Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Actuation Strategy of a Virtual Skydiver Derived by Reinforcement Learning

Anna Clarke * **Per-Olof Gutman** **

* *Technion Autonomous Systems Program, Technion - Israel Institute of Technology, Haifa 32000, Israel (e-mail: anna.clarke@campus.technion.ac.il)*
** *Faculty of Civil and Environmental Engineering, Technion - Israel Institute of Technology, Haifa 32000, Israel (e-mail: peo@technion.ac.il).*

Abstract: An innovative approach of training motor skills involved in human body flight is proposed. Body flight is the art of maneuvering during the free fall stage of skydiving. The key idea is gradually constructing the movement patterns which are the combinations of body degrees-of-freedom that are activated synchronously and proportionally as a single unit, and turning this process into a coaching strategy. The proposed method is iterative: at each skill level an optimal movement pattern is constructed from the basic elements of the current movement repertoire. The free-fall maneuvers of each learning stage can be executed using any one of the basic elements. The construction has two stages: 1. tracking the desired maneuver while the body is actuated by each one of the basic patterns; 2. finding an optimal combination of these patterns to form a new way of body actuation. This hierarchical design resolves stage 2 by Reinforcement Learning with pure exploration and a minimal number of episodes. The method was tested in a Skydiver Simulator and resulted in deriving a movement pattern that showed a superior performance of the studied maneuver. The states and the reward of the Reinforcement Learning algorithm were converted into motor learning aids.

Keywords: autonomous systems, simulators, multiple degrees-of-freedom, reinforcement learning

1. INTRODUCTION

The human body has kinematic redundancy, enabling us to perform a variety of motor activities. It is known that some muscles, segments, and joints are activated in the body synchronously and proportionally, as a single unit. Combinations of such Degrees of Freedom (DOFs) are referred to as *movement patterns* (MPs). Motor learning is the process during which these MPs emerge (Schmidt and Wrisberg, 2008). First, the MPs are simple (coarse), providing just the basic functionality. As the learning continues, the MPs become more complex (fine), providing adaptation to perturbations and uncertainties, and improved performance (Tani et al., 2014). The objective of this research is to construct MPs involved in body flight - the free-fall stage of skydiving, making it possible to actuate an autonomous skydiver (Clarke and Gutman, 2017), and intended to aid training of novices: a vital and unresolved problem of the skydiving sport.

In sports biomechanics the research on MPs focuses on comparing MPs between experts and amateurs (see e.g. roller skiing in Gløersen et al. (2018), and ice hockey in Robbins et al. (2018)). This identifies what technique elements the amateurs need to change to improve their performance. In some sports simulations of kinematics and dynamics are used to find optimal values for the most important technique elements. For example, for V-style ski jumping the distance can be maximized by finding the optimal angle between the two skis and between the skis and the torso (Seo et al., 2004). Unfortunately, both of these approaches are not applicable for skydiving. In free-fall a great variety of maneuvers is executed by only slight changes in body posture. The same maneuver can be achieved utilizing completely different body DOFs; the MPs highly depend on individual body parameters and type of equipment. Consequently, every experienced skydiver has an individual style not transferable to others, involving multiple equally important DOFs.

Therefore, our method suggests to guide the trainees through the process of developing their individual optimal MPs. Initially, one trains several simple MPs, each involving 1-2 DOFs. Each MP is trained by performing a free-fall maneuver that is easily achievable by this set of MPs. Next, the trainees practice the same maneuver but free to use any combination of the learnt MPs. We hypothesize that the trainee's body will construct efficient MPs from the previously practiced elements. To accelerate this process, visual cues will be displayed in real time via an augmented reality interface (Clarke and Gutman, 2018). A question is which variables are informative and convertible to visual cues for improvement of the task performance. In order to identify such variables the above learning process is simulated in this work using the skydiving simulator developed in Clarke and Gutman (2017). The text is organized as follows: In section 2 six basic

MPs are defined and used to perform a given maneuver. In section 3 the performance is improved by combining the basic MPs using reinforcement learning tools. In section 4 an optimal MP for the maneuver under investigation is constructed. Implications of the results on the intended coaching strategy are given in section 5.

