Realtime Avoidance of Fast Moving Objects: A Dynamical System-based Approach

S. Mohammad Khansari-Zadeh and Aude Billard
Ecole Polytechnique Federale de Lausanne, LASA Laboratory
{mohammad.khansari,Aude.Billard}@epfl.ch

***Abstract*—In this paper, we provide an extension to our previous approach [1] to perform obstacle avoidance in the presence of multiple fast moving and rotating obstacles. Our approach leverage on the notion of DS to generate robot motions that are inherently robust to perturbations and can instantly adapt to changes in the target and obstacles' positions in a dynamically moving environments. We validate our method in the challenging experiment of dodging a fast moving and rotating box on the 7-degrees of freedom (DoF) KUKA DLR arm.**

I. Introduction

Classical approaches to modeling robot motions rely on decomposing a task execution into two separate processes: *planning* and *control* [2]. The former is used as a means to generate a feasible path that can satisfy the task's requirements, and the latter is designed so that it follows the generated feasible path as closely as possible. These approaches consider any deviation from the desired path (due to perturbations or changes in environment) as the tracking error, and various control theories have been developed to efficiently suppress this error in terms of some objective functions. Despite the great success of these approaches in providing powerful robotic systems, particularly in factories, they are ill-suited for robotic systems that are aimed to work in the close vicinity of humans, and thus alternative techniques must be sought.

In robotics, DS-based approaches to motion generation have been proven to be interesting alternatives to classical methods as they offer a natural means to integrate planing and control into one single unit [3], [4], [5], [6], [7]. For instance when modeling robot reaching motions with DS, all possible solutions to reach the target are embedded into one single model. Such a model represents a global map which specifies *instantly* the correct direction for reaching the target, considering the current state of the robot, the target, and all the other objects in the robot's working space. Clearly such models are more similar to human movements in that they can effortlessly adapt its motion to change in environments rather than stubbornly following the previous path. In other words, the main advantage of using DS-based formulation can be summarized as: "Modeling movements with DS allows having robotic systems that have *inherent adaptivity* to changes in a dynamic environment, and that can *instantly* adopt a new path to reach the target". This advantage is particularly important in situations where there is no time to plan (or re-plan), no matter how fast the planning technique may be, and instant adaptation to a dynamically changing environment is required.

Despite the above features, most of the DS-based approaches to generating robot motions relies on a simplistic assumption that presume there is no object in the robot working space [3], [4], [7], [8]. Such assumption could be very limiting since many real world tasks require robotic systems that should work in cluttered environments where the robot may face several objects, which may appear suddenly during the task execution. Hence, it is essential to endow the existing DS-based control policy with the ability to avoid obstacles. In the face of fast moving obstacles, the devised algorithm should be computationally light so that can be used in closed-loop.

In our previous work, we have presented a novel approach to perform realtime obstacle avoidance based on dynamical systems that ensures impenetrability of multiple convex objects in quasi-static conditions [1]. This approach has a level of reactivity similar to existing local obstacle avoidance methods, while it ensures convergence to the target proper to global obstacle avoidance techniques[1]. In this paper, we extend this work for realtime obstacle avoidance in the presence of multiple fast moving and rotating objects, where the quasi-static assumption no longer holds. The presented method is free from local minima, and can ensure convergence of all trajectories to the target (as long as the target is reachable). We validate our method on the 7-DoF KUKA DLR arm in the experiment of dodging a fast moving and rotating box in 20 trials with various linear and angular velocities. Next we provide a recap of our previous work, and then present the extension for obstacle avoidance in the presence of fast moving objects.

II. DS-based obstacle avoidance

In this section we provide a brief overview of our DS-based approach for obstacle avoidance that is presented in [1]. This work assumes that the robot motion is driven by a continuous and differentiable DS in the absence of obstacle(s):

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}), \qquad \boldsymbol{f}: \mathbb{R}^d \mapsto \mathbb{R}^d \qquad \text{autonomous DS} \quad (1)$$

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(t, \boldsymbol{\xi}), \qquad \boldsymbol{f}: \mathbb{R}^+ \times \mathbb{R}^d \mapsto \mathbb{R}^d \qquad \text{non-auto. DS} \quad (2)$$

where $\boldsymbol{\xi} \in \mathbb{R}^d$ is a state variable that defines the state of the robot, t corresponds to time, and $\boldsymbol{f}(.)$ could be either an autonomous or non-autonomous DS. For simplicity, we further

[1]Please refer to [1] for more discussion on differences between this work and relevant state-of-the-art obstacle avoidance algorithms.

use the notation $\boldsymbol{f}(.)$ to refer to both autonomous and non-autonomous DS. Given an initial point $\{\boldsymbol{\xi}\}^0$, the robot motion along time can be computed by integrating $\boldsymbol{f}(.)$ recursively.

