Robotics 2

Visual servoing

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Visual servoing

- objective

  use information acquired by vision sensors (cameras) for feedback control of the pose/motion of a robot (or of parts of it)

$\boxed{+}$ data acquisition ~ human eye, with very large information content in the acquired images

$\boxed{-}$ difficult to extract essential data, nonlinear perspective transformations, high sensitivity to ambient conditions (lightening), noise

Some applications

automatic navigation of robotic systems (agricolture, automotive)

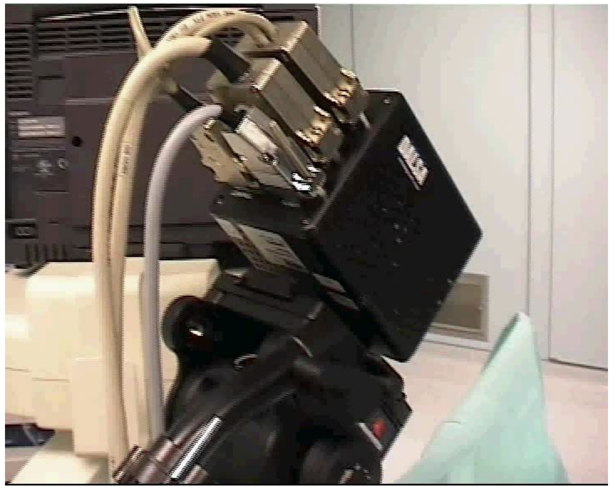
video

video

surveillance

bio-motion synchronization (surgical robotics)