2. BASIC MOVEMENT REPERTOIRE

Six basic MPs for turning and side-sliding are proposed, see Fig. 1. The first two MPs involve the internal rotation DOF of the right and left shoulders, respectively. This causes the forearms to press on the airflow and turn to the opposite (of the engaged shoulder) direction. The third and fourth MPs involve the horizontal adduction DOF of the left and right shoulders. This movement causes the upper-arms to press on the airflow and turn/slide in the direction of the engaged shoulder. The fifth and sixth MPs involve the bending DOF of the hip and knee of the right and left legs. This movement will drop the knee down causing the thighs and shins to press on the airflow and turn in the opposite but slide in the same direction of the engaged leg. The first four MPs involve only one single DOF, and the last two leg MPs involve two DOFs each. The eigenvectors for these six basic MPs are:

	P_1	P_2	P_3	P_4	P_5	P_6
right forearm	1	0	0	0	0	0
left forearm	0	1	0	0	0	0
right upper arm	0	0	1	0	0	0
left upper arm	0	0	0	1	0	0
right hip	0	0	0	0	0.56	0
left hip	0	0	0	0	0	0.56
right leg	0	0	0	0	-0.83	0
left leg	0	0	0	0	0	-0.83

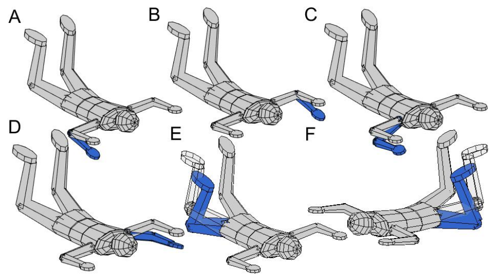

Fig. 1. Basic MPs: B,C,F cause right turn; A,D,E cause left turn. The MPs are organized in three sets according to the aerodynamic surface producing the turn: forearms (A,B), upper arms (C,D), and thighs (E,F).

The trajectory in Fig. 2 can be tracked using each one of these six MPs. However, the MPs in Fig. 1 suggest utilizing MPs A,D,E (B,C,F) turning left (right) only. The ergonomic reason is that the opposite action in the case of each MP would require a less natural movement, e.g. for the shoulder joint it is abduction instead of adduction, external instead of internal rotation, that might be hard for those who lack body flexibility. Thus, three sets of two MPs can be used to track the desired trajectory, each set including an MP for turning right and one for turning left. Each set is using a particular aerodynamic surface to produce turning: forearms (set A,B), upper arms (set C,D), and thighs (set E,F).

For the tracking purpose three PD controllers were designed:

$$u = K_P \cdot error + K_D \cdot \dot{error}$$
$$error = atan(\frac{Xpath(t_{LA}) - X_{pred}}{Ypath(t_{LA}) - Y_{pred}}) - atan(\frac{V_x}{V_y}) \quad (1)$$

where V_x, V_y is the current inertial horizontal velocity of the skydiver, $Xpath, Ypath$ is the desired trajectory, X_{pred}, Y_{pred} is the predicted position of the skydiver while the prediction time, and the look ahead time t_{LA}, are parameters conveying the delay and reaction times. The physical meaning of $error$ is the disparity between the current direction of the skydiver and the required direction, assuming that: 1. The same direction will be kept during the 'prediction' time; 2. It is needed to move towards a point on the desired path located ahead of the skydiver. The look-ahead time represents the time it will take to converge to the desired direction. As found in simulations, typical parameters for skydiving in a belly-to-earth stable pose are: prediction time 1.5 s, and look-ahead time 4.5 s. Large delay and slow reaction time are the reason for using the derivative part $K_D \cdot \dot{error}$.

The delay conveys the physics of skydiving: airflow around the moved limb is rearranged in a new pattern, forming new areas of high and low pressure. This changes the aerodynamic forces and moments that induce linear and angular accelerations, which sum up, if the limb retains its new position, generating linear and angular velocities. In turn, these velocities cause the desired change in position and orientation.