In this paper we take an imitation learning approach to construct $\boldsymbol{f}(.)$. Given a set of N demonstrations $\{\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}\}_{t=0,n=1}^{T^n,N}$, an estimate of $\boldsymbol{f}(.)$ can be built using different techniques such as Stable Estimator of Dynamical Systems (SEDS) [3]. Note that throughout this paper we assume the DS $\boldsymbol{f}(.)$ is provided by the user, and henceforth we will call it the *original DS*. Next we describe our DS-based obstacle avoidance.

A. Analytical description of obstacles

Consider a d-dimensional object centered at a reference point $\boldsymbol{\xi}^o$. We denote the position of a point $\boldsymbol{\xi} \in \mathbb{R}^d$ with respect to the frame of reference centered at $\boldsymbol{\xi}^o$ with $\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi} - \boldsymbol{\xi}^o$. Suppose a continuous function $\Gamma(\tilde{\boldsymbol{\xi}})$ that projects $\mathbb{R}^d$ into $\mathbb{R}$. The function $\Gamma(\tilde{\boldsymbol{\xi}})$ has continuous first order partial derivatives (i.e. $\mathcal{C}^1$ smoothness) and increases monotonically with $\|\tilde{\boldsymbol{\xi}}\|$. The level curves of Γ (i.e. $\Gamma(\tilde{\boldsymbol{\xi}}) = c, \forall c \in \mathbb{R}^+$) enclose a convex region. By construction, the following relation holds at the surface of the obstacle:

$$\Gamma(\tilde{\boldsymbol{\xi}}) = 1 \tag{3}$$

For example $\Gamma(\tilde{\boldsymbol{\xi}}) : \sum_{i=1}^{d} (\tilde{\boldsymbol{\xi}}_i / \boldsymbol{a}_i)^2 = 1$ corresponds to a d-dimensional ellipsoid with axis lengths $\boldsymbol{a}_i$. We can divide the space spanned by Γ into three regions $\boldsymbol{\mathcal{X}}^o$, $\boldsymbol{\mathcal{X}}^b$, and $\boldsymbol{\mathcal{X}}^f$ to distinguish between points inside the obstacle, at its boundary, and outside the obstacle respectively:

$$\text{Interior points :} \quad \boldsymbol{\mathcal{X}}^o = \{\boldsymbol{\xi} \in \mathbb{R}^d : \Gamma(\tilde{\boldsymbol{\xi}}) < 1\} \tag{4}$$

$$\text{Boundary points :} \quad \boldsymbol{\mathcal{X}}^b = \{\boldsymbol{\xi} \in \mathbb{R}^d : \Gamma(\tilde{\boldsymbol{\xi}}) = 1\} \tag{5}$$

$$\text{Free region :} \quad \boldsymbol{\mathcal{X}}^f = \{\boldsymbol{\xi} \in \mathbb{R}^d : \Gamma(\tilde{\boldsymbol{\xi}}) > 1\} \tag{6}$$

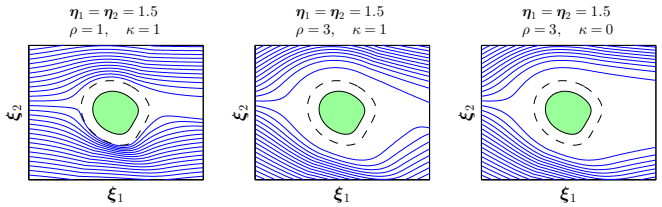
Note that in case of non-convex objects, we could consider a smooth convex envelope (also known as convex bounding volume) that fits tightly around the object. When the point cloud description of an object is available, one could also use one of the estimation techniques such as the one presented in [9] to approximate a $\mathcal{C}^1$ smoothness bounding volume around the object.

B. Modulation

Given the original DS and an analytical formulation describing the surface of K obstacles $\Gamma^1(\boldsymbol{\xi}^{o,1}) \cdots \Gamma^K(\boldsymbol{\xi}^{o,K})$, we would like to determine a combined dynamic modulation matrix $\bar{\boldsymbol{M}}$ so as to instantly modify the robot motion to avoid collision with multiple static obstacles:

$$\dot{\boldsymbol{\xi}} = \bar{\boldsymbol{M}}(\boldsymbol{\xi}) \boldsymbol{f}(.) \tag{7}$$

Algorithm 1 provides the procedure on how to determine $\bar{\boldsymbol{M}}$. For brevity of this paper, we do not describe here the effect of each line in Algorithm 1 on the combined dynamic modulation matrix, and refer interested readers to [1]. However, there are three parameters that are noteworthy: the safety factors $\boldsymbol{\eta}^k$, the reactivities ρ^k, and the tail effects κ^k.



(a) Safety factor **(b)** Reactivity **(c)** Tail effect

Fig. 1: Illustration of the effect of the safety factor, reactivity, and the tail effect on the modulation. In these graphs, the original dynamics is a uniform flow moving from left to right: $\dot{\boldsymbol{\xi}} = [1; 0]$. **(a)** The area between the dashed line and the obstacle boundary is the safety margin. **(b)** By increasing the reactivity to 3, the trajectories now deflect earlier in time and go further out. **(c)** By setting $\kappa = 0$, We could remedy the tendency of the trajectories to follow the obstacle shape after passing it. Note that in this situation, the slight modulation of the trajectories after passing the obstacle is still required in order to ensure the continuity in the velocity.

