Contributions to visual servoing for legged and linked multicomponent robots

By

Zelmar Echegoyen Ferreira

http://www.ehu.es/ccwintco

Dissertation presented to the Department of Computer Science and Artificial Intelligence in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

PhD Advisor:

Prof. Alicia d'Anjou d'Anjou

Prof. Manuel Graña Romay

At

The University of the Basque Country
Donostia - San Sebastian

2009

Agradecimientos

Contributions to visual servoing for legged and linked multicomponent robots by

Zelmar Echegoyen Ferreira

Submitted to the Department of Computer Science and Artificial Intelligence on October 2, 2009,
in partial fulfillment of the requirements for the degree of Doctor of Philosophy

abstract

Partiendo de una revisión exhaustiva del estado del arte en control mediante realimentación visual (visual servoing) se han realizado dos aportaciones en el trabajo de esta tesis sobre dos tipos de robots distintos. Sobre un robot con patas (el robot Aibo de Sony) se ha realizado el desarrollo formal y riguroso del cálculo de la cinemática inversa basado en la minimización del error visual, teniendo en cuenta todos los grados de libertad del robot. Se ha realizado una experimentación exhaustiva para determinar de forma empírica el rango de aplicación del modelo y su sensibilidad. La segunda aportación se refiere a los sistemas robóticos multicomponente ligados, que consiste en grupos de robots portando un elemento pasivo unidimensional (manguera o cable). Se ha realizado un modelado previo de la dinámica del sistema basado en splines dinámicos que permite la simulación de esquemas de control heurísticos y de la cinemática inversa deducida de forma analítica. Dicho modelo está parametrizado por características de la manguera, como el peso o la rigidez. Se ha realizado la implementación de algoritmos de control heurísticos de realimentación visual centralizada sobre un grupo de robots SR1 portando un cable rígido.

Contents

List of Figures

List of Tables

Chapter 1

Introducción

Hoy en día los sistemas robóticos se enfrentan con el problema de trabajar en entornos muy poco estructurados, como astilleros o en la construcción. En estos entornos, las tareas suelen ser poco repetitivas, las condiciones de trabajo son difíciles de ser modeladas, y el tamaño del espacio es grande. Una tarea común es la de desplazar alugnos tipos de mangueras flexibles. Pueden ser mangueras para transporte de agua o aceite, lineas de alimentación eléctrica, u otras. En este trabajo nos interesamos en el diseño de una arquitectura de control para un sistema multi-robots que debe tratar con este problema. Una colección de robots cooperativos que sostienen una manguera deben ser capaces de desplazarla hacia una configuración determinada. En este trabajo hemos identificado los siguientes sub-problemas: modelado de un objeto flexible unidimensional, sensorización distribuida entre los robots para obtenter información del entorno y/o de la configuración del sistema, incluyendo los robots y la manguera, cinemática inversa del sistema, diseño estructural estable, control altamente adaptativo por medio de mecanismos cognitivos de alto nivel. En este trabajo nos concentramos en el modelado de una manguera y en la generación de estrategias de control para una colección de robots autónomos que la sujetan.

Chapter 2

Review in Visual servoing

2.1 Introduction

Visual Servoing is known as the task of positioning one or more robots in order to get poses of their final effectors that modify the environment in which their are working, using as feedback, in the control closed loop, the estimated positioning error from the visual information extracted of the environment by one or more video-cameras. For the robotics manipulators the visual servoing definition refers to the control of the pose of the final effector relative to a target or set of image features, while for the mobil robotic it refers to the robot pose relative to some landmarks in the environment. The pose of the final effector is defined as the position and orientation of the last element of its chain of articulations. The systems that use a visual control in a closed loop do not need to know exactly the structure of the environment and the position of the robot articulations, because they can compensate the deviations trough the visual feedback. However, the visual feedback needs a high bandwidth and a high frequency in image processing.

The first works known in the Visual Servoing domain are from W. Wichman [67] in 1967 and Y. Shirai and H. Inoue [62] in 1973. By the way, from this last work the "visual feedback" term started to be used for the systems that used the visual information in a control closed loop for robotics manipulators. At the end of the 70s some works about the use of visual control were developed in the SRI International (originally known as Stanford Research Institute), within the first works stand out [55, 56], which describes the use of visual loops in screwing on a screw and picking parts over a moving

conveyor belt. The Visual Servoing term appeared for first time in the publication of J. Hill and W. T. Park [26] in 1979, in which the term was used for differentiating the real time visual control from the system that until this time alternated cyclically between image tacking and robot movements. In 1981, Sanderson and Weiss [60] defined a classification of the Visual Servoing systems according to the space in which the error signal is defined, appearing for the first time the differentiation between Position Based Visual Servoing (PBVS) and Image Based Visual Servoing (IBVS), at the same time it defines a classification between systems that use internal control loop for the articulations (positioning sensors, known as encoder), and those that in spite of using encoders directly use the visual information in the stabilization of the articulations positioning.

In the 80s the development of this area was slow due to the difficulty in getting hardware capable of doing the real time image processing at high speed that allow the servo-motors control. Before the personal computers appeared at the beginning of the 90s it was necessary to use specialized hardware for pixels processing was very expensive. In this first years some works remarked as the developed by [21] in 1981, in which it is described a screwing on using as feedback the information provided by a stereoscopic vision system. In 1984 Weiss [66] proposed the use of an adaptive control to the dynamic control of robots based on image features, that tries to compensate the effect generated by visual information in the closed loop dynamic when used as feedback.

The technological advance at the beginning of the 90s allowed better results and an increase in scientific publications. An exhaustive revision of Visual Servoing with huge bibliography references was resumed by Corke [7] in 1993, which contains a description of the historical evolution and the main applications reported until this moment. In 1996, S. Hutchinson et al. [30], developed a tutorial on Visual Servoing which has been used as a reference since this moment, and as a beginners material for this area, resuming the diverse applications in which Visual Servoing had been used.

On the main works resumed in [7, 30] remarks [26], where a binary image processing in estimating of positions and distances is described, based on distances between knowledge features, showing that the bi-dimensional and tri-dimensional visual guided movement may be done in tracking and picking of moving parts. Similar works were developed after this in [47, 31] for the tracking of a swinging grasp, estimating the next position of the grasp in every instant, using a predictor. In [11] a video digital processing system for

determining the position of the target in image window is described, using the extracted information in a closed loop positioning control.

From the 90s the study of the relationship between robots systems and vision systems has acquired huge importance for research. The main advantage of vision systems is that they allow to get an integral description of the environment, with huge information, in a non intrusive way, and that this huge amount of information can be interpreted by the human in a natural way, so they are specially useful in low structured environments in where the environment features are constantly varying.

In robotics applied to low structured environments, as the industrial ones, the environment adapts itself to a set of sensors in order to allow them to extract the desired information. In order to get the correct running of the sensorization system it is necessary that the structure of the system keeps invariant. In the vision systems, however, is the system who has the task of extracting the relevant information from the environment trough the analysis of the images sequences taken by one ore more video-cameras. The vision systems are specially useful in the integration of robotic systems for less structured environments.

Generally, the industrial applications of positioning that use visual information do they work by an open loop system, known as Look-then-move, in which the visual information is analyzed in order to get an environment description and finally act in consequence; this model has huge disadvantages, as for instance a high sensitivity to the perturbations and the calibration of the system. It is necessary to use a visual feedback that allows to improve the system performance and reduce the sensitivity it has with perturbations and calibrations errors.

There is a lot of knowledge areas that participate in the definition of visual servoing systems, remarking the real time image analysis, the robots kinematic and dynamic study, control theory, real time computing, visual recognition, tracking and tri-dimensional recovery.

2.2 Image features

In a control system design that uses visual sensorization, the first step is to determine what relevant information is desired to be extracted from images. The information may be very simple as borders, lines, circles, or more complex as curves, surfaces, specific patterns or global properties of the im-

age. The visual information analysis consumes a lot of hardware resources in function of the images resolution, some technics initially use a low resolution that go increasing into windows that allow to analise the image areas where some features are expected to be founded.

An image feature is defined as any structural information that can be extracted from the image. Every feature corresponds to the projection of a real physic feature over the camera plane. From the concept of an image feature an *image feature parameter* is defined as any real quantitatively measurable value and obtained from one or more image features. Some examples of image features parameters are: the coordinates in the camera reference system of corners or points over edges, the distance between two points or the orientation of the straight line that pass over them, another possibility may be the centroid of a points cloud or features of more complex forms.

From a set of k image features parameters the *image features parameters vector* is defined as $s = [s_1, \ldots, s_k]^T$, with $s \in \mathcal{F}$, being $\mathcal{F} \subset \mathbb{R}^k$ the image features parameters space .

From the position of a point in the task space, the position of its projection over the image plane is defined taking into account the *camera system model*. The traditional camera model used is the perspective projection, known as pin-hole, in which all the beams that come from an object pass through a point known as projection center, after passing through the image plane. In figure 2.1 we can see the projection of a point over the image plane.

The camera reference system is placed in the projection center, being the x-axis aligned with the axial axis. The camera plane is parallel to the plane defined by x and y axis, and is placed at a distance λ over the x-axis, known as focal distance. The intersection between the x-axis and the image plane is known as the *main point* and defines the center of the image reference system, which has the u-axis parallel to the y-axis, and the v-axis parallel to the z-axis.

The projection of a point $P = x, y, z$ expressed in the camera reference system has the following coordinates (u, v) in the image reference system:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{\lambda}{X} \begin{pmatrix} Y \\ Z \end{pmatrix} \tag{2.1}$$

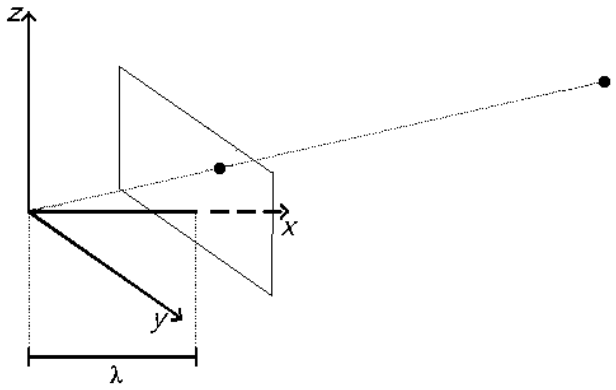


Figure 2.1: Camera reference system

2.3 Cameras configuration

There exist two kinds of basic configurations for the vision camera ([30]), eye-in-hand and fixed camera.

In the eye-in-hand model (figure 2.2), the camera is placed on the final effector of the robot, allowing the video sequences acquisition in the work space, being the target object the element over which the image caption process is centered around. In this system the relation between the camera pose and the final effector pose of the robot is known and constant, because the camera moves with the final effector of the robot. The main disadvantage of this model is the possibility of occlusions of the target object, because the robot may lie between the vision camera and the target object.

When using a fixed camera system (figure 2.3), on the other hand, the camera is fixed in a position of the task space, so it can capture the robot and the work space simultaneously. In contrast with the previous case, in the fixed camera model the captured images are independent of the robot movements and it exists a fixed relationship between the camera reference system and the referetnce system of the robot base.

For both models, eye-in-hand and fixed camera, it is necessary to calibrate the camera, its intrinsic parameters (internal geometry and optics features) and its extrinsic features (position and orientation).

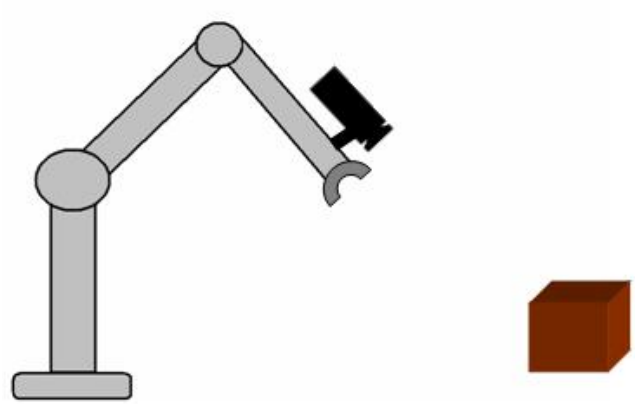


Figure 2.2: Eye-in-hand configuration

Frequently, the robotic tasks are defined respect to one or more reference systems. For example, the pose of an object obtained from the information of an image is used to be expressed in the camera reference system, while the pose of a target to be picked by a robot is usually expressed in the robot base reference system. Given two reference systems, it is possible to express the relation between them from the composition of the homogeneous transformations that bring from one reference system to the other, this transformations usually are rotations and translations. If we define ${}_bI_c$ as the transformation from the camera reference system to the robot base reference system, and we define ${}_eI_b$ as the transformation from the robot base reference system to the robot final effector system reference, then a point P in the camera reference system can be expressed in the robot final effector reference system as $({}_eI_b \cdot {}_bI_c)P$.

In the tri-dimensional reconstruction from the image features it is necessary to have additional information that allows to determine the position of the points in the 3D space, generally it is used to determine the point depth (its x coordinate in the camera reference system, in other words the distance from the point to the image plane) so then determine its coordinates in the task space by the following equation $(x, y, z)^T = (p, \frac{xu}{\lambda}, \frac{xv}{\lambda})$, being p the point depth and λ the camera focal distance. In order to detect the depth of a point using an only camera, it is necessary to obtain additional information as can be the use of redundancy in the image features, moreover they incorporate the disadvantage of the ocurrence of singularities and local minims. The depth of a point can be previously estimated in an adaptive way or taking

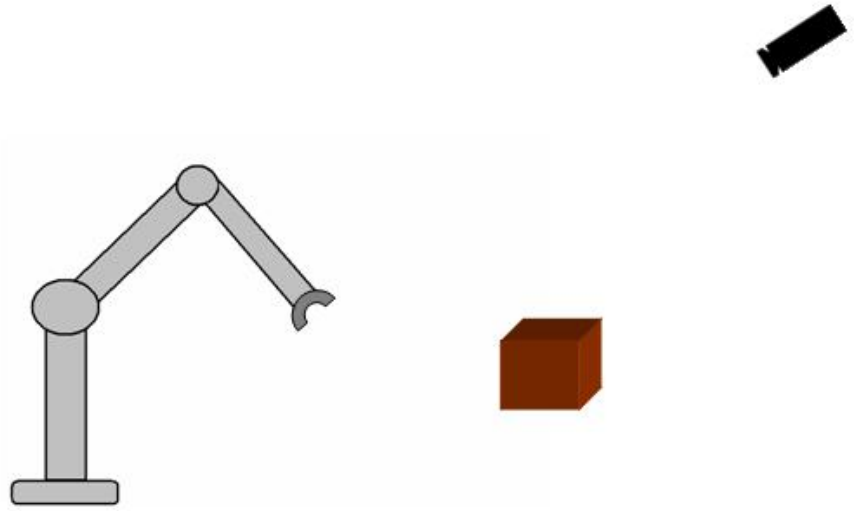


Figure 2.3: Fixed camera configuration

into account some kind of information about the object dimensions, asuming the target does not move significantly from one view to the other, sequences of views from one camera may be interpreted to derive depth information.

Some works use more than one vision camera to get a tri-dimensional reconstruction, using the epipolar geometry properties. It is assumed that the first visual servoing work in using stereo vision systems is [21] in 1981, in which the screw on of a screw is carry out. Other of the pioneer applications in using two video cameras for visual servoing is [35], that using a pair of parallel video cameras estimates the image jacobian. In [29] a trajectory generator is used to avoid obstacles using a stereo vision system. The stereo system add robustness and grater smoothness in the movements of the robot.

Some works use more than two cameras, which incorporate redundant information that give certain robustness against partial occlusions. The use of two cameras add the disadvantage of increasing the execution time. In [32] it is defined a multi-camera visual servoing system as a part of the approach based on the task function, showing experimental results using image based visual servoing ($2D$) and hybrid visual servoing ($2\frac{1}{2}$D) with two cameras that observe two different parts of and object. In [34] it is assumed that there is no a priory knowledge of the object model, so initially the final effector is positioned respect to the target performing learning movements around

it, then the obtained information about the relationship between the robot final effector and the camera is used to pile up the jacobian matrices (also known as interaction matrices) of both cameras. In [4] the study of stereo vision system is performed, getting empiric results that show a more lineal convergence in using 3D coordinates instead of using 2D coordinates, and as consequence the image features tend more to keep inside of the camera range in the 3D systems when the task movements are being developed.

2.4 Architectures and classifications

From the work developed by Sanderson and Weiss [60] in 1980, the visual servoing systems are classified in order to the space in which the error signal is defined. Other classification is based on the feedback at the joints level, between those who use internal control loops from positions sensors (encoders) and those who directly use the viual information in the stabilization of the joints positions.

2.4.1 Classification in the joints space

Sanderson and Weiss the visual servoing systems in look-then-move systems and *visual servo control*. Because the term visual servo control finally became a standard to generically describe any kind of visual control for a robotic system, it has been taken the convention of defining the systems look-then-move as *indirect visual servoing systems* and the systems that Sanderson and Weiss called visual servoing systems as *direct visual servoing*.

In the indirect visual servoing systems, *look-then-move*, (figure 2.4) it exists a controller at the level of the articulations of the robot, with a closed loop used as the feedback that gives the controller the information of the encoders on the articulations positions. There exist two main classifications for indirect systems; the first one receives the name *static look-then-move* and works in a sequential way capturing an image, processing it and then sending the movement command to the robot joints controller, until the robot reaches the desired position it does not start a new cycle. The robot, then, executes a movement assuming that the environment remains invariant, this scheme maintain isolated the joints control loop and the visual control loop. The second classification receives the name *dynamic look-then-move*, these systems do not wait until the desired positions of the joints have reached

Joints level feedback	Visual information
Dynamic look-then-move	Direct visual control

Table 2.1: Visual Servoing systems classification taking in account the type of joints state feedback

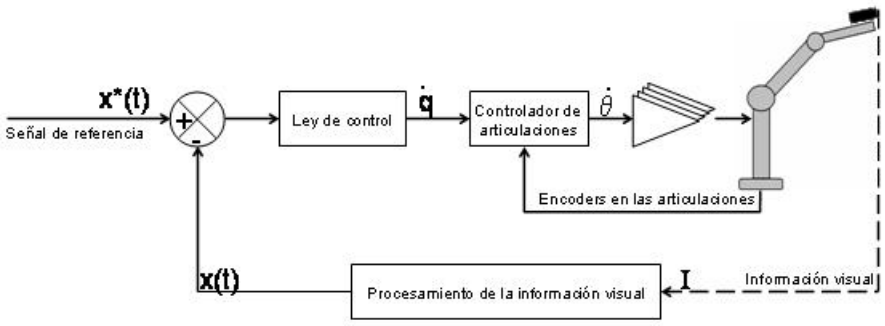


Figure 2.4: Indirect visual servoing systems (look-then-move)

to start processing a new image, in this case the joints control loop and the visual control loop are attached because the visual control loop allows to update the new positions of the joints while the robot still continue executing the previous movement; in these systems a greater frequency to the joint loop than to the visual loop is used.

In direct visual servoing systems (figure 2.5) there no exists a control loop at the joints level, the robot joints positions are estimated from the visual information at the frequency of the camera. In this case is the visual control loop who allows the control and stabilization of the servomotors of the robot. This approach was initially less used because of the required high frequency in visual information processing, and to the fact that most of the robots have incorporated an interface that allows incremental commands in Cartesian position and velocity, which simplifies the construction of a visual servoing system. Another reason for the not extended use of direct visual servoing systems was due to the excessive influence of perturbations in real time estimation of the joint states.

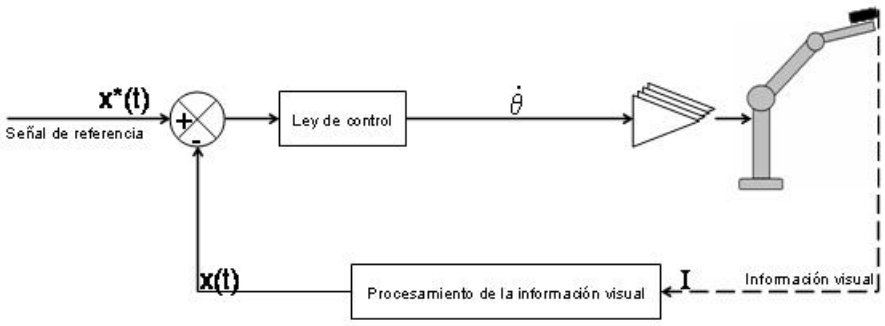


Figure 2.5: Direct Visual Servoing systems

Joints level feedback	Visual information
Static joints control	Dynamic joints control

Table 2.2: Classification of visual servoing systems according to the vision cameras configuration

2.4.2 Classification on the control space

The classification of visual servoing systems based on the control space distinguishes between position based control (3D control) and image based control (2D control), see table 2.3.

2.4.2.1 Position Based Visual Servoing

In Position Based Visual Servoing (PBVS), it is assumed that an a priori knowledge of the environment structure, the object model and the vision camera model are given, therefore the image features, known as s, are extracted and then used in estimating the pose of the target respect to the camera through a tri-dimensional reconstruction of the environment, because

Cartesian space (3D)	Image features (2D)
Position Based Visual Servoing	Image Based Visual Servoing

Table 2.3: Classification of visual servoing systems according to the control space

the image features are interpreted as a correspondence with the target geometric model. The difference between the desired pose and the current pose of the target respect to the camera constitutes the input signal of the control system. In these systems, a separation between the control task and the task of estimating the target pose respect to the camera exists; so, the error signal is defined in the task space, in other words in the 3D space.

In this approach, the estimation of the objects pose in the environment is developed using the images taken by the video cameras with the tridimensional knowledge of the target and its environment, and the intrinsic parameters of the camera.

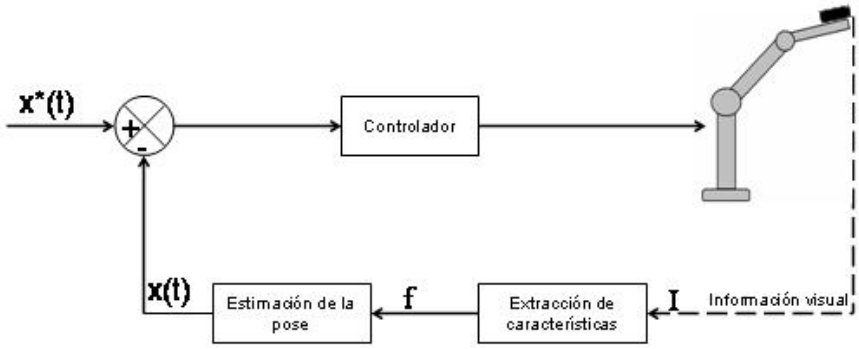


Figure 2.6: PBVS - Position Based Visual Servoing

2.4.2.2 Image Based Visual Servoing

In Image Based Visual Servoing (IBVS), the values for the control parameters are computed as a directly function of the image features. Unlike PBVS, the error signal is defined in the bi-dimensional system of the image and it is used directly as the input of the control system, for this reason the IBVS is known as 2D system. In this case exists a direct link between the image features and the robot joints, this relationship is encapsulated in the interaction matrix (also known as image jacobian matrix), because it defines a relationship between the variations in the pose (position and orientation) of the target respect to the camera, and the observed image features.

The error signal is defined as $E : T \rightarrow \mathbb{R}^l$,with $l \leq k$, being k the dimension of the space of image features parameters. It is necessary to link

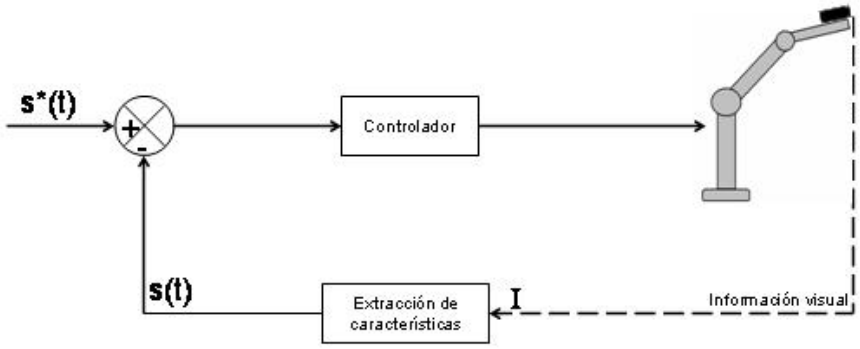


Figure 2.7: IBVS - Image Based Visual Servoing

the final effector pose variations with the variations in the image features parameters. This relationship is defined by the image jacobian.

Being r the coordinates of the final effector in the task space T, and then $\dot{r}$ its velocity, and f the image features parameters vector, being also $\dot{f}$ its velocities vector. We define the *Image jacobian*, also known as *Interaction matrix* and *Features sensitivity matrix*, J_v, as a lineal transformation from the tangential space at T to r at the tangential space at F in f. This way:

$$\dot{f} = J_v \dot{r} \tag{2.2}$$

with $J_v \in \mathbb{R}^{k \times m}$, and

$$J_v(r) = \left[\frac{\partial}{\partial r}\right] = \begin{bmatrix} \frac{\partial v_1(r)}{\partial r_1} & \cdots & \frac{\partial v_1(r)}{\partial r_m} \\ \vdots & & \vdots \\ \frac{\partial v_k(r)}{\partial r_1} & \cdots & \frac{\partial v_k(r)}{\partial r_m} \end{bmatrix}.$$

This matrix describes how the image features parameters varies respect to the variations in the robot pose, nevertheless in Visual Servoing the interest is in determining the final effector velocity of the robot, $\dot{r}$, needed to get a desired value in the velocities of the image features parameters vector, $\dot{f}$.

The most used aproach is to apply the task function in order to bring the image features parameters to the desired ones, based on a linear relation between variations in image features and variations in the robot pose. In general an a priori knowlege of the geometric features is needed, as for example contours, corners or edges [6, 10, 24], another approach is the use of visual marks [16]. An alternative, when there is no knowledge about geometric features is the image motion-based visual servoing [50, 63, 12, 13]

(also known as 2D+dt visual servoing) which uses dynamic features extracted from the information of the perceived motion in the image plane between two successive images and uses it as feedback int the control loop. The task of maintaining the target in a determined position in the image is equivalent to make its 2D projection keeps a null velocity.

2.4.2.3 Comparison between IBVS and PBVS

The basic difference between the approaches IBVS and PBVS is in the error signal space. PBVS is more sensitive to calibrations errors, because they influence on the tri-dimensional reconstruction of the environment, that generates errors in the movements execution; while IBVS systems link directly the image features to the robot joints. In PBVS, due to the fact that the task is defined based on the localization in the Cartesian space, the camera trajectory follow a straight line, in this way there not exist singularities in the task space, which is the error space in PBVS, and therefore practically do not exist local minims. In contrast, these systems depend too much on the camera calibration errors or the target geometric model; moreover, it is necessary to get additional information in order to reconstruct points in the Cartesian space.

The main advantage of IBVS systems is the fact that precision is independent of the calibration and the exact knowledge of the target model. In contrast, it is not possible to assure the global stability and usually appear singularities in the image features space, as for example in case of occlusions. Although IBVS is more robust in its dependence to calibration errors, it poses the disadvantage that some movements do not induce changes in image features, and so they produce singularities in the jacobian matrix. This way, the task of finding an inverse of the features matrix turns difficult if it is not enough conditioned. The trajectories in the image plane are straight lines, although the robot trajectories in the task Cartesian space are low intuitive and produce not desired large movements.

In 1997 Malis et al. [5] published a new approach that makes use of the advantages and avoids the disadvantages of PBVS and IBVS, which is in a medium point between both classic systems. From IBVS it incorporates the advantage of not needing a geometrical model of the target, and from the PBVS the possibility of assuring the convergence or the control law in the work space. This new approach uses 3D information and 2D information, so it is denominated as Visual Servoing $2\frac{1}{2}$ or *Hybrid Visual Servoing*.

2.5 System Control

In this section we analize the control system, the task of obtaining the camera velocity $\dot{r}$ int the task Cartesian space as a function of the error of the state variable x. The positioning task ends when $e(q, t*) = 0$, being $t*$ the instant in which the state takes the value $x*$. In order to get the desired positioning of the robot it is usually used a regulator which has the objective of obtaining the velocities , $\dot{r} \in \mathbb{R}^6$, that the final effector of the robot has to exert and which are sent to the joints control subsystem of the robot. The positioning task, so, boil down to the control of the robot final effector trajectory in the Cartesian space.

2.5.1 Positioning kinematic error

In [54, 59] the robotic problem is defined as a regulation to zero of the positioning error in a finite time. The *Kinematic error function,* $e : T \rightarrow \mathbb{R}^m$, is defined as:

$$e(q, t) = C(x(q, t) - x^*) \tag{2.3}$$

being:

- q the vector of the robot joints positions,
- $x(q, t)$ the vector of the obtained configuration from visual information, the pose of the final effector for PBVS and the image features for IBVS.
- x^* the desired configuration.
- C the combination matrix of dimension $n \times m$, being n the number of degrees of freedom of the robot and m the dimension of the state vector.

We also define J_e as the *Task jacobian*:

$$J_e = \frac{\partial e}{\partial q} \tag{2.4}$$

The jacobian task is usually represented as the composition of two jacobian matrices:

$$J_e = L_e.J, \tag{2.5}$$

being:

$J = \frac{\partial r}{\partial q}$, the *Robot jacobian*, the matrix that relates the velocity of the final effector in the Cartesian space with the joints velocity of the robot.

$L_e = \frac{\partial e}{\partial r}$, the *Task jacobian*, the matrix that relates the kinematic error with the robot Cartesian velocity.

It is said that the task has the admissibility property if exists an only trajectory of q for which the error function is zero in the time limit ($e(q, t^*) = 0$) and at the same time J_e is regular over this trajectory. For the IBVS systems, it is required the achieving of the visibility condition, in other words that exists an enough number of visual features inside the vision range of the camera, while for the PBVS systems and hybrid systems, it is also required the conditions that the task has the visibility property and the estimation of the target pose respect to the camera must be done.

2.5.2 Final effector trajectory in the task space

The aim of the Visual Serving system is the positioning of the final effector of the robot respect to a target, so it is assumed that the vector state x is differentiable as a function of the final effector pose r. The velocity of the state vector can be expressed as a function of the velocity of the camera position respect to the target:

$$\dot{x} = \frac{\partial x}{\partial r}\frac{\partial r}{\partial t} + \frac{\partial x}{\partial t} \tag{2.6}$$

Defining the *Interaction matrix* $L_x = \frac{\partial x}{\partial r}$, as the jacobian matrix that relates the state velocity with the final element of the robot in the Cartesian space v_r, equation 2.6 can be rewritten of the following way:

$$\dot{x} = L_x.v_r + \frac{\partial x}{\partial t} \tag{2.7}$$

At the same time, we can use the interaction matrix in the expression of the task jacobian, L_E, that measures the sensitivity of the task function respect to the Cartesian velocity of the final effector of the robot. If we suppose that the combination matrix C does not contain explicitly at r in its definition, the task jacobian can be rewritten as:

$$L_e = CL_x. \tag{2.8}$$

Going back to equation 2.3, if we want an exponential decreasing of the kinematic error in time, using the time constant λ, the error trajectory can be rewritten by the following lineal differential equation:

$$\dot{e} = -\lambda e \tag{2.9}$$

The time derivate of the error kinematic function, $\dot{e}$, can we written as function of the velocity of the robot joints, $\dot{q}$, and if we also use the decomposition $\frac{\partial e}{\partial q} = \frac{\partial e}{\partial r}\frac{\partial r}{\partial q}$, we get the following expression:

$$\dot{e} = \frac{\partial e}{\partial r}\frac{\partial r}{\partial q}\dot{q} + \frac{\partial e}{\partial t} \tag{2.10}$$

The expression $\frac{\partial e}{\partial r}$ is the task jacobian, L_e (see equation 2.4), while the expression $\frac{\partial r}{\partial q}\dot{q}$ can be substituted by the velocity of the final element of the robot v_r. Using this notations we get the following simplified expression of equation 2.10:

$$\dot{e} = L_e.v_r + \frac{\partial e}{\partial t} \tag{2.11}$$

getting $\dot{r}$ definition from equation 2.11 we define the following expression:

$$v_r = L_e^+ \left(\dot{e} - \frac{\partial e}{\partial t} \right) \tag{2.12}$$

Substituting $\dot{e}$ by expression in equation 2.9 we get the following expression for the final effector robot velocity as a function of the kinematic error:

$$v_r = -L_e^+ \dot{\left(\lambda e + \frac{\partial e}{\partial t} \right)} \tag{2.13}$$

However L_e and ∂e are estimated, so the Cartesian velocity of the final effector of the robot is defined as:

$$v_r = -\hat{L_e}^+ \left(\lambda e + \frac{\hat{\partial e}}{\partial t} \right) \tag{2.14}$$

2.5.3 Stability and convergence

Substituting v_c by expression 2.14 in equation 2.11, we get the following expression for the velocity of the kinematic error function:

$$\dot{e} = -L_e \hat{L_e}^+ \left(\lambda e + \frac{\hat{\partial e}}{\partial t} \right) + \frac{\partial e}{\partial t} \tag{2.15}$$

From equation 2.14 we get the following sufficient condition that assures the reduction of the norm of the kinematic error function $\|e\|$:

$$L_e \hat{L_e}^+ > 0 \tag{2.16}$$

Assuming that the combination matrix C does not contain explicitly r in its definition, we can rewrite the previous condition as:

$$C L_x (C \hat{L_x})^+ > 0 \tag{2.17}$$

The state vector x represents the information used as feedback in the control loop, depending on the architecture type of the system it can represents the image features parameters vector on IBVS or the pose of the robot final effector on PBVS.

When the type of control is based on image, $2D$, the notation s for vector x is used as standard.The task jacobian is then expressed as:

$$\hat{L_e} = C \hat{L_s}, \tag{2.18}$$

being $L_s = \frac{\partial s}{\partial r}$ the jacobian of the image features parameters vector as function of the Cartesian pose of the final effector of the robot. Generally the interaction matrix is chosen equal to the identity, so the stability condition reduces to :

$$L_s \hat{L_s}^+ > 0 \tag{2.19}$$

When the control is based on position, $3D$, the interaction matrix is used to be expressed as the following matrices composition:

$$L_x = L_{xs} L_s \tag{2.20}$$

Matrix L_s is the interaction matrix $2D$, while L_{xs} represents the $3D$ reconstruction from the image features.

The stability condition is defined in the following way:

$$L_x \hat{L_x}^+ > 0 \tag{2.21}$$

2.5.4 Trajectory Generation

In PBVS it is easy to generate the reference trajectory due to the fact that the target information is given in 3D, the task space. In contrast, in IBVS the visual information is given in 2D and therefore the robot may not reach some intermediate positions in the task space through the trajectory defined by the visual based control between the current image features and the desired ones, in consecuence the desired features should not be far from the current image features because the robot may follow an undesired trajectory.

Despite this disadvantages, IBVS is less sensitive to vision calibration errors than PBVS (imprecise positioning) and this is the general case in real situations. When distance between initial and final positions in the image is big we want to optimize the trajectory in the work space from the image error function, the objective is to keep the features inside the image. The task of finding an optimum trajectory that allows the target object to tracks a desired path in the image is known as *Trajectory generation.*

The first work on Trajectory generation, from our knowledge, was developed by J. Feddema in 1989 [18] mantaining a continous trajectory by matching the velocity accelerations and jerk of the features but not the features position. In [58] an on-line trajectory generation method is presented, where a weighting matrix is chosen taking into account a maximum velocity for each joint and applying the weighted , this matrix has the effect of suppressing the reference velocity. A new approach considering the constraint of maintaining the target object in the camera field of view is proposed in [36], where a potential field that induces repulsive forces is defined to create a potential barrier around the camera field of view in order to assure that all the features are always observable. In [44] a trajectory of a gripper which moves through a straight line is generated using an uncalibrated stereo rig; the trajectory is achieved by descomposing the projective coordinates of initial and desired points into a special rigid displacement and a triangular matrix. A similar work is developed in [43, 45] where intermediate configurations, between initial points an desired ones, are constructed in the projective space using a conjugate transformation and projective invariants and then reprojected onto the image planes to get an image-based trajectory. In [37]

a modified potential field method is used to determine discrete trayectories that are then interpolated by b-splines in order to obtain continous curves that allow an improvent of the dynamic behaviour of the system.

