Modeling a legged robot for visual servoing

Zelmar Echegoyen, Alicia d'Anjou, and Manuel Graña

Computational Intelligence Group, Dept. CCIA

Paseo Manuel de Lardiazabal, 1 20018 San Sebastian - Spain

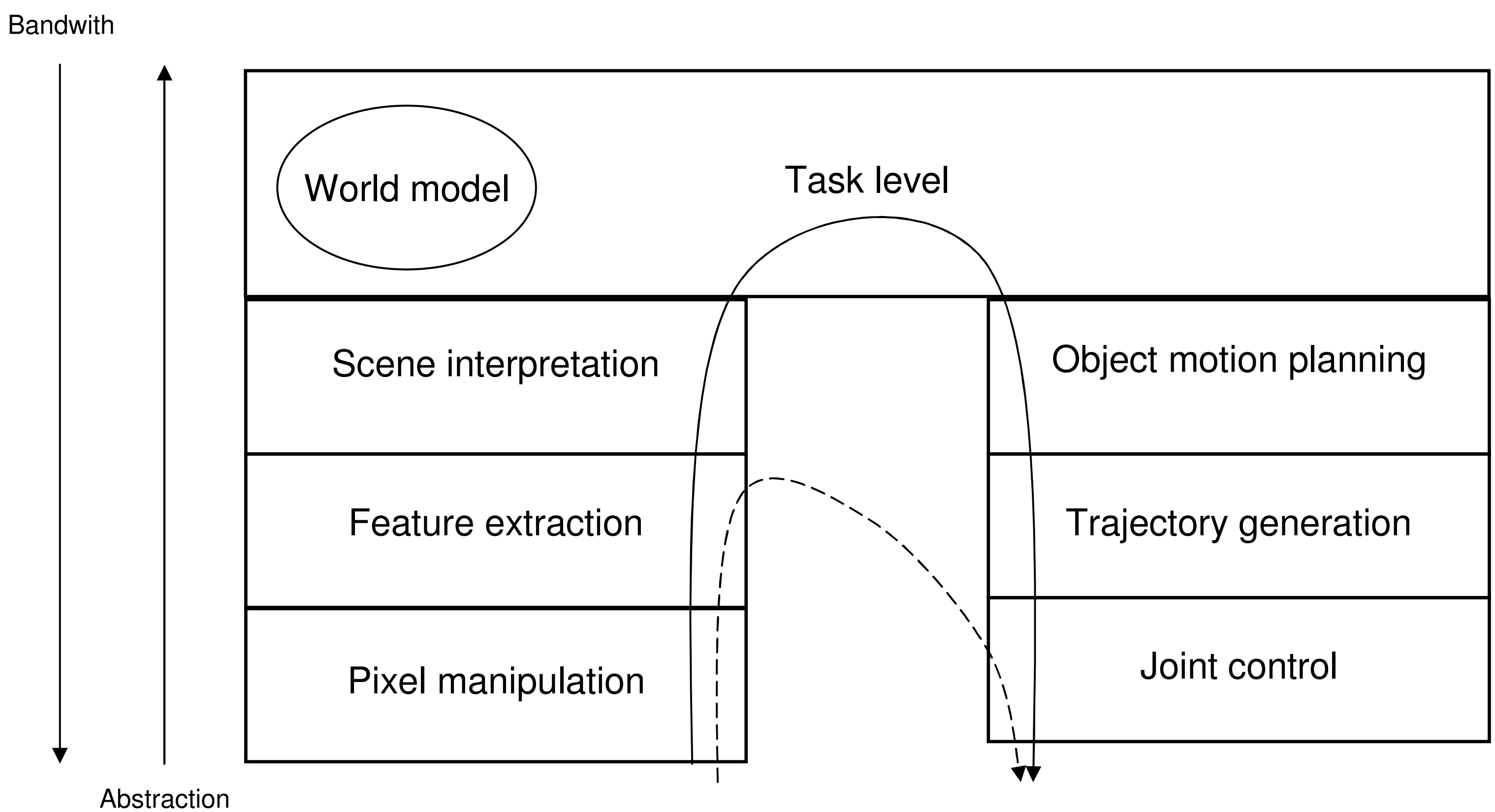
eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea
The University of the Basque Country

Overview

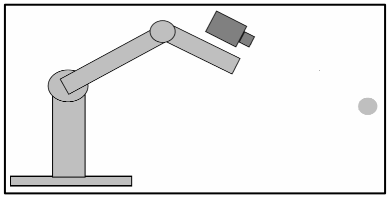
- Visual servoing
- Direct Aibo kinematics
- Inverse kinematics
- Experimental results
- Conclusions
- Future work
- Conclusion

Visual servoing



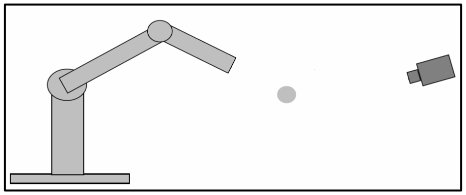
Visual servoing

Camera configuration



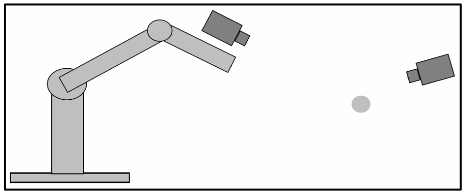
Eye_in_hand configuration

Positioning accuracy depends directly on the accuracy of the hand-eye calibration



Fixed camera configuration

Positioning accuracy is independent of hand-eye calibration error.
occlusion of the end-effector



Mix configuration

The implementation often requires solution of a more demanding vision problem, due to the need of tracking the end-effector as well as the target object.

Visual servoing

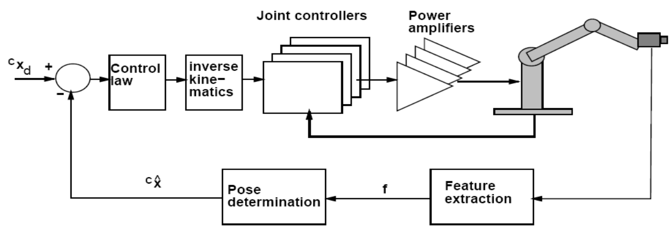
Servoing Architectures

A – Does the vision system provide a set-points as input to the robot´s joint-level control?, or does the visual controller directly compute the joint-level inputs?