For each obstacle $k \in 1..K$, the safety factor $\boldsymbol{\eta}^k \in \mathbb{R}^d$ with $\boldsymbol{\eta}_i^k \geq 1, \forall i \in 1..d$, controls the required safety margin around the object by virtually inflating the object along each direction $\tilde{\boldsymbol{\xi}}_i$ with the magnitude $\boldsymbol{\eta}_i^k$ (in the obstacle frame of reference). The reactivity parameter $\rho^k > 0$ controls the responsiveness of the robot to the presence of each obstacle. The larger the reactivity, the earlier the robot responds to the presence of an obstacle. This parameter is very crucial for practical purposes as one may prefer to deflect the robot trajectory earlier when it goes toward a fire flame than when it is just heading towards a soft pillow. The last parameter $\kappa \in \{0, 1\}$ controls whether the robot motion should still be modified after passing the object ($\kappa = 1$ corresponds to this situation). Figure 1 shows the effect of these three factors on the modulation.

Note that the multiplication of the combined dynamic modulation matrix guarantees the impenetrability of all the K obstacles. However in many robot experiments, not only should the robot avoid the obstacle, but it should also reach a target. In other words, we would like the modified motion to preserve the convergence property of the original dynamics while still ensuring that the motion does not collide with the object(s). As described in [1], the multiplication of $\bar{\boldsymbol{M}}$ does not change the critical points of the original dynamics. Hence, if the original DS is globally stable (i.e. all trajectories reach the target point) when there is no obstacle in the robot working space, it also remains stable in the presence of obstacles[2].

III. Extension to Multiple Moving Obstacles

In our previous work [1] we have considered situations where obstacles are static. In this section we extend our previous formulation to perform obstacle avoidance in the presence of multiple moving obstacles with linear and/or rotational velocities. In the presence of one single obstacle, this extension is straight forward and can be achieved by

[2]It should be noted that the modulation term $\boldsymbol{M}(\tilde{\boldsymbol{\xi}})$ may also create other possible equilibrium points at the boundary of obstacles. As these possible equilibrium points *solely* appear on the obstacles' boundary, they can be tackled by using a contouring mechanism.

Algorithm 1 DS-Based obstacle avoidance as described in [1]

Input: $\boldsymbol{\xi}$, $\boldsymbol{f}(.)$, and $\{\boldsymbol{\eta}^k, \rho^k, \kappa^k\}$

1: **for** each obstacle $k, k \in 1..K$ **do**

2: $\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}} = (\boldsymbol{\xi} - \boldsymbol{\xi}^{o,k})./\boldsymbol{\eta}^k$

3: $\boldsymbol{E}^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) = \begin{bmatrix} \boldsymbol{n}^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) & \boldsymbol{e}^{1,k}(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) & \cdots & \boldsymbol{e}^{d-1,k}(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) \end{bmatrix}$

4: $\omega^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) = \begin{cases} 1 & \text{if } K = 1 \\ \prod_{i=1, i\neq k}^{K} \frac{(\Gamma^i(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})-1)}{(\Gamma^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})-1)+(\Gamma^i(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})-1)} & \text{otherwise} \end{cases}$

5: $\begin{cases} \lambda^k_1(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) = \begin{cases} 1 - \frac{\omega^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})}{|\Gamma(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})|^{\frac{1}{\rho}}} & \boldsymbol{n}(\tilde{\boldsymbol{\xi}})^T\dot{\boldsymbol{\xi}} < 0 \ \textbf{or} \ \kappa = 1 \\ 1 & \boldsymbol{n}(\tilde{\boldsymbol{\xi}})^T\dot{\boldsymbol{\xi}} \geq 0 \ \textbf{and} \ \kappa = 0 \end{cases} \\ \lambda^k_i(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) = 1 + \frac{\omega^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})}{|\Gamma(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})|^{\frac{1}{\rho}}} \quad 2 \leq i \leq d \end{cases}$

6: $\boldsymbol{D}(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) = \begin{bmatrix} \lambda^k_1(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \lambda^k_d(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) \end{bmatrix}$

7: $\boldsymbol{M}^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}}) = \boldsymbol{E}^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})\, \boldsymbol{D}^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})\, \left(\boldsymbol{E}^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})\right)^{-1}$

8: **end for**

9: $\bar{\boldsymbol{M}}(\boldsymbol{\xi}) = \prod_{k=1}^{K} \boldsymbol{M}^k(\tilde{\boldsymbol{\xi}}^k_{\boldsymbol{\eta}})$