2.5.5 Integration of visual servoing and force control

The combination of visual servoing and force control has been growing since the increasing of processing power and low cost vision systems have allowed the aplicability of this approach. This combination is highly complementary due to the fact that force sensors apport 3D information about the contact between the robot and the target object, while vision sensors give information about the 3D environment. The devolopment of sensor integration as the simultaneous use of vision and force control has been based on the area of force controlled manipulators as an extension.

In force controlled manipulators [61] are two main approaches, hybrid position/force control [52] and impendance control [27]. In hybrid position/force control, the control space is separated in position and force controlled directions, defining a feedback loop for force and a feedback loop for position, which are independent and parallel. The force subspace is known as wrench space while the position subspace is known as twist space. Hybrids approachs allow faster dynamics but requiring model-based compensation. In impedance control a relationship between motion and force is established, by translating a task into a desired impedance.

The main problem for integrating vision and force information is that they do not provide a common data representation and in consecuence are used in differente stages. An impedance based visual/force approach was defined in [41], where a level based view of the use of vision/force controller is established defining the independent tasks of: traded, hybrid and shared control.

In [28] an adaptative hybrid visual/force controller is used to do visual servoing while the robot makes contact forces on a surface, which has an online estimator for the parameters of the unknown constraint surface that only needs the knowledge about the manipulator kinematics. A control algorithm in the impedance control approach is defined in [40], performing a peg in a hole insertion using a 7 axis robot manipulator where the reference trayectory to the impedance controller is generated on line by an IBVS loop. An hybrid control approach is presented in [48], where an extraction of a book on a shelf is developed. In [61] a framework based on the Task frame formalism (TFF)

for distinguishing between differente types of shared control is presented. A method for tracking trajectories, known as movement flow-based visual servoing, is presented in [46]; this method uses the Kalman filter as the criteria for asigning weights to variables for each sensor system.

2.5.6 Invariant Visual Servoing

In Visual servoing the task can be clasiffied by the knowledge or not knowledge of the target object model. If a model of the target object is known a model-based aproach is used. In contrast, if it is not a model of the target a model-free approach is used. In the model-free approach an initial learning step is performed in order to get reference images of the target that allow to estimate the object model. Changes in the intrinsic parameters of the camera affect the servoing performance, so a new learning step should be neccesary.

The Invariant visual servoing aproach was introduced by Ezio Mails [15], and then an extension of this work in[33], trying to extend the teaching-by-showing technique when different cameras are used for teaching and servoing, without an explicit calibration between them. This new approach works in a projective space invariant to camera intrinsic parameters and to the knowledge of the tri-dimensional model of the target object, and allows the use of differente cameras for the learning step and servoing task. In [19] a redefinition of invariant visual servoing approach is developed in order to allow the use of zooming during a positioning task, using weighted image features to avoid the discontinuities produced by the appeareance and disappearance of image features during the control task. In [20] a study on how to select some of the parameters of the weight function is done; a stability analysis of invariant visual servoing with weighted features is also proposed. In [17] the use of the invariant visual servoing approach is proposed for the reconstruction of underwater objects, simulating an active underwater stereovision system mounted on a 6 DOF manipulator arm effector over an underwater vehicle.

2.5.7 Partitioned Visual Servoing

In Image based Visual Servoing the image Jacobian defines a mapping between image space velocities and velocities of the robot joints, it results in a shortcoming with respect to posistion based approaches since it does not need a tri-dimensional reconstruction. But on the other hand, the pure use of the image Jacobian lead to control problems if it is poor conditioned and

due to the ocurrence of singularities. Moreover, because this approach works on the image plane the trayectories of the robot in the cartesian space are quite contorted and may drive the robot towards singularities in the image Jacobian.

In [8] a partitioned approach is introduced trying to avoid this problem by decoupling the motions in the axis of the camera reference system perpendiculars to the image plane, usually known as z-axis. The traditional IBVS takes the form $\dot{r}_{xyz} = J^{+}_{xyz}\dot{s}$ while this partitioned takes the form $\dot{r}_{xy} = J^{+}_{xy}\{\dot{s} - \dot{J}_z \dot{r}_z\}$ for the xy movements, being s the feature point coordinate error and, J_{xyz}, J_{xy}, J_z the respective images jacobian for the three cartesian axes of the camera reference system, the camera plane axes (x, y) and the z-axis perpendicular to the image plane. The movements in the z-axis are described based on two new features for translational and rotational movements.

2.5.8 Neural Networks

In IBVS the use of the image Jacobian is needed in order to define a relation between the image features space and the robot's movements space, this matrix is not easily constructed even in the knowledged of the robot's kinematics and the camera model. Moreover, the use of the inverse of the image Jacobian does not provide large features movements in the image due to the large motions errors derived of the implicit linearization, in the worst case the image jacobian might be singular. Some approaches have tried to avoid the inconvenientes derived from the image jacobian using a tragetory generation. The works done in [38, 39] proposed the use of nueral networks in the design of a learning control system for visual servoing, where the a priori knowledge of the robot kinematics or the pose of the target respect to the robot was not assumed. Then, in [23] a self-organizing visual servoing system was proposed, where the learning of a the feature Jacobian is done despite the geometric dimensions. The use of a fuzzy controller with a supervised capability was proposed in [**?**], where the elements of the image Jacobian do not take into account the relative distance between the target and the robot, using only the image features information.

Chapter 3

Contribution to legged Visual Servoing

This chapter presents a contribution to the visual tracking of objects using all the degrees of freedom of a legged robot. We approach this issue in a principled way applying ideas of visual servoing. Nowadays visual tracking solutions for this kind of robots inspired in the visual servoing approach only move the effectors linked directly to the camera or use a learning kinematics matrix. In this work we take into account all the effectors which can affect the resulting image. We construct from a general description of the robot the matrix that describes its kinematics. Visual servoing is performed computing the seudoinverse of this matrix.

3.1 Direct Kinematic

We build the robot kinematics as a transformation from the ground supporting plane to the camera coordinate system, composing the diverse transformations that correspond to the limbs, and then going up trough the upper robot's degrees of freedom that take part of the articulations chain from the body center reference system to the camera reference system.

As illustrated in figure 3.1 the legs of a legged robot are the elements which support de robot's body and therefore define a relationship between the body and the support plane, so we need to be able to determine the 3D coordinates of the leg's points in contact with the ground at any time. We call support points, l, to the leg's points that are in contact with the ground

and therefore support the robot body. The support points are shown by a red circle around them.

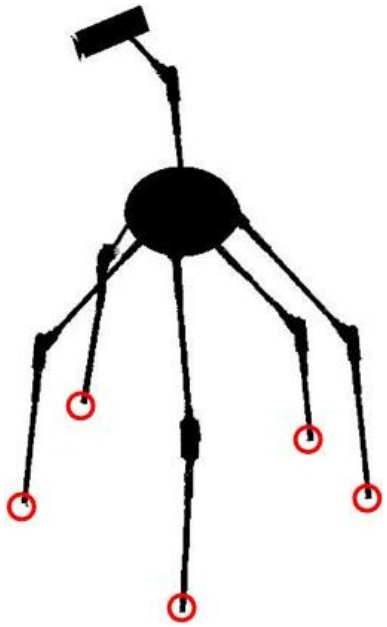


Figure 3.1: Points of contact with the supporting surface

Coordinate reference systems In order to obtain the features expressed in a fixed reference system it is necessary to define the relevant reference systems of the robot. we use the notation I_j refering to the generic reference system j, and we use the notation ${}_iI_j$ when refering to the transformation from generic reference system j to generic reference system i.

First of all, we need a fixed base reference system over the ground, I_g; then we need a reference system of the body, I_b, beyound the legs articulations: finally we define the camera reference system, I_c. Having the three basic reference systems we have to define the transformation matrices between them. See that every transformation uses a subset of the robot joints: transformations between I_g and I_b depend on legs articulation joints of the support points, θ_l, while transformations between I_b and I_c depend on the joints that take part in the articulations chain from body center to the camera, θ_u. These reference systems are illustrated in figure 3.2.

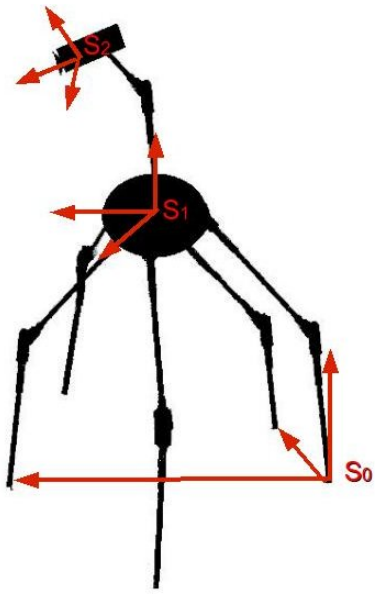


Figure 3.2: Reference Systems of the robot

3.1.1 Legs degrees of freedom

Each leg has a chain of articulations, as shown in figure 3.4. The legs degrees of freedom are used indirectly towards the support points, so we introduce this concept.

3.1.1.1 Support points

The support points are the points of the robot limbs that determine the plane where it is standing on. These points must be determined in the coordinate system of the robot body. From the point of view of the centre of the body the supporting plane apparently varies when the robot servos are affected when the physical reality is that the plane remains fixed and the robot changes its pose.

Each leg has a unique support point, and, according to the restriction that the robot must be standing, at least three of the legs must have their supporting points in contact with the ground; therefore there may be several possible support planes if we take into account all feasible combinations support points that may give us a standing configuration of the robot. In order to determine which combination of supporting points coincides with the

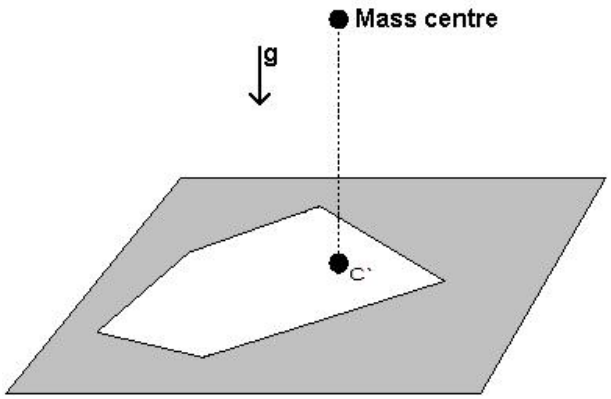


Figure 3.3: Condition for supporting points on the ground plane

physical supporting surface we obtain the plane equation for every possible combination.

For a given combination of support points we have the plane equation $\pi : ax + by + cz + d = 0$, then we evaluate to which hemisphace belong the points that have not been taken into account to build the plane equation; if for any one of these q points we find $(q_x, q_y, q_z)^T : aq_x + bq_y + cq_z + d < 0$, it means that this point is under the plane and therefore this plane is not the ground surface; in contrast, if we find $(q_x, q_y, q_z)^T : aq_x + bq_y + cq_z + d > 0$, it means that this point is above the plane and therefore may be the ground surface, but this point is not a support point. We deffine a tolerance, tol, and for every point for which $|aq_x + bq_y + cq_z + d| < tol$ we accept this points as a supporting point.

Besides, in order for the robot to be standing in a stable pose, the projection of the body center of mass, in the direction of the gravity, must lie inside of the convex hull (closed polygonal chain) defined by the supporting points in contact with the ground surface. This condition is illustrated in figure 3.3. Therefore, the search for the ground support points is guided by testing this condition on each n-tuplet of leg supporting points, being n the number of supporing points. For those tuplets that meet the condition, we fit the plane equation and test that the remaining supporting points remain above this plane.

In order to obtain the ground supporting plane, it is necessary to determine which is the extrem coordinates for each leg in the reference space centered on the robot body center. It is necessary to determine the positions

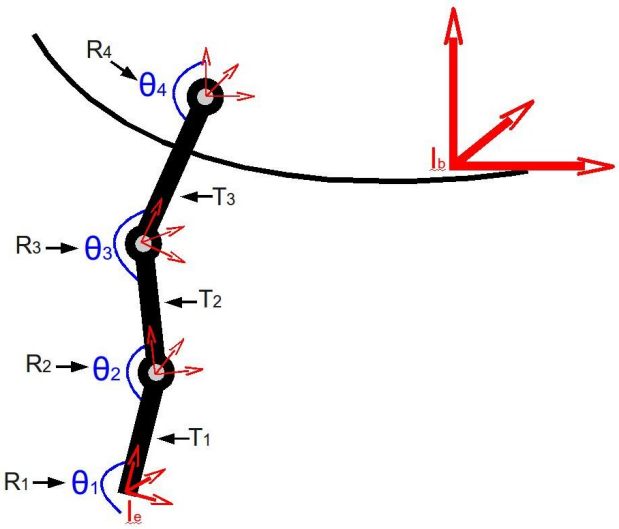


Figure 3.4: Geometry of the leg articulations

in function of the articulations states, given by their torsion angles.

We find the extreme position of a leg using the coordinate system transformations in the articulation chain from the supporting points to the body center. This transformations are descripted in terms of rotation and translation matrices in homogenean coordinates. For each joint we define a rotational matrix determined by its torsion angles, and for every pair of linked joints we define a traslation matrix.

In homogeneous coordinates the transformation from the reference system of the extreme of a leg to the reference system of the body center can be described as the product of the elemental transformation matrices:

$$\begin{pmatrix} q_x \\ q_y \\ q_z \\ 1 \end{pmatrix} = (R_n.T_{n-1}.R_{n-1}...T_1.R_1). \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

3.1.1.2 Transformation between ground system I_g and body system I_b

In order to define the coordinates changes between the ground system I_g and the body system I_b, we define the I_g vectors in the system I_b, and then do

the translation between them. So, we separate the transformation in rotation and translation, although it exists an scale component.

The entire transformation uses three basic supporting points positions: $\alpha = (\alpha_x, \alpha_y, \alpha_z, 1)^T$, $\beta = (\beta_x, \beta_y, \beta_z, 1)^T$, $\gamma = (\gamma_x, \gamma_y, \gamma_z, 1)^T$. We use the position point α as the origin of S_g, and the vectors $\overrightarrow{\alpha\beta}$ y $\overrightarrow{\alpha\gamma}$ as the two first vectors, and we built the third vector as the vectorial product of the two first.

So, we built the rotational matrix, R, from the three vectors of S_g:

$$R_0 = \begin{pmatrix} \downarrow & \downarrow & \downarrow & 0 \\ \beta - \alpha & \gamma - \alpha & \langle \gamma - \alpha, \beta - \alpha \rangle & 0 \\ \downarrow & \downarrow & \downarrow & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

and we define the translational matrix from the origin of S_b to the origin of S_g,

$$T_0 = \begin{pmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.2}$$

So, composing the two transformations we finally obtain the matrix change from S_g to S_b,

$$_bI_g = T_0 R_0. \tag{3.3}$$

We define the vector of basic support points in the body reference system, S_b, as:

$$\Gamma = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \tag{3.4}$$

The three basic support points may be every combination of three support points.

3.1.2 Upper degrees of freedom

After have obtained the relationship between the center of the body robot and the leg's extrems, we have to determine the relationship from the body reference system to the camera reference system. We construct the Transformation applying the rotational and translational matrices that bring the

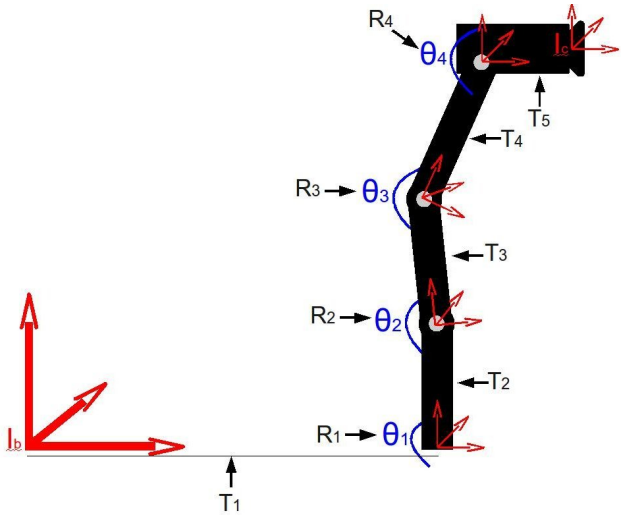


Figure 3.5: Articulations supporting the camera

camera reference system to the robot body reference system We call upper degrees of freedom to the articulations that take part in the chain of the robots articulations from the base reference system to the camera reference system..

Transformation between body system I_b and Camera system I_c The transformation between the body system and the camera system can be done through the compositions of more elemental transformations. We compose the elemental transformations that go from the body system I_b, to the camera system I_c. The result of the matricial composition is the transformation from system I_c to I_b: ${}_bI_c$.

We call upper articulations, θ_u, to the articulation joints that take parte in the joints chain from the body reference system to the camera reference system.

3.1.3 Image features

The stated goal is to bring the image features to the desired ones, therefore we define the vector of parameters of image features as s and the vector of the desired parameters of image features as s_*. But these parameteres must be expressed in terms of the robot degrees of freedom, in order to use

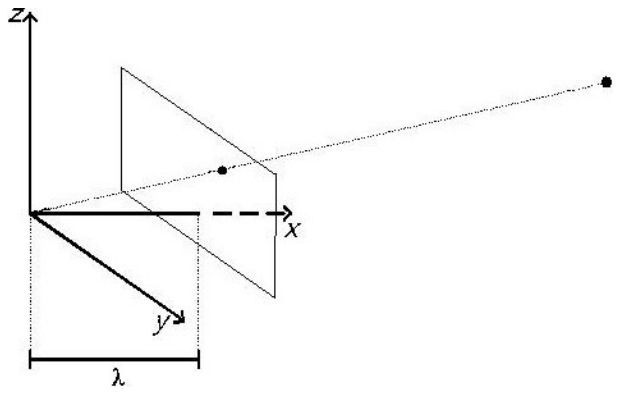


Figure 3.6: Projection of a point over the image plane

the Jacobian to determine the feature sensitivity respect to each articulation positions changes.

The camera reference system define the image features positions according to the vision camera of the robot. In figure 3.6 is shown the projection of a point over the vision camera plane.

The parameters of an image point feature, $s_i = (u_i, v_i)^T$, are determined by its position in the camera system $(x_i, y_i, z_i)^T$, according to the following relation:

$$s_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \frac{\lambda}{x_i} \begin{pmatrix} y_i \\ z_i \end{pmatrix}. \tag{3.5}$$

The features are expressed in terms of their positions in the system I_c, but as we had supposed the object was fixed respect to I_g, we could obtain the features expressed in function of the camera robot articulations and the support points positions, using the features points positions in I_g and the transformation between I_g and I_c.

$$\begin{pmatrix} u \\ v \end{pmatrix} = function({}_cI_g({}^gs)) \tag{3.6}$$

3.1.3.1 Construction of the robot's jacobian matrices

In order to construct the Image jacobian matrix that relates the variations of the diverse degrees of freedom of the robot with the variations in the image plane, we need to obtain the relations between the components of the robot body and the image plane. We start obtaining the jacobian matrix

that defines the dependence of the image features on the camera reference system, J_{sc}, then we obtain the jacobian matrices for the dependence of the camera reference system on the basic support points positions, $J_{c\Gamma}$, and on the upper articulations, $J_{c\theta_u}$.Then we define the jacobian matrix that relates the basic support points positions with the positions of all the support points, $J_{\Gamma l}$. Finally, we obtain the jacobian matrix for the dependence of the support points positions with the articulations of their legs $J_{l\theta_l}$..

Dependence of image features on the target object Deriving the equation 3.5 we get the following relation for an image point s_i:

$$\partial s_i = \begin{pmatrix} \frac{-\lambda . y_i}{x_i^2} & \frac{\lambda}{x_i} & 0 & 0 \\ \frac{-\lambda . z_i}{x_i^2} & 0 & \frac{\lambda}{x_i} & 0 \end{pmatrix} \cdot \partial \left({}^c s_i \right) \tag{3.7}$$

We call this matrix $J_{s_i c}$. In order to contemplate all the features parameters in the features vector, we pile up the jacobian matrices $J_{s_i c}$ for all the features points parameters, obtaining the following block diagonal matrix:

$$J_{sc} = \begin{pmatrix} J_{s_1 c} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & J_{s_k c} \end{pmatrix} \tag{3.8}$$

We call J_{sc} the Jacobian matrix of equation 3.8. Then J_{sc} defines a lineal transformation from variations of points positions in the camera reference system I_c into variations of the image features vector s.

$$\Delta s \simeq J_{sc} \cdot \Delta \left({}^c s \right) \tag{3.9}$$

Dependence of target object on the robot articulations In the construction of the jacobian matrix that defines the dependence of the tridimensional points of the target object in the camera reference system with the robot articulations we need to express the relation between the basic support points and the support points articulations, so we introduce this relation as the first step.

Dependence of support points on the basic support points The following matrix relates variations in the vector of the extreme points positions of legs $\delta \mathbf{l}$ whit variations in the vector of the three basic support points

positions, $\partial\mathbf{\Gamma}$.

$$\begin{pmatrix} \delta l_1 \\ \vdots \\ \delta l_n \end{pmatrix} = \begin{pmatrix} M_{1\alpha} & M_{1\beta} & M_{1\gamma} \\ \vdots & \vdots & \vdots \\ M_{n\alpha} & M_{n\beta} & M_{n\gamma} \end{pmatrix} \begin{pmatrix} \delta\alpha \\ \delta\beta \\ \delta\gamma \end{pmatrix} \tag{3.10}$$

We call $J_{l\Gamma} \in \mathbb{R}^{4n\times 12}$ the Jacobian matrix of equation 3.10, and we define this matrix as a composition of matrixes of size 4×4, being:

$M_{ij} = Identity$, if the leg i has the support point j,

$M_{ij} = 0$, if the leg i has a support point different from j or if the leg i has not a support point.

This relation is resumed as:

$$\delta\mathbf{l} = J_{l\Gamma}.\partial\mathbf{\Gamma} \tag{3.11}$$

Dependence on support points articulations First, we need to obtain a linear transformation between the variations in all the support points positions and the variatios in their degrees of freedom. This dependence is expressed in the body reference system.

We model the changes in the support points coordinates according to the degrees of freedom variations, using the Jacobians as follow:

$$\Delta l_i \simeq J_{l_i} \cdot \Delta\theta_i \tag{3.12}$$

being $J_{l_i}$the jacobian matrix of the the support point of leg i as function of its joints positions and $\theta_i = \theta_{i1}, \theta_{i2}, \ldots, \theta_{im_i}$ the value of the degrees of freedom in leg i.

Composing the leg's extrems Jacobians at every moment, we obtain the following jacobian matrix:

$$\begin{pmatrix} \delta l_1 \\ \vdots \\ \delta l_n \end{pmatrix} = \begin{pmatrix} M_1 & 0 & 0 & 0 \\ 0 & M_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & M_n \end{pmatrix} \begin{pmatrix} \partial\theta_{l_1} \\ \partial\theta_{l_2} \\ \vdots \\ \partial\theta_{l_n} \end{pmatrix} \tag{3.13}$$

The jacobian matrix receives the name $J_{l\theta_l}$, where M_i is:

- Jl_i, if the leg i has a support point.

- *Zero* (the matrix with all the elements equal 0) if this leg i has not a support point on the plane.

The size of each M_i matrix is $4 \times m_j$, being m_j the numberr of joints of leg j.

.

The dependence of the support points positions on their joints degrees of freedom is summarized as follows:

$$\Delta l \simeq J_{l\theta_l}.\Delta\theta_l \tag{3.14}$$

We use the matrix $J_{l\theta_l}$ in order to obtain the relation between the target object in the camera reference system and the support points articulations by defining the Jacobian matrix $J_{\Gamma\theta_l}$ as the product $J_{l\Gamma}^{+} \cdot J_{l\theta_l}$, we get the following dependence relation between the variations on basic support points positions and all the support points articulations:

$$\Delta\Gamma \simeq J_{\Gamma\theta_l} \cdot \Delta\theta_l \tag{3.15}$$

Dependence of target object on basic support points and upper articulations We saw that the features positions in the camera system could be expressed as a function of the support points and the upper robot articulations, equation 3.6. By deriving this equation we get the Jacobian matrix that relates the variations in the feature positions in the camera reference system I_c with the variations in the basic support points positions and the upper articulations:

$$J_{cp} = \frac{\delta({}_cI_b \circ_b I_g)}{\delta\mathbf{p}} ({}^gs) \tag{3.16}$$

being p the following vector of upper joints and support points positions:

$$\mathbf{p} = \begin{pmatrix} \theta_u \\ \Gamma \end{pmatrix} \tag{3.17}$$

Using the chain rule, we rewrite equation 3.16:

$$J_{cp} = \left[\frac{\delta({}_cI_b)}{\delta\mathbf{p}} \circ ({}_bI_g) + ({}_cI_b) \circ \frac{\delta({}_bI_g)}{\delta\mathbf{p}} \right] ({}^gs) \tag{3.18}$$

As ${}_cI_b$ is a function of θ_u (camera articulations) and ${}_bI_g$ is a function with parameter Γ (support points positions), we descompose eq. 3.18 in:

$$J_{c\theta_c} = \frac{\delta({}_cI_b)}{\delta\theta_u} \circ ({}_bI_g)({}^gs) \tag{3.19}$$

$$J_{c\Gamma} = ({}_cI_b) \circ \frac{\delta({}_bI_g)}{\delta\Gamma}({}^gs) \tag{3.20}$$

In order to obtain a single matrix to describe this relation we define the following block diagonal matrix:

$$J_{cp} = \begin{pmatrix} J_{c\theta_u} & 0 \\ 0 & J_{c\Gamma} \end{pmatrix} \tag{3.21}$$

The dependence between the variations in the ball position, and the variations in the camera degrees of freedom and in the support points positions can be summarized by:

$$\Delta\,({}^cs) \simeq J_{cp}\mathbf{p} \tag{3.22}$$

Dependece on the robot articulations If we compose matrices $J_{\Gamma\theta_l}$ and $J_{c\Gamma}$ we obtain the following relation between variations of the support points leg articulations positions and variations of features points in the camera reference system:

$$J_{c\theta_l} = J_{c\Gamma}J_{\Gamma\theta_l} \tag{3.23}$$

If we pile up matrices $J_{c\theta_u}$ and $J_{c\theta_l}$, we obtain the following block diagonal matrix:

$$J_{c\theta} = \begin{pmatrix} J_{c\theta_u} & 0 \\ 0 & J_{c\theta_l} \end{pmatrix} \tag{3.24}$$

This matrix encapsulates the relation between the position of the target poin in the camara reference system with the robot's degrees of freedom:

$$\triangle c = J_{c\theta}.\triangle\theta \tag{3.25}$$

3.1.3.2 Image Jacobian matrix

Finally we define the full Jacobian matrix that models the dependence of the image features on the diverse degrees of freedom of the robot, by composing the jacobian matrices 3.9, 3.19 and 3.23 obtained in the previous section.

$$\Delta s = J_{sc} \circ [J_{c\theta_c} \Delta\theta_c + J_{c\theta_l} \Delta\theta_l]. \tag{3.26}$$

If we compose J_{sc} with $J_{c\theta}$ we get the following jacobian matrix between image features and robot's articulacions.

$$J_{s\theta} = J_{sc} J_{c\theta} \tag{3.27}$$

The equation 3.26 is rewrited in the following way:

$$\Delta s = J_{s\theta} \Delta\theta. \tag{3.28}$$

In algorithm 3.1 we present a top-down review of the steeps done in the construction of the full jacobian matrix.

Algorithm 3.1 Full Jacocbian Matrix

The full jacobian matrix is defined as $J_{s\theta} = J_{sc}J_{c\theta}$, and determines the image features dependences on the robot articulations. This jacobian matrix is composed by J_{sc}, that encapsulate the dependences of image features on the camera reference system, and $J_{c\theta}$, that encapsulates the dependence of the camara reference system on the robot articulations.

The jacobian $J_{sc} = \begin{pmatrix} J_{s_ic} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & J_{s_ic} \end{pmatrix}$ is determined by piling up the jacobians of each feature point.

The jacobian $J_{c\theta} = \begin{pmatrix} J_{c\theta_u} & 0 \\ 0 & J_{c\theta_l} \end{pmatrix}$is composed by matrices $J_{c\theta_u}$, that defines a point in the camera reference system as a dependence on the upper articulations, and $J_{c\theta_l}$,. that defines a point in the camera reference system as a dependence on the support points articulations.

The jacobian $J_{c\theta_l} = J_{c\Gamma}J_{\Gamma\theta_l}$ is composed by matrces $J_{c\Gamma}$, that defines the camera reference system dependences on the basic support points articulations, and $J_{\Gamma\theta_l}$, that defines the basic support points positions dependences on the support legs articulations

Finally, the jacobian $J_{\Gamma\theta_l} = J_{\Gamma l}J_{l\theta_l}$is composed by matrices $J_{\Gamma l}$, that defines the basic support points positions dependences on all the support points positions, and $J_{l\theta_l}$, that defines the support points positions dependences on the support points articulations.

3.2 Inverse Kinematics

The goal of the stated visual servoing problem is to determine the instantaneous of each of the robot degrees of freedom that will be needed to bring the features to the desired ones.

In order to determine the velocity at each robot degree of freedom we should obtain the inverse of the $J_{s\theta}$ matrix in equation 3.28. However, this is not possible because the matrix is not invertible. As we have more degrees of freedom than image features, the problem is overconstrained, because there are not sufficient features to determine the movements in an only way.

The general solution is to use the seudoinverse of, $J^{+}_{s\theta}$, by minimum

squares.

$$\dot{\theta} = J_{s\theta}^{+}\dot{s} + (I - J_{s\theta}^{+}J_{s\theta})n \tag{3.29}$$

Being n an arbitrary vector of R^{15}.

In general, $(I - J_{s\theta}^{+}J_{s\theta})n \neq 0$, and all the vectors of the form $(I - J_{s\theta}^{+}J_{s\theta})n$ belong to the kernel of the transformation associated to $J_{s\theta}$.

This solution minimizes the norm

$$\left\| \dot{s} - (J_{s\theta})\dot{\theta} \right\| \tag{3.30}$$

As our objective is to center the ball in the image, we will not get into more details about the movements which minimize de error, so our solution will be

$$\dot{\theta} = J_{s\theta}^{+}\dot{f} \tag{3.31}$$

But this solution does not take into account the restriction of keeping the distances constant. So, we need to determine how these variations in the supporting points positions affect the distances between them.

Differencing d we get the Jacobian matrix that relates these changes.

$$J_{dl} = \begin{pmatrix} \frac{\delta d_1}{\delta l_1} & \cdots & \frac{\delta d_1}{\delta l_n} \\ \vdots & \vdots & \vdots \\ \frac{\delta d_n}{\delta l_1} & \cdots & \frac{\delta d_n}{l_n} \end{pmatrix} \tag{3.32}$$

This Jacobian matrix receives the name J_{dl}.

In order to include the variations in the camera degrees of freedom, we define the jacobian matrix J_{dp} from J_{dl}:

$$J_{dp} = \begin{pmatrix} Zero_{nxm} & 0 \\ 0 & J_{dl} \end{pmatrix} \tag{3.33}$$

Finally this dependence is resumed in the following equation:

$$\Delta d \simeq J_{dp} \cdot \Delta p \tag{3.34}$$

Now we use the Jacobian matrix J_{ps} with J_{dp} and $J_{p\theta}$ to get the variations on the robot articulations that make the image features converge to the desired features, keeping constants the distances between the supporting points.

To mantain the distances constants, the vectorΔp must belong to the kernel of the transformation associated to Δd belongs to the kernel of J_{dp}.

$$\Delta p = [(I - J_{dp}^{+} J_{dp})\{(I - J_{dp}^{+} J_{dp}) J_{ps}^{+}\}^{+}]\Delta s \tag{3.35}$$

As our final objective is to get the articulations variations we add the seudoinverse of $J_{p\theta}$, also we add a velocity constant to control the advance velocity of the robot

$$\Delta\theta = J_{p\theta}^{+}(I - J_{dp}^{+} J_{dp})\{(I - J_{dp}^{+} J_{dp}) J_{ps}^{+}\}^{+}\{k_i \Delta s\}. \tag{3.36}$$

This equation allows us to determine the variations on the robot degrees of freedom to get the desired configuration of the image. However, this equation is unrestricted and may drive the robot into unstable configurations, that is, to articulation configurations out of the region of stable poses in configuration space. Stable poses are characterized by the tuplet of ground support points which fulfill the condition illustrated in figure 3.3. When this does not happen, or the projection point is too close to the triangle boundary, we restrict the visual servoing to the head degrees of freedom, using the transformation ${}_{g}I_{c}$ instead of ${}_{b}I_{c}$, to construct a reduced Jacobian J_c that relates the image features to the camera degrees of freedom. Its seudoinverse gives the control for the head degrees of freedom. This reduced approach has already been applied in [51, 53].

3.3 Experimentation with an Aibo ERS-7 Robot

In this section we show the results obtained from applying the ideas of visual servoing developed in chapter 3 to the visual tracking of a ball using all the degrees of freedom of a Sony's Aibo ERS-7 robot. Nowadays visual tracking solutions for this kind of robots inspired in the visual servoing approach only move the head effectors or use a learning kinematics matrix. In this work we take into account all the effectors which can affect in the resulting image. We construct from the description of the robot the matrix that describes the kinematics of the robot.

Figure 3.7 illustrates the main feedback loop in image-based visual servoing with the Aibo.

In the RoboCup robot soccer matches some visual servoing approaches [51, 53] have been implemented in the Aibo robot to track the ball. However,

these approaches are limited to the movement of the head effectors in order to keep the ball into the video image. The space in which the ball can be followed is restricted by the robot body pose.

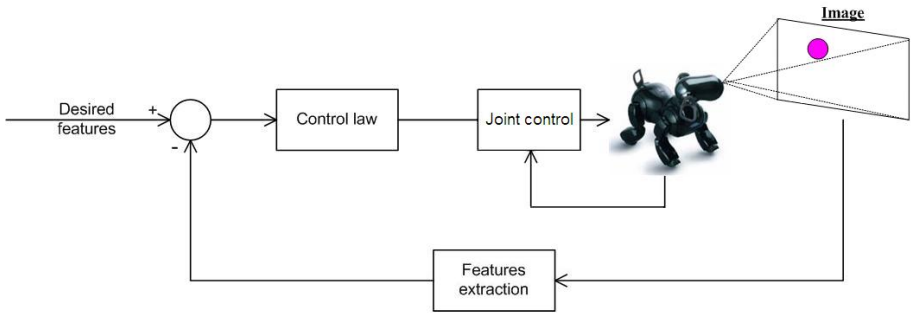


Figure 3.7: Visual servoing feedbak loop

In this chapter we address the problem of maintaining the playing ball in the center of the robot camera image. The only visual feature considered is the center of the ball region in the image identified by the color detection routines implemented in the robot. We have profited from the Carnegle Mellon University's SDK [68] and the SONY's SDK [9]. The image error is the distance in image space between the image center and the centroid of the blob corresponding to the ball. The image features considered are very naive when compared with recent works in other domains, however they are the current state of the art in the Aibo environment.

In order to follow the construction of the image jacobian matrix defined in section 3.1.3.2, we need to define the following jacobian matrices:

J_{sc}- dendences of the image features on the target object points positions in the camera reference system

$J_{c\theta_u}$- dependence of the target object points positions, in the camera reference system, on the upper degrees of freedom.