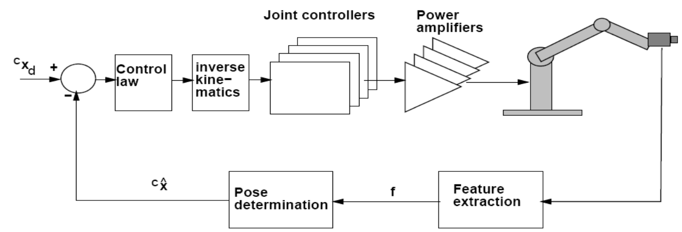
Dynamic look and move system/ Direct visual servo

B - Is the error signal defined in 3D (task space) coordinates?, or directly defined in terms of image features?

Position based/ Image based



Dynamic position-based look-and-move structure



Position-based visual servo (PBVS) structure

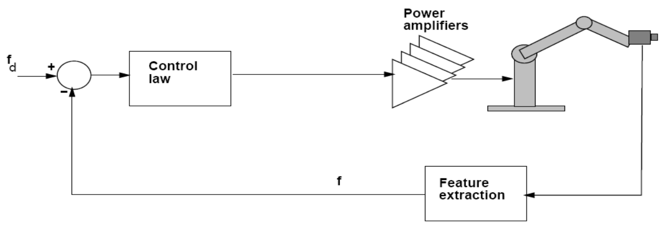
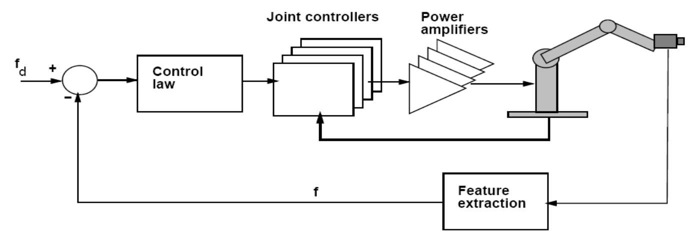


Image-based visual servo (IBVS) structure



Dynamic image-based look-and-move structure

Visual servoing

Servoing Architectures

Dynamic look and move system vs. Direct visual servo

Dynamic look and move system	Direct visual servo
Make use of joint feedback to internally stabilize the robot	Eliminates the robot controller entirely, computing directly the joint inputs, using vision alone to stabilize the mechanism

•Separate the kinematic singularities of the mechanism from the visual controller, allowing the robot to be considered as an ideal cartesian motion device

→ the system is greatly simplified

•An internal feedback with a high sampling rate usually presents the visual controller with idealized axis dynamics.

•Many robots have an interface for accepting cartesian velocity or incremental position commands

→ Simplifies the construction of the visual servo system, and makes the methods more portable.

•On relatively low sampling rates from vision → direct control of a robot is an extremely challenging control problem (complex nonlinear dynamics)

Visual servoing

Servoing Architectures

Position based vs. Image based

Position based → Features are used in conjunction with a geometric model of the target and the known camera model in order to estimate the pose of the target with respect to the camera.

Image based → Control values are computed on the basis of image features directly.

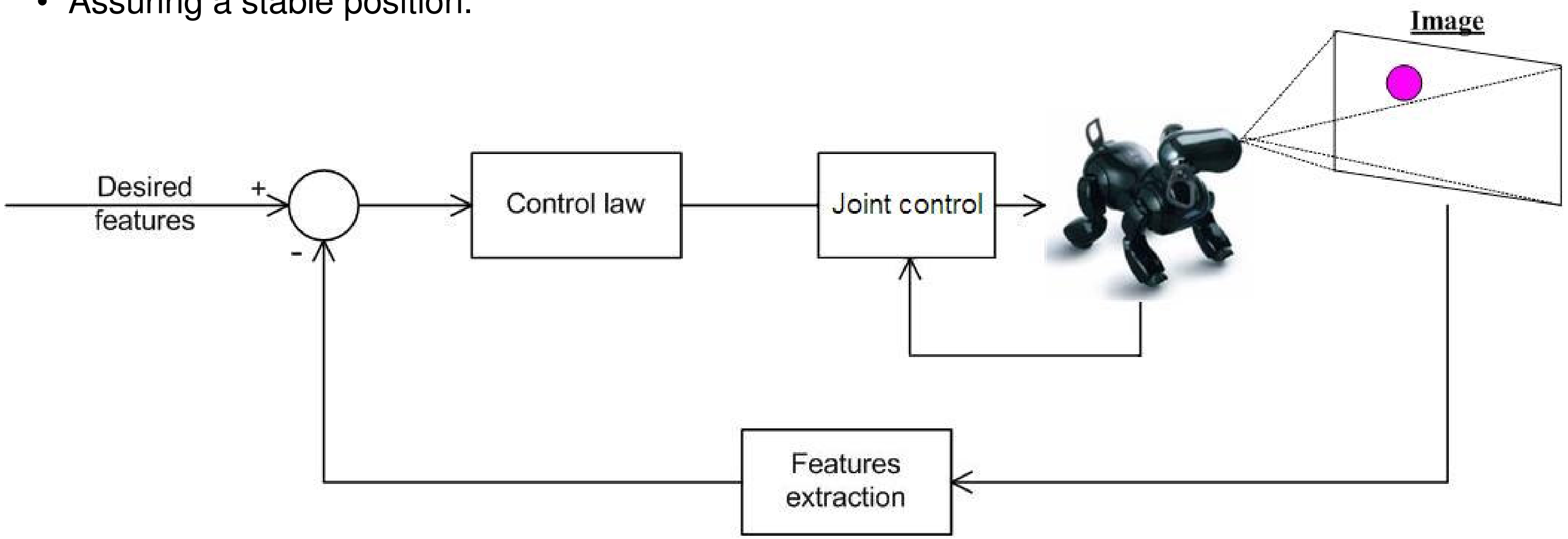
Position based
- Facilitates Cartesian path planning, the trajectory of the robot is well stated
- It directly controls the camera trajectory in Cartesian space.
- It is free of local min ima or singularities at control level
- The image features used in the pose estimation may leave the image
- Sensitive to camera calibration and the chosen pose estimation algorithm

Image based
- May reduce computational delay
- Eliminates the necessity for image interpretation
- Eliminates errors due to sensor modeling and camera calibration
- It presents a significant challenge to controller design since the system is non-linear and highly copule.
- Certain control tasks can lead to singularities in the image Jacobian, resulting in system failure

Objective

Attemp to Model a legged robot for visual servoing ⟶ **Tracking a ball**

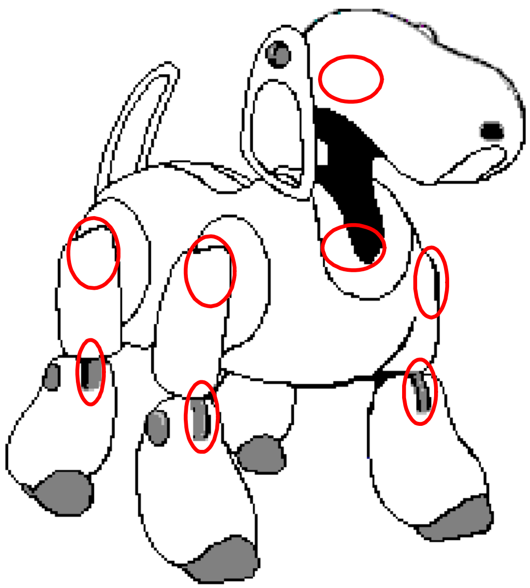
- Special emphasis on obtaining the image jacobian for all the robot joints.
- Defining and keeping a fixed coordinates system.
- Assuring a stable position.



