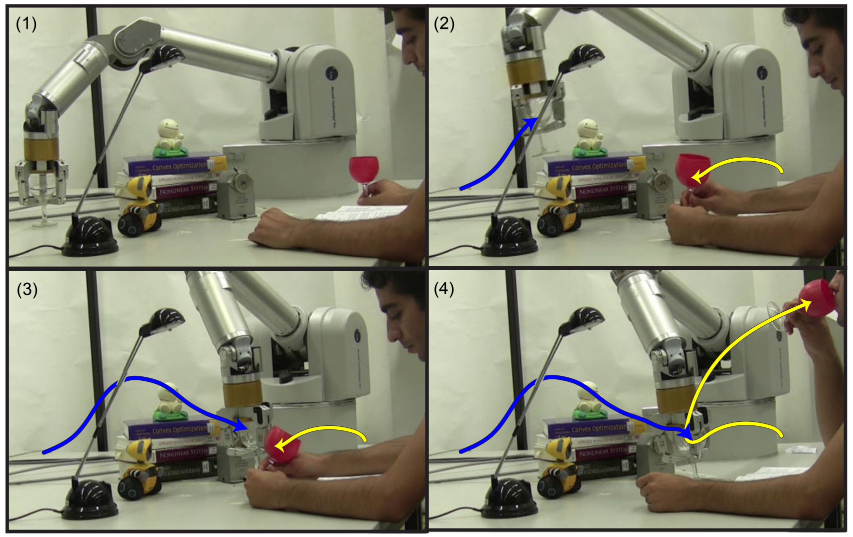


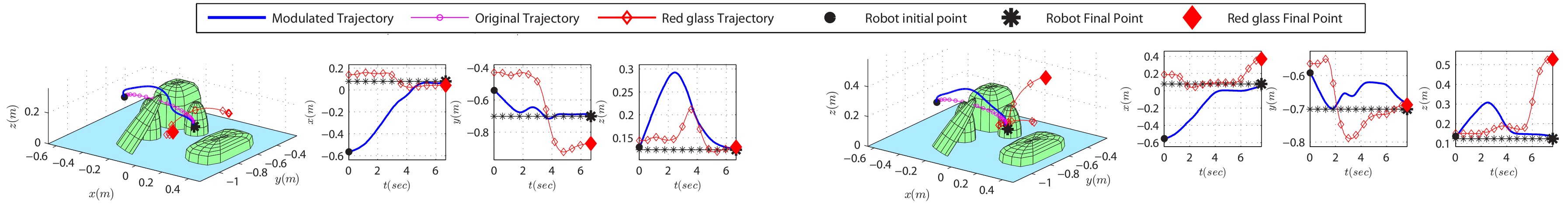
Figure 10: The hitting motion on the WAM. The ball and the hole are continuously tracked by a stereovision system.



(a) Sequences of the motion for the first trial. In this experiment, the person moves the red glass from his right to his left hand side while the robot is approaching the target point. (1): The initial configuration. (2): The person intentionally moves the glass in a way that crosses the robot trajectory to the target point. (3) & (4): In order to avoid hitting the red glass, the robot deflects its trajectory towards the left side of the person, and then approaches the target from that side. The trajectories of the robot and the red glass are shown with blue and yellow curves, respectively. The traveled path is indicated with a solid line.



(b) Sequences of the motion for the second trial. (1) & (2): In this experiment, the person takes the glass from its right hand side and moves it to the target position while the robot is approaching. (3): In this situation, the robot stops near the red glass (and the target) since it cannot get any closer to the target. The robot waits at this position until the person clears the areas. (4): When the red glass is lifted, the robot moves towards the target point. The trajectories of the robot and the red glass are shown with blue and yellow curves, respectively.



(c) Illustration of the robot, target and obstacle motions for the first trial.

(d) Illustration of the robot, target and obstacle motions for the second trial.

Figure 11: Evaluation of the proposed method in a complex environment. In this experiment, the robot is required to put a glass on the desk and in front of the person, while avoid hitting several objects including a desk lamp, a pile of books, a Wall-E toy, a pencil sharpener, an open book, a (red) glass, and a desk. All the objects except the red glass are fixed and their convex envelope are shown in green. The trajectory of the red glass is indicated by red diamonds (for the clarity of the graph, we do not display the envelope of the red glass).

In this document, we report on two trials of this experiment. We use the same DS function that was described in the previous robot experiments to control the robot motions. In the first trial, the person moves the red glass from his right to his left hand side (i.e. along the negative direction of the y-axis) while the robot is approaching the target point. The person intentionally moves the glass in a way that crosses the robot trajectory to the target point (see Fig. 11a). In order to avoid hitting the red glass, the robot deflects its trajectory towards the negative direction of y-axis, and then approaches the target from its left side (in Fig. 11c, see the robot trajectory along y-axis in the time period $t = [3 \;\; 4]$ seconds).

