2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Robotic handling of liquids with spilling avoidance: a constraint-based control approach

Riccardo Maderna[1], Andrea Casalino[1], Andrea Maria Zanchettin[1] and Paolo Rocco[1]

Abstract— **Handling liquids with spilling avoidance is a topic of interest for a broad range of fields, both in industry and in service robotic applications. In this paper we present a new control architecture for motion planning of industrial robots, able to tackle the problem of liquid transfer with sloshing control. We do not focus on a complete sloshing suppression, but we show how to enforce an anti spilling constraint. This less conservative approach allows to impose higher accelerations, reducing motion time.**
A constraint-based approach, amenable to an on-line implementation, has been developed. The proposed controller generates trajectories in real time, in order to follow a reference path, while being compliant to the spilling avoidance constraint. The approach has been validated on a 6 degree of freedom industrial ABB robot.

I. Introduction

Liquid sloshing control is a classical control problem in several areas of application. Significant fields are, for instance, motion control for automatic pouring of molten metal in casting industries (see [1] or [2]) or liquid transfer and packaging in chemical and pharmaceutical industries. In such scenarios, it is essential to limit sloshing in order to prevent overflows as well as any deterioration in quality due to contamination, while at the same time shorten the total operational time in order to improve productivity.
The aim of this paper is to propose a new solution to the problem of handling liquids with spilling avoidance using robotic manipulators. This topic is relevant not only in industrial scenarios, but also for service robotics. The desirable control solution has to be efficient, non-invasive, cost-effective and multi-purpose. Specifically, the use of instrumented containers or other specific set-ups must be avoided. Moreover, the liquid must not require modifications, like artificial colouring to ease detection.
Most previous contributions tackle the problem with the aim of achieving complete sloshing suppression. In [1] an open loop H_∞ controller is proposed, with optimal command input computed through off-line optimization techniques. Only rectilinear motions are considered, while [2] extends the same approach to general 3D motions designing independent controllers for each of the three axes, thus not considering the coupling effects that are present in the complete non-linear model of sloshing dynamics.
The difficulties in retrieving reliable measurements of liquid sloshing lead to a predominance of open loop solutions, however few closed-loop schemes can be also found in the literature. [3] describes a sliding mode control approach based on partial feedback linearization. Nevertheless, the paper does not solve the problem of providing sloshing measurements, assuming complete availability of states. In order to overcome this limitation, [4] makes use of an observer to estimate sloshing values, in the absence of direct measurements.
In most contexts it is sufficient to enforce just a spilling avoidance constraint rather than complete suppression of sloshing, achieving faster motion. Nevertheless, only few published works are limited to a spilling avoidance control problem; the most significant is [5]. The paper considers a minimum-time feedforward motion control problem for an open container carrying a liquid. The method relies on linear programming optimization which is solved off-line to compute the input to the system and sticks to horizontal and straight motion.
In [6] the use of a robotic manipulator is explicitly considered to handle the liquid container as it considers a service robotics application (serving beverages to customers). The proposed solution does not consider a mathematical model of sloshing dynamics and only requires liquid natural frequency and transfer model of the container to tune the controller. However, although it is able to achieve good results in terms of sloshing suppression, a significant increase in motion time is needed to compensate for simplicity of tuning. A faster solution can be found in [7], which uses an input shaping approach to transfer a liquid container with complete sloshing suppression along a straight path. Finally, even if it still deals with slosh-free motion, [8] designs an open loop controller to move liquids with a robotic manipulator along generic 3D trajectories. The approach is based on setpoint shaping by means of an exponential filter.
In this paper, a constraint-based control architecture to ensure spilling avoidance will be proposed. The final result is an open-loop controller that works on-line: trajectories are generated in real-time based on information coming from a model of sloshing dynamics; when the spilling avoidance constraint is about to be violated, the robot deviates from the planned motion to ensure sloshing control. The validity of our approach has been tested on a six degree of freedom industrial manipulator (ABB IRB140). In our experiments a depth camera has been used to detect the liquid surface inside the container and to measure the sloshing angle. This camera has been used for offline identification of the sloshing model.
The rest of this article is organized as follows. In Section II we give an overview of the control algorithm, Section III

[1] The authors are with Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza L. Da Vinci 32, 20133, Milano, Italy (e-mail: name.surname@polimi.it).

shows the dynamic model employed to ensure the spilling avoidance constraint. Section IV contains some experimental results while some final considerations are reported in Section V.

II. Motion planning with spilling avoidance

As already discussed in the previous Section, the aim of this work is to design an on-line controller for spilling avoidance, setting an upper limit α_{lim} to the sloshing angle. The control architecture that is proposed in this work, mainly follows from the approach presented in [9], which will be here briefly reviewed. The basic idea is to continuously update the reference trajectory in order to always satisfy the process constraints. Fig. 1 shows the control structure, compared with the standard pipeline approach. In traditional robot programming, see Fig. 1a, trajectories are computed off-line and only on-line evaluated. With this architecture it is difficult and inefficient to react to task-related events, since planned motion can be only adjusted at a slow rate. On the other hand, the proposed method allows for immediate deviations from the planned trajectory based on information coming to an optimization algorithm under the form of time-varying constraints. Still, existing commercial robots usually pose strong limitations when it comes to interface them with external regulators. However, a scheme similar to that sketched in Fig. 1b can be designed to exploit the existing proprietary axis controller of the robot. In this work we make use of an on-line trajectory generation block (OTG) that feeds a constrained optimization algorithm, whose solution at each sampling time provides the actual references to the low-level axis controller. Let $\boldsymbol{x}$ be the vector containing task variables, describing position and orientation of the robot end-effector (in this case the transported vessel). Trajectories are specified and then planned in the task space, since the focus is on the motion of the liquid container.