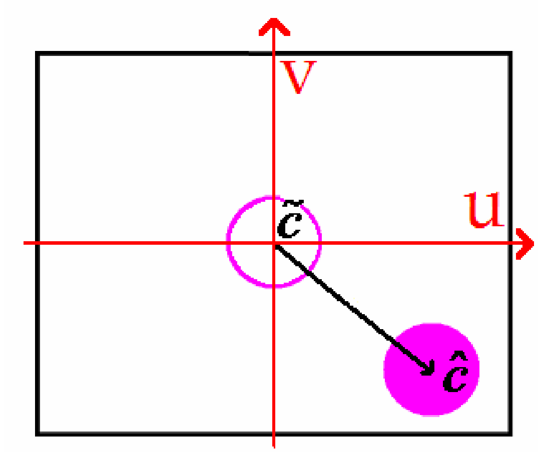
Objective

Minimize $\left\| \tilde{c} - \hat{c}(\theta) \right\|$

Restricted to $\begin{cases} \text{Stable positions} \\ \text{Joints ranges} \end{cases}$



θ : joints values

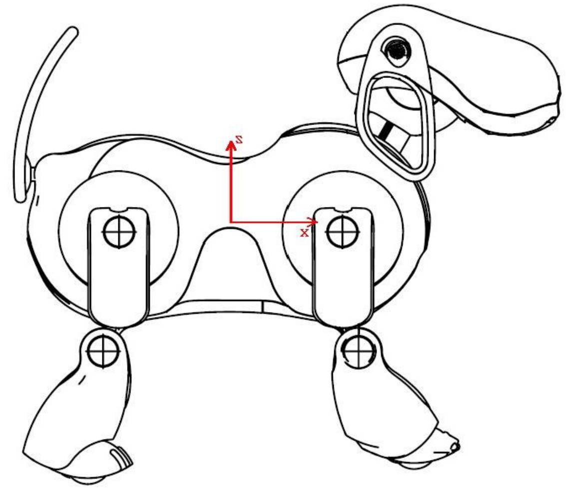


$\tilde{c}$: desired features

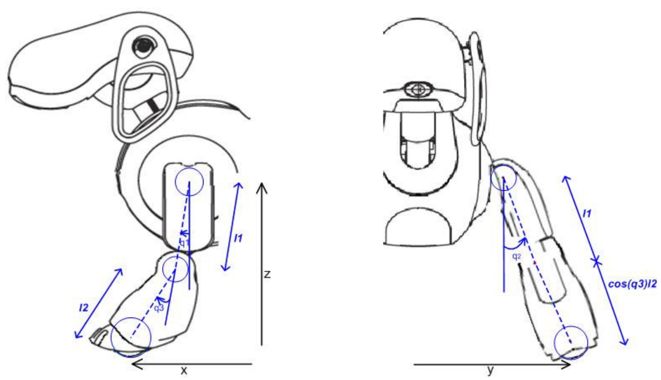
$\hat{c}$: true features

Direct Aibo Kinematics

Degrees of freedom

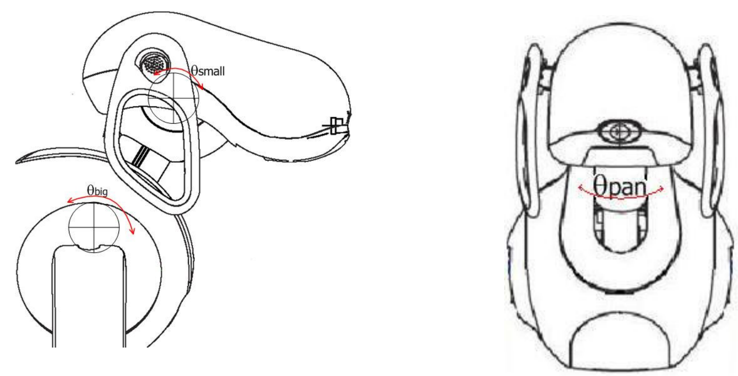


Legs degrees of freedom



Three joint per leg J_1, J_2 and J_3

Head degrees of freedom



Three head joints θ_{big}, θ_{small}, θ_{pan},

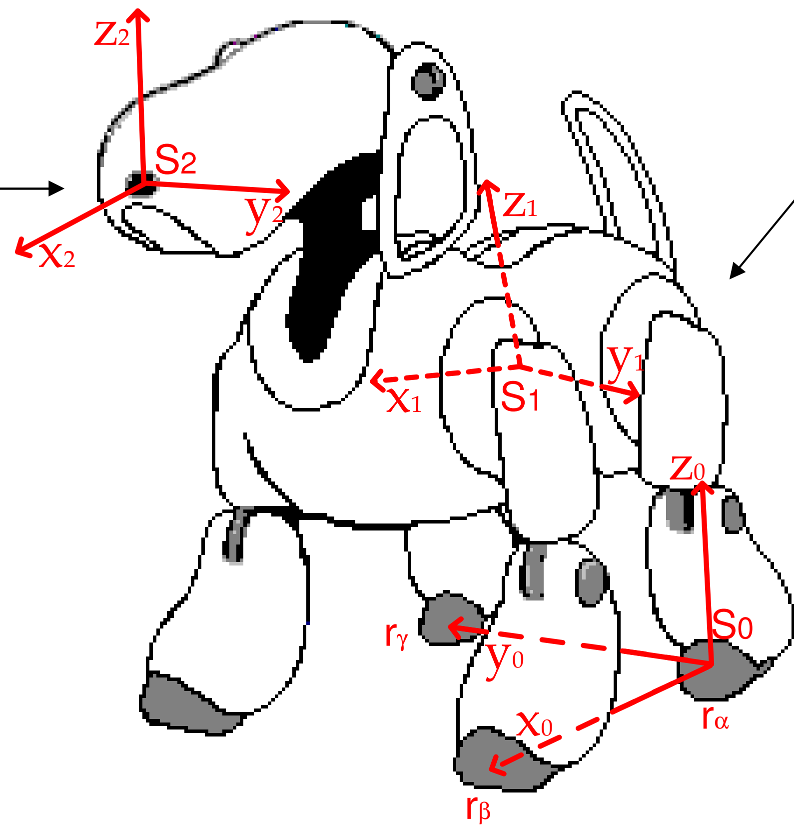
Direct Aibo Kinematics

Coordinate systems

Camera system

Fixed to the image plane

Ortonormal



Body system

Robot reference system

Ortonormal

Fixed system

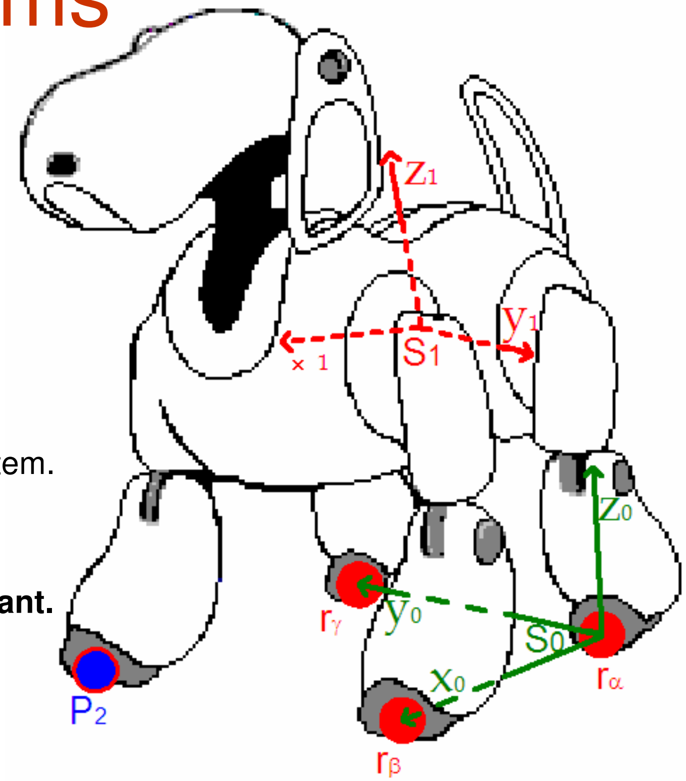
Absolute invariant system

Not ortonormal

Restriction: keep the distances between r_α, r_β and r_γ constant → It will be made at the inverse kinematics

Direct Aibo Kinematics