In the second trial, the person takes the glass from its right hand side and moves it to the target position while the robot is approaching (see Fig. 11b). In this situation, the robot stops near the red glass (and the target) since it cannot get any closer to the target (in Fig. 11d, see the time evolution of the robot trajectory in the time period $t = [4 \;\; 6]$ seconds). The robot waits at this position until the person clears the areas. When the red glass is lifted, the robot moves towards the target point.

3.4 Dodging a fast moving box

In this experiment we evaluate our approach in the presence of a fast moving obstacle. The experiment consisted of having the 7-DoF KUKA DLR arm stay in a default target position while a box is slid towards the robot at high speed. Thus the robot should react quickly and change its position so that the box passes without any collision (see Fig. 12).

The KUKA robot is controlled in the Cartesian coordinate system, and the control commands are sent at 1000Hz. A SEDS model is used to control the robot motion by generating velocity commands to keep the robot's end-effector close or, when it is feasible, at the target point. The geometry of the box and the table is defined and given to the system. The box's position and orientation are tracked at 240Hz using an OptiTrack vision system. We use a Kalman filter to reduce the noise effect on estimations. The position of the working table is set fixed in the whole experiment.

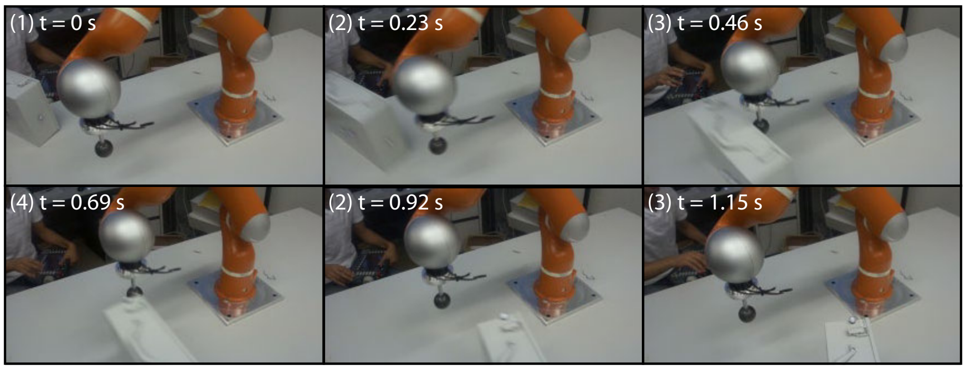
In total we ran 20 trials, lasting between 0.8 to 1.3 seconds, in which the box was slid from different initial configurations with various linear and angular velocities. In each trial, the box was set to an initial distance of about 0.5 meter away from the robot and was thrusted so as to reach a maximum linear velocity of $0.6 \sim 1.5$ m/s and/or a maximum angular velocity of $40 \sim 120$ deg/s. In 16 out of the 20 trials, the robot successfully managed to dodge the box. Figure 12 shows sequences of the motion for four of the trials.

3.5 Comparison to Alternative Approaches

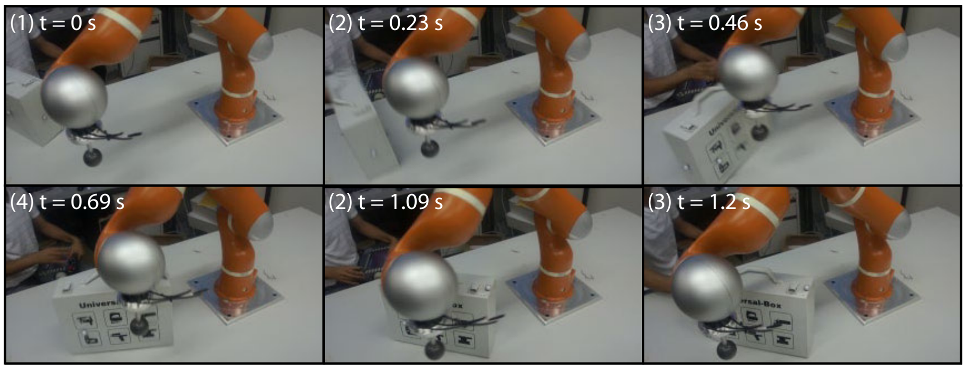
As a part of the European project AMARSi, a benchmark framework was developed to evaluate and compare different DS-based approaches of generating reaching movements within the consortium. This benchmark evaluates the DS-based methods on a library of human motions and on the basis of different criteria such as max/mean jerks, error in reaching the target, error in following the desired behavior, etc. Figure 13 shows the final result from the benchmark. As it is illustrated, our framework is about twice as performant as the runner-up method. Please refer to (AMARSi Deliverable, 2011) for further details about the benchmark and results.

