Optimize and coordinate multiple DMPs under constraints to achieve a collaborative manipulation task

Ali H. Kordia[1] Francisco S. Melo[2]

***Abstract*— This paper addresses a significant challenge in achieving collaborative tasks; how can a robot or multiple robots, endowed with a library of pre-learned primitive movements, generate multiple simultaneous *coordinated* robotic movements, adapting and optimizing those in the library, to complete one collaborative task? This work can thus be seen as a follow-up to the work with a motion presented as a dynamic movement primitive (DMP) that now considers collaborative tasks and the existence of multiple robots/manipulators. Specifically, we start with a simple task using one DMP and extend it to accommodate the coordinated execution of multiple DMPs in robots with multiple manipulators or—alternatively—multiple robots with a single manipulator. We investigate mechanisms to jointly optimize multiple DMPs to perform one task in a coordinated fashion. The joint trajectory is built from the initial DMPs learned for a single manipulator, and its optimization must comply with task-specific constraints. We illustrate the application of our approach both in a simulated environment and in a simulated and real Baxter robot.**

I. INTRODUCTION

In collaborative tasks, team members work together to accomplish a particular goal. Such teams may be composed of humans and robots, possibly more than one robot or more than one human. Collaboration towards the common goal requires coordinated actions between the members according to the stages of task completion and their skills. Usually, the skills of team members are complementary to each other to accomplish the required task [1].

It is crucial to clarify that the scope of the problems addressed in this paper does not encompass tasks that can be decomposed into sub-tasks, each capable of independent execution by individual robots, as is frequently encountered in multi-robot scenarios [2]. Our focus, instead, revolves around tasks necessitating the simultaneous execution of coordinated and synchronized movements, all the while accounting for environmental conditions and constraints. This scenario frequently arises in domains such as robotics [3] and gaming [4].

Several approaches exist that address the problem of coordinating multiple motions. For example, Stavridis et al [5] addresses the problem of assembling two parts where—instead of using a manipulator just to hold one part steady and the second to assemble the second part—both manipulators move to render the assembly easier. The movement of the two manipulators is described using a dynamical system; the trajectories of the system are then adjusted to ensure that the task constraints are verified. Constraints include the desired position for the parts to be assembled, as well as the obstacles in the environment that should be avoided. There are two key differences in our work. First, an initial movement for both manipulators is provided upfront, which is then optimized using a *hierarchical quadratic programming approach*. In our work, we depart from a single-manipulator movement that is then used to construct the motion of *both* manipulators; additionally, following the approach in work [6], We have opted for Black-Box Optimization (BBO) over a quadratic programming optimizer, affording us the advantage of reduced model complexity. Unlike specific quadratic programming methods that necessitate linear or quadratic approximations of the objective function, BBO techniques operate directly on the black-box objective function without simplifications or approximations.

Gams et al [7] use *iterative learning control* (ILC) to learn a coupling term that limits two DMPs to enforce coordination. Given two DMPs, Gams et al [7] define one DMP as the "leader" and the other as the "follower". The leader constrains the follower to ensure the successful completion of the task. This requires that the leader DMP be already provided in such a way that successful task completion is possible. In contrast, our method optimizes both movements simultaneously to comply with the task constraints.

Nemec et al [8] propose an approach for bimanual robots that engage in a cooperative task with humans. The motion for the robot is taught by demonstration and stored as a (bi-manual) DMP. At execution time, the robot stiffness is initially set to allow the robot to comply with the human motion—i.e., the human "retains" control of the task. With successive executions, the robot gradually "takes control" from the human by adjusting the stiffness of the movement execution. They propose *speed-scaled DMPs* as an extension to DMPs that allows for compliant execution.

In collaborative robot learning tasks involving multiple manipulators, generating coordinated and synchronized movements is essential for achieving effective teamwork, especially when robots have different structures and capabilities or operate within the same robot.

This paper presents a novel approach that employs Dynamic Movement Primitives (DMPs) to model robot motions in collaborative tasks. Assuming access to a library of general movements, our approach enhances coordination by addressing challenges related to the robots' physical structures, task environments, obstacles, and the task itself. Through a simple task demonstration incorporating environmental and task

[1]Ali Hani Kordia is with the SnT Centre, the University of Luxembourg, Luxembourg, and with INESC-ID and Instituto Superior Técnico, University of Lisbon, Portugal; ali.kordia@tecnico.ulisboa.pt.