video



video

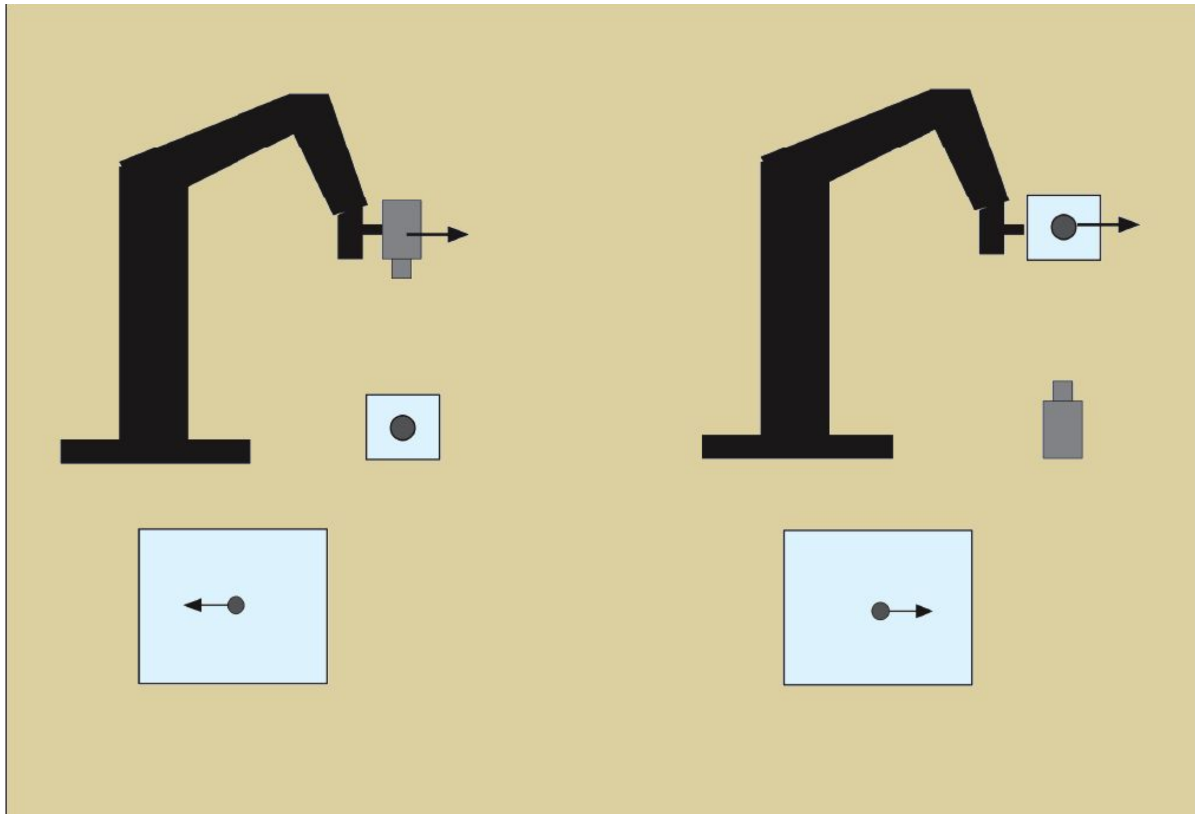
Image processing

- real time extraction of characteristic parameters useful for robot motion control
  - features: points, area, geometric center, higher-order moments, …
- low-level processing
  - binarization (threshold on grey levels), equalization (histograms), edge detection, …
- segmentation
  - based on regions (possibly in binary images)
  - based on contours
- interpretation
  - association of characteristic parameters (e.g., texture)
  - problem of correspondence of points/characteristics of objects in different images (stereo or on image flows)

Configuration of a vision system

- one, two, or more cameras
  - grey-level or color
- 3D/stereo vision
  - obtained even with a single (moving) camera, with the object taken from different (known) point of views
- camera positioning
  - fixed (eye-to-hand)
  - mounted on the manipulator (eye-in-hand)
- robotized vision heads
  - motorized (e.g., stereo camera on humanoid head or pan-tilt camera on Magellan mobile robot)

