POLITECNICO DI MILANO

School of Industrial and Information Engineering

Master of Science in Automation and Control Engineering

POLITECNICO
MILANO 1863

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

Supervisor: Prof. Andrea Maria ZANCHETTIN

Co-supervisor: Prof. Paolo ROCCO

Author:

Riccardo MADERNA ID: 850231

Academic Year: 2016 – 2017

To my mother

Acknowledgments

I would like to thank Prof. Paolo Rocco, who gave me the opportunity to work on this challenging thesis, and Prof. Andrea Maria Zanchettin, who followed and guided me during these months. Thanks also to Renzo, who helped me save a lot of time with his tutorials.

I would also like to say thank you to all my fellow students and in particular to Luca. We shared infinite back and forth trips by train, connections in Gallarate, apprehensions and qualms – until the very end.

A special thank to my family, that supports (and bears) me in everything I do, not without sacrifices. I may not always seem really grateful, but you all know this is not the truth. Finally, a thank with love to Raffaella for being always present, also when distances separate us. If I became as I am now, it is also your credit. . . or fault.

Milan, October 2017 R. M.

Your mind is like this water my friend,
when it is agitated it becomes difficult to see.
But if you allow it to settle,
the answer becomes clear.
(Master Oogway)

Contents

List of Figures

List of Tables

Abstract

Continuous evolution of the current industrial scenario forces companies to be more and more competitive. Therefore, efficiency, flexibility and cost reduction in manufacturing processes are key aspects for success. Advancements in industrial robotics are providing companies with powerful tools that can be applied in several areas to achieve the aforementioned objectives. One of those is the task of handling liquids with spilling avoidance, which is a topic of interest for a broad range of fields, both in industry and in service robotics applications.

This thesis presents a new control architecture to tackle the problem of liquid transfer with sloshing control. Specifically, a constraint-based open loop controller that works online is developed. The result is an efficient, non-invasive and cost-effective solution that has been tested on a real robotic system. The controller generates trajectories in real time, then control commands are computed at each time instant in order to follow the reference while being compliant to task constraints, such as the spilling avoidance one. Sloshing information to the controller come from a model of sloshing dynamics, whereas a depth camera has been used to retrieve measurements of liquid slosh during experiments in order to validate the control algorithm.

Keywords: Liquid handling, Constraint-based control, Optical sensors, Robotics

Sommario

La continua evoluzione del panorama industriale spinge le aziende ad essere sempre più competitive. Per questo motivo l'efficienza, la flessibilità e la riduzione dei costi sono aspetti chiave per il successo. Gli avanzamenti nel campo della robotica industriale forniscono alle aziende strumenti che possono essere sfruttati in diversi ambiti di applicazione per raggiungere questi obiettivi. Uno di questi è la movimentazione sicura dei liquidi, che è un problema di interesse per un ampio spettro di campi, sia nell'industria che in applicazioni di robotica di servizio.

Questo elaborato presenta un nuovo schema di controllo per affrontare il problema del trasporto di liquidi con controllo dello sloshing, ovvero l'oscillazione della superficie libera del liquido. In particolare, si è sviluppato un controllore in linea e ad anello aperto basato sulla teoria del controllo vincolato. Il risultato è una soluzione efficiente, non invasiva e non costosa che è stata poi testata su un reale sistema robotico. Il controllore genera le traiettorie in tempo reale, quindi le variabili di comando sono calcolate ad ogni istante di tempo in modo da seguire il riferimento e contemporaneamente soddisfare i vincoli, come quello di evitare il versamento del liquido. Le informazioni sull'oscillazione della superficie liquida arrivano al controllore da un modello delle dinamiche del fluido, mentre un sensore di profondità è stato usato per misurare l'angolo durante gli esperimenti condotti in modo da poter validare l'algoritmo di controllo.

Parole chiave: Trasporto di liquidi, Controllo vincolato, Sensori ottici, Robotica

Chapter 1

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 or liquid transfer and packaging in chemical and pharmaceutical industries. Moreover, liquid container transfer systems have been developed to replace pipelines in the beverage and food industries. In all these cases, it is essential to avoid sloshing in order to prevent overflows as well as any deterioration in quality due to contamination. It is also important to shorten the total operational time in order to improve productivity.

The proliferation of works on sloshing control is determined by the fact that it is a challenging problem. This is mainly due to the difficulties of measuring the liquid slosh. To overcome this limitation, passive methods were traditionally used to dissipate energy, such as slosh absorbers and baffles, but with the drawback of adding complexity, weight, and cost to the system. Moreover, classical automated solutions for liquid transfer use conveyor belts to slowly move containers along straight paths. On the other hand, in the present industrial scenario companies need to become more and more competitive. To do so, high production performance and flexibility, as well as cost reduction, are crucial. In this context, advancement in industrial robotics is playing a key role to provide companies with an adaptable and powerful tool capable of enhancing efficiency and flexibility of manufacturing processes. In the focus of handling liquids, the use of an industrial manipulator also enables fast and 3D motions without dedicated extra-hardware.

1.1 Thesis objectives

In the light of the just described framework, the aim of this thesis is to propose a new solution to the problem of handling liquids with spilling avoidance using a robotic manipulator. Despite the presence of several works on sloshing control, the contribution of the presented work is not deemed to be useless or redundant, as it deals with the problem from a different perspective, which takes into account the newest developments of the robotic field. In particular, we make reference to the growing interest in smart robotic cells in industry and the recently born service robots applications, such as household or waiter robots. Therefore, the following objectives for control design have been considered: the desired 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.

In order to satisfy all requirements, a constraint-based control architecture is developed in this thesis. The final result is an open-loop controller that works online: 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. A characteristic feature of the algorithm is that the purpose is not that of complete sloshing suppression, but only to prevent the liquid to spill out the container. The latter is a less conservative constraint, hence allowing for faster motions that increase productivity. The control law has been then implemented on a real system, which simulates an industrial robotic cell, in order to validate the algorithm and check performance. In order to do so a depth sensor has been used to detect the liquid surface inside the container and to measure the sloshing angle during experiments. Eventually, the proposed controller naturally suits a future extension to a closed-loop scheme while remaining compliant to the aforementioned objectives.

1.2 Chapters organization

The remainder of this thesis work is organized as follows:

The second chapter gives an overview of the state of the art, in particular about sloshing suppression methods and about solutions to the liquid detection problem.

The third chapter presents the solution adopted to measure the sloshing angle of a liquid inside a container. Reasons for the choice of sensors are discussed, together with the details of all steps of the sloshing angle extraction procedure.

The fourth chapter describes the proposed control strategy, both in term of control architecture and control law. It explains what is intended with constraint-based control and presents a model for the liquid sloshing dynamics that is needed to derive the spilling avoidance constraint.

The fifth chapter shows the results of simulations and experiments with the aim of validating the control algorithm and verifying performance.

The last chapter draws the conclusions of the whole thesis, analysing objectives given the experimental results. Moreover, it proposes possible future work and developments.

Chapter 2

State Of The Art

The aim of this chapter is to give a brief overview of the state of the art of related topics before going into details of the control architecture proposed in this thesis. Hence, a literature survey regarding sloshing control and liquid detection methods is presented in the following.

2.1 Sloshing control

Scientific papers about liquid container transfer with sloshing control are spread at least along the last two decades. Most contributions tackle the problem with the aim of achieving complete sloshing suppression. Examples come, for instance, from the field of casting and steel industry [27] [28], where the objective is to prevent the molten metal to rapidly cool or be contaminated by external impurities while preserving productivity. In [27] an open loop H_∞ controller based on nominal model of sloshing dynamics is proposed, with optimal command input computed through off-line optimization techniques. Only rectilinear motions are considered and the aim is to minimize end point residual vibrations. To achieve complete sloshing suppression another degree of freedom is added to the system, namely a rotational motion of the container. However, vessel rotation may lead to unexpected out of plane oscillations of liquid surface.

The contribution of [28] extends the previous work to 3D motions. In this case controllers are independently designed for each of the three axes, thus not considering the coupling effects that are present in the complete non-linear model of sloshing dynamics. Moreover, the proposed method, called hybrid shape approach, requires a long time to tune the weighting matrices and results in a high order regulator that does not clearly shape the transient response.

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. [21] describes a sliding mode control approach based on partial feedback linearization with the aim of achieving complete sloshing suppression. Nevertheless, the paper does not solve the problem of providing sloshing measurements, assuming complete availability of states and validating the control law by means of simulations only. In order to overcome this limitation, [4] makes use of an observer to estimate sloshing values, thus reverting to an open loop architecture.

Complete sloshing suppression is a uselessly conservative constraint for several applications, such as those concerning packaging problems, chemical and food industries, as well as in the service robotics field. When only a spilling avoidance constraint is enforced, it is possible to achieve faster motion and, as a consequence, an increase of efficiency. Despite this fact, only few published works can be found that limit themselves to a spilling avoidance control problem; the most significant is [8]. The paper considers a minimum-time feedforward motion control problem for an open container carrying a liquid. Limitations of this work are that the proposed method relies on linear programming optimization which is solved off-line to compute the input to the system and the fact that it sticks to horizontal and straight motion.

[20] explicitly considers the use of a robotic manipulator to handle the liquid container as it considers a service robotics application, specifically serving beverages to customers. The proposed solution is a gain scheduling

controller based on the hybrid shape approach. The method is very practical since it 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 [3], which uses an input shaping approach to control an industrial manipulator for transferring a liquid container with complete sloshing suppression along a straight path. Finally, even if it still deals with slosh-free motion, [23] 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.

2.2 Liquid detection

Detection of a moving liquid surface is a more recent research field, in fact its evolution goes side by side with the development of reliable and efficient vision systems, which started to spread out in the last few years. Thanks to sophisticated mathematical tools, like Computational Fluid Dynamics (CFD), we are able to obtain good results in the modeling and simulation of a time-varying surface of a liquid; however, capturing the real-time behavior of real liquids remains a challenging problem. Scientific papers on this topic can be divided in two groups depending on their main objective.

Works belonging to the first group [10] [24] [29] mainly focus on the complete reconstruction of the moving liquid surface. The main focus is on transparent fluids and refractive surfaces, since they represent the most challenging situation. For instance, [10] places a known pattern underneath the liquid and uses a camera array to track the distorted features points over time and across cameras in order to retrieve the shape of the fluid surface. This method clearly works only with non-opaque liquids and requires a complex device for image acquisition, which is also sensitive to calibration errors. Moreover, it is not designed for real-time computation as it suffers from a high computational load. A similar solution is described in [24], which is able to track the pattern distortions exploiting the stereo vision of only two cameras. However, cases of ambiguity in surface reconstruction are possible due to the use of a simpler acquisition system. In order to eliminate ambiguity and increase accuracy, [29] replaces the pattern below the container with a Bokode, which is a computational optical device that emulates a pinhole projector. However, the method requires a long training phase and it is able to reconstruct only a small area of the liquid surface.

The second group of papers aims at liquid level estimation inside containers. This objective suits the purposes of the thesis more, since we are interested in measuring liquid sloshing rather then reconstructing the entire fluid surface in detail. Moreover, simpler acquisition systems composed of a single camera are sufficient to solve the problem without the aid of added patterns underneath the liquid. For instance, [19] estimates the liquid level using an RGB camera. Firstly, the container is identified through image segmentation, then an edge detection algorithm highlights the cup and liquid surface profile. Finally, level is estimated measuring the distance between the fluid surface and the cup bottom. The main limitation of this solution is that it requires a clear color separation among liquid, cup and background. Otherwise, segmentation and edge detection algorithms may give incorrect responses. Instead, an RGB-D camera is used in [11]. Although the proposed method still relies on color separation for edge detection, the use of depth information allows for estimate of transparent liquid surfaces. The paper presents a probabilistic approach that does not require a prior knowledge of the fluid refractive index. However, several images from different points of view are needed for a correct estimate, thus preventing the possibility to use the algorithm to track the motion of a moving surface. [17] develops a method for service robotics applications that relies only on depth measurements, which are translated into a 3D point cloud. Then, the container is recognized in the image and the liquid surface is extracted. The algorithm works for both transparent and opaque liquids, with the requirement of knowing beforehand the type of fluid (transparent or opaque) and its refractive index (if transparent).

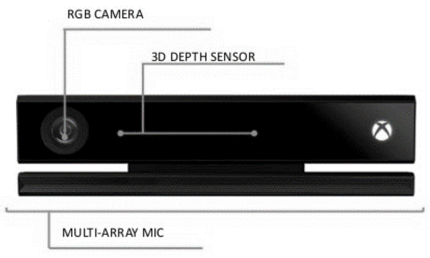
Chapter 3

Sloshing Measurement

The task of measuring liquid sloshing is not simple. In the discussion about the state of the art of liquid detection, several methods that can be used to extract measurements of the sloshing angle have been presented. Examples exist that employ one or more RGB cameras to capture the distortion of known patterns [10] or the level of a coloured liquid [19],[23]. Other works on sloshing suppression methods solve the problem in different ways: using pressure [28] or electric-resistance [3] sensors attached onto the container to estimate liquid level, or lasers sensors that reflect on an artificially coloured liquid surface [8]. However, all these solutions rely either upon modification of the liquid properties or upon specific set-ups and instrumented containers.

This thesis work develops an open loop on-line controller, hence sloshing measurements are only used for validation and verification purposes. Nevertheless, also in the view of a possible future modification of the control architecture to a closed-loop one, a non-invasive and low-cost solution for sloshing measurement has been preferred, capable to handle both opaque and transparent liquids. In fact, possible end-users include chemical and pharmaceutical industries, for which the fluid is the main product and therefore can not be altered. Moreover, avoiding ad-hoc designed instrumentation reduces costs and broadens the possible application scenarios, as

(a) Microsoft Kinect v2 sensor.



(b) Sensors location on Kinect v2.

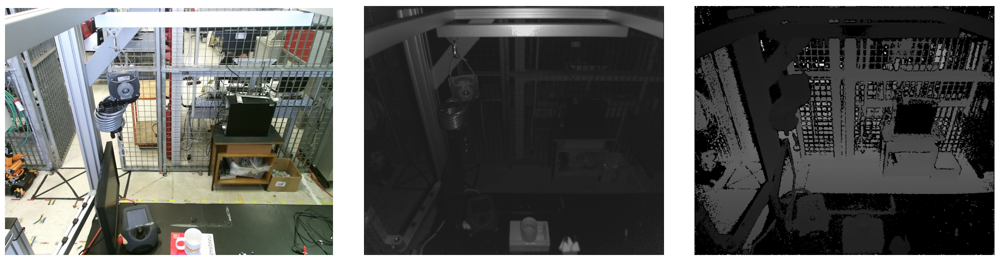
Figure 3.1: Image of the Microsoft Kinect v2 sensor (a) and scheme of sensors location (b).

in the case of household robots. Considering all said factors, possibilities for the choice of sloshing sensors reduce considerably. A good one is to use depth sensors, such as Microsoft Kinect or ASUS Xtion, which are flexible, not expensive and non-invasive devices. This thesis work starts from the contributions presented in [11] and [17], while adapting algorithms to the specific task under study. Since the focus of the thesis is on the design of the control law, it has been chosen to consider only opaque liquids for experimental validation. In this way the developed method for the extraction of the sloshing angle is kept simpler, as it does not have to delve into details of solving the problem of detecting transparent liquids. However, the works from [11] and [17] show that it is also possible to confront the latter class of fluids using only depth information.

In the following, Section 3.1 describes how depth sensors work, while Section 3.2 goes into details of the algorithm used to retrieve liquid sloshing measurements.

3.1 Depth sensor

In the experimental set-up described in chapter 5, a Microsoft Kinect v2 has been used (see Fig. 3.1a). Inside its case, a Kinect sensor contains (see Fig. 3.1b):



(a) RGB image (cropped). **(b)** Infra-red image. **(c)** Depth image.

Figure 3.2: Examples of acquisition of the same image using different sensors of Microsoft Kinect v2.

- An RGB camera that stores three channel data (one per colour) in a 1920x1080 resolution.
- An infra-red (IR) emitter and an IR depth sensor with a resolution of 512x424 for both IR and depth image acquisition.
- A multi-array microphone, which contains four microphones for capturing sound. The presence of multiple microphones allows to record audio as well as to find the location of the sound source.

Figure 3.2 shows examples of colour, infra-red and depth images acquired by a Kinect v2, however only depth measurements are exploited in this thesis. In particular, Kinect uses time-of-flight as core mechanism for depth information retrieval: the emitter emits infra-red light beams which are reflected back to the sensor; based on the elapsed time between emission and reception, the reflected beams are converted into depth values that measure the distance between an object and the sensor. The result is a 512x424 image where each pixel contains the Cartesian distance, in millimetres, from the camera plane to the nearest object at that particular (x, y) coordinate, as shown in Figure 3.3. The (x, y) coordinates of a depth frame do not represent physical units in the room; instead, they represent the location of a pixel in the depth image. Figure 3.4 shows the depth range of the sensor in meters: only objects located between 0.4 m and

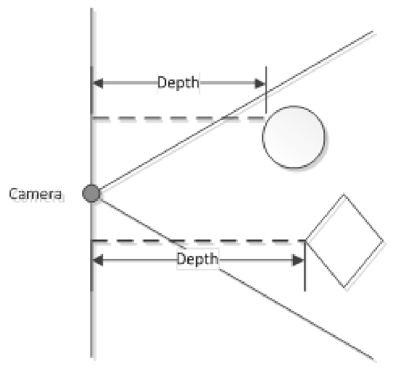


Figure 3.3: Scheme depicting depth values as computed by the Kinect sensor.

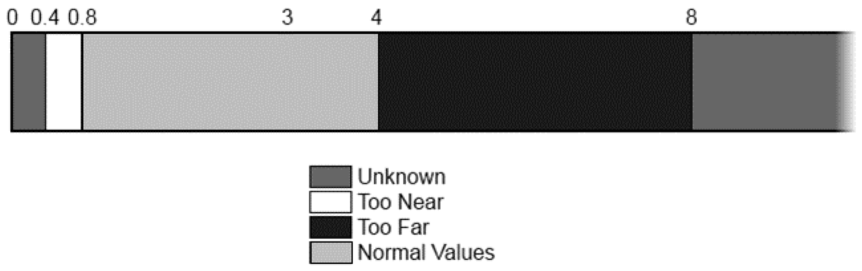


Figure 3.4: Measurement range of Kinect's depth sensor.

4 m are reliably detected, whereas out of range data are set to zero depth value. A Kinect is able to stream depth images at a rate of 30 Hz. All the aforementioned specifications of the Microsoft Kinect's depth sensor are summarized in Table 3.1

3.2 Sloshing angle extraction algorithm

Clearly, depth images coming from the Kinect sensor stream do not directly provide sloshing angle measurements, indeed a procedure to extract such information starting from the depth map is needed. The remaining of the chapter describes in details the algorithm used to pass from Kinect raw data to sloshing measurements, together with examples of the achieved results. The outline of the procedure is sketched in Fig. 3.5: the starting

Table 3.1: Kinect depth sensor specifications

Tecnology	IR time-of-flight
Resolution	512x424
Stream rate	30 Hz
Field of view	60° vertical by 70° horizontal
Depth range	0.4–4 m

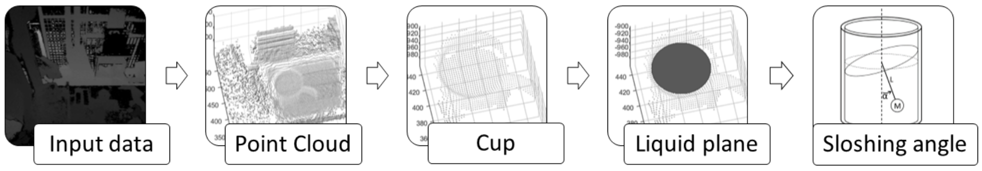


Figure 3.5: Scheme of the pipeline procedure for sloshing angle extraction from depth image.

point is a frame of the depth stream, whose information are translated in a 3D point cloud (first step); then, the liquid container is detected (second step) and the liquid plane inside is found (third step); finally the value of the sloshing angle is computed (fourth step).