The controllers parameters were tuned in simulation:

$$K_{Parms} = 0.12 \qquad K_{Darms} = 0.8$$
$$K_{Plegs} = 0.3 \qquad K_{Dlegs} = 1.7 \quad (2)$$

which means that all MPs associated with the arms can be controller by using K_{Parms}, K_{Darms}, and MPs associated with the legs - by K_{Plegs}, K_{Dlegs}. Notice that when the controller command is positive (corresponds to a left turn) one of the left-turning MPs is enabled, and for negative commands - one of the right turning MPs is enabled (see Fig. 3). Also, it is assumed that each DOF has a limit on the rate of change of its value: 15 [deg/sec] for DOFs associated with the arms, and 45 [deg/sec] for DOFs associated with the legs.

It is possible to track the trajectory shown in Fig. 2 by the means of each one of the three options: using forearms, upper-arms, or thighs as an aerodynamic surface. However, in each case the position errors and control effort are significant, see Fig. 2, 3. In the next section a learning algorithm is applied in order to combine the basic MPs for improving trajectory tracking accuracy and reducing the control effort.

3. REINFORCEMENT LEARNING

It is desired to learn which combination of the primitive MPs engaged at each instant of time will provide the most accurate trajectory tracking. This problem includes two hierarchical tasks, that we choose to solve separately. The first, high level task is deciding which MPs to engage and quantifying the desired relative effort corresponding to each chosen MP. For example, do 90% of the work by the

Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

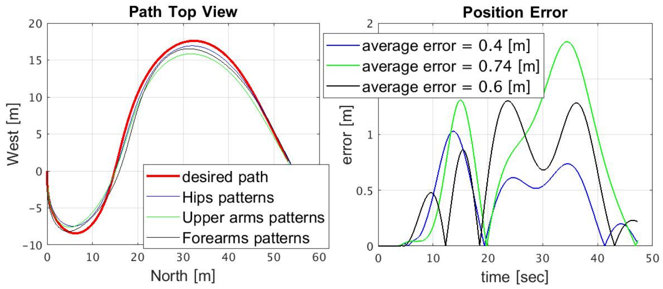

Fig. 2. Simulation of following the desired path by the means of three choices of aerodynamic surfaces: forearms, upper arms, and thighs.

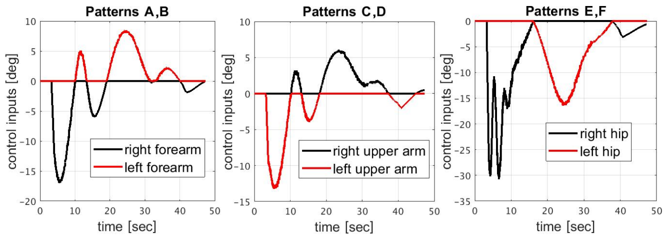

Fig. 3. Control inputs during simulation of following the desired path by the means of three choices of aerodynamic surfaces.

means of hips and 10% of the work by the means of upper-arms. Thus, the high level task is providing the solution to the human kinematic redundancy.

The second, low level task, is tracking the desired trajectory given a choice of MPs. This is a classical control task that involves designing a set of controllers for each of the MPs in the movement repertoire. At each instant of time the overall controller command is a posture composed of all available MPs:

$$posture = NeutralPosture + \sum_{i=1}^{N} w_i \cdot u_i \cdot P_i$$
$$u_i = \begin{cases} u & \text{if } u \geq 0 \text{ and } i = 1, 4, 5, \text{ or} \\ & \text{if } u < 0 \text{ and } i = 2, 3, 6 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $N = 6$ is the number of MPs, w_i is the weight of each MP computed by the high level algorithm, u is the control command computed according to Eqs. (1), (2), P_i is the eigenvector corresponding to each MP, and $NeutralPosture$ is the default posture when no turning MPs are activated. The default posture, in our case, includes extending the legs in order to generate forward speed. Thus, applying the default posture will cause the skydiver to fly forward in a straight line, and applying the turning MPs will enable her to follow the curves of the desired trajectory.