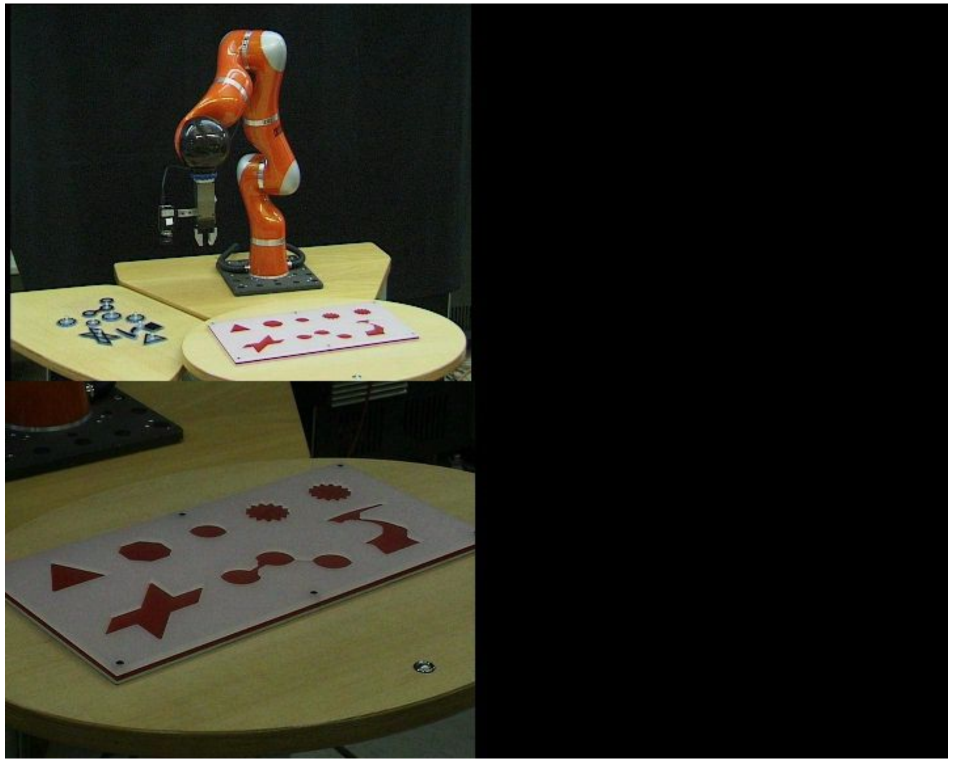
Camera positioning



eye-in-hand eye-to-hand

Vision for assembly

video

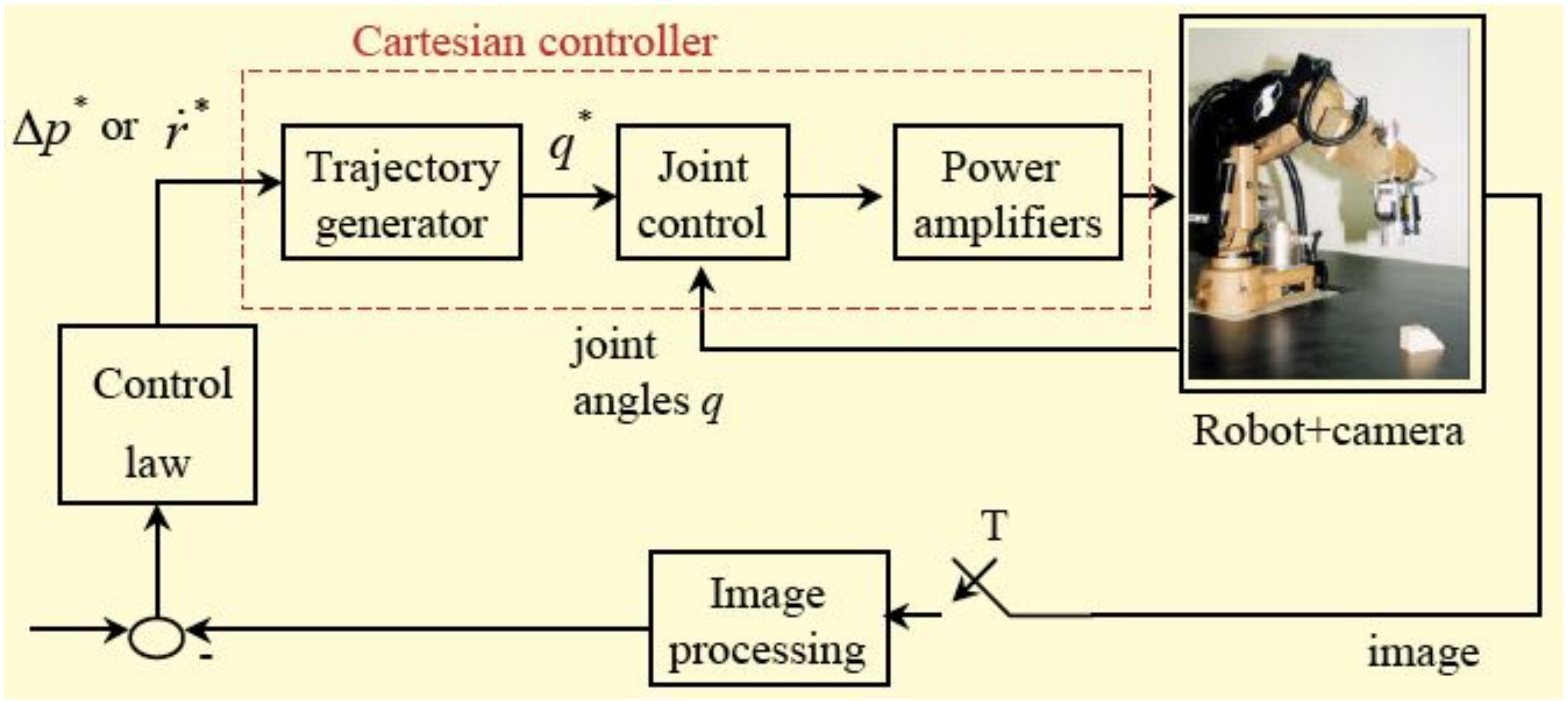


PAPAS-DLR system
(eye-in-hand, hybrid force-vision)