3.2.1 Point cloud creation

As already said, the coordinates of a pixel in a depth frame do not convey measurements expressed in physical units. Indeed, the depth image represents the perceived environment in the depth space, which is a non-homogeneous 2.5-dimensional space where two elements are the coordinates of the projection of an observed point on the sensor plane and the third element is the depth of this point from such plane. In order to operate a transformation from depth-space coordinates to Cartesian coordinates, we can model the depth sensor as a pin-hole camera [15], which is characterized by a focal length f and the position (c_x, c_y) of the focal axis in the image plane. Moreover, a rotation matrix R and a translation vector $\boldsymbol{t}$ define

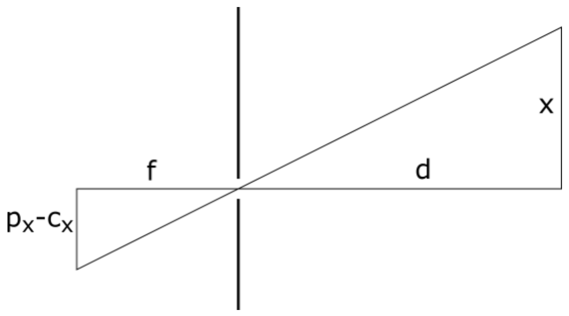


Figure 3.6: Pin-hole camera model.

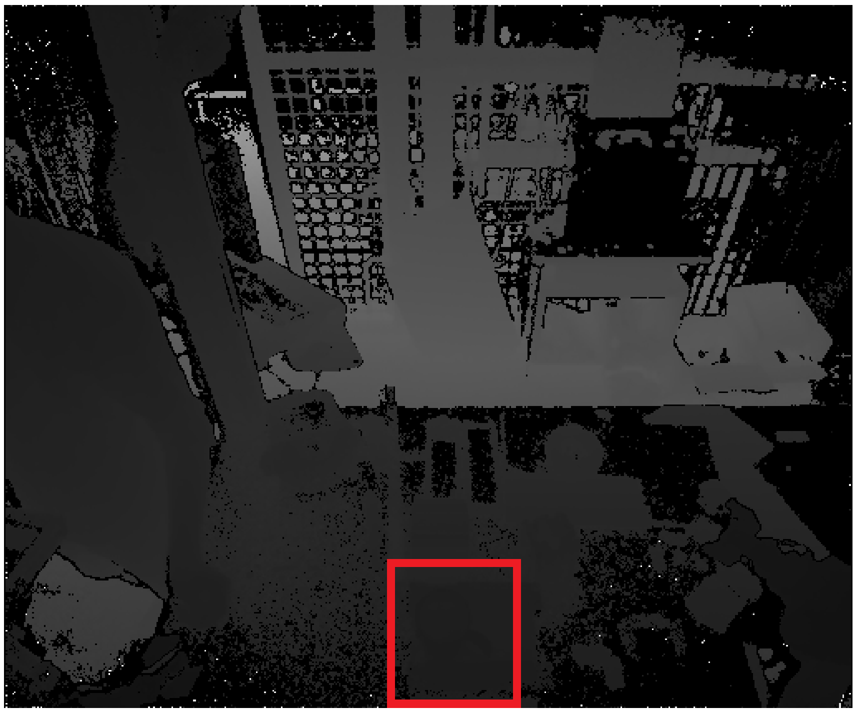
position and orientation of the world reference frame C_w with respect to the sensor one C_s. Let ${}^{D}\boldsymbol{P} = (p_x, p_y, d)$ be a generic point in the depth space, then its projection ${}^{C_s}\boldsymbol{P} = ({}^{C_s}x, {}^{C_s}y, {}^{C_s}z)$ in the sensor space is given by:

$${}^{C_s}x = \frac{(p_x - c_x)d}{f} \qquad {}^{C_s}y = \frac{(p_y - c_y)d}{f} \qquad {}^{C_s}z = d$$

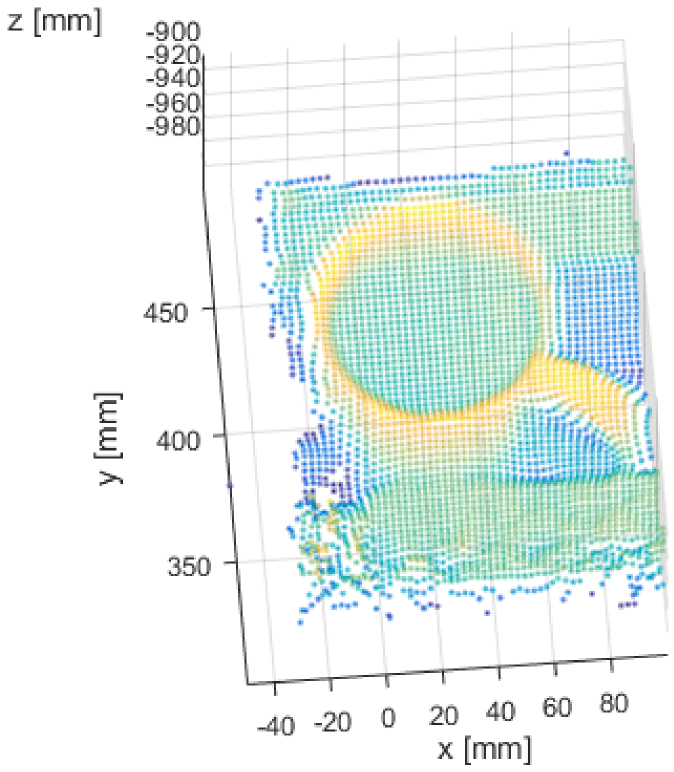
Figure 3.6 gives a pictorial representation of the above relations as the similarity of two triangles. The same point is also represented in the world frame by vector ${}^{C_w}P = ({}^{C_w}x, {}^{C_w}y, {}^{C_w}z)$, which is equal to:

$${}^{C_w}\boldsymbol{P} = R\, {}^{C_s}\boldsymbol{P} + \boldsymbol{t}$$

The first step of the algorithm is thus to map each image of the Kinect depth stream to a 3D point cloud that refers to the Cartesian world reference frame. In order to decrease the computational load, only points inside a region of interest (ROI) containing the liquid container are projected. Selection of the correct ROI enhances performance also for the subsequent stages, as it excludes the image background and other objects that are present in the camera field of view. Figure 3.7 shows an example of results of depth-to-point cloud translation, in particular 3.7a displays the initial depth frame with an approximate indication of the region of interest while 3.7b presents the related point cloud. Furthermore, Figure 3.8 gives further insight into the point cloud representation, showing the same 3D image from different points of view and displaying hints for interpretation.

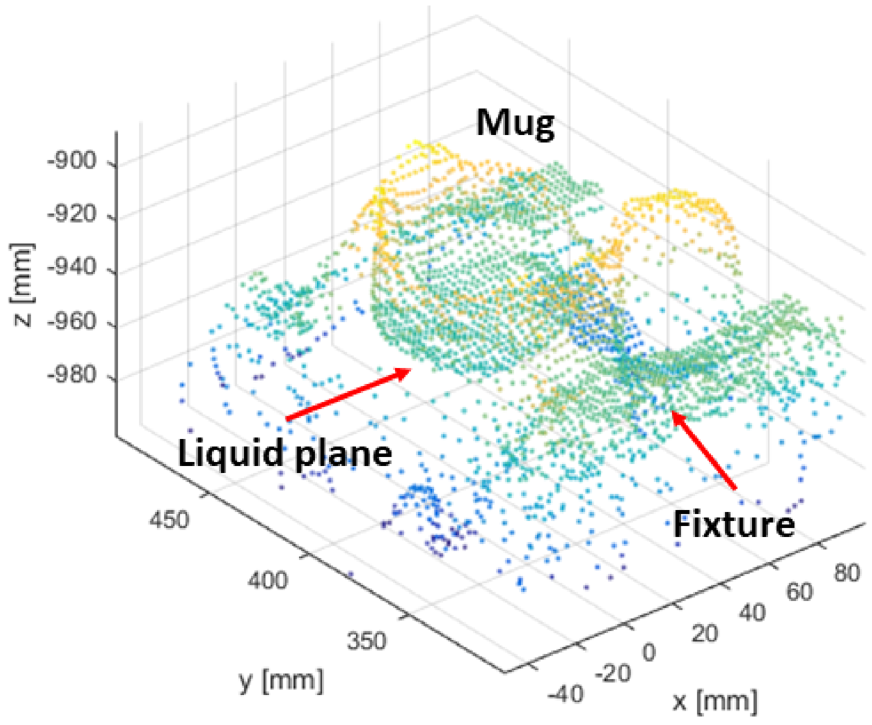


(a) Initial depth frame and ROI (in red).

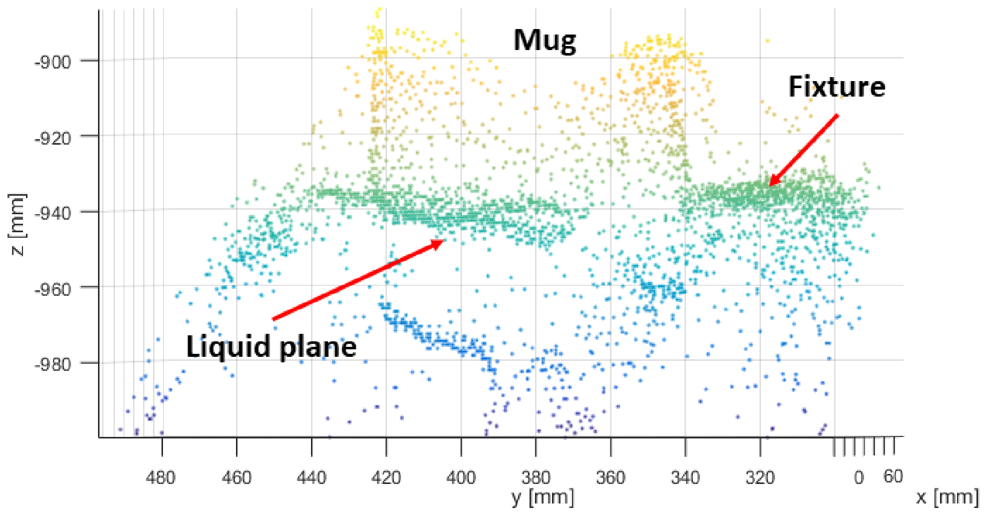


(b) Resulting ROI point cloud.

Figure 3.7: Example of the first step of the sloshing angle extraction algorithm.



(a) Angular view.



(b) Side view.

Figure 3.8: Resulting 3D point cloud seen from different points of view.

3.2.2 Liquid container detection

Once the 3D image has been generated, the algorithm searches in the scene for the liquid container, which is supposed to be a cylinder. In particular, the procedure tries to fit a cylinder model in the point cloud using the M-estimator Sample Consensus (MSAC) method [25], which is a modification of Random Sample Consensus (RANSAC) algorithm presented in [14]. RANSAC iteratively estimates the parameter of the cylinder model from data points while neglecting outliers, which does not have any influence on the outcome. This is an extremely valuable property in the framework of our problem: both to get rid of the noise of Kinect's depth measurements and to exclude points belonging to other objects that may be close to the container, such as the robotic arm holding the cup. In order to achieve this result, the algorithm relies on a voting mechanism and roughly works as follows:

1. Randomly select a subset containing the minimum number of data points needed to fit the model and compute its parameters.

2. Use the identified model to determine the set of all points that are consistent with the model within some tolerance (consensus set). If a point does not fit, it will be considered as an outlier and neglected in next steps.

3. If the cardinality of the consensus set is not sufficiently high, make a new attempt repeating the above procedure; otherwise, the model is reasonably good. If, after some predetermined number of trials, no suitable consensus set has been found, the algorithm terminate in failure.

4. Finally, refine the model by estimating it again using all members of the consensus set.

The whole process can be repeated multiple times to find a better estimate, in this case at each iteration the new obtained model is kept only if its

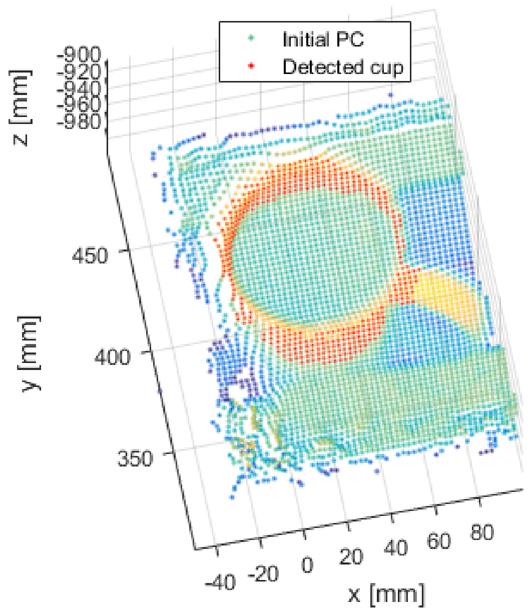


Figure 3.9: Point cloud highlighting in red the container detected by the second step of the sloshing angle extraction algorithm.

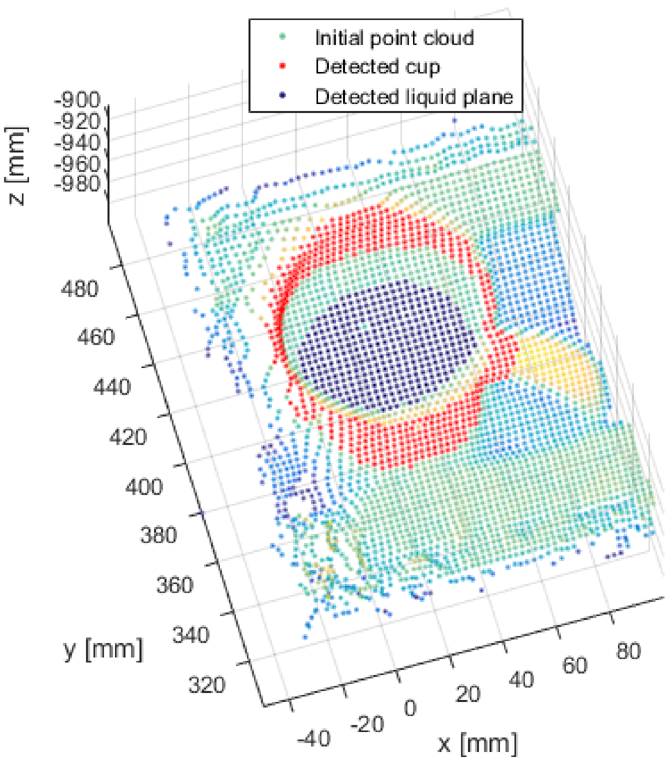
consensus set is larger than that of the previously saved model. MSAC only differs from the standard RANSAC algorithm because points in the consensus set are not equally weighted, but each data point is evaluated according to how well it fits the identified model. The result is a more robust and reliable estimate, without increasing the computational load. Furthermore, in the present application, reliability of the fitting model is also increased by specifying the orientation of the liquid container, which is always kept in the upright position.

Eventually, the algorithm gives as outputs the identified cylinder model parameters, namely its radius, the centre coordinates and the axis orientation. If dimensions of the container to be detected are known in advance, it is possible to use the estimated value of the radius to check the validity of the model. In fact, if it differs too much from the real cup radius, it is likely that an erroneous identification occurred, for instance because a different object has been detected or the container has been merged with a nearby body. Figure 3.9 shows the result of cup detection starting from the point cloud in Fig. 3.7b obtained in the previous step. As one can see, the algorithm is able to correctly detect the container, leaving outside the handle, even if fixtures occlude part of the object.

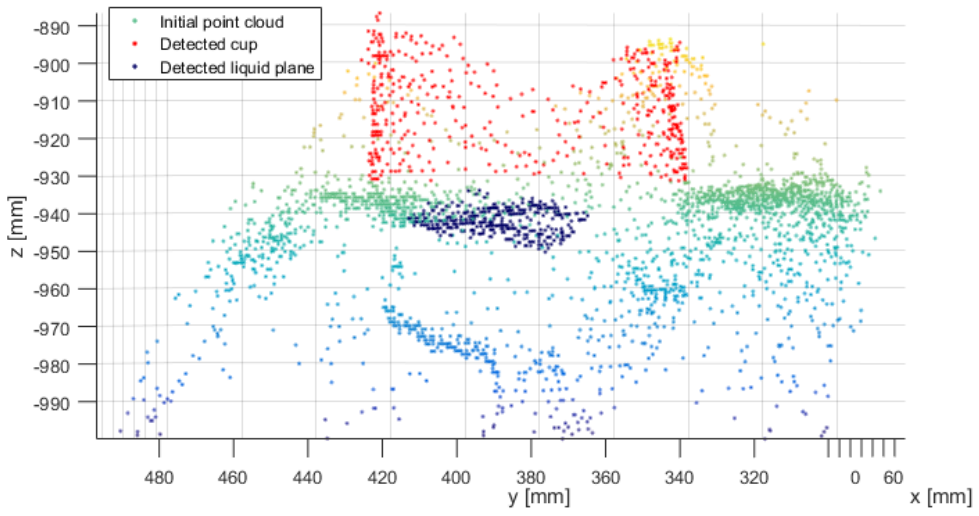
3.2.3 Liquid plane detection and evaluation of the sloshing angle

Once the liquid container has been isolated, it is possible to reduce the complete point cloud selecting only data points that lie inside the cup. A point is deemed to be inside the container if its distance from the cylinder vertical axis is less or equal to a fraction of the cylinder radius. Considering a fraction (e.g. 80%) of the radius instead of its whole value prevents the algorithm to include points that belong to the container surface in the reduced point cloud. Afterwards, we search for the liquid surface among the reduced point cloud using again MSAC method, this time to fit a plane. The output of this step is the vector normal to the liquid surface. In case of a moving container, assuming a planar model for the liquid surface means to consider only the first asymmetric mode of oscillation of the sloshing dynamics, which is however sufficient for sloshing control purposes as discussed in detail in Chapter 4. Example of the progress of the algorithm is presented in Figure 3.10, that shows the outcome of liquid detection in case of moving liquid.

Finally, the fourth and last step of the algorithm is to compute the sloshing angle, defined as the angle between the vector normal to the liquid surface and the reference vertical axis.



(a) Top view.



(b) Side view.

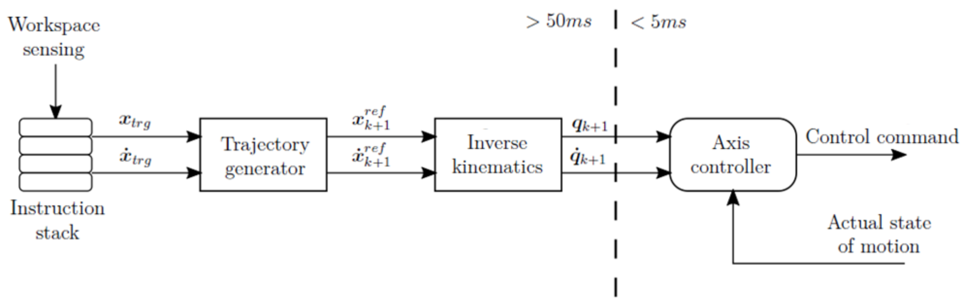
Figure 3.10: Point cloud highlighting in dark blue the liquid plane detected by the third step of the sloshing angle extraction algorithm.