Output: $\dot{\boldsymbol{\xi}} = \bar{\boldsymbol{M}}(\boldsymbol{\xi})\boldsymbol{f}(.)$

computing the modulation in the obstacle's frame of reference. Suppose an obstacle $\Gamma(\tilde{\boldsymbol{\xi}})$ that is moving with linear and rotational velocities $\dot{\boldsymbol{\xi}}^o_L$ and $\dot{\boldsymbol{\xi}}^o_R$, respectively. The modulated dynamics for obstacle avoidance becomes:

$$\dot{\boldsymbol{\xi}} = \bar{\boldsymbol{M}}(\tilde{\boldsymbol{\xi}})\big(\boldsymbol{f}(.) - \dot{\boldsymbol{\xi}}^o_L - \dot{\boldsymbol{\xi}}^o_R \times \tilde{\boldsymbol{\xi}}\big) + \dot{\boldsymbol{\xi}}^o_L + \dot{\boldsymbol{\xi}}^o_R \times \tilde{\boldsymbol{\xi}} \quad (8)$$

where $(.) \times (.)$ denotes the cross product and $\bar{\boldsymbol{M}}(\tilde{\boldsymbol{\xi}})$ is the modulation given by Algorithm 1. In this equation, the term $\boldsymbol{f}(.) - \dot{\boldsymbol{\xi}}^o_L - \dot{\boldsymbol{\xi}}^o_R \times \tilde{\boldsymbol{\xi}}$ transforms the velocity of the robot to the obstacle's coordinates system. Then, the modulation is performed in this coordinates system where the object is static, yet the robot is moving with a different speed. After applying the modulation, the result is transformed back to the world's frame of reference through the last term.

Equation (8) ensures impenetrability of a single moving obstacle. To verify this, suppose a point $\boldsymbol{\xi}^b$ on the boundary of the moving obstacle at time t. After multiplying the modulation matrix, the radial velocity of the robot is canceled, and hence the robot can only move along the tangential hyperplane at $\boldsymbol{\xi}^b$. However, this is still not enough as the robot may hit the obstacle in the next moment t^+ since the obstacle is moving. The collision can be avoided by adding the instant velocity of the point $\boldsymbol{\xi}^b$ due to obstacle motion, which is given by $\dot{\boldsymbol{\xi}}^o_L + \dot{\boldsymbol{\xi}}^o_R \times (\boldsymbol{\xi}^b - \boldsymbol{\xi}^o)$, to the modulated velocity.

As a side effect, Eq. (8) could induce some unnecessary movements to the robot even when the robot is far from the obstacle (note that the angular velocity grows proportionally with $\|\tilde{\boldsymbol{\xi}}\|$). This can be tackled by adding an exponential term that diminishes the induced velocity due to the obstacle's movement as $\|\tilde{\boldsymbol{\xi}}\|$ increases:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{M}(\tilde{\boldsymbol{\xi}})\Big(\boldsymbol{f}(.) - e^{-\frac{1}{\sigma^o}(\Gamma(\tilde{\boldsymbol{\xi}})-1)}\big(\dot{\boldsymbol{\xi}}^o_L + \dot{\boldsymbol{\xi}}^o_R \times \tilde{\boldsymbol{\xi}}\big)\Big) + \cdots + e^{-\frac{1}{\sigma^o}(\Gamma(\tilde{\boldsymbol{\xi}})-1)}\big(\dot{\boldsymbol{\xi}}^o_L + \dot{\boldsymbol{\xi}}^o_R \times \tilde{\boldsymbol{\xi}}\big) \quad (9)$$

where σ^o is a positive scalar controlling the rate of decay of the exponential term. The higher the σ^o, the earlier the robot responds to the obstacle motion. The above change does not compromise impenetrability of the obstacle as we have $e^{-\frac{1}{\sigma^o}(\Gamma(\tilde{\boldsymbol{\xi}})-1)} = 1$ on the boundary of the obstacle.

In the presence of multiple moving obstacles, further considerations should be taken so that the above transformation smoothly shift from one obstacle to another based on the current position of the robot. To achieve this goal, we use the weighting coefficients that are computed in the 4th line of Algorithm 1 to control the priorities of obstacles.

Let us consider K disconnected obstacles that are described by $\Gamma^k(\tilde{\boldsymbol{\xi}}^k)$, $k \in 1..K$, with associated translational and rotational velocities $\dot{\boldsymbol{\xi}}^{o,k}_L$ and $\dot{\boldsymbol{\xi}}^{o,k}_R$, respectively. We define the net shift in velocity due to the presence of these obstacles as:

$$\bar{\dot{\boldsymbol{\xi}}}^o = \sum_{k=1}^{K} \dot{\boldsymbol{\xi}}^{o,k} = \sum_{k=1}^{K} e^{-\frac{1}{\sigma^{o,k}}(\Gamma^k(\tilde{\boldsymbol{\xi}}^k)-1)}\omega^k(\tilde{\boldsymbol{\xi}}^k)\big(\dot{\boldsymbol{\xi}}^{o,k}_L + \dot{\boldsymbol{\xi}}^{o,k}_R \times \tilde{\boldsymbol{\xi}}^k\big) \quad (10)$$

where $\omega^k(\tilde{\boldsymbol{\xi}}^k)$ are computed according to Algorithm 1. In case the tail effect is not desired (i.e. $\kappa = 0$), one could remove the modulation effect by setting $\dot{\boldsymbol{\xi}}^{o,k} = 0$ for each obstacle that is moving away from the robot (i.e. when $\big(\dot{\boldsymbol{\xi}}^{o,k}\big)^T\tilde{\boldsymbol{\xi}}^{o,k} < 0$).