4 Main Contributions and Conclusion

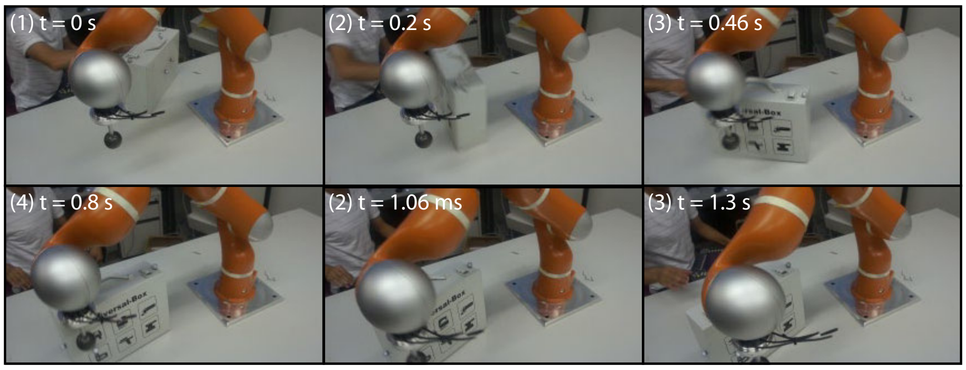
The main contribution of the proposed framework lies in providing a generic and unified framework based on DS, capable of generating various robot discrete movements ranging from simple pick and place motions to agile striking movements. The SEDS framework can build an estimate of nonlinear multi-dimensional DS from a set of examples while ensuring its global asymptotic stability at the target. As highlighted in (Khansari-Zadeh, 2012), to date, existing DS-based approaches to encode robot motions rely either on some heuristics with the aim to build a locally stable estimate of nonlinear DS without any guarantee that such a model is feasible, or they depend on a (time-dependent) switching mechanism to ensure stability by switching from an unstable nonlinear DS to



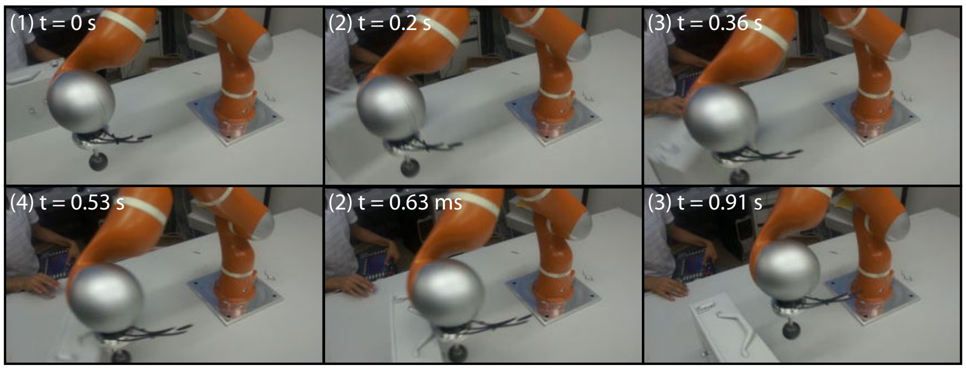
(a) First trial.



(b) Second trial.



(c) Third trial.



(d) Forth trial.

Figure 12: Illustration of sequences of motion for 4 out of the 20 executed trials. In this experiment the robot was required to dodge a sliding box that was launched 20 times from different initial configurations with various linear and angular velocities. For further information please refer to Section 3.4.

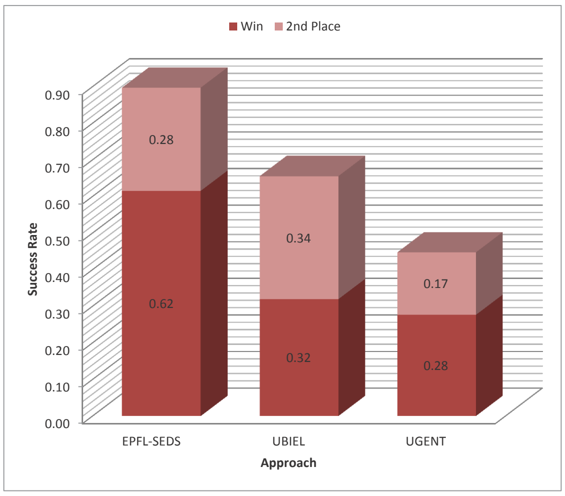


Figure 13: The the final result from the benchmark conducted within the European project AMARSi. As it is illustrated, our framework (with the name SEDS) is about twice as performant as the runner-up method. The graph is taken from (AMARSi Deliverable, 2011).

a stable linear DS. This was the first time that a statistical-based learning algorithm was suggested which can actually ensure global stability of nonlinear DS during the training phase.