Chapter 4

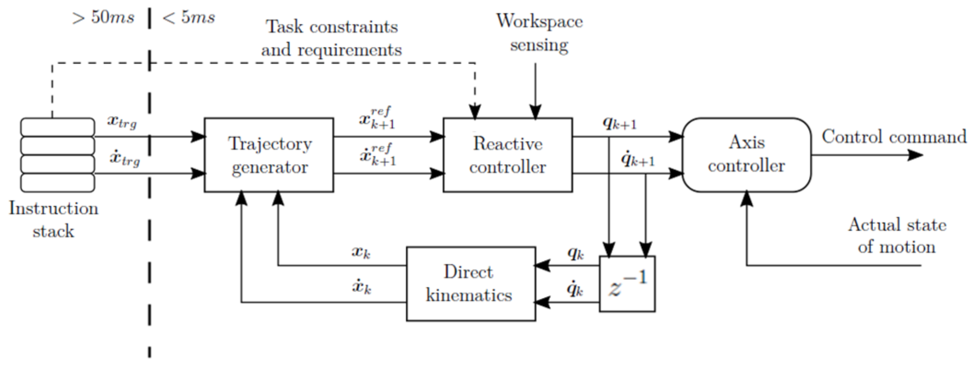
Control Strategy

As already discussed in the Introduction, the aim of this work is to design an on-line controller for spilling avoidance. However, the objective is not that of achieving complete sloshing suppression as in [3], [23] or [28], but only to prevent the liquid to spill out the container by setting an upper limit α_{lim} to the sloshing angle, which is only addressed in [8]. In this way a faster motion can be obtained since a less conservative constraint is enforced. Moreover, 3D motion of the container is considered, unlike for instance [3] and [8] that stick to planar motion.

In order to achieve all goals, a different approach with respect to what is possible to find in the literature regarding sloshing control has been followed. Most of papers deal with the problem either by properly shaping the input signal [3],[8] or the reference signal [23] as resulting from off-line computations. Differently, this work presents an on-line solution obtained through a constraint-based control method, which allows to specify the necessary process requirements in a natural manner by means of suitable constraints. In this way there is no need to delve into the details of how to handle all situations that arise at execution time, letting the controller be in charge of satisfying all constraints at each time instant.



(a) Traditional pipeline architecture.



(b) Reactive controller architecture.

Figure 4.1: Comparison between traditional (top) and reactive (bottom) control structure.

4.1 Constraint-based control

During last years there has been a push towards alternatives ways of programming robots that are more capable of describing complex tasks than traditional robot programming languages. An important issue is also to keep the process as intuitive and user-friendly as possible. Examples of this trend are teaching by demonstration [5] or intuitive HMIs [2]. Constraint-based programming follows this development of robotic research, since it represents the most natural way to define a broad variety of tasks that are characterized by several requirements to be fulfilled simultaneously.

The control architecture which is proposed in this work follows the approach presented in [30]. The idea is to continuously update the reference trajectory in order to always satisfy the process constraints; Fig. 4.1 shows the control structure used in this thesis, compared with the standard pipeline approach. In traditional robot programming, see Fig. 4.1a, 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 to be interfaced with external regulators. However, a scheme similar to that sketched in Fig. 4.1b can be designed to exploit the existing proprietary axis controller of the robot.

Overall, the high level controller used for this thesis work is composed 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. In general, the reactive controller which implements the optimization problem receives information about task constraints from the environment. The complete motion task is thus specified by the initial and target positions, plus a suitable cost function and constraints for the optimization problem.

4.1.1 Trajectory generation

In this work the focus is on the motion of the liquid container, thus trajectories are specified and then planned in the task space. Task variables describe position and orientation of the robot end-effector, which in this case is the vessel. A closer attention has to be paid to the orientation representation, in fact, for the problem under study two out of three possible rotatory motions are constrained by limiting the container to stay in upright position. ZYZ Euler angles are chosen in order to obtain a convenient description, which is also free from singularities for all admissible robot configuration (see Appendix A). In particular letting o_{z_0}, o_y and

o_{z_1} be the three Euler angles, singular points occur when $\sin(o_y) = 0$ and hence $o_y = k\pi$ for all integer values of k. With this choice the orientation is described by a three dimensional vector $\boldsymbol{o}$ where o_{z_0} is the only orientation degree of freedom left:

$$\boldsymbol{o} = \begin{bmatrix} o_{z_0} \\ o_y \\ o_{z_1} \end{bmatrix} = \begin{bmatrix} o_{z_0} \\ \pi/2 \\ \pi \end{bmatrix} \tag{4.1}$$

The constant values of angles o_y and o_{z_1} are those needed to keep the container vertical according to the adopted convention for the robot tool center point reference frame.

Let $\boldsymbol{x}$ be the whole vector of task variables and $\boldsymbol{q}$ the vector of joint variables, then the following kinematic equations holds that relates the task space with the joint one:

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

where $J(\boldsymbol{q}) = \frac{\partial f}{\partial \boldsymbol{q}}$ 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 the actual state to the target state of motion, while considering usual constraints, like maximum task velocities $\dot{\boldsymbol{x}}_{max}$ and accelerations $\ddot{\boldsymbol{x}}_{max}$. The output of the OTG is the reference trajectory evaluated at the next sampling time, both in terms of reference position $\boldsymbol{x}_{k+1}^{ref}$ and velocity $\dot{\boldsymbol{x}}_{k+1}^{ref}$. Given that in the spilling avoidance problem two orientation variables are indeed fixed, as previously discussed, it is sufficient to generate trajectories for a four dimensional task space. The computed values are finally sent to the controller that solves the optimization algorithm. For the implementation, Reflexxes Motion Libraries [13][16] are used for on-line trajectory generation. They guarantee a fast computation of the next state of motion (within a millisecond) and allow an easy integration in the control algorithm with simple Matlab and C interfaces.

4.1.2 Reactive controller

The reactive controller receives as inputs the reference values coming from the OTG and tracks the trajectory while satisfying all task-related constraints. The block output consists in the next state of motion in the joint space $(\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1})$ that constitutes the command sent to the robot axis controller. Because of the presence of constraints to be fulfilled, these values may be different from those needed to exactly follow the setpoint as given by the trajectory generator, that can 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 pair of discrete time equations:

$$\begin{aligned} \boldsymbol{q}_{k+1} &= \boldsymbol{q}_k + T_s\dot{\boldsymbol{q}}_k + 0.5T_s^2\boldsymbol{u}_k \\ \dot{\boldsymbol{q}}_{k+1} &= \dot{\boldsymbol{q}}_k + T_s\boldsymbol{u}_k \end{aligned} \tag{4.2}$$

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

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

The controller computes joint accelerations $\boldsymbol{u}_k$ as the solution of a constrained optimization problem that can be written as follows:

$$\min_{\boldsymbol{u}_k} \left(\frac{1}{2}\boldsymbol{e}^T Q_p \boldsymbol{e} + \frac{1}{2}\dot{\boldsymbol{e}}^T Q_v \dot{\boldsymbol{e}}\right) \tag{4.3}$$

subject to

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + T_s J_k \dot{\boldsymbol{q}}_k + 0.5T_s^2(\dot{J}_k\dot{\boldsymbol{q}}_k + J_k\boldsymbol{u}_k) \tag{4.3a}$$

$$\dot{\boldsymbol{x}}_{k+1} = J_k\dot{\boldsymbol{q}}_k + T_s(\dot{J}_k\dot{\boldsymbol{q}}_k + J_k\boldsymbol{u}_k) \tag{4.3b}$$

$$\boldsymbol{u}_{inf} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{sup} \tag{4.3c}$$

$$-\dot{\boldsymbol{x}}_{max} \leq \dot{\boldsymbol{x}}_{k+1} \leq \dot{\boldsymbol{x}}_{max} \tag{4.3d}$$

$$-\ddot{\boldsymbol{x}}_{max} \leq \dot{J}_k\dot{\boldsymbol{q}}_k + J_k\boldsymbol{u}_k \leq \ddot{\boldsymbol{x}}_{max} \tag{4.3e}$$

$$h_k(\boldsymbol{u}_k) \leq 0 \tag{4.3f}$$

where $J_k = J(\boldsymbol{q}_k)$ and $\dot{J}_k = \dot{J}(\boldsymbol{q}_k)$, T_s is the sampling time and Q_p,Q_v are suitable positive definite wighting matrices for the tracking error. Equations (4.3a) and (4.3b) update task position and velocity based on joint velocities and accelerations, while equations (4.3d) and (4.3e) describe the bounds set for velocity and acceleration of the task variables, respectively. Constraints (4.3c) 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 of the plain 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 [6]. At last, generic inequalities (4.3f) stand for additional constraints that can be specified to define specific requirements of the problem. In the case under study there will be the constraint that force the container to remain in upright position and, more interestingly, the constraint that limits the sloshing angle to its maximum value. While the former is specified in the following of this section, the latter is described in Section 4.3.

4.1.3 Optimization problem

In the definition of the reactive control law, it is possible to consider 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 (4.3a) and (4.3b) can be substituted in the other equations. After some algebraic computations the optimization problem is reduced to:

$$\min_{\boldsymbol{u}_k} \left(\frac{1}{2} \boldsymbol{u}_k^T H_k \boldsymbol{u}_k + g_k^T \boldsymbol{u}_k \right) \tag{4.4}$$

subject to

$$A_k \boldsymbol{u}_k \leq b_k \tag{4.4a}$$

$$h_k(\boldsymbol{u}_k) \leq 0 \tag{4.4b}$$

where the matrices that define the quadratic cost function are:

$$\begin{aligned} H_k &= J_k^{up\ T} \left(\frac{T_s^2}{4} Q_p + Q_v \right) J_k^{up} \\ g_k &= \frac{1}{2} \left[\boldsymbol{x}_k - \boldsymbol{x}_{k+1}^{ref} + T_s \left(J_k^{up} + \frac{T_s}{2} \dot{J}_k^{up} \right) \dot{\boldsymbol{q}}_k \right]^T Q_p J_k^{up} + \\ &+ \frac{1}{T_s} \left[\left(J_k^{up} + T_s \dot{J}_k^{up} \right) \dot{\boldsymbol{q}}_k - \boldsymbol{x}_{k+1}^{ref} \right]^T Q_v J_k^{up} \end{aligned}$$

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

Finally, the additional constraints in (4.4b) still need to be specified. They refer to the orientation of the container and the limit to sloshing angle. The latter is described in section 4.3 whereas for the former, recalling the expression (4.1), the update rule in (4.3a) and letting $o^c = [o_y \quad o_{z_1}]^T$ we have that:

$$
\begin{aligned}
o^c_{k+1} = o^c_k &= \begin{bmatrix} \pi/2 \\ -\pi \end{bmatrix} \\
o^c_{k+1} = o^c_k + T_s J^{dw}_k \dot{\boldsymbol{q}}_k &+ \frac{T_s^2}{2}(\dot{J}^{dw}_k \dot{\boldsymbol{q}}_k + J^{dw}_k \boldsymbol{u}_k) \\
&\text{hence} \\
o^c_k = o^c_k + T_s J^{dw}_k \dot{\boldsymbol{q}}_k &+ \frac{T_s^2}{2}(\dot{J}^{dw}_k \dot{\boldsymbol{q}}_k + J^{dw}_k \boldsymbol{u}_k) \\
\frac{T_s^2}{2} J^{dw}_k \boldsymbol{u}_k &= -T_s \left(J^{dw}_k \dot{\boldsymbol{q}}_k + \frac{T_s}{2} \dot{J}^{dw}_k \dot{\boldsymbol{q}}_k \right) \\
J^{dw}_k \boldsymbol{u}_k &= -\frac{1}{T_s} \left(2 J^{dw}_k + T_s \dot{J}^{dw}_k \right) \\
A^o_k \boldsymbol{u}_k &= b^o_k
\end{aligned}
\tag{4.5}
$$

which finally defines the constraint.

4.2 Sloshing model

In this section a model for the liquid sloshing first mode is presented. The need for a model that describes the oscillatory dynamics of the liquid inside the moving container is twofold. Firstly, the derivation of the constraint that avoid spilling is obviously based on the knowledge of the sloshing behavior; 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 would 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 of several sloshing modes at different oscillation frequencies. However, the resulting model would be uselessly complex for control

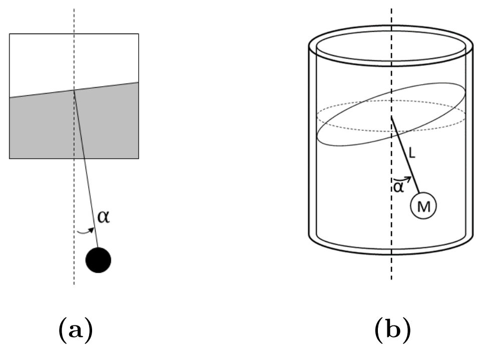


Figure 4.2: Planar pendulum (a) and spherical pendulum based model (b) for liquid sloshing.

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 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. 4.2a). As far as planar motion is concerned a simple pendulum is sufficient, like in [4],[21], [27] or others, while in case of 3D motion a spherical pendulum must be considered [23] as sketched in Fig. 4.2b.

4.2.1 Mathematical description

A spherical pendulum is characterized by the rod length L, a damping coefficient C and its mass M, which is supposed to be concentrated at one end of the rod. Figure 4.3 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 $[x_P \quad y_P \quad z_P]^T$. In the framework of this thesis the inputs for the model are the accelerations of the pendulum fulcrum P.

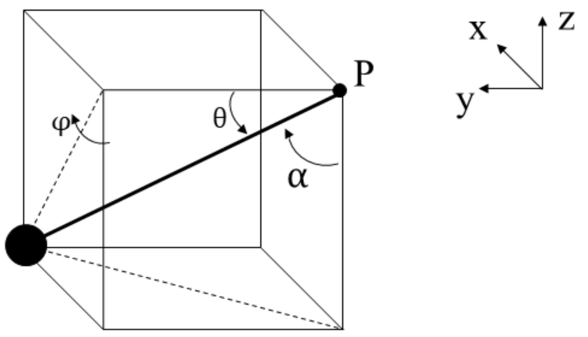


Figure 4.3: Pendulum reference frame and angles.

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:

$$\boldsymbol{p} = \begin{bmatrix} \vartheta \\ \dot{\vartheta} \\ \varphi \\ \dot{\varphi} \end{bmatrix} \tag{4.6}$$

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

$$\begin{aligned} \cos(\alpha) &= \sin(\vartheta)\cos(\varphi) \\ \sin(\alpha) &= \sqrt{\cos^2(\vartheta) + \sin^2(\varphi)} \end{aligned} \tag{4.7}$$

Pendulum angles, mass and fulcrum positions are linked through the following relations:

$$\begin{cases} x_M = x_P + L\sin(\vartheta)\sin(\varphi) \\ y_M = y_P + L\cos(\vartheta) \\ z_M = z_P - L\sin(\vartheta)\cos(\varphi) \end{cases} \tag{4.8}$$

Whenever $\sin(\vartheta) = 0$, a singularity of representation occurs. However, this happen only when $\vartheta = 0$ or $\vartheta = \pi$, that is when the pendulum is horizontal and correspondingly the liquid surface is vertical. It is clear that this is not a meaningful configuration for our problem, both because the aim is to limit sloshing to a certain amount and because at such at high level of oscillations the model is no longer able to correctly describe the system behavior in any case.

Model equations are derived using the Euler-Lagrange method, thus expressions for the kinetic energy T, potential energy V and dissipation function D are needed:

$$T = \frac{1}{2}M(\dot{x}_M^2 + \dot{y}_M^2 + \dot{z}_M^2)$$
$$V = Mgz_M$$
$$D = \frac{1}{2}C(\dot{x}_M^2 + \dot{y}_M^2 + \dot{z}_M^2)$$

and also

$$L_{ag} = T - V$$

where g is the gravitational acceleration and L_{ag} the Lagrangian. Then the two Euler-Lagrange equations, one per degree-of-freedom of the system, are found as:

$$\frac{d}{dt}\left(\frac{\partial L_{ag}}{\partial \dot{\vartheta}}\right) - \frac{\partial L_{ag}}{\partial \vartheta} + \frac{\partial D}{\partial \dot{\vartheta}}$$
$$\frac{d}{dt}\left(\frac{\partial L_{ag}}{\partial \dot{\varphi}}\right) - \frac{\partial L_{ag}}{\partial \varphi} + \frac{\partial D}{\partial \dot{\varphi}}$$

which eventually give as result the following two second-order differential equations, where relations in (4.8) are exploited:

$$\begin{gathered}
L^2M\ddot{\vartheta} + DL^2\dot{\vartheta} - \frac{1}{2}L^2M\sin(2\vartheta)\,\dot{\varphi}^2 - LMg\cos(\varphi)\cos(\vartheta) + \\
+ DL\cos(\vartheta)\sin(\varphi)\,\dot{x}_P - DL\sin(\vartheta)\,\dot{y}_P - DL\cos(\varphi)\cos(\vartheta)\,\dot{z}_P + \\
+ LM\cos(\vartheta)\sin(\varphi)\,\ddot{x}_P - LM\sin(\vartheta)\,\ddot{y}_P - LM\cos(\varphi)\cos(\vartheta)\,\ddot{z}_P = 0 \\
\\
LM\sin(\vartheta)\,\ddot{\varphi} + DL\sin(\vartheta)\,\dot{\varphi} + L\sin(\vartheta)\left(Mg\sin(\varphi) + 2LM\cos(\vartheta)\,\dot{\varphi}\dot{\vartheta}\right) + \\
+ D\cos(\varphi)\,\dot{x}_P + D\sin(\varphi)\,\dot{z}_P + M\cos(\varphi)\,\ddot{x}_P + M\sin(\varphi)\,\ddot{z}_P = 0
\end{gathered} \tag{4.9}$$

The above equations can be easily rewritten with respect to the state variables $\boldsymbol{p}$ in (4.6), obtaining a non-linear expression in the form:

$$\dot{\boldsymbol{p}} = f(\boldsymbol{p}, \boldsymbol{u}_P) \qquad \boldsymbol{u}_P = [\ddot{x}_P \quad \ddot{y}_P \quad \ddot{z}_P]^T$$

The complete non-linear model of sloshing dynamics is employed to simulate the behavior of liquid inside the motion container in the most accurate way. Results of the simulation are used in the control scheme as inputs to the reactive controller to predict at each time instant the sloshing angle and its derivative.

In order to derive the spilling avoidance constraint, the model as developed in equations (4.9) is still too complicated. In fact, the objective of designing an on-line controller poses a strong limitation on the complexity of such constraint as the optimization problem in (4.3) should be solved, if possible, within one sampling period, that is usually of few milliseconds (see Fig. 4.1b). Therefore, the ideal situation would be to derive a linear constraint, as it happens in [30] based on the theory from [26], that ensures the compliance with the maximum allowed sloshing angle α_{lim}. For this reasons, it is useful 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:

$$\begin{cases} \delta\ddot{\vartheta} = -\dfrac{C}{M}\delta\dot{\vartheta} - \dfrac{g}{L}\delta\vartheta + \dfrac{1}{L}\ddot{y}_P \\ \delta\ddot{\varphi} = -\dfrac{C}{M}\delta\dot{\varphi} - \dfrac{g}{L}\delta\varphi - \dfrac{1}{L}\ddot{x}_P \end{cases}$$

These are actually the equations of two decoupled simple pendula. It is also possible to write the model in term of state-equation as:

$$\delta\dot{\boldsymbol{p}} = A_p\boldsymbol{p} + B_p\boldsymbol{u}$$

where

$$A_p = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{g}{M} & -\frac{C}{M} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{g}{M} & -\frac{C}{M} \end{bmatrix} \qquad B_p = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{L} & 0 \\ 0 & 0 & 0 \\ -\frac{1}{L} & 0 & 0 \end{bmatrix} \tag{4.10}$$

From matrix B_p it is apparent that the input $\ddot{z}_P$ does not have any influence on the linearized system, therefore its contribution can be neglected by eliminating the third column of the matrix. Taking into account this last consideration, system input-output transfer function is then computed in the usual way from (4.10), which results in a second-order MIMO system in the general form:

$$G_P(s) = \begin{bmatrix} 0 & G(s) \\ -G(s) & 0 \end{bmatrix} \qquad G(s) = \frac{\mu\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{4.11}$$

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

$$\omega_n = \sqrt{\frac{g}{L}} \qquad \xi = \frac{C}{2\omega_n M} \qquad \mu = \frac{1}{g} \tag{4.12}$$

Transfer function $G_P(s)$ will be the starting point to define the spilling avoidance constraint in the next section, as it provides a simpler description of liquid dynamics.