[2]Francisco S. Melo is with INESC-ID and with Instituto Superior Técnico, University of Lisbon, Portugal; fmelo@inesc-id.pt.

constraints, our framework generates synchronized DMPs that enable robots to overcome these complexities. Our research integrates task—and case-level coordination, linking improvements across simultaneous task paths while accounting for the demands of collaboration, environmental factors, and the robots' physical characteristics. We validate our approach using two manipulators—Baxter robot arms—to simplify and demonstrate its effectiveness in improving submovement synchronization and overall task performance.

The rest of the paper is organized as follows. Section II provides an overview of DMPs and the optimization approach adopted (CMA-ES). Section III describes our approach, and Section IV illustrates its application in several tasks. Section V concludes.

II. Background

A. Dynamic movement primitives

A *dynamic movement primitive* [9], [10] is a representation of a smooth trajectory using a stable dynamical system. Specifically, a one-dimensional DMP corresponds to the nonlinear system of equations

$$\ddot{y}(t) = \alpha_y(\beta_y(y_{\text{ref}} - y(t)) - \dot{y}(t)) + f(x(t)), \tag{1a}$$

$$\dot{x}(t) = \alpha_x x(t), \tag{1b}$$

where $y(t)$ is the state of the DMP and $x(t)$ is the so-called *phase variable*. In the formulation above, (1a) is known as the *transformed system*, while (1b) is known as the *canonical system*. Under a suitable choice of α_y, β_y and α_x, $y(t) \to y_{\text{ref}}$ and $\dot{y}(t) \to 0$ as $x(t) \to 0$.

The particular shape of the trajectory by which $y(t) \to y_{\text{ref}}$ depends on the force function f, which is usually represented as the linear combination of a set of *base functions* Ψ_n,

$$f_w(x) = \frac{\sum_{n=1}^{N} w_n \Psi_n(x)}{\sum_{n=1}^{N} \Psi_n(x)} x. \tag{2}$$

The weights $w_n, n = 1, \ldots, N$ can now be adjusted to yield a smooth trajectory between $y(0)$ and y_{ref}.

In the context of robot motion, each degree of freedom of the robot is controlled by an individual DMP, and synchronization is achieved through a shared canonical system. Using a canonical system instead of a synchronized clock ensures that the resulting system is time-homogeneous, bringing advantages in terms of control. For more details [6][9].

B. Covariance matrix adaptation evolution strategy

In our approach, we adopt a constrained version of a well-established black-box optimizer known as *covariance matrix adaptation evolutionary strategy*, or CMA-ES [11]. CMA-ES can be used to optimize a function that is not known in advance, but which can be evaluated at any desired query points. CMA-ES belongs to a large family of evolutionary strategy algorithms that seek to optimize the objective function while minimizing the number of required function evaluations [12], [13], [14].

For the sake of concreteness, let us suppose that we are interested in maximizing a real-valued objective function J defined over R^N. At each iteration i, CMA-ES proceeds by:

1) **Sampling:**
   - Sample K points $\{w_{i,k}, k = 1, \ldots, K\}$ from a multivariate Gaussian, $\text{Normal}(\mu_i, \sigma^2\Sigma_i)$, with $\mu_i \in R^N$ and $\Sigma_i \in R^{N \times N}$;
2) **Evaluating:**
   - Compute $J(w_{i,k})$ for $k = 1, \ldots, K$;
   - Select the best K_0 samples from $\{w_{i,k}, k = 1, \ldots, K\}$;
3) **Adaptating:**
   - Update the mean μ_i to the weighted average of the best K_0 samples from $\{w_{i,k}, k = 1, \ldots, K\}$ selected in the previous step;
   - Update the covariance matrix Σ_i to increase the likelihood of previously successful search steps.

Pseudo-code for CMA-ES is provided in Algorithm 1. It receives, as parameters, the number of samples per generation, K, the number of samples used for updating the mean, K_0, with $K_0 < K$ (typically, $K_0 \leq K/2$), and a number

Algorithm 1 CMA-ES algorithm [15].