A closer attention has to be paid to the orientation representation: in fact, for the problem under study, two out of three possible rotary motions are constrained by limiting the container to stay in upright position. ZYZ Euler angles are chosen in order to obtain a convenient description. In particular letting o_{z_0}, o_y and o_{z_1} be the three Euler angles, the orientation is described by a three dimensional vector $\boldsymbol{o}$ where o_{z_0} is the only orientation degree of freedom left:

$$\boldsymbol{o} = \begin{bmatrix} o_{z_0} & o_y & o_{z_1} \end{bmatrix}^T = \begin{bmatrix} o_{z_0} & \pi/2 & \pi \end{bmatrix}^T \qquad (1)$$

The constant values of angles o_y and o_{z_1} are those needed to keep the container vertical according to the adopted convention.

Let $\boldsymbol{q}$ be the vector of joint variables, then the following kinematic equations hold that relate the task space to the joint one:

$$\boldsymbol{x} = f(\boldsymbol{q}) \qquad \dot{\boldsymbol{x}} = J(\boldsymbol{q})\,\dot{\boldsymbol{q}} \qquad \ddot{\boldsymbol{x}} = \dot{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}} + J(\boldsymbol{q})\,\boldsymbol{u} \quad (2)$$

where $J(\boldsymbol{q}) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}$ is the robot task Jacobian, while $\boldsymbol{u}$ represents the joint accelerations, which are considered as inputs for the control law.

At each sampling time the trajectory generator finds a path connecting, in the task space, the actual state to the target state of motion, while considering common constraints, like maximum task velocities $\dot{\boldsymbol{x}}_{max}$ and accelerations $\ddot{\boldsymbol{x}}_{max}$. For this purpose we use the Reflexxes library [10]. The output of the trajectory generator is evaluated at the next sampling time, both in terms of reference position $\boldsymbol{x}^{ref}_{k+1}$ and velocity $\dot{\boldsymbol{x}}^{ref}_{k+1}$. These references are sent to the controller, which then solves the optimization algorithm, whose output consists in the next state of motion in the joint space $(\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1})$ to be sent to the robot axis controller. Clearly, because of the presence of constraints to be fulfilled, these values may be different from those needed to exactly track the setpoints given by the trajectory generator, which could be infeasible at the current time instant. In order to compute the output, the controller finds the joint accelerations $\boldsymbol{u}_k$ that are then integrated to obtain the state of motion. The update rule is the following:

$$\begin{aligned} \boldsymbol{q}_{k+1} &= \boldsymbol{q}_k + T_s\dot{\boldsymbol{q}}_k + 0.5T_s^2\boldsymbol{u}_k \\ \dot{\boldsymbol{q}}_{k+1} &= \dot{\boldsymbol{q}}_k + T_s\boldsymbol{u}_k \end{aligned} \qquad (3)$$

Let $\boldsymbol{e}$ and $\dot{\boldsymbol{e}}$ be the error between the next state of motion and its reference value, thus

$$\begin{aligned} \boldsymbol{e}_{k+1} &= \boldsymbol{x}^{ref}_{k+1} - \boldsymbol{x}_{k+1} \\ \dot{\boldsymbol{e}}_{k+1} &= \dot{\boldsymbol{x}}^{ref}_{k+1} - \dot{\boldsymbol{x}}_{k+1} \end{aligned}$$

The controller computes joint accelerations $\boldsymbol{u}_k$ as the solution of the following constrained optimization problem:

$$\min_{\boldsymbol{u}_k} \left(\frac{1}{2}\boldsymbol{e}^T Q_p \boldsymbol{e} + \frac{1}{2}\dot{\boldsymbol{e}}^T Q_v \dot{\boldsymbol{e}} \right) \qquad (4)$$

subject to

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + T_s J_k \dot{\boldsymbol{q}}_k + 0.5T_s^2(\dot{J}_k\dot{\boldsymbol{q}}_k + J_k\boldsymbol{u}_k) \qquad (4a)$$

$$\dot{\boldsymbol{x}}_{k+1} = J_k\dot{\boldsymbol{q}}_k + T_s(\dot{J}_k\dot{\boldsymbol{q}}_k + J_k\boldsymbol{u}_k) \qquad (4b)$$

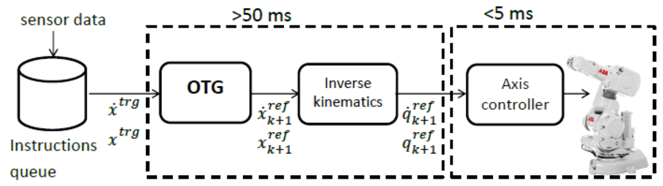
$$\boldsymbol{u}_{inf} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{sup} \qquad (4c)$$

$$-\dot{\boldsymbol{x}}_{max} \leq \dot{\boldsymbol{x}}_{k+1} \leq \dot{\boldsymbol{x}}_{max} \qquad (4d)$$