The hierarchical design of the overall system allows to solve each task by the tools best suited and specifically developed for these tasks. Choosing the way of actuating the body is resolved by Reinforcement Learning, and tracking the desired trajectory given the chosen actuators is resolved by Control Theory. Given the plant model (see Clarke and Gutman (2017)) it is possible to determine the ranges of w_i such that the weighted average of commands computed by several linear controllers will provide a stable system with desired specifications. The *actions* of the Reinforcement Learning algorithm, i.e. options for w_i combinations, can thus be chosen form these ranges. This way all actions will result in a stable and reasonably accurate trajectory tracking. It remains to learn which options work better than others depending on the situation (*state*).

3.1 Problem Formulation

The main challenge in solving realistic problems by reinforcement learning methods is formulating the problem in terms of a set of actions and a set of states. In our case, the variables defining the state will be the best candidates for becoming the cues of the training system. The reason is that a successful reinforcement learning process will indicate that the chosen states contain all the information required for describing the situation at each instant of time. Variables used for calculation of the *reward* should also be used as training cues as they contain information required for choosing the best action in each situation. A

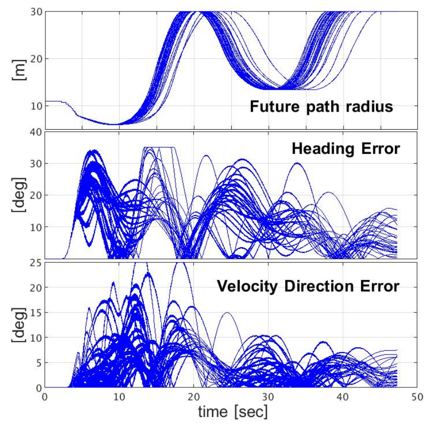

Fig. 4. States obtained during simulations of 30 exploration episodes

conventional application of reinforcement learning assumes simulating the task multiple times and processing hundreds of episodes, while continuously updating the *value function*. The most frequently adopted *policy* is choosing the action that maximizes the current estimate of the value function, and once in a while choosing a random action for exploration purposes. In our case, however, it is desired to complete the learning process after a small amount of trials, as each trial means a new parachute jump. Current training methods actually require hundreds of jumps in order to acquire basic skills. The training program we propose aims to reduce this amount by an order of magnitude. Thus, the learning scheme described below uses pure (100%) exploration, meaning that the number of episodes will be equal to the number of actions, while the policy is choosing a new action for each new episode. An episode is flying along the desired trajectory from its initial to final point. The pure exploration is possible due to the hierarchical structure of our solution: during the phase of learning how to combine available MPs, the control problem has already been resolved.

Actions are represented by a set of weights w_i (Eq. 3). Notice that the weights for six MPs defined in Fig. 1 are defined by three parameters, since these MPs are engaged in pairs: (A,B), (C,D), and (E,F). The following 30 actions are chosen:

$$\begin{array}{c|cccccccccc} w_1 & 1 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & 0 & 0 \\ w_2 & 0 & 1 & 0 & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{2}{3} \\ w_3 & 0 & 0 & 1 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{array}$$

$$\begin{array}{c|cccccccccc} w_1 & -\frac{2}{3} & -\frac{2}{3} & -\frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 \\ w_2 & \frac{1}{3} & \frac{2}{3} & 1 & \frac{4}{3} & 0 & \frac{1}{3} & \frac{2}{3} & 1 & \frac{4}{3} & \frac{4}{3} \\ w_3 & \frac{4}{3} & 1 & \frac{2}{3} & \frac{1}{3} & \frac{4}{3} & 1 & \frac{2}{3} & \frac{1}{3} & 0 & -\frac{1}{3} \end{array}$$

$$\begin{array}{c|cccccccccc} w_1 & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & 1 & 1 & 1 & \frac{4}{3} & \frac{4}{3} \\ w_2 & -\frac{1}{3} & 1 & -\frac{2}{3} & -\frac{1}{3} & \frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & 0 \\ w_3 & 1 & -\frac{1}{3} & 1 & \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} & \frac{1}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} \end{array}$$

Out of many variables that are related to the current position, orientation, velocity, desired path and motion the following three were found as most effective states:

(1) The mean local radius of the upcoming path segment. The segment starts at the desired path point closest to the current skydiver's position, and it's length is 7.5 [m]. Given three consecutive segment points (x_1, y_1), (x_2, y_2), and (x_3, y_3) spaced at 0.1 [m], the local radius R is computed as follows:

$$\begin{aligned} m1 &= (y_2 - y_1)/(x2 - x1) \\ m2 &= (y_3 - y_2)/(x3 - x2) \\ \phi &= atan\left(\frac{m1 - m2}{1 + m1 \cdot m2}\right) \\ R &= \left|\frac{0.1}{sin(\phi)}\right| \end{aligned} \tag{4}$$

(2) Disparity between the desired and actual skydiver's heading. The desired heading angle Ψ_{des} is assigned to each path point when the desired path is constructed. For the heading error calculation the desired heading associated with the path point closest to the current skydiver's position is taken. The actual skydiver's heading Ψ is continuously updated by the Skydiver Simulator. The disparity H is computed as:

$$H = \Psi_{desired} - \Psi \tag{5}$$

(3) Disparity between the desired and actual motion direction. The motion direction is computed from two consecutive desired and actual path points, respectively. The first path point (X_{des}, Y_{des}) is the one closest to the current skydiver's position (X, Y). The second desired (Xp_{des}, Yp_{des}) and actual (Xp, Yp) points are the preceding ones. The disparity D is computed as:

$$D = atan\left(\frac{Y_{des} - Yp_{des}}{X_{des} - Xp_{des}}\right) - atan\left(\frac{Y - Yp}{X - Xp}\right) \tag{6}$$

Fig. 4 shows these three states during simulations of 30 exploration episodes. Notice, that all states are given in their absolute values and are trimmed to certain maximal values. This is preparation for states discretization, summarized in the table below, and resulting in total of 120 states. The range of each of the continuous variables in Fig. 4 is divided into a small number of tiles (n):

variable	n	tiles centers	width
R [m]	4	[3.5, 10.5, 17.5, 24.5]	7
H [deg]	6	[2.5, 7.5, 12.5, 17.5, 22.5, 27.5]	5
D [deg]	5	[2.5, 7.5, 12.5, 17.5, 22.5]	5

The states discretization is a step towards developing a simple user interface, as these states will be converted to visual cues.

Expression for the reward that produced best results consists of two parts: a punishment proportional to position error squared, and a constant reward for keeping position error below the threshold of 5 [cm]:

$$Reward = \begin{cases} 3 - (PosE)^2 & , \quad |PosE| < 0.05 \\ -(PosE)^2 & , \quad otherwise \end{cases} \tag{7}$$

where $PosE$ is position error, calculated as the shortest Euclidean distance from the current skydiver's position to the desired path.

Out of many suitable reinforcement learning algorithms (see Sutton and Barto (2018) for a review) State-Action-Reward-State-Action (SARSA) was chosen. The Q-function or action-value $Q(s, a)$ is the long-term return (as opposed to the short-term *Reward*) of the current state s, taking action a under the current policy. The Q-function at step $k - 1$ is updated using the *Reward* and Q-function at step k as follows:

$$\begin{aligned} Q(s_{k-1}, a_{k-1}) &= Q(s_{k-1}, a_{k-1}) + \alpha d_{k-1} \\ d_{k-1} &= Reward_k + \gamma Q(s_k, a_k) - Q(s_{k-1}, a_{k-1}) \end{aligned} \tag{8}$$

where α is a learning rate and γ is a discounting factor. The objective is to learn the Q-function (let it converge) and then at each step the action that maximizes the Q-function given the current state can be chosen. It is possible to implement the SARSA algorithm in many variations. In our case, it is advantageous to use SARSA with Linear Function Approximation (LFA), when the Q-function is represented by a linear combination of numerical features $F_1(s, a)$, .. $F_n(s, a)$ of the state and the action:

$$Q_\eta(s, a) = \sum_{i=1}^{n} \eta_i F_i(s, a) \tag{9}$$

where the weights η_i have to be learned. We define $n = N_{states} N_{actions} = 120 \cdot 30 = 3600$ feature functions, while most of them are zero at any given moment. The computation of $F(s_i, a_j)$ for state $i = 1, \ldots I, \ldots N_{states}$ and action $j = 1, \ldots J, \ldots N_{actions}$, where I, J are the indexes of the state and action at the current moment, is shown in (10), (11).