Coordinate systems



Why the distances between r_α, r_β and r_γ must be constants?

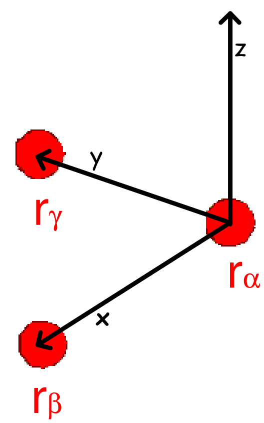
- The enviroment is not estructured
- A reference is needed to avoid odometry errors
- The ball can be moved, so it is not possible to use it as a reference

It is needed to use a fixed reference system

- Only the points over the ground could be used to determine a fixed reference system.
- If the distances between feet are constant, then their positions are constant too.

Three feet are required to define a system, so the fourth foot position is redundant.

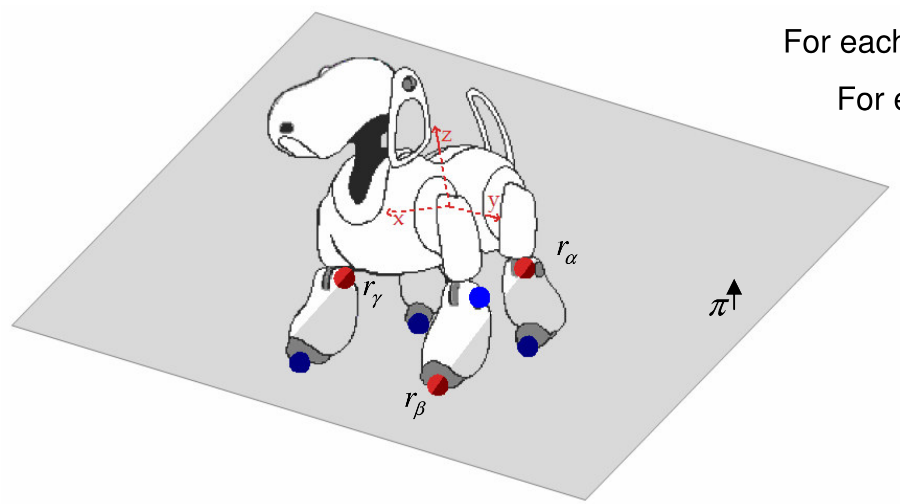
Supporting points
$$\begin{cases} x_0 = r_\beta - r_\alpha \\ y_0 = r_\gamma - r_\alpha \\ z_0 = y_0 \times x_0 \end{cases}$$



Direct Aibo Kinematics

How to determine the supporting points?

A support point for a leg could be either the foot or de knee



For each legs combination (α, β, γ):

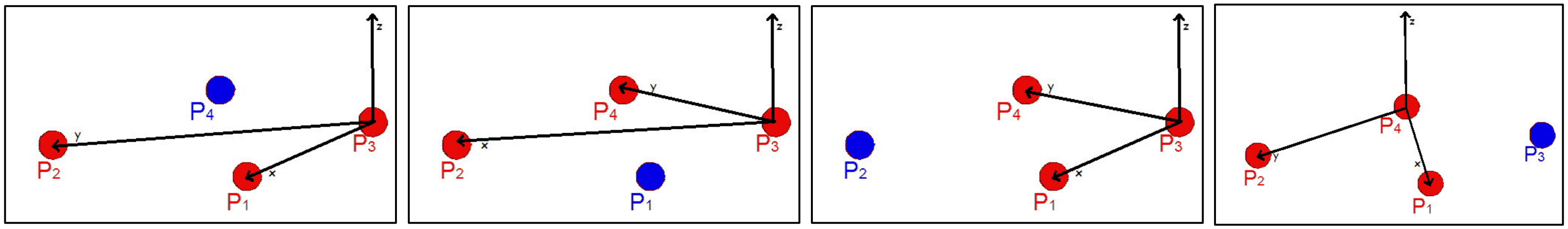
For each supporting points combination r_α r_β r_γ