video

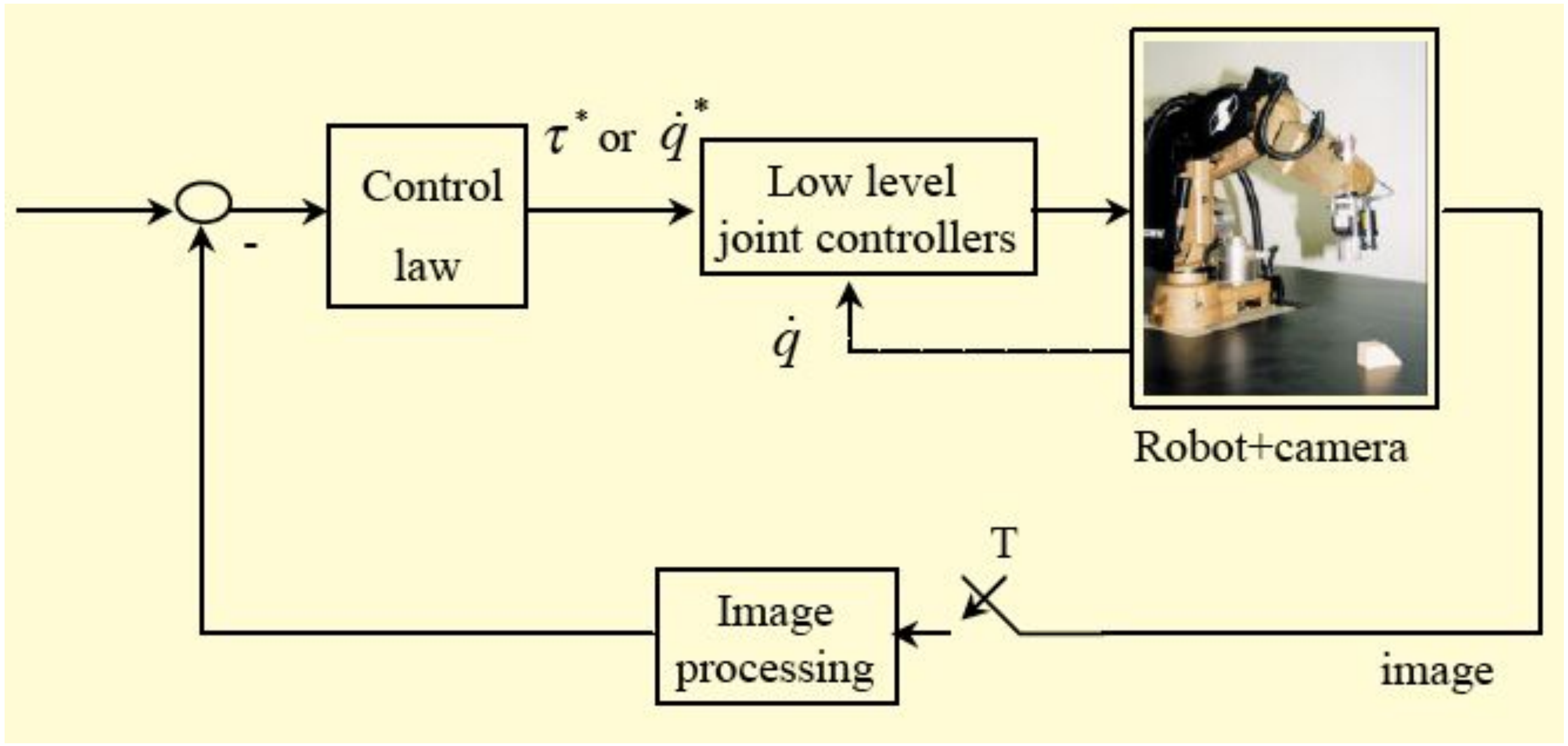
robust w.r.t. motion of
target objects

Indirect/external visual servoing



vision system provides set-point references to a Cartesian motion controller
- "easy" control law (same as without vision)
- appropriate for relatively slow situations (control sampling < 50Hz)

Direct/internal visual servoing



replace Cartesian controller with one based on vision that directly computes joint reference commands