Our framework can also be used in cluttered environment as it is endowed with the obstacle avoidance ability. This feature is crucial in order to provide a useful control policies when multiple static and moving objects exist in the robot's workspace. The proposed approach has a level of reactivity similar to existing local obstacle avoidance methods, while it ensures convergence to the target proper to global obstacle avoidance techniques.

The prominent features of the presented DS-based framework can be summarized as follows: 1) it allows a naive user to program robots to perform discrete movements using a natural means of demonstration, namely Imitation Learning, 2) it generates robot motions that are inherently robust to perturbations, and can instantly adapt to new situations in a dynamically changing environment, 3) it provides a means to perform collision avoidance in the presence of multiple static and moving obstacles, and most importantly at the kinematic level 4) it guarantees convergence of all trajectories to the target (if it is feasible).

SEDS framework is validated through various simulation and real-world robot experiments. Moreover, its applicability to different robotic systems are verified on a number of robotic platforms with varying degrees of freedom including: the humanoid robots HOAP-3 and iCub, and the robot arms KATANA, WAM, and LWR.

References

AMARSi Deliverable. (2011, February). *Exension of D4.1 with reaching benchmarks.* Retrieved from http://www.amarsi-project.eu/deliverables

Billard, A., Calinon, S., Dillmann, R., & Schaal, S. (2008). Handbook of Robotics. In (chap. Robot Programming by Demonstration). MIT Press.

Billard, A., & Hayes, G. (1999). DRAMA, a connectionist architecture for control and learning in autonomous robots. *Adaptive Behavior Journal*, *7*(1), 35–64.

Brock, O., & Khatib, O. (1999). Elastic Strips: A framework for integrated planning and execution. In *Proceedings of the International Symposium on Experimental Robotics* (Vol. 250, pp. 328–338). Springer Verlag.

Burns, B., & Brock, O. (2005). Toward Optimal Configuration Space Sampling. In *Proc. of Robotics: Science and Systems.*

Calinon, S. (2009). *Robot Programming by Demonstration: A Probabilistic Approach.* EPFL/CRC Press.

Calinon, S., Guenter, F., & Billard, A. (2007). On Learning, Representing and Generalizing a Task in a Humanoid Robot. *IEEE transactions on systems, man and cybernetics*, *37*(2), 286–298.

Coates, A., Abbeel, P., & Ng., A. Y. (2008). Learning for Control from Multiple Demonstrations. In *Proc. 25th Int. Conf. on Machine Learning (ICML)* (pp. 144–151).

Diankov, R., & Kuffner, J. (2007). Randomized Statistical Path Planning. In *Proc. of IEEE/RSJ Int. Conf. on Robots and Systems (IROS)* (pp. 1–6).

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA)* (pp. 1398–1403).

Kelso, J. (1995). *Dynamic patterns: The self-organization of brain and behavior.* Cambridge, MA: MIT Press.

Khansari-Zadeh, S. M. (2012). *A Dynamical System-based Approach to Modeling Stable Robot Control Policies via Imitation Learning.* Phd thesis, École Polytechnique Fédérale de Lausanne. Retrieved from http://infoscience.epfl.ch/record/182663 doi: 10.5075/epfl-thesis-5552

Kuniyoshi, Y., Inaba, M., & Inoue, H. (1989). Teaching by showing: Generating robot programs by visual observation of human performance. In *International Symposium of Industrial Robots* (pp. 119–126).

Lozano-Perez, T. (1983). Robot programming. In *proceedings of the IEEE* (Vol. 71, pp. 821–841).

Münch, S., Kreuziger, J., Kaiser, M., & Dillmann, R. (1994). Robot Programming by Demonstration (RPD) - Using Machine Learning and User Interaction Methods for the Development of Easy and Comfortable Robot Programming Systems. In *Proc. of the 24th Int. Symposium on Industrial Robots* (pp. 685–693).

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, *3*(6), 233–242.

Schaal, S., Ijspeert, A. J., & Billard, A. (2003). Computational Approaches to Motor Learning by Imitation. *Philosophical Transactions: Biological Sciences (The Royal Society)*(1431), 537–547.

Schaal, S., Kotosaka, S., & Sternad, D. (2000). Nonlinear dynamical systems as movement primitives. In *proceedings of the IEEE-RAS International Conference on Humanoid Robots.*

Selverston, A. I. (1980). Are central pattern generators understandable? *The Behavioral and Brain Sciences*, *3*, 555–571.

Toussaint, M. (2009). Robot Trajectory Optimization using Approximate Inference. In *Proc. 25th Int. Conf. on Machine Learning (ICML)* (pp. 1049–1056).

Wolpert, D., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, *11*, 1317–1329.