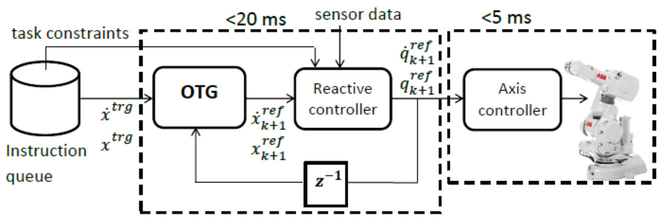
$$-\ddot{\boldsymbol{x}}_{max} \leq \dot{J}_k\dot{\boldsymbol{q}}_k + J_k\boldsymbol{u}_k \leq \ddot{\boldsymbol{x}}_{max} \qquad (4e)$$

$$h_k(\boldsymbol{u}_k) \leq 0 \qquad (4f)$$

where Q_p,Q_v are suitable positive definite weighting matrices for the tracking error, $J_k = J(\boldsymbol{q}_k)$ and $\dot{J}_k = \dot{J}(\boldsymbol{q}_k)$ and T_s is the sampling time. Equations (4a) and (4b) update task position and velocity based on joint velocities and accelerations, while equations (4d) and (4e) describe the bounds set for velocity and acceleration of the task variables, respectively. Constraints (4c) bound the joint accelerations and are introduced to maintain the robot's next state of motion $(\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1})$ within the maximum invariant set $\mathcal{Q}_\infty$. The latter represents the largest region in plane $q - \dot{q}$ such that from any point in the set there exists at least one feasible input acceleration allowing the system to remain within the region itself. Eventually, values $\boldsymbol{u}_{inf}$ and $\boldsymbol{u}_{sup}$ can be computed at each time instant and depend upon robot configuration and joint velocities. Details can be found in [11]. The generic inequalities (4f) include both the constraint



(a) Traditional pipeline architecture.



(b) Reactive controller architecture.

Fig. 1: Comparison between traditional (left) and reactive (right) control structure.

forcing the container to remain in upright position and the constraint that limits the sloshing angle to a maximum possible value. While the former is specified in the following, the latter is described in Section III.

In the definition of the reactive control law, it is worth considering the fact that the on-line generator plans trajectories only for the four dimensional task space. Moreover, objective function and constraints equations can be manipulated in order to obtain a more compact and efficient representation of the minimization problem. Firstly, Jacobian matrix and its derivative can be partitioned as

$$J_k = \begin{bmatrix} J_k^{up} \\ J_k^{dw} \end{bmatrix} \qquad \dot{J}_k = \begin{bmatrix} \dot{J}_k^{up} \\ \dot{J}_k^{dw} \end{bmatrix}$$

where J_k^{up} and $\dot{J}_k^{up}$ are 4-by-6 matrices related to the free coordinates and J_k^{dw} and $\dot{J}_k^{dw}$ are 2-by-6 matrices related to the constrained orientation variables. Secondly, constraints (4a) and (4b) can be substituted in the other inequalities. After some algebraic manipulations the optimization problem is reduced to:

$$\min_{\boldsymbol{u}_k} \left(\frac{1}{2} \boldsymbol{u}_k^T H_k \boldsymbol{u}_k + g_k^T \boldsymbol{u}_k \right) \tag{5}$$

$$A_k \boldsymbol{u}_k \leq b_k \tag{5a}$$

$$h_k(\boldsymbol{u}_k) \leq 0 \tag{5b}$$

where the matrices that define the quadratic cost function are:

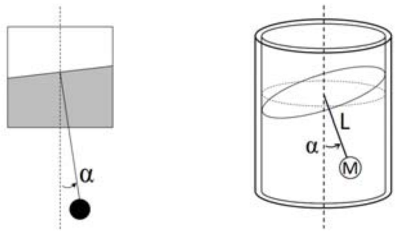
$$H_k = J_k^{up\ T} \left(\frac{T_s^2}{4} Q_p + Q_v \right) J_k^{up}$$

$$g_k = \frac{1}{2} \left[\boldsymbol{x}_k - \boldsymbol{x}_{k+1}^{ref} + T_s \left(J_k^{up} + \frac{T_s}{2} \dot{J}_k^{up} \right) \dot{\boldsymbol{q}}_k \right]^T Q_p J_k^{up} + \\ + \frac{1}{T_s} \left[\left(J_k^{up} + T_s \dot{J}_k^{up} \right) \dot{\boldsymbol{q}}_k - \dot{\boldsymbol{x}}_{k+1}^{ref} \right]^T Q_v J_k^{up}$$

while constraints are defined by:

$$A_k = \begin{bmatrix} J_k^{up} \\ -J_k^{up} \\ J_k^{up} \\ -J_k^{up} \end{bmatrix} \qquad b_k = \frac{1}{T_s} \begin{bmatrix} \dot{\boldsymbol{x}}_{max} - \left(J_k^{up} + T_s \dot{J}_k^{up} \right) \dot{\boldsymbol{q}}_k \\ \dot{\boldsymbol{x}}_{max} + \left(J_k^{up} + T_s \dot{J}_k^{up} \right) \dot{\boldsymbol{q}}_k \\ \ddot{\boldsymbol{x}}_{max} - T_s \dot{J}_k^{up} \dot{\boldsymbol{q}}_k \\ \ddot{\boldsymbol{x}}_{max} + T_s \dot{J}_k^{up} \dot{\boldsymbol{q}}_k \end{bmatrix}$$

