Online Motion Planning for Safe Human-Robot Cooperation using B-Splines and Hidden Markov Models

Giovanni Braglia[1], Matteo Tagliavini[2], Fabio Pini[1] and Luigi Biagiotti[1]

***Abstract*—When humans and robots work together, ensuring safe cooperation must be a priority. This research aims to develop a novel real-time planning algorithm that can handle unpredictable human movements by both slowing down task execution and modifying the robot's path based on the proximity of the human operator. To achieve this, an efficient method for updating the robot's motion is developed using a two-fold control approach that combines B-Splines and Hidden Markov Models. This allows the algorithm to adapt to a changing environment and avoid collisions. The proposed framework is thus validated using the Franka Emika Panda robot in a simple start-goal task. Our algorithm successfully avoids collision with the moving hand of an operator monitored by a fixed camera.**

***Index Terms*—Human-Robot Interaction, Obstacle Avoidance, Splines, Hidden Markov Models**

I. INTRODUCTION

In modern industries, the demand for collaborative robots is constantly increasing because of their versatility and precision in a wide variety of applications. In a collaborative cell, humans and robots work together, sharing the same workspace. Therefore, the robot must adapt to external events while complying with safety regulations [1]. Safe human-robot cooperation can be achieved if the control system can gather information about the state of the robot as well as the state of the worker inside the collaborative workspace. Obstacle avoidance algorithms are employed whenever the presence of a target, such as an object or an operator, should be accounted for the proper and safe operation of the robot. The latter must be able to adapt its behavior in real-time with respect to dynamically changing environments [1].

In general obstacle avoidance applications [2], the robot uses the position of a target object to modify its path and avoid collision with it. Although safety distance margins can be imposed on the target object, unexpected movements can cause the robot to collide if the algorithm is not reactive enough to prevent the impact [3]. Moreover, to enhance productivity, the robot should restore its original path after avoiding a collision and when the obstacle no longer interferes with its motion.

In this work, the moving obstacle is represented by the hand of a human operator who shares the same workspace as a cobot and whose position is monitored by a fixed camera. The robot must repeat a pre-planned task, but according to the proposed framework, when the risk of a collision occurs, it can either modify the geometry of the path or reduce its velocity according to the requirements of the specific application. Then, when the obstacle is no longer within the robot's reach, the programmed motion is restored.

The literature is full of methods that guarantee obstacle avoidance and safe coexistence of humans and robots. To highlight the contribution of our proposal, the main state-of-the-art techniques are reviewed and briefly discussed in the following.

A. *Related Work*

Collision avoidance methods for robotic systems operating in dynamic environments with obstacles moving at high velocity typically rely on a combination of real-time trajectory planning and reactive control. These methods may vary in their level of integration, ranging from pure re-planning to the use of repulsive forces that act directly on the robot.

On one hand, collision avoidance motion planning problems can be treated as an optimal control problem [4]. These approaches are widely used because they allow for the minimization of cost functions subject to disparate constraints, such as danger fields, temporal aspects, or even metrics for evaluating safety in collaborative manufacturing [3], [5]. Optimization methods can be applied to both motion planning and control problems. A noteworthy optimization-based algorithm is CHOMP [6], i.e. Covariant Hamiltonian Optimization for Motion Planning, which guarantees collision-free and locally optimal trajectory generation by means of a covariant gradient update rule. Another example is presented in [7], where a parallel optimization scheme is successfully used to manipulate a cable-towed load with multiple collaborative quadrupeds.

Probabilistic inference methods [8]–[10] optimize trajectories subject to task constraints, goals, and motion priors, replacing classical cost functions with joint distributions formulated as conditional dependencies. In [11], the author introduces a Learning by Demonstration (LbD) framework that exploits Gaussian Mixture Models (GMM) to encode multiple tasks and retrieve the associated skill by means of regression techniques.

---

*This work has been partially funded by the ECOSISTER Project. Project funded under the National Recovery and Resilience Plan (NRRP), Mission 04 Component 2 Investment 1.5 – NextGenerationEU, Call for tender n. 3277 dated 30/12/2021. Award Number: 0001052 dated 23/06/2022.

[1] The authors are with the engineering department Enzo Ferrari of the University of Modena and Reggio Emilia, Italy {`giovanni.braglia`, `fabio.pini`, `luigi.biagiotti`}`@unimore.it`

[2] The author is with Ideativa, an academic spin-off for the development of products, processes and services in industrial and collaborative robotics, Modena, Italy `matteo.tagliavini@ideativa.it`

A different class of methods for motion planning and collision avoidance is composed of so-called sample-based planning algorithms [12]; according to these approaches, the environment is randomly sampled using techniques such as probabilistic road maps or exploring random trees. An example is given in [13], where sampled-via points are transformed into a set of candidate collision-free trajectories subject to predefined kinodynamic limits. In [14], if a moving obstacle will collide with the manipulator, the motion is locally replanned by using a Bi-RRT-based algorithm connecting the current robot position with a target on the unperturbed trajectory.

While the above-mentioned algorithms mainly work at the planning levels, an approach for reactive collision avoidance that acts directly on the robot control relies on potential field methods, originally introduced in [15]. Virtual repulsive and attractive fields are utilized to move the robot towards a target while avoiding obstacles. Specifically, repulsive fields are associated with obstacles, while attractive fields are associated with the target. The original concept has been further refined and applied to properly modify the shape of motion trajectories, which are seen as elastic bands subject to virtual forces [16], and is now a standard technique for real-time trajectory planning [2], [17].

If all the techniques cited above modify the geometry of the robot motion, state-of-the-art methods in the industrial practice for guaranteeing safety in human-robot applications are based on a completely different philosophy. They basically consist of stopping or slowing down robot motion in the presence of humans. For instance, according to ISO/TS 15066, the velocity and acceleration of the robot must be set to safe values based on the minimum distance from the operator [18]. This approach is commonly used in collaborative applications, where the position of human workers is continuously monitored with different types of cameras and the timing along the curve is properly scaled. See, for example, [5], [19], among many other publications on this topic. Note that, in the case of dynamic obstacles, this technique based on the velocity modulation of the robot can also be a way to avoid obstacles that are currently on the desired geometric path but could move in the following instants.