Require: $K, K_0, c_\sigma, c_c, c_{\text{cov}}, d_\sigma, \{\alpha_k, k = 1, \ldots, K_0\}$
Require: $\mu_0 \in R^N, \sigma_0 \in R_+$
Initialize $\Sigma_0 = I$, $p_c = p_\sigma = 0$, $i = 0$
while not terminate do
  for $k = 1, \ldots, K$ **do**
    Sample $w_k \sim \text{Normal}(\mu_i, \sigma_i^2\Sigma_i)$
    Compute $J(w_k)$
  Sort $w_1, \ldots, w_K$ according to $J(w_k)$, with $w_{k:K}$ denoting the kth best sample out of K
  Set

$$\mu_{i+1} = \mu_i + \sum_{k=1}^{K_0} \alpha_k (w_{k:K} - \mu_i)$$

  Set

$$p_\sigma = (1 - c_\sigma)p_\sigma + \frac{\rho}{\sigma_i}\sqrt{1 - (1 - c_\sigma)^2}\,\Sigma_i^{-\frac{1}{2}}(\mu_{i+1} - \mu_i),$$

  where $1/\rho = \sqrt{\sum_{k=1}^{K_0} \alpha_k^2}$
  Set

$$\sigma_{i+1} = \sigma_i \exp\left\{\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{\mathbb{E}\|\text{Normal}(0, I)\|} - 1\right)\right\}$$

  Set $p_c = (1 - c_c)p_c + \frac{\rho}{\sigma_i}\sqrt{1 - (1 - c_c)^2}(\mu_{i+1} - \mu_i)$
  Set

$$\Sigma_{i+1} = (1 - c_\mu - c_{\text{cov}})\Sigma_i + \frac{c_{\text{cov}}}{\rho^2} p_c p_c^T + \frac{c_\mu}{\sigma_i^2} \sum_{k=1}^{K_0} \alpha_k (w_{k:K} - \mu_i)(w_{k:K} - \mu_i)^T \tag{3}$$

  Set $i = i + 1$
return μ_i

of constants used in the covariance matrix update (c_σ, c_c, c_{cov} and d_σ). It also receives as parameters the weights $\alpha_1, \ldots, \alpha_{K_0}$ used in the update of the mean.

As outlined above, the mean is updated as a convex combination of the K_0 best samples (lines 6 and 7). The covariance matrix update (lines 10 and 11), on the other hand, comprises several main components:

- The weighted covariance of the best samples—corresponding to the last term in (3); this term focuses the distribution towards more promising regions of the search space.
- A *bootstrapping term*—corresponding to the first term in (3); this term helps to ensure that Σ_{i+1} retains rank N, by using information from previous estimates, Σ_i.
- A *cumulation term*—corresponding to the second term in (3); p_c tracks successive updates to the mean, and is, therefore, called an *evolution path*. The cumulation term is a rank 1 term that accounts for the general "direction" that the algorithm has moved.

Besides the covariance matrix Σ_i, CMA-ES also uses a *step-size* σ_i that is updated as the algorithm progresses (lines 8 and 9). The step size is used, among other things, to avoid the samples collapsing too soon. Its update makes use of a conjugate evolution path, p_σ, which is compared with the direction-independent length, $\mathbb{E}\|\text{Normal}(0, I)\|$. Several recent works have further extended its applicability and properties [16], [17], [18].

III. Coordinated Trajectory Optimization

The pipeline for the proposed approach is illustrated in Fig. 1. Following the general approach in [6], we collect a (single manipulator) demonstration from a human user that is then used to build a DMP (or a library thereof). Then, given the specification of a task (goal, constraints), we use the movements in the library to generate and optimize a *joint DMP*, providing the desired trajectory for the multiple manipulators. The environment in which the task is carried on (obstacles, environment layout) may differ from that where the demonstration was originally provided, and we leave it to the optimizer to adjust the DMPs in accordance.

A. Constrained CMA-ES

We adopt, as the optimizer, a variation of CMA-ES known as *constrained CMA-ES*, or CCMA-ES. This method is an extension to the CMA-ES method discussed before in section II for constrained optimization problems.

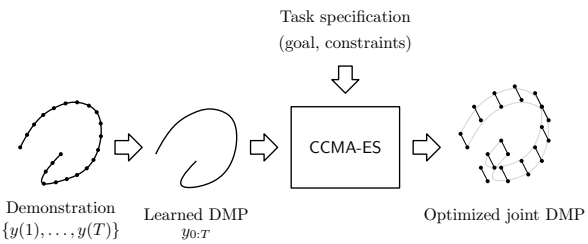


Fig. 1: Pipeline for the proposed approach.