π = rα rβ rγ

If for the others potential supporting points r: π(r) ≥ 0

π is the supporting plane and rα, rβ, rγ are the supporting points

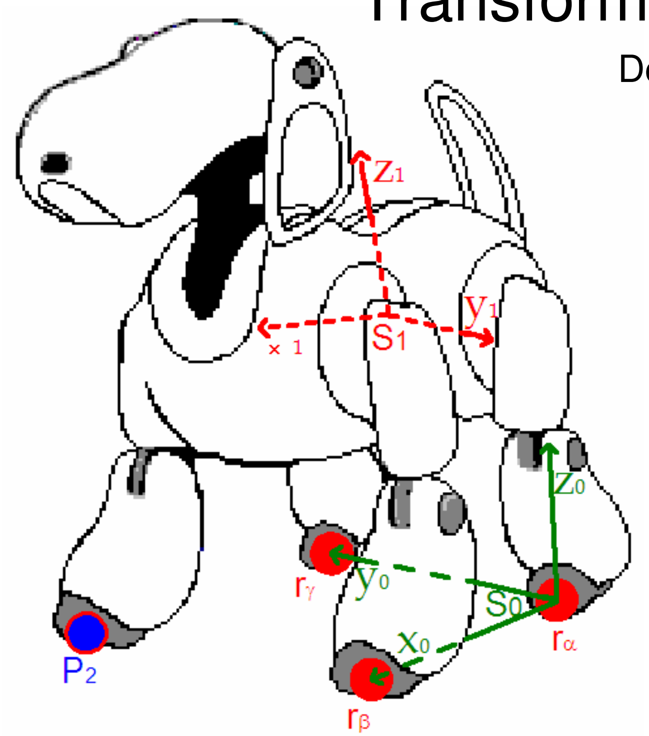
Supporting points configuration



Direct Aibo Kinematics

Coordinate systems

Transformation between S_1 and S_0



Dependence on the leg joints

Rotation/scale matrix

$$R = \begin{pmatrix} -x_\beta + x_\alpha & (x_\gamma - x_\alpha)(z_\beta - z_\alpha) & (x_\gamma - x_\alpha)(z_\beta - z_\alpha) - (x_\gamma - x_\alpha)(y_\beta - y_\alpha) & 0 \\ -y_\beta + y_\alpha & (y_\gamma - y_\alpha)(z_\beta - z_\alpha) & (y_\gamma - y_\alpha)(z_\beta - z_\alpha) - (z_\gamma - z_\alpha)(x_\beta - x_\alpha) & 0 \\ -z_\beta + z_\alpha & (x_\gamma - x_\alpha)(z_\beta - z_\alpha) & (x_\gamma - x_\alpha)(z_\beta - z_\alpha) - (y_\gamma - y_\alpha)(x_\beta - x_\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translation matrix

$$T = \begin{pmatrix} 1 & 0 & 0 & x_\alpha \\ 0 & 1 & 0 & y_\alpha \\ 0 & 0 & 1 & z_\alpha \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Supporting points positions

$$r_\alpha = (x_\alpha, y_\alpha, z_\alpha, 1)^T \quad r_\beta = (x_\beta, y_\beta, z_\beta, 1)^T \quad r_\gamma = (x_\gamma, y_\gamma, z_\gamma, 1)^T$$

$$\boxed{{}_{S_1}I_{S_0} = R \cdot T}$$

Direct Aibo Kinematics

Coordinate systems

Transformation between S_1 and S_2

Translation from camera to the neck base

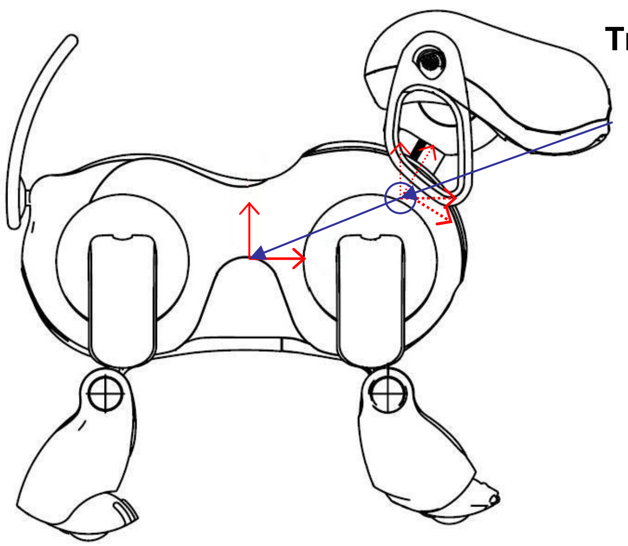
$$T_1 = \begin{pmatrix} 1 & 0 & 0 & x_c \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

nod/pan rotation

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

Tilt rotation

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



Translation from neck base to the robot centre

$$T_2 = \begin{pmatrix} 1 & 0 & 0 & -l\sin(tilt) + x_b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l\cos(tilt) + z_b \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boxed{{}_{S_1}I_{S_2} = T_2 \cdot R_2 \cdot R_1 \cdot T_1}$$

Direct Aibo Kinematics

Feature Jacobian matrix

Dependence of the image features on the diverse degrees of freedom of the robot

Composition of previous transformations:

- J_{cb} - dependence on the feature
- J_{br} - dependence on the target object
- $J_{r\theta}$ - dependence on the support points

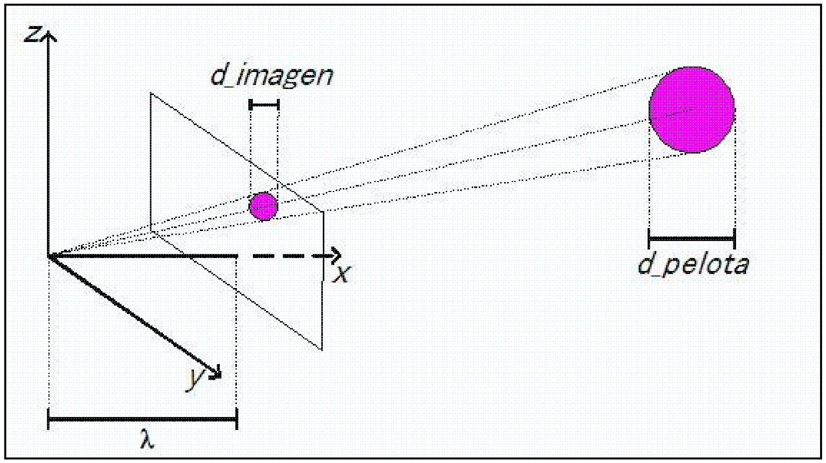
$$\boxed{\Delta c \simeq \underbrace{[(J_{cb} \circ J_{br}) \circ J_{r\theta}]}_{J_{c\theta}} \bullet \Delta_{\theta}}$$