The combined modulation that considers the net effect of all moving/static obstacles is then given by:

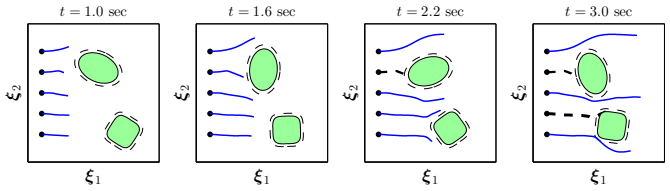
$$\dot{\boldsymbol{\xi}} = \bar{\boldsymbol{M}}(\boldsymbol{\xi})\big(\boldsymbol{f}(.) - \bar{\dot{\boldsymbol{\xi}}}^o\big) + \bar{\dot{\boldsymbol{\xi}}}^o \quad (11)$$

where $\bar{\boldsymbol{M}}(\boldsymbol{\xi})$ is given by Algorithm 1. Equation (11) ensures the impenetrability of all the K obstacles. For a point $\boldsymbol{\xi}^b$ on the boundary of the k-th obstacle, only $\omega^k = 1$ and all the other weighting coefficients are zeros. Hence $\bar{\boldsymbol{M}}(\boldsymbol{\xi}^b) = \boldsymbol{M}^k(\boldsymbol{\xi}^b)$ and $\bar{\dot{\boldsymbol{\xi}}}^o = \dot{\boldsymbol{\xi}}^{o,k}_L + \dot{\boldsymbol{\xi}}^{o,k}_R \times \tilde{\boldsymbol{\xi}}^{b,k}$, and thus the obstacle is impenetrable. Similarly to the static case, by moving from one obstacle to another, the weighting coefficients smoothly change between zero and one, and by this, impenetrability is always ensured for all the obstacles.

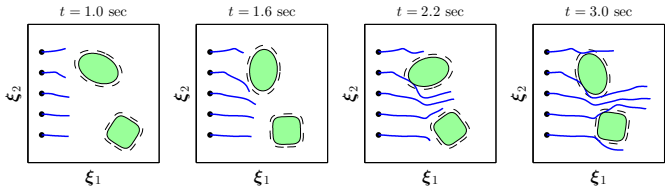
Figure 2 shows an example of obstacle avoidance in the presence of two moving obstacles. It compares two situations: 1) The quasi-static case where the obstacles' motion are neglected, and the modulation is computed at each time based on the instantaneous position and orientation of the obstacles (see Fig. 2a), and 2) The dynamic case where the obstacles' motion are taken into account (see Fig. 2b). As we can see, in the quasi-static case the impenetrability of the obstacles are no longer ensured, whereas in the dynamic case all the trajectories can safely pass the obstacles.

IV. Robot Experiments

In this section we evaluate our approach in the presence of a fast moving obstacle, where the quasi static-assumption is no longer valid. The experiment consisted of having the 7-DoF KUKA DLR arm stay in a default target position while a box is slid towards the robot at high speed. Thus the robot should react quickly and change its position so that the box passes without any collision (see Fig. 3).



(a) Without considering the obstacles' motion (the quasi-static case). The dashed black lines show the failure cases where the robot actually collides with the obstacles.



(b) With considering the obstacles' motion. In this case, collision avoidance for all trajectories is ensured.

Fig. 2: Illustration of the obstacle avoidance in the presence of two moving obstacles. As we can see, solely in the dynamic case, where the obstacles' motion is considered, the trajectories can safely pass the obstacles. In this example, the trajectories move from left to right with $\dot{\boldsymbol{\xi}} = [2; 0]$ m/s. The oval-shaped object has the linear velocity $\dot{\boldsymbol{\xi}}_L^{o,1} = [-0.4; -0.2]$ m/s and the rotational velocity $\dot{\boldsymbol{\xi}}_R^{o,1} = -2$ rad/s. These values for the square-shaped obstacle are $[-0.4; -0.2]$ m/s and 1 rad/sec. Both objects have the safety factor of $\boldsymbol{\eta} = 1.2$. The variance σ^o is set to 2 and 10 for the oval and square-shaped obstacles, respectively.

The KUKA robot is controlled in the Cartesian coordinate system, and the control commands are sent at 1000Hz. We use the damped least square pseudo-inverse kinematics to compute the robot's joint angles. The torque command to the robot is computed based on the desired kinematic command using the KUKA built-in PID controller. The original DS, deriving the robot motion in the absence of obstacles, is modeled using the Stable Estimator of Dynamical Systems (SEDS) [3]. SEDS builds an estimate of a globally asymptotically stable DS from a set of demonstrations (in the absence of obstacles) provided by the user. This DS is then used to control the robot motion by generating velocity commands to keep the robot's end-effector close or, when it is feasible, at the target point.