We formulate the problem of coordinated motion as a constrained optimization problem, given a target position y^* and a set of M obstacles, $\{o_1, \ldots, o_M\}$, we define the objective function for evaluating the black-box output for the CMA-ES samples:

$$J = \sum_{t=0}^{T} \Bigg[c_{\text{goal}} \|y(t) - y^*\|^2 + c_{\text{vel}} \|\dot{y}(t)\|^2 + c_{\text{obst}} \sum_{m=1}^{M} \mathbb{I}(y(t) \in o_m) \Bigg], \tag{4}$$

where c_{goal}, c_{vel} and c_{obst} weight the relative importance of reaching the goal, avoiding large velocities and avoid obstacles, and $\mathbb{I}(y(t) \in o_n)$ is a binary value that indicates whether, at time step t, the trajectory $y(t)$ hit obstacle o_n.

Where this objective function is defined by the task and environment (as the work [6], and the constraints ensure coordination between the multiple DMPs being optimized. While different approaches exist for constrained optimization of DMPs [19], we follow up on an approach from [6] and adopt CCMA-ES.

Suppose that we want to optimize a function $J : R^P \to R$, while ensuring that

$$h_j(w) \leq 0, \quad j \in \{1, \ldots, m\},$$

for some set of functions $h_j : R^P \to R$. The above formulation is quite general in that it also allows for the consideration of equality constraints of the form $g(w) = 0$, simply by requiring, simultaneously, that $g(w) \geq 0$ and $g(w) \leq 0$. In our settings, the constraints will typically arise from the coordination required between the two manipulators, although other constraints may also be considered.

At each iteration of CCMA-ES, the algorithm samples a number of tentative points and verifies whether these points verify the constraints. If it does, it proceeds as in standard CMA-ES. If not, the new points are used to align the covariance matrix with the violated constraints, forcing the algorithm to produce samples that eventually verify all constraints.

To describe the algorithm in further detail, let us again consider the three steps outlined in the high-level description of CMA-ES, in the previous section. CCMA-ES keeps a *fading record* $v_j \in R^P$ for each constraint $h_j, j = 1, \ldots, m$. These records are updated to trace (with forgetting) which constraints are violated by the different samples, and to update the covariance matrix so that it remains aligned with the constraint surfaces. CCMA-ES can then be summarized as follows [20].

1) **Sampling:**
   - Sample K points $\{w_{i,k}, k = 1, \ldots, K\}$ from a multivariate Gaussian, $w_{i,k} \sim \text{Normal}(\mu_i, \sigma^2 \Sigma_i)$, with $\mu_i \in R^P$ and $\Sigma_i \in R^{P \times P}$;
2) **Verifying constraints and resampling:**
   - Update records v_j for which h_j is violated for at least one sample $w_{i,k}$;

- Adjust covariance matrix Σ_i according to the records v_j;
- Re-sample the points that violate constraints and repeat, until no violations are observed;

3) **Evaluating:**
- Compute $J(w_{i,k})$ for $k = 1, \ldots, K$;
- Select the best K_0 samples from $\{w_{i,k}, k = 1, \ldots, K\}$;

4) **Adaptating:**
- Update the mean μ_i to the weighted average of the best K_0 samples from $\{w_{i,k}, k = 1, \ldots, K\}$ selected in the previous step;
- Update the covariance matrix Σ_i to increase the likelihood of previously successful search steps.

For each point $w_{i,k}$ such that $h_j(w_{i,k}) > 0$, v_j is updated as

$$v_j = (1 - c_v)v_j + \frac{c_v}{\sigma}(w_{i,k} - \mu_i), \tag{5}$$

where c_v is an adjustable scalar for fading rate. Moreover, writing

$$a_j(w) = \begin{cases} 1 & \text{if } h_j(w) > 0, \\ 0 & \text{otherwise,} \end{cases} \qquad a_0(w) = \sum_{j=1}^{m} a_j(w),$$

the covariance matrix is updated according to

$$\Sigma_i = \Sigma_i - \frac{\beta}{a_0(w_{i,k})} \sum_{j=1}^{m} a_j(w_{i,k}) \frac{v_j v_j^T}{\|v_j\|^2}. \tag{6}$$

In a nutshell, CCMA-ES is obtained from CMA-ES (Algorithm 1) by including an intermediate step where all samples are checked against all constraints and, if a violation is detected, the covariance matrix is instantly adjusted and a new sample generated. This process is repeated until no constraints are violated, ensuring that the solution obtained by CCMA-ES verifies all constraints [21].