$J_{c\Gamma}$ - dependence of the positions of the target object, in the camera reference system, on the basic support points.

$J_{l\theta_l}$ - dependence of the support points positions on the support points leg´s articulations.

We start defining the image features vector from the ball parameters, then we define the jacobian matrices introducing the direct linear kinematics of the robot in order ot obtain the image jacobian, then we apply the inverse

kinematics and we end up with some discussion of the physical experimentation, the observed robot behavior and future work lines.

3.3.1 Image features

The stated goal is to bring the ball in the image centre, so the target features are the image centre coordinates and the observed features from the real world are the coordinates of the ball region centre and its diameter.

The camera reference system is fixed to the robot head, and define the ball position according to the vision camera of the robot. In figure 3.8 the projection of the ball over the robot camera plane is presented.

The image features parameters vector, s, is determined by the ball position in the camera system, according to the following relation:

$$s = \begin{pmatrix} u \\ v \end{pmatrix} = f({}^c ball) = \frac{\lambda}{{}^c ball_x} \begin{pmatrix} {}^c ball_y \\ {}^c ball_z \end{pmatrix}. \tag{3.37}$$

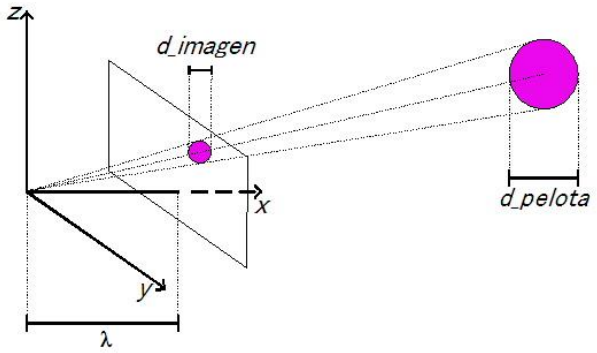


Figure 3.8: Proyección de la pelota en la cámara

The ball position in the camera reference system is determined by estimating the distance from the camera to the ball, in order to get the distance we have to take into account the real diameter of the ball and its diameter in the image plane, the following equation shows this relation:

$$distance = \lambda \frac{1}{2} \frac{diam_r}{tan(\frac{1}{2} diam_i)} \tag{3.38}$$

Being $diam_r$ the real diameter of the ball and $diam_i$ the proyected diameter in the image plane.

Then, we define the ball position in the camera reference system using the estimated distance:

$$^{c}ball = distance \begin{pmatrix} cos(v).cos(u) \\ sin(u) \\ sin(v) \end{pmatrix} \tag{3.39}$$

The features are expressed in terms of the ball position in the system I_c, but as we had supposed the ball was fixed respect to I_g, we could obtain the features expressed in function of the head robot articulations and the support points positions, using the ball position in I_g and the transformation between I_g and I_c.

$$\begin{pmatrix} u \\ v \end{pmatrix} = f(_{c}I_{g}(^{g}ball)) \tag{3.40}$$

3.3.2 Direct Kinematic

We build the Aibo kinematics as a transformation from the ground supporting plane to the head coordinate system, composing the diverse transformations that correspond to the limbs and head degree of freedom. We start from the supporting points and go up to the head.

As illustrated in figure 3.9 the robot's feet and the knees are the possible robot support points therefore we need to be able to determine their 3D coordinates at any time.

3.3.2.1 Legs degrees of freedom

Each leg has three articulations, as shown in figure 3.10. The legs degrees of freedom are used indirectly towards the support points. The support points are the points of the robot limbs that determine the plane where it is standing on. We use the robot body center of mass because the Aibo possesses an inertial sensor than gives us feedback on the motion of this point.

Each leg has a unique support point that can be the foot as well as the knee, and, according to the restriction that the robot must be standing, at least three of the legs must have their supporting points in contact with the ground; therefore there are 32 possible support planes if we take into account all feasible combinations support points that may give us a standing configuration of the robot.

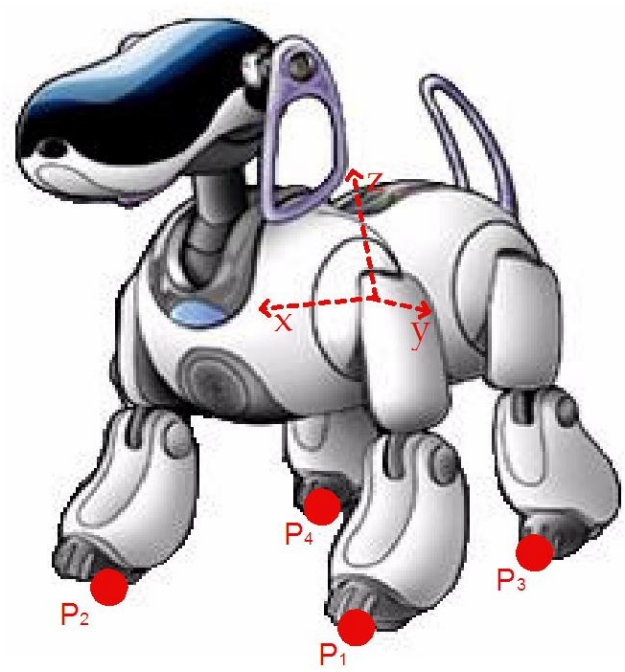


Figure 3.9: Points of contact with the supporting surface

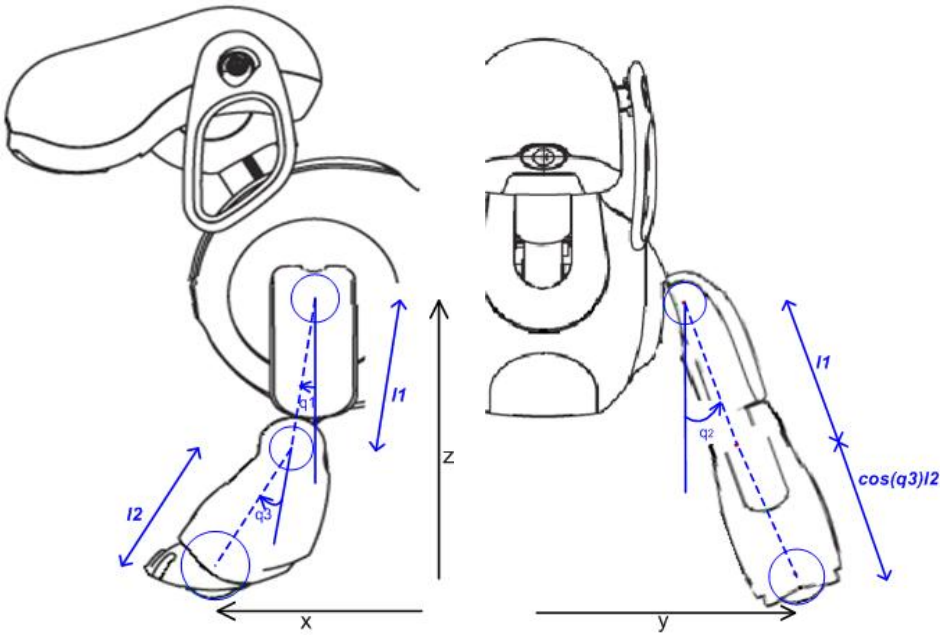


Figure 3.10: Geometry of the leg articulations

Feet and knees positions In order to obtain the ground supporting plane, is necessary to determine which are the supporting point coordinates for each leg in the reference space centered on the robot body center of mass. It is necessary to determine the positions of the feet and knees in function of the articulation states, given by their torsion angles.

We nd the foot centre position, for the front left leg, using the following coordinate system transformations.

T_1: Translation along the z-axis of length l_1.

R_1: Clockwise rotation about y-axis by angle q_1.

R_2: Counterclockwise rotation about x-axis by angle q_2.

R_1: Clockwise rotation about y-axis by angle q_3.

T_2: Translation along the z-aixs of lenght l_2.

T_l: Translation along the x-axis of length $\frac{1}{2}l$, being l the robot body length.

T_a: Translation along the y-axis of length $\frac{1}{2}a$, being a the robot body width.

In homogeneous coordinates the transformation from the body center to

the foot coordinate system can be described as the product of transformation matrices:

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix} = (R_1.R_2.T_1.R_3.T_2). \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

This equation is valid for the robot front left leg; however, due to the symmetry of leg coordinate systems, only a few signs must be changed to get the positions of the other three leg's feet. In order to find the coordinates of each knee in the body reference system we only have to do the three first and the two last transformations used to determine the foot coordinates.

3.3.2.2 Head degrees of freedom

The Aibo ERS-7 has three degrees of freedom in the head. That introduces ambiguity in the control trajectories needed to track the ball trajectory.

Figure 3.11 shows the two tilt degrees of freedom of the Aibo, denoted θ_{big} y θ_{small}. The first head tilt degree of freedom corresponds to the neck base pivoting along part of the dog chest, while the second one allows the head to move vertically using as the rotation centre the joint between the neck and the head. The third degree of freedom, called θ_{pan}, allows a perpendicular rotation to the previous one, moving the head from side to side.

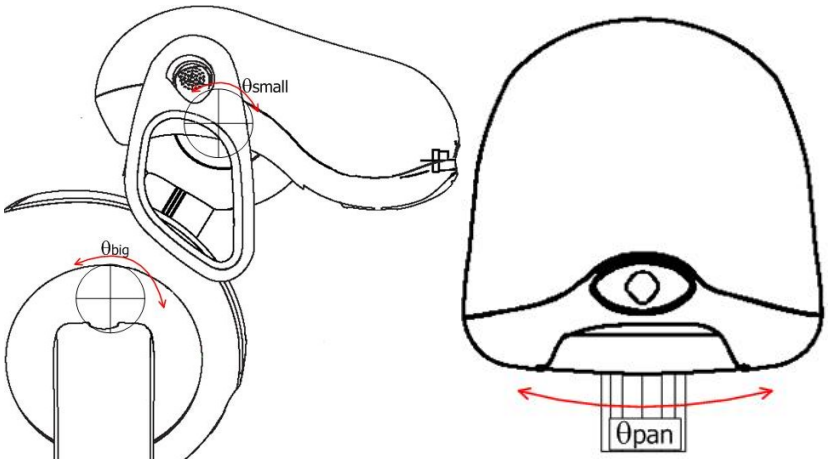


Figure 3.11: Aibo head degrees of freedom

3.3.2.3 Coordinate reference systems

We define the reference systems as explained in section In order to obtain the ball position expressed in the I_g ground system it is necessary to obtain the transformation matrices between the different systems.

These reference systems are illustrated in figure3.12, and are define acording to the reference system previously defined in section 3.1.

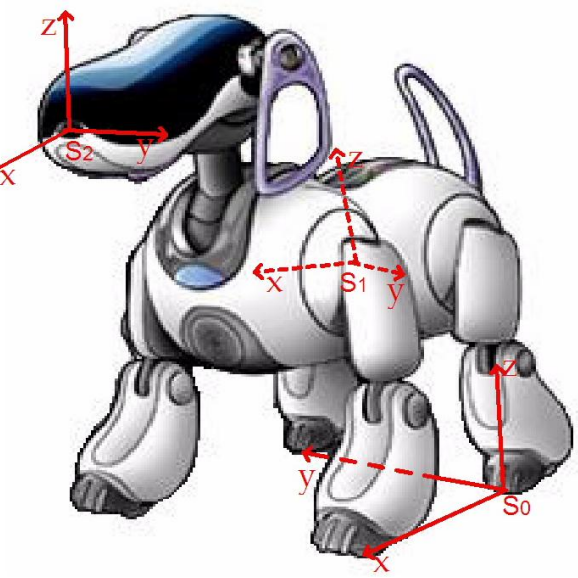


Figure 3.12: Aibo reference systems

Transformation betweenI_g and I_b In order to define the coordinates changes between the ground system I_g and the body system I_b, we define the I_g vectors in the system I_b, and then do the translation between them, as explained in section 3.1.1.2.

The entire transformation uses the supporting points positions: α, β and γ. Becase we assume that the four legs of the Aibo are on the ground surface, we define the left back leg as α, the left front as β and the right back as γ, assuming that the front right leg will be adjusted in the inverse kinematic to remain in the ground plane.

The rotational matrix, R, and the translational matrix, T, are obtained as explained in section 3.1.1.2, and then composing the two transformations

we nally obtain the matrix change from I_g to I_b,

$$ {}_bI_g = T \cdot R. \tag{3.41} $$

Transformation betweenI_b and I_c The transformation between these systems can be done through the compositions of more elemental transformations. We compose the transformations that go from the body system I_b, to the head system I_c.

The first transformation is a translation from the camera base to the top of the neck, T_1:

$$ T_1 = \begin{pmatrix} 1 & 0 & 0 & camera_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & camera_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.42} $$

Next, we have to rotate the head, taking into account the *nod* and *pan* articulations, we call this rotational matrix R_1:

$$ R_1 = \begin{pmatrix} cos(pan)cos(nod) & -sen(pan) & -cos(pan)sen(nod) & 0 \\ sen(pan)cos(nod) & cos(pan) & -sen(pan)sen(nod) & 0 \\ sen(nod) & 0 & cos(nod) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.43} $$

Then, the translation T_2, between the neck base and the neck conection with the head, take the system to the I_b origin:

$$ T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & neck_l \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.44} $$

Then, we have to use the tilt articulation defining the rotational matrix, R_2:

$$ R_2 = \begin{pmatrix} cos(tilt) & 0 & -sen(tilt) & 0 \\ 0 & 1 & 0 & 0 \\ sen(tilt) & 0 & cos(tilt) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.45} $$

Finally, we bring the system to the center of the body by translation T_3:

$$T_3 = \begin{pmatrix} 1 & 0 & 0 & neck_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & neck_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.46}$$

The result of the matrices composition ${}_bI_c$ is the transformation between the systems I_b and I_c:

$${}_bI_c = T_3 R_2 T_2 R_1 T_1 \tag{3.47}$$

3.3.2.4 Feature Jacobian matrix

Now we will construct the Jacobian matrix that relates the variations of the diverse degrees of freedom of the Aibo with the variations in the image plane.

Dependence of image features on the target object The features must be expressed in terms of the robot degrees of freedom, in order to use the Jacobian to determine the feature sensitivity respect to each articulation positions changes.

Deriving the equation 3.37 we get the following relation:

$$\begin{pmatrix} \delta u \\ \delta v \end{pmatrix} = \begin{pmatrix} \frac{-\lambda . y_p}{x_p^2} & \frac{\lambda}{x_p} & 0 & 0 \\ \frac{-\lambda . z_p}{x_p^2} & 0 & \frac{\lambda}{x_p} & 0 \end{pmatrix} \begin{pmatrix} \delta^c ball_x \\ \delta^c ball_y \\ \delta^c ball_z \\ 0 \end{pmatrix} \tag{3.48}$$

We call J_{sc} the Jacobian matrix of the equation 3.48. Then J_{sc} defines a lineal transformation from variations of the positions of the ball in S_c into variations of the image features.

$$\Delta s \simeq J_{sc} \cdot \Delta({}^c ball) \tag{3.49}$$

Dependence of support points on the basic support points The following matrix relates the variations in the basic support points, δl, with the variations in all the support points.

$$\begin{pmatrix} \delta l_1 \\ \delta l_2 \\ \delta l_3 \\ \delta l_4 \end{pmatrix} = \begin{pmatrix} M_{1i} & M_{1j} & M_{1k} \\ M_{2i} & M_{2j} & M_{2k} \\ M_{3i} & M_{3j} & M_{3k} \\ M_{4i} & M_{4j} & M_{4k} \end{pmatrix} = \delta \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \tag{3.50}$$

We call $J_{l\Gamma} \in \mathbb{R}^{15\times 12}$ the Jacobian matrix of equation 3.10, and we define this matrix as a composition of the following matrixes of size 3×3matrices:

$M_{ij} = Id$, if the leg i has the support point j,

$M_{ij} = 0$, if the leg i has a support point different from j,

So, if the leg i has a generic support point, then the row i has a identity matrix and three null matrix.

The dependence between the variations in the legs articulations with the supporting points positions and the head articulations variations can be resumed in the following equation:

$$\Delta l \simeq J_{l\Gamma} \cdot \Delta\Gamma \tag{3.51}$$

Dependence on support points articulations The next step is obtaining a linear transformation between the variations of the legs degrees of freedom and the ground support points coordinates in the body reference system.

First we observe that according to which part of the leg is in contact with the ground there are two possible jacobean matrices, one for the foot (J_{f_i}) an another for the knee (J_{k_i}). We model the changes in the foot and the knees coordinates according to the degrees of freedom variations, using the Jacobians as follows:

$$\Delta f_i \simeq J_{f_i} \cdot \Delta\theta_i \tag{3.52}$$

$$\Delta k_i \simeq J_{k_i} \cdot \Delta\theta_i \tag{3.53}$$

Being θ_{i1}, θ_{i2} y θ_{i3} the degrees of freedom of leg i.

Composing with the support points Jacobian at every moment, we obtain the following jacobian matrix:

$$\partial \begin{pmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{pmatrix} = \begin{pmatrix} M_1 & 0 & 0 & 0 \\ 0 & M_2 & 0 & 0 \\ 0 & 0 & M_3 & 0 \\ 0 & 0 & 0 & M_4 \end{pmatrix} .\partial \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{pmatrix} \tag{3.54}$$

The jacobian matrix receives the name $J_{l\theta_l}$, where M_i is:

Jf_i, if the support point for the leg i is the foot.

Jk_i, if the support point for the leg i is the knee.

$Zero$ (the matrix with all the elements equal 0) if this leg has not a support point on the plane.

The dependence of the support ground points on the limb's degrees of freedom is summarized as follows:

$$\Delta l \simeq J_{l\theta_l}.\Delta\theta_l \tag{3.55}$$

Dependence of target object on basic support popints and upper articulations According to the development done in section 3.1.3.1, we obtain the matrices that defines the relation between the ball position in the camera system and the support points and the head robot articulations, by derving equation 3.40. The Jacobian matrix that relates the variations in the image ball position with the variations in the support points positions is defined as:

$$J_{c\Gamma} = \frac{{}_cI_b.{}_bI_g({}^gball)}{\delta\Gamma} \tag{3.56}$$

The transformation matrix ${}_bI_g$ is defined by the coordinates of the support points in the base reference system as follows:

$${}_bI_g = \begin{pmatrix} [\beta-\alpha] & [\gamma-\alpha] & [(\gamma-\alpha)\wedge(\beta-\alpha)] \end{pmatrix} \tag{3.57}$$

Therefore the Jacobian matrix is composed by the following derivatives:

$$J_{c\Gamma} = ({}_cI_b)\begin{pmatrix} \frac{\partial_bI_g({}^gball)}{\partial\alpha} \\ \frac{\partial_bI_g({}^gball)}{\partial\beta} \\ \frac{{}_bI_g({}^gball)}{\partial\gamma} \end{pmatrix}^T \tag{3.58}$$

Being: $\frac{\partial_bI_g({}^gball)}{\partial\eta} = \begin{pmatrix} \frac{\partial_bI_g}{\partial\eta_x}({}^gball) \\ \frac{\partial_bI_g}{\partial\eta_y}({}^gball) \\ \frac{\partial bI_g}{\partial\eta_z}({}^gball) \end{pmatrix}$, for $\eta = \alpha, \beta, \gamma$.

The Jacobian matrix that relates the variations in the image ball position with the variations in the head degrees of freedom is defined as:

$$J_{c\theta_u} = \frac{\delta({}_cI_b)({}_bI_g)({}^g ball)}{\delta\theta_{head}} \tag{3.59}$$

Only he transformation matrix ${}_cI_b$ depends on θ_{head} and is defined as the composition of elemental matrices, so equation 3.59 can be rewrited as:

$$\frac{\delta({}_cI_b)}{\delta\theta_{head}} = \frac{\delta(T_3R_2T_2R_1T_1)}{\delta\theta_{head}} \tag{3.60}$$

In this compostion, matrices T_1, T_2 and T_3 do not depend on any of the head articulations, therefore only the rotational matrices R_1 and R_2 are derived in order to obtain derivative of the transformation matrix. Because R_1 depends on small and pan articulations and R_2 depends on tilt articulations, we can express the derivative of the transformation matrix from ground system to base system as follow:

$$\frac{\delta({}_cI_b)}{\delta\theta_{head}} = \frac{\delta({}_cI_b)}{\delta\begin{pmatrix}\theta_{pan}\\ \theta_{nod}\\ \theta_{tilt}\end{pmatrix}} = \begin{pmatrix} T_3R_2T_2\frac{\delta R1}{\delta\theta_{pan}}T_1 \\ T_3R_2T_2\frac{\delta R1}{\delta\theta_{nid}}T_1 \\ T_3\frac{\delta R2}{\delta\theta_{tilt}}T_2R_1T_1 \end{pmatrix}^T \tag{3.61}$$

3.3.3 Inverse Kinematics

In order to determine the velocity at each robot degree of freedom we should apply the pseudoinverse approach of equation 3.36. As we have more degrees of freedom than image features, the problem is overconstrained, because there are not sufficient features to determine the movements in an only way.

This equation allows us to determine the variations on the robot degrees of freedom to get the desired configuration of the image. However, this equation is unrestricted and may drive the robot into unstable configurations, that is, to articulation configurations out of the region of stable poses in configuration space. Stable poses are characterized by the existence of a triplet of ground support points which fulfill the condition illustrated in figure **??**. When this does not happen, or the projection point is too close to the triangle boundary, we restrict the visual servoing to the head degrees of freedom, using the transformation ${}_gI_c$ instead of ${}_bI_c$, to construct a reduced Jacobian J_h that relates the image features to the head degrees of freedom. Its seudoinverse

gives the control for the head degrees of freedom. This reduced approach has already been applied in [51, 53].

3.3.4 Results

In this section we show the results obtained from applying the aproach proposed in chapter 3 to the visual tracking of a ball using all the degrees of freedom of a Sony's Aibo ERS-7 robot. We examinate the behavior of the robot for different values of the velocities of its articulations.

The first part of the experimentation consists on defining a pose for the Aibo robot in a fixed reference system and then applying the proposed aproach for several positions of the ball distributed within the domain area of the video-camera. The second part of the experimentation consists on defining the same initial pose for the Aibo and then applying the visual tracking approach for some trayectories of the ball.

The initial pose of the Aibo for both experimentations is defined as a stable position of its body, the values of the joints are shown on table 3.1 and figure 3.13.

Body element	joint	value	joint	value	joint	value
Left front leg	1	0.1	2	0.1	3	0.1
Right front leg	1	0.1	2	0.1	3	0.1
Left back leg	1	0.1	2	0.1	3	0.1
Right back leg	1	0.1	2	0.1	3	0.1
Head	tilt	0.1	pan	0.1	nod	0.1

(*) the Aibo joints values are given in radians.

Table 3.1: Initial configuration of the joints of the Aibo

3.3.4.1 Visual tracking for a fixed ball

For the experiment, we want to use an homgenous distribution of the ball positions within the vision range of the Aibo, therefore we define a distance of $30cm$ as the reference distance between ball positions in x and y axis. The horizontal range of the camera is $\frac{\pi}{3}$ radians (aprox. 60^{o}), and we delimitate the vertical distance from the camera to the ball between $0.5m$ and $2m$. Defining the proyection of the initial center of the Aibo body over the ground

Figure 3.13: Initial configuration of the joints of the Aibo

as the origin of the fixed reference system, we define the positions showed in table 3.2, 18 positions in total. At every position we draw a circle of radio 13 cm. and into these circles we marked 32 points homogenusly distributed with 2 cm of distance between points in each axis. In figure 3.14 we show the distribution of the points into the circle.

	Coordinates			Coordinates			Coordinates	
Circle	x	y	Circle	x	y	Circle	x	y
1			7			13		
2			8			14		
3			9			15		
4			10			16		
5			11			17		
6			12			18		
	(a)			(b)			(c)	

(*) Coordinates of the ball are given in centimeters

Table 3.2: Positions of the ball

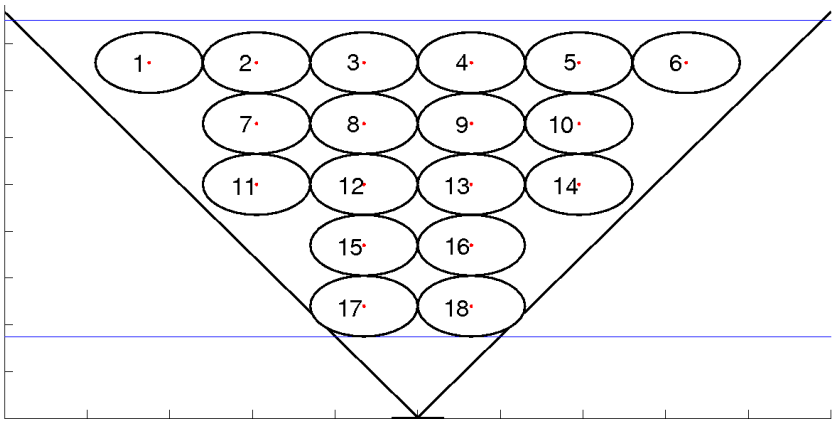


Figure 3.14: Aibo initial configuration and positions of the ball

We have 18 circles with 23 positions for the ball into each circle, so we totaly have 576 points into the vision range of the Aibo robot.

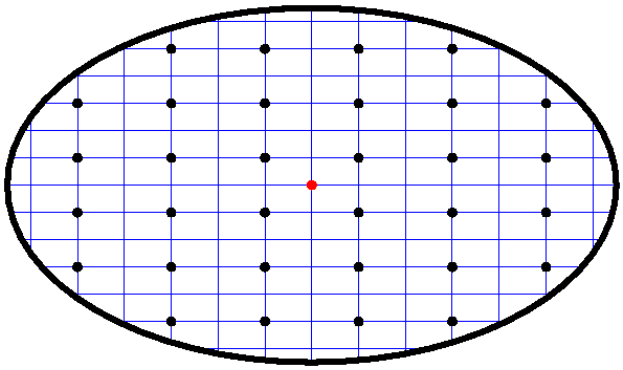


Figure 3.15: Stadistical region

The aibo robot estimate the distance from the ball to the camera taking into account the real diameter of the ball and its diameter in the image plane (equation 3.38) and then estimate de ball position in the camera reference system (equation 3.39). Figure 3.16 shows the teorical real distance from the ball in the camera reference system, the estimated distance by the robot, and the error between the real and estimated distance.

Errors in the estimated distance produce errors in the estimated position of the ball in the camera reference system. Figure 3.17 shows the teorical real

Figure 3.16

Figure 3.17

position and estimad position of the ball in the camera reference systems, and the error between them.

The average norm of the final error in the image plane is 0.0783 with a variance of 0.0027, the error distribution by distance is presented in figure 3.18.

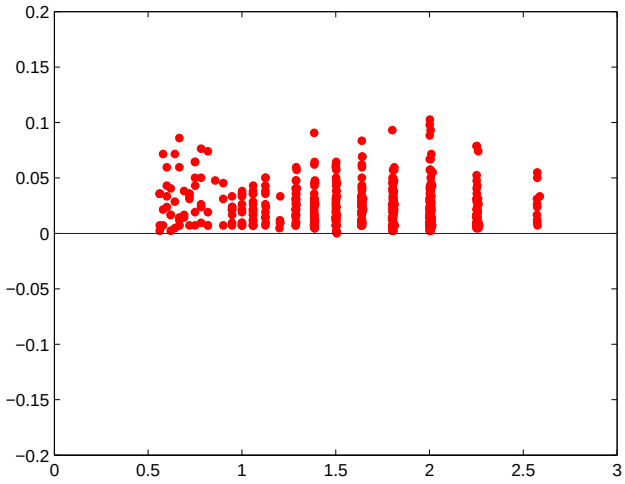
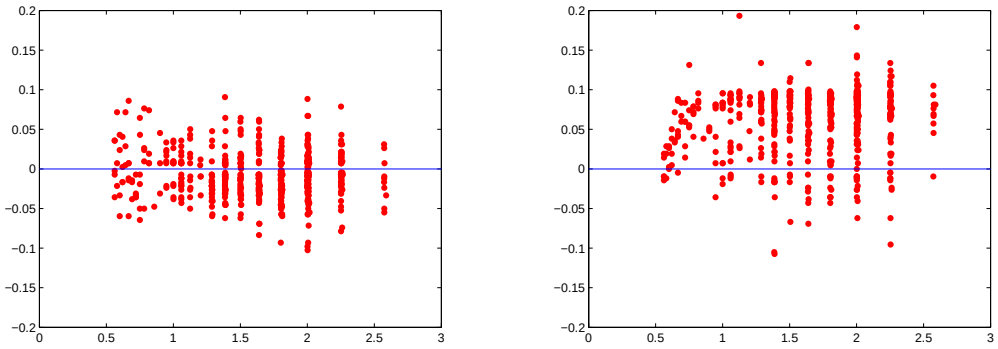


Figure 3.18: Final error norm distribution over initial distance in 3D the ball

The distribution of the error in the axes u and v is presented in figure 3.19. The average value for the u axis is -0.0166 with a variance of 0.0102, while for the v axis the average value is 0.1246 with a variance of 0.0095



(a) x-axis (b) y-axis

Figure 3.19: Final error distribution over initial distance in 3D

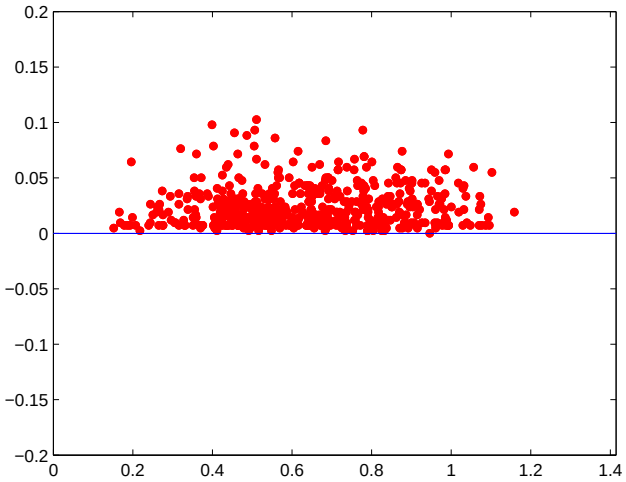
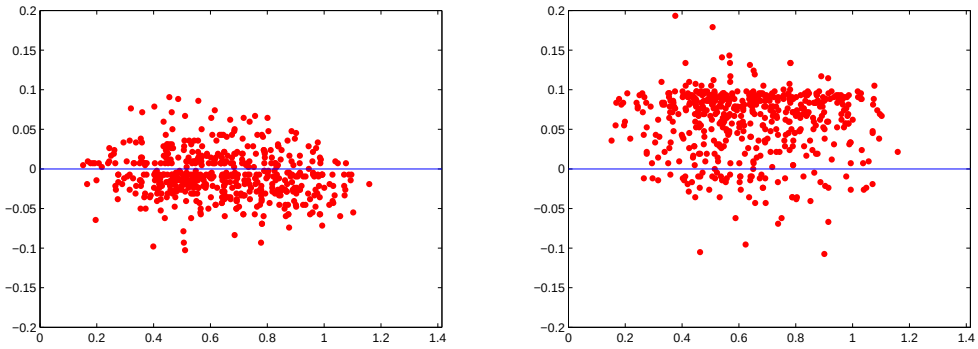
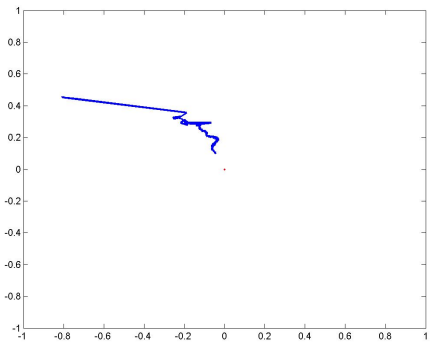


Figure 3.20: Final error norm distribution over intial distance in image plane

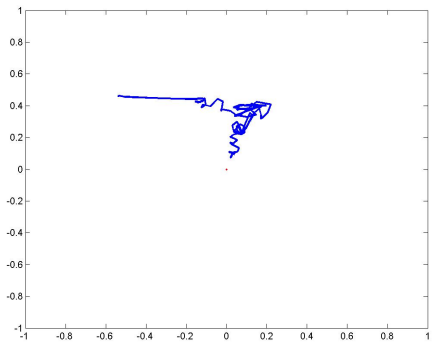


(a) u-axis (b) v-axis

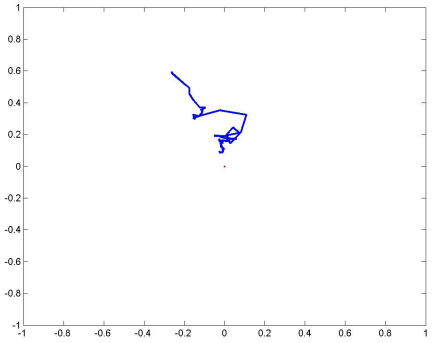
Figure 3.21: Final error distribution over initial distance in image plane



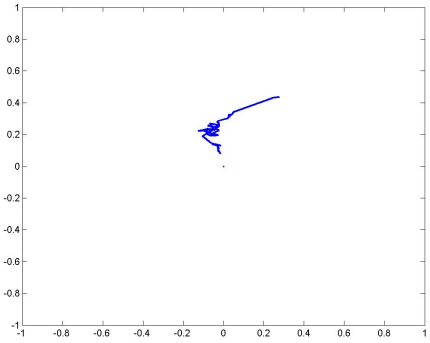
(a)



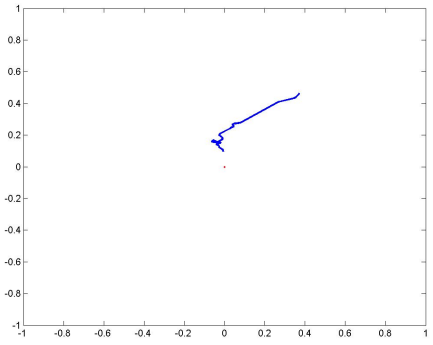
(b)



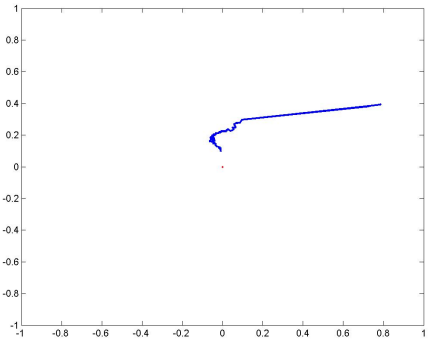
(c)



(d)



(e)



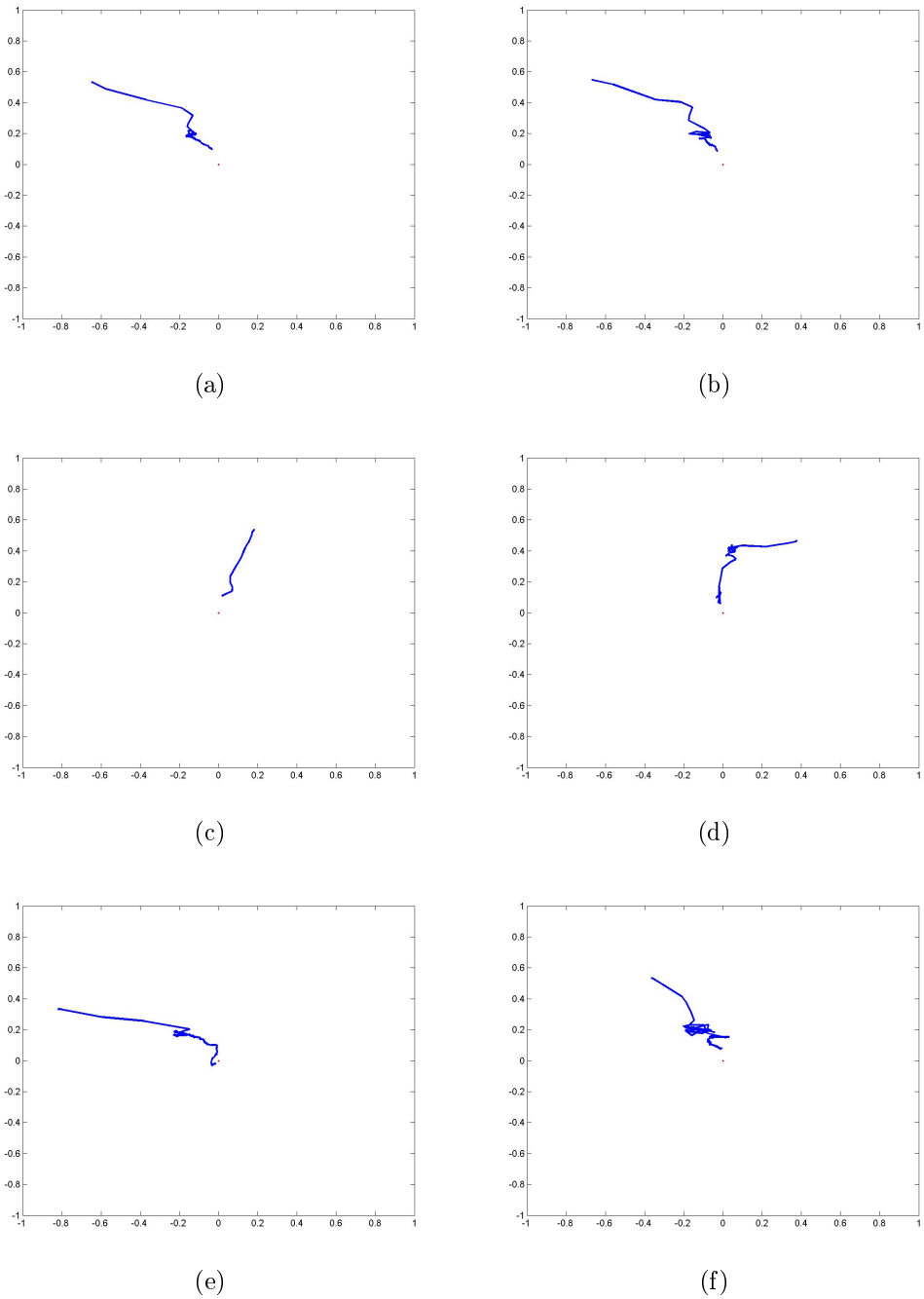
(f)

Figure 3.22: Trajectories of the ball from circle 1 to circle 6

In table 3.3 the error average and variance of the ball final positions in each circle is presented.