Direct Aibo Kinematics

Feature Jacobian matrix

Dependence on the features

Image features



Ball coordinates in reference system S_i

$$B_i = \begin{pmatrix} x_{b_i} \\ y_{b_i} \\ z_{b_i} \end{pmatrix}$$

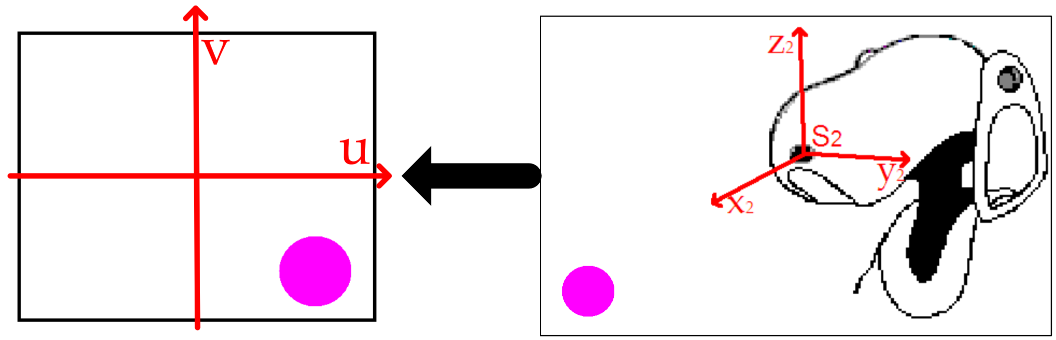
Ball position features in terms of head coordinates

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{\lambda}{x_{b_2}} \begin{pmatrix} y_{b_2} \\ z_{b_2} \end{pmatrix} = f(B_2)$$

Direct Aibo Kinematics

Feature Jacobian matrix

Dependence on the features

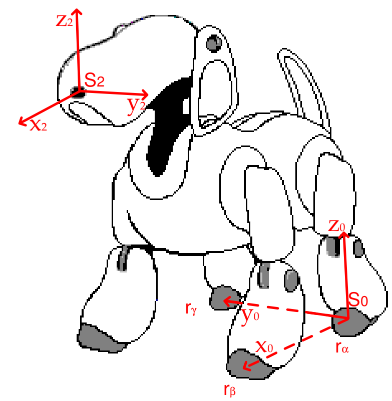


Deriving the relationship between images features and ball position in S2

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{\lambda}{x_{b_2}} \begin{pmatrix} y_{b_2} \\ z_{b_2} \end{pmatrix} = f(B_2) \Rightarrow \begin{pmatrix} \delta u \\ \delta v \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{-\lambda y_p}{x_p{}^2} & \frac{\lambda}{x_p} & 0 & 0 \\ \frac{-\lambda z_p}{x_p{}^2} & 0 & \frac{\lambda}{x_p} & 0 \end{pmatrix}}_{J_{cb}} \begin{pmatrix} \delta x_{b_2} \\ \delta y_{b_2} \\ \delta z_{b_2} \\ 0 \end{pmatrix}$$

$$\boxed{\Delta c \simeq J_{cb} . \Delta b}$$

Direct Aibo Kinematics
Feature Jacobian matrix



Dependence on the target object

r: supporting points positions and head joints

b: ball position in S_0

Deriving the transformation matrix between S_0 and S_2, as a function of the support points and the head articulations

$$J_{br} = \frac{\delta({}_{S_2}I_{S_1} \circ {}_{S_1}I_{S_0})}{\delta r} b_0$$

As (${}_{S_2}I_{S_1}$) is a function of r_{head} (head articulations) and (${}_{S_1}I_{S_0}$) is a function of r_{legs} (support points positions),

$$J_{br} = [\frac{\delta({}_{S_2}I_{S_1})}{\delta r_{head}} \circ ({}_{S_1}I_{S_0}) + ({}_{S_2}I_{S_1}) \circ \frac{\delta({}_{S_1}I_{S_0})}{\delta r_{legs}}] b_0$$

The dependence between the variations in the ball position and the variations in the head degree of freedoms and in the legs positions can be summarized by:

$$\boxed{\Delta b \simeq J_{br} \cdot \Delta r}$$

Direct Aibo Kinematics

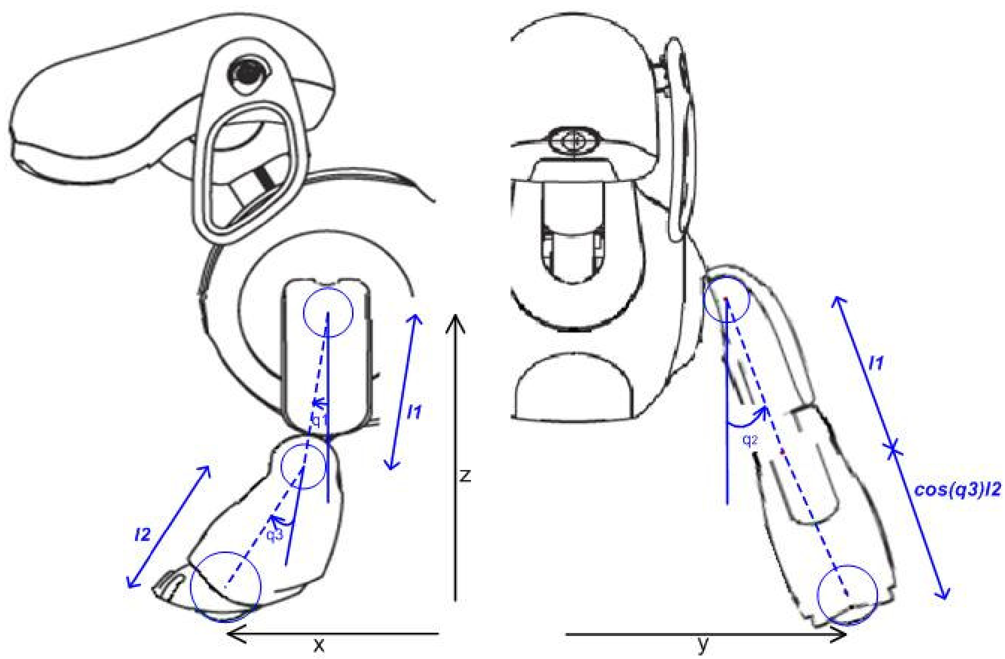
Feature Jacobian matrix

Dependence on the robot articulations

Feet position

- T_1: Translation along de 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_3: Clockwise rotation about y-axis by angle q_3.
- T_2: Translation along de z-axis with length l_2.
- T_l: Translation along de x-axis with length l_2.
- T_a: Translation along de y-axis with length l_2.

$$\vec{P}_f = (T_l \circ T_a \circ R_2 \circ R_1 \circ T_1 \circ R_3 \circ T_2) \bullet \vec{0}$$



Knees position

- T_1: Translation along de 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.
- T_l: Translation along de x-axis with length l_2.
- T_a: Translation along de y-axis with length l_2.

$$\vec{P}_k = (T_l \circ T_a \circ R_2 \circ R_1 \circ T_1) \bullet \vec{0}$$

Direct Aibo Kinematics

Feature Jacobian matrix

Dependence on the robot articulations

Which part of a leg that is in contact with the ground?