In all the above equations task variables $\boldsymbol{x}$ and their derivatives are intended as four dimensional, as they are composed of just the three position variables and the free orientation



(a) (b)

Fig. 2: Planar pendulum (a) and spherical pendulum based model (b) for liquid sloshing.

one. In fact, as already mentioned, it is not necessary to explicitly take into account the other two orientation variables, as they are anyway constrained to specific and constant values in order to keep the liquid container vertical.

Finally, the additional constraints in (5b) still need to be fully specified. Regarding the orientation constraint, recalling the expression (1), the update rule in (4a) and letting $o^c = [o_y \quad o_{z_1}]^T$ we have that:

$$o_{k+1}^c = o_k^c = \left[\pi/2 \quad -\pi \right]^T$$

$$o_{k+1}^c = o_k^c + T_s J_k^{dw} \dot{\boldsymbol{q}}_k + \frac{T_s^2}{2} (\dot{J}_k^{dw} \dot{\boldsymbol{q}}_k + J_k^{dw} \boldsymbol{u}_k)$$

hence, after some manipulation we obtain

$$J_k^{dw} \boldsymbol{u}_k = -\frac{1}{T_s} \left(2 J_k^{dw} + T_s \dot{J}_k^{dw} \right)$$

$$A_k^o \boldsymbol{u}_k = b_k^o$$

which finally defines the orientation constraint.

III. Sloshing dynamic model

This Section deals with the mathematical definition of spilling avoidance constraint. At first we will model the dynamics of a liquid into a container. The need for this kind of a model is twofold. Firstly, the derivation of the constraint that avoids spilling is obviously based on the knowledge of the sloshing behaviour; secondly, information about the state of the fluid surface are required at each sampling time by the reactive controller to evaluate such constraint in order to include it in the optimization algorithm.

A rigorous description of the liquid sloshing dynamics should be based on the Navier-Stokes equations, which is a set of non-linear partial differential equations. These equations describe the complete fluid motion as a superimposition

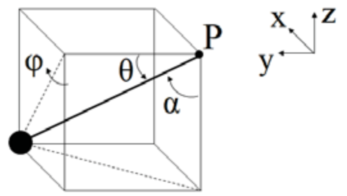


Fig. 3: Pendulum reference frame and angles.

of several sloshing modes at different oscillation frequencies. However, the resulting model would be uselessly complex both for control purposes and to derive the spilling avoidance constraint. Furthermore, it would be also difficult to correctly identify all the necessary parameters, which take into account both liquid and vessel characteristics. On the other hand, there is a general agreement (see [5], [3] or [8]) that for sloshing control it is sufficient to take into account only the first asymmetrical mode, that is the one with the lowest frequency, while neglecting all higher frequency modes of oscillation. With this approximation the model of the moving liquid surface is equivalent to that of a damped pendulum whose rod direction is always orthogonal to the liquid plane (see Fig. 2a). Since we consider a generic 3D motion of container, a spherical pendulum must be considered [8] as sketched in Fig. 2b, characterized by the rod length L, a damping coefficient D and its mass M, which is supposed to be concentrated at one end of the rod. Fig. 3 shows the selected reference frame; the position of the point mass is then described by $[x_M \quad y_M \quad z_M]^T$, while the fulcrum position is described by $[x_P \quad y_P \quad z_P]^T$. The inputs for the model are the accelerations of the pendulum fulcrum P. Moreover, angle θ is defined as the angular distance of the rod from the x-y plane and it is zero when the pendulum is in the horizontal position; ϕ is the angle between the pendulum and the y-z plane and it is equal to zero when the pendulum is vertical. Let $\boldsymbol{p}$ be the vector of the state variables for the model, then:

$$\boldsymbol{p} = \begin{bmatrix}\theta & \dot\theta & \phi & \dot\phi\end{bmatrix}^T \tag{6}$$

The sloshing angle α is the angle between the pendulum and the z axis and it is related with state variables through the expressions:

$$\begin{aligned}\cos(\alpha) &= \sin(\theta)\cos(\phi)\\ \sin(\alpha) &= \sqrt{\cos^2(\theta)+\sin^2(\phi)}\end{aligned} \tag{7}$$