B. Methodology and Contributions

In this work, we propose a novel framework for collision avoidance that merges the modification of the geometric path of the robot with a speed adaptation mechanism. The basic idea, derived from observations reported in [20], is to split the task representation into two fundamental categories: the *trajectory* and the *symbolic* level. The former comprises continuous signals that change over time, such as the position or orientation of the end-effector (EE); the latter uses sequential or hierarchical information to establish a discrete set of movements with predefined rules. The goal of this work is to provide evidence, through an obstacle avoidance application, that task performance can benefit from the merging of these two domains.

The two basic tools used to implement this framework are B-Splines, which encode spatial data and modify the path in Cartesian space [21], and Hidden Markov Models (HMMs), which encode temporal and sequential information by scaling down task velocity [22].

The main goals and contributions of this research work are:

1) Design an online controller that can fit generic tasks and smoothly avoid collision with dynamic obstacles.
2) Include the possibility to restore the original task whenever the robot is not prone to any collision.
3) Exploit a probabilistic framework to gather information about the obstacle and modify the robot's velocity accordingly.
4) Combine trajectory and symbolic domains in a unified framework.

The paper is structured as follows: Section II details the theoretical background of the proposed solution. Section III describes the online control algorithm conceived in this work. Section IV describes the experimental setup and validates the results of our framework. Finally, section V concludes this work by discussing the results and possible future steps.

II. TASK ENCODING

The simplest way to define a robotic task is by specifying the trajectory, which is a mapping between time and space, that the end-effector or joints must track. However, a task can also be interpreted as a sequence of action units or elements that follow loops or rules, as is the case with a symbolic representation as seen in [23] and [24].

This work attempts to bridge and combine these two domains for an obstacle avoidance application. This is achieved through a control system that modifies the nominal robot path defined by B-Spline functions [21], while also varying the phase of the task - and thus its velocity - using HMM [22]. The following paragraphs provide a brief review of the theoretical background of these techniques.

A. Spatial encoding based on B-Splines

Spline functions are extensively used for generating smooth and optimal-time trajectories in robotic applications [13] [25] [26]. B-Splines are a particular representation of generic spline curves based on a linear combination of N *basis functions* [25], i.e.

$$\boldsymbol{p}(s) = \sum_{i=1}^{N} \boldsymbol{p}_i \beta_i^d(s), \quad s_{min} \leq s \leq \ s_{max} \tag{1}$$

where $\beta_i^d(s)$ are basis functions of degree d, which only depends on the phase variable s (that in many applications coincides with time t), while $\boldsymbol{p}_i$ are called control points and determine the geometric shape of the curve. As shown in Fig.1, a B-spline is basically a smooth approximation of the control polygon defined by the control points. Usually, they are defined by imposing some interpolation condition $\boldsymbol{p}(s_i) = \boldsymbol{p}_i, \ i = 1, ..., N$, depending on the desired task. A noteworthy property of the basis functions $\beta_i^d(s)$ is that

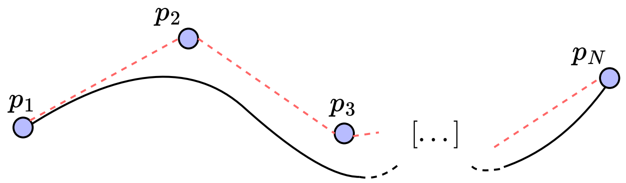


Fig. 1. Example of B-Spline trajectory: $\boldsymbol{p}(s)$ (black line) is an approximation of the control polygon (red dashed line) defined by the control points $\boldsymbol{p}_i$.

their value is zero everywhere except in the interval defined by knots $[s_i, s_{i+d+1}]$. As a consequence, a change in the i-th control point $\boldsymbol{p}_i$ only influences the B-Spline in this interval, allowing for local modifications in the trajectory described. Another useful property of B-Splines is time-scaling, i.e. the possibility of changing the velocity along the curve by simply applying the transformation $\hat{s} = \alpha s$ to the knot vector s, where α is a constant. For more details, see [21].

B. Temporal and Sequential encoding based on Hidden Markov Models

Hidden Markov Models (HMMs) are based on a Markov chain, which describes the probabilities of sequences of random variables, called *states*, that take values from a defined set [27]. In many applications, such as gesture and speech recognition [22], [24], HMMs are used to model human behavior, with the aim of identifying gestures or predicting the most likely pattern of subsequent control states. Specifically, given a set of *hidden states* $H = h_1, h_2, \dots, h_M$ and a sequence of T observations $O = o_1, o_2, \dots, o_T$, the purpose of HMMs is to analyze sequences of events in terms of a reduced set of parameters $\boldsymbol{\lambda} = [\mathbf{\Pi}, \mathbf{A}, \mathbf{B}]$, where:

- $\mathbf{\Pi}$ is called the *prior distribution*, which tells us about the probability of starting a sequence in state h_i;
- $\mathbf{A}$ is the *transition probability matrix*, where each element $a_{i,j}$ encodes the probability of moving from state i to state j;
- $\mathbf{B}$ are the *observation likelihoods*, also called emission probabilities, where each element $b_i(o_t)$ expresses the probability of an observation o_t being generated from state i.