	Error norm		u-axis error		v-axis error	
Circle	Average	Variance	Average	Variance	Average	error
1	0.0821	0.0024	-0.0372	0.0030	0.0600	0.0012
2	0.0732	0.0007	-0.0063	0.0012	0.0505	0.0023
3	0.0642	0.0011	-0.0070	0.0009	0.0478	0.0020
4	0.0886	0.0097	-0.0235	0.0079	0.0699	0.0043
5	0.0757	0.0006	-0.0028	0.0003	0.0672	0.0015
6	0.0874	0.0062	0.0111	0.0069	0.0696	0.0021
7	0.0704	0.0008	-0.0125	0.0010	0.0515	0.0020
8	0.0791	0.0016	-0.0206	0.0013	0.0624	0.0023
9	0.0935	0.0016	-0.0027	0.0006	0.0866	0.0023
10	0.0860	0.0002	-0.0063	0.0007	0.0708	0.0020
11	0.0766	0.0010	-0.0238	0.0006	0.0661	0.0014
12	0.0868	0.0056	-0.0291	0.0065	0.0659	0.0015
13	0.0698	0.0016	-0.0001	0.0011	0.0539	0.0025
14	0.0737	0.0008	-0.0019	0.0010	0.0565	0.0021
15	0.0740	0.0097	0.0048	0.0071	0.0509	0.0056
16	0.0880	0.0004	0.0047	0.0005	0.0837	0.0007
17	0.0692	0.0038	-0.0161	0.0011	0.0530	0.0045
18	0.0665	0.0006	0.0344	0.0007	0.0503	0.0007
Total	0.0783	0.0027	-0.0083	0.0026	0.0623	0.0024

Table 3.3: Circles final error

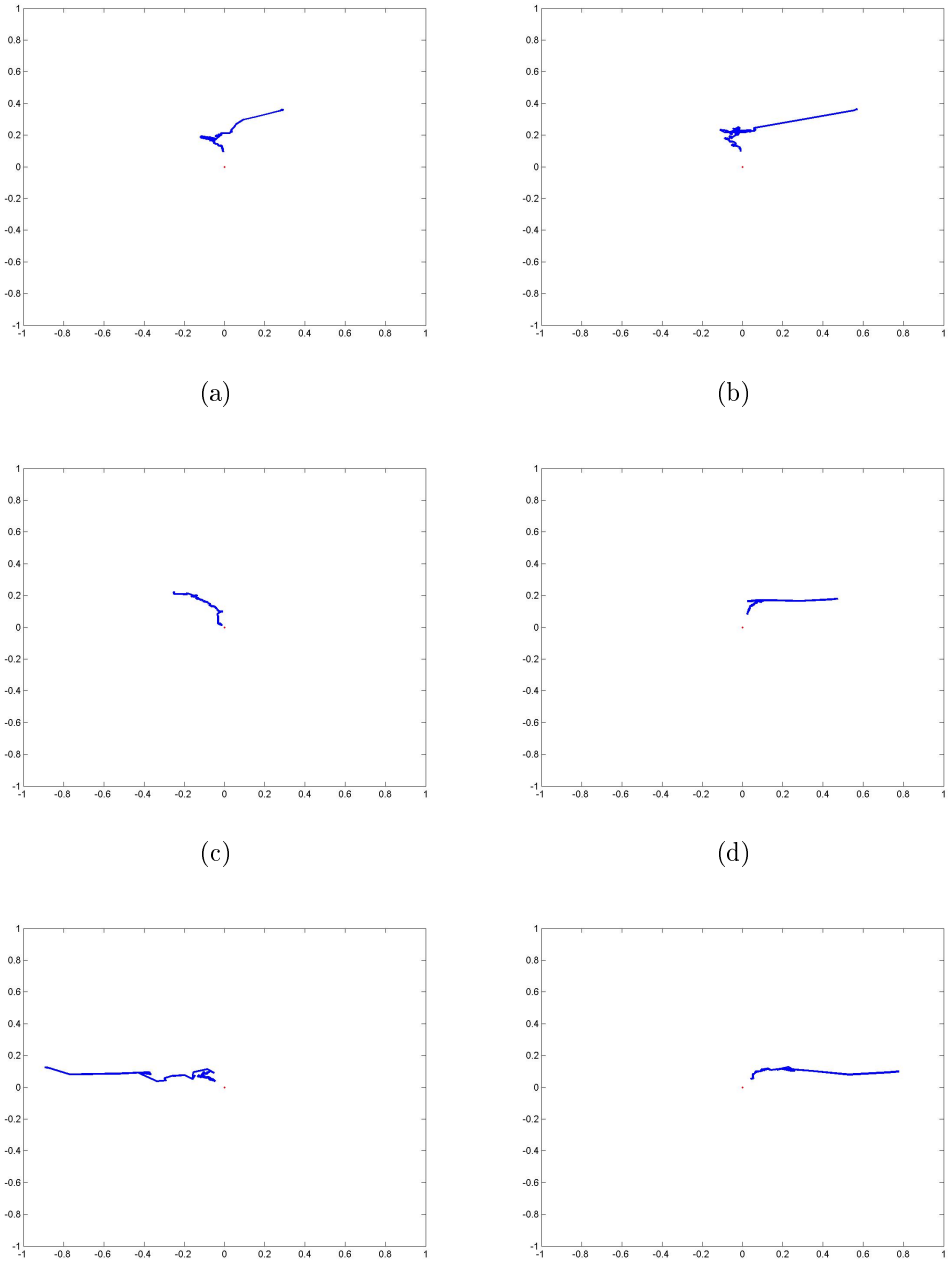


(a) (b)

(c) (d)

(e) (f)

Figure 3.23: Trajectories of the ball from circle 7 to circle 12



(a) (b)

(c) (d)

(e) (f)

Figure 3.24: Trajectories of the ball from circle 13 to circle 18

	Norm		u-axis error		v-axis error	
Circle	Average	Variance	Average	Variance	Average	error
1	0.0065	0.0016	0.0065	0.0016	0.0010	0,00018
2	0.0065	0.0003	0.0065	0.0003	0.0057	0,00015
3	0.0059	0.0002	0.0059	0.0002	0.0051	0,00011
4	0.0065	0.0009	0.0065	0.0009	0.0048	0,00051
5	0.0052	0.0005	0.0052	0.0005	0.0027	0,00006
6	0.0089	0.0027	0.0089	0.0027	0.0045	0,00032
7	0.0061	0.0004	0.0061	0.0004	0.0036	0,00009
8	0.0050	0.0002	0.0050	0.0002	0.0032	0,00007
9	0.0059	0.0003	0.0059	0.0003	0.0040	0,00013
10	0.0090	0.0006	0.0090	0.0006	0.0041	0,00008
11	0.0050	0.0005	0.0050	0.0005	0.0032	0,00007
12	0.0029	0.0002	0.0029	0.0002	0.0022	0,00006
13	0.0051	0.0007	0.0051	0.0007	0.0039	0,00021
14	0.0066	0.0005	0.0066	0.0005	0.0036	0,00008
15	0.0051	0.0002	0.0051	0.0002	0.0038	0,00007
16	0.0030	0.0002	0.0030	0.0002	0.0038	0,00004
17	0.0029	0.0002	0.0029	0.0002	0.0023	0,00006
18	0.0011	0.0000	0.0011	0.0000	0.0036	0,00000
Total	0.0052	0.00054	0.0052	0.00054	0.0036	0.00013

Table 3.4: Circles trajectory error variations

3.3.4.2 Visual tracking for a moving ball

For the experimentation of tracking a moving ball, we define an initial position of the ball and some linear trajectories. We start defining horizontal trajectories respect to the initial robot configuration, moving the ball from side to side at 0.5m and 2m from the camera. Figure 3.25 shows the four horizontal movements over the reference system..

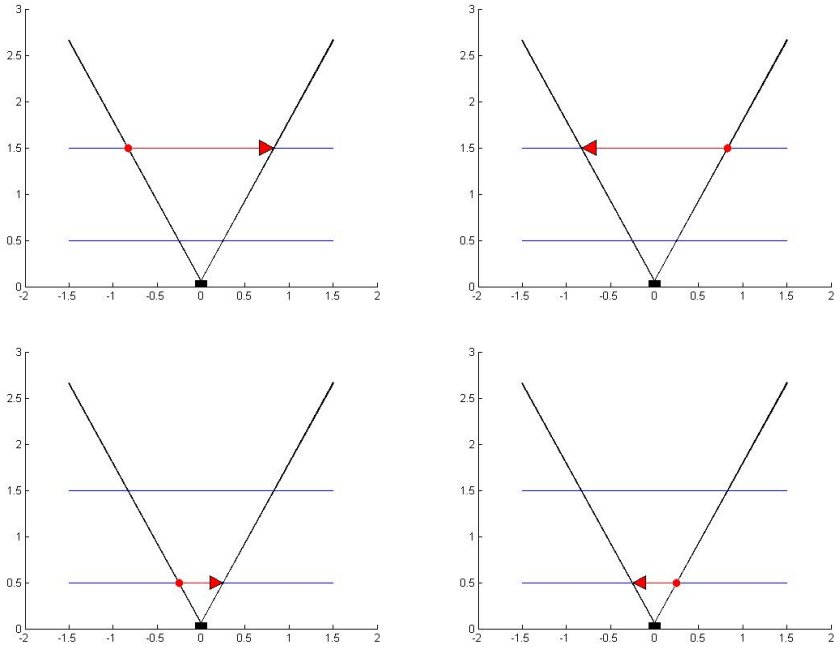


Figure 3.25: Horizontal movements of the ball

Then, we define the vertical movements choosing as coordinate x the most left and right positions for the center of the ball when it is over the horizontal line at $0.5m$ from the camera. The number of vertical trajectories is four, as shown in figure 3.26.

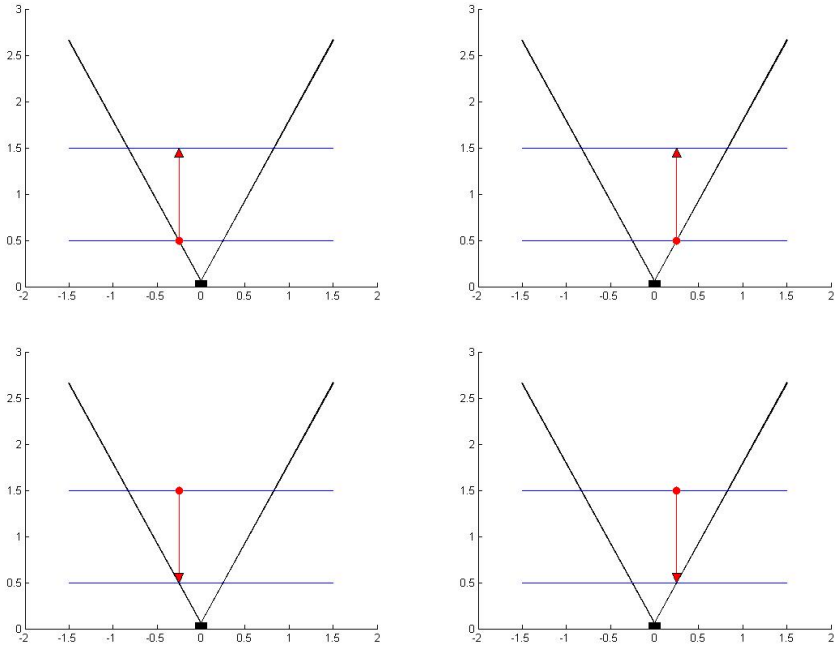


Figure 3.26: Vertical movements of the ball

Finally, we define transverse trajectories, between 0.5m and 2m. in y-axis from the camera, and the most left and right positions of the ball in the domain of vision.

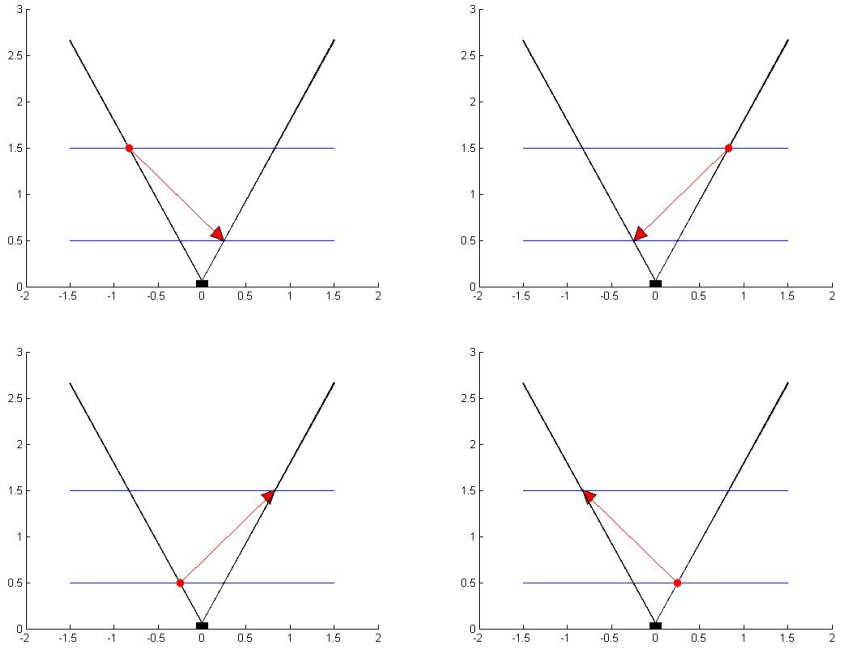


Figure 3.27: Transverse movements of the ball

3.4 Conclusion

We have developed the visual servoing for the whole set of degrees of freedom of the Aibo 7 following a principled approach. From the geometrical description of the robot we have constructed the full Jacobian matrix that linearizes the functional dependence of the image plane viewed by the robot camera on the robot degrees of freedom. The seudoinverse of this Jacobian matrix provide the desired controls. The blind application of this control strategy may lead the robot to unstable or unfeasible configurations for a standing pose. Therefore, we test the stability of the robot configuration. When it is compromised we restrict the visual servoing to the head. The implementation shows that the approach gives real time response when the seudoinverse is computed in the onboard processor of the robot. We are actually performing the real time experiments and collecting performance information.

Chapter 4

Control of a Multi-robot Hose System

Nowadays robotic systems are facing the challenge of working in very unstructured environments, such as shipyards or construction sites. In these environments, the tasks are non repetitive, the working conditions are difficult to be modeled or predicted, and the size of the spaces is huge. Moreover, there are complex tasks that a single robot can not accomplish and the use of a multi-agent system is required. In these environments, a common task is the displacement of some kind of flexible hose. It can be a water hose or a power line, or other. We are interested here in the design of a control architecture for a multi-robot system dealing with this problem. A collection of cooperating robots attached to the hose must be able to displace it to a desired configuration. We have identified the following sub-problems: modeling a flexible elongated object, distributed sensing on the robots to obtain information of the environment and/or of the configuration of the system including robots and the hose, inverse kinematics of the whole system, stable structural design, highly adaptive control via high level cognitive mechanisms. Here we focus on the hose modeling and the generation of control strategies for a collection of autonomous robots attached to it.

This chapters reports initial steps in the study of control strategies for a multi-robot system trying to move a flexible hose, this work belongs to the research line of Multi-component Robotic Systems [14]. Our starting point is the hose geometry modeling taking into account physic models for the internal dynamics. The control problem is then stated as the problem of reaching a desired configuration of the spline control points from an initial

configuration and the transport of the hose defining a trajectory for a leader robot and a vision based approach for the tracking trajectories of the other robots.

4.1 Objective

Our objective is focused on the transport of a hose of 2 meters long, by three robots, in a straight line. The hose is attached to the robots by a revolving platform in every robot that tightly grabs the hose. The positions of the hose in contact with the robots are constant, and the hose does not rotate in the contact points with the robots, moreover the leader and last robot will be in the ends of the hose.

The idea is to investigate the area of uni-dimensional objects modeling in order to obtain an appropriate representation for the hose model, then the simulation of the hose-robots system is required in order to obtain a control strategy that allows the transport of the hose by a given trajectory for the leader robot. Our finally objective is the real control of a hose, using a vision-camera sensorization and three real robots.

4.2 Hose Model

Modeling uni-dimensional objects has great application for the representation of wires in industry and medicine. The most popular models use differential equations [42], rigid body chains [25] and spring-mass systems [22]. Spring-mass systems and rigid body chains allow to simulate a broad spectrum of flexible objects, and they are rather versatile when simulating deformations. They are very fast to compute. However they are imprecise for uni-dimensional object modeling. The combination of spline geometrical modeling and physical constrains was introduced by [49] and they allow a continuously definition of the object. In many works the splines model have been used but due to the fact that are based exclusively on the control points of the spline they are not good for representing the hose torsion. The work of [64] has improved the spline representation by combining the splines modeling with the Coesserat rods theory, allowing to model the twisting of the hose. This new approach, known as Geometrically Exact Dynamic Splines (GEDS), represents the control points of the splines by the three Cartesian

coordinates plus a fourth coordinate representing the twisting state of the hose. Because we have chosen this approach for modeling the hose, we continue with the Cosserat rods theory and the explanation of the GEDS. We have chosen the GEDS, because they allow a continuous definition of the hose that accounts for the rotation of the transverse section at each point in the curve, and that an exhaustive and rigorous mechanical analysis has been developed.

The Cosserat rod theory [57, 2] is usually used in modeling uni-dimensional objects because it permits to model its physics behavior. In Cosserat rod theory an uni-dimensional object is described by a space curve $r(s)$ and a coordinate frame of directors $[e_1, e_2, e_3](s)$ attached to each point of the curve, the parameter s goes from an end of the curve in $s = 0$ to the other end in $s = L$, being L the length of the hose. The curve and the directors are joined into a coordinate frame $E(s) = [e_1, e_2, e_3, r](s)$. A graphic representation of the hose by the curve and the frame directors is shown in figure 4.1.

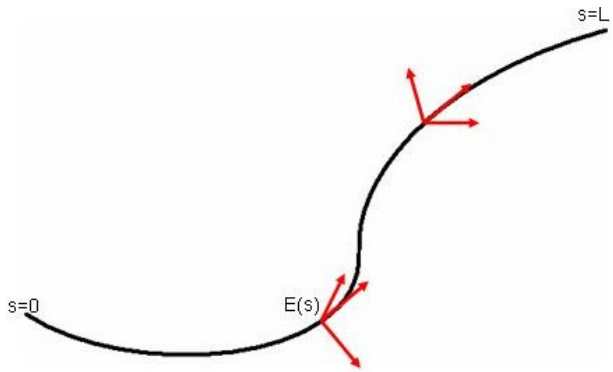


Figure 4.1: Cosserat rod model of a hose

The GEDS defines the uni-dimensional object by splines but taking into account the Cosserat rod approach in order to model the twisting behavior of the hose. A spline is a piecewise polynomial function. See figure 4.2 for an illustration. Splines define a curve by means of a collection of Control Points, which define a function that allows to compute the whole curve.

The spline expression for a curve is a summation of the controls points pondered by a polynomial value as function of a parameter u defined in $[0, 1)$. There are several kinds of polynomials and depending of the type selected the curve follows a specific form. In the following equation 4.1 the spline definition is presented.

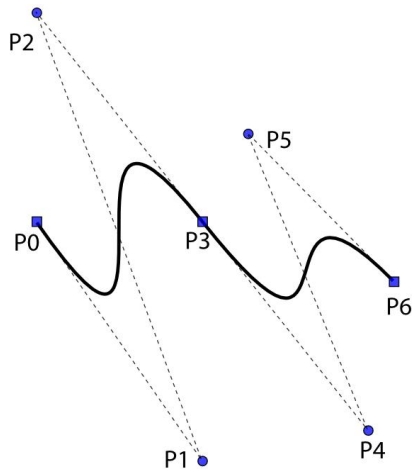


Figure 4.2: Cubic spline

$$q(u) = \sum_{i=1}^{n} b_i(u).p_i \tag{4.1}$$

Being $b_i(u)$ the polynomial for the control point p_i, and $q(u)$ the position of the curve at the parameter value u. It is possible to follow continually the curve by varying the u parameter value, starting at the end in $u = 0$ and finishing at the end in $u = 1$.

In our work we have used B-spline cubic curves for modeling the hose because it is a spline function that has minimal support with respect to a given degree, smoothness, and domain partition. Moreover, a fundamental theorem states that every spline function of a given degree, smoothness, and domain partition, can be represented as a linear combination of B-splines of that same degree and smoothness, and over that same partition [3]. When designing a B-spline curve, we only need a set of control points, a set of knots and a set of coefficients, one for each control point, so that all curve segments are joined together satisfying certain continuity condition.

Given $n+1$ control points $\{p_0, p_1, \ldots, p_n\}$ and a knots vector $U = \{u_0, u_1, \ldots, u_m\}$, the B-spline cubic curve of degree p defined by these control points and knots vector U is:

$$q(u) = \sum_{i=1}^{n} N_{i,p}(u).p_i \tag{4.2}$$

where $N_{i,p}(u)$ are B-spline basis functions of degree p.

The base functions are calculated by the Cox de Boor's algorithm:

$$N_{i,0}(u) = \begin{cases} 1 & u_i \leq u < u_{i+1} \\ 0 & c.c. \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} \cdot N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} \cdot N_{i+1,p-1}(u)$$

Because the B-splines representation uses the term $N_{i,p}$ for representing the polynomial of the control point p_i, we will continue by using $N_{i,p}$ in spite of b_i.

Due to the fact that the control points of the curve will vary in time, we rewrite equation **??** in terms of the temporal variable t, the use of the time parameter in the control points for splines representations receives the name of *Dynamic splines*:

$$q(u,t) = \sum_{i=1}^{n} b_i(u).p_i(t) \tag{4.3}$$

When modeling a hose, we assume that it has a constant sectional diameter, and that the transverse sections are not deformed in any way. If we do not take into account the hose internal dynamics, an spline passing through all the transverse section centers suffices to define the hose, as can be appreciated in figure **??**. If we want to take into account the hose internal dynamics, we need also to include the hose twisting at each point given by the rotation of the transverse section around the axis normal to its center point, in order to compute the hose potential energy induced forces. In the GEDS model, the hose follows the Cosserat rod approach and then is described by the collection of transverse sections. To characterize them it suffices to have: the curve given by the transverse section centers $c = (x, y, z)$, and the orientation of each transverse section θ. This description is summarized by the following notation: $q = (c, \theta) = (x, y, z, \theta)$. In figure **??**, the relation between the Cosserat rod directors and the twisting angle θ is shown, where Vector $\vec{t}$ represents the tangent to the curve at point c, and vectors $\vec{n}$ and $\vec{b}$ determine the angle θ of the transverse section at point c.

The hose mathematical representation is given by the following polynomial spline:

$$q(u,t) = \sum_{i=1}^{n} N_i(u).p_i(t) \tag{4.4}$$

where $N_i(u)$ is the basis function associated to the control point p_i, $u \in [0, L]$.

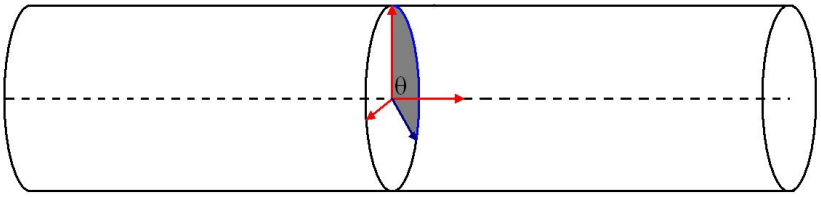


Figure 4.3: Hose section

From the Cosserat representation an energy and forces representation is obtained based on the Lagrange equation (equation 4.5). The study of the energy of the hose and the forces acting on it is needed to understand the dynamic behavior of the hose. The Cosserat representation allows a continuum definition of the hose that using the generalized coordinates q_i. The external forces acting on the hose and the forces induced by its potential energy are related by the Lagrange equations:

$$\frac{d}{dt}\left(\frac{\delta T}{\delta \dot{p}_i}\right) = F_i - \frac{\delta U}{\delta p_i} \tag{4.5}$$

The Lagrange equations use the potential energy U and the system's kinetic energy T. The kinetic energy is the motion energy, while the potential energy is the energy stored because of the hose position. F is the model of the external forces acting on the hose. It is usually assumed that mass and stress are homogeneously distribute among the n degrees of freedom of the hose.

Being $\dot{p}_i$ the time derivatives of the generalized coordinate p_i, and $\mathbf{p} = \{p_1, \ldots, p_i, \ldots, p_n\}$ the set of generalized reference coordinates.

4.2.1 Potential Energy

It is necessary to determine the forces that will be generated in the hose as a consequence of its energy configuration.

In figure 4.4 we can appreciate the forces and torques $\mathscr{F} = (\mathscr{F}_s, \mathscr{F}_t, \mathscr{F}_b)^t$ that deform the hose and perform some influence on its potential energy. The stretching force, $\mathscr{F}_s$, is the force normal to the hose transverse section and its application results in its lengthening. The tension torque, $\mathscr{F}_t$, makes the transverse section to rotate around the kernel curve. The curve torquing, $\mathscr{F}_b$, modifies the orientation of the transverse section. The forces acting on the

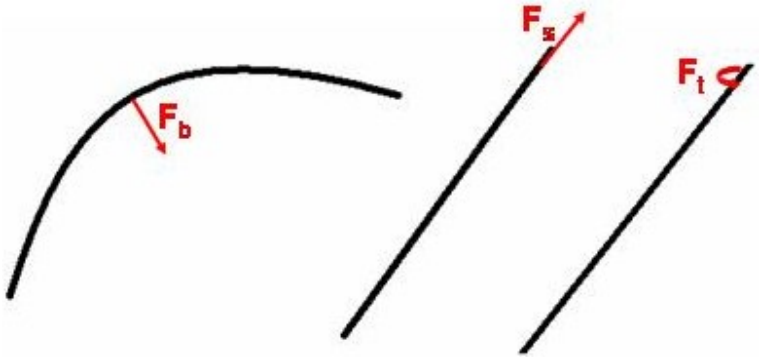


Figure 4.4: Forces induced by Potential energy of the hose

transverse section plane are neglected, because it is accepted the Kirchhoff assumption that considers that the transverse sections are rigid and that only the hose curvature may be distorted. Forces $\mathscr{F}$ are proportional to the strain ϵ.

In mechanics and physics an approximation for linear-elastic materials is used by the Hooke's law. This law establishes that the extension of a spring is in direct proportion to the load applied to it. Resuming, the Hooke's law for a spring-mass system establishes:

$$F = -kx \tag{4.6}$$

Being x the displacement of the spring due to the load applied to it, k the spring constant and F the restoring force experimented by the the spring due to its material properties.

In general the Hooke's law is applied to elastic materials because they behavior is similar to the spring as its molecules return to the initial state of stable equilibrium, quickly regaining the object its original shape after a force has been applied.

If the hose is of length L and with a transverse section of area A, the hose extension is linearly proportional to the resistance of the hose to be deformed:

$$\triangle L = \frac{F}{EA}L \tag{4.7}$$

Being E the modulus of elasticity, which is the mathematical description for the hose resistance to be deformed when a force is applied to it.

Working out the value of F from equation 4.7 we have:

$$F = EA\frac{\triangle L}{L} \tag{4.8}$$

Defining the strain ϵ as the deformation of the hose relative to the transverse area, $\epsilon = A\frac{\triangle L}{L}$, we can rewrite 4.8 as:

$$F = E\epsilon \tag{4.9}$$

When a small stress is considered for a relative big radius of the hose in comparison with the transverse section, it is said that lineal elasticity exists, and then the force definition of 4.9 may be applied.

The matricial version for the stretching, twisting and bending forces is:

$$\mathscr{F} = H\epsilon = \begin{pmatrix} Es & 0 & 0 \\ 0 & Et & 0 \\ 0 & 0 & Eb \end{pmatrix} \epsilon \tag{4.10}$$

The stress vector ϵ, is composed by the stretching stress ϵ_s, the twisting stress ϵ_t and the bending stress ϵ_b. The Hooke matrix, H, is composed by: Es the stretching rigidity, Et the twisting rigidity and Eb bending rigidity.

Continuing with the analogy for the spring-mass system, the potential energy, U, is defined by $U = \frac{1}{2}kx^2$, that for the hose is defined by the following integration from the end in $s = 0$ to the end in $s = L$:

$$U = \frac{1}{2}\int_0^L \epsilon^t \mathscr{F} ds \tag{4.11}$$

Using the definition of $\mathscr{F}$ from equation 4.10 in equation 4.11 we have:

$$U = \frac{1}{2}\int_0^L \epsilon^t H\epsilon ds$$

Note that this model is appropriated for a hose that in rest configuration is stiffed and not twisted or bended, but for a cable as a telephone cord or a spring the rest configuration of the hose is different to zero, so ϵ should be replaced by $(\epsilon - \epsilon_0)$, being ϵ_0 the rest strain.

4.2.2 Kinetic energy

Due to the fact that the hose is defined by the position and rotation over every point of the hose center curve, the kinetic energy T is composed by translation energy T_t and rotation energies T_r. The translation energy represents the displacement of a point in the hose while the rotation represents the rotation of the hose:

$$T = T_t + T_r \tag{4.12}$$

The kinetic energy is given by:

$$T_t = \frac{\mu A}{2} \int_0^L \|\dot{q}\|^2 \, ds \tag{4.13}$$

$$T_r = \frac{\mu}{2} \int_0^L \Omega^T I \Omega ds \tag{4.14}$$

Being Ωangular velocity vector and μ the lineal density.

Because the points are give in Cartesian positions and the twisting angle, a simplified version of the kinetic energy expression is done by defining the inertial matrix J, invariant over all the hose points due to the fact that the hose diameter is assumed to be constant.

$$J = \begin{pmatrix} \mu & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & I_0 \end{pmatrix}$$

The kinetic energy of the hose T is then defined by::

$$T = \frac{1}{2} \int_0^L \frac{dp^t}{dt} J \frac{dp}{dt} ds \tag{4.15}$$

Being I_0 the polar momentum of inertia.

4.2.3 Dynamic

The kinetic energy take into account the translational and rotational movements of the hose, so obtaining it we may determine the accelerations of every point of the hose, deriving equation 4.15 in order to obtain the left expression of equation 4.5 we have:

$$\frac{d}{dt}\left(\frac{\delta T}{\delta \dot{p}_i}\right) = \frac{1}{2}\int_0^L \frac{d}{dt}\frac{\partial \dot{p}^t J \dot{p}}{\partial \dot{p}_i} ds \tag{4.16}$$

Taking into account the Lagrange formulation (equation 4.5) and supposing we know the external forces over the hose, in order to get the value of expression 4.16 we have to derived the potential energies:

Deriving the potential energy in function of a generalized coordinate we get:

$$\frac{\partial U}{\partial p_i} = \frac{1}{2}\int_0^L \frac{\partial \epsilon^t H \epsilon}{\partial p_i} ds \tag{4.17}$$

The aim is to determine the accelerations of the hose, so depending on its geometrical model the accelerations are obtained in the GEDS model substituting q in the expression of equation 4.16 by the right side of equation 4.3:

$$\frac{d}{dt}\frac{\partial T}{\partial p_i} = \sum_{j=1}^{n} J \int_0^L (b_i(s) b_j(s)) ds \frac{d^2 p_j}{dt^2} \tag{4.18}$$

Then the following definitions are done: $M = J\int_0^L (b_i(s)b_j(s))ds$ and . Using M and A in the Lagrange equation (4.5):

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{p}_i} = \sum_{j=1}^{n} M_{i,j} A_j \tag{4.19}$$

Using equations 4.19 and 4.17 the Lagrange equation is written in a matricial way:

$$MA = F + P \tag{4.20}$$

Being the four subsystem for x, y, z and θ independents.

4.2.4 Hose configuration

We define the configuration of the system composed by the hose an the robots, taking into account the control points and knots of the B-spline representation, and the u parameter values for the robots.

More formally, a configuration, h, of the hose-robots system is defined as:

$$h = \{\mathbf{p}, \mathbf{U}, \mathbf{U_r}\} \tag{4.21}$$

Being:

- $\mathbf{p}$, the control points vector of the hose.
- $\mathbf{U}$, the knots vector.
- $\mathbf{U_r} \subset \mathbf{U}$, the robots knots vector.

The robots knots vector, U_r, contain the values of the parameter u for which the spline obtains the positions of the robots. If we define as u_{r_i} the value for the i-th robot in $\mathbf{U_r}$, then the position of the spline at u_{r_i} is equal to the position of the i-th robot, r_i:

$$q(u_{r_i}) = \sum_{i=1}^{n} b_i(u_{r_i}).p_i = r_i \tag{4.22}$$

We do not assume that a configuration for the hose-robots system is always given, in contrast we assume that our starting point is a sequences of points of the hose center curve, τ, containing the Cartesian position of every point (x, y, z) and its torsion angle θ. Then, we construct the hose configuration by an interpolating method that, given a sequence of points for which the hose pass, generate the control points of the B-spline cubic curve that interpolates this points. The method we used for interpolating is the Interpolating forward backward algorithm for clamped B-spline cubic curves, and a description of this method is presented in Appendix A.