Two possible jacobean matrixes

$$M_i = \frac{\delta P_{fi}}{\delta \theta_i} \quad \text{- for the foot}$$

$$M_i = \frac{\delta P_{ki}}{\delta \theta_i} \quad \text{- for the knee}$$

$$\boxed{\Delta P_i \simeq M_i . \Delta \theta_i}$$

Changes in the foot and the knee coordinates according to the legs joints changes

All the legs in a single jacobian matrix

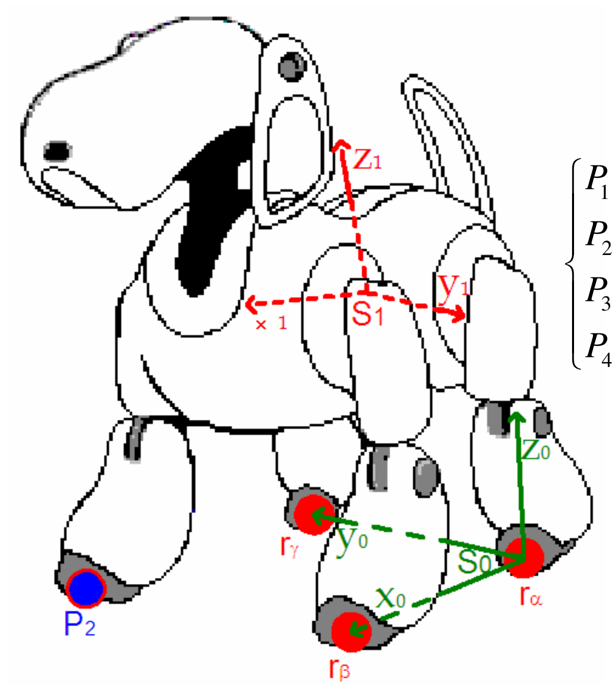
$$\begin{pmatrix} \delta P_1 \\ \vdots \\ \delta P_2 \\ \vdots \\ \delta P_3 \\ \vdots \\ \delta P_1 \end{pmatrix} = \begin{pmatrix} M_1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & M_2 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & M_3 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & M_4 \end{pmatrix} \begin{pmatrix} \delta \theta_1 \\ \vdots \\ \delta \theta_2 \\ \vdots \\ \delta \theta_3 \\ \vdots \\ \delta \theta_4 \end{pmatrix}$$

$$\boxed{\Delta P \simeq J_{p\theta} . \Delta \theta}$$

Direct Aibo Kinematics

Feature Jacobian matrix

Dependence on the supporting points



$$\begin{Bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{Bmatrix} \Leftrightarrow \begin{Bmatrix} r_\alpha \\ r_\beta \\ r_\gamma \end{Bmatrix} \Longrightarrow \begin{pmatrix} \delta p_h \\ \vdots \\ \delta p_1 \\ \vdots \\ \delta p_2 \\ \vdots \\ \delta p_3 \\ \vdots \\ \delta p_4 \end{pmatrix} = \begin{pmatrix} Id & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & I_{1\alpha} & \cdots & I_{1\beta} & \cdots & I_{1\gamma} \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & I_{2\alpha} & \cdots & I_{2\beta} & \cdots & I_{2\gamma} \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & I_{3\alpha} & \cdots & I_{3\beta} & \cdots & I_{3\gamma} \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & I_{4\alpha} & \cdots & I_{4\beta} & \cdots & I_{4\gamma} \end{pmatrix} \begin{pmatrix} \delta r_h \\ \vdots \\ \delta r_\alpha \\ \vdots \\ \delta r_\beta \\ \vdots \\ \delta r_\gamma \end{pmatrix} \Longrightarrow \boxed{\Delta p \simeq J_{pr}.\Delta r}$$

I_{xy} = Id, if the leg x has the support point y.
I_{xy} = 0, if the leg x has not the support point y.

r – head joints and generic support points

p – head joints and real support points

Direct Aibo Kinematics

Feature Jacobian matrix

Support points and joints

Previous defined jacobian $J_{p\theta}$ and J_{pr}:

$$\Delta P \simeq J_{p\theta}.\Delta\theta$$ - dependence on the robot articulations

$$\Delta p \simeq J_{pr}.\Delta r$$ - dependence on the support points

Defining the following Jacobian matrix $J_{r\theta} = J_{pr}{}^{+}.J_{p\theta}$

Dependence relation between the variations of support points positions and robot articulations

$$\boxed{\Delta r \simeq J_{r\theta} \cdot \Delta\theta}$$

Direct Aibo Kinematics

Feature Jacobian matrix

Full Jacobian matrix

Dependence of the image features on the diverse degrees of freedom of the robot

Composition of previous transformations:

- J_{cb} - dependence on the feature
- J_{br} - dependence on the target object
- $J_{r\theta}$ - dependence on the support points

$$\boxed{\Delta c = [(J_{cb} J_{br}) J_{r\theta}] \cdot \Delta\theta}$$

We call this matrix $J_{c\theta}$

Inverse Kinematics

Determine the instantaneous of each of the robot degrees of freedom that will be needed to bring the ball centre to the image centre

We should obtain the inverse of the $J_{c\theta}$ matrix. However, this is not possible because the matrix is not invertible → get the seudoinverse of $J_{c\theta}$, by minimum squares

$$\dot{\theta} = J_{c\theta}{}^{+}\dot{c} + (I - J_{c\theta}{}^{+}J_{c\theta})n$$ Being n an arbitrary vector of $\mathbb{R}^{15}$

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

✗ It does not take into account the restriction of keeping the distances constants

Inverse Kinematics

Objective → **center the ball in the image**

Restriction: keeping the distances between supporting points constants

It is needed to determine how these variations in the supporting points positions afect the distances between them

$$l = \begin{pmatrix} \|l_1\| \\ \|l_2\| \\ \|l_3\| \end{pmatrix} = \begin{pmatrix} \|r_\alpha - r_\beta\| \\ \|r_\beta - r_\gamma\| \\ \|r_\gamma - r_\alpha\| \end{pmatrix}$$