$$F(s_i, a_j) = \begin{cases} e^{-\frac{1}{2}(\Delta_r + \Delta_h + \Delta_d)} & , \quad if\ j = J \\ 0 & , \quad otherwise \end{cases} \tag{10}$$

$$\begin{aligned} \Delta_r &= \frac{(R - R_{tiles}(i1))^2}{dR^2} r \\ \Delta_h &= \frac{(H - H_{tiles}(i2))^2}{dH^2} h \\ \Delta_d &= \frac{(D - D_{tiles}(i3))^2}{dD^2} d \end{aligned} \tag{11}$$

where $i = (i3 - 1) * N_{R_{tiles}} * N_{H_{tiles}} + (i2 - 1) * N_{R_{tiles}} + i1$ is the index of state s, and $i1, i2, i3$ are the indexes of its three discrete components $R_{tiles}, H_{tiles}, D_{tiles}$, while the current continuous state is R, H, D; dR, dH, dD are the tiles' widths, and r, h, d are the tuning parameters defining the width in each direction of the radial feature function. We used more 'narrow' distributions during exploration $r = 4,\ h = 4,\ d = 4$ than during exploitation $r = 0.7,\ h = 1,\ d = 1$. This produced smoother control inputs and less control effort. Radial as opposed to binary feature functions allow a faster convergence of the Q-function, and are similar to the fuzzification stage during fuzzy control design.

Since in our case the number of learning episodes is small and it is desired to increase the efficiency of learning, it is advantageous to combine the SARSA algorithm with an eligibility trace. Eligibility traces are a basic mechanism for temporal credit assignment. When the *Reward* is computed, only the eligible states-actions pairs are assigned credit or blame for the error, which is weighted according to the eligibility trace. Normally, the state-action pair of the previous simulation step s_{k-1}, a_{k-1} is considered eligible for the update with the weight of 1, as stated by Eq. 8, and the state-action pairs visited during earlier steps receive fading-out weights. In our case we can utilize our knowledge of the system reaction time: The eligible state-actions pairs will be the ones visited $te = [0.5833,\ 0.5667,\ 0.55,\ 0.5333]$ seconds ago with weights $We = [1,\ 0.7,\ 0.4,\ 0.1]$ accordingly.

Due to the small number of episodes we introduce one more modification to the conventional SARSA algorithm: scaling the updates of the Q-function according to the total number of visits to each state-action pair. Conventional reinforcement learning algorithms assume that the total number of visits to each state-action pair will be sufficient for convergence (infinite for all engineering purposes). In our case, however, it is known that the number of visits to state-action pairs will be small and unequal, i.e. some parts of the Q-function will be updated more often. Scaling introduced below allows to exploit the Q-function that is obtained from a small number of learning episodes before it actually converges.

In summary, our modified version of the SARSA algorithm can be formulated as follows:

Initialization:

$$\begin{aligned} \underline{\eta} &= zeros(N_{states}N_{actions}, 1) \\ vsa &= zeros(N_{states}N_{actions}, 1) \\ \gamma &= \beta = 1 \end{aligned} \tag{12}$$

where vsa is the number of visits to an (s, a) pair. *Simulation Step:*

Compute the vector of features $\underline{F}$ for the current state and action (Eq. 11), the immediate *Reward* (Eq. 7), and the update of η for each of the 4 eligible vectors of features $\underline{F}_{te(i)}$, $i = 1, 2, 3, 4$:

$$\begin{aligned} \delta &= Reward + \gamma \underline{F}^T \underline{\eta} - \underline{F}^T_{te(i)} \underline{\eta} \\ vsa_{Prev} &= vsa \\ vsa &= vsa + \underline{F}_{te(i)} We(i) \end{aligned} \tag{13}$$

for each nonzero element k of vsa:

$$\eta(k) = \eta(k) \frac{vsa_{Prev}(k)}{vsa(k)} + \beta\delta \frac{\underline{F}_{te(i)}(k) We(i)}{vsa(k)} \tag{14}$$

The number of Simulation Steps is the number of episodes (30) multiplied by the length of each episode (2820 steps = 47 sec). After this learning process the obtained $\underline{w}$ can be used to choose at each instant of time the optimal action:

$$a = \underset{a}{argmax}\ \underline{F}(s, a)^T \underline{\eta} \tag{15}$$

where the current state s is composed from its three components R, H, D and the elements of $\underline{\mathrm{F}}$(s,a) are computed according to (11).