More precisely, we make an uniform selection of the sequence of points from τ in order to obtain an uniform B-spline interpolating that approximate correctly the form of the hose, avoiding the occurrence of peaks, protuberances and loops. The selection is done taking into account the number of knots we want to use, depending on the relation between precision and time cost we desire.

This construction of the hose configuration from a sequence of points is useful when we have a vision system that allows us to obtain a sequence of points from the hose and the positions of the robots contact points.

4.3 Hose control

We arise two goals in the positioning of the hose by the positioning of several autonomous robots $\mathbf{r} = \{r_1, \ldots, r_m\}$ attached to it. The first objective is to

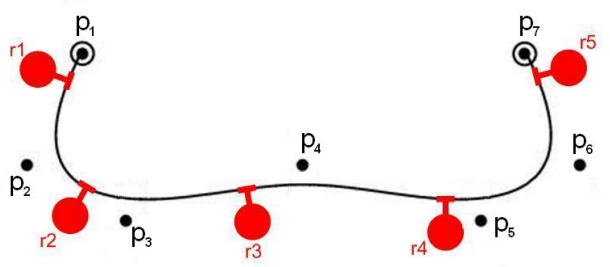


Figure 4.5: Spline Control Points $\mathbf{p}_i$ and positions of the robots $\mathbf{r}_i$.

bring the hose from an initial configuration to a final configuration by obtaining the velocities that the robots must experiment, this work is developed in the following section 4.3.1. The second objective is: given a trajectory for the leader robot, the robot at the extreme of the hose in $u = 0$, define a control strategy for the rest of robots in order to make the transport of the hose, this work is developed in section 4.3.2.

4.3.1 Hose configuration control

The first objective is based on a given initial and final hose configurations, as well as the initial robot positions. In figure 4.5 it can be appreciated the problem configuration, with a hose described by parametric cubic splines with control points p_i and a collection of robots r_j attached to it.

Let it be:

- $\mathbf{h}_0$ the initial hose representation given by a sequence of points representing the center curve and torsion of the hose.
- $\mathbf{h}_*$ the desired hose representation, specified by a sequence of points representing the center curve and torsion of the hose.
- $\mathbf{r_0} = \{r_1, \ldots, r_m\}$ the robot initial positions.

We are interested in obtaining the motion of the attached robots, given by the instantaneous velocities of the hose attachment points $\dot{\mathbf{r}}$, that will approximate the hose from the initial configuration representation $\mathbf{h_0}$to a desired configuration $\mathbf{h}_*(u)$ with an initial configuration of the robots $\mathbf{r_0}$. In order to use the model of the hose based on splines, we need to obtain a B-spline

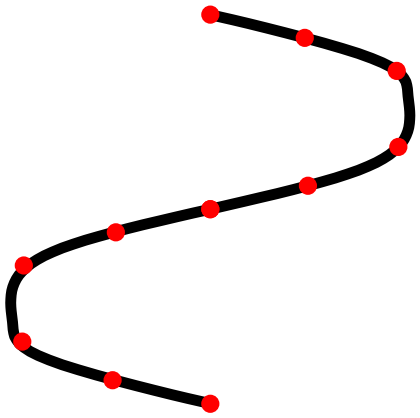


Figure 4.6: Uniform selection of the interpolating points

representation for the hose, so we define the hose configurations by interpolating an uniform selection of the given sets of points $\mathbf{h}_0$ and $\mathbf{h}_*$ as explained in section 4.2.4. The new configurations of the hose are now given by a sequence of control points, p_0 for the initial configuration and p_* for the desired configuration.

The uniform selection of the interpolating points from the sequence representing the hose, $\mathbf{h}$, is done by dividing the hose length by $n-1$, being n the number of control points, and choosing for each division point the point of h most closed to it.. Figure 4.6 shows the hose in red and the selected interpolating points in black, for a number of control points $n = 11$.

4.3.1.1 Control law

In order to obtain the desired velocities of the hose control points that reduce the distance between they real positions and its desired ones, we define the error hose configuration as the difference between the current control points and the desired one as $e(\mathbf{p}) = (\mathbf{p}_* - \mathbf{p})$.

The error function allows us to define the following simple proportional control law:

$$\dot{\mathbf{p}} = k.e(\mathbf{p}) \tag{4.23}$$

This expression defines a differential equation on the control points, making out the control points position we obtain the following trajectory for the control points:

$$\mathbf{p(t)} = e^{-K(t-t_0)} + \mathbf{p}_*$$

Being t_0 the time instant at which $p(t_0) = p_0$, and $K \in \mathbb{R}^{4n}$ the vector with all its elements equals to k.

Because our systems is expressed in function of accelerations, we have to obtain the accelerations that generate the news velocities. Te control points acceleration is the temporal derivative of its velocity:

$$\ddot{\mathbf{p}}(t) = \frac{d\dot{\mathbf{p}}(t)}{dt}$$

Because we define an iterative control, we define the acceleration in the $k+1$ step in terms of the desired velocity in $k+1$, the current velocity in k and the temporal increment:

$$\ddot{\mathbf{p}}_{k+1} = \frac{\dot{\mathbf{p}}_{k+1} - \dot{\mathbf{p}}_k}{\triangle t} \tag{4.24}$$

Once we have defined the desired accelerations on the control points, we need to determine the forces that robots should apply in order to obtain this accelerations.

4.3.1.2 Forces applied by robots

In equation 4.20 we defined an expression that relates the control points acceleration with the intern energy of the hose and the external forces applied to it. We differentiate between the nett external forces, F, the generalized forces of the applied ones by the robots, F_p, from the other external forces, F_e:

$$F = F_p + F_e \tag{4.25}$$

So, we rewrite equation 4.20 in order to define the forces the robot must apply as function of the control points desired accelerations, the generalized external forces on the hose and the energy configuration:

$$F_p = MA - F_e - P \tag{4.26}$$

Because the dynamic is continuously defined over the spline, a force f applied in a particular point of the hose is discomposed by the generalized forces f_i in the spline control points p_i. When differencing the power, $W =$

Fq, respect to the control points p_i, the correspondent generalized force f_i is obtained:

$$f_i = \frac{\partial W}{\partial P_i} = f\frac{\partial q}{\partial p_i} = fb_i \tag{4.27}$$

We compute the partial derivative of a point $q(u)$ in the curve as a function of the control point P_i :

$$\frac{dq(u)}{dP_i} = N_{i,p}(u) \tag{4.28}$$

Defining the Jacobian matrix J_{rq} of the robots contact points with the hose as a function of the control points, we have:

$$J_{pr} = \begin{pmatrix} \frac{\partial q(u_{r_1})}{\partial p_1} & \cdots & \frac{\partial q(u_{r_m})}{\partial p_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial q(u_{r_1})}{\partial p_n} & \cdots & \frac{\partial q(u_{r_m})}{\partial p_n} \end{pmatrix} = \begin{pmatrix} b_1(u_{r_1}) & \cdots & b_1(u_{r_m}) \\ \vdots & \ddots & \vdots \\ b_n(u_{r_1}) & \cdots & b_n(u_{r_m}) \end{pmatrix} \tag{4.29}$$

Being u_{rj} the attachment point of the robot r_j to the hose.

So, after have obtained the forces F_p that must be generated by the robots over the control points, we have to determine the forces, F_r, that the robots must apply in the robot contact points of the hose. We use matrix J_{pr}, defined in equation **??** and we define u_{r_j} as the value of parameter u in the robot contact point r_j of the hose, obtaining the following expression that defines the relation between the applied forces in the robot contact points, $\mathbf{F_r}$, and the generalized forces on the control points $\mathbf{F_p}$:

$$\mathbf{F_p} = J_{pr}.F_r \tag{4.30}$$

Now we want to determine the instant forces that every robot must apply in order to the get the desired generalized forces in the control points of the hose. In order to obtain this forces we have to obtain the inverse of matrix J_{pr} (equation 4.30), but it is not possible due to the fact that in general matrix J_{pr} is not invertible.

We take into account three possible cases depending of the number of control points,n , and the number of robots, m. If $m = n$, then exists the inverse of J_{pr} and we can conclude that $F_r = J_{pr}^{-1}F_p$.

In contrast, if $m \neq n$, the inverse does not exist. Assuming that J_{pr} is full rank, then its pseudo-inverse by least squares can be computed, whose general solution is:

$$F_r = J_{pr}^{+} F_p + (I - J_{pr}^{+} J_{pr}) v_m \tag{4.31}$$

Being J_{pr}^{+} a pseudo-inverse of J_{pr} and $v_m \in R^m$. The solution by least squares allows us to obtain a value for Fr that minimizes the following norm $\|F_p - J_{pr} F_r\|$.

In obtaining the pseudo-inverse we have to take into account two possible cases. First, if $n > m$ there are more control points than robots and then, from the theorem of the implicit function the control points $p_{m+1} \dots p_n$ can be expressed as a combination of the control points $p_1 \dots p_m$. So, we deduce that there are $n - m$ redundant control points. In this case, the appropriate pseudo-inverse is:

$$J_{pr}^{+} = (J_{pr}^{T} J_{pr})^{-1} J_{pr}^{T} \tag{4.32}$$

In this case we have that $(I - J_{pr}^{T} J_{pr}) = 0$, because the dimension of the kernel of J_{pr} es 0. So, the solution can we rewrites as:

$$F_r = J_{pr}^{+} F_p \tag{4.33}$$

In the second case, if $n < m$ the system is under-constrained, so there are not enough degrees of freedom to uniquely determine the forces that the robot must apply. In this case the appropriate pseudo-inverse is:

$$J_{pr}^{+} = J_{pr}^{T} (J_{pr} J_{pr}^{T})^{-1} \tag{4.34}$$

In general, for $n < m$, $(I - J_{pr}^{T} J_{pr}) \neq 0$, and all of the vector of the form $(I - J_{pr}^{T} J_{pr}) b$ belongs to the kernel of J_{pr}, so the solution is given by:

$$F_r = J_{pr}^{+} F_p + (I - J_{pr}^{+} J_{pr}) b \tag{4.35}$$

4.3.1.3 Velocities and accelerations of the robots

After have obtained the forces the robot must apply to the hose, we have to determine the velocities of the robots contact points as consequence of the applied forces, because most of the robots do not accept forces commands

but accept Cartesian velocities commands and then regulates its actuators in order to obtain this velocities.

We have obtained the forces that the robot must apply on the contact points with the hose, but this forces not necessary generate the desired generalized ones, because in generally the inverse of J_{pr} do not exist we have obtained an approximation by least squares. So, we obtain the generalized forces, $\hat{F}_p$, that generate the applied ones by the robots:

$$\hat{F}_p = J_{pr}.Fr \tag{4.36}$$

Introducing $\hat{F}_p$ in the Lagrange equation (4.20) we get:

$$M\hat{A} = \hat{F}_p + P \tag{4.37}$$

Using a LU decomposition of M we obtain the generalized accelerations $\hat{A}$, and then we obtain the robots accelerations by using J_{pr}:

$$\hat{A}_r = J_{pr}^{+}.\hat{A} \tag{4.38}$$

We obtain the estimated robots velocities for next step, $\hat{V}_r(k+1)$, by making out them from equation 4.24.

Because the robots have physics limitations, neither every velocities nor every accelerations may be experimented by a robot, so we define v_m and a_m as the respective max norm of the velocity and acceleration that a robot can apply. In order to contemplate this limits in our approach, after have obtained the desired accelerations of the robots we limit them by the rule defined in algorithm 4.1.

The velocities $\hat{V}_r$ obtained after have applied algorithm 4.1 are used as the commands for the robots.

4.3.2 Hose transport control

4.3.2.1 Configuration trajectory generation

In this section we apply the hose transport for a set of robots by defining a trajectory for the spline model of the hose, and deriving from the configuration of the hose the velocities of the robots. So, we aim to obtain the desired velocities of the robots that make the hose follows a given trajectory.

Defining the velocity magnitude reference as v_{ref}, we attempt to obtain a norm of the mean control points velocity, $\left\|\bar{\dot{\mathbf{p}}}\right\|$, as close to v_{ref} as possible,

Algorithm 4.1 Physic velocity and acceleration limitations of the robots

1. $\bar{v}$=max norm of the robots velocities $\hat{V_r}(k+1)$.
2. if $\bar{v}$>v_m then
   (a) for each $\hat{v_r} \subset \hat{V_r}$ and $\hat{a_r} \subset \hat{A_r}$
      i. $\hat{v_r}(k+1) = \hat{v_r}\frac{v_m}{\bar{v}}$
      ii. $\hat{a_r} = \frac{\hat{v_r(k+1)} - \hat{v_r(k)}}{\triangle t}$
3. if $\bar{a}$>a_m then
   (a) for each $\hat{v_r} \subset \hat{V_r}$ and $\hat{a_r} \subset \hat{A_r}$
      i. $\hat{a_r} = \hat{a_r}\frac{a_m}{\bar{a}}$
      ii. $\hat{v_r}(k+1) = \hat{v_r(k)} + \hat{a_r}.\triangle t$

until we were in the proximity of desired control points positions $\mathbf{p}_*$. As we saw in equation 4.23, the velocities of the hose control points depends on the error between the current and the desired control points positions by a constant gain, $\dot{\mathbf{p}} = k.(\mathbf{p}_* - \mathbf{p})$. In order to approximate the reference magnitude, we use the variable f in terms of the current and desired control points positions, $\mathbf{p}$ and$\mathbf{p}_*$, and the velocity magnitude reference v_{ref} . Because we want to obtain a continuous advance velocity magnitude of the hose until it is close to the desired configuration, we rewrite the proportional control law as follow:

$$\dot{\mathbf{p}} = k.(f(v_{ref}, \mathbf{p}_*, \mathbf{p}) - \mathbf{p}) \tag{4.39}$$

We interpolate the configurations of the sequence, h_k, by a clamped B-interpolating curve, denoted by $\phi(\xi)$, with $\xi \in [0,1]$, $\phi(0) = \mathbf{p_0}$ and $\phi(1) = \mathbf{p}_*$. The number of control points, m, of ϕ is the same as the number of configurations of the sequence h_k. Then, the norm of the mean control points velocity may be written as:

$$\left\|\bar{\dot{\mathbf{p}}}_\mathbf{i}\right\| = k\left\|\overline{\phi_i(\xi) - \mathbf{p}}\right\|, \; i \in [1, m] \tag{4.40}$$

So, the work is reduced to find, in every step, the value t that approxi-

mates the velocity reference magnitude v_{ref}.

$$\min_{\xi} \left(v_{ref} - \left\|\overline{\phi_i(\xi) - \mathbf{p}}\right\|\right) \tag{4.41}$$

The interpolating clamped B-splines curve is explained in appendix A.

4.3.2.2 Follow the leader approach

In this section we explain the work done for a set of robots transporting a hose by applying the strategy of follow the leader. In this work the trajectory of the leader robot is assumed to be given, and the aim is to define a strategy for to followers robots. We develop the control of the hose based on the configuration of the hose segments and on the distances between robots.

We define an heuristic for the transport of a hose taking into account the form of the hose segments between robots. In this approach every robot, except the leader, controls its velocity and direction based on the segment of the hose between it and its previous robot. We assume that if a pair of robots are close the hose segment between them follow a curve but if they are sufficiently separated, the hose is stretched and approximate the straight line between the robots.

The idea is that if a the segment of the hose between a pair of adjacent robots is very stretched then the back robot must increase its velocity in order to allow a curvature of the hose. In contrast, if the hose segment is very curved then the back robot must reduce its velocity in order to allow an stretching of the hose segment. Figure 4.7 shows this idea.

The curvature of a hose segment is measured as the proportion between the maximum distance d_h from the hose curve h to the line L_{r_1,r_2} defined by the robot's positions (r_1,r_2) and the distance between robots, d_r.

$$\mathfrak{c} = \frac{d_h}{d_r} \tag{4.42}$$

where

$$d_h = max \left\|h_i - L_{r_1,r_2}\right\|, \ \forall h_i \in h$$

$$d_r = \|r_1 - r_2\|$$

The distance from the hose to the straight line defined by the robots positions is computed as the largest distance between the straight line $L_{r_1,r_2}$and

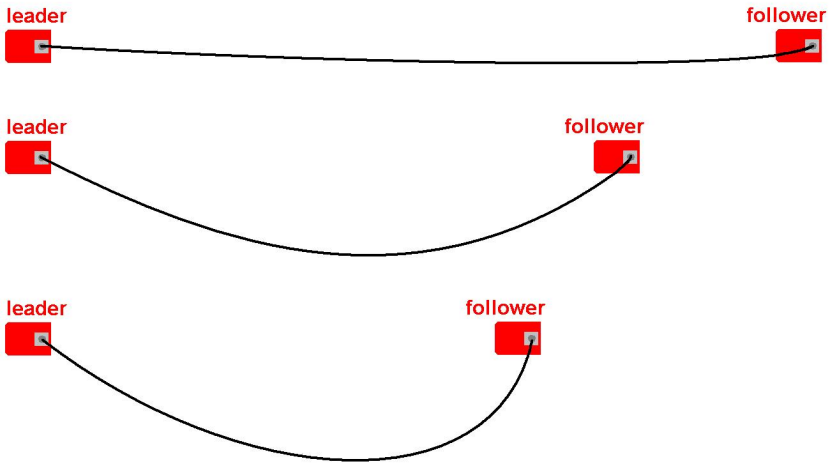


Figure 4.7: Hose segment according tho the robots distance

the points h_i of the segment hose h. The distance defined that way is showed in figure 4.8.

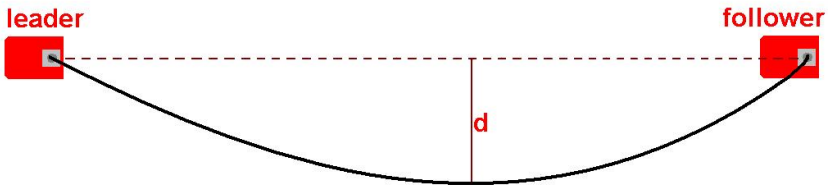


Figure 4.8: Segment width

The magnitudes d_h and d_r give us the relation between the dimensions of the rectangle that encloses the hose segment, being d_r the length of the rectangle and d_h its width. We define a maximum and minimum segment curvature for the transport of the hose, denoted by $\underline{\mathfrak{c}}$ and $\bar{\mathfrak{c}}$, and three velocity magnitude levels, w_0, v_1 and w_2 assigning the medium magnitude velocity, w_1, for the leader robot, the follower robots determine their velocities by the heuristic presented in algorithm 4.2.

Besides the control heuristic for the the magnitude of the robot velocity, me define a strategy for the velocity direction. We aim to maintain the robots formation in a straight line, so we define in every processing step the velocity direction of a follower robots by an intermediate directions between

Algorithm 4.2 Heuristic for the velocity of a follower robot

1. if $\mathfrak{c}_i < \underline{\mathfrak{c}}$then
   (a) $w_{i+1} = w_2$
2. else
   (a) if $\mathfrak{c}_i > \bar{\mathfrak{c}}$ then
      i. $w_{i+1} = w_0$
   (b) else
      i. $w_{i+1} = v_1$

Being c_i the curvature of segment i and w_{i+1} the velocity magnitude for robot $i+1$.

its current velocity direction and the direction of the previous robot in order to gradually align the robots in the transport of the hose. In figure 4.9 we define v_l as the velocity of the leader robot and v_f as the velocity of the follower robot.

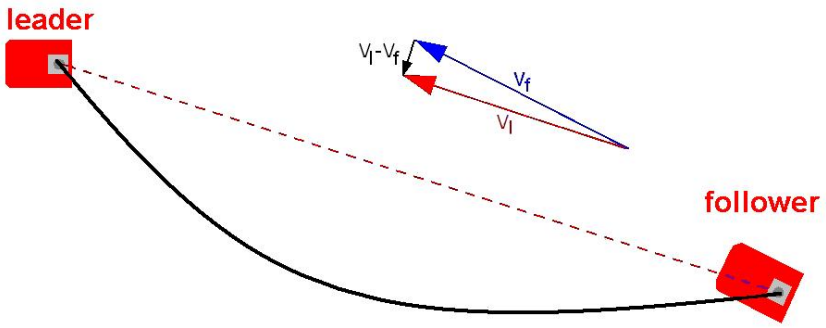


Figure 4.9: Velocity direction for the follower robot

We modify the velocity direction of a follower robot respect to its previous robot, by the vectorial sum showed in figure 4.10, adding to the follower robot a vector in the direction of the difference between the previous robots direction and its current direction, then we divide this vector by its norm in order to obtain a vector of unit norm.

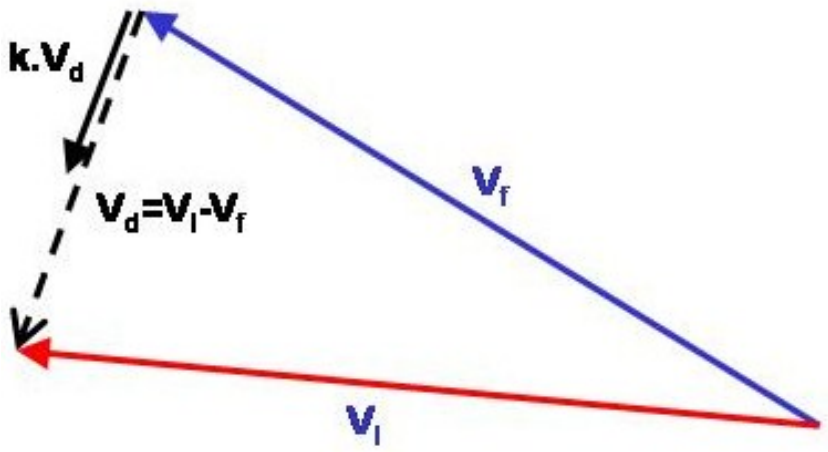


Figure 4.10: Follower robot velocity

time-step	Discretization	large	density
1 ms	1 cm	1m	200 grs/m

Table 4.1: hose parameters

The new direction for the follower robot velocity is defined by the following equation, being k a constant that determine the speed at which the follower robot approximate its direction, $\vec{v_{i+1}}$, to the previous robot direction, $\vec{v_i}$.

$$\vec{v_{i+1}} = \frac{\vec{v_{i+1}} + k(\vec{v_i} - \vec{v_{i+1}})}{\|\vec{v_{i+1}} + k(\vec{v_i} - \vec{v_{i+1}})\|} \tag{4.43}$$

Finally, the velocity of robot i is defined as $w_i\vec{v_i}$.

4.4 Simulation

We modeled the hose dynamic in Matlab for 3, 5 and 11 robots. For the physic model of the hose we only implemented the bending and stretching forces, because we assume that the twisting forces are not insignificant for the movements of the robots due to the fact that the grasping of the hose is very tightly and then the robot do not twist the hose. The parameters of the hose are resumed in table 4.1.

We assumed that robots make decisions of control in a frequency of 30ms, so every 30 ms the robots make desertions about changes in their velocities.

4.4.1 Hose configuration control

In the simulation of the hose configuration control of the hose we start from a sequences of points of the initial hose state $\mathbf{p_0}$ and the initial robots positions $\mathbf{r_0}$. In every step of the simulation we obtain the velocities of the robots $\mathbf{v_r}$ that make decrease the distance from the initial hose state $\mathbf{p_0}$ to the the desired hose state $\mathbf{p}*$.

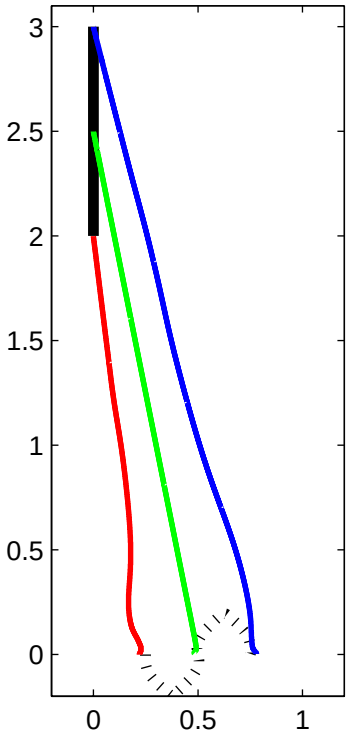
From $\mathbf{p_0}$ we obtain an uniform sequence of m interpolating points, $\mathbf{q_0}$, knots vector $\mathbf{U}$ and the robots knots vector $\mathbf{U_r}$, and from $\mathbf{p}_*$ a sequence of m interpolating points, $\mathbf{q}_*$ as inputs of the algorithm 4.3, where we present the simulation steps in the control of the hose configuration.

We simulate the control of the hose by defining an initial an final configuration of the hose, and then applying the control defined in section 4.3.1 to reach the desired configuration. As have been presupposed, the proportional control of the hose may reach a local minimum in which does not exist a decreasing direction that reduce the distance between the current and desired configurations of the hose. Moreover, some desired configurations of the hose can not be reached because it is impossible to obtain movements in the robots for obtaining a stable configuration of the hose with minim energy.

The aim of the hose configuration control is triying to get a desired configuration of the hose, but it is not possible to reach any arbirtray configuration due to the fact that there are not enough robots for determining the desired movements in the degrees of freedom. Assuming the velocity of every control point of the hose can be controlled, a proportional control law might be used in order to obtain the trajectory from the initial configuration to the desired configuration of the hose. In figure 4.11 we present the initial configuration of the hose in continous black and the desired configuration in discontinous black, the trajectory of the three robot are presented in red, green and blue, althoug the control law is defined on the control points space.

$$\dot{\mathbf{p}} = k.e(\mathbf{p}) \tag{4.44}$$

The derived velocities of the robots contract points are aprroximatly a striaght line until the robots are near to their final positions, then their trajectories vary in order to enhance the contact points positions. Finally



(a) Ideal robots contact points trajectories

Figure 4.11: Ideal trajectory of robots without dynamics

Algorithm 4.3 Hose configuration control

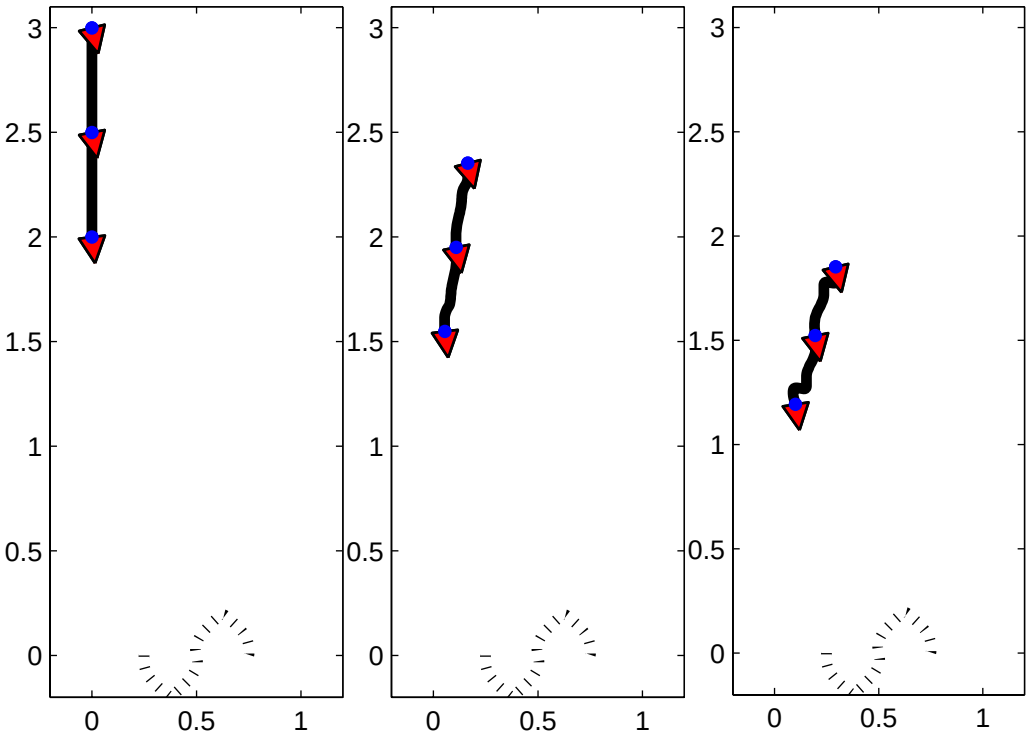
1. $h_0(\mathbf{p_0},\mathbf{U},\mathbf{U_r})$= interpolating B-spline $(\mathbf{q_0},\mathbf{U},\mathbf{U_r})$
2. $h_*(\mathbf{p}*,\mathbf{U},\mathbf{U_r})$=interpolating B-spline$(\mathbf{q}_*,\mathbf{U},\mathbf{U_r})$
3. Jir=getRobotsJacobian$(\mathbf{U_r})$
4. loop ($\|\mathbf{p}_* - \mathbf{p_0}\|$>tolerance)
   (a) M= MassMatrix
   (b) P= Derivatives of potential energy
   (c) Fe=Generalized external forces
   (d) $V_{next} = k(p * -p)$ <− proportional control law
   (e) $\mathbf{A}$=getAcceleartions($V_{current}$,V_{next}) <− physics limitations of the robots
   (f) $\mathbf{F} = M\mathbf{A} - \mathbf{P} - F_e$
   (g) $\mathbf{F_r} = \mathbf{J^+_{ri}}.\mathbf{F}$
   (h) $\mathbf{F_r}$=getRobotForces($\mathbf{F_r}$,$\bar{f}$)
   (i) $\mathbf{A} = M^+[J_{ir}F_r + P + F_e]$
   (j) $\mathbf{V}$=integrateAccelerations(A)
   (k) $V_r = J^+_{ir}V$

the control points converge to their desired positions. A sequence of the ideal trajectory is presented in figures 4.12 and 4.13.

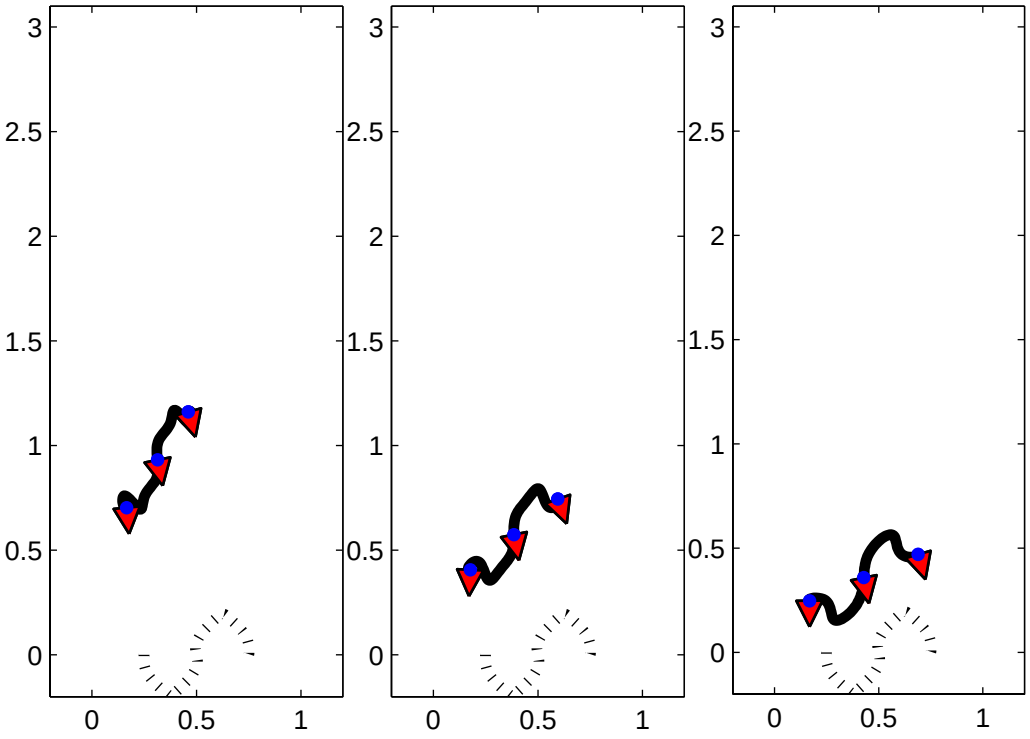
Then, we repeat the experiment but in spite of defining the control law in the control points space we define it in the robots velocities, applying this velocities to the robots in a simulation of a hose with internal dynamics. We define the velocities not taking into account the dynamic of the hose but the we apply the obtained velocities for the robots to a hose with internal dynamics. The control law is as follow:.

$$\dot{\mathbf{r}} = J_{ri}[k.e(\mathbf{p})] \tag{4.45}$$

Although we do not take into account the dynamics of the hose in de-

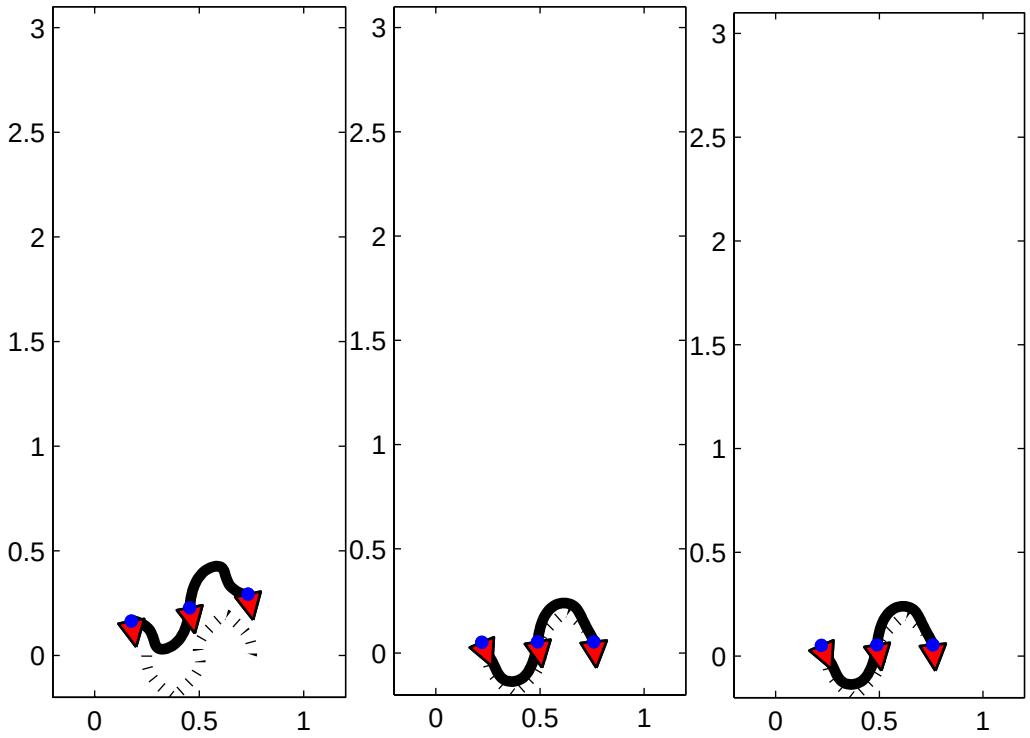


(a) At time 0s (b) At time 0.5s (c) At time 1s

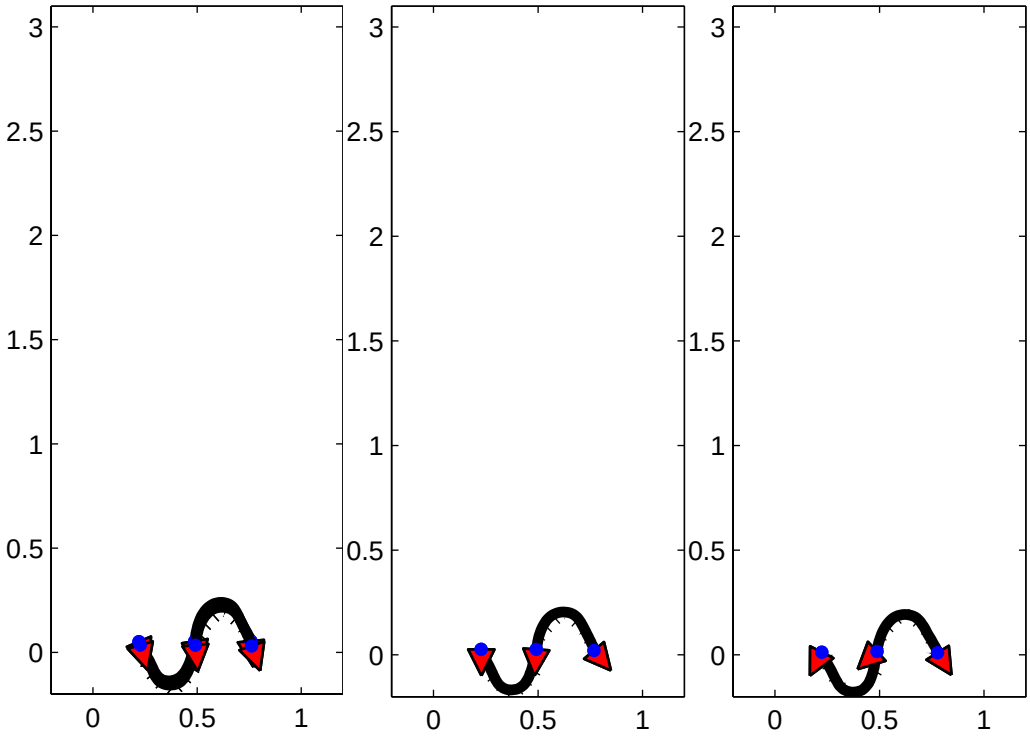


(d) At time 2s (e) At time 4s (f) At time 5s

Figure 4.12: Ideal sequence of the hose without dynamics



(a) At time 7s (b) At time 9s (c) At time 0s



(d) At time 11s (e) At time 13s (f) At time 18s

Figure 4.13: Ideal sequence of the hose without dynamics (cont.)

termining the velocities of the robots, the hose adopt a different form as a consecuence of the dynamic model, so the robot's velocities are different in comparison with the preovious case because the control points are not placed at the same positions in both approaches. In figure 4.14 the trajectories of robots are presented, where every trajectory is a straight line from the intial position to the final position of the robot. This case, the robots are not able to enhance the control points positions, in other words the matrix J_{ri} is not invertible, so not any trajectory of the control points can be determined by a trajectory of the robots.