Diferencing l we get the Jacobian matrix that relates these changes. →

$$J_{lr} = \begin{pmatrix} 0 & \cdots & \frac{\delta l_1}{\delta r_\alpha} & \cdots & \frac{\delta l_1}{\delta r_\beta} & \cdots & \frac{\delta l_1}{\delta r_\gamma} \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \frac{\delta l_2}{\delta r_\alpha} & \cdots & \frac{\delta l_2}{\delta r_\beta} & \cdots & \frac{\delta l_2}{\delta r_\gamma} \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \frac{\delta l_3}{\delta r_\alpha} & \cdots & \frac{\delta l_3}{\delta r_\beta} & \cdots & \frac{\delta l_3}{\delta r_\gamma} \end{pmatrix}$$

* The zeros at the first column are for the head efectors, which must not be affected by this restriction

The dependence is resumed in the following equation: $\boxed{\Delta l \simeq J_{lr} . \Delta r}$

Inverse Kinematics

$$\left.\begin{matrix} J_{rc} \\ J_{lr} \\ J_{r\theta} \end{matrix}\right\} \longrightarrow \Delta r = \left[(I - J_{lr}{}^{+} J_{lr}) \left\{ (I - J_{lr}{}^{+} J_{lr}) J_{rc}{}^{+} \right\}^{+} \right] \Delta c$$

In order to get the articulations variations we add the seudoinverse of $J_{r\theta}$, we also add a velocity constant to control the advance velocity of the robot

$$\boxed{\Delta \theta = J_{r\theta}{}^{+} \left(I - J_{lr}{}^{+} J_{lr} \right) \left\{ \left(I - J_{lr}{}^{+} J_{lr} \right) J_{rc}{}^{+} \right\}^{+} \{ k_i \Delta c \}}$$

This equation allows us to determine the step variations on the robot degrees of freedom to get the desired conguration of the image, and determines the following sucesion:

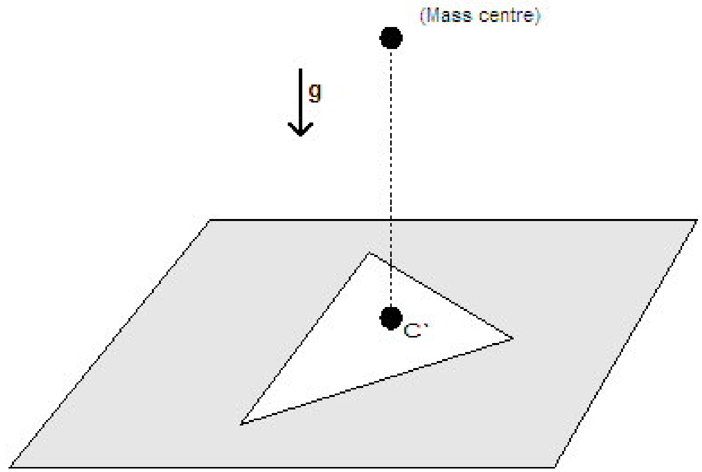
$$\boxed{\Delta c_{i+1} = \left[J_{c\theta} J_{r\theta}{}^{+} (I - J_{lr}{}^{+} J_{lr}) \left\{ (I - J_{lr}{}^{+} J_{lr}) J_{rc}{}^{+} \right\}^{+} k_i \right] \Delta c_i}$$

If the velocity constant, k_i, is small enough, Δc_i converge to 0: $c = \left\| \tilde{c} - \hat{c}_n(\theta) \right\| \xrightarrow[n \to \infty]{} 0$

Inverse Kinematics

Avoid unstable congurations

The equation is unrestricted and may drive the robot into unstable configurations

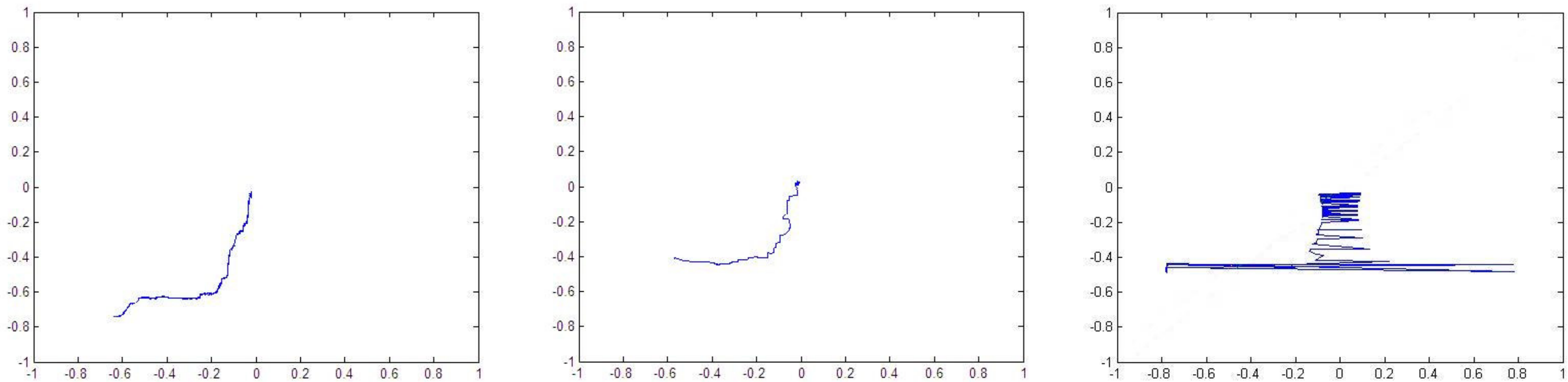


- Mass centre proyection outside the support points triplet
- The projection point is too close to the triangle boundary
- The joints exceeds the limit values

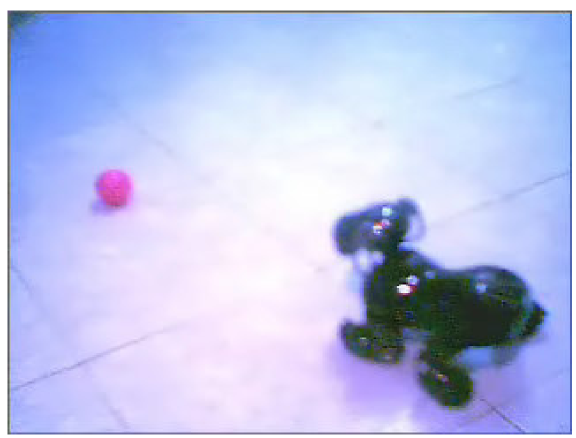
We restrict the visual servoing to the head degrees of freedom

$$\delta\theta = M_h^{+} \delta c$$

Experimental results



Velocity constant value



Future Work

- Stability Analysis
- Advance behavior

Conclusions

- The system provides the desired controls.
- Real time response when the seudoinverse is computed in the onboard processor of the robot.

Questions?