3.2 Results

A linear combination of three controllers weighted according to the policy stated in (15) produced fruitful results: the trajectory was tracked with the mean position square error below 0.03 [m], and the generated limbs movements were smooth engaging at every instant of time several joints, see Fig. 5, 6.

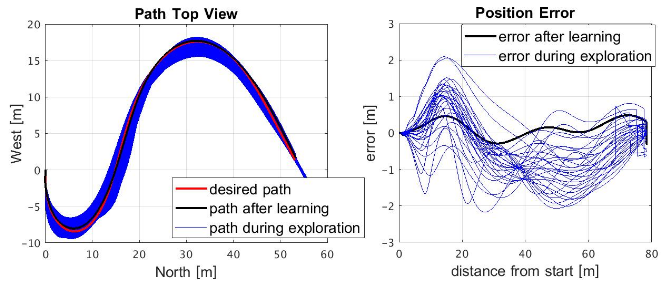

Fig. 5. Trajectory tracking during simulations of 30 exploration episodes and one exploitation run

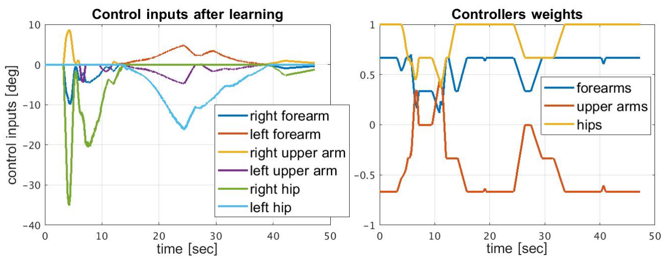

Fig. 6. Controller weights and limbs movements during the simulation exploiting the learned Q-function

It appears that the learning process resulted in finding an optimal combination of synchronously activated joints in order to solve the desired task - accurately track a particular trajectory. This, however, is the definition of an MP. In other words, it seems that an MP efficient for the given task was constructed in the two design steps described above: 1. designing three controllers that involve three different sets of joints; 2. learning the optimal linear combination of the weights for these controllers.

If the above hypothesis is correct it will be possible to extract this MP from simulations and replace the three controllers and their weighting policy by **one controller using the obtained MP for body actuation**. We also expect that in the later case the tracking accuracy will be better especially if the task is altered. The reason is that the learned Q-function provides an optimal policy only for the original task. However, if this knowledge is implemented in one MP, a controller driving its input signal will be able to deal with task variations and disturbances.

In the next section we present a simulation of a slightly modified task from which the learned MP is extracted and the above hypothesis is non-falsified.

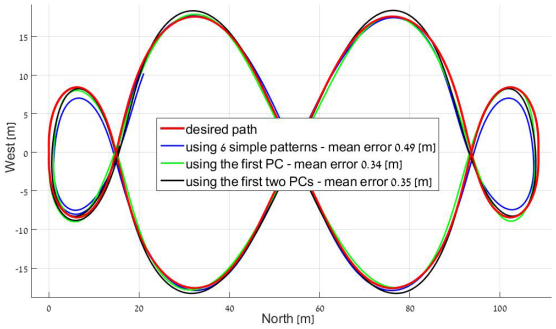

Fig. 7. Comparison of tracking the desired trajectory by different control and body actuation options

4. THE OPTIMAL MOVEMENT PATTERN

Fig. 7 shows the modified desired trajectory. The original trajectory was tracked from the stopped position while slowly accelerating, hence the tight curve in the beginning was tracked at a slower speed than the rest of the path. This time, the skydiver will arrive to the tight curve segments at a higher speed. The Principal Component Analysis (PCA) of following this trajectory exploiting the learned policy reveals two MPs:
$PC1 = [-0.11, -0.1, 0.07, 0.06, -0.4, 0.37, 0.6, -0.55]^T$
$PC2 = [0.11, -0.11, -0.06, 0.07, 0.37, 0.4, -0.55, -0.6]^T$
where the order of DOFs is the same as in P_1-P_6. Apparently, the second MP was formed due to the fact that the three original controllers used different limbs for implementing positive and negative input signals. The reason was ergonomic: it is easier to drop the left hip to turn right rather than trying to lift the right hip, which is less natural for the human body, even though it allows the use of the same limb for positive and negative inputs.