B. CCMA-ES for Coordinated Motion Generation

In the context of our work, CCMA-ES is used to generate/optimize the weights of the force function describing the motion of the two manipulators. Given a specific task—typically described by a target configuration for the robots' movements—and a set of constraints that the motion of the two manipulators must verify—for example, the two end-effectors must maintain a constant distance—our approach proceeds by: [1]

- Modulating the original DMP for each of the two manipulators target position. As a result, we obtain a single DMP $(y^1_{0:T}, y^2_{0:T})$ that describes the motion of the two manipulators, synchronized by a shared canonical system.
- Optimizing the joint DMP parameters (the weights of the corresponding force function) while ensuring that the constraints are verified.

[1]The approach in this work can apply to tasks that need multiple movements more than two movements. But in this research, we are working on a bimanual manipulation to simplify the results and make the method straightforward and easier to understand.

For example, we can optimize two DMPs to allow a robot to move a rigid object of length D by solving the constrained optimization problem

$$\underset{w_1,\ldots,w_N}{\text{minimize}} \quad J(y^1_{0:t}, y^2_{0:T}) \tag{7a}$$

$$\text{subject to} \quad \|y^1(t) - y^2(t)\| = D, \quad t = 0, \ldots, T, \tag{7b}$$

Where J is the objective function (4). We note that environmental obstacles can be incorporated either into the objective function J or as actual constraints. We note also that the DMP used to generate the original movement for the two manipulators can be obtained by combining several simpler DMPs, following [6]. This means that the overall framework enables the robot to generate complex motions in different environments while verifying task-specific coordination constraints.

Moreover, we can apply the method described in the [6] for moving obstacles, applying the same modulation matrix on both DMPs—as depicted in Fig 2. When the moving obstacle is close to the coordinated DMPs, we build the modulation matrix M for the DMP closest to the obstacle and apply it to both DMPs at the same state.

IV. Experiments and Results

We now illustrate the application of our method both in simulation and using the Baxter robot. We start by presenting results in simulation, where we show, side-by-side, the results obtained in a geometric simulator and in the more realistic Baxter simulator.

A. Simulation Results

We start by using the approach in [6] to generate an initial DMP that successfully navigates the environment. The resulting DMP is shown in Fig. 3. Note that the gap between the obstacles is relatively wide, so the optimization algorithm can easily find a path that leads the movement from the

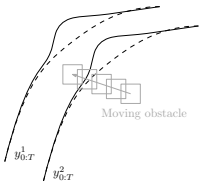


Fig. 2: Application of the modulation matrix to the two DMPs, y^1, and y^2, to avoid a moving obstacle.

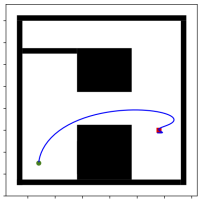


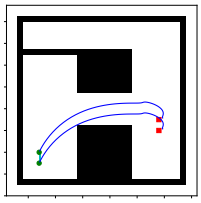
Fig. 3: Initial DMP used in the experiments, generated using the framework from [6]. The green circle and red square correspond to the start and endpoints.

initial position (the green circle) to the target position (the red square).

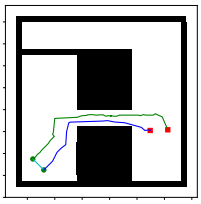
We then ran our proposed approach in the same environment, imposing a restriction where the distance between the two DMPs must be maintained constant. The task consists of moving a rigid object of length D from one place to another. To succeed in this task, the two manipulators must be coordinated to ensure that the object does not fall or hit any environmental obstacle. Therefore, a constraint is placed on the optimization process to enforce a fixed distance between the two endpoints of the robotic arms. The environment still retains the two obstacles separated by a corridor. The goal is to optimize the two DMPs describing the motion of the two arms to allow for safe crossing of the corridor.