.

Then, we use a control law defined in the robots velocities space, but deriving the robots velocities from the dynamic of the hose, as explained in section 4.3.1. The Trajectory of the robots for a hose with internal dynamics, obtaining the robots velocities from the dynamic of the hose is presented in figure 4.17.

4.4.2 Hose transport control

4.4.2.1 Follow the leader approach

First, we implemented the proposed approach for the transport of the hose, this approach has been acquired for a determined range of velocities, but the higher the velocity is, the lees efficacy of this approach. The process for the hose transport control simulation is presented in algorithm 4.4.

The bending force, as the force derived from the potential energy that resists to the bending of the hose, only depends on the form of the hose, so its magnitude is the same independently of the velocities that robots apply to the hose. Then, depending on the value of parameter EB and the distance between robots, and in minor importance on the friction force, the occurrence of loops may be possible, In figure 4.20 the ocurrence of a loop between the leader and the second robot is presented.

Increasing the parameter EB of the bending force, the ocurrence of a loop disapeared for the same velocities and distances between robots, obtaining the constant configuration of the hose when the hose form is stabilized. In figure 4.21 the configuration of the hose is shown, where can be appreciated the form of the hose between the leader robot and its immediately follower without loops. At this value for the bending parameter, the approach considering the hose segments in determining the magnitude of velocities has

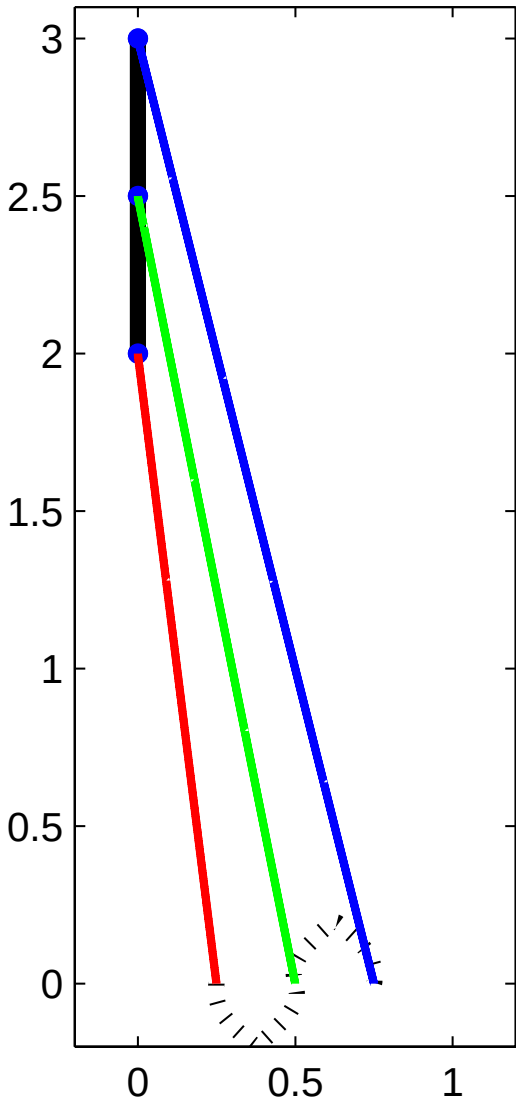
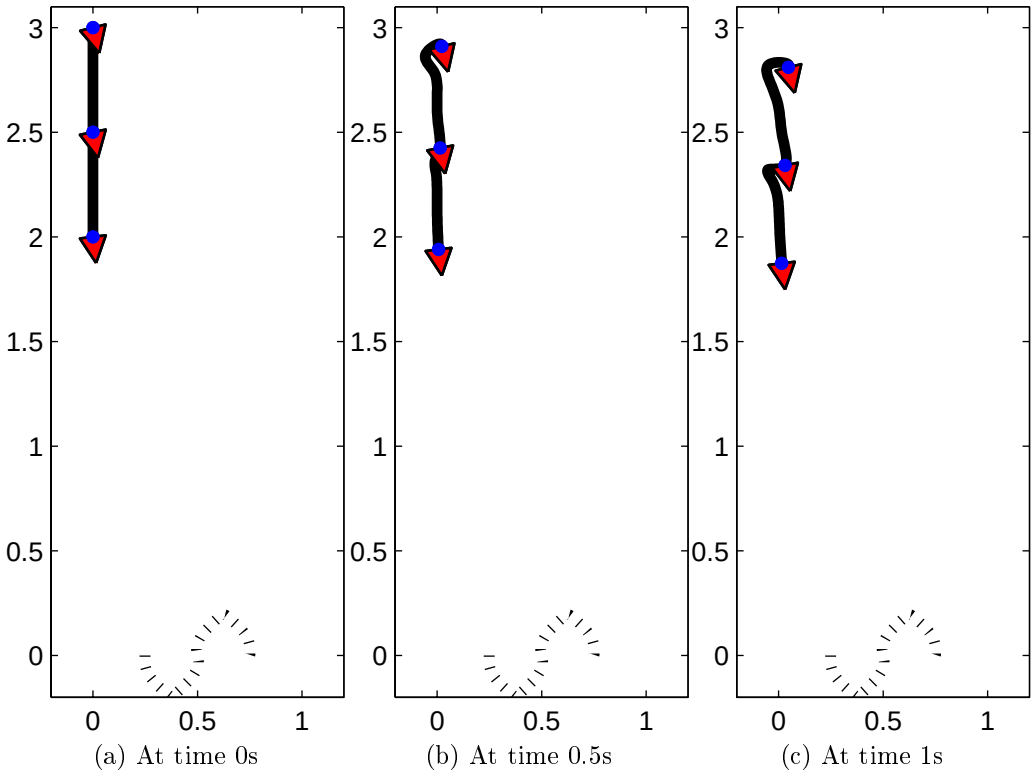
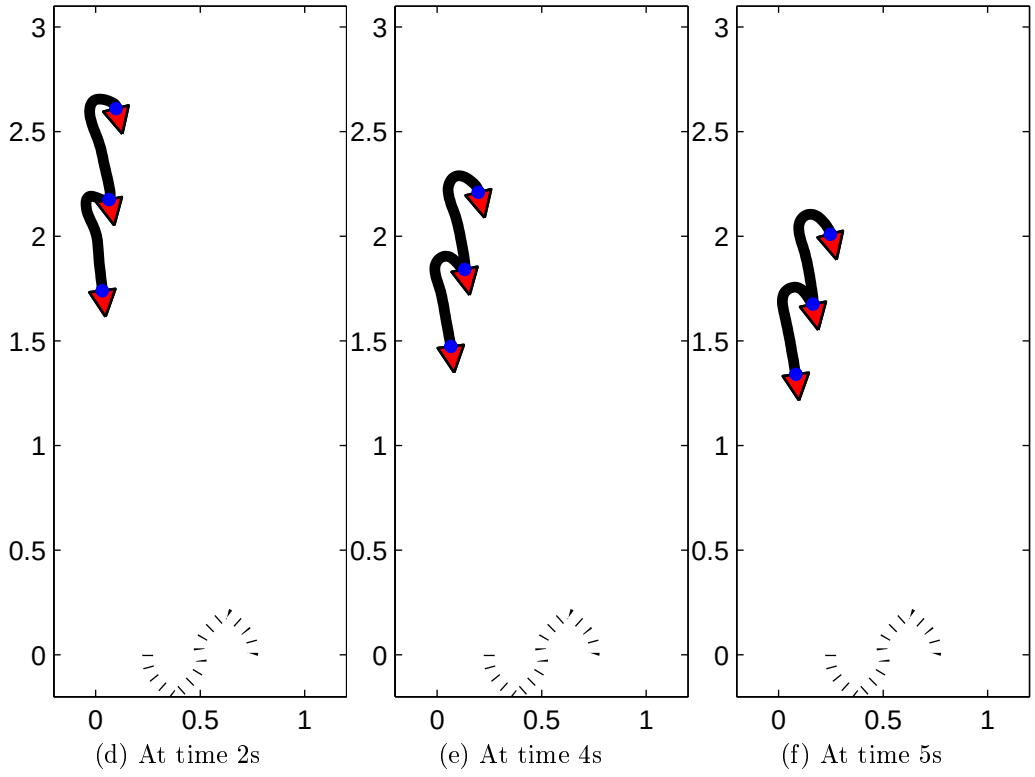


Figure 4.14: Trajectories of the robots without dynamics in the contro lawl

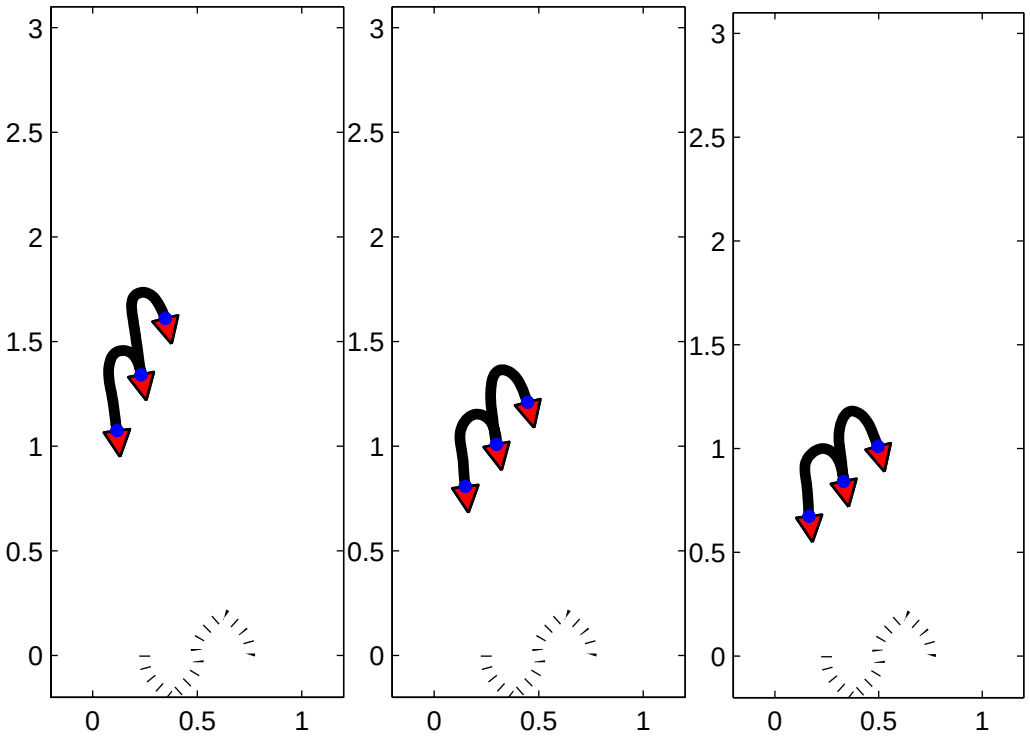


(a) At time 0s (b) At time 0.5s (c) At time 1s

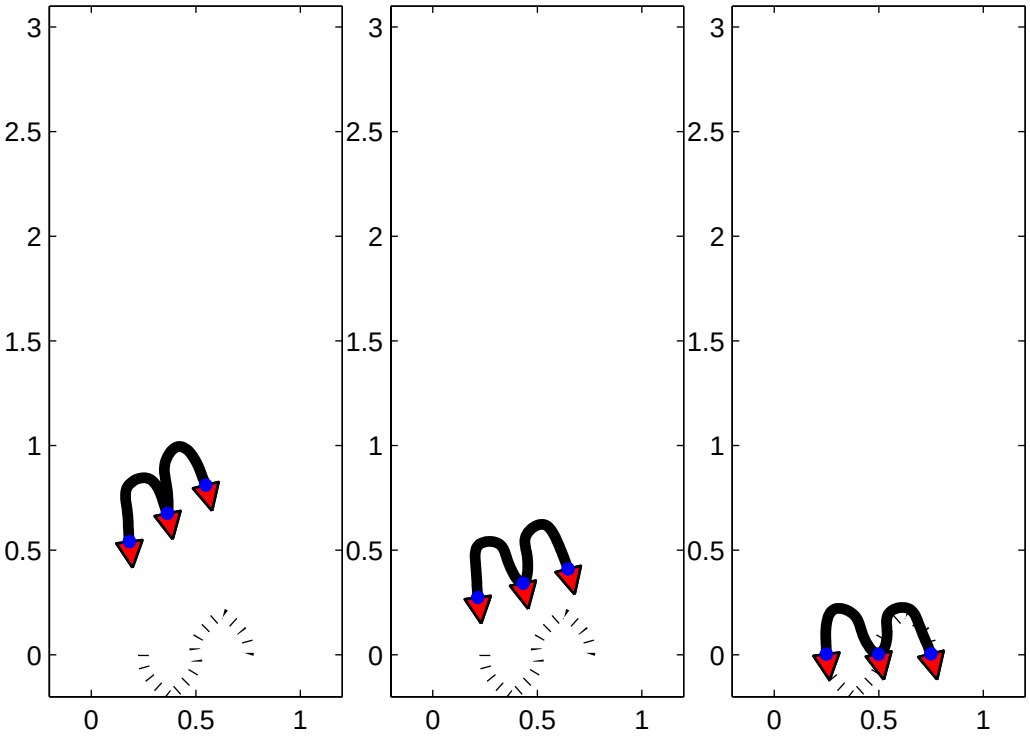


(d) At time 2s (e) At time 4s (f) At time 5s

Figure 4.15: Sequence of the hose with hose internal dynamics



(a) At time 7s (b) At time 9s (c) At time 0s



(d) At time 11s (e) At time 13s (f) At time 18s

Figure 4.16: Ideal sequence of the hose with hose internal dynamics (cont.)

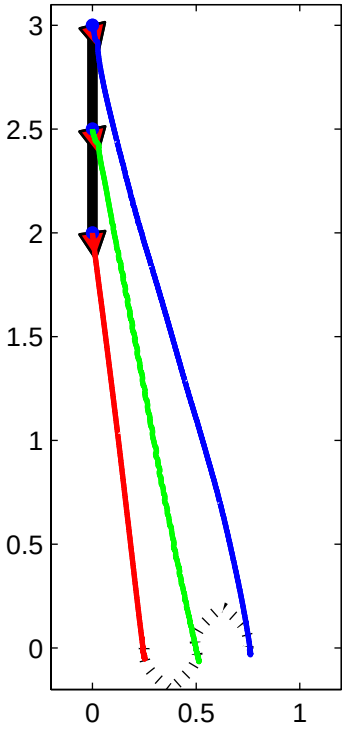
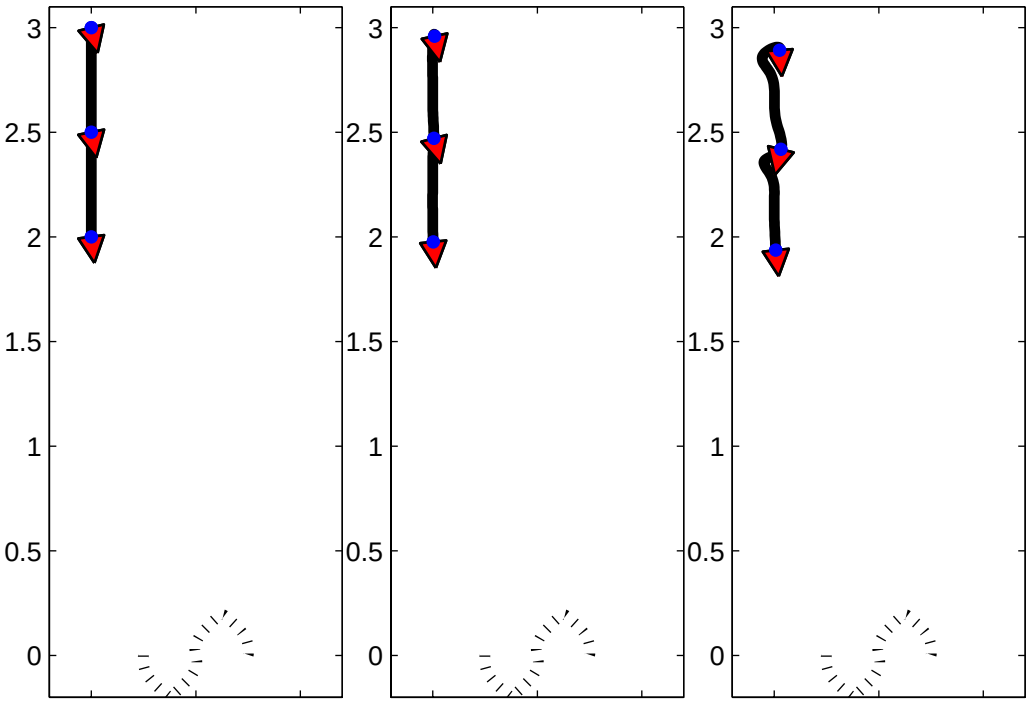
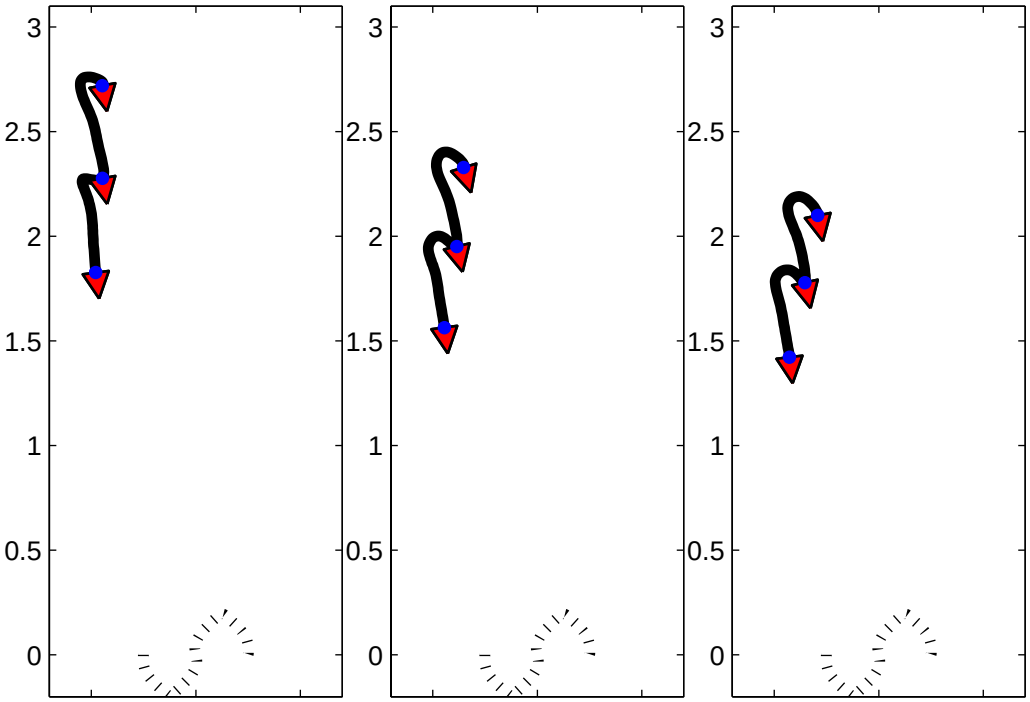


Figure 4.17: Trajectories of the robots from the dynamic in the control law

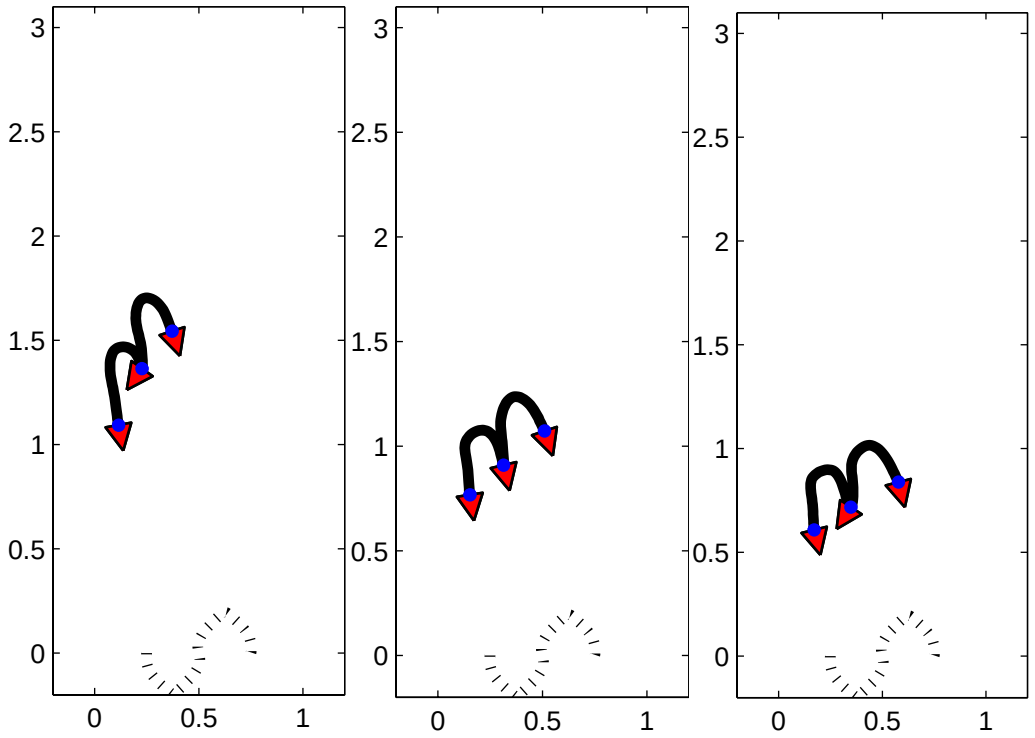


(a) At time 0s (b) At time 0.5s (c) At time 1s

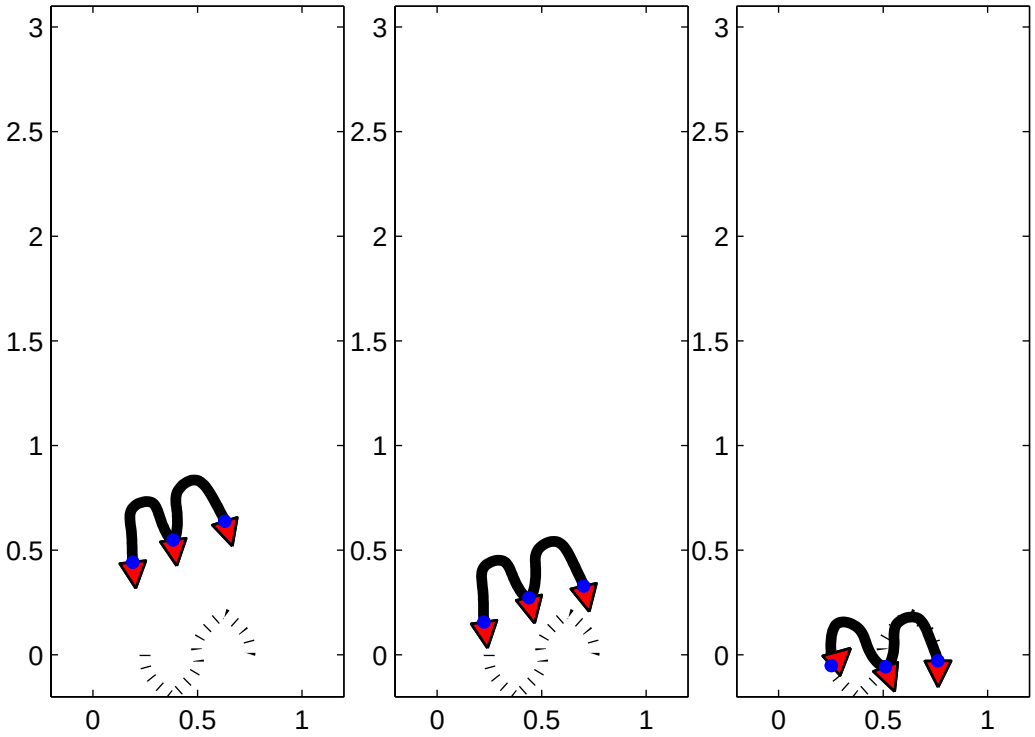


(d) At time 2s (e) At time 4s (f) At time 5s

Figure 4.18: Sequence of the hose with hose internal dynamics and dynamic control law



(a) At time 7s (b) At time 9s (c) At time 0s



(d) At time 11s (e) At time 13s (f) At time 18s

Figure 4.19: Ideal sequence of the hose with hose internal dynamics and dynamic control law(cont.)

Algorithm 4.4 Hose transport control

1. $h_0(\mathbf{p_0},\mathbf{U},\mathbf{U_r})$= interpolating B-spline $(\mathbf{q_0},\mathbf{U},\mathbf{U_r})$
2. Jir=getRobotsJacobian($\mathbf{U_r}$)
3. loop
   (a) M= MassMatrix
   (b) P= Derivatives of potential energy
   (c) Fe=Generalized external forces
   (d) for each follower robot j
      i. $\mathbf{V}r_j$=getVelocity
      ii. $\mathbf{A_{r_j}}$=getLimitedAcelerations(V_{r_j},$\bar{v}$,$\bar{a}$)
   (e) $F = (MJ_{ir})A_r - \mathbf{P} - F_e$
   (f) $F_r = J_{ir}^{+}F$
   (g) $A = M^{+}[(\mathbf{J_{ir}F_r}) - \mathbf{P} - F_e]$
   (h) A= fmincon(A,J_{ri},A_r);
   (i) $\mathbf{p}$=integrateAccelerations(A)

the desired behavior.

We applied both approaches for the transport of the hose, the approach considering the segment width and the approach considering the distance between robot, by three robots for a U-trajectory of the leader robot. In figure 4.22 the trajectories for the distances based and segment curvature based approaches are presented, with color red for the leader, color green for the second robot an color blue for the third one.

The trajectory of the leader robot is well tracked by the follower ones, the adaptive speed depends on the value for the orientation velocity. We also applied the approach considering the distance between robots for 3, 5 and 11 robots transporting the hose, starting from the initial configurations shown in figure 4.23.

When defining the initial configurations our aim was to define curved hoses with minimal potential energy in order to get hoses in rest. When all

Figure 4.20: Hose advancing at robot's velocity of 1m/s.

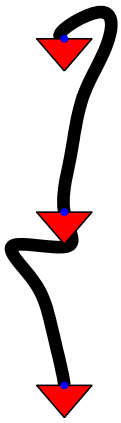


Figure 4.21: Hose advancing at robot's velocity of 0.2m/s.

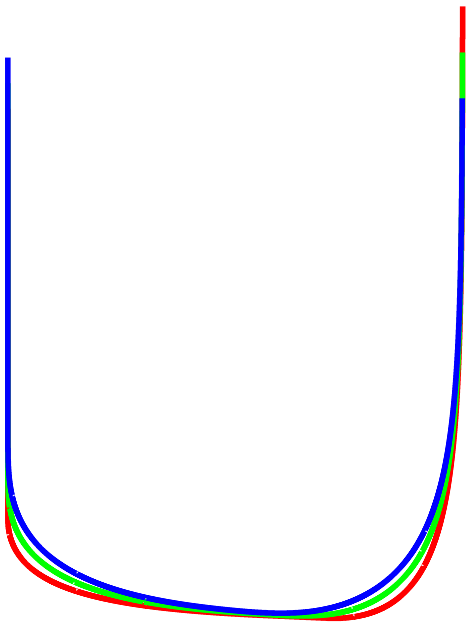


Figure 4.22: Robot's trajectories in the distance based approach

the robots reach the reference velocity the hose finally stabilizes its configuration. We present the hose configuration of the hoses for the rectilinear advance of the robots in the reference velocity at 0.2 m/s in figure 4.24.

The configurations of the hose when transporting it are presented in figure 4.25.

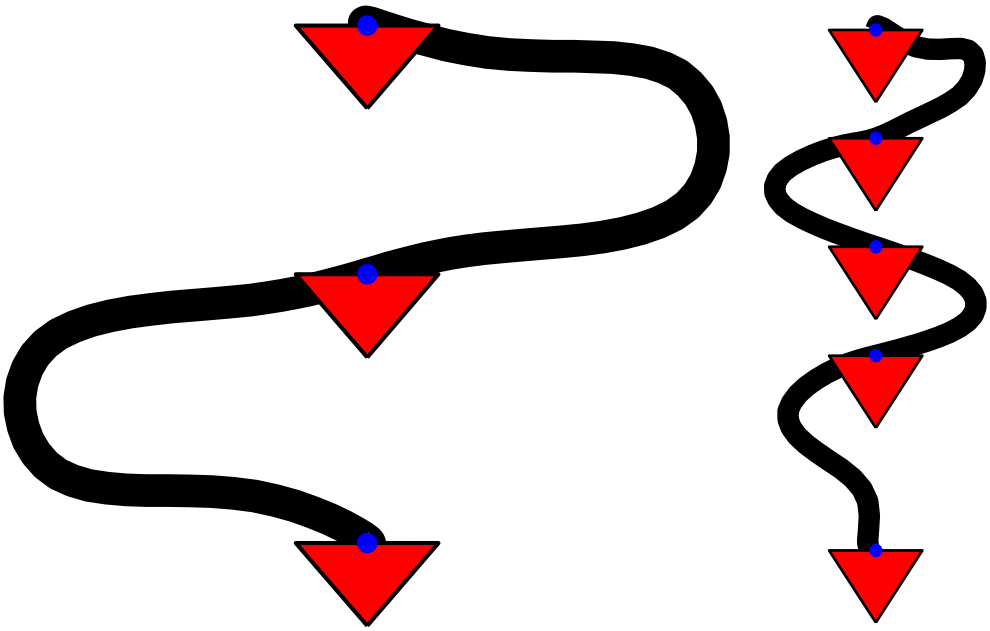
In figure 4.26 we present the velocities of the robots in the x and y axes.

If we do not limit the accelerations and the forces applied by the robots, the simulation does not have a good behavior, because we force the hoses more of what the simulation allows. The forces applied by the robots are presented in figure 4.27.

After have applied the distance based approach for the transport of the hose, we apply the approach based on the hose segments curvature, the trajectory of the robots is presented in figure 4.28.

The velocities applied by robots is presented in figure 4.29. In the segment curvature based approach the velocities for the leader robot are the sames as for the distance based approach.

In the case of robot 2, the velocities are similar for both approaches, unless some accelerations and accelerations are applied until a stable form of



(a) (b)

Figure 4.23: Hose initial configurations

the hose segment is reached.

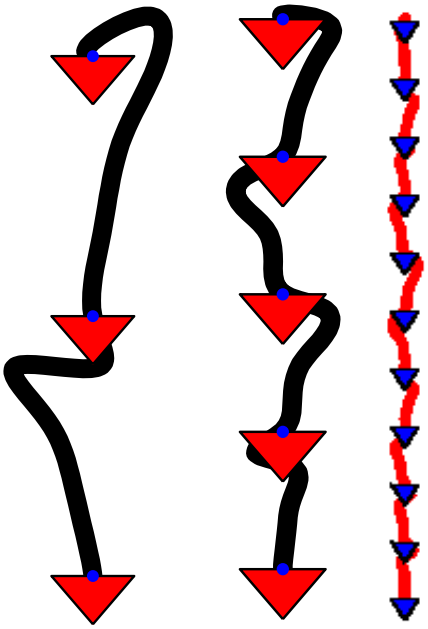
In the case of robot 3, the velocities are similar for both approaches, and as for the robot 2 the differences are in the accelerations and accelerations the robots apply until a stable form of the hose segment is reached.

In figure 4.30 the trajectories in the distance based approach and segment curvature based approach are presented together.

4.4.2.2 Configuration trajectory generation

Now we aim to obtain the transport velocities of robots from a given trajectory of the hose; in other words, from the sequence of configurations of the hose, h_k, we want to get the desired velocities of the robots $\dot{R}$ that make the hose follows the given trajectory.

As we saw in section 4.3.2.1, the velocities of the control points are determined by the following proportional control law:



(a) (b) (c)

Figure 4.24: Hose rectilinear advance

$$\left\|\bar{\mathbf{p}}_{\mathbf{i}}\right\| = k \left\|\overline{\phi_i(\xi) - \mathbf{p}}\right\| \tag{4.46}$$

In order to simulate this approach we select a sequence of the hose configurations, h_k, in the trajectory of the distance based approach done in section 4.4.2.1. We select 100 configurations of the hose uniformly distributed along the trajectory of the hose, and a speed reference,v_{ref}, of $0.2m/s$. We have selected the same speed as for distance based in order to compare the results of both approaches. The maximum force the robot may apply is defined as $4N$.

In figure 4.31 we present the trajectoy of the robots, where according to the criteria used until now, the leader robot is presented in red, the 2nd robot in green and the third robot in blue. In this approach the trajectories of 2nd and 3rd robot are much more closed to the leader robot, this is due to the fact that not only the positions of the robots are considered but also the whole hose curve, then the trajectory is perceived as a continuity and not as the robots independents velocities or taking into account the indirect

connection done by an external object.

As in the other simulations, we take control decisions at $30ms$ rate, although the time-step of the simuation is 1ms, in order to follow a control proccess that take images from a video camera at a frequency of $30ms$ and letting $30ms$. to the proccessing task.. The velocities on the robots are presented in figure 4.32. In this simulation not the maximum accelerations or velocities where defined but the forces, because the velocities and acclerations of the robots are obtained from the interaction of forces in the hose-robots system; Nevertheless the maximum velocity and acceleration are obey by the system trhough the use of maximum forces for robots.

The forces applied by robots are presented in figure 4.34, where only the 2nd robot reachs the maximum force.