4.3 Sloshing constraint definition

In Section 4.1.3 the complete optimization problem is stated, except for the additional constraints that define process requirements as written in equation (4.4b). The orientation constraint in (4.5) gives a first refinement of such task specification, what is missing is only the spilling avoidance constraint. The idea for its design is to ensure that at each time instant t_0 it is always possible to find a value for the joints acceleration $\boldsymbol{u}$ such that the sloshing angle α remains less or equal to the established limit α_{lim}, that is:

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

In order to obtain the constraint expression, firstly the system time response for $\vartheta(t)$ and $\varphi(t)$ to an acceleration input on the pendulum fulcrum is computed, then its maximum is found and related to the maximum value of $\alpha(t)$, which is set less or equal to the α_{lim} bound.

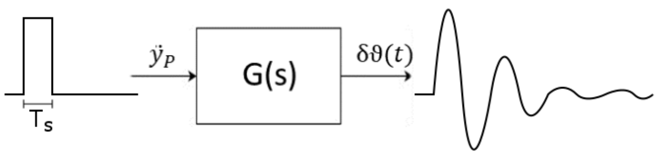


Figure 4.4: Basic input/output scheme for system response computation.

4.3.1 System response to discrete impulse

As already seen in Section 4.2, where a model for the sloshing has been derived, liquid dynamics inside the container can be approximated by a linearized system which receives the pendulum fulcrum acceleration as input and has the two angles (with their velocities) that define the state of a spherical pendulum as state variables. This resulted in the 2-by-2 transfer function $G_P(s)$ in (4.11), which in turn is composed by two decoupled transfer function $G(s)$ and $-G(s)$ that are equal to each other a part from the sign. Hence, in the following, computations are carried out considering two separate SISO systems with transfer functions $G(s)$ and $-G(s)$, respectively, instead of the complete MIMO system.

In order to compute the system time response, the shape of the input must be determined first. Since in the implementation the controller is a digital one, commands are updated at discrete time instants, while being constant during the sampling period. Therefore, we can consider as unitary input a discrete impulse having a width of a sampling period T_s, as portrayed in Fig. 4.4, which reads as:

$$\begin{bmatrix} \ddot{x}_P \\ \ddot{y}_P \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} [\text{step}(t) - \text{step}(t - T_s)] \tag{4.13}$$

where X and Y stands for the impulse amplitude values.

The transfer function $G(s)$ can be equivalently described in state-space form through its observable canonical representation, which is defined by matrices:

$$A = \begin{bmatrix} 0 & -\omega_n^2 \\ 1 & -2\xi\omega_n \end{bmatrix} \qquad B = \begin{bmatrix} \mu\omega_n^2 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \qquad D = 0$$

In this way, the system time response to discrete impulse in (4.13) can be computed using Lagrange formula [7] as follows:

$$\begin{aligned} \delta\vartheta(t) &= Ce^{At} \begin{bmatrix} \vartheta_0 \\ \dot{\vartheta}_0 \end{bmatrix} + \int_0^{T_s} e^{A(t-\tau)} BY d\tau \\ \delta\varphi(t) &= Ce^{At} \begin{bmatrix} \varphi_0 \\ \dot{\varphi}_0 \end{bmatrix} - \int_0^{T_s} e^{A(t-\tau)} BX d\tau \end{aligned} \tag{4.14}$$

with the minus sign in the second equation taking into account the different sign of the two transfer functions gains.

Solving the Lagrange equation and going through some algebraic simplification the time response if obtained:

$$\delta\vartheta(t) = \frac{e^{-t\omega_n\xi}}{\sqrt{1-\xi^2}} \left[\Theta_1 \cos\left(t\omega_n\sqrt{1-\xi^2}\right) + \Theta_2 \sin\left(t\omega_n\sqrt{1-\xi^2}\right)\right] \tag{4.15}$$

where $\Theta_1 = \Theta_1(\omega_n, \xi, \mu, T_s, \vartheta_0, \dot{\vartheta}_0, Y)$ and $\Theta_2 = \Theta_2(\omega_n, \xi, \mu, T_s, \vartheta_0, \dot{\vartheta}_0, Y)$ are highly nonlinear functions in the parameters. Because of the clear similarity between the two expressions in equations (4.14), here and in the following only computation steps concerning $\delta\vartheta(t)$ are reported. Eventually, the final results for both degrees of freedom are shown.

4.3.2 Maximum of function $\delta\vartheta\,(t)$

The next step is to search for the maximum of the time response of function $\delta\vartheta\,(t)$. This is required to guarantee that the sloshing bound is satisfied at any time instant, as it is not sufficient to fulfill the constraint at the next sampling time. In fact, it can happen that even if an input value is admissible at the actual time t_0, it will bring the system in a state where no admissible values for the input can be chosen at time $\bar{t} > t_0$.

However, equation (4.15) is far too complex to obtain a result in closed form which is also useful to derive a simple spilling avoidance constraint. For this reason, a preliminary stage of approximation is required.

Let us consider a generic function:

$$y(t) = a\cos(\omega t) + b\sin(\omega t) \tag{4.16}$$

an upper-bound for y can be found by imposing:

$$\frac{dy}{dt} = 0$$

So we have:

$$\begin{aligned} \frac{dy}{dt} &= -a\omega\sin(\omega t) + b\omega\cos(\omega t) = \\ &= -a\sqrt{1-\cos(\omega t)} + b\cos(\omega t) = 0 \end{aligned}$$

$$\text{hence}$$

$$\begin{aligned} b\cos(\omega t) &= a\sqrt{1-\cos(\omega t)} \\ b^2\cos^2(\omega t) &= a^2\left(1-\cos(\omega t)\right) \\ \cos^2(\omega t) &= \frac{a^2}{a^2+b^2} \\ \cos(\omega t) &= \pm\frac{a}{\sqrt{a^2+b^2}} \end{aligned}$$

where the negative solution has to be neglected, while the positive one is substituted in (4.16) to obtain:

$$y_{up} = \sqrt{a^2+b^2} \tag{4.17}$$

The just derived result can be used in equation (4.15) to get the upper-bound $\delta\vartheta_{up}(t)$ of $\delta\vartheta(t)$, which reads as:

$$\delta\vartheta_{up}(t) = \frac{e^{-t\omega_n\xi}}{\sqrt{1-\xi^2}}\sqrt{\Theta_1^2+\Theta_2^2}$$

from which it is easy to see that the maximum $\delta\vartheta_{max}$ occurs at $t=0$:

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

Recalling that Θ_1 and Θ_2 are functions of the transfer function parameters, the initial conditions and the input amplitude Y it is possible to rewrite the expression of $\delta\vartheta_{max}$ highlighting the dependency from Y. Moreover, following identical steps the same expression related to $\delta\varphi$ is also obtained:

$$\begin{aligned} \delta\vartheta_{max} &= \sqrt{t_0 + t_1 Y + t_2 Y^2} \\ \delta\varphi_{max} &= \sqrt{p_0 + p_1 X + p_2 X^2} \end{aligned} \tag{4.18}$$

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, $\dot{\vartheta}_0$.

4.3.3 Constraint on $\alpha\,(t)$

In the previous paragraph, the maximum values for the pendulum state angles are found. However, our aim is to find a constraint on the sloshing angle α. In order to do so, we recall its relations with the pendulum angles that are stated in equations (4.7). In particular, we can rewrite the sine of the sloshing angle as a function of the state variables of the linearized model $\delta\vartheta$ and $\delta\varphi$. Thus:

$$\sin(\alpha) = \sqrt{\sin^2(\delta\vartheta) + \sin^2(\delta\varphi)}$$

For limited values of the angles, the sine can be approximated with the angle itself, obtaining:

$$\alpha \approx \sqrt{\delta\vartheta^2 + \delta\varphi^2}$$

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

$$\alpha^2 \approx \delta\vartheta^2 + \delta\varphi^2 \tag{4.19}$$

Then, α^2_{max} is found by evaluating (4.19) for the values $\delta\vartheta_{max}$ and $\delta\varphi_{max}$ as written in (4.18). Finally, imposing $\alpha^2_{max} \leq \alpha^2_{lim}$, a quadratic expression for the spilling avoidance constraint is found:

$$p_0 + t_0 + \begin{bmatrix} p_1 & t_1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} X & Y \end{bmatrix} \begin{bmatrix} p_2 & 0 \\ 0 & t_2 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} \leq \alpha^2_{lim} \tag{4.20}$$

where the matrix defining the quadratic part can be easily proved to be positive definite by checking that coefficients $t_2 = t_2(\omega_n, \xi, \mu, T_s, \vartheta_0, \dot{\vartheta}_0)$ and $p_2 = p_2(\omega_n, \xi, \mu, T_s, \vartheta_0, \dot{\vartheta}_0)$ are positive for all values of the parameter, hence defining a convex constraint.

4.4 Complete control architecture

In section 4.1 the general architecture for constrained control has been presented and the optimization problem to be solved at each sampling time by the reactive controller has been defined in 4.1.3, together with the task-related constraints about the container orientation and sloshing avoidance property. In particular, the previous section has described a way to obtain a quadratic expression for spilling avoidance constraint. The presence of both a quadratic objective function and a quadratic constraint means that the minimization in equation (4.4) is a so called quadratically constrained quadratic programming (QCQP) optimization problem. Since it is fully convex, an interior-point algorithm can be used to compute the solution and in the experiments IBM ILOG CPLEX is used as solver [9],[12]. However, state-of-the-art algorithms for QCQP problems are unable to always find an optimal solution within the robot control loop sampling time, which lasts 4 ms. Instead, an average of 7–8 ms is required for iterations when the quadratic constraint is active. A safe assumption is to run the optimization algorithm at a frequency equal to one third of that of the robot controller, thus with a sampling period of 12 ms.

For this reason, an adjustment to the overall control scheme presented in Fig. 4.1b is required. The final architecture, which is the one implemented in practice to obtain the experimental results presented in Chapter 5, is shown in Figure 4.5. The "Controller" block includes the on-line trajectory generator and the optimization algorithm, it receives as input the target values for the task variables and computes the joint accelerations $\boldsymbol{u}_k$. Joint velocities and positions at the following sampling time are update using

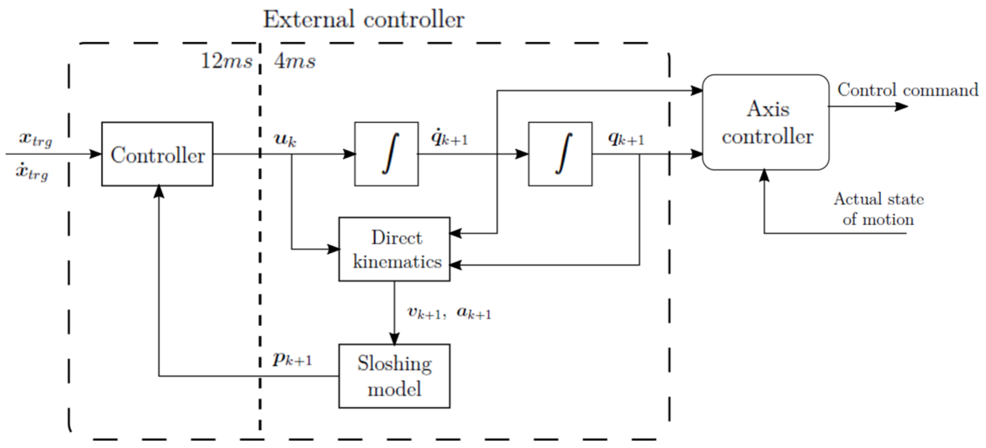


Figure 4.5: Complete on-line control architecture.

equations (4.2). These values are then sent as reference for the robot axis controller on one side, while they are also used to compute the liquid container speed and acceleration, which in turn act as input to the block that simulates sloshing dynamics. Finally, the pendulum state is sent back to the controller to virtually close a control loop and information are used to evaluate the spilling avoidance constraints. From a practical point of view, a Simulink scheme implements the sub-system that works at 4 ms, whereas a program written in C language is used for the slower part of the control architecture.

Chapter 5

Experimental Results

This chapter presents the results obtained applying the developed control architecture to a real robotic system. Firstly, the experimental set-up is described in Section 5.1. Secondly, Section 5.2 presents the results of control algorithm validation, performed through simulations. Finally, the outcome of real experiments is described in Section 5.3.

5.1 Experimental set-up

The control law being developed in this thesis has been tested on an experimental set-up that aims at simulating an industrial robotic cell. In Figure 5.1 it is possible to see all equipment needed for the experiments:

- a robotic system, which is composed of an industrial manipulator endowed with a pneumatic gripper and its control unit;
- a Microsoft Kinect v2 sensor connected to an external computer, used to retrieve sloshing measurements for validation purposes;
- a glass and a mug full of milk, which constitute the containers to be handled.

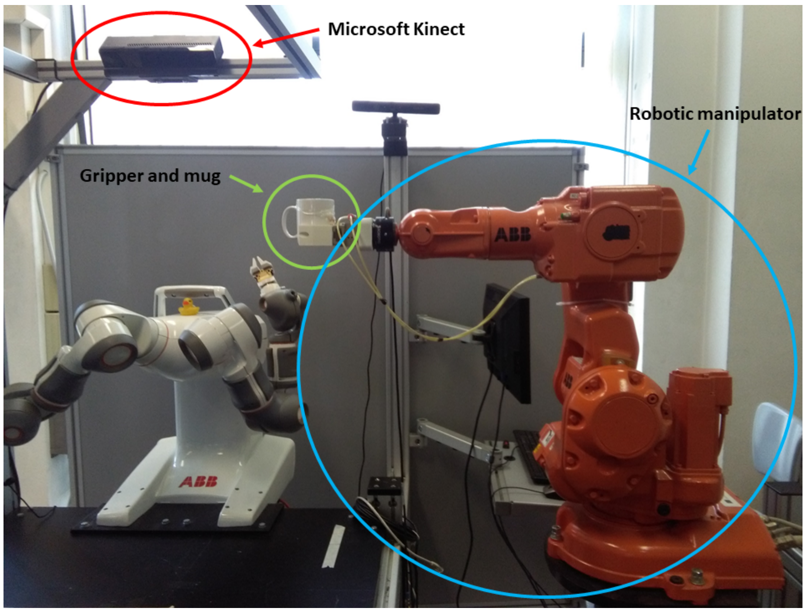


Figure 5.1: Robotic cell simulation

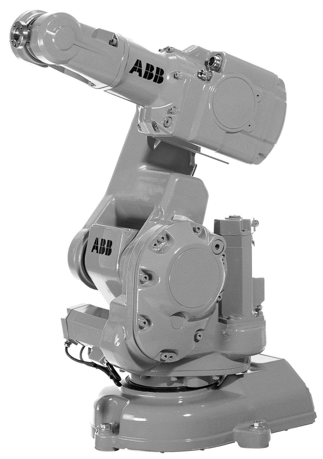
5.1.1 Robotic system

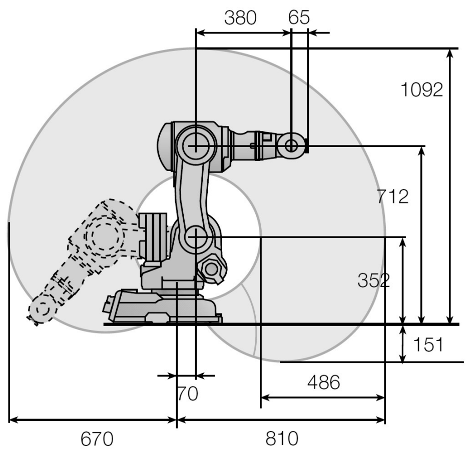
The robot used in this project is an IRB 140 industrial manipulator (see Fig. 5.2a), produced by ABB [22], which is a 6 degree-of-freedom serial robot with a maximum payload of 6 kg and a reach of 810 mm. The robot is equipped with a IRC5 control unit, that provides all functions for motion control. In the following a brief description of the main features of the robotic manipulator are summarized; then, an insight of the control unit operating principle is given to the reader, especially regarding the inclusion of an external custom controller.

Among all information that are present on the IRB datasheet, it is of fundamental importance to know the limits of the robot joint rotations, which define the working region depicted in Figure 5.2b:

$$\begin{bmatrix} \boldsymbol{q}_{min} \\ \boldsymbol{q}_{max} \end{bmatrix} = \begin{bmatrix} -180^\circ & -90^\circ & -230^\circ & -200^\circ & -115^\circ & -400^\circ \\ 180^\circ & 110^\circ & 50^\circ & 200^\circ & 115^\circ & 400^\circ \end{bmatrix}$$

Also, limits to the maximum achievable joint velocity are present, which


(a) ABB IRB 140.



(b) Working range.

Figure 5.2: Image of the ABB IRB 140 industrial manipulator and its working range.

amount in absolute values to:

$$\dot{\boldsymbol{q}}_{max} = \begin{bmatrix} 200°/s & 200°/s & 260°/s & 360°/s & 360°/s & 450°/s \end{bmatrix}$$

However, for safety reasons reduced velocity bounds have been used during experiments. Values of $\boldsymbol{q}_{min}$, $\boldsymbol{q}_{max}$ and $\dot{\boldsymbol{q}}_{max}$ are needed in the control law both for on-line trajectory generation and for the definition of constraints (4.3c) of the optimization problem that is implemented in the reactive controller.