The resulting trajectories are depicted in Fig. 4a. As the plot shows, the optimizer is able to successfully drive both DMPs from the initial position (green circles) to the target position (red squares), maintaining the distance between the two DMPs and avoiding obstacles. Figure 4b shows the result of our approach in a second environment, now featuring a narrower passage. In order to comply with the obstacles in the environment and the constraints imposed by the task, the trajectories are much less smooth than the ones in Fig. 4a.

Figure 5 shows the value of the objective function and the constraint as the movement progresses, in the environment from Fig. 4b. We can see that the DMPs steadily approach

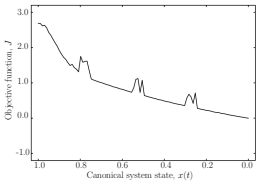


(a) Joint DMPs optimized to achieve a target configuration while maintaining the constant distance between the two manipulators.

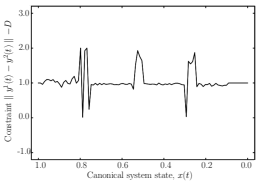


(b) Joint DMPs optimized to achieve a target configuration while maintaining the constant distance between the two manipulators in an environment with a narrower passage.

Fig. 4: Optimized joint DMPs in two configurations of the environment—the original configuration and a configuration with a narrow passage.



(a) The value of the objective function steadily approaches 0.



(b) Constraint violation during execution. Constraint $\left\|y^1(t) - y^2(t)\right\| - D$.

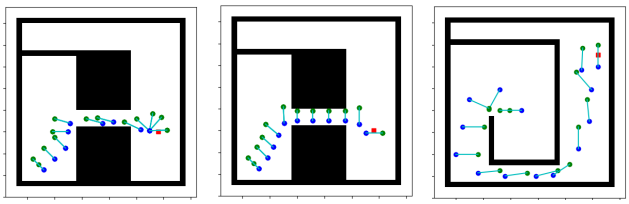
Fig. 5: Value of the objective function and constraint during the execution of the movement in Fig. 4b.

the target (the objective function value steadily approaches 0). We can also see some jumps in the objective function, when the DMP approaches the obstacles in the middle of the environment. Similarly, the constraint remains approximately 0 throughout the trajectory except in those same points, where the constraints exhibit some small violations.

Finally, we also compare our approach with an approach where the DMPs are jointly taught but where constraints are not considered during the optimization [22]. The resulting trajectories are shown in Fig. 6 (a & b), where we now explicitly showcase the different time steps, for easier inspection of whether the constraints are verified or not. When the learned DMPs are optimized to address environment restrictions (such as obstacles), but no task constraints are enforced to ensure coordination, the resulting DMPs may violate such constraints, even if the original joint DMP is learned by demonstration ensuring that the constraints are met—in the example, the original joint DMP was demonstrated in the environment with larger gap, so the constraints were easily verified in the demonstration. In contrast, our approach adapts to the new obstacle configuration while ensuring that the constraints are verified.

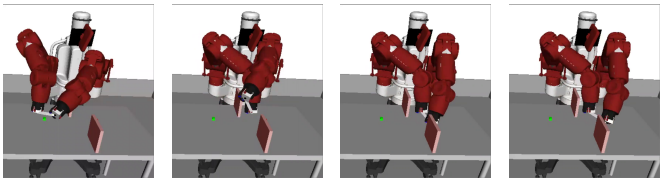
We also compared our work with the approach of [7], encountering similar difficulties—namely, the inability to include new environment information when adjusting the motion. Our approach includes the environmental information obstacles inside our cost function that evaluate the samples compatible with the task conditions and constraints. Also, the need for human annotation to determine the leader and follower DMPs restricts the outcome of this method; it may prevent the algorithm from finding specific solutions.