Note that, according to the usual definition of HMMs, $\boldsymbol{\lambda}$ describes a model in terms of discrete observation likelihoods [28]. However, since the proposed framework deals with a continuous observation space, i.e., trajectories in Cartesian space, the HMM can be represented as

$$\boldsymbol{\lambda} = [\mathbf{\Pi}, \boldsymbol{A}, \boldsymbol{\mu}_i, \mathbf{\Sigma}_i] \qquad i = 1, ..., M \tag{2}$$

where M is the number of components characterized by mean and covariance values $(\boldsymbol{\mu}_i, \mathbf{\Sigma}_i)$ of the multivariate Gaussian Mixture Model (GMM). Thus, the $\boldsymbol{\lambda}$ parameters can be learned to encode the desired robot task, such that the regression of the model resembles the desired end-effector path [29].

The proposed methodology can be explained as follows: using control points $\boldsymbol{p}_i$ as hidden states h_i, a forward-chained left-right HMM is adopted (see Fig.2) to account for the evolution of the trajectory [30]. With this model, in nominal conditions (i.e., no obstacles in the workspace), $\pi_1 = 1$ and transition probabilities $a_{i,i+1} \approx 1$, meaning that the robot evolves from one control point to the other with a probability close to one. In particular, let's analyze the transition matrix $\boldsymbol{A}$ for the HMM in Fig.2:

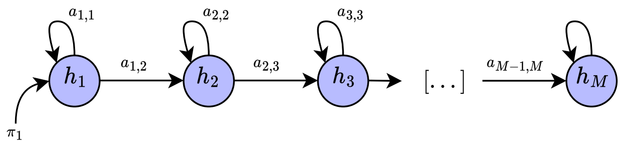


Fig. 2. Example of forward chained left-right HMM: being in state h_i, it is only possible to move to the next state h_{i+1} or remain in the current state.

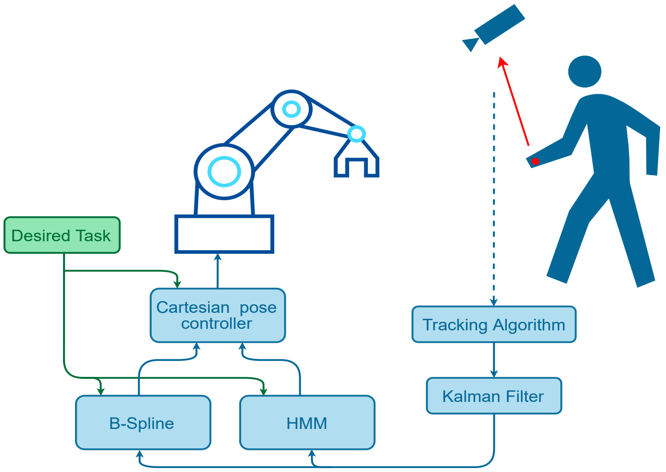


Fig. 3. Block scheme representation of the proposed framework for collision avoidance.

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & 0 & \cdots & 0 & 0 \\ 0 & a_{22} & a_{23} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{M-1M-1} & a_{M-1M} \\ 0 & 0 & 0 & \cdots & 0 & a_{MM} \end{bmatrix} \tag{3}$$

Here, the elements of each row of $\boldsymbol{A}$ sum up to 1. Parameter $a_{i,i}$ is the probability of staying in state h_i, while $a_{i,i+1}$ is the probability of moving from state h_i to state h_{i+1}. Thus, the transition probabilities $a_{i,i+1}$ can be used to slow down or eventually stop the robot every time their values warn the system about a potential collision, as will be explained in the next section.

III. PROPOSED METHOD

We consider a typical industrial scenario in which a robot is required to perform a specific task within a monitored workspace, using a fixed camera. When an obstacle enters the workspace, such as the hand of a human operator, its actual position is tracked, filtered through a Kalman filter, and sent to the controller that implements the proposed framework in order to prevent possible collisions. The functional blocks of the entire algorithm are displayed in the scheme shown in Fig.3, while the working principles are detailed below.

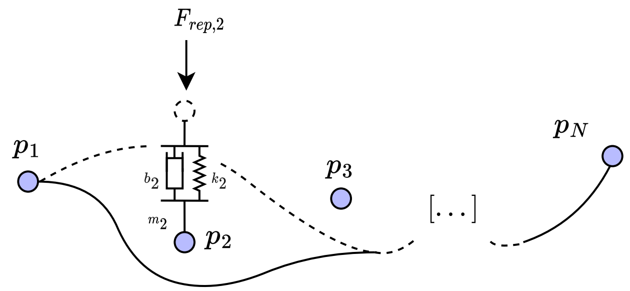


Fig. 4. B-Spline with associated dynamical systems to control points $\boldsymbol{p}_i$: when an external force $F_{rep,2}$ acts on $\boldsymbol{p}_2$, it locally changes the desired path.

A. Spatial modulation with dynamical control points

Given the presence of moving obstacles, we assume that the task is modified only if they interfere with the nominal trajectory. However, if the obstacles move far from the robot, the task should restore to its original trajectory. To achieve this, we use B-Spline functions to induce a reversible deformation on the original trajectory. Specifically, a dynamical system is associated with each control point $\boldsymbol{p}_i$ [31]:

$$\boldsymbol{M}_i,\ddot{\tilde{\boldsymbol{p}}}_i + \boldsymbol{B}_i,\dot{\tilde{\boldsymbol{p}}}_i + \boldsymbol{K}_i,\tilde{\boldsymbol{p}}i = \boldsymbol{F}_{rep,i} \tag{4}$$

where, $\tilde{\boldsymbol{p}}_i = \boldsymbol{p}^{(i)} - \boldsymbol{p}_i$ represents the displacement of the actual position $\boldsymbol{p}^{(i)}$ of the i-th control point with respect to its original location $\boldsymbol{p}_i$. The matrices $\boldsymbol{M}_i = \text{diag}\{m_i\}$, $\boldsymbol{B}_i = \text{diag}\{b_i\}$, and $\boldsymbol{K}_i = \text{diag}\{k_i\}$ are 3-by-3 diagonal matrices, where m_i, b_i, and k_i are positive constants such that the system in Eq. (4) is critically damped. Finally, $\boldsymbol{F}_{rep,i}$ is a virtual repulsive force applied in the direction from the obstacle to the i-th control point. This idea is explained in Fig. 4. The module of the repulsive force $\boldsymbol{F}_{rep,i}$ acting on the i-th control point does not depend on its Cartesian distance from the obstacle but varies in intensity according to the Mahalanobis distance:

$$D_M(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}$$

that accounts for the occupation probability of the obstacle $\hat{\boldsymbol{p}}_{obs} \sim \mathcal{N}(\boldsymbol{\mu}_{obs}, \boldsymbol{\Sigma}_{obs})$, caused by uncertainties in the estimation of its location [4], [32]. The module of the force can therefore be computed as follows [4]

$$|F_{rep,i}| = \frac{\chi_i}{D_M(\boldsymbol{p}^{(i)}; \boldsymbol{\mu}_{obs}, \boldsymbol{\Sigma}_{obs}\ )} \tag{5}$$

where χ_i is a free parameter that enables to vary the repulsive force field intensity.

B. Temporal modulation with varying transition probabilities in HMM

In our framework, the path of the task is encoded by $\boldsymbol{p}(s)$ in (1), while velocity is varied through HMM's parameters $\boldsymbol{\lambda}$, accounting for the temporal modulation.

The connection between B-Splines and HMM domains is established through the continuous observation likelihoods $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = b_i(o_t)$, $i = 1, ..., M$, as shown in (2). In particular, $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ are the components of a GMM that encodes the likelihood of a point $\boldsymbol{r} \in \mathbb{R}^3$ being described by the HMM's model $\boldsymbol{\lambda}$:

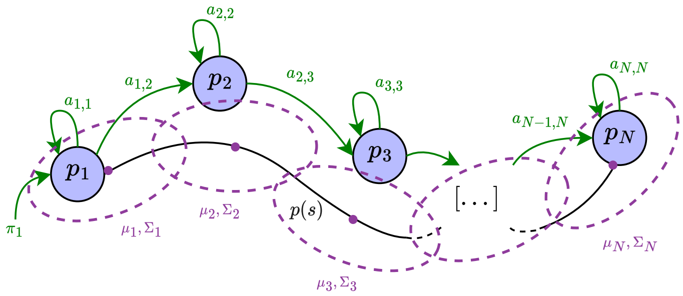


Fig. 5. B-Spline and HMM combined together: while the former describes the nominal path $\boldsymbol{p}(s)$ in the Cartesian domain, the latter provides a probabilistic map of it as described in (6), together with transition probabilities $a_{i,j}$ among control points $\boldsymbol{p}_i$ computed such that a left-right HMM is obtained, thus ensuring always going from a start position to a goal one.

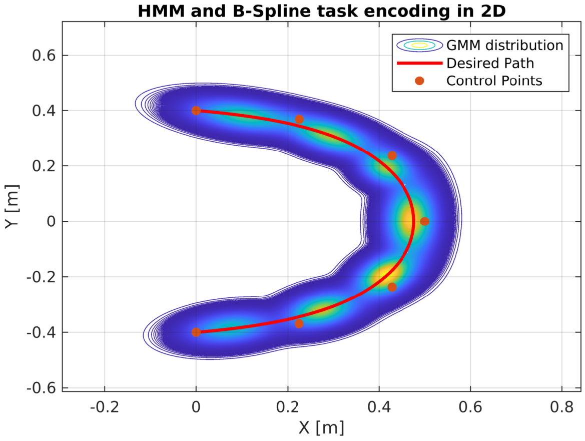


Fig. 6. Control at both trajectory and symbolical level using B-Splines and HMM ($\xi = 0.5$). A planar task, also used in the experiments, has been considered for illustration purposes. It is encoded using a B-Spline exploiting 7 control points (with fixed height $z = 0.3$m), and an HMM with observation space described by a GMM with 7 components.

$$\mathbf{B} \sim P(\boldsymbol{r}) = \sum_{i=1}^{M} \mathcal{W}_i(\boldsymbol{r}) \mathcal{N}(\boldsymbol{r}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{6}$$

where the weighting factor $\mathcal{W}i$ with $\sum_{i=1}^{M} \mathcal{W}_i = 1$ is called the mixing coefficient and represents the influence of the i-th component [11]. Gaussian distributions are encoded such that their regression corresponds to the nominal trajectory $\boldsymbol{p}(s)$ [33]. Furthermore, in equation (6), $\mathbf{B}$ accounts for the variability introduced by the movement of control points, thus giving a stochastic topological representation of the task. Figure 5 illustrates a graphical representation of this approach. It is worth noting that the number of states M of the HMM can be generally different from the number of control points N that define the B-spline. For instance, it is possible to encode with the same state a segment of the trajectory curve, corresponding to a subtask, composed of several control points. However, for the sake of simplicity, it is convenient to assume $M = N$. As explained in Section II, the evolution of the B-spline trajectory can be modified by applying the transformation $\hat{s} = \alpha s$ to the knots vector of $\boldsymbol{p}(s)$. In this way, the desired velocity $\dot{s}$ is multiplied by a factor $\alpha \in [0, 1]$ and reduced accordingly. To implement a mechanism that is able to slow down the robot and eventually stop it in case of a collision risk, the transition probabilities of the HMM have been directly mapped into proper values of α, as explained below. Assume that the robot's trajectory is in state h_i (associated with the control point $\boldsymbol{p}_i$), and consider the transition probability $a_{i,i+1}$ to the next state h_{i+1}. For the purposes of this project, it is required that:

- If $a_{i,i+1} = 1.0$ the robot moves at nominal velocity.
- If $0.0 < a_{i,i+1} < 1.0$ the robot slows down.
- If $a_{i,i+1} = 0.0$ the robot stops.