We define the box reference point at $x^{o,B} = x^{c,B}$, $y^{o,B} = y^{c,B}$, and $z^{o,B} = 0$, and model it with the analytical formulation $\Gamma(\tilde{\boldsymbol{\xi}})^B$: $((x - x^{o,B})/0.055)^2 + ((y - y^{o,B})/0.165)^2 + ((z - z^{o,B})/0.23)^4 = 1$. Other parameters are set as follows: $\boldsymbol{\eta} = [3.5 \;\; 2.0 \;\; 1.5]^T$, $\rho = 2$, $\sigma = 30$, and $\kappa = 0$. The box's position and orientation are tracked at 240Hz using an OptiTrack vision system. We use a Kalman filter to reduce the noise effect on estimations. The working table is modeled with $x^{o,T} = y^{o,T} = 0$, $z^{o,T} = -0.01cm$ and $\Gamma(\tilde{\boldsymbol{\xi}})^T$: $((x - x^{o,T})/3)^6 + ((y - y^{o,T})/3)^6 + ((z - z^{o,T})/0.01)^4 = 1$. We set the safety factor of the table to $\boldsymbol{\eta} = 1.3$. The position of the table is set fixed in the whole experiment.

In total we ran 20 trials, lasting between 0.8 to 1.3 seconds, in which the box was slid from different initial configurations with various linear and angular velocities. In each trial, the box was set to an initial distance of about 0.5 meter away from the robot and was thrusted so as to reach a maximum linear velocity of $0.6 \sim 1.5$ m/s and/or a maximum angular velocity of $40 \sim 120$ deg/s. In 16 out of the 20 trials, the robot successfully managed to dodge the box. Figure 3 shows sequences of the motion for four of the trials. The trajectories of the robot's end-effector and the box, and the magnitude of the box's linear and angular velocities are also illustrated in Fig. 4.

The four failure cases could possibly be due to two factors that are not currently considered in our formulation: 1) The filtering of the object's position and orientation introduces a lag in determining the current linear and angular velocities of the box. In situations where the box is moving and rotating fast at a very close distance to the robot, the presence of this lag could yield collision with the obstacle. 2) The robot's joints cannot move faster than a certain value due to the hardware limitation, and hence collision with the obstacle is inevitable. Figure 5 shows the sequences of the motion for one of the failure cases. In this trial, though the avoidance seems successful at the initial stage of the motion, the box hit the end-effector from the backside due to the wrong estimation of the object's angular velocity.

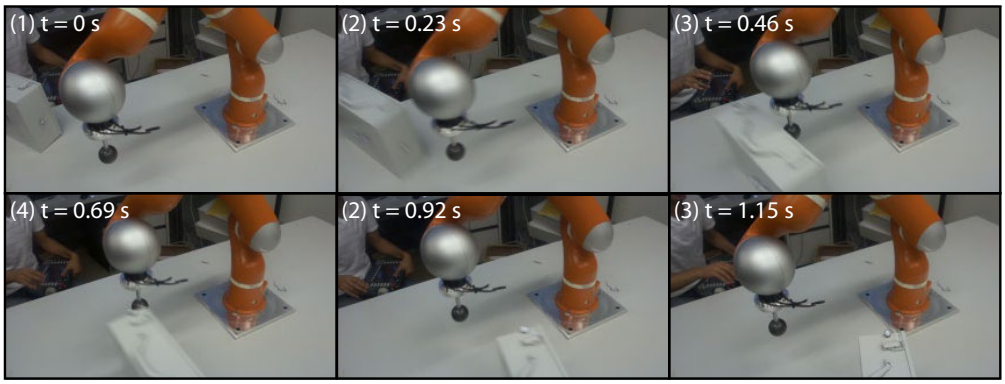
The first factor can be alleviated by using a more advanced filter or by increasing the safety factor. However, the second factor cannot be easily tackled. Some improvements might be achieved by using a planner technique that could take into account such hardware limitations during the path generation. However, as in the above failure situations the obstacle is moving fast at a very close distance to the robot, this planner should be extremely fast to provide a valid solution within an order of millisecond (recall the robot is controlled at 1000Hz).