To prove the quality of our work, we applied it to another



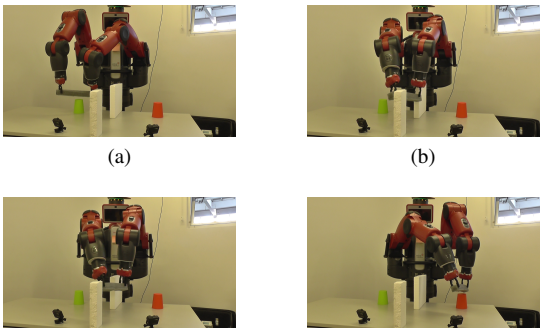
(a) (b) (c)

Fig. 6: Showcasing every two corresponding states from each trajectory while achieving the collaborative task. In (a & b) Comparison of our approach with an approach not enforcing constraints [22]. Green and blue circles represent different points along each DMP. The red squares correspond to the endpoint, and the naive bar between the two points represents the object that needs to be transferred using two optimized DMPs. Figure (b) shows the violation of the constraints on the distances between both corresponding states inside the path. In figure (a), all corresponding states follow the constraints of the task under the environmental conditions. Figure (c) shows the application of our work in a new complex environment with narrow gaps between the obstacles.



(a) (b) (c) (d)

Fig. 7: Snapshots of the Baxter simulator executing the trajectories generated by our approach to accomplish a coordinated task. Our approach could successfully optimize two DMPs for Baxter's arms to achieve a collaborative task that imposed constraints on the trajectory. Baxter needs to transfer an object between two positions.



(a) (b) (c) (d)

Fig. 8: Baxter uses its two arms by coordinating its DMPs to do the task of transporting an object through a path that can accommodate this object's size.

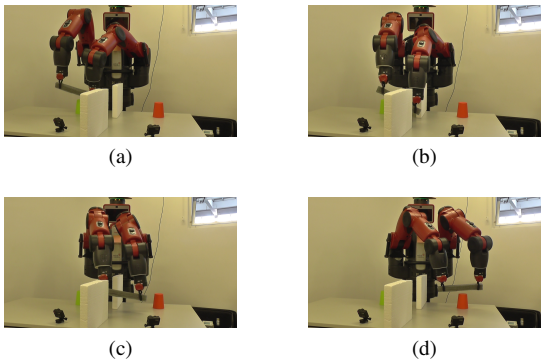
environment more complex than the previous one; we can see in Fig. 6c how our method can adapt the corresponding states at each DMP to achieve the collaborative task inside this environment with maintain the task constraints.

We conclude by showcasing the same results now using the Baxter simulator. Figure 7 shows different snapshots of the resulting trajectory, where the vertical obstacles reproduce the layout of the environment in the previous geometric simulation. Using the joint DMPs computed by our approach, the robot was able to complete the task and bypass the obstacles successfully.

B. Robot Experiments

We also applied our work in the physical Baxter robot, where the task was, once again, to move a solid object (a "stick") between two pre-specified positions using Baxter's two arms.

In Figs. 9 we showcase two views of the movement executed by Baxter when the passage is significantly reduced. With the narrow passage between the obstacles, Baxter must change the orientation of the stick to pass through successfully. The figures show two distinct views (captured with different cameras) of the same trajectory. We notice that—unlike in the first scenario, where the DMPs could be used almost unchanged—in this setting, the robot required a more significant adjustment of the joint DMP in order to go



(a) (b) (c) (d)

Fig. 9: Baxter uses its two arms by coordinating its DMPs to do the task of transporting an object through a tiny path that cannot accommodate this object's size without hitting obstacles.

through the path without colliding with the obstacles while maintaining the distance between the two end effectors.

V. Conclusions

In this work, we contributed a novel approach for movement generation in collaborative tasks requiring more than one robot arm to complete, using a single initial DMP optimized to operate in the environment. Our approach builds on the framework of [6] to build an initial DMP already considering the environment; this DMP is then used to build the joint DMP optimized to match the task-specific constraints. All DMPs are optimized together to accomplish the desired collaborative task and ensure successful coordination.

Our method is also amenable to further optimization to overcome the increasing difficulty imposed by the obstacles while maintaining the synchronization of the individual DMPs, thus efficiently transferring a demonstrated movement to novel scenarios involving multiple robots/arms. Besides that, our results show the ability to adapt to different environmental changes and task conditions.

Our research also opens exciting avenues for future work. For example, it would be interesting to find a way to optimize and coordinate multiple DMPs to bypass moving obstacles while maintaining collaborative task performance. The approach briefly discussed in Section III-B could be a starting point. Still, some experimental validations are necessary to assess whether the deformation imposed by the modulation matrix indeed maintains the trajectories within the constraints. Besides, the proposed approach does not explicitly consider the constraints, and it would be interesting to extend the modulation-based approach to enforce those constraints.