4.5 Experimentation

We experimented the transport of a hose by three robots follwoing the hose segment curvature based approach presented in section 4.3.2.2. The image processing is done from the images taken by a camera placed in a nearly zenital position that observes the system, extracting the reflectance based on the dicromatic reflectance model. The purpose of the segmentation is to obtain the robots positions and the diverse parts of the hose. The control process is centralized, and the communication is done via radio-modems over the same channel (channel commuting had given so much problems). The heuristic control based on the hose segments curvature, consist on determining if the hose is stretched in relation with the distance between robots and the width of the involvement rectangle of the segmented hose piece.

4.5.0.3 Experiment statements

The transport of a hose of 2 meters in length has to be done in straight line by three robots attached to the hose. The points of contact between robots and hose must be fixed but allowing the hose to freely rotate over the robots. The leader robot has to be attached at the starting end of the hose, the second robot at exactly the middle and the third robot at the final end.

Starting conditions The initial configuration of the system may be any configuration that keep the following conditions:

- The leader robot in front, more advanced that the rest of robots, and the second robot more advanced than the third robot.
- The orientation of the robots must be closed each other.
- The hose can be bended but not enough to disturb the movements of the robots due to the forces that the hose potential energy configuration exert to the robots.
- The area in which the robots will move must be obstacles cleared.

Objective The aim of the experiment is the transport of the hose by a set of robots in a straight line until the leader robot reaches a desired position or goes out of the field of view of the camera. While the end of the experiment is not reached, the system must keep the following restrictions:

- The leader direction must be followed by the rest of robots.
- Robots must avoid the formations of loops, trying to maintain an enough separation each other.
- Robots must avoid the excessive stretched of the hose that produces dragging between robots, by trying to be close enough.

Control The robots control has to try to keep the straight line formation of them, taking into account the following conditions:

- The orientation of the leader robot will be manually controlled, but the speed will be autonomously controlled.
- Second and third robot will be autonomously controlled.
- The control process will be centralized.

Perception The perception of the experiment must be centralized by a single fixed video camera in a high position. The video camera has to be placed at approximately 2.5 meters high and has to be pointed to the floor covering an area over $2m^2$.

4.5.0.4 Hardware

A configuration of the hose-robots system is presented in figure 4.35, with a robot at each end of the hose and the remaining robot in the middle of the hose.

Robot We used a set of three SR1-robots [1] for the transport of the hose, at which we attached a revolving platform for grasping the black wire of 1 cm. of diameter and 300grs./m of density. In figure 4.36 a picture of the SR1 robot is presented. The SR1 robot poses a BasicX24P micro-controller of 8-bytes developed by Netmedia, Inc., that allows the control of its two wheel at the back, attached to two independents servomotors; the SR1 robot has a third free wheel without traction at the front.

The bearing platform is composed by a metallic square bolted to the roof of the SR1, with a revolving grasp over it, as shown in figure 4.37. We have painted the robots in blue in order to avoid the segmentation process errors due to the similarity of the robots color with the color of the laboratory in which we usually do experiments. The SR1 allows the communication by a radio modem that sends and receives signals at 19.200 bauds within a range of 50 m. The relative orientations between robots is estimated by a digital compass in every robot, that must be previously calibrated.

Hose We have used as the hose, a three-phase electrical black wire of $1cm$ width and $1m$ long. A picture of the wire is presented in figure 4.38.

Video camera The usb video-camera used for the perception is a Philips SPC 900NC with 1.3 MP, and it was placed on the ceiling at 2.5 meters high. In figure 4.35 an image taken by the camera over the hose-robots system is presented.

4.5.0.5 Software

The control of the system is done by a master program in Matlab, which makes the image processing and determines the movements the robots must do. Then, via a usb radio-modem the commands are send as text to the BaiscX24 micro-controller of the robots. The robots program is written in the BasicX language, for which a programming framework exist in Microsoft Windows. BasicX is a sub-language of Visual Basic.

In figure 4.39 the communication diagram is presented, for the master-slaves architecture of the system.

4.5.1 Experiment description

The hose curvature limits, according to section 4.3.2.2, are defined as $\underline{c} = 0.15$ and $\bar{c} = 0.30$. The velocity levels are defined as $w_1 = 0m/s$, $w_2 = 0.10m/s$ and $w_3 = 0.20m/s$.

4.5.1.1 Perception

The centralized perception is provided by a single camera that captures the scene with the three robots and the hose. The images acquired are segmented in search for the three robots and the hose. This segmentation process assumes several conditions on the environment's configuration:

1. Uniform floor of bright color, close to white.
2. Blue robots.
3. Non-blue, dark colored hose.
4. White, uniform illumination.

Segmentation will be composed of two separated processes, since we can profit from the different characteristics of the objects to look for (robots and hose) to use different techniques optimal for each one.

Robot's segmentation For the segmentation of the robots we are mainly interested in avoiding reflections and improving the colors in order to bring out the blue robots from the bright floor. This is achieved by means of a preprocessed step in which a Specular Free (SF) image [**?**] is created. Taking the Dichromatic Reflection Model (DRM) [**?**], images are the sum of two components: the diffuse component (which represents the chromacity of the observed surfaces) and the specular component (which represents the chromacity of the source of light which illuminates the scene). This model implies that the pixels with the reflections we want to avoid have a specular component. A Specular Free image is, then, an image geometrically identical to the original one but with its specular component removed. Several algorithms have been proposed in the literature for computing SF images

[?, ?, 69]. One important step in those algorithms is the estimation of the chromacity of the source of illumination. As we already know this chromacity, we have used a custom algorithm to obtain a SF image adapted to our needs.

Using a white light source, in this algorithm we profit from the characteristics of the RGB cube, as reflections will be very close to the axis (0,0,0) (1,1,1) of the RGB space while diffuse components will move away from it and closer to the pure color axes. This property is used to reduce the intensity of the specular pixels and improve the intensity of the diffuse ones proportionally to their distance with the axis (0,0,0)(1,1,1). Given an input RGB image $X = \{x(i,j)\}$, where $x(i,j) = \{r_{ij}, g_{ij}, b_{ij}\}$, an intensity image $I = \{d(i,j)\}$ is computed as

$$d(i,j) = max\{r_{ij}, g_{ij}, b_{ij}\} - min\{r_{ij}, g_{ij}, b_{ij}\} \tag{4.47}$$

The RGB image is then transformed to HSV space and its intensity channel is replaced with the computed intensity image I. This HSV image is then transformed back to RGB space. The result of this process can be appreciated in figure 4.40. Since we are looking for blue robots, they can be easily found in the SF image looking for the regions with highest intensities in the B channel.

Hose's segmentation The segmentation of the hose profits from the configuration of the environment, in which the hose will be a dark object over a bright floor. Given the original RGB frame and the regions obtained from the robot's segmentation, hose's segmentation is performed following a four step process:

1. The frame is skeletonized.
2. Label obtained regions.
3. Discard regions with very few pixels.
4. Discard regions that do not connect two regions containing a robot.

Each region obtained after this process is considered a segment of the hose.

In summary, the outputs of the visual perception system are the rectangular regions $R = \{R_1, ..., R_n\}$ of the image in which the n robots are located

Algorithm 4.5 Perception step of the robot-hose transportation system.

For each step k, given an input RGB image X_k and the regions $R_{k-1} = \{R_{k-1_1}, ..., R_{k-1_n}\}$ containing the n robots in the previous step:

1. Define a region of interest (ROI_k) in X_k using R_{k-1}.
2. Segment the robots:
   (a) Compute the intensity image I_k for the ROI_k (equation 4.47).
   (b) Transform ROI_k to HSV, replace intensity channel with I_k, transform back to RGB, obtaining an image SF_k.
   (c) Obtain new regions $R_k = \{R_{k_1}, ..., R_{k_n}\}$ as the regions with highest intensity in the B channel of SF_k and close to R_{k-1}.
3. Segment the hose:
   (a) Skeletonize ROI_k.
   (b) Discard small and isolated regions.
   (c) Discard regions not connecting two regions in R_k.
   (d) Return resulting $S_k = \{S_{k_1}, ..., S_{k_{n-1}}\}$ regions as detected hose segments.

(robot position p_i will be the centroid of that region) and several collections of points $S = \{S_1, ..., S_{n-1}\}$ corresponding to detected hose segments. Those hose segments are checked to make sure that they connect two robots. The segments that do not connect two robots are discarded. Each step, the regions of interest in which robots and segments where located are used to speed up the search in the new acquired images. The full perception process is outlined in algorithm 4.5.

Conditions of the visual sensing

1. Floor of an uniform bright color, with the tiled floor close to the white color.
2. A completely black hose, of 2 meters long and 1 cm. of diameter.

3. Robots of blue color.

4. Illumination of white color, uniformly distributed over the scene.

4.5.2 Results

A result obtained from the experimentation with the SR1 robots is presented in figure 4.41. The hose is shown in red color for a better illustration although its real color is black, and each robot is marked with a label containing its state. The state of a robot may be advancing, stretching or shrinking. The leader robot has the advancing state at every moment, while the followers only when the hose segment between them and their precedent robot is neither so much stretching nor so much shrinking, according tho the values defined in section 4.5.1 ($\underline{c} = 0.15$, $\bar{c} = 0.30$). When the robot is in stretching state the hose segment between it and its previous robot is not enough stretched ($c < \underline{c}$) and the robot reduce its velocity respect to the precedent robot in order to allow a stretching of the hose segment. When the robot is in shrinking state the hose segment between it and its previous robot is so much stretched ($c > \bar{c}$) and the robot increase its velocity respect to the precedent robot in order to allow an increasing of the curvature of the hose segment.

Te sequences of image in figure 4.41 shows the following situations in the transport sequence:

- Figure 4.41a: The starting position of the experiment is presented with the leader robot advancing at cruise speed ($v_1 = w_1$) at front of the hose while 2nd and 3th robots are waiting ($v_2 = w_0$, $v_3 = w_0$) until their hose segments are stretching enough ($c_1 = 1$, $c_2 = 0.74$).

- Figure 4.41b: After the leader robot has advancing enough, the first hose segment is below the maximum limit $\bar{c}$ ($c_1 = 0.27$) so the 2nd robot starts advancing at cruise speed ($v_2 = w_1$). Third robot is still waiting ($v_3 = w_0$) the stretching of its hose segment ($c_2 = 0.67$).

- Figure 4.41c: First hose segment is so much stretched ($c_1 = 0.11$), so the 2nd robot reduce its speed ($v_2 = w_0$) in order to allow the bending of the segment. Third robot is still waiting the stretching of the hose segment ($v_2 = w_0$, $c_2 = 0.6$).

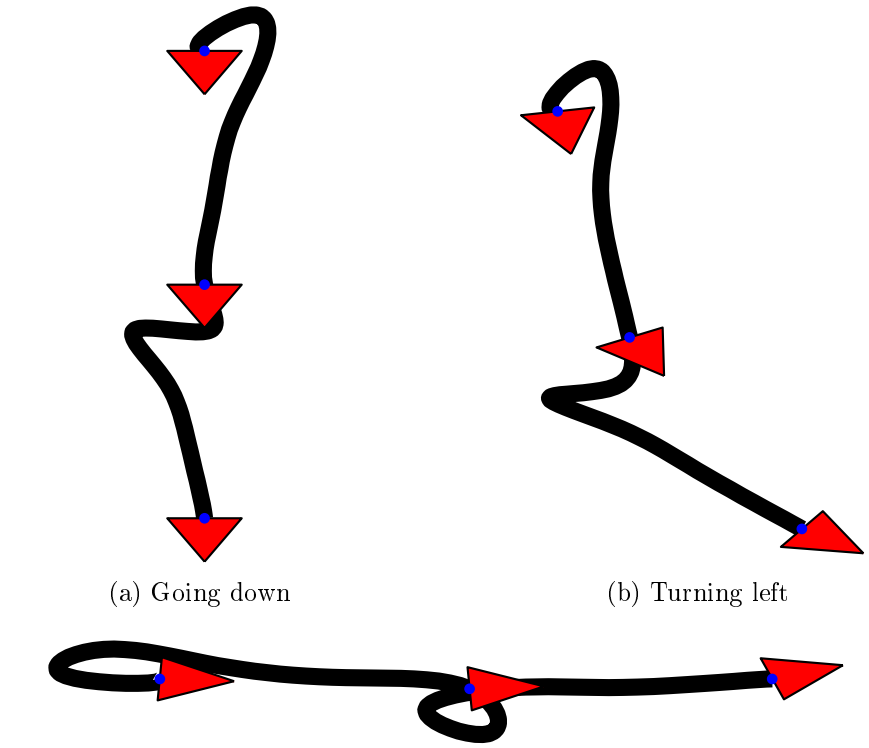
- Figure 4.41d: First segment has curved enough ($c_1 = 0.24$) so the second robot start advancing at cruise speed ($v_2 = w_1$). Second segment has stretched enough ($c_2 = 0.28$), so the third robot start advancing at cruise speed ($v_3 = w_1$).

- Figure 4.41e: First segment curvature ($c_1 = 0.20$) continues between the acceptability range, so the second robot maintain the cruise speed ($v_2 = w_1$). Second segment has increased its curvature ($c_2 = 0.32$) more than maximum limit $\bar{c}$, so the third robot reduces its speed ($v_3 = w_0$) in order to allow a stretching of the hose.

- Figure 4.41f: First segment still maintain its curvature ($c_1 = 0.27$) between the acceptability range, so the second robot maintain the cruise speed ($v_2 = w_1$). Second segment has stretched to much ($c_2 = 0.15$), so third robot increase its speed ($v_3 = w_2$) in order to allow a stretching of the segment.

4.6 Conclusions and future work

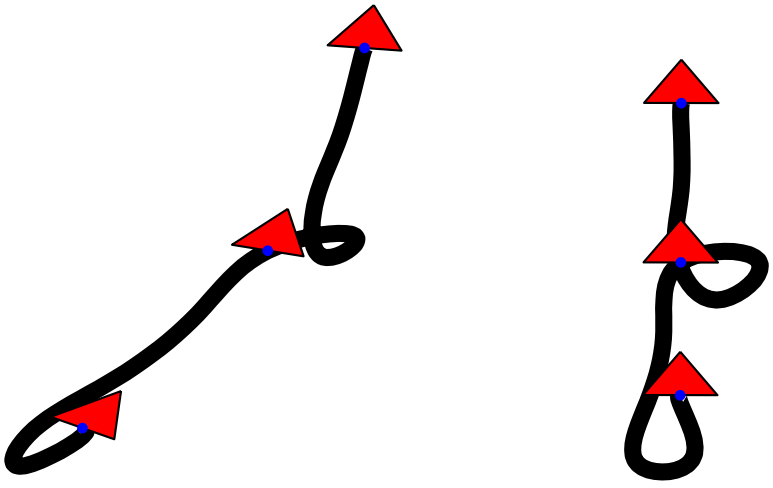
We have obtained the forces expression that robots must apply in the contact points with the hose, wire or every Hemos obtenido una expresión para las fuerzas que deben ser aplicadas en los puntos de contacto de un grupo de robots con con una manguera o un cable un objeto elástico unidimensional para alcanzar una configuración deseada del objeto. En este momento del programa de desarrollo, el siguiente paso es construir simulaciones convenientes del ojbeto unidimensional, con el objetivo de testear las expresiones de control obtenidas. El siguiente paso es el desarollo de algunos tipos control distribuido, en los que el conocimiento sobre el estado global del sistema pueda ser obtenido en base al intercambio de información. Un procedimiento de identificación para los parámetros dinámicos de la manguera será necesario para obtener implenetaciones del sistema en tiempo real, en las que los robots deben aprender las características del objeto con el que están tratando. Son necesarios algoritmos de path planning para determinar los efectos de obstaculos en el desarrollo de la tarea. Finalmenet, deberemos trabajar sobre la configuración física los robots.



(a) Going down

(b) Turning left

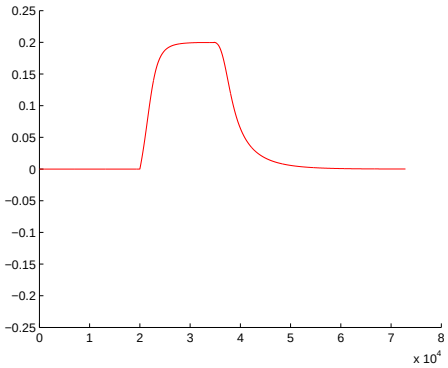
(c) Going right



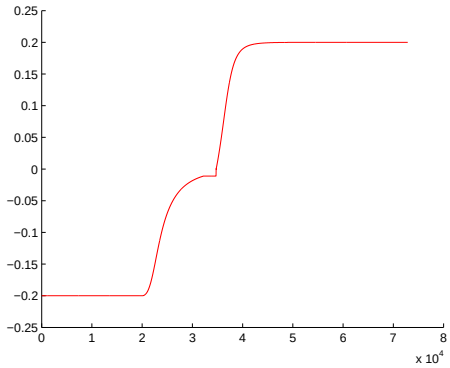
(d) Turning left

(e) Going up

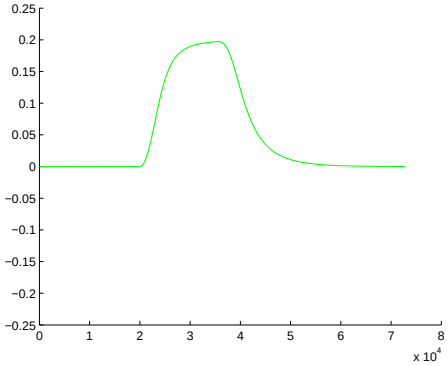
Figure 4.25: Hose configurations



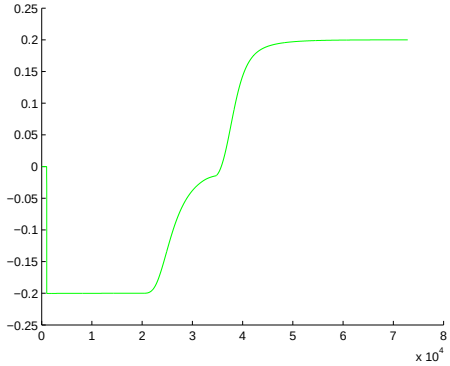
(a) x-axis for leader robot



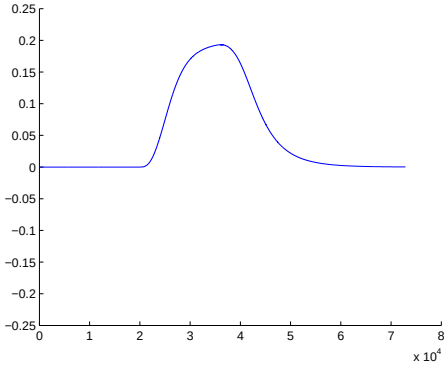
(b) y-axis for leader robot



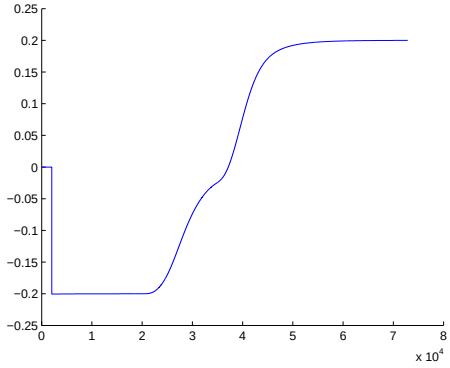
(c) x-axis for second robot



(d) y-axis for second robot

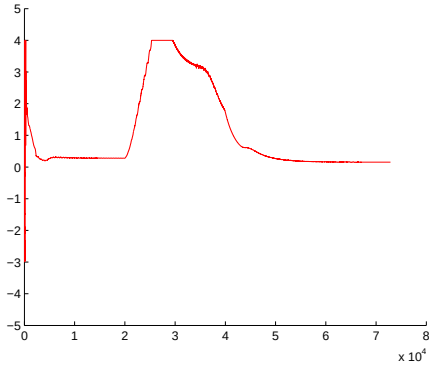


(e) x-axis for third robot

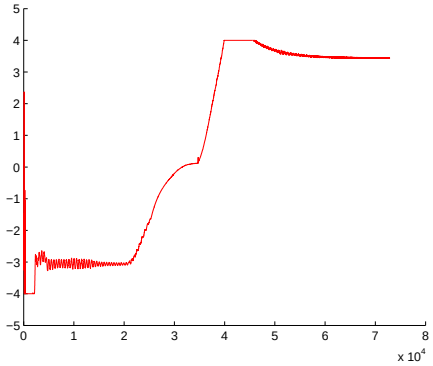


(f) y-axis for third robot

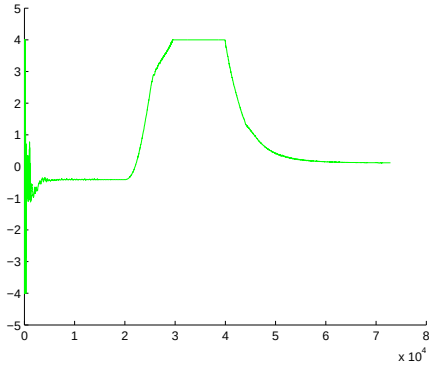
Figure 4.26: Robots velocities for the distance based approach



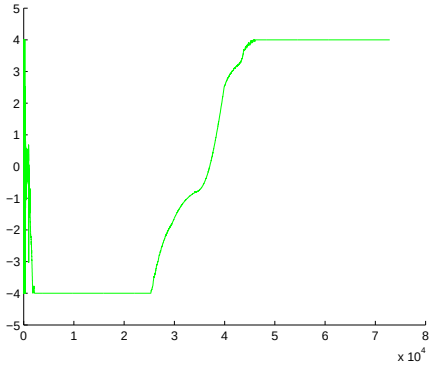
(a) x-axis for leader robot



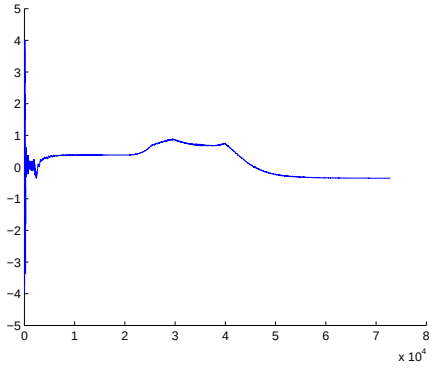
(b) y-axis for leader robot



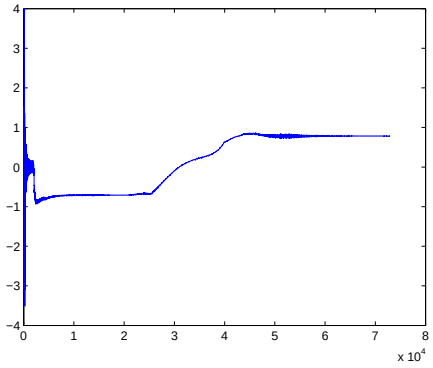
(c) x-axis for second robot



(d) y-axis for second robot



(e) x-axis for third robot



(f) y-axis for third robot

Figure 4.27: Forces applied by robots for the distance based approach

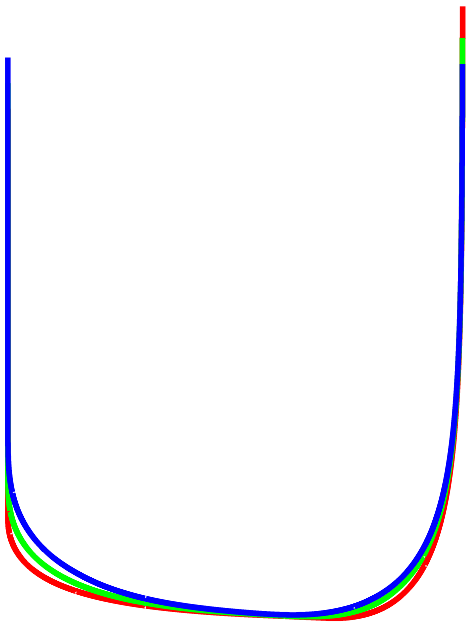
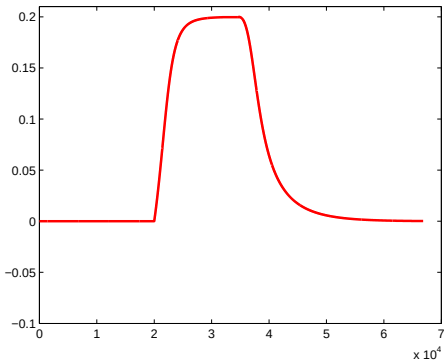
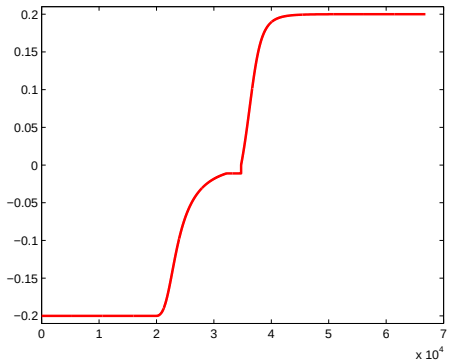


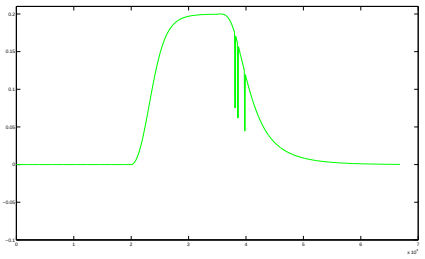
Figure 4.28: Robots trajectories in the segment curvature based approach



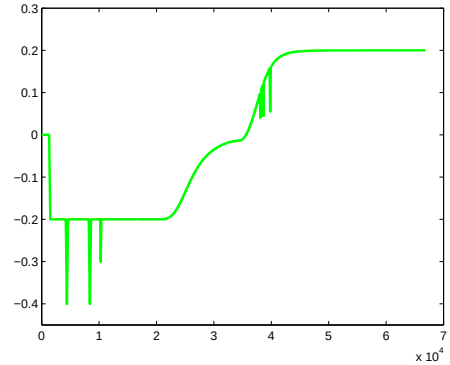
(a) x-axis for the leader robot



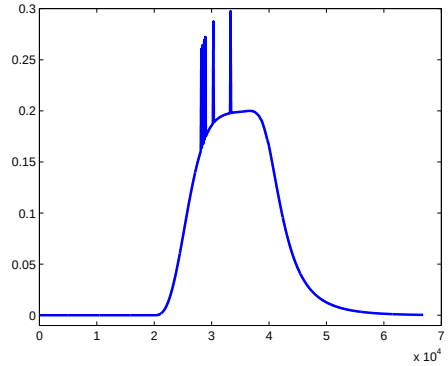
(b) y-axis for the leader robot



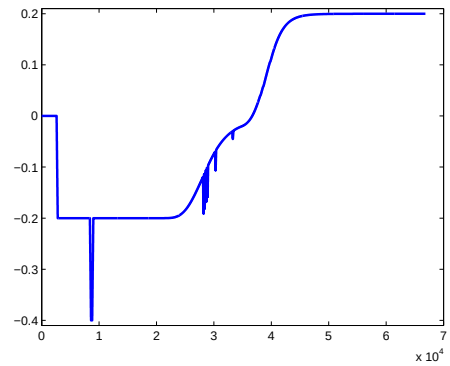
(c) x-axis for second robot



(d) y-axis for second robot

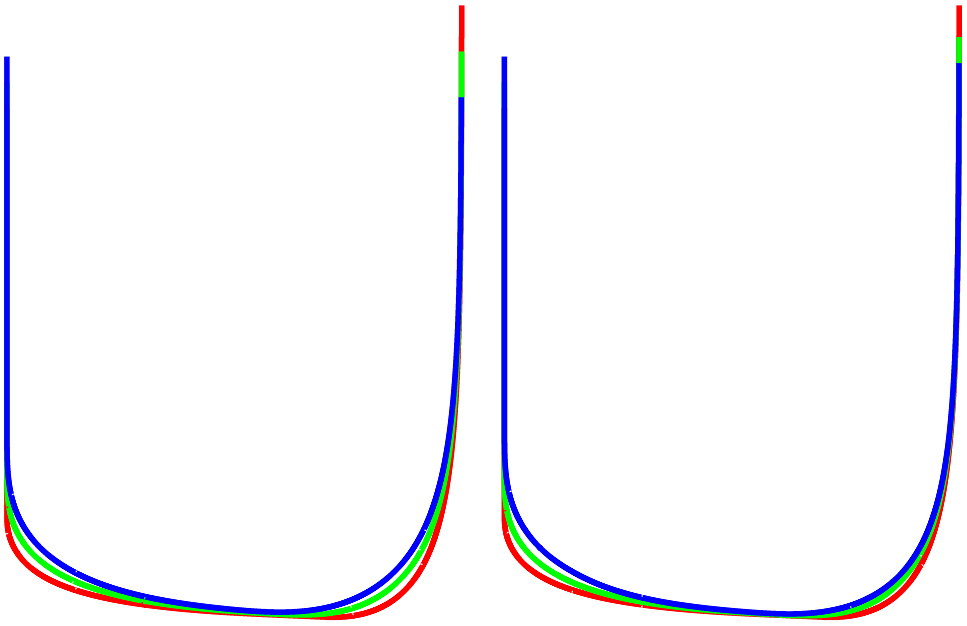


(e) x-axis for third robot



(f) y-axis for third robot

Figure 4.29: Robots velocities for the segment based approach



(a) Distance based (b) Segment curvature based

Figure 4.30: Robot's trajectories in the distance based approach

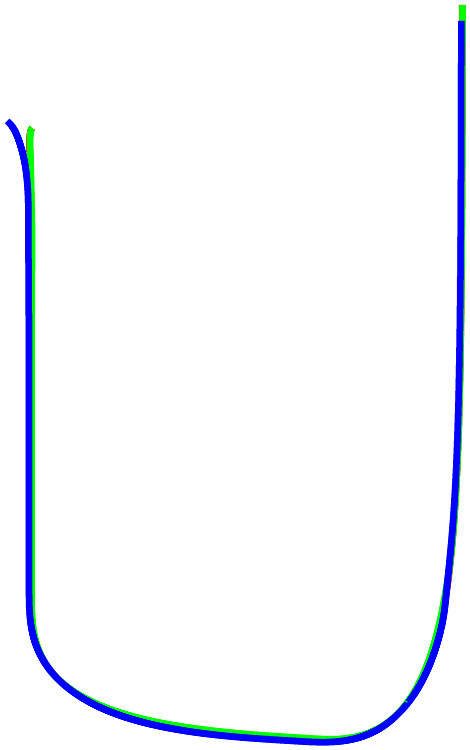
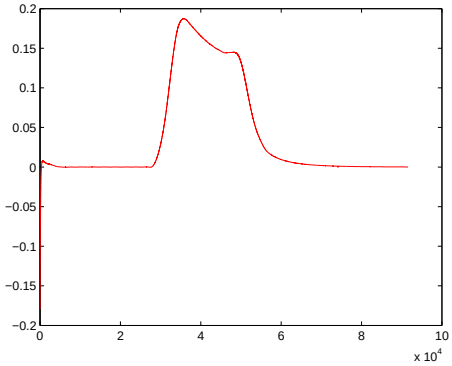
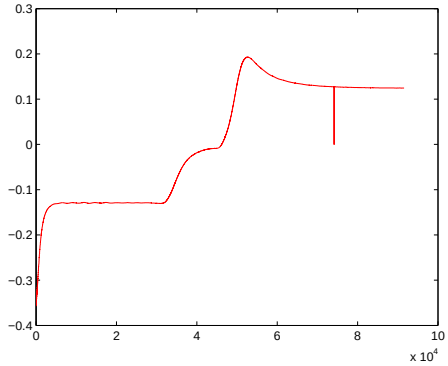


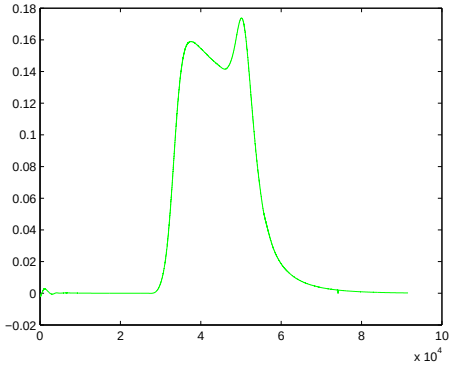
Figure 4.31: Robots trajectory in the configuration trajectory approach



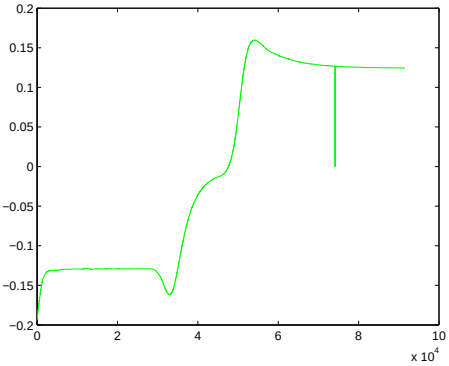
(a) x-velocity for the leader robot



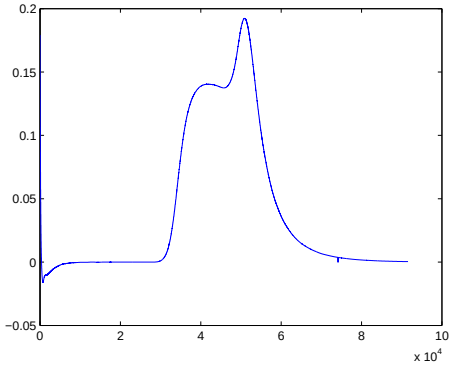
(b) y-velocity for the ladder robot



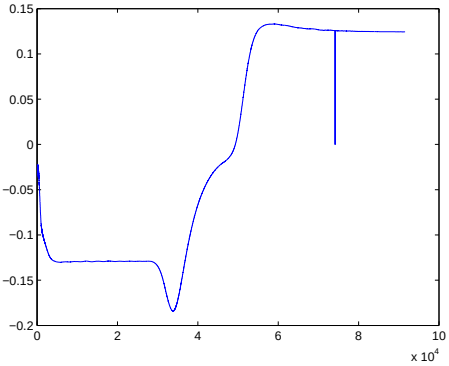
(c) x-velocity for the second robot



(d) y-velocity for the second robot

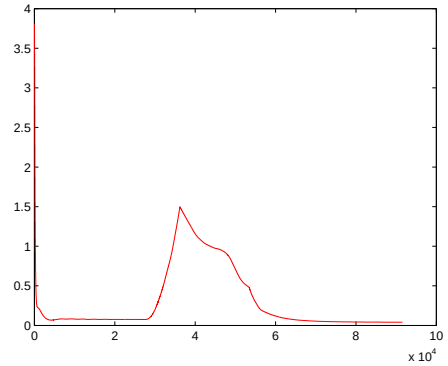


(e) x-velocity for the third robot

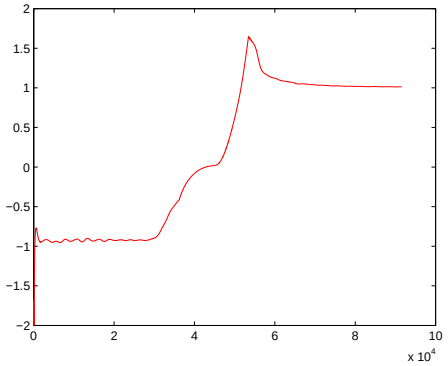


(f) y-velocity for the third robot

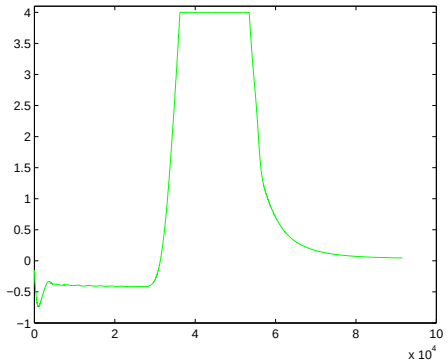
Figure 4.32: Robots velocities in the configuration trajectory approach