The proposed control solution also requires the evaluation of direct robot kinematics in order to know the values of task variables and their derivatives at each sampling time. To do so in a systematic way, it is convenient to define a reference frame for each link, which has been done using Denavit-Hartenberg convention. The resulting D–H parameters are displayed in Table 5.1, while Figure 5.3 shows the reference frame location for each link of the IRB 140. The values for the parameters can be deducted

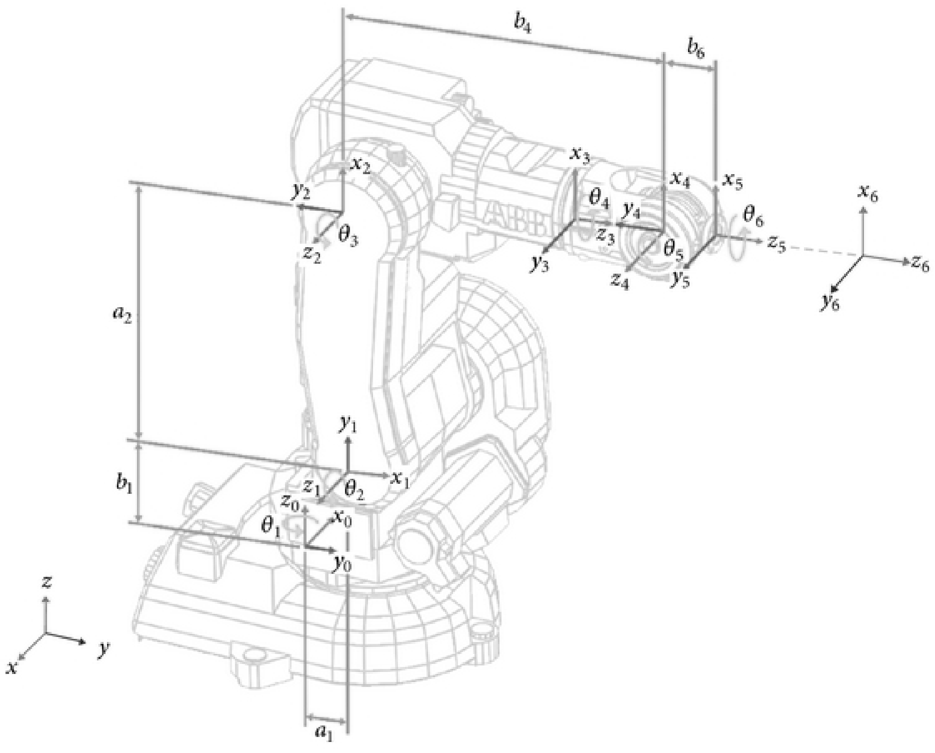


Figure 5.3: Denavit-Hartenberg reference frames of IRB 140

Table 5.1: Denavit-Hartenberg parameters of IRB 140

	Link 1	**Link 2**	**Link 3**	**Link 4**	**Link 5**	**Link 6**
$\boldsymbol{a_i}$	a_1	a_2	0	0	0	0
$\boldsymbol{\alpha_i}$	$-\frac{\pi}{2}$	0	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	0
$\boldsymbol{d_i}$	b_1	0	0	b_4	0	b_6
$\boldsymbol{\vartheta_i}$	q_1	$q_2 - \frac{\pi}{2}$	q_3	q_4	q_5	q_6

from Fig. 5.2b and are equal to:

$$
\begin{aligned}
a_1 &= 0.07\,\mathrm{m} \\
a_2 &= 0.36\,\mathrm{m} \\
b_1 &= 0.352\,\mathrm{m} \\
b_4 &= 0.38\,\mathrm{m} \\
b_6 &= 0.065\,\mathrm{m}
\end{aligned}
$$

The IRC5 is the control unit of the IRB 140 manipulator. It is composed of a *Main computer*, which implements the control law and computes position and velocity reference set-points, and an *Axis computer*, which is in charge of controlling the joint axes. In a normal industrial environment, a human operator manages the robot behaviour giving instructions to the control unit using the provided flexpendant. In this operating mode it is either possible to start an automatic procedure or to manually control joints motion by means of a joystick. However, IRC5 unit has been extended to allow the robot to interface with an external computer where one can built and then load custom control schemes inside the Matlab-Simulink environment. The logic structure is the one sketched in Figure 5.4. OPCOM software is in charge of managing communication between the IRC5 control unit and the external computer; in particular, the system can operate in four different modes:

1. Unload mode: the custom control scheme has not been loaded yet and there is no communication between the two units;

2. Load mode: the control algorithm is loaded, but still communication is not in place;

3. Submit mode: communication from IRC5 to external device starts, thus all data coming from the robot are read and elaborated, but results are not transmitted. It is a test phase equivalent to a closed T_1 switch;

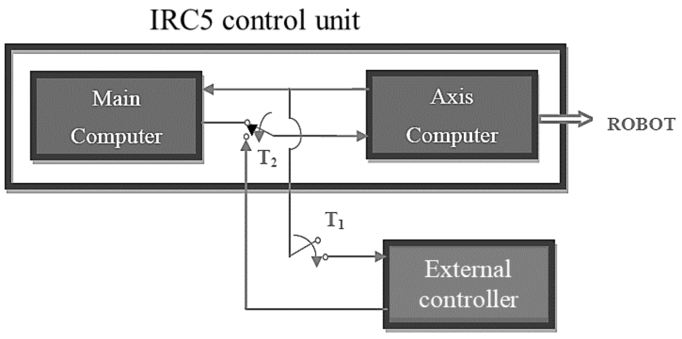


Figure 5.4: Scheme of the extended control unit.

4. Obtain mode: both T_1 and T_2 are closed and a bilateral exchange of data starts. Information are read by the external computer, elaborated and then transmitted back to the robotic manipulator.

5.1.2 Pneumatic gripper

In the experiments, a common glass and a mug full of milk have been used as examples of liquid containers to be handled. In order to enable the robot to hold the vessels, the industrial manipulator has been endowed with a Schunk PGN 100 pneumatic gripper for which a pair of custom fingers have been specifically designed to fit the containers. Drawings of the right finger, with annotations of notable dimensions, are reported in Figure 5.5; the left finger is perfectly symmetrical. Figures 5.6a and 5.6b show the gripper mounted on the robot flange in the open configuration and in closed configuration holding the mug.

5.1.3 Depth sensor

One Microsoft Kinect v2 has been used as depth sensor, whose features have been already presented in Section 3.1. During the robot motion, the Kinect sensor acquires depth information at 30 frames per second and streams data to an external computer. Data are then off-line analyzed using the algorithm described in Section 3.2 in order to extract the sloshing

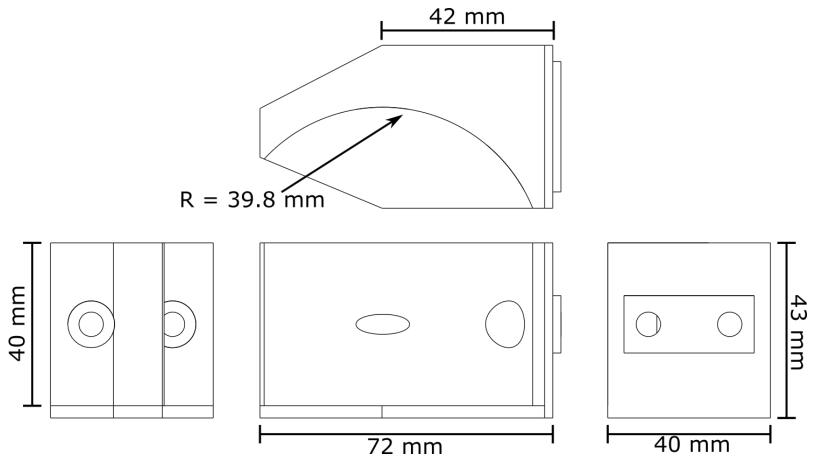
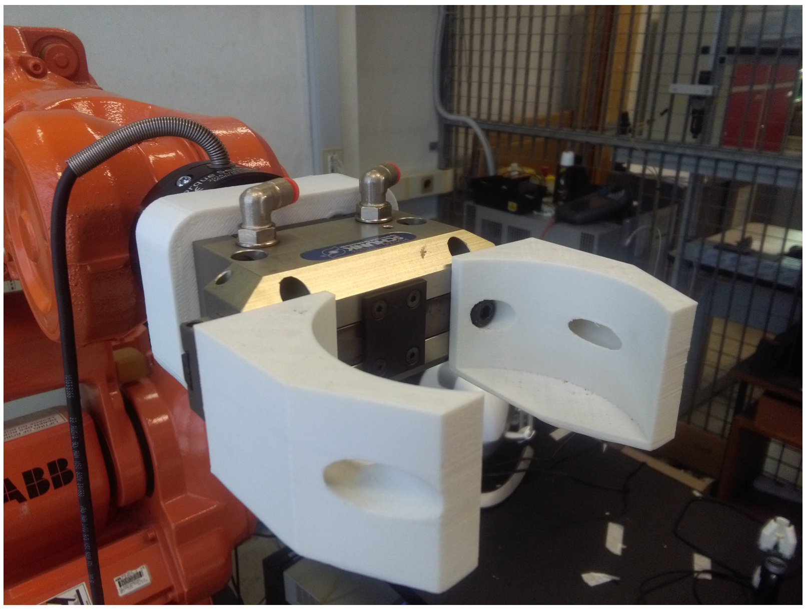


Figure 5.5: Orthogonal projection of the right gripper finger.

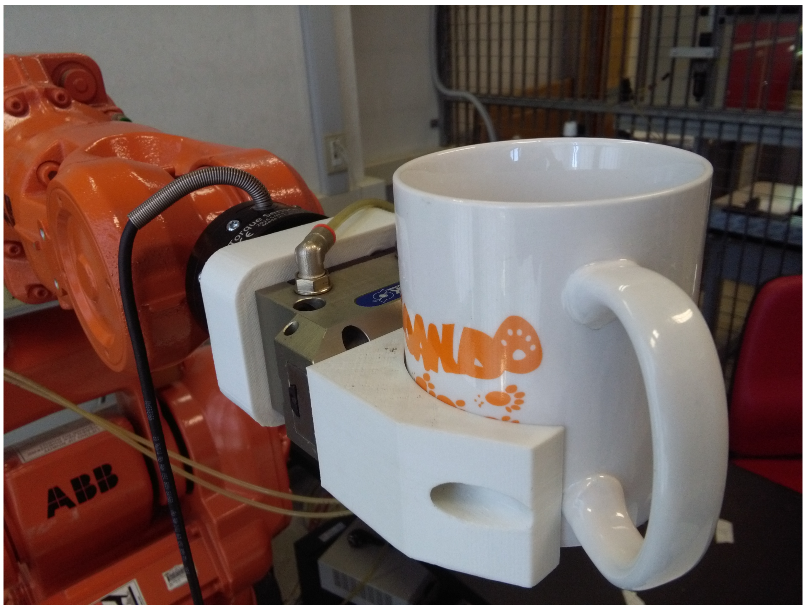
angle measurements. The extraction procedure can be performed multiple times on the same depth image, computing the average of the estimated sloshing angle to get rid of measurement noise. As already discussed, knowing the location of the liquid container with respect to the depth sensor allows for a proper selection of the region of interest that defines the area of each depth frame that must be converted into a point cloud. Therefore, we need to know the homogeneous transformation matrix A_{vision} from the sensor Cartesian reference frame C_s to the mug reference frame C_m. Let A_{robot} and A_{kinect} describe the transformations from the world frame C_w to the mug frame and the Kinect frame, respectively; then it holds (see Fig. 5.7):

$$A_{robot} = A_{kinect} A_{vision}$$
$$A_{vision} = A_{kinect}^{-1} A_{robot}$$

Where A_{robot} can be computed by means of robot direct kinematics, knowing the value of joint positions, while A_{kinect} has been derived based on measurements taken directly on the experimental set-up and samples of



(a) Open configuration.



(b) Closed configuration with mug.

Figure 5.6: Image of Schunk PGN 100 gripper mounted on the IRB 140 flange.

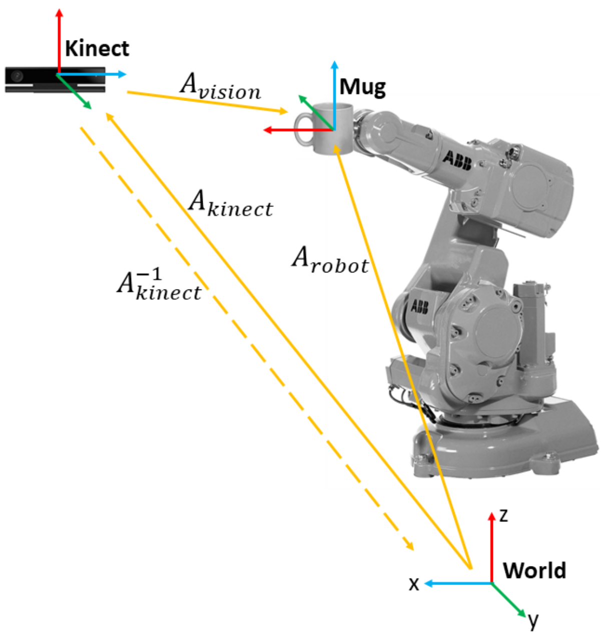


Figure 5.7: Scheme of mutual locations of world, Kinect and mug reference frames.

depth frame acquisitions. Its value is considered to be equal to:

$$A_{kinect} = \begin{bmatrix} -1 & 0 & 0 & 1.02 \\ 0 & 1 & 0 & -0.03 \\ 0 & 0 & 1 & 1.10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_{kinect}^{-1} = \begin{bmatrix} -1 & 0 & 0 & 1.02 \\ 0 & 1 & 0 & 0.03 \\ 0 & 0 & 1 & -1.10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with quantities of the translation vector representing meters. It is worth noticing that the determinant of the rotational part of the transformation matrix, i.e. the upper-left 3 by 3 matrix, is equal to -1. This happens because the reference frame that refers to the Kinect is a left-handed one. The reason for this choice is motivated by characteristics of the pinhole camera model: an image on the camera plane is indeed mirrored with respect to the real environment, thus a left-handed frame allows to retrieve the correct representation.

5.2 Simulation results

Prior to perform real experiments, identification of the model parameters of liquid sloshing dynamics is required. Afterwards, the complete control scheme has been simulated in different situations to test its behavior. Section 5.2.1 presents the results of parameter identification, while noteworthy outcomes of simulation experiments are described in Section 5.2.2.

5.2.1 Parameter identification

Since the knowledge of liquid dynamics is of great importance in several fields, in the literature it is possible to find several theoretical studies whose aim is to derive expressions for sloshing model parameters. Empirical formulae also exist to compute values for parameters of the pendulum-based model of sloshing dynamics, both for the complete non-linear equations in (4.9) and the linearized system described by transfer function (4.11). Overall, we need to know the pendulum length L, the sloshing mass M, the damping coefficient C and the transfer function parameters μ, ω_n and ξ. For instance, [18] states the expression for the length of the pendulum rod that models the first asymmetrical mode of sloshing, while in [1] an approximation for the damping ratio can be found. According to these relations, parameters values depend on tank geometry and liquid properties. In case of the mug used in the experiments we have that $R = 0.033\,\mathrm{m}$ is the tank radius and $d = 0.05\,\mathrm{m}$ the liquid depth, then:

$$\begin{aligned} L &= \frac{R}{1.841}\tanh^{-1}\left(\frac{1.841d}{R}\right) = 0.018\,\mathrm{m} \\ \xi &= 0.79\sqrt{Re}\left[1 + \frac{0.318}{\sinh\left(\frac{1.84d}{R}\right)}\left(1 + \frac{1-\frac{d}{R}}{\cosh\left(\frac{1.84d}{R}\right)}\right)\right] = 0.004 \end{aligned} \tag{5.1}$$

where Re stands for the reverse Reynolds number:

$$Re = \frac{\nu}{\sqrt{g\,(2R)^3}}$$

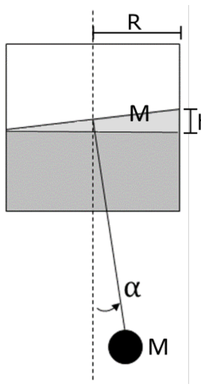


Figure 5.8: Sloshing mass M

and $\nu = 1.13 \times 10^{-6}\ \mathrm{m^2/s}$ is the kinematic viscosity of milk. Afterward, the sloshing mass is defined as the mass of the liquid undergoing a displacement. Fig. 5.8 sketches the situation, the moving volume of milk is equal to half a cylinder with radius R and height h, thus obtaining:

$$M = V\rho = \frac{\pi R^2 h \rho}{2} = 0.03\,\mathrm{kg} \quad \text{with} \quad h = 2Rsin(\alpha_{lim})$$

where $\rho = 1030\,\mathrm{kg/m^3}$ is the milk density and α_{lim} has been chosen to be 15°. Everything else can be then computed using relations written in equations (4.12), thus obtaining:

$$\omega_n = \sqrt{\frac{g}{L}} = 23.34\,\mathrm{rad/s} \qquad \mu = \frac{1}{g} = 0.102\,\mathrm{s^2/m}$$
$$C = 2\xi\omega_n M = 0.006\,\mathrm{Ns/m}$$

However, since the designed controller is an open-loop one, the more correctly the model describes sloshing dynamics the better performance are achieved by the control law. For this reason, model parameters have also been identified through experiments and then compared with the corresponding theoretical values. The general approach that has been followed consists in estimating natural frequency and damping ratio of milk dynamics by means of least-squares method using data coming from free motions of liquid surface inside the container. A detailed description of the identification procedure is presented in the remaining part of the section.

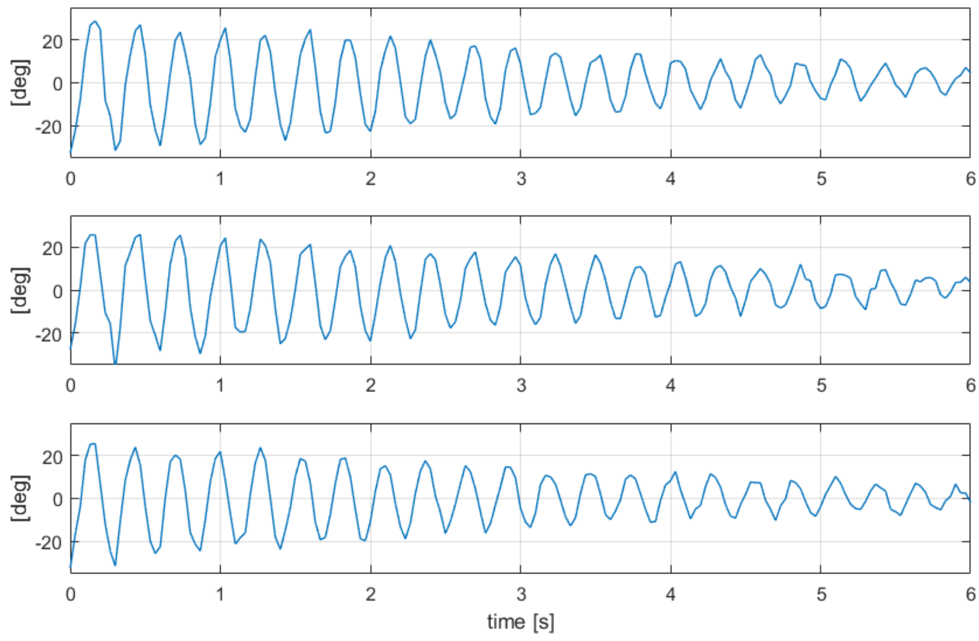


Figure 5.9: Examples of free motion acquisitions used for model identification purposes.

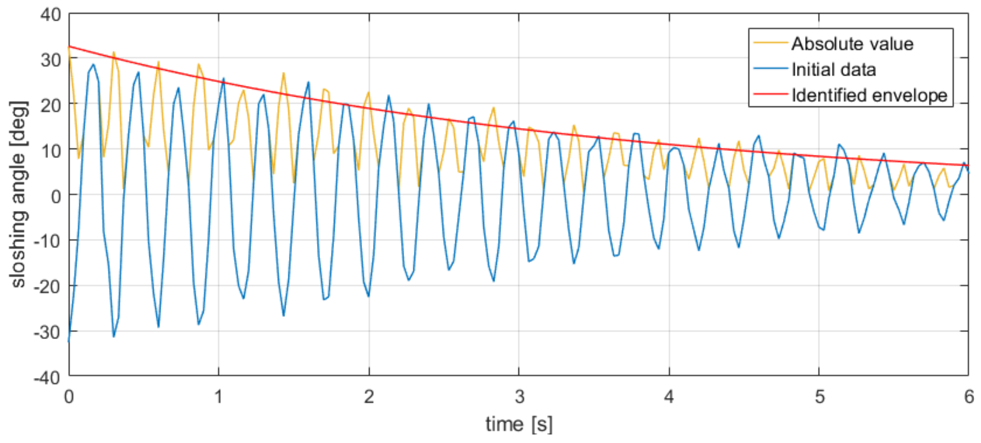


Figure 5.10: Example of acquired data and identified exponential envelope.

To obtain free motion of the liquid surface, impulse responses to acceleration input have been simulated on the mug and acquired using the Kinect depth sensor. In order to ensure a straight motion of the container, the mug has been placed on an horizontal surface between two rails. In total, ten runs have been acquired and for each run the sloshing angle measurements have been extracted ten times from depth frames. Firstly, data from each experiment have been cropped to only consider significant measurements, that is neglecting the initial instants when the mug is subjected to the input and the final ones when oscillations become negligible. Then, values coming from the same sampling time instant of the same run have been averaged to get rid of measurement noise, thus obtaining the free motion curves considered for least-squares computation. Figure 5.9 shows three examples of cropped and averaged free motion data used to identify model parameters.