Acknowledgements

This work was supported by national funds through Fundaço para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020, the Center for Responsible AI Project with reference - C628696807-00454142, and RELEvaNT Project with reference PTDC/CCI-COM/5060/2021, INESC-ID, and the University of Luxembourg.

REFERENCES

[1] A. Bauer, D. Wollherr, and M. Buss, "Human-robot collaboration: A survey," *Int. J. Humanoid Robotics*, vol. 5, no. 01, pp. 47–66, 2008.

[2] S. Liemhetcharat and M. Veloso, "Modeling and learning synergy for team formation with heterogeneous agents," in *Proc. 11th Int. Conf. Autonomous Agents and Multiagent Systems*, 2012, pp. 365–374.

[3] P. Morasso, V. Mohan, G. Metta, and G. Sandini, "Motion planning and bimanual coordination in humanoid robots," *Frontiers in Artificial Intelligence and Applications*, vol. 196, pp. 169–185, 2009.

[4] R. Prada, P. Lopes, J. Catarino, J. Quitério, and F. Melo, "The geometry friends game AI competition," in *Proc. 2015 IEEE Conf. Computational Intelligence and Games*, 2015, pp. 431–438.

[5] S. Stavridis and Z. Doulgeri, "Bimanual assembly of two parts with relative motion generation and task related optimization," in *Proc. 2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2018, pp. 7131–7136.

[6] A. Kordia and F. Melo, "An end-to-end approach for learning and generating complex robot motions from demonstration," in *Proc. 16th Int. Conf. Control, Automation, Robotics and Vision*, 2020, pp. 1008–1014.

[7] A. Gams, B. Nemec, A. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Trans. Robotics*, vol. 30, no. 4, pp. 816–830, 2014.

[8] B. Nemec, N. Likar, A. Gams, and A. Ude, "Bimanual human robot cooperation with adaptive stiffness control," in *Proc. 16th IEEE-RAS Int. Conf. Humanoid Robots*, 2016, pp. 607–613.

[9] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. 2002 IEEE Int. Conf. Robotics and Automation*, vol. 2, 2002, pp. 1398–1403.

[10] ——, "Learning attractor landscapes for learning motor primitives," in *Adv. Neural Information Processing Systems 16*, 2003, pp. 1547–1554.

[11] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proc. 1996 IEEE Int. Conf. Evolutionary Computation*, 1996, pp. 312–317.

[12] I. Loshchilov, "A computationally efficient limited memory CMA-ES for large scale optimization," in *Proc. 2014 Annual Conf. Genetic and Evolutionary Computation*, 2014, pp. 397–404.

[13] F. Stulp and O. Sigaud, "Robot skill learning: From reinforcement learning to evolution strategies," *Paladyn J. Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, 2013.

[14] ——, "Path integral policy improvement with covariance matrix adaptation," in *Proc. 29th Int. Conf. Machine Learning*, 2012, pp. 1547–1554.

[15] A. Auger and N. Hansen, "Tutorial CMA-ES: Evolution strategies and covariance matrix adaptation," in *Proc. 14th Companion Conf. Genetic and Evolutionary Computation*, 2012, pp. 827–848.

[16] A. Abdolmaleki, "Information theoretic stochastic search," Ph.D. dissertation, University of Aveiro, 2018.

[17] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[18] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber, "Efficient natural evolution strategies," in *Proc. 11th Annual Conf. Genetic and Evolutionary Computation*, 2009, pp. 539–546.

[19] C. Cardoso, L. Jamone, and A. Bernardino, "A novel approach to dynamic movement imitation based on quadratic programming," in *Proc. 2015 IEEE Int. Conf. Robotics and Automation*, 2015, pp. 906–911.

[20] D. Arnold and N. Hansen, "A (1+1)-CMA-ES for constrained optimisation," in *Proc. Genetic and Evolutionary Computation Conference*, 2012, pp. 297–304.

[21] G. Arampatzis, D. Wälchli, P. Weber, H. Rästas, and P. Koumoutsakos, "(μ, λ)-CCMA-ES for constrained optimization with an application in pharmacodynamics," in *Proc. Platform for Advanced Scientific Computing Conference*, 2019, pp. 1–9.

[22] H. Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters, "Generalization of human grasping for multi-fingered robot hands," in *Proc. 2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2012, pp. 2043–2050.