This way, we expect that it is possible to track the given trajectory by actuating the body with just the first Principle Component (PC1). In this case the limbs inputs will include joint angles hard for execution. This can be eliminated by introducing the second MP (PC2), which is expected to be driven by the same control signal as the first one (PC1), but taking into account its sign.

These assumptions were non-falsified by designing controllers for both options (Eq. 16) and testing them in simulation.

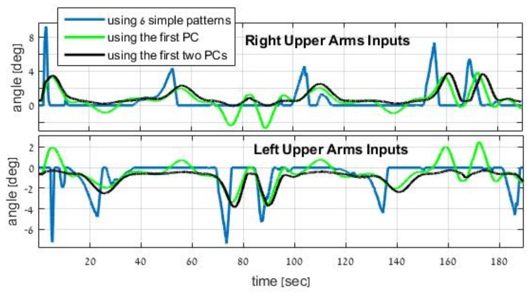

Fig. 8. Comparison of upper arms angles during tracking the desired trajectory using different control options

$$\begin{aligned} pose_1 &= M \ + \ u \cdot PC1 \\ pose_2 &= M \ + \ u \cdot PC1 \ - \ 0.7 \cdot u \cdot sign(u) \cdot PC2 \end{aligned} \tag{16}$$

where u is computed as in Eq. (1) with $[K_P, \ K_D] = [0.37, 1.85]$ when only PC1 is used, and $[0.27, 1.3]$ for PC1-PC2 combination, and M is the mean of all recorded postures computed by PCA.

The results confirm all the previously discussed hypothesis. From the top view shown in Fig. 7 we can see that, as expected, the tight curves are followed not so accurately with an increased speed when the three controllers with the optimal weights are used. However, the accuracy is improved when using only one controller with PC1 or with a combination PC1-PC2. The limbs movements are smoother and the control effort is smaller when one controller is used, see example in Fig. 8. It is also clearly seen that at some moments utilizing PC2 provides a more ergonomic actuation of the body.

5. CONCLUSION

The main result of this work is justification for the suggested training method. The design process shown in sections 2-4 resembles the natural learning of a motor skill: First, we learn some body movements matching our current skill level. Then, we perform the activity trying out and combining these movements until a new MP emerges. In the presented simulation a new MP, better suited for tracking a given trajectory, has emerged from actuating the body by a weighted combination of primitive MPs, while the weights were found by a reinforecement learning algorithm. Its states and reward can thus be used as visual cues in the training system which we aim to develop next.

REFERENCES

Clarke, A. and Gutman, P.O. (2017). Modelling and control of a virtual skydiver. *IFAC-PapersOnLine*, 50(1), 369–374.

Clarke, A. and Gutman, P.O. (2018). Computerized methods and systems for motor skill training. US Patent App. 15/658,548.

Gløersen, Ø., Myklebust, H., Hallén, J., and Federolf, P. (2018). Technique analysis in elite athletes using principal component analysis. *Journal of sports sciences*, 36(2), 229–237.

Robbins, S.M., Renaud, P.J., and Pearsall, D.J. (2018). Principal component analysis identifies differences in ice hockey skating stride between high-and low-calibre players. *Sports biomechanics*, 1–19.

Schmidt, R.A. and Wrisberg, C.A. (2008). *Motor learning and performance: A situation-based learning approach.* Human Kinetics.

Seo, K., Murakami, M., and Yoshida, K. (2004). Optimal flight technique for v-style ski jumping. *Sports Engineering*, 7(2), 97–103.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement learning: An introduction.* MIT press.

Tani, G., Corrêa, U.C., Basso, L., Benda, R.N., Ugrinowitsch, H., and Choshi, K. (2014). An adaptive process model of motor learning: Insights for the teaching of motor skills. *Nonlinear Dynamics Psychology and Life Sciences*, 18, 47–65.