The general form of the free motion of a one degree of freedom second-order linear system is given by:

$$y(t) = y_0 e^{-\sigma t}\cos(\omega_d t + \varphi) \qquad \sigma = \xi\omega_n \qquad \omega_d = \omega_n\sqrt{1-\xi^2}$$

where ω_d is the damped natural frequency and φ a phase delay. To obtain linear relations, it is required to consider the logarithm of the exponential envelope corresponding to the cosine equal to one.

$$\begin{aligned} y(t) &= y_0 e^{-\sigma t} \\ \log(y(t)) &= \log(y_0) - \sigma t \\ \sigma t - \log(y_0) &= -\log(y) \end{aligned} \tag{5.2}$$

The local maxima of the absolute value of each curve have been extracted to obtain twice as many points for the exponential envelope as shown in Fig. 5.10, which also displays the exponential envelope computed with the identified parameter values. From equation (5.2) it is possible to derive the expression used in the least-squares method to compute a value for σ

as:

$$\begin{bmatrix} t_1 & 1 \\ \vdots & \vdots \\ t_n & 1 \end{bmatrix} \begin{bmatrix} \sigma \\ -\log(y_o) \end{bmatrix} = \begin{bmatrix} -\log(y_1) \\ \vdots \\ -\log(y_n) \end{bmatrix}$$

$$A_{ls}\boldsymbol{x} = \boldsymbol{b}_{ls}$$

$$\boldsymbol{x} = \left(A_{ls}^T A_{ls}\right)^{-1} A_{ls}\boldsymbol{b}_{ls}$$

$$\sigma = \boldsymbol{x}(1)$$

where subscript $i = 1, 2 \dots n$ indicates the i-th data point of the run. Next, the circular frequency of the damped system has been identified as the mean time interval between two consecutive maxima of $y(t)$.

$$T = \frac{t_n - t_1}{n - 1} \qquad \omega_d = \frac{2\pi}{T}$$

At this point, an estimate of the values of parameters ω_d and σ is obtained for each run. Since it is correct to assume that ω_d and σ are properties of the system and do not change in different experiments, the final values for the parameters are simply computed as the average of the results from each run. Moreover, since the damping ratio is expected to be very low, as resulting from theoretical computations in equation (5.1), it is possible to assume $\omega_n \approx \omega_d$; hence, $\xi = \sigma/\omega_n$. Finally, once ω_n and ξ have been found and considering μ and M to have the same values as previously computed, C and L follow form relations in (4.12). Table 5.2 shows the results of parameters identification for the ceramic mug and compares them with the theoretical values. As it is apparent, the natural frequency takes consistent values. On the other hand, the identified damping ratio is different from the expected theoretical value. However, as stated in [1], when the container dimensions are small, as in the case of the milk mug, the liquid surface tension plays a prominent role to determine the amount of damping in the system, which is thus increased with respect to the quantity predicted by the empirical formula. Eventually, results have been validated comparing the output of the identified model with new free motion acquisitions that were not used in the identification procedure;

Table 5.2: Comparison of theoretical and identified sloshing model parameters

Parameter		Theoretical	Identified	Variation %
Natural frequency ω_n	rad/s	23.34	22.65	-2.96
Damping ratio ξ	–	0.004	0.012	200
Gain μ	s^2/m	0.102	–	–
Sloshing mass M	kg	0.03	–	–
Damping coefficient C	Ns/m	0.006	0.016	166.7
Pendulum length L	m	0.018	0.019	5.56

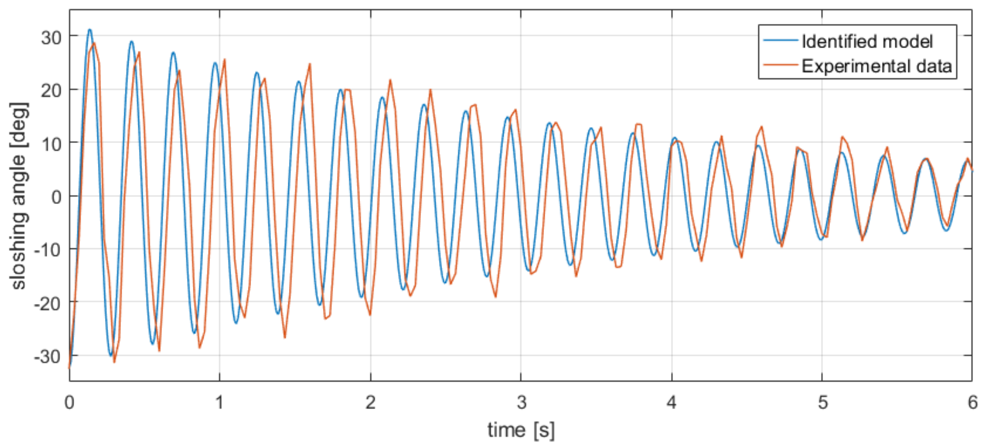


Figure 5.11: Comparison between experimental data and identified model.

as one can see from the example in Figure 5.11 the model consistently describes experimental data.

The same identification procedure has been performed to derive a model of sloshing dynamics inside the glass. In this case the obtained natural frequency is $\omega_n = 20.84\,\text{rad/s}$, whereas the damping ratio ξ is equal to 0.017, as for the mug model.

Table 5.3: Initial and target positions used in simulation expressed in joint and task spaces.

Initial position	
Joint space	Task space
$[30°, 20°, 10°, 0°, -30°, 0°]^T$	$[0.509, 0.294, 0.500, \pi/6, \pi/2, \pi]^T$
Target position	
Joint space	Task space
$[-40°, 50°, 20°, 0°, -70°, 0°]^T$	$[0.414, -0.348, 0.224, -2\pi/9, \pi/2, \pi]^T$

5.2.2 Simulations

The aim of the performed simulations is to validate the complete control architecture before implementing it on the real system. Moreover, simulation results allow to analyse performance and underline peculiar features of the control algorithm, that are useful to better understand the outcome of experiments carried out on the physical robotic system. In order to do so, the entire control scheme has been implemented in Matlab-Simulink environment together with a model of the industrial manipulator. This section presents the results coming from a single robot trajectory, whose initial and target states of motion are displayed in Tab. 5.3. Nevertheless, experimental results in the next Section prove that the designed control architecture is capable to behave well also for different motion paths. The robot is always supposed to start and complete the motion at zero speed.

Reference test

First of all, a simulation test has been made without imposing any limit to the maximum sloshing angle α, thus considering $\alpha_{lim} = \infty$. This is used as a base reference to compare all other simulation outcomes. In Figures 5.12 and 5.13 it is possible to see the robot trajectory in the task and joint spaces, whereas Fig. 5.14 plots the behaviour of the linearized pendulum angles $\delta\vartheta$ and $\delta\varphi$, together with the sloshing angle α. As one can see, the motion is a straight segment in the task space between the initial and target position; it is important to notice how the last two task variables are correctly constrained to a constant value in order to keep the liquid container in the upright position. The time response of the sloshing angle is mostly affected by container accelerations, in fact it reaches its highest values at the initial and final instants of motion, when the robot is accelerating to start moving and decelerating to stop; after the end of the trajectory the liquid surface undergoes a free motion until reaching the horizontal equilibrium position.

Constrained tests

Since the sloshing angle α stays always below 20°, α_{lim} has been set equal to 15°, and then 10°, in the following simulations when the spilling avoidance constraint has been tested. Resulting sloshing angle curves are displayed in Figures 5.15 and 5.16. One can see that the limit on α is satisfied at any time instant, even if the spilling avoidance constraint is based on some approximations of the sloshing dynamics model, specifically linearization of model equations and upperbound of time responses. However, this is not sufficient to say that performance are good: it is worthy to analyse also how motion is modified by the constraint in terms of trajectories and time of execution. On one hand, both when task and joint spaces are considered, trajectories for each variable do not change significantly with respect to the reference one. As expected, differences are mainly in the initial and final part of motion, when the spilling avoidance constraint activates reducing accelerations. As examples, Figure 5.17 compares the trajectories of the third element of task variables vector, with details of motion start and finish, whereas Fig. 5.18 does the same with $q_2(t)$. As far as joint variables are concerned, trajectories move from the reference for at most 0.5°. On the other hand, limits to the sloshing angle are satisfied with an increment of the motion execution time that remains below 5%. It is worth noticing that the additional time needed to satisfy the spilling avoidance constraint is very low if compared with that required to achieve complete sloshing suppression in related works, such as [23], [27] or [28], where the time increment goes from 20% to more than 30%.

Performance tests

Simulations performed so far indicate that the developed control law is able to correctly handle liquid containers while limiting sloshing and it has been shown that it is possible to set different values for α_{lim}. Although in principle there should be no lower bound to the maximum allowed sloshing angle, simulations results contradict this statement. Figure 5.19 presents simulation outcome when $\alpha_{lim} = 7°$ for the same robot trajectory of the previous tests. If on one hand the controller manages to satisfy the spilling avoidance constraint at any time instant, on the other hand it is not able to correctly reach the target position. Instead, the motion never stops and the robot start oscillating around its goal; specifically, only x_1 and x_2 variables keep fluctuating in the task space, but that results in an oscillation of all joints, as shown by the plot in 5.19a. This is of course a limitation of the control solution presented in this thesis, although it arises only with small value of sloshing angle bound. Moreover, since the spilling avoidance is still ensured, in cases when the accuracy in reaching the final position is not a strict requirement, one can think of an ad-hoc logic to stop the motion in a neighborhood of the target. An example of this kind of application could be that of a household robot, handing out a cup to a person: in this case it is not really important to move exactly in a predetermined point, but more to get close to the human.

In Section 4.4, the complete control architecture has been described highlighting the fact that the controller is split into two part: one running with a sampling period of 4ms, the other running at 12ms. In particular,

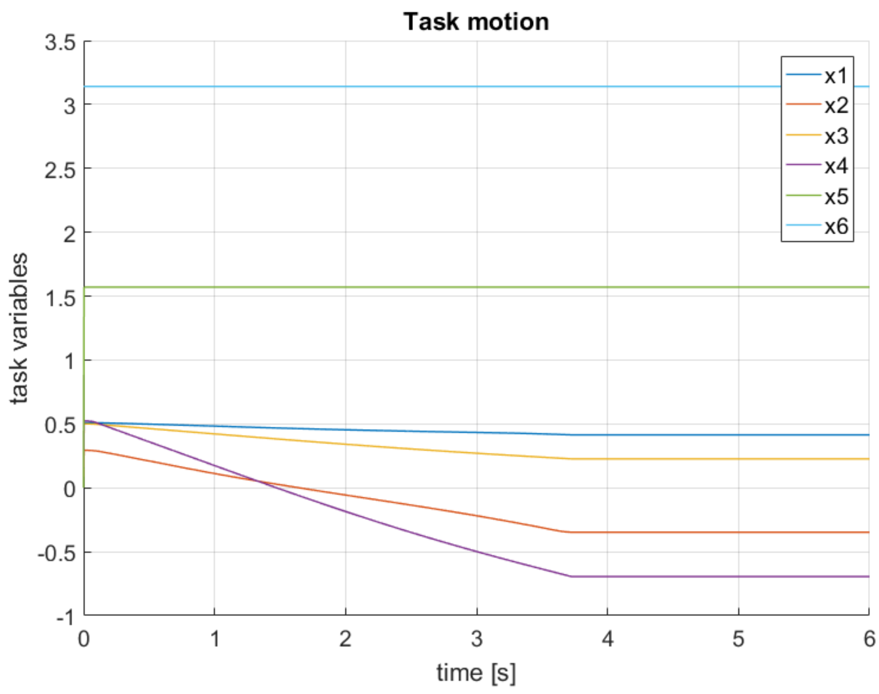


Figure 5.12: Robot trajectories in task space obtained with $\alpha_{lim} = \infty$.

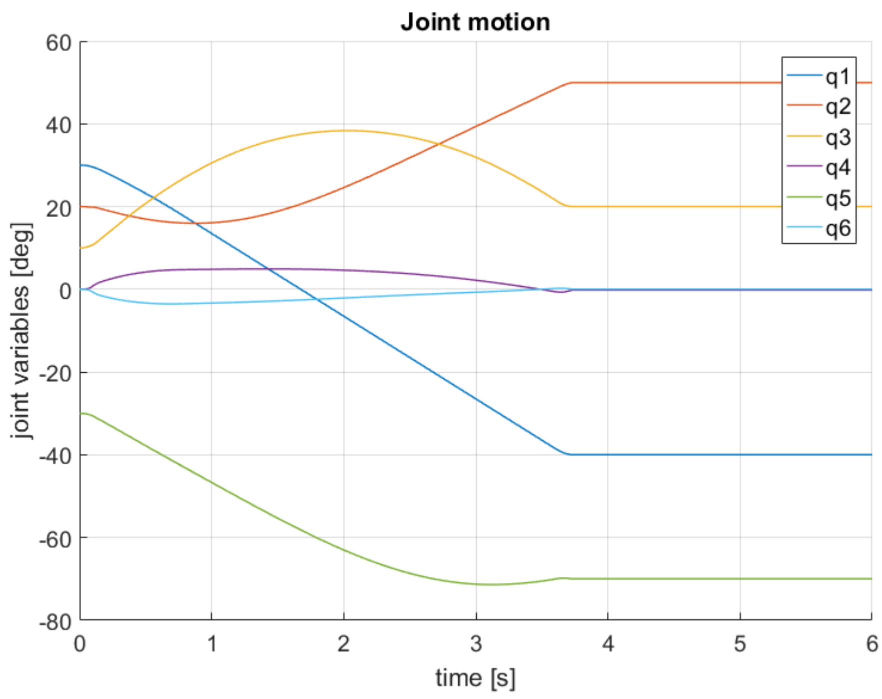


Figure 5.13: Robot trajectories in joint space obtained with $\alpha_{lim} = \infty$.

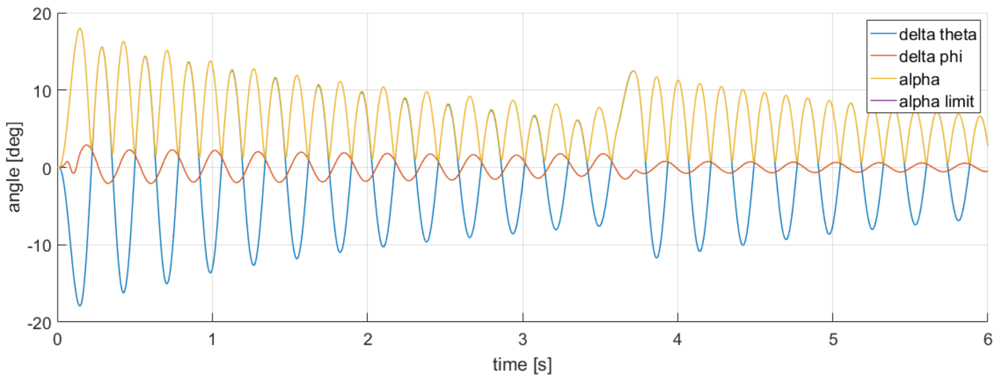


Figure 5.14: Pendulum and sloshing angles obtained with $\alpha_{lim} = \infty$.

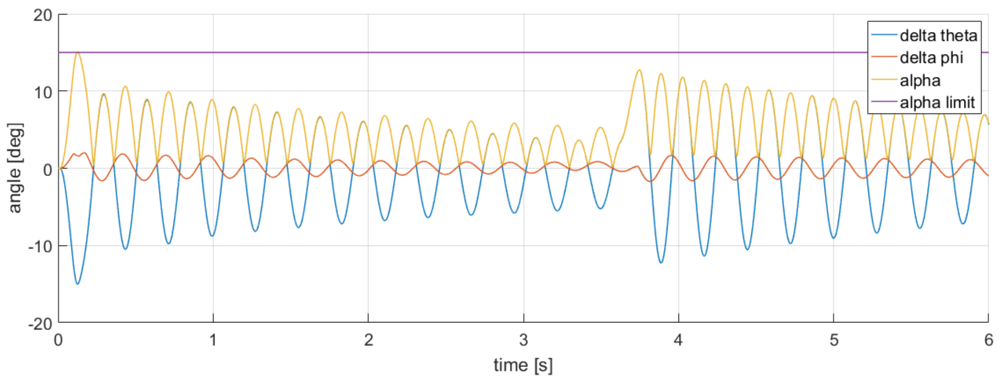


Figure 5.15: Pendulum and sloshing angles obtained with $\alpha_{lim} = 15°$.

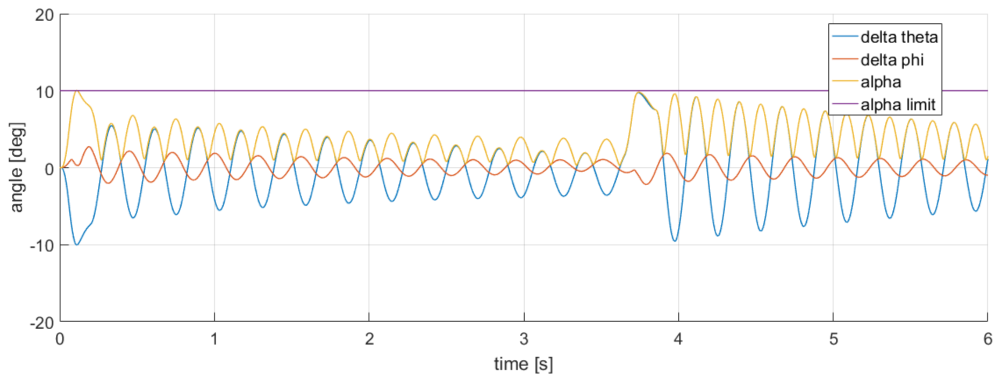


Figure 5.16: Pendulum and sloshing angles obtained with $\alpha_{lim} = 10°$.

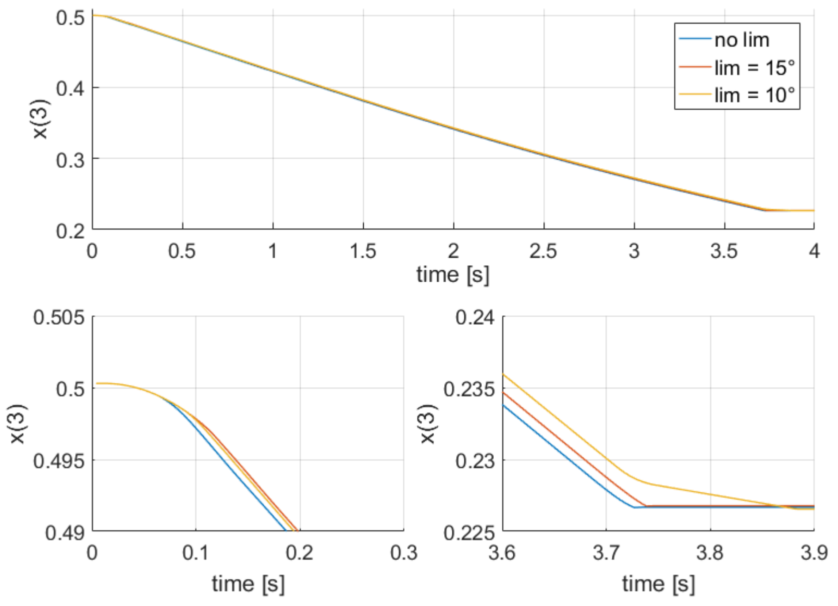


Figure 5.17: Comparison of the trajectories of $x_3(t)$ for different α_{lim}. Full motion (top), initial (bottom-left) and final (bottom-right) part.