By computing the transition coefficient as a Sigmoid function ϕ based on the Mahalanobis distance $D_M(\hat{\boldsymbol{p}}_{obs}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ between the obstacle position and the observation space of the HMM, i.e.,

$$a_{i,i+1} = \phi(D_M) = \frac{1}{1 + e^{-(D_M + \gamma)}}, \tag{7}$$

its value changes according to the probability of colliding with the obstacle. Note that the free parameter γ can be used to tune the sensitivity of the robot with respect to the presence of an obstacle in the workspace. Finally, if the scaling factor α is computed as

$$\alpha = a_{i,i+1}, \tag{8}$$

the nominal velocity along the B-spline trajectory is modulated as required by the specifications reported above.

In this way, the coefficients $a_{i,i+1}$ acquire a new meaning, which is the probability of passing from control point $\boldsymbol{p}_i$ to $\boldsymbol{p}_{i+1}$ without collision. Figure 6 reports an example of 2D encoding merging the two techniques introduced in the previous paragraphs.

C. Switching between Trajectory and Symbolic domains

The proposed motion planner combines two different control domains. However, there could be situations where operating at the trajectory level is preferred over the symbolic one, or vice versa. The algorithm is provided with a knob parameter, $\xi \in [0, 1]$, so the user can choose the influence of each controller. Precisely, we expect the following behaviors:

- $\xi = 0.0 \implies$ only the time-scaling mechanism based on HMM is active;
- $0.0 < \xi < 1.0 \implies$ B-Spline and HMM cooperate with different proportions;
- $\xi = 1.0 \implies$ only B-Spline modification algorithm is applied.

To achieve this, the parameters (m_i, b_i, k_i) in equation (4) for trajectory control, and α in equation (8) for symbolic control, are modified according to the following laws:

$$(m_i, b_i, k_i) = \begin{cases} \dfrac{(\hat{m}_i, \hat{b}_i, \hat{k}_i)}{2\xi} & \text{if } \ 0 \le \xi < 0.5 \\ (\hat{m}_i, \hat{b}_i, \hat{k}_i) & \text{if } \ 0.5 \le \xi \le 1 \end{cases} \tag{9a}$$

$$\alpha \ = \begin{cases} \hat{\alpha} & \text{if } \ 0 \le \xi \le 0.5 \\ (2\hat{\alpha} - 1)(1 - \xi) + \xi & \text{if } \ 0.5 < \xi \le 1 \end{cases} \tag{9b}$$

where $(\hat{m}_i, \hat{b}_i, \hat{k}_i)$ are the default values assigned in equation (4), and $\hat{\alpha}$ is the result of equation (8). In this way, as ξ approaches 0, the values of the parameters (m_i, b_i, k_i) increase rapidly[1], making the system in (4) extremely rigid and thus insensitive to external (virtual) forces. As a consequence, no spatial modifications of the trajectory are observed. On the other hand, as ξ approaches 1, the spatial modification mechanism is restored, but in this case $\alpha \to 1$ and therefore no changes in the velocity profile occur. Finally, when $\xi \approx 0.5$, both the trajectory and symbolic level controllers work together, producing spatial and temporal modulation of the trajectory.

IV. EXPERIMENTAL VALIDATION

In this section, the experimental setup is described and the results of the proposed framework are shown.

A. Experimental setup and methodology

For the experimental tests, we used a Franka Emika Panda collaborative robot under a Cartesian pose controller developed in C++ running on a standard PC equipped with an Intel i7 8-core CPU and 8 GB RAM. In the same PC the trajectory/symbolic motion control has been implemented using the Matlab/Simulink environment. As already mentioned, the moving obstacle was represented by the hand of a human, sharing the same working area of the robot, and whose position in the Cartesian space was constantly tracked by a fixed camera. In particular, an Intel RealSense D435 depth camera has been used, whose output was elaborated by a second PC, with characteristics similar to the first one. All the software components was connected using ROS [34], in order to simplify the communication among them.

Since the precise detection of the obstacles is not the main focus of this research work but is prerequisite for the successful execution of the proposed approach, to simplify the computation of the pose of the hand from the camera'S data an ArUco marker [35] directly placed on the hand of the operator have been exploited.

Finally, the position coordinates derived from the images are sent to a multi-step Kalman filter to predict the future positions of the hand [32] [36]. This is particularly useful in situations where the camera's view of the scene is obstructed. It is worth noting that the output of the Kalman filter, denoted as $\hat{\boldsymbol{x}}$, can be associated with a uniformly distributed random variable, i.e., $\hat{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{\mu x}, \boldsymbol{\Sigma x})$ [37]. This is important because it justifies the use of the Mahalanobis distance in modifying the task, as shown in (5) [4].

For the tests, a cubic B-spline with $N = 7$ disposed in the plane $x - y$ is used at the trajectory level. The values of (m_i, b_i, k_i) in equation (4) are chosen such that the stiffness related to the first and last control points is high enough to keep their positions constant. At the symbolic level, the task is encoded into a GMM with 7 components (i.e. $M = N$) in 3D space. These components are used as observation likelihood

---

[1] In practical experiments an upper bound on the three parameters is imposed.

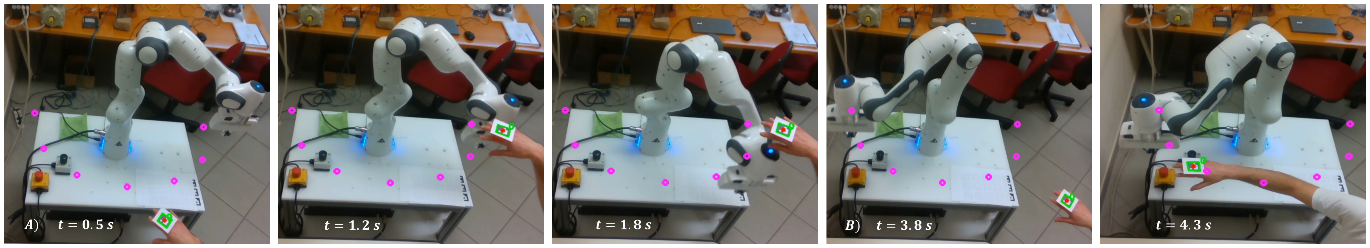


Fig. 7. Five snapshots of experiments conducted on the B-Spline-HMM based controller. The control points corresponding to the nominal path are displayed in magenta. In sequence A, the hand holding the marker moves slightly above the end-effector, inducing the robot to slow down and modify its path to pass under the hand. In sequence B, the hand suddenly moves in front of the end-effector, giving the robot no time to modify its path. Therefore, the robot controller decides to stop and wait until the hand changes position. Here is a link to the related video: https://www.youtube.com/watch?v=z6HBSz7o4qo

parameters in equation (2) to obtain a left-right HMM.