For limited values of the angles (that is an acceptable assumption when dealing with spilling avoidance), $\sin(\alpha)$ can be approximated with α itself, obtaining:

$$\alpha \approx \sqrt{\delta\theta^2 + \delta\phi^2}$$

As it always holds that $\alpha > 0$, the above equation is also equivalent to :

$$\alpha^2 \approx \delta\theta^2 + \delta\phi^2 \tag{8}$$

The equation of motion of the pendulum can be derived using the Euler-Lagrange method, obtaining:

$$\begin{cases} L^2M\ddot\theta + DL^2\dot\theta - \frac{1}{2}L^2MS_{2\theta}\dot\phi^2 + \\ -LMgC_\phi C_\theta + DLC_\theta S_\phi \dot x_P - DLS_\theta \dot y_P + \\ -DLC_\phi C_\theta \dot z_P + LMC_\theta + S_\phi \ddot x_P + \\ -LMS_\theta \ddot y_P - LMC_\phi C_\theta \ddot z_P = 0 \\ \\ LMS_\theta \ddot\phi + DLS_\theta \dot\phi + LS_\theta(MgS_\phi + \\ +2LMC_\theta \dot\phi\dot\theta) + DC_\phi \dot x_P + DS_\phi \dot z_P + \\ +MC_\phi \ddot x_P + MS_\phi \ddot z_P = 0 \end{cases} \tag{9}$$

where the operator $S_.$ stand for $\sin(.)$ and similarly $C_.$ for $\cos(.)$. For control purposes, we would like to derive a simple expression for the constraint on spilling avoidance. With this aim, it is convenient to consider the linearized model around the vertical stable equilibrium $\bar{\boldsymbol{p}} = [\pi/2 \quad 0 \quad 0 \quad 0]^T$, so that the new state variables become $\delta\boldsymbol{p} = \boldsymbol{p} - \bar{\boldsymbol{p}}$ and the approximate system reduces to:

$$\begin{cases} \delta\ddot\theta = -\frac{D}{M}\delta\dot\theta - \frac{g}{L}\delta\theta + \frac{1}{L}\ddot y_P \\ \delta\ddot\phi = -\frac{D}{M}\delta\dot\phi - \frac{g}{L}\delta\phi - \frac{1}{L}\ddot x_P \end{cases} \tag{10}$$

As can be seen from the last equation, the linearization around the equilibrium point leads to the motion equation of two simple independent pendula.

The system input-output transfer function results in a second-order MIMO system in the form (after removing $\ddot z_P$ from the input list):

$$G_P(s) = \begin{bmatrix} 0 & G(s) \\ -G(s) & 0 \end{bmatrix} \quad G(s) = \frac{\mu\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

where:

$$\omega_n = \sqrt{\frac{g}{L}} \qquad \xi = \frac{D}{2\omega_n M} \qquad \mu = \frac{1}{g}$$

ω_n is the natural frequency, ξ the damping ratio and μ the gain, which are all related with the pendulum parameters.

We now explain how to enforce a spilling avoidance constraint. The idea is to ensure that at each time instant t_0 it is possible to find a value for the manipulator joint accelerations $\boldsymbol{u}$ such that the sloshing angle α remains less or equal to the established limit α_{lim} for the future evolution of the system, that is:

$$\forall t_0 \quad \exists \boldsymbol{u}(t_0) : \alpha(t) \le \alpha_{lim} \quad \forall t > t_0$$

Since the controller works with a certain sampling time, we have to consider the time response of the system to piecewise constant accelerations. Therefore, we are interested in considering the time response of the pendulum to a discrete impulse having a width of a sampling period T_s, which reads as:

$$\begin{bmatrix}\ddot x_P & \ddot y_P\end{bmatrix}^T = \begin{bmatrix}X & Y\end{bmatrix}^T(\text{step}(t) - \text{step}(t - T_s)) \tag{11}$$

where X and Y stand for the impulse amplitude values.

The time response of the system can be expressed as [1]

$$\delta\theta(t) = \frac{e^{-t\omega_n\xi}}{\sqrt{1-\xi^2}}[\Theta_1 \cos(t\omega_n\sqrt{1-\xi^2}) + \\ +\Theta_2 \sin(t\omega_n\sqrt{1-\xi^2})] \qquad (12)$$

where $\Theta_1 = \Theta_1(\omega_n, \xi, \mu, T_s, \theta_0, \dot{\theta}_0, Y)$ and $\Theta_2 = \Theta_2(\omega_n, \xi, \mu, T_s, \theta_0, \dot{\theta}_0, Y)$ are highly nonlinear functions in the parameters. It is not difficult to show that an upper-bound $\delta\theta_{max}(t)$ of $\delta\theta(t)$, can be written as:

$$\delta\theta_{max} = \sqrt{\frac{\Theta_1^2 + \Theta_2^2}{1-\xi^2}}$$

It is possible then to rewrite the expression of $\delta\theta_{max}$ highlighting the dependency from Y. Moreover, following identical steps the same expression related to $\delta\phi$ is also obtained:

$$\begin{aligned} \delta\theta_{max} &= \sqrt{t_0 + t_1 Y + t_2 Y^2} \\ \delta\phi_{max} &= \sqrt{p_0 + p_1 X + p_2 X^2} \end{aligned} \qquad (13)$$

where the coefficients t_0, t_1, t_2, p_0, p_1, p_2 are all functions of ω_n, ξ, μ, T_s and initial conditions θ_0, $\dot{\theta}_0$, ϕ_0 and $\dot{\phi}_0$. Now, exploiting eq. (8) we can impose the following spilling avoidance constraint:

$$\alpha_{max}{}^2 = \theta_{max}{}^2 + \phi_{max}{}^2 \le \alpha_{lim}{}^2$$

Manipulating the above expression we obtain:

$$p_0 + t_0 + \begin{bmatrix} p_1 & t_1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} X & Y \end{bmatrix} \begin{bmatrix} p_2 & 0 \\ 0 & t_2 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} \le \alpha_{lim}^2 \qquad (14)$$

where the matrix defining the quadratic part can be easily proved to be positive definite by checking that coefficients $t_2 = t_2(\omega_n, \xi, \mu, T_s, \theta_0, \dot{\theta}_0)$ and $p_2 = p_2(\omega_n, \xi, \mu, T_s, \theta_0, \dot{\theta}_0)$ can take only positive values. This implies that the spilling avoidance constraint is convex. Indeed the overall optimization problem solved by the controller has a quadratic cost function with both linear inequalities constraints and quadratic convex constraint, for which many solvers exist, like for example [12]. However, solving a quadratically constrained optimization problem typically requires a time higher than the robot control cycle. For this reason we adjusted the control scheme so that control commands $\boldsymbol{u}$ are computed at a slower rate, whereas next state of motions are updated at each cycle according to equation (3).

IV. Experiments

This Section presents the results obtained applying the developed control architecture to a real robotic system. In Fig. 4 it is possible to see all equipment employed. In the proposed experiments, a six degree of freedom industrial manipulator (the IRB140 of ABB) has to handle a container full of milk. The scene is monitored by a Microsoft Kinect v2 sensor. The robot is equipped with an IRC5 control unit, that provides all functions for motion control. The controller has been extended to allow interface with an external computer.

[1]A similar expression can be derived for $\delta\phi(t)$.

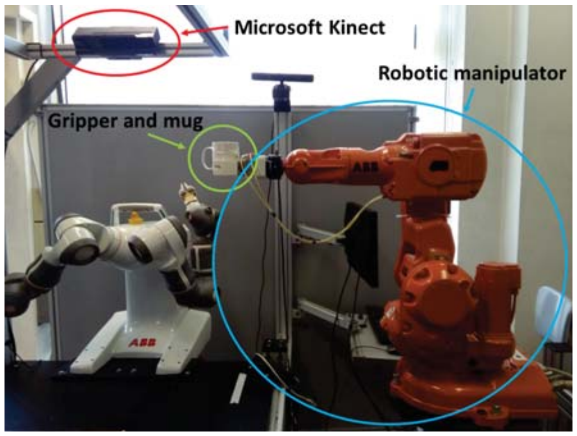


Fig. 4: Robotic cell environment

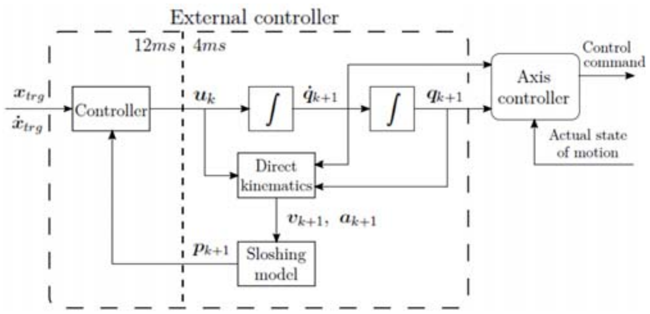


Fig. 5: The complete open-loop strategy. The optimization problem is solved by the controller block.

Accelerations computed by the algorithm described so far, are sent by the external computer to the axis controller of the IRB140. In the above experiments, the accelerations was computed with a sampling time T_s equal to 12 ms.

An open-loop control strategy was adopted (see Fig. 5). The sloshing dynamics model (eq. 9) is exploited to keep track of the state of the system $\boldsymbol{p}$, from an initial known state $\boldsymbol{p}_0$ (that is the state of the system when the experiment starts). Then the value of $\boldsymbol{p}$ is exploited to compute on-line the accelerations sent to the manipulator.

Measures perceived from the vision sensor are here used to estimate the real value of the sloshing angle during time, with the aim of validate our control strategy (the α measured response will be compared with the one estimated using eq. 9). Indeed it is possible to extract from a depth map a point cloud describing the position of some points on the surface of the liquid. Then it is possible to interpolate this set of points with a plane, which can be used as a reference to compute the sloshing angle. We do not get into detail regarding this matter, but one can refer to [13], [14].

A series of spilling avoidance experiments has been conducted using a glass full of milk as a container. In a prior stage an identification of the model parameters of liquid sloshing dynamics was done, with the aim of estimating natural frequency and damping ratio of milk. In all the experiments the robot starts and ends in the same positions, even though for every single run a different value for the sloshing