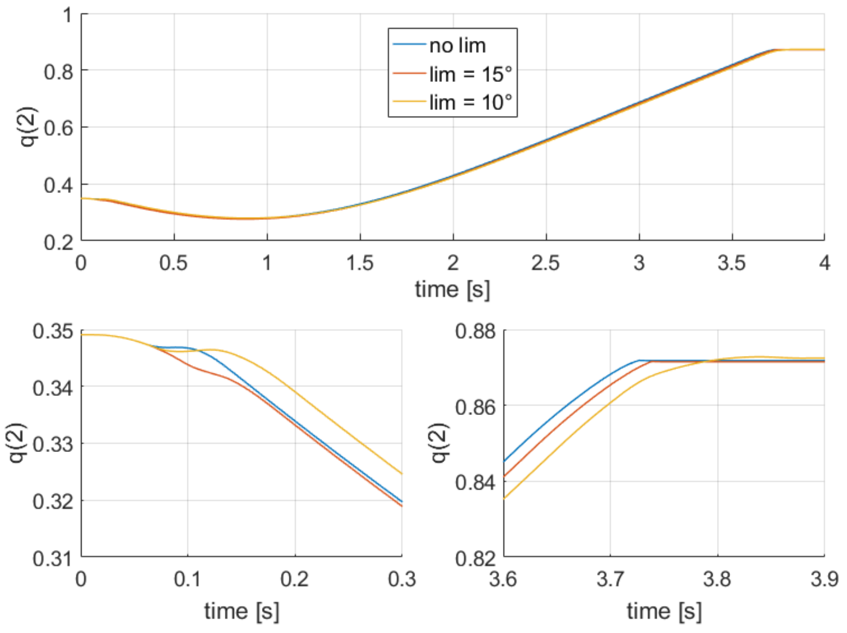
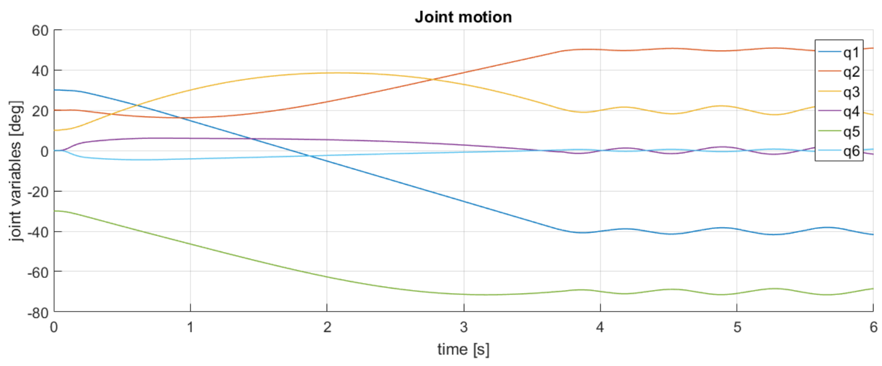
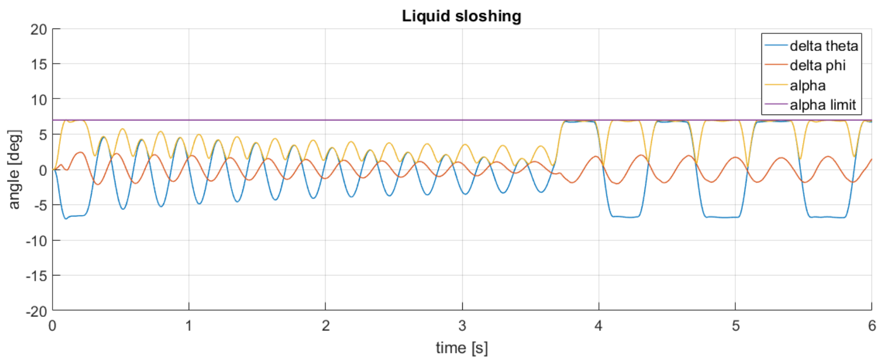


Figure 5.18: Comparison of the trajectories of $q_2(t)$ for different α_{lim}. Full motion (top), initial (bottom-left) and final (bottom-right) part.



(a) Robot trajectories in joint space.



(b) Pendulum and sloshing angles.

Figure 5.19: Simulation outcomes with $\alpha_{lim} = 7°$: the controller is not able to stop the motion

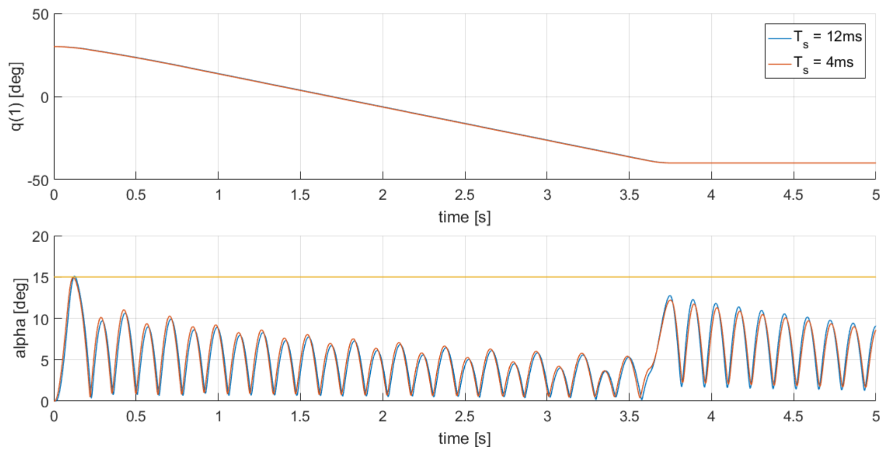


Figure 5.20: Performance loss due to increased T_s for QCQP problem solution. First joint trajectory (top) and sloshing angle response (bottom).

Table 5.4: Initial and target positions used for experiments expressed in joint and task spaces.

Initial position	
Joint space	Task space
$[19°, 37°, -30°, 0°, -7°, 0°]^T$	$[0.689, 0.237, 0.593, 19\pi/180, \pi/2, \pi]^T$
Target position	
Joint space	Task space
$[40°, 70°, -35°, 0°, -35°, 0°]^T$	$[0.601, 0.504, 0.257, 2\pi/9, \pi/2, \pi]^T$

the slower part implements a quadratically constrained optimization problem (QCQP), which requires time to be solved also when state-of-the-art algorithms are used. The output of the minimization are the robot joint accelerations to be commanded at the current time instant. With this in mind, it is worth investigating how the decrease in frequency of commands delivery impacts the overall performance of the controller. Therefore, a test simulating the ideal situation of a control algorithm working entirely at 250 Hz has been compared with the outcome of the actual control scheme. Figure 5.20 plots results of computer-generated robot motion in the case $\alpha_{lim} = 15°$; it is possible to conclude that neither trajectories nor the sloshing angle are significantly affected and there is no degradation of performance in the control scheme which is actually implemented in practice.

5.3 Experiments

This Section presents the results obtained applying the designed controller to the robotic system described in Section 5.1. A first series of experiments have been conducted using a glass full of milk as a container, then the same tests have been repeated using a mug to contain the liquid in order to test the control algorithm against a different sloshing behavior.

Liquid height has been set equal to 4.5 cm inside the glass and equal to 5 cm inside the mug. The initial and final target positions for robot trajectory are shown in Table 5.4 and they are different from those used in the simulations described in the previous Section. The reason is twofold: firstly, this can act as a proof that the control algorithm is capable of governing different robot motions; secondly, the trajectory for the experiments has been chosen in order to achieve a positioning of the liquid container inside the Kinect sensor field of view at each time instant. The same motion has been performed with different values for the sloshing bound α_{lim}, namely ∞ (no limit), 15° and 10°. Figure 5.21 shows the trajectory followed by the robot in task and joint space when no limit is imposed to the maximum sloshing angle.

In order to give a better understanding of the execution of experiments, Figure 5.22 shows frames taken from a video of the robot in a case when the glass is used to contain the liquid. In Fig. 5.22a robot is in the initial position and the liquid is still, then the robot starts moving (Fig. 5.22b) causing milk sloshing. Motion continues until the industrial manipulator reaches the target configuration and stops (see Fig. 5.22e). Finally, the liquid undergoes a free motion until it gets to the rest position (Fig. 5.22f).

In the following, the experimental results obtained in different cases are presented. Measurements of the sloshing angle are acquired using a Kinect depth sensor and the algorithm described in Chapter 3.

Firstly, a transparent glass has been used and an initial run has been performed with no imposed limit to the maximum sloshing angle. It is possible to perform this experiment because the conservative bounds on the maximum robot joint velocities that are chosen for safety reasons make sure that the liquid never spills out the container. Figure 5.23 shows the measured time response of α and the angles ϑ and φ of the pendulum-based sloshing model. Values coming from the Kinect sensor are compared with the expected behavior obtained through simulation, whereas the vertical dashed line indicates the instant when the robot reaches the target state

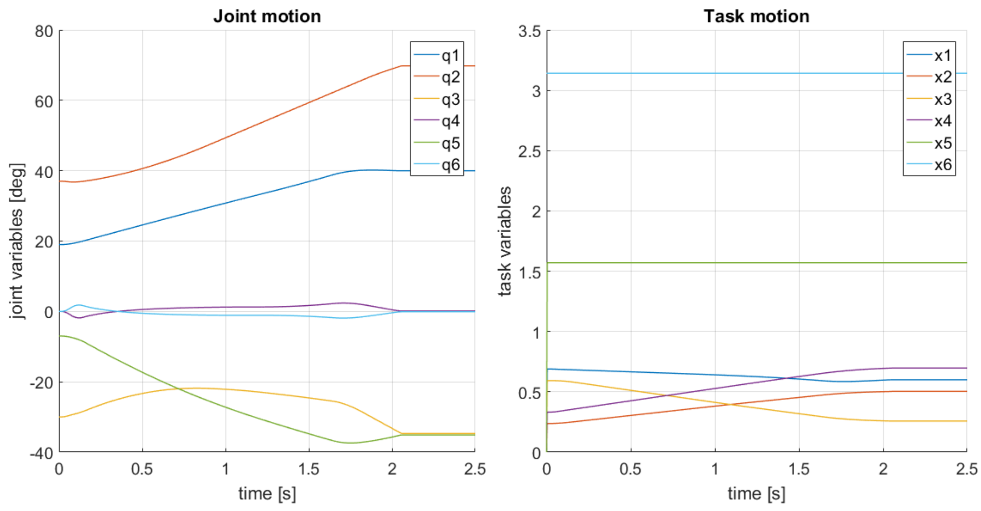


Figure 5.21: Trajectory of the robot in joint space (left) and task space (right).

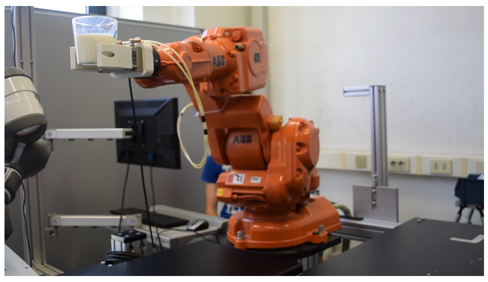
and the motion stops. After that time instant, the liquid surface undergoes a free motion until it reaches the rest equilibrium. Overall, it is possible to say that the real liquid behavior follows the model, both in term of the sloshing angle and the two state angles of the pendulum. However, some exceptions are present, such as the missing peak around 1 s, that are probably due to measurement errors. One important thing to notice is that, after the end of motion, the measured sloshing is greater then expected. The larger amplitude of the liquid surface free motion is caused by vibrations of the robot that arise at the moment when the motion stops. Such vibrations are not modeled in the robot dynamics and generate additional input accelerations that increase sloshing of the milk inside the glass.

Afterwards, α_{lim} has been set to finite values. Again, Kinect sensor is used to measure liquid surface oscillations. Results when the sloshing bound is set equal to 15° are presented in Figure 5.24. It is apparent that the control algorithm is able to correctly satisfy the spilling avoidance constraint. In fact there is only one point around 1.4 s which is above the limit, but it is clearly a measurement error, as it is completely out of range.

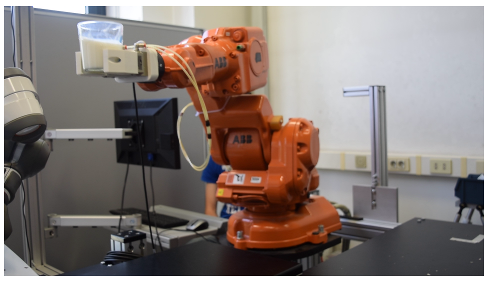
Moreover, the time response obtained on the real system is consistent with the one expected from simulation results. On the other hand, when the limit is further reduced to 10° (see Fig.5.25) more points occur to be outside the maximum sloshing angle bound, although the general behavior is still consistent with the expected one. In this case, apart from possible measurement errors, critical points of motion are the initial part, when the robot starts accelerating, and the very end of the motion, when the robot stops. Regarding the latter, as previously said the reason is to be searched for in vibrations that arise at motion stop, which are not controlled and have a greater impact when the sloshing angle is kept small. As far as the first critical point is concerned, it is possible that higher frequency modes of vibrations are not completely negligible when the sloshing angle becomes small.

Figures from 5.26 to 5.28 shows the experimental results obtained using a ceramic mug as liquid container instead of the glass. Cup dimensions and liquid depth are different from those of the glass, thus the system natural frequency and damping ratio are not the same, as discussed in Section 5.2.1. Therefore, parameter values of the sloshing dynamics model inside the control scheme have been changed accordingly in order to compute the right control commands. On the other hand, trajectories and values for α_{lim} are equal to the previous series of experiments. From the obtained results it is possible to say that the controller is still able to satisfy the spilling avoidance constraint. Moreover, performance are not affected by the change in sloshing dynamics. However, measured angles responses differ from simulation curves more than those obtained during experiments performed using the glass as container. This is not a matter of poor control, but a measurement issue. In fact, sloshing angle extraction from depth images is in general less reliable when the container is opaque, as in the mug case, rather than when the liquid is contained in a transparent vessel. This happens because an opaque container occludes part of the liquid surface, which is thus out of the Kinect field of view. Moreover, during movement, the occluded part may further increase, making the sloshing estimation more difficult. In fact, liquid plane identification is only based on the visible part of the fluid surface; if the visible points are few, the plane fitting is more unreliable. One possible solution to solve the problem could be a more vertical positioning of Microsoft Kinect sensor, in order to reduce occlusion, as well as the use of two depth sensors placed with orthogonal orientations. However, the latter solution would require two image streams, leading to a high computational load needed to retrieve sloshing measurements.

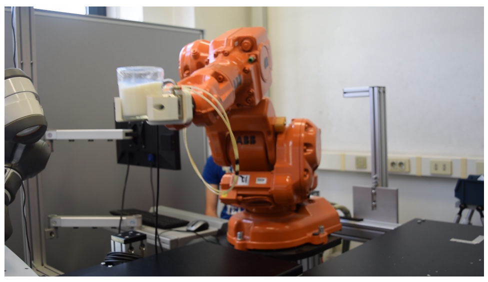
Overall, the control architecture proposed in this thesis behaves well in handling different liquid containers while ensuring spilling avoidance. However, measuring the sloshing angle is still a challenging problem to cope with.



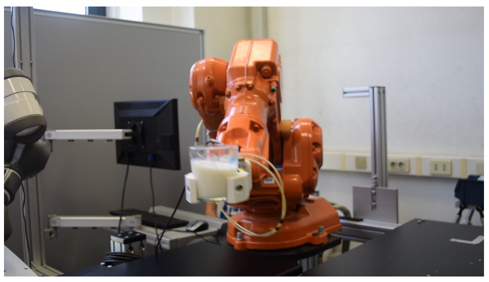
(a)



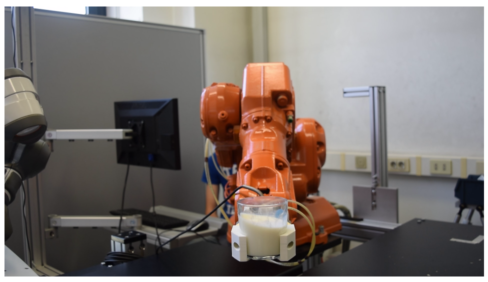
(b)



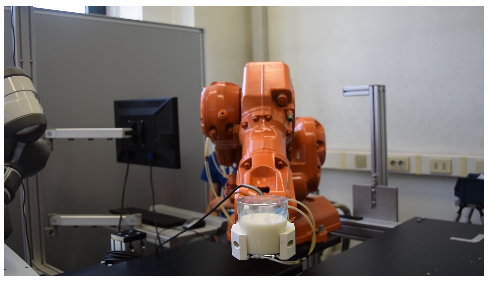
(c)



(d)



(e)



(f)

Figure 5.22: Screenshoots from video of experiment with robot holding glass and $\alpha_{lim} = 15°$.

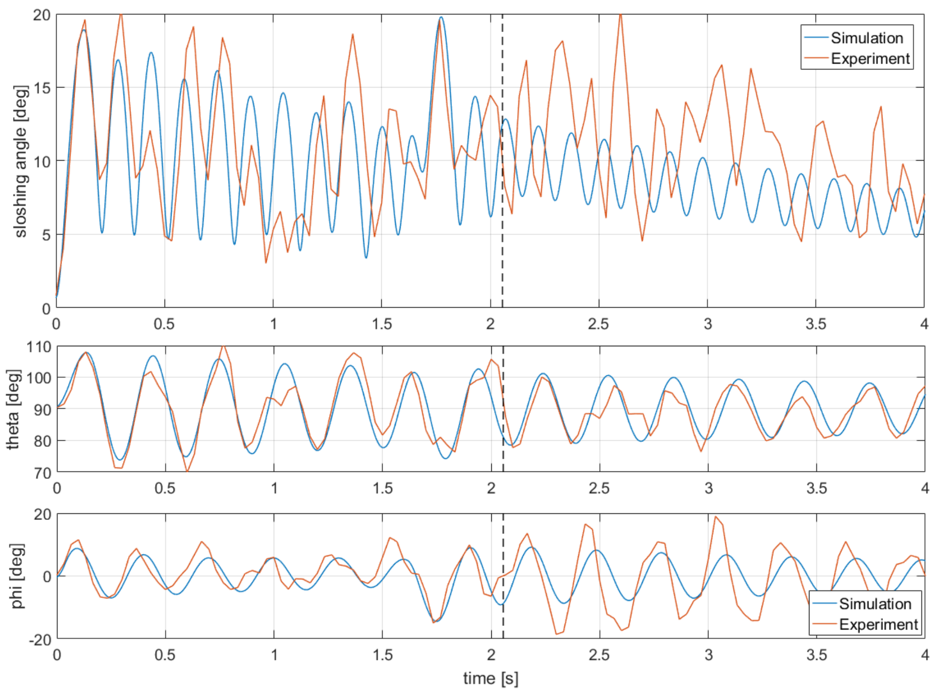


Figure 5.23: Experimental results with $\alpha_{lim} = \infty$, glass case.

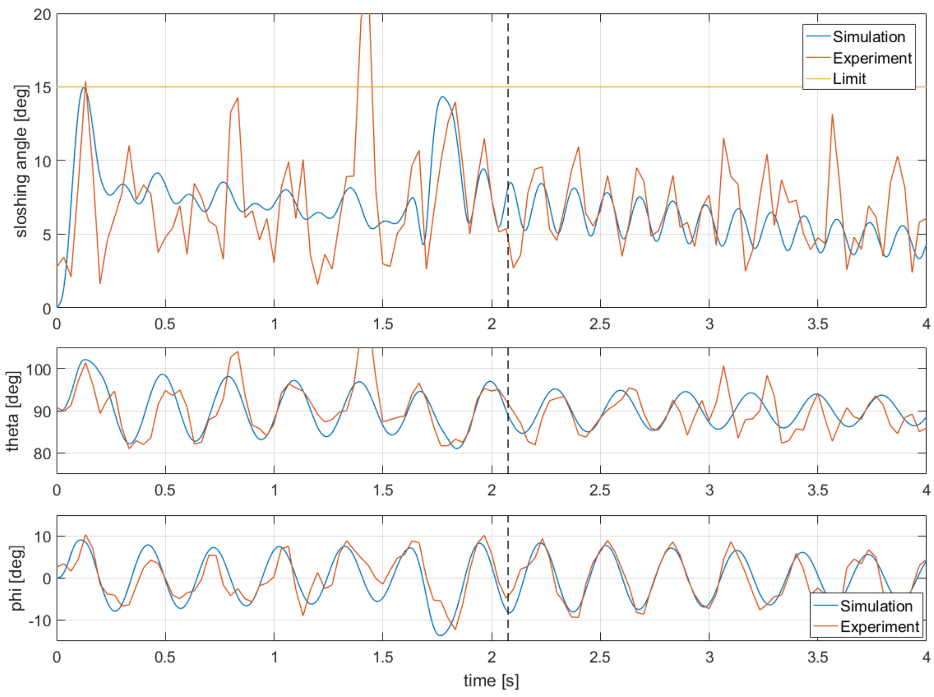


Figure 5.24: Experimental results with $\alpha_{lim} = 15°$, glass case.