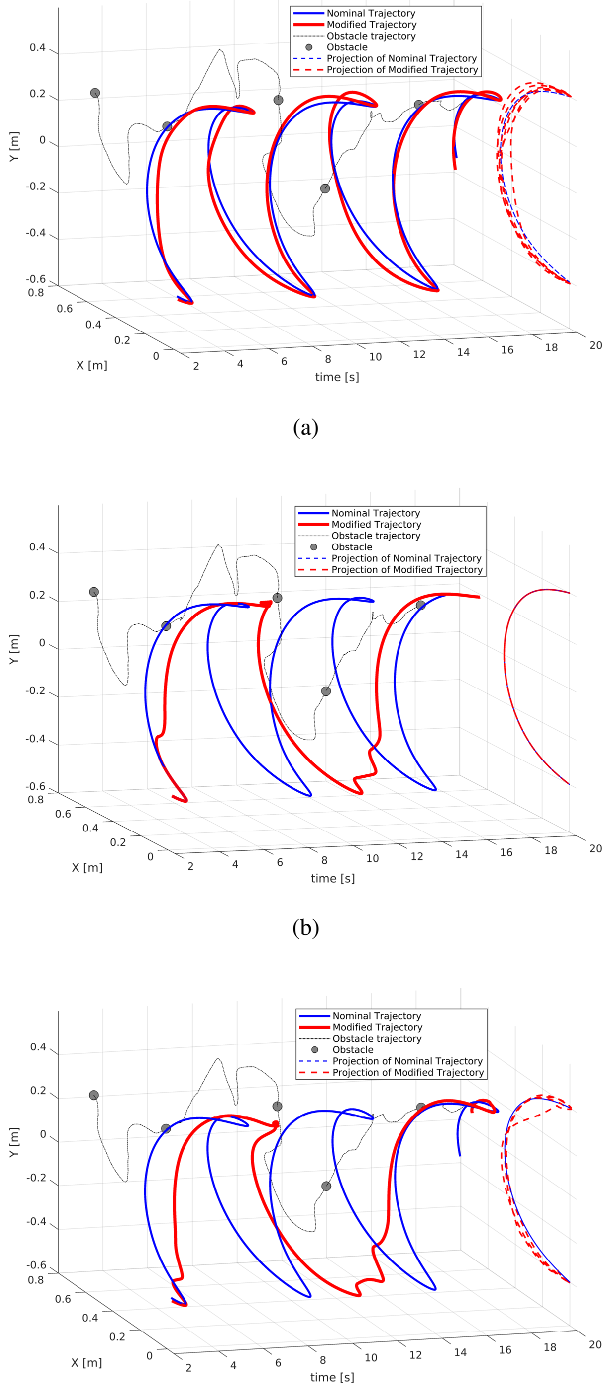
During the experiments, the robot was programmed to follow a desired trajectory while an operator moved their hand close to it to simulate collisions. The ξ parameter in (9) was varied to individually analyze the B-Spline control, HMM, and the ensemble of the two controllers. By doing so, we aimed to provide evidence of the improvements that can be achieved by combining the two techniques, and to validate our novel planning methodology for obstacle avoidance.

Performance was evaluated using four parameters: the average computation time for a single iteration T_{comp}, the normalized average path deviation ΔL, which was calculated as the mean deviation of the traveled distance normalized over the nominal path, the average stop time T_{stop}, which was calculated as the average time taken for parameter α in (8) to reach zero (i.e., stop the robot) once the hand was perceived to collide, and finally, the success rate ρ. The success rate was defined as the percentage between the number of cases where the distance did not go down under a given safety threshold and the total number of situations where an incipient collision was detected. Figure 7 shows snapshots of the real experiments conducted on the robot to test the B-Spline-HMM-based controller.

B. Results

In Fig. 8, for comparison purposes, the trajectory of the hand was recorded and then reproduced virtually in three different tests with B-Spline, HMM, and B-Spline-HMM-based controllers. Here, we can observe path modifications (in the XY projections reported on the right) and temporal modulation (along the time axis) behaviors produced by the three controllers. On the other hand, several experiments have been conducted in which the robot executes the same task repeatedly in different scenarios where ξ, task execution speed, and acceleration are varied. In these experiments, the operator moves their hand randomly to simulate collisions with the robot. The results are summarized in Table I and Fig. 9, and explained in the following paragraphs. For the controller at the trajectory level only (i.e., $\xi = 1.0$), an average computation time of $T_{comp} = 3.515$ ms was recorded for a single iteration. A total of 59 possible collision situations were realized in 6 different experiments, resulting in a success rate of $\rho = 77.97\%$ with a normalized average path deviation of $\Delta L = 0.590$. In this case, no value of T_{stop} was reported as the controller does not include temporal modulation In the second test, the same task was performed but only with the controller at the symbolic level (i.e., $\xi = 0.0$), using an HMM with an observation space described by a GMM with 7 components, and recording an average computation time of $T_{comp} = 1.480$ ms. During the experiment, a total of 58 possible collision situations were registered in 5 different experiments, resulting in a success rate of $\rho = 89.66\%$ with an average stop time of $T_{stop} = 0.431$ s and $\Delta L = 0.0$ as no spatial controller was implemented. Finally, the encoding of the task using control at both the trajectory and symbolic levels was validated, i.e. $\xi = 0.5$. The same configuration parameters as the first and second tests were maintained. The results showed that the algorithm runs at an average computation time of $T_{comp} = 4.649 ms$, which is clearly higher than the previous cases, but still within $16 ms$ to meet real-time requirements [2]. In this test, 143 possible collision situations were evaluated in 10 different experiments, obtaining a success rate of $\rho = 94.41\%$, with a normalized average path deviation of $\Delta L = 0.304 m$ and an average stop time of $T_{stop} = 0.514 s$.