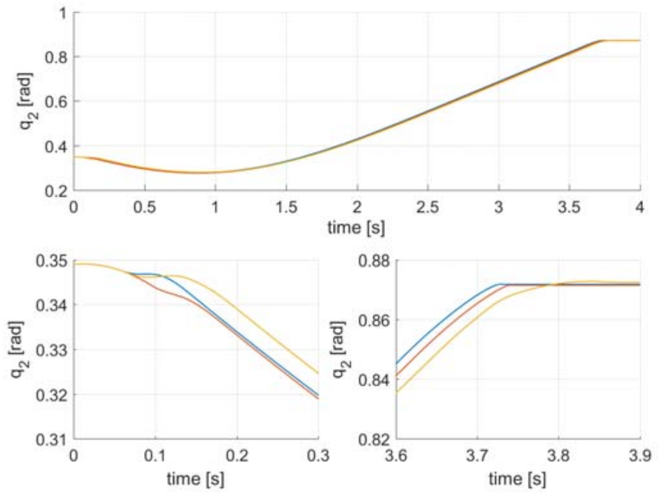


Fig. 6: Comparison of trajectories of joint 2 for different values of α_{lim}: ∞ (blue), $15°$ (red) and $10°$ (yellow). Detail of start (bottom left) and stop (bottom right) of motion.

bound α_{lim} was imposed. Three cases were considered for α_{lim}: ∞ (no limit), $15°$ and $10°$. It is possible to perform the experiment with no limit to α_{lim} since conservative bounds on the maximum robot joint velocities have been set making sure that the liquid never spills out the container. Fig. 7 shows frames taken from a video of the robot for one of the experimental case proposed. In Fig. 7a robot is in the initial position and the liquid is still, then the robot starts moving (Fig. 7b) causing milk sloshing. Motion continues until the industrial manipulator reaches the target configuration and stops (see Fig. 7e). Finally, the liquid undergoes a free motion until it gets to the rest position (Fig. 7f).

Fig. 8 shows the time responses of α, comparing the measured ones with the ones predicted by the sloshing model dynamics. Observing all plots in Fig. 8, we can claim that the spilling avoidance constraint is effectively enforced. However, some exceptions are present, that are probably due to measurement errors. It is possible to note that, after the end of motion, the measured value for the sloshing angle differs significantly from the predicted one in all tests. The reason lies in vibrations transmitted by robot that arise at the moment when the motion stops, due to joints elasticity. Such vibrations are not modeled, thus a limit exists to the minimum value of α_{lim} that can be satisfied in practice.

The proposed control strategy is able to meet the spilling avoidance constraint without significantly affecting the tracking of the reference trajectory. As an example, Fig. 6 shows a typical result obtained for one of the robot joints for different values of α_{lim}. As expected, differences are mainly located in the initial and final part of motion, when the spilling avoidance constraint activates reducing accelerations.

V. Conclusion

The aim of this work was to design a controller for the problem of handling liquids with spilling avoidance using a robotic manipulator. The proposed solution relies on an open-loop control algorithm based on constrained control: trajectories are generated step by step according to information coming from a model of sloshing dynamics. Afterwards, the control law has been implemented on a real system, which simulates an industrial robotic cell, in order to validate the algorithm and check performance. Experimental results showed that the controller is able to prevent the liquid from spilling out of the container and to cope with different maximum sloshing bounds. One natural extension to our work is to consider a closed-loop control strategy. Indeed for this case we can retrieve on-line information from sensors, which can be then used to compute an estimate $\tilde{\boldsymbol{p}}$ of the state relating to sloshing dynamics (through the usage of an observer). Then control accelerations can be computed according to $\tilde{\boldsymbol{p}}$. This can clearly compensate for imprecisions in the system parameter identification stage.

References

[1] K. Yano and K. Terashima, "Robust liquid container transfer control for complete sloshing suppression," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 3, pp. 483–493, May 2001.

[2] ——, "Sloshing suppression control of liquid transfer systems considering a 3-d transfer path," *IEEE/ASME Transactions on Mechatronics*, vol. 10, no. 1, pp. 8–16, Feb 2005.

[3] S. Kurode, B. Bandyopadhyay, and P. S. Gandhi, "Sliding mode control for slosh-free motion of a container using partial feedback linearization," in *2008 International Workshop on Variable Structure Systems*, June 2008, pp. 367–372.

[4] B. Bandyopadhyay, P. S. Gandhi, and S. Kurode, "Sliding mode observer based sliding mode controller for slosh-free motion through pid scheme," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 9, pp. 3432–3442, Sept 2009.

[5] L. Consolini, A. Costalunga, A. Piazzi, and M. Vezzosi, "Minimum-time feedforward control of an open liquid container," in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Nov 2013, pp. 3592–3597.

[6] Y. Komoguchi, M. Kunieda, and K. Yano, "Liquid handling control for service robot by hybrid shape approach," in *2008 SICE Annual Conference*, Aug 2008, pp. 1737–1740.

[7] W. Aribowo, T. Yamashita, K. Terashima, and H. Kitagawa, "Input shaping control to suppress sloshing on liquid container transfer using multi-joint robot arm," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 3489–3494.

[8] L. Moriello, L. Biagiotti, C. Melchiorri, and A. Paoli, "Control of liquid handling robotic systems: A feed-forward approach to suppress sloshing," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4286–4291.

[9] A. M. Zanchettin and P. Rocco, "Motion planning for robotic manipulators using robust constrained control," *Control Engineering Practice*, vol. 59, pp. 127 – 136, 2017.

[10] T. Kroeger, "Opening the door to new sensor-based robot applications - the reflexxes motion libraries," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4.