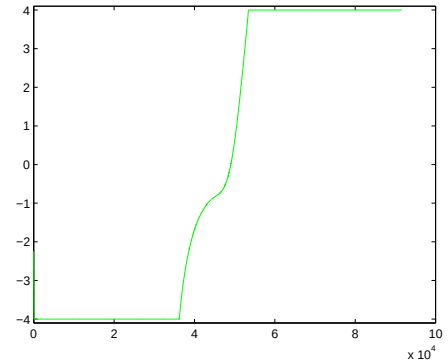
(a) x-axis for leader robot



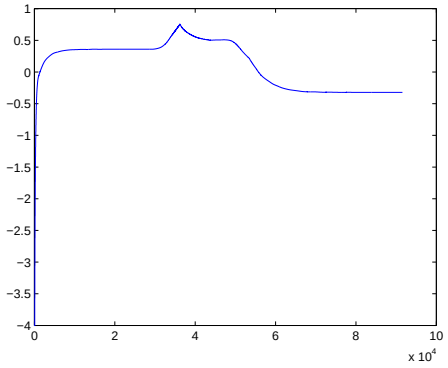
(b) y-axis for leader robot



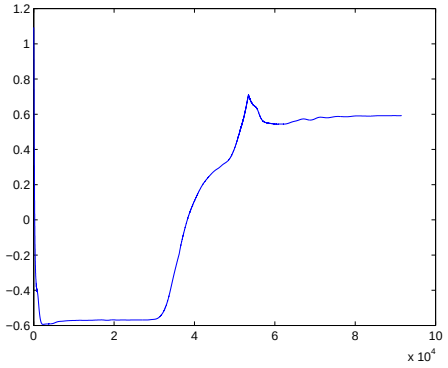
(c) x-axis for second robot



(d) y-axis for second robot



(e) x-axis for third robot



(f) y-axis for third robot

Figure 4.33

Figure 4.34: Forces applied by robots for the configuration based approach

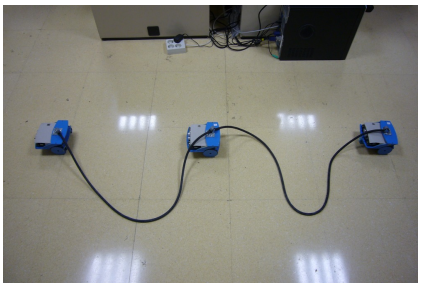


Figure 4.35: Hose-robots physic system

Figure 4.36: SR1 robot

Figure 4.37: Blue SR1-robot with a bearing platform

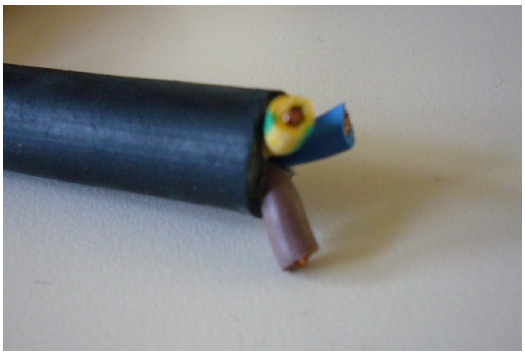


Figure 4.38: Wire of 1m long and 1cm width as the hose

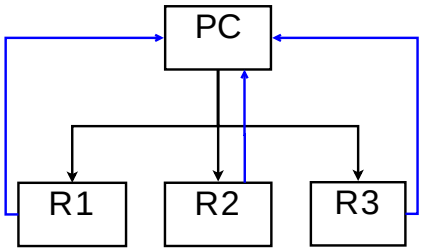
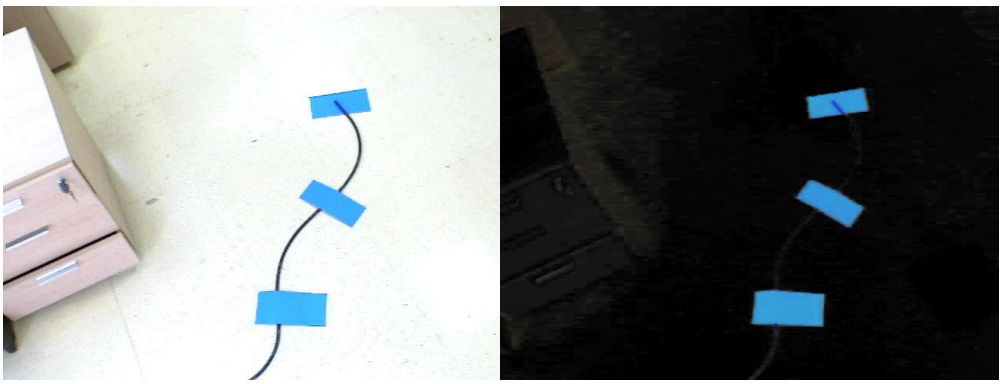
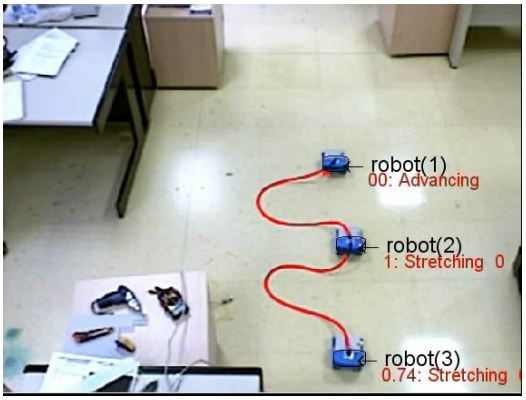


Figure 4.39: Communication between robots and the central process

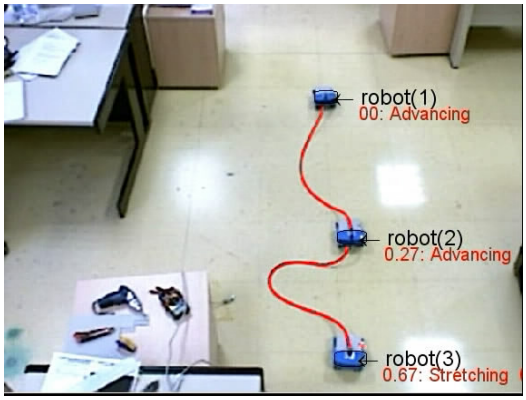


(a) Original (b) Resulting SF images

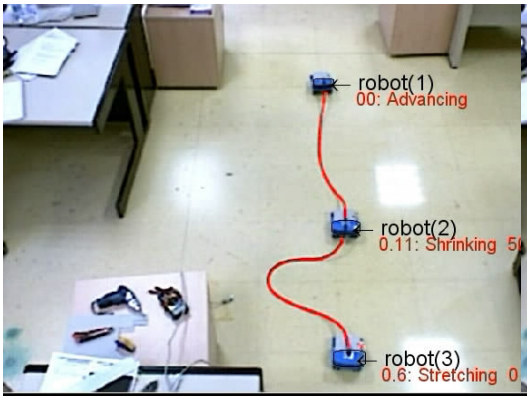
Figure 4.40: Specular Free image computation



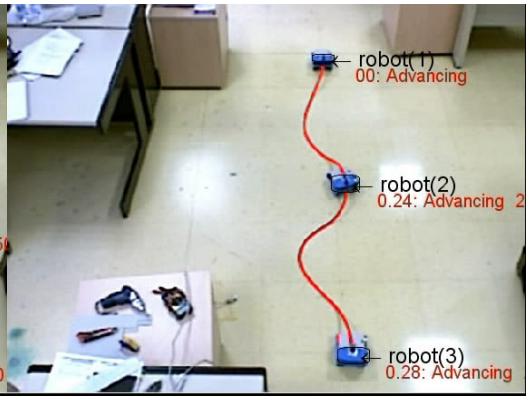
(a) Starting position



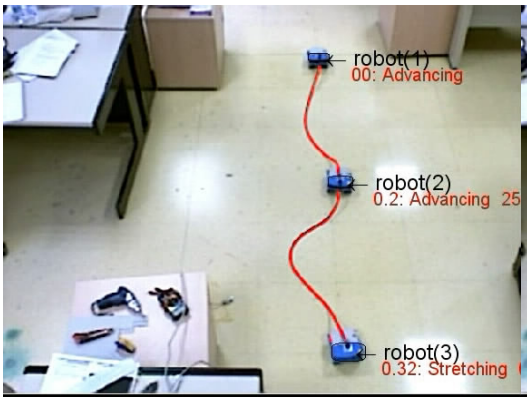
(b) Robot 2 advancing and robot 3 shrinking



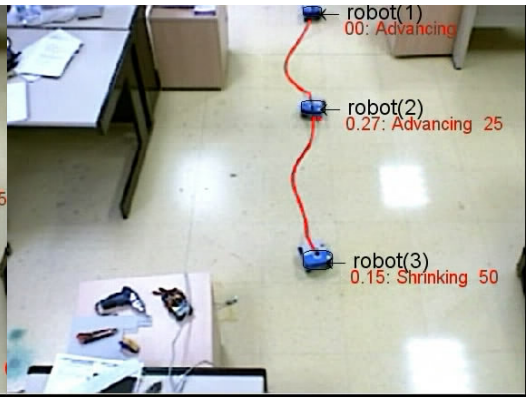
(c) Robot 2 stretching and robot 3 stretching



(d) Robots 2 advancing and robot 3 advancing



(e) Robot 2 shrinking and robot 3 advancing



(f) Robots 2 stretching and robot 3 advancing

Figure 4.41: Snap-shoots of the experimentation

Appendix A

Interpolating clamped B-spline

The interpolating process consists on the construction of a curve that passes trough a sequence of preset points in 2D or 3D. Given a set of points $D = \{d_0, ..., d_l\}$, known as interpolating points, there exist infinite curves that pass trough these points. In our case we use a clamped B-spline cubic curve and we define the interpolating algorithm for this kind of curves.

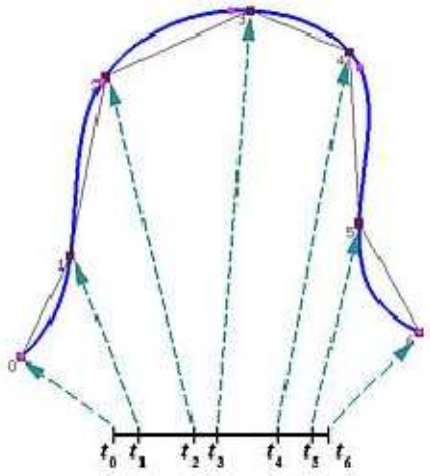


Figure A.1: Interpolating cubic B-spline curve

We must take into account that the B-spline curves keep the following condition:

$$m = n + p + 1$$

Being $(n+1)$ the number of control points, m the number of knots and p the curve degree.

A clamped cubic B-spline curve ($p = 3$) has a knots vector:

$$U = \left(\underbrace{u_0, \ldots, u_3}_{4}, \underbrace{u_4, \ldots, u_n}_{n-3}, \underbrace{u_{n+1}, \ldots, u_{n+4}}_{4} \right)$$

where, if the domain of the curve is the interval $[0, 1]$, the knots vector of the clamped B-spline curve is:

$$U = \left(\underbrace{0, 0, 0, 0}_{4}, \underbrace{u_4, \ldots, u_n}_{n-3}, \underbrace{1, 1, 1, 1}_{4} \right)$$

The curve is exclusively defined from $q(u_3)$ to $q(u_{n+1})$ and, for these values of the control parameter u, the curve interpolates the first and last control points. In other words,

$$\begin{aligned} q(u_3) &= p_0 \\ q(u_{n+1}) &= p_n \end{aligned}$$

The first step of the interpolating algorithm consists of selecting the curve set of knots. The most simple knots vector is a non periodic and uniform. If the clamped knots vector is uniform, the values of its knots are computed in the following way:

$$\begin{cases} u_0 = u_1 = u_2 = u_3 = 0 & \\ u_i = \frac{i-3}{n-2} & 4 \leq i \leq n \\ u_{n+1} = u_{n+2} = u_{n+3} = u_{n+4} = 1 & \end{cases}$$

where the difference between two consecutive knots is always constant; in other words:

$$u_{i+1} - u_i = \frac{1}{n-2}$$

If the interpolating points are not uniformly distributed, these knots vector may get not desired results as peaks, protuberances and loops. This is as consequence of the oscillations of the B-spline, due to the fact that the uniform knots vector does not take into account the geometry of the interpolating points.

Nevertheless, it does not matter which the knots vector is, the curve should interpolate the interpolating points. In other words, the following restrictions must be accomplished:

$$\begin{aligned} d_0 &= q\left(u_3\right) \\ d_1 &= q\left(u_4\right) \\ &\dots \\ d_{n-2} &= q\left(u_{n+1}\right) \end{aligned}$$

where the k-th point is:

$$d_k = q\left(u_{k+3}\right) = \sum_{i=0}^{n} p_i \cdot N_{i,3}\left(u_{k+3}\right) \qquad 0 \leq k \leq n-2 = l$$

being

$$u_0, u_1, u_2, u_3 = 0 \qquad u_{n+1}, u_{n+2}, u_{n+3}, u_{n+4} = 1$$

The relation between l and n is given by:

$$l + 1 = n - 1$$

where $(l+1)$ is the number of interpolating knots and $(n+1)$ is the number of control points.

When computing these points, $(n+1)$ equations are needed. However, we only have $(l + 1 = n - 1)$ equations. So, two more restrictions are given by repeating the first and last control points of the curve:

$$p_0 = p_1 \text{ y } p_{n-1} = p_n$$

Given the necessaries restrictions, the equations system is resolved in an effective and efficient way, as we continue explaining. It is remarkable that in the interval u_{k+3}, do not exist more than four not nulls base functions:

$$N_{k,3}\left(u\right) \quad N_{k+1,3}\left(u\right) \quad N_{k+2,3}\left(u\right) \quad N_{k+3,3}\left(u\right)$$

If we examine the expansion in the recursion tree of the Cox-de-Boor algorithm, we can compute efficiently the values of these functions.

For instance, the three first base functions take not nulls values while the last base function is zero. The values of these base functions are given by:

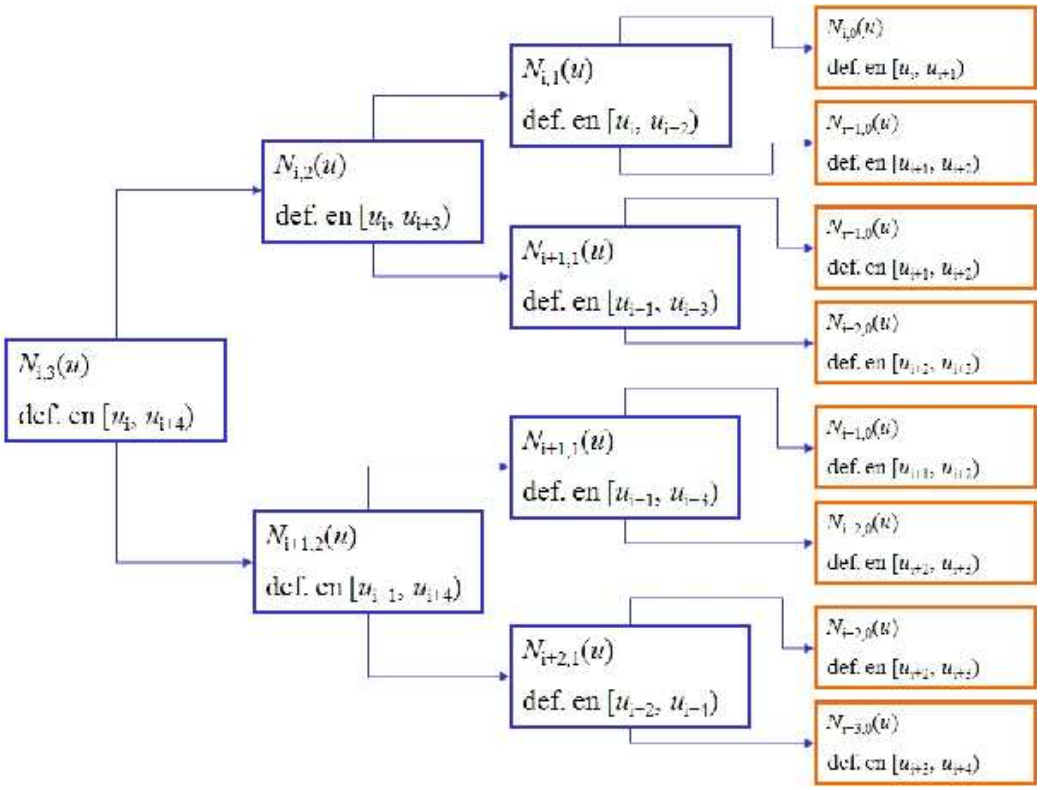


Figure A.2: Recursion tree of the Cox-de-Boor algorithm

$$N_{k,3}\left(u_{k+3}\right) = \frac{\left(u_{k+4} - u_{k+3}\right)^2}{\left(u_{k+4} - u_{k+1}\right)\left(u_{k+4} - u_{k+2}\right)} = \alpha_k$$

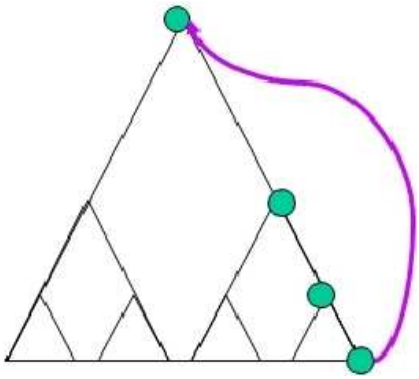


Figure A.3

$$N_{k+1,3}\left(u_{k+3}\right)=\frac{\left(u_{k+3}-u_{k+1}\right)\left(u_{k+4}-u_{k+3}\right)}{\left(u_{k+4}-u_{k+1}\right)\left(u_{k+4}-u_{k+2}\right)}+$$
$$+\frac{\left(u_{k+5}-u_{k+3}\right)\left(u_{k+3}-u_{k+2}\right)}{\left(u_{k+5}-u_{k+2}\right)\left(u_{k+4}-u_{k+2}\right)}=\beta_k$$

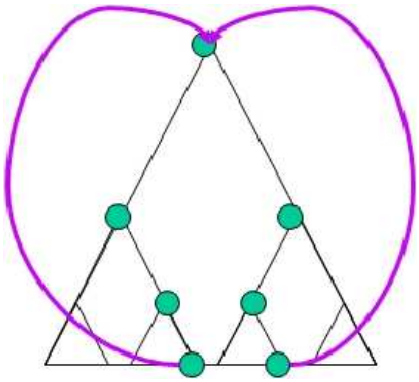


Figure A.4

$$N_{k+2,3}\left(u_{k+3}\right)=\frac{\left(u_{k+3}-u_{k+2}\right)^2}{\left(u_{k+4}-u_{k+2}\right)\left(u_{k+5}-u_{k+2}\right)}=\gamma_k$$

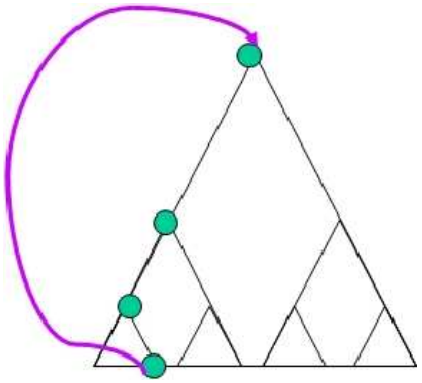


Figure A.5

So, the interpolating points are given by the following polynomial:

$$\begin{gathered} d_k = q\left(u_{k+3}\right) = N_{k,3}\left(u_{k+3}\right) \cdot p_k + N_{k+1,3}\left(u_{k+3}\right) \cdot \\ \cdot p_{k+1} + N_{k+2,3}\left(u_{k+3}\right) \cdot p_{k+2} = \\ = \alpha_k \cdot p_k + \beta_k \cdot p_{k+1} + \gamma_k \cdot p_{k+2} \end{gathered}$$

and the equations system can be expressed in a matricial way as:

$$\begin{bmatrix} \beta_0 & \gamma_0 & & & & & \\ \alpha_1 & \beta_1 & \gamma_1 & & & & \\ & & \cdots & & & & \\ & & & \alpha_{n-3} & \beta_{n-3} & \gamma_{n-3} & \\ & & & & \alpha_{n-2} & \beta_{n-2} & \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n-2} \\ p_{n-1} \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-3} \\ d_{n-2} \end{bmatrix}$$

being $\beta_0 = \beta_{n-2} = 1$.

The resolution of this equations system obtains the $(n-1)$ control points $\{p_1, \ldots, p_{n-1}\}$ of the B-spline curve. And the remaining two control points, as we have seen previously, are obtained by repeating the first and last control points. This way, given $(l+1)$ interpolating points, the clamped B-spline curve that passes trough every interpolating points in the preset order by the user, have the following$(n+1)$ control points :

$$\{p_0 = p_1, p_1, \ldots, p_{n-1}, p_n = p_{n-1}\}$$

Added to this equations system, that is formulated as matrices for the tri-diagonals systems where only the elements of the diagonal and its two

neighbors at left and right are nulls, we can find an efficient resoluter algorithm of lineal order ($O(n)$) [65].

Given the following tri-diagonal system of equations:

$$\begin{bmatrix} \beta_0 & \gamma_0 & & & & & \\ \alpha_1 & \beta_1 & \gamma_1 & & & & \\ & & \cdots & & & & \\ & & & \alpha_{n-3} & \beta_{n-3} & \gamma_{n-3} & \\ & & & & \alpha_{n-2} & \beta_{n-2} & \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{n-3} \\ X_{n-2} \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-3} \\ d_{n-2} \end{bmatrix}$$

we can use a two stepped algorithm that resolves this system.

The first step, known as forward step, transforms this tri-diagonal system in the following equations system:

$$\begin{bmatrix} 1 & \lambda_0 & & & & \\ & 1 & \lambda_1 & & & \\ & & \cdots & & & \\ & & & 1 & \lambda_{n-3} & \\ & & & & 1 & \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{n-3} \\ X_{n-2} \end{bmatrix} = \begin{bmatrix} \delta_0 \\ \delta_1 \\ \vdots \\ \delta_{n-3} \\ \delta_{n-2} \end{bmatrix}$$

In order to do so, calculations between the rows of the matrices or between multiples of this rows are done as follows:

$$\lambda_0 = \tfrac{\gamma_0}{\beta_0} \qquad \lambda_i = \tfrac{\gamma_i}{\beta_i - \alpha_i \lambda_{i-1}} \qquad i = 1, \ldots, n-3$$

$$\delta_0 = \tfrac{d_0}{\beta_0} \qquad \delta_i = \tfrac{d_i - \alpha_i \delta_{i-1}}{\beta_i - \alpha_i \lambda_{i-1}} \qquad i = 1, \ldots, n-2$$

The second step, known as backward step, works out the unknown quantities, X_i, by a cycle from backward to forward in the following way:

$$X_{n-2} = \delta_{n-2}$$

$$X_i = \delta_i - \lambda_i \cdot X_{i+1} \quad i = n-3, \ldots, 0$$

Finally, the set of control points is the set of the obtained variables, X_i, from the backward step:

$$p_0 = p_1 \qquad \underset{i=0,\ldots,n-2}{p_{i+1} = X_i} \qquad p_n = p_{n-1}$$

Bibliography

[1] Sr1 - multifuncional mobile robot.

[2] S. S. Antman. *Nonlinear Problems of Elasticity*. Springer-Verlag, 1995.

[3] Carl De Boor. *A Practical Guide to Splines*. Springer, August 1994.

[4] E. Cervera, F. Berry, and P. Martinet. Is 3d useful in stereo visual control? In *IEEE International Conference on Robotics and Automation, ICRA '02, Washington DC, USA*, volume 2, pages 1630–1635, May 2002.

[5] F. Chaumette, E. Malis, and S. Boudet. 2d 1/2 visual servoing with respect to a planar object. In *Workshop on New Trends In Image-Based Robot Servoing, IROS'97, Grenoble, France*, pages 45–52, September 1997.

[6] C. Colombo, E. Kruse, A.M. Sabatini, and P. Dario. Vision-based relative positioning through active fixation and contour tracking. In *Proceedings 2nd International Symp. on Intelligent Robotic Systems, SIRS'94 , Grenoble, France*, pages 319–325, July 1994.

[7] Peter Corke. *Visual Control of Robot Manipulators - A Review*, volume 7 of *Robotics and Automated Systems*, pages 1–31. World Scientific, 1993.

[8] P.I. Corke and S.A. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation*, 17(4):507–515, August 2001.

[9] Sony Corporation. Open-r sdk model information for ers-7, 2003.

[10] Eve Coste-Manière, Philippe Couvignou, and Pradeep K. Khosla. Visual servoing in the task-function framework: A contour following task. *Journal of Intelligent and Robotic Systems*, 12(1):1.–21, 1995.

[11] P.Y. Coulon and M. Nougaret. Use of a tv camera system in closed-loop position control of mechanisms. In A. Pugh, editor, *International Trends in Manufacturing Technology ROBOT VISION*, pages 117–127. IFS Publications, 1983.

[12] A. Cretual and F. Chaumette. Positioning a camera parallel to a plane using dynamic visual servoing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS '97, Grenoble, France*, volume 1, pages 43–48, September 1997.

[13] A. Cretual and F. Chaumette. Image-based visual servoing by integration of dynamic measurements. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'98, Louvain, Belgium*, volume 3, pages 1994–2001, May 1998.

[14] Richard J. Duro, Manuel Graña, and Javier de Lope. On the potential contributions of hybrid intelligent approaches to multicomponent robotic system development. *Information Sciences*. Accepted for publication.

[15] E. Enzio Malis. Visual servoing invariant to changes in camera intrinsic parameters. *Robotics and Automation, IEEE Transactions on*, 1:704–709, 2001.

[16] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8:313–326, 1992.

[17] Vincent Bandou Anne-Gaelle Allais Michel Perrier Ezio Malis, Patrick Rives. Active stereovision using invariant visual servoing. In *IROS*, 2006.

[18] J.T. Feddema and O.R. Mitchell. Vision-guided servoing with feature-based trajectory generation [for robots]. *IEEE Transactions on Robotics and Automation*, 5(5):691–.700, Oct 1989.

[19] N.M. Garcia and E. Malis. Preserving the continuity of visual servoing despite changing image features. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2004,*, volume 2, pages 1383–1388, September-October 2004.

[20] N. Garcia-Aracil, E. Malis, R. Aracil-Santonja, and C. Perez-Vidal. Continuous visual servoing despite the changes of visibility in image features. *IEEE Transactions on Robotics*, 21(6):1214–1220, Dec. 2005.

[21] C. Geschke. A robot task using visual tracking. *Robotics Today*, pages 39–43, Winter 1981.

[22] Mireille Grégoire and Elmar Schömer. Interactive simulation of one-dimensional flexible parts. *Computer-Aided Design*, 39(8):694–707, 2007.

[23] M. Sato F. Harashima H. Hashimoto, T. Kubota. Visual servo control of robotic manipulators based on artificial neural network. In *IEEE Int. Conf. on Industrial Electronics*, 1989.

[24] Greg Hager. The "xvision" system: A general purpose substrate for real-time vision-based robotics. In *Proceedings of the Workshop on Vision for Robotics*, pages 56–63, 1995.

[25] E. Hergenröther and P. Dhne. Real-time virtual cables based on kinematic simulation. In *Proceedings of the WSCG*, 2000.

[26] J. Hill and W.T. Park. Real-time control of a robot with a mobile camera. In *Proceedings of the 9th ISIR*, pages 233–246, Washington D.C., March 1979.

[27] N. Hogan. Impedance control - an approach to manipulation. i - theory. ii - implementation. iii - applications. *ASME Transactions Journal of Dynamic Systems and Measurement Control B*, 107:1–24, March 1985.

[28] K. Hosoda, K. Igarashi, and M. Asada. Adaptive hybrid visual servoing/force control in unknown environment. In *IEEE Int. Conf. on Intelligent Robots and Systems, Osaka, Japan*, pages 1097–1103, September 1996.

[29] K. Hosoda, K. Sakamoto, and M. Asada. Trajectory generation for obstacle avoidance of uncalibrated stereo visual servoing without 3d reconstruction. In *Proceedings of the International Conference on Intelligent Robots and Systems, IEEE/RSJ, IROS '95*, volume 3, pages 29–34, Washington, DC, USA, October 1995. IEEE Computer Society.

[30] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *Robotics and Automation, IEEE Transactions on*, 12(5):651–670, Oct 1996.

[31] A. G. Makhlin. Stability and sensitivity of servo vision systems. In *Proc 5th International Conference on Robot Vision and Sensory Controls - RoViSeC 5*, pages 79–89, October 1985.

[32] E. Malis, F. Chaumette, and S. Boudet. Multi-cameras visual servoing. In *IEEE International Conference on Robotics and Automation, ICRA '00, San Francisco, CA, USA*, volume 4, pages 3183–3188, April 2000.

[33] Ezio Malis. *Stability Analysis of Invariant Visual Servoing and Robustness to Parametric Uncertainties*. Springer Berlin/Heidelberg, 2003.

[34] P. Martinet and E. Cervera. Stacking jacobians properly in stereo visual servoing. In *IEEE International Conference on Robotics and Automation, ICRA'01, Seoul, Korea*, volume 1, pages 717–722, May 2001.

[35] N. Maru, H. Kase, S. Yamada, A. Nishikawa, and F. Miyazaki. Manipulator control by using servoing with the stereo vision. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems '93, IROS '93*, volume 3, pages 1866–1870, July 1993.

[36] Y. Mezouar and F. Chaumette. Path planning in image space for robust visual servoing. In *IEEE International Conference on Robotics and Automation, ICRA '00, San Francisco, CA, USA*, volume 3, pages 2759–2764, April 2000.

[37] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4):534–549, August 2002.

[38] W. T. Miller. Sensor based control of robotic manipulators using a general learning algorithm. *IEEE Transactions on Robotics and Automation*, RA-3, nº 2:157–165, 1987.

[39] W.T. Miller. Real-time application of neural networks for sensored-based control of robots with vision. *IEEE Transactions system, Man and Cybernetics*, 19 n4:825–831, 1989.

[40] G. Morel, E. Malis, and S. Boudet. Impedance based combination of visual and force control. In *IEEE International Conference on Robotics and Automation, Leuven, Belgium*, volume 2, pages 1743–1748, May 1998.

[41] Bradley Nelson, James Morrow, and Pradeep Khosla. Robotic manipulation using high bandwidth force and vision feedback. *Mathematical and Computer Modelling - An International Journal*, 24(5/6):11–29, 1995.

[42] Dinesh K. Pai. Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum*, 21(3):347–352, 2002.

[43] Jae Seok Park and Myung Jin Chung. Image space trajectory generation for image-based visual servoing under large pose error. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1159–1164, 2001.

[44] J.S. Park and M.J. Chung. Trajectory generation for image-based visual servoing using uncalibrated stereo rig. In *Proceedings of International Conference on Automation, Robotics and Computer Vision, Singapore*, volume 12, 2000.

[45] J.S. Park and M.J. Chung. Image-based trajectory generation for visual-servoing using projective invariants. In *Proceedings of International Symposium on Robotics, Seoul, Korea*, volume 4, pages 972–981, 2001.

[46] J. Pomares and F. Torres. Movement-flow-based visual servoing and force control fusion for manipulation tasks in unstructured environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(1):4–15, Feb. 2005.

[47] R.E. Prajoux. Visual tracking. In D. Nitzan et al., editor, *Machine intelligence research applied to industrial automation*, pages 17–37. SRI International, August 1979.

[48] M. Prats, R. Ramos-Garijo, P.J. Sanz, and A.P. Del Pobil. Recent progress in the uji librarian robot. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 6, pages 5227–5232, October 2004.

[49] Hong Qin and Demetri Terzopoulos. D-nurbs: A physics-based framework for geometric design. Technical Report 1, Los Alamitos, CA, USA, 1996.

[50] P. Questa, E. Grossmann, and G. Sandini. Camera self orientation and docking maneuver using normal flow. In *SPIE AeroSense95, Orlando, Florida, USA*, April 1995.

[51] M. Quinlan, C. Murch, T. Moore, R. Middleton, L. Li, R. King, and S. Chalup. The 2004 nubots team report. Technical report, 2004.

[52] M.H. Raibert and J.J. Craig. Hybrid position/force control of manipulators. *ASME Journal of Dynamic Systems, Measurement, and Control*, 102(2):126–133, 1981.

[53] Th. Röfer, H.-D. Burkhard, U. Düert, J. Homann, D. Göhring, M. Jüngel, M. Lötzsch, O. v. Stryk, R. Brunn, M. Kallnik, M. Kunz, S. Petters, M. Risler, M. Stelzer, I. Dahm, M. Wachter, K. Engel, A. Osterhues, C. Schumann, and J. Ziegler. Germanteam robocup 2003. Technical report, http://www.robocup.de/germanteam/GT2003.pdf, 2003.

[54] Patrick Rives, François Chaumette, and Bernard Espiau. Visual servoing based on a task function approach. In *Experimental Robotics I, Proceedings of the First International Symposium on Experimental Robotics, Montreal, Canada*, pages 412.–428, June 1990.

[55] C. Rosen, D. Nitzan, G. Agin, A. Bavarsky, G. Gleason, J. Hill, D. McGhie, and W. Park. Machine intelligence research applied to industrial automation. Technical Report NSF Grant APR-75-13074, SRI Project 4391, 6th Report, SRI International, Menlo Park, CA, November 1976.

[56] C. Rosen, D. Nitzan, G. Agin, A. Bavarsky, G. Gleason, J. Hill, D. McGhie, and W. Park. Machine intelligence research applied to industrial automation. Technical report, 8th Report, SRI International, August 1978.

[57] M. B. Rubin. *Cosserat Theories: Shells, Rods and Points*. Kluwer, 2000.

[58] J.R. Ryoo, D.Y. Kim, and M.J. Chung. An on-line trajectory generation for control of active head-eye system. In *Proceedings of Asian Control Conference, Shanghai, China*, pages 2983–2988, July 2000.

[59] Claude Samson, Michel Le Borgne, and Bernard Espiau. Robot control: The task function approach. In *Oxford Engineering Science Series*, volume 22. Clarendon Press, Oxford University Press, UK, 1 edition, April 1991.

[60] A. C. Sanderson and L. E. Weiss. Image-based visual servo control using relational graph error signals. *Proceedings of the IEEE International Conference on Cybernetics and Society*, pages 1074–1077, 1980.

[61] Joris De Schutter and Johan Baeten. *Integrated Visual Servoing and Force Control: The Task Frame Approach*, volume 8 of *978-3-540-40475-0*. Springer Tracts in Advanced Robotics, 2003.

[62] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5(2):99–108, June 1973.

[63] V. Sundareswaran, P. Bouthemy, and F. Chaumette. Exploiting image motion for active vision in a visual servoing framework. *International Journal of Robotics Research*, 15(6):629–645, 1996.

[64] A. Theetten, L. Grisoni, C. Andriot, and B. Barsky. Geometrically exact dynamic splines. *Computer-Aided Design*, 40(1):35–48, January 2008.

[65] Alan Watt and Mark Watt. *Advanced animation and rendering techniques*. ACM, New York, NY, USA, 1991.

[66] L. E. Weiss. *Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach*. PhD thesis, Carnegie-Mellon University, April 1984.

[67] William M. Wichman. Use of optical feedback in the computer control of an arm. Technical report, Standford AI project, AI memo 55, August 1967.

[68] www.tekkotsu.org.

[69] Kuk-Jin Yoon, Yoojin Choi, and In So Kweon. Fast separation of reflection components using a specularity-invariant image representation. In *Image Processing, 2006 IEEE International Conference on*, pages 973–976, 8-11 Oct. 2006.