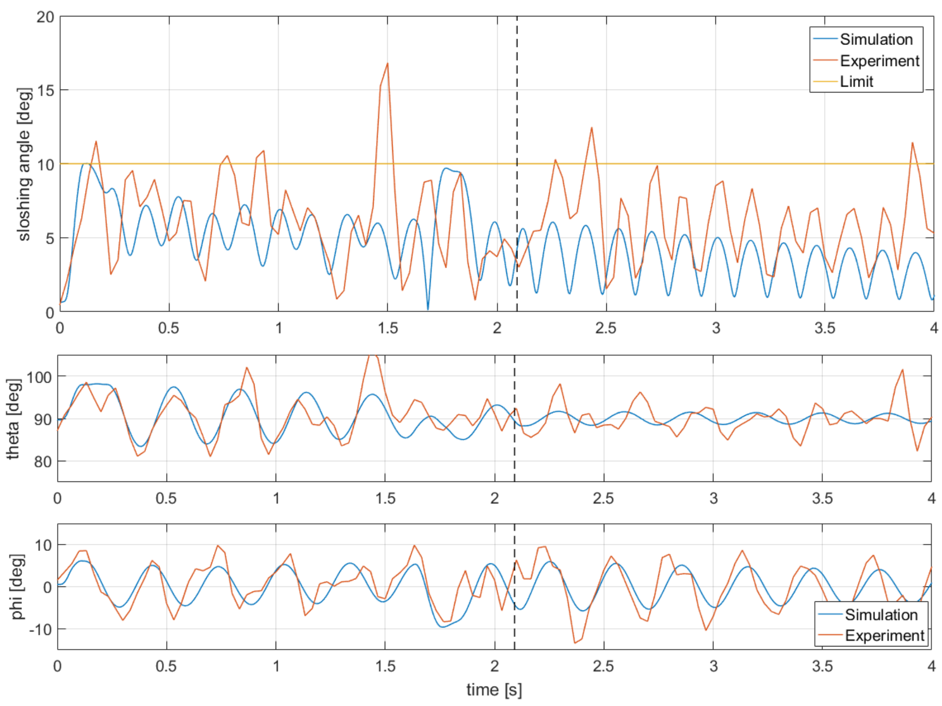


Figure 5.25: Experimental results with $\alpha_{lim} = 10°$, glass case.

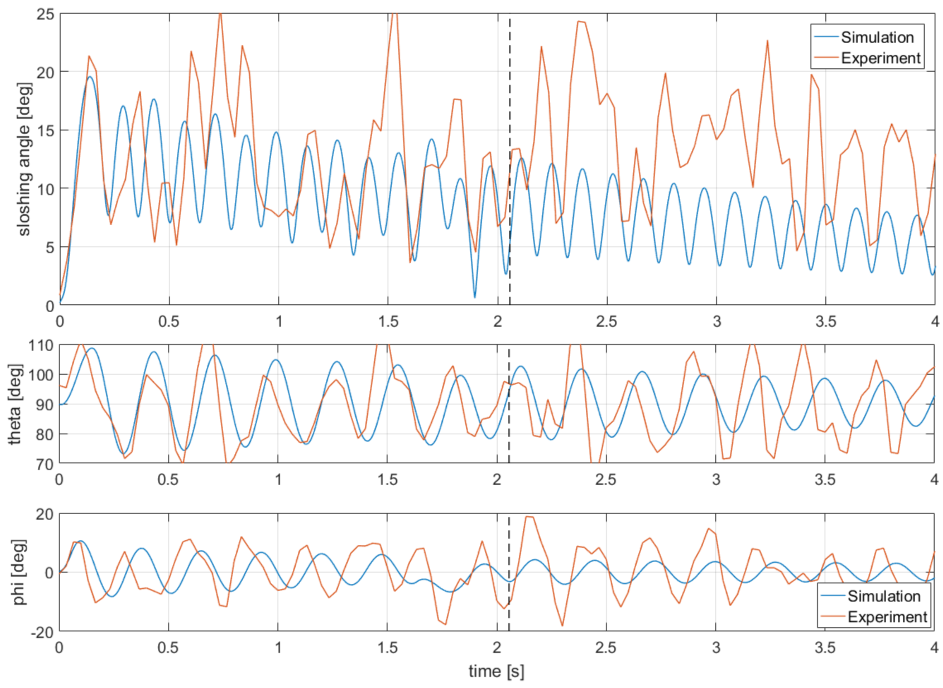


Figure 5.26: Experimental results with $\alpha_{lim} = \infty$, mug case.

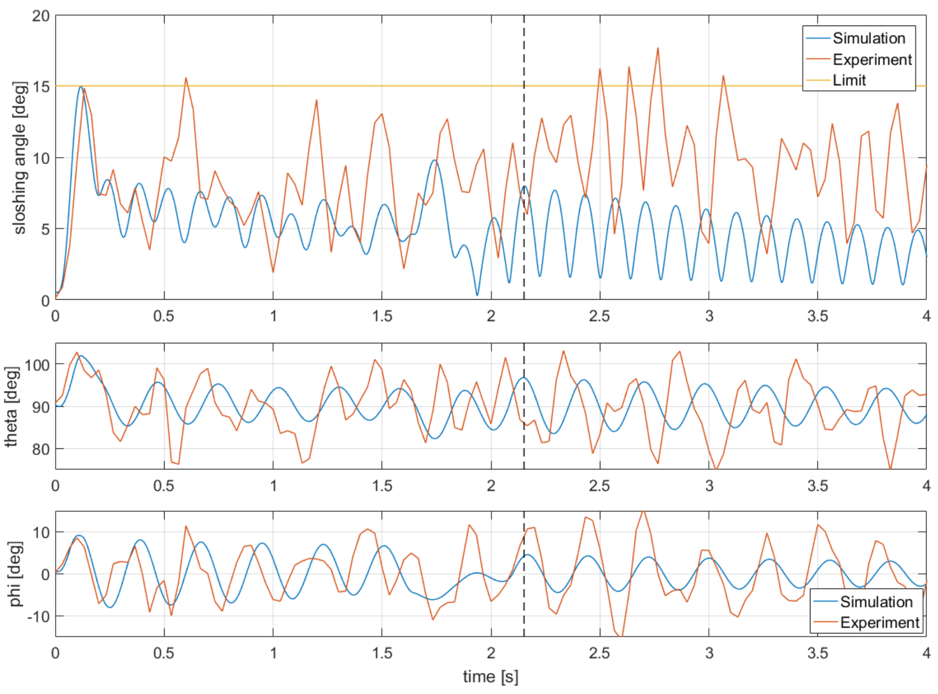


Figure 5.27: Experimental results with $\alpha_{lim} = 15°$, mug case.

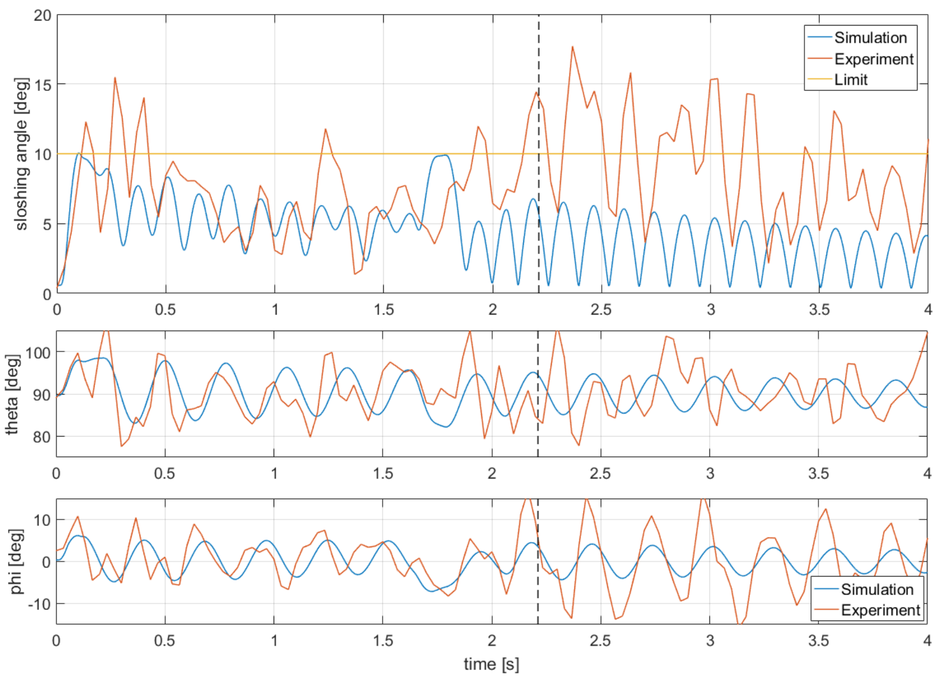


Figure 5.28: Experimental results with $\alpha_{lim} = 10°$, mug case.

Chapter 6

Conclusion and Future Work

This chapter summarizes the results achieved in the thesis and suggests possible future developments.

The aim of the present work is 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 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. 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. Sloshing measurements are retrieved from images streamed by a depth sensor.

Experimental results show that the controller is able to prevent the liquid from spilling out the container and to cope with different maximum sloshing bounds and different liquid containers. In order to do so, only small deviations from the planned trajectory and a small increase in the execution time are required. Limitations to the designed control law exist mainly in the form of lower-bounds to the value that can be set for α_{lim}. Firstly, for very small sloshing angle limit the controller does not manage to correctly stop the motion even if it is still able to ensure spilling avoidance.

Secondly, unmodeled vibrations that arise when the robot stops increase the amplitude of free motion oscillations. The more conservative the constraint, the more critical is this effect.

A natural extension of this work is a modification toward a closed-loop control scheme. In order to do so, measurement of the sloshing angle coming from the analysis of depth images must be retrieved on-line and at real-time. Then, measurements are sent to the reactive controller instead of the values of the state variables of the sloshing model computed through simulation. Several problems have to be considered:

- Sloshing measurements extraction must be fast and reliable: in this thesis analysis of depth images is performed off-line and it is sometimes affected by measurement errors, however the adopted solution does not suit a closed-loop controller. One idea to solve the problem is to fully exploit the knowledge of the relative position between sensor and liquid container, thus using real-time information of the robot end-effector position to directly identify the liquid surface in the depth image, reducing the steps of the extraction algorithm. The problem of occlusions due to the use of opaque containers must be also taken into account, since poor measurements of the sloshing angle can compromise performance of the closed-loop control law.

- Depth sensor acquisition rate does not match the controller frequency: Kinect sensor streams depth data at 30 Hz while the controller works with a sapling period of 12 ms, required to solve the optimization problem. However, sloshing measurements from depth data have to be input for the control law. A method to make them compatible with the controller frequency is required.

- Transparent liquid handling: this thesis does not deal with sloshing angle measurement for transparent liquids, as the open loop solution can be applied to all kind of fluids without directly measuring sloshing. However, the topic has to be addressed to obtain a closed loop solution which is able to handle transparent liquid. Papers that can be found in the literature, like [11] and [17], suggest that the problem can be solved using only RGB-D camera information.

Another possible extension to the present work is to develop a control law that does not require prior knowledge on the liquid and tank properties, especially the natural frequency. This is of increasing interest in the field of service robotics, where robots works in unstructured environments and can deal with continuously new liquids. Other works already pay attention to this problem, such as [11] and [20].

Appendix A

ZYZ Euler angles

This appendix wants to give further details about ZYZ Euler angles, that are used in the thesis to define the robot end-effector orientation.

Euler angles gives a minimal representation of the orientation of an object by means of three subsequent rotations, which are needed to describe the transformation from the reference world frame to the object frame. In particular, ZYZ Euler angles are defined as:

1. a rotation of an angle o_{z_0} around $z-$axis of the world frame to go from the world frame (x, y, z) to an intermediate frame (x', y', z');

2. a rotation of an angle o_y around the y'-axis obtaining a new (x'', y'', z'') frame;

3. a rotation of an angle o_{z_1} around the z''-axis to reach the object frame orientation.

Figure A.1 gives a graphical representation of the three stages.

Let the following abbreviations hold:

$$\begin{array}{lll} c_0 = cos(o_{z_0}) & c_y = cos(o_y) & c_1 = cos(o_{z_1}) \\ s_0 = sin(o_{z_0}) & s_y = sin(o_y) & s_1 = sin(o_{z_1}) \end{array}$$

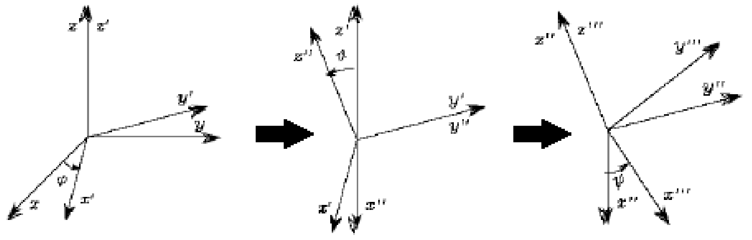


Figure A.1: Euler angles rotations.

Then the rotation matrix which corresponds to Euler angles rotations is:

$$R(o_{z_0}, o_y, o_{z_1}) = \begin{bmatrix} c_0c_yc_1 - s_0s_1 & -c_1s_0 - c_0c_ys_1 & c_0s_y \\ c_yc_1s_0 + c_0s_1 & c_0c_1 - c_ys_0s_1 & s_0s_y \\ -c_1s_y & s_ys_1 & c_y \end{bmatrix}$$

It is important to notice that a singularity of representation occurs if $sin(o_y) = 0$ and thus $o_y = k\pi \quad \forall k \in \mathbb{N}$.

On the other hand, if the rotation matrix is already known and $o_y \in (0, \pi)$, it is possible to compute the related Euler angles as:

$$\begin{aligned} o_{z_0} &= atan2(r_{23}, r_{13}) \\ o_y &= atan2(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}) \\ o_{z_1} &= atan2(r_{32}, -r_{31}) \end{aligned}$$

where r_{ij} represents the element (i, j) of the rotation matrix R.

When the differential kinematic is considered, it is important to know how to relate the geometric Jacobian J with the analytic Jacobian J_A, which depends upon the chosen minimal representation for the orientation. In case of the ZYZ Euler angles the two quantities are linked by the equation:

$$J = \begin{bmatrix} I & 0 \\ 0 & T(\boldsymbol{o}) \end{bmatrix} J_A \qquad T(\boldsymbol{o}) = \begin{bmatrix} 0 & -s_o & c_0s_y \\ 0 & c_0 & s_0s_y \\ 1 & 0 & c_y \end{bmatrix}$$

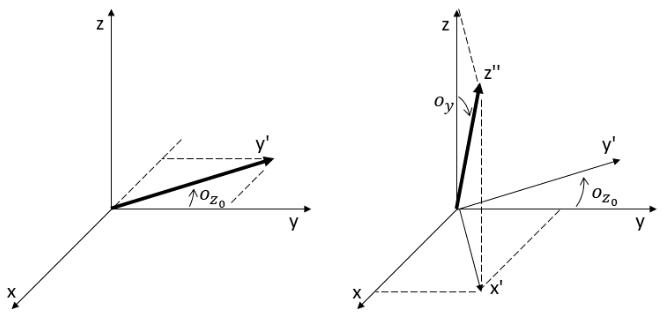


Figure A.2: Projection of y'-axis (left) and z''-axis (right) on the world frame axes

Where $\boldsymbol{o} = [o_{z_0} \quad o_y \quad o_{z_1}]^T$ stands for the vector of Euler angles, while matrix $T(\boldsymbol{o})$ defines the transformation from the derivative of the Euler angles $\dot{\boldsymbol{o}}$ to the object's angular velocities vector $\boldsymbol{\omega} = [\omega_x \quad \omega_y \quad \omega_z]^T$. To derive the expression of T, it is possible to consider the angular velocity vector arising from each of the Euler angles elementary rotation seen in Fig. A.1. In particular, for the first rotation around z-axis it trivially holds that:

$$\omega = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{o}_{z_0}$$

For the second and third rotation is it possible to look at the components of the axes of rotation, namely y' and z'', on the world frame axes. The situation is graphically represented in Figure A.2. Therefore, one can see that it holds:

$$\omega = \begin{bmatrix} -s_0 \\ c_0 \\ 0 \end{bmatrix} \dot{o}_y \qquad \omega = \begin{bmatrix} c_0 s_y \\ s_0 s_y \\ c_y \end{bmatrix} \dot{o}_{z_1}$$

Bibliography

[1] H. Abramson. *The Dynamic Behavior of Liquids in Moving Containers: With Applications to Space Vehicle Technology*, volume 106. ser. NASA SP. Government Press, 1967.

[2] B. Akan, A. Ameri, B. Çürüklü, and L. Asplund. Intuitive industrial robot programming through incremental multimodal language and augmented reality. In *2011 IEEE International Conference on Robotics and Automation*, pages 3934–3939, May 2011.

[3] 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*, pages 3489–3494, Oct 2010.

[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*, 56(9):3432–3442, Sept 2009.

[5] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, pages 1371–1394. Springer, Secaucus, NJ, USA, 2008.

[6] F. Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[7] P. Bolzern, R. Scattolini, and N. Schiavoni. *Fondamenti di Controlli Automatici*. McGraw - Hill, 2008.

[8] 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*, pages 3592–3597, Nov 2013.

[9] IBM Corporation. 1 new orchard road, armonk, new york 10504-1722, united states. Homepage: https://www.ibm.com.

[10] Y. Ding, F. Li, Yu Ji, and J. Yu. Dynamic fluid surface acquisition using a camera array. In *2011 International Conference on Computer Vision*, pages 2478–2485, Nov 2011.

[11] 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)*, pages 2075–2080, Oct 2016.

[12] IBM ILOG CPLEX documentation. https://www.ibm.com/us-en /marketplace/ibm-ilog-cplex.

[13] Reflexxes Motion Libraries documentation. http://www.reflexxes.ws /software/typeiirml/v1.2.6/docs/index.html.

[14] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[15] F. Flacco and A. De Luca. Real-time computation of distance to dynamic obstacles with multiple depth sensors. *IEEE Robotics and Automation Letters*, 2(1):56–63, Jan 2017.

[16] Reflexxes GmbH. Sandknoll 7, d-24805 hamdorf, germany. Homepage: http://www.reflexxes.com.

[17] 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*, pages 5049–5055, Sept 2014.

[18] R. Ibrahim. *Liquid Sloshing Dynamics: Theory and Applications.* Cambridge University Press, 2005.

[19] M. Kim, J. Jang, K. Jeong, D. Kim, and J. Paik. Liquid-level estimation using region-based segmentation for automatic beverage refilling service. In *2015 International Symposium on Consumer Electronics (ISCE)*, pages 1–2, June 2015.

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

[21] 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*, pages 367–372, June 2008.

[22] ABB Ltd. Affolternstrasse 44, ch-8050 zurich, switzerland. Homepage: http://new.abb.com/.

[23] 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)*, pages 4286–4291, May 2017.

[24] N. J. W. Morris and K. N. Kutulakos. Dynamic refraction stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1518–1531, Aug 2011.

[25] P. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Journal of Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[26] J. Wolff and M. Buss. Invariance control design for costrained nonlinear systems. *IFAC Proceedings Volumes*, 38(1):37 – 42, 2005. 16th IFAC World Congress.

[27] K. Yano and K. Terashima. Robust liquid container transfer control for complete sloshing suppression. *IEEE Transactions on Control Systems Technology*, 9(3):483–493, May 2001.

[28] K. Yano and K. Terashima. Sloshing suppression control of liquid transfer systems considering a 3-d transfer path. *IEEE/ASME Transactions on Mechatronics*, 10(1):8–16, Feb 2005.

[29] J. Ye, Y. Ji, F. Li, and J. Yu. Angular domain reconstruction of dynamic 3d fluid surfaces. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 310–317, June 2012.

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