In particular, we observed that in the first scenario where only B-Spline modification was used as control, poor performance was achieved, i.e. $\rho = 77.97\%$, as the controller could not modify the robot's path quickly enough to avoid a collision. The same applies to the controller implementing only HMM control, where the performance is better than the B-Spline controller ($\rho = 89.66\%$) as the robot stops or slows down when its position is too close to the operator. However, this behavior is not sufficient in all cases where the operator's hand moves toward the robot. In this case, significant improvements can be obtained by combining B-Spline and HMM controllers, i.e. $\rho = 94.41\%$. Despite the improved success rate, a higher value of T_{stop} was registered for the B-Spline-HMM controller compared to the symbolic level-only controller. However, this can be explained as the robot always tries to slow down and modify its path until the safety distance is violated, before completely stopping. On the other hand, we observed a lower ΔL value compared to the trajectory level-only controller since the robot now reduces its velocity toward the hand and eventually stops before the collision. Finally, note that we can always reduce ΔL by simply increasing the values of (m_i, b_i, k_i) in (4).



(a)

(b)

(c)

Fig. 8. Three different experiments using the same obstacle trajectory (represented by the black dashed line). The trajectory was virtually reproduced in each case. In every figure, the nominal path is plotted in blue and the modified path is plotted in red. The plots show how the XY coordinates of the end-effector vary over time with respect to the obstacle position. For clarity, the overall trajectory is projected onto the XY plane on the right-hand side of the figure, allowing us to appreciate the geometric path modifications. In experiment (a), only B-Spline control was used ($\xi = 1.0$). Here, the path was modified but not the task duration, as can be seen from the fact that the nominal and modified trajectories are aligned in time. In experiment (b), only HMM control was used ($\xi = 0.0$). Here, we can observe that even though the end-effector trajectory slows down when close to the obstacle, no path modification was registered as the XY projections of the nominal and modified trajectories perfectly overlap. In experiment (c), control was implemented at both the trajectory and symbolic levels using B-Splines and HMM ($\xi = 0.5$). When the obstacle approached the robot, both the spatial and temporal evolutions were modified.

TABLE I

VALIDATION TESTS RESULTS IN TERMS OF AVERAGE COMPUTATION TIME FOR SINGLE ITERATION T_{comp}, NORMALIZED AVERAGE PATH DEVIATION ΔL, AVERAGE STOP TIME T_{stop} AND SUCCESS RATE ρ.

	$T_{comp}[ms]$	ΔL	$T_{stop}[s]$	ρ
B-Spline	3.515	0.590	/	**77.97%**
HMM	1.480	0.0	0.431	**89.66%**
B-Spline+HMM	4.649	0.304	0.514	**94.41%**

V. CONCLUSION AND FUTURE WORK

In this paper, a novel framework for obstacle avoidance applications is introduced. After several real-world experiments, we believe that our methodology is suitable for human-robot shared environments, meeting real-time requirements for safe cooperation. The proposed work introduces a motion planning algorithm that can be adapted to different scenarios with moving and static obstacles, achieving a success rate of 94.41%. By combining the trajectory and symbolic domains, our framework can smoothly adapt the Cartesian path and slow down the execution of the task in the proximity of a human operator. Moreover, by associating a dynamical system with each control point, the controller autonomously restores to its original task, while using HMM the robot never stops after overtaking the operator's hand, ensuring correct and complete task execution.

Future work might involve the use of multiple cameras to span over a wider workspace and improve the detection of obstacles, no longer limited to the operator's hand. Moreover, it could be desirable to improve the algorithm that predicts the motion of moving obstacles and operators to obtain a larger time-horizon, more consistent with a smooth obstacle avoidance application.

REFERENCES

[1] S. M. Khansari-Zadeh and O. Khatib, "Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors," *Autonomous Robots*, vol. 41, no. 1, pp. 45–69, 2017.

[2] H. Liu, D. Qu, F. Xu, Z. Du, K. Jia, J. Song, and M. Liu, "Real-time and efficient collision avoidance planning approach for safe human-robot interaction," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 4, pp. 1–21, 2022.

[3] K. Merckaert, B. Convens, C.-j. Wu, A. Roncone, M. M. Nicotra, and B. Vanderborght, "Real-time motion control of robotic manipulators for safe human-robot coexistence," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102223, 2022.

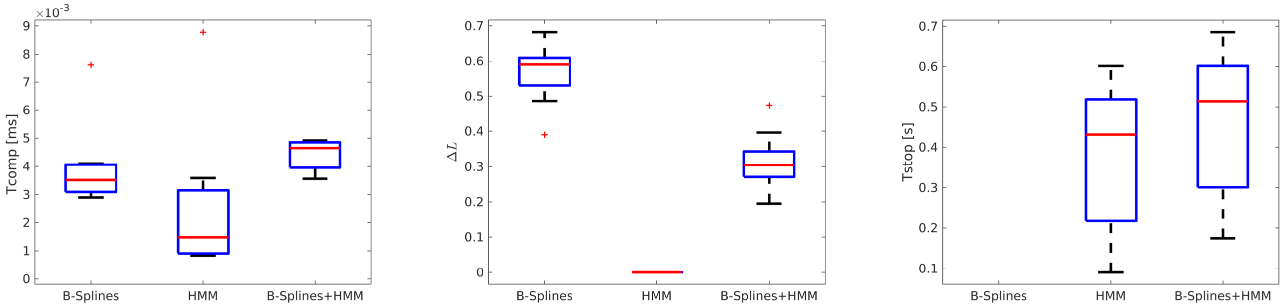
[4] A. Kanazawa, J. Kinugawa, and K. Kosuge, "Adaptive motion planning for a collaborative robot based on prediction uncertainty to enhance human safety and work efficiency," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 817–832, 2019.