[11] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[12] "IBM ILOG CPLEX Optimizer," urlhttp://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, Last 2010.

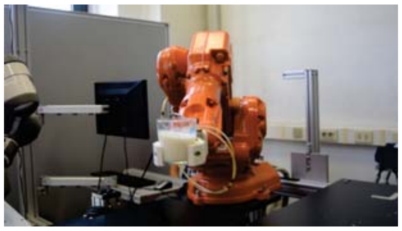
[13] C. Do, T. Schubert, and W. Burgard, "A probabilistic approach to liquid level detection in cups using an rgb-d camera," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2075–2080.

[14] Y. Hara, F. Honda, T. Tsubouchi, and A. Ohya, "Detection of liquids in cups based on the refraction of light with a depth camera using triangulation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 5049–5055.

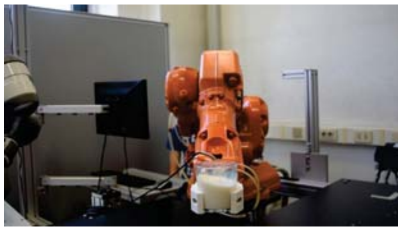
(a)

(b)

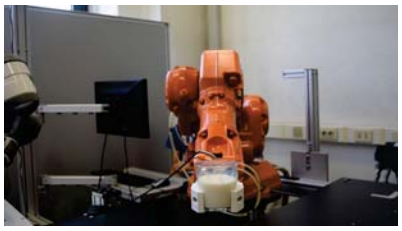
(c)



(d)

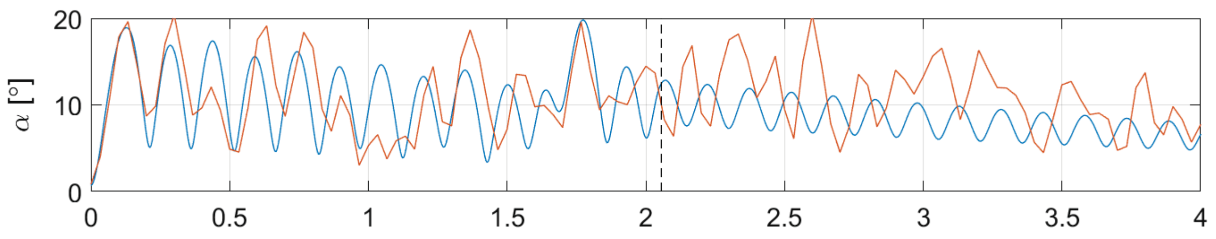


(e)

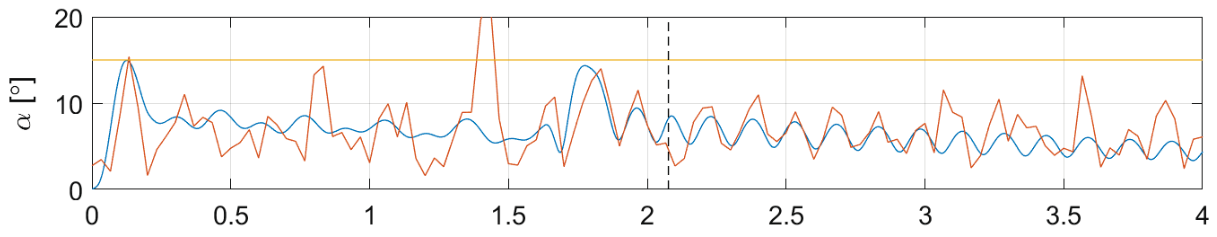


(f)

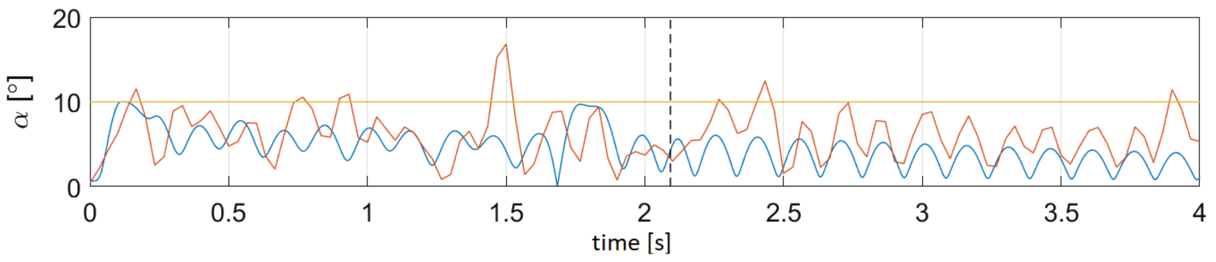
Fig. 7: Screenshoots from video of experiment with robot holding glass and $\alpha_{lim} = 15°$.



(a) Sloshing angle with $\alpha_{lim} = \infty$



(b) Sloshing angle with $\alpha_{lim} = 15°$



(c) Sloshing angle with $\alpha_{lim} = 10°$

Fig. 8: Experimental results. In all the above figures, the red curve is the value perceived from sensor, while in blue the one related to sloshing dynamics model. Yellow curve is the imposed limit angle, while dashed black vertical line indicate when robot has ended its motion.