V. Summary and Conclusion

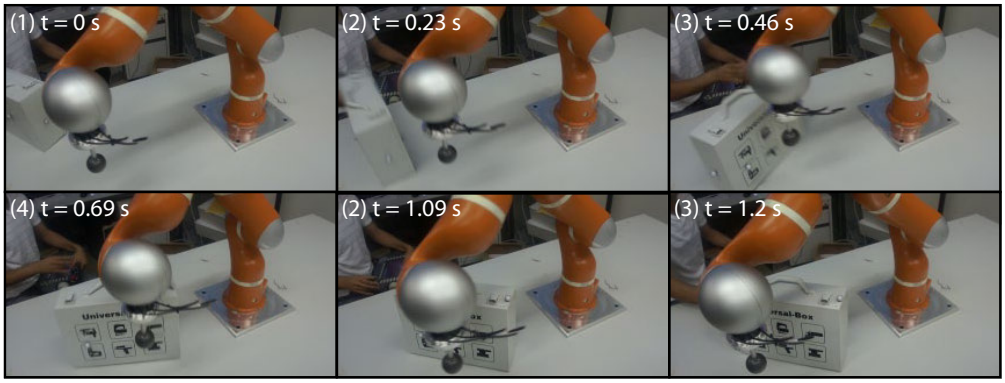
In this paper, we have extended our previous approach to perform obstacle avoidance in the presence of fast moving objects, where the quasi-static assumption no longer holds. The proposed approach requires the user to provide a DS model that governs the robot motion in the absence of obstacles and a smooth analytical formulation describing a convex bounding volume around each obstacle. Given the above information as well as the realtime position and orientation of obstacles, it instantly provides a modulation to the DS model of the motion so as the robot does not collide with the obstacles.

We have validated the applicability of our method in a real robot experiment where a fast moving and rotating box was slid towards the robot at various speeds. In most trials, the robot manages to successfully dodge the box despite its fast motion at a very close distance. Due to high speed motion of the box, the accuracy in estimating its position and orientation play an important role for the safe collision avoidance. In our implementation, there is an upper bound for the maximum amount of inaccuracies that can be handled, which is a function of the safety factor, reactivity parameter, and the object's velocity.

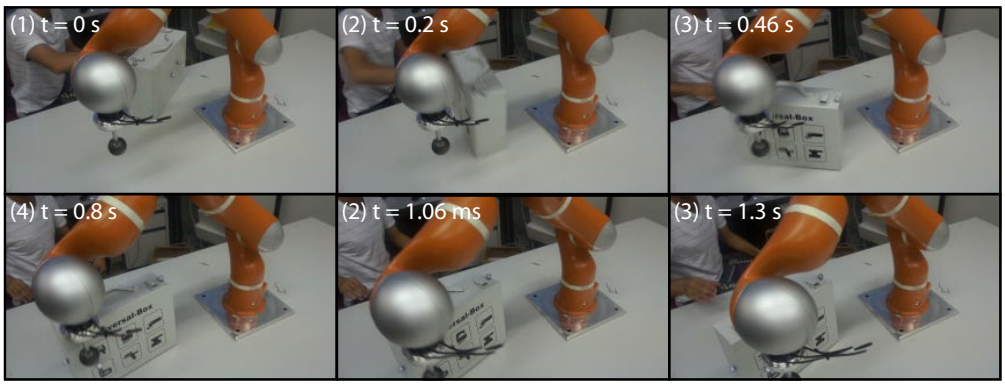
Our approach is currently limited in that it does not consider the robot's hardware limitations during the avoidance. The



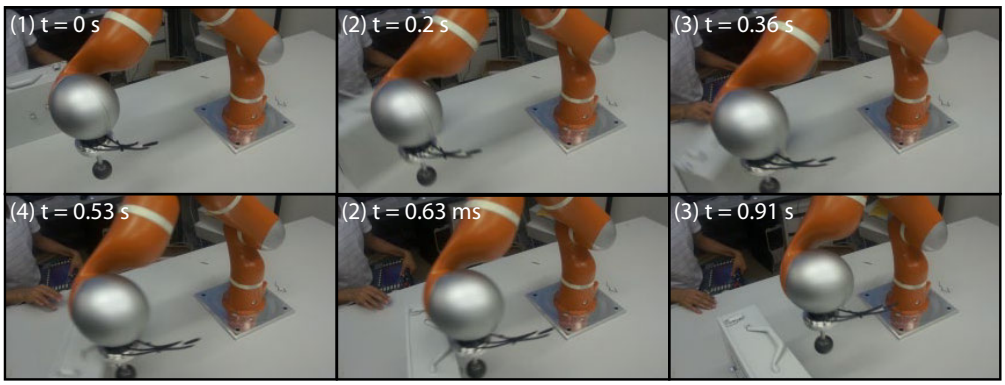
(a) First trial.



(b) Second trial.