[5] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, "Safety in human-robot collaborative manufacturing environments: Metrics and control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 882–893, 2015.

[6] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[7] C. Yang, G. N. Sue, Z. Li, L. Yang, H. Shen, Y. Chi, A. Rai, J. Zeng, and K. Sreenath, "Collaborative navigation and manipulation of a cable-towed load by multiple quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 041–10 048, 2022.

[8] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.



(a) (b) (c)

Fig. 9. Box-plot of the values reported in Table I; in plots (a), (b) and (c) are displayed the values of T_{comp}, ΔL and T_{stop}, respectively.

[9] M. Toussaint and C. Goerick, “A bayesian view on motor control and planning,” in *From Motor Learning to Interaction Learning in Robots*. Springer, 2010, pp. 227–252.

[10] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan, “Probabilistically safe robot planning with confidence-based human predictions,” *arXiv preprint arXiv:1806.00109*, 2018.

[11] S. Calinon, F. Guenter, and A. Billard, “On learning, representing, and generalizing a task in a humanoid robot,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[12] L. E. Kavraki and S. M. LaValle, “Motion planning,” in *Springer Handbook of Robotics*. Springer, 2016, pp. 139–162.

[13] J. Jankowski, L. Brudermüller, N. Hawes, and S. Calinon, “Vp-sto: Via-point-based stochastic trajectory optimization for reactive robot behavior,” *arXiv preprint arXiv:2210.04067*, 2022.

[14] D. Han, H. Nie, J. Chen, and M. Chen, “Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection,” *Robotics and Computer-Integrated Manufacturing*, vol. 49, pp. 98–104, 2018.

[15] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[16] S. Quinlan and O. Khatib, “Elastic bands: connecting path planning and control,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 1993, pp. 802–807 vol.2.

[17] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, “A depth space approach to human-robot collision avoidance,” in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 338–345.

[18] *Robots and robotic devices - collaborative robots*, International Organization for Standardization Std., 2016.

[19] M. J. Rosenstrauch, T. J. Pannen, and J. Krüger, “Human robot collaboration - using kinect v2 for iso/ts 15066 speed and separation monitoring,” *Procedia CIRP*, vol. 76, pp. 183–186, 2018, 7th CIRP Conference on Assembly Technologies and Systems (CATS 2018).

[20] S. Calinon, *Robot programming by demonstration*. EPFL Press, 2009.

[21] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[22] A. Pentland and A. Liu, “Modeling and prediction of human behavior,” *Neural computation*, vol. 11, no. 1, pp. 229–242, 1999.

[23] G. E. Hovland, P. Sikka, and B. J. McCarragher, “Skill acquisition from human demonstration using a hidden markov model,” in *Proceedings of IEEE international conference on robotics and automation*, vol. 3. Ieee, 1996, pp. 2706–2711.

[24] L. Roveda, M. Magni, M. Cantoni, D. Piga, and G. Bucca, “Human–robot collaboration in sensorless assembly task learning enhanced by uncertainties adaptation via bayesian optimization,” *Robotics and Autonomous Systems*, vol. 136, p. 103711, 2021.

[25] L. Biagiotti and C. Melchiorri, “Online trajectory planning and filtering for robotic applications via b-spline smoothing filters,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5668–5673.

[26] N. Sayols, A. Sozzi, N. Piccinelli, A. Hernansanz, A. Casals, M. Bonfè, and R. Muradore, “Global/local motion planning based on dynamic trajectory reconfiguration and dynamical systems for autonomous surgical robots,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8483–8489.

[27] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 02 2008, vol. 2.

[28] G. Mongillo and S. Deneve, “Online learning with hidden markov models,” *Neural computation*, vol. 20, no. 7, pp. 1706–1716, 2008.

[29] D. Vasquez, T. Fraichard, and C. Laugier, “Incremental learning of statistical motion patterns with growing hidden markov models,” *IEEE Transactions on intelligent transportation systems*, vol. 10, no. 3, pp. 403–416, 2009.

[30] M. Walter, A. Psarrou, R. Psarrou, and S. Gong, “Learning prior and observation augmented density models for behaviour recognition,” 1999.

[31] D. Chiaravalli, F. Califano, L. Biagiotti, D. De Gregorio, and C. Melchiorri, “Physical-consistent behavior embodied in b-spline curves for robot path planning,” *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 306–311, 2018.

[32] M. Meuter, U. Iurgel, S.-B. Park, and A. Kummert, “The unscented kalman filter for pedestrian tracking from a moving host,” in *2008 IEEE Intelligent Vehicles Symposium*. IEEE, 2008, pp. 37–42.

[33] J. Wiest, M. Höffken, U. Kreßel, and K. Dietmayer, “Probabilistic trajectory prediction with gaussian mixture models,” in *2012 IEEE Intelligent Vehicles Symposium*. IEEE, 2012, pp. 141–146.

[34] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: a practical introduction to the Robot Operating System*. ” O’Reilly Media, Inc.”, 2015.

[35] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[36] M. Kohler *et al.*, *Using the Kalman filter to track human interactive motion: modelling and initialization of the Kalman filter for translational motion*. Citeseer, 1997.

[37] S. J. Julier and J. K. Uhlmann, “New extension of the kalman filter to nonlinear systems,” in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068. Spie, 1997, pp. 182–193.