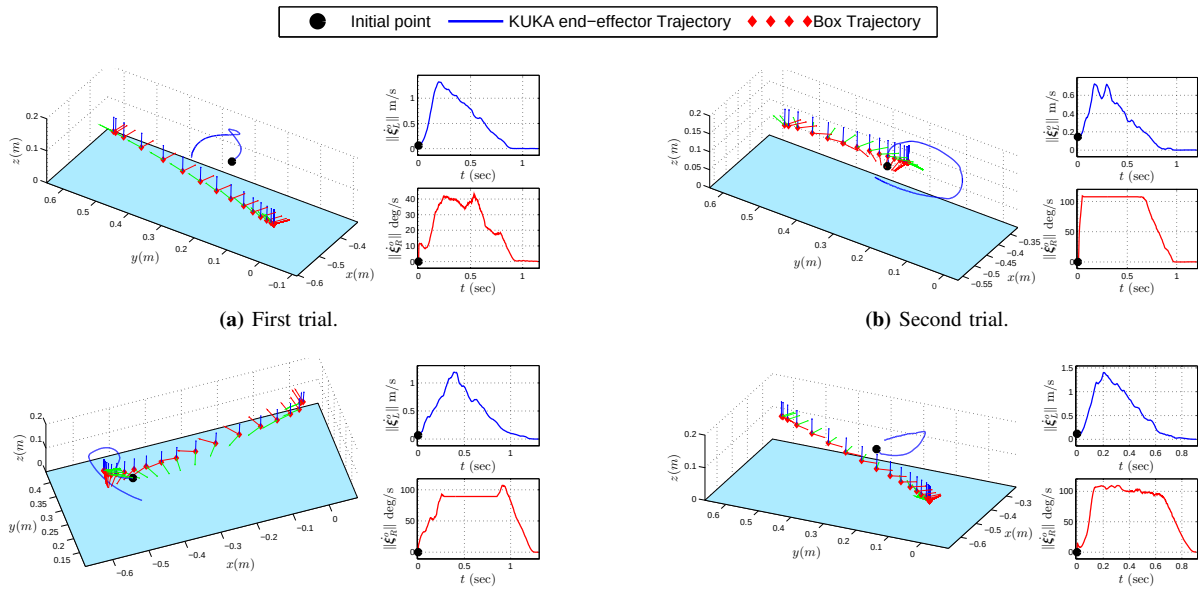


(c) Third trial.



(d) Forth trial.

Fig. 3: Illustration of sequences of motion for 4 out of the 20 executed trials. In this experiment the robot was required to dodge a sliding box that was launched 20 times from different initial configurations with various linear and angular velocities. For further information please refer to Section IV.



(a) First trial. **(b)** Second trial. **(c)** Third trial. **(d)** Forth trial.

Fig. 4: Illustration of trajectories of the robot's end-effector and the box, and the magnitude of the box's linear and angular velocities for the four trials shown in Fig. 3. In these graphs, the x, y, and z axes of the box's frame of reference are shown with red, green, and blue vectors, respectively. For further information please refer to Section IV.

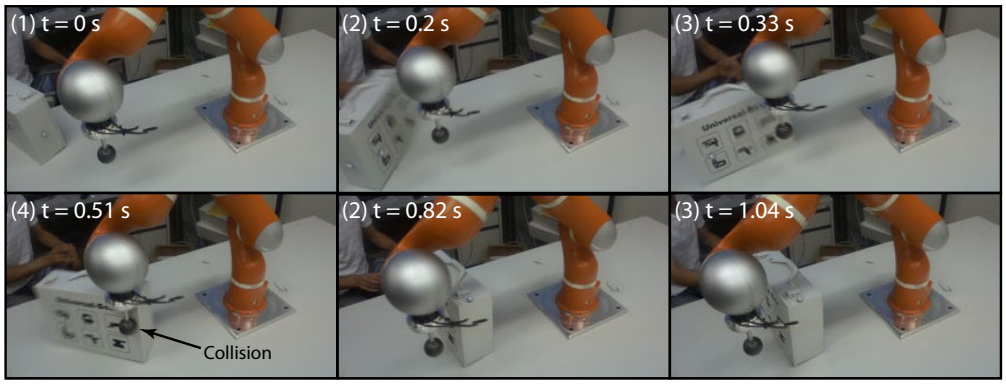


Fig. 5: Illustration of sequences of motion for one of the four cases in which the robot failed to successfully dodge the box.

DS modeling can compensate for deviations (due to hardware limitations) from the desired trajectory, by instantly adapting a new trajectory for the new position of the robot. However, an inevitable outcome of such compensation is that the robot executes the motion at a slowest pace than what is expected, which may yield to collision.

ACKNOWLEDGMENT

This work was supported by the European Commission through the EU Project AMARSI (FP7-ICT-248311).

REFERENCES

[1] S.-M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, pp. 433–454, 2012.

[2] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *Int. Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[3] S. M. Khansari-Zadeh and A. Billard, "Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models," *IEEE Trans. on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[4] S. Calinon, A. Pistillo, and D. G. Caldwell, "Encoding the time and space constraints of a task in explicit-duration hidden Markov model," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[5] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, pp. 1–19, 2012.

[6] P. Kormushev, S. Calinon, R. Saegusa, and G. Metta, "Learning the skill of archery by a humanoid robot iCub," in *Proc. IEEE Int. Conf. on Humanoid Robots (Humanoids)*, 2010, pp. 417–423.

[7] S. M. Khansari-Zadeh, K. Kronander, and A. Billard, "Learning to Play Minigolf: A Dynamical System-based Approach," *Advanced Robotics*, 2012.

[8] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009, pp. 1293–1298.

[9] M. Benallegue, A. Escande, S. Miossec, and A. Kheddar, "Fast C1 Proximity Queries using Support Mapping of Sphere-Torus-Patches Bounding Volumes," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 483–488.