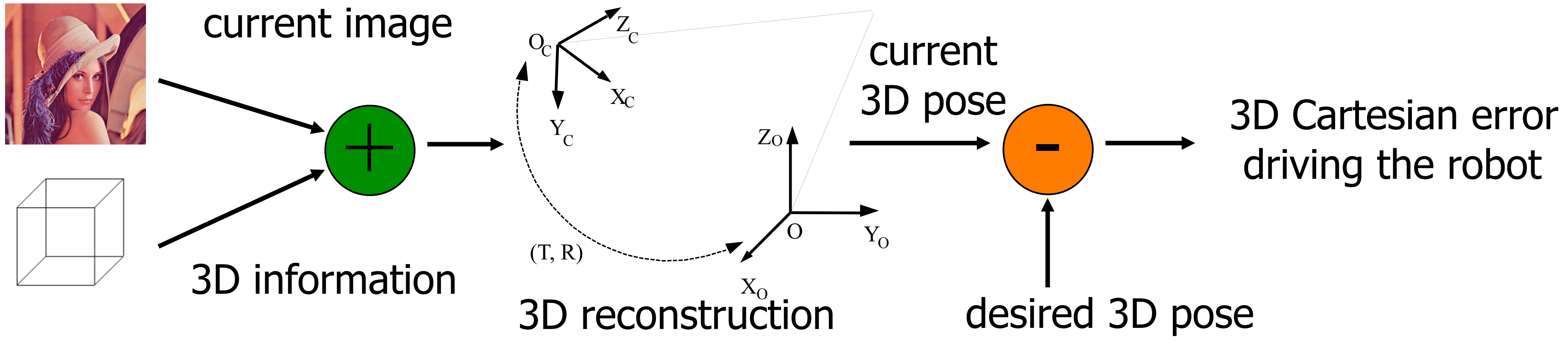
- control law is more complex (involves robot kinematics/dynamics)
- preferred for fast situations (control sampling > 50Hz)

Classification of visual servoing schemes

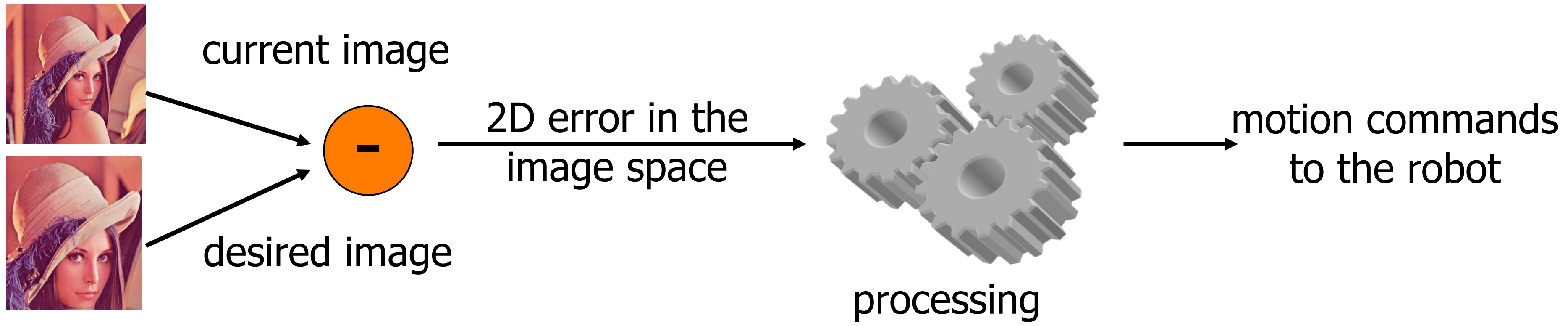
- position-based visual servoing (**PBVS**)
  - information extracted from images (features) is used to reconstruct the current 3D pose (pose/orientation) of an object
  - combined with the knowledge of a desired 3D pose, we generate a Cartesian pose error signal that drives the robot to the goal
- image-based visual servoing (**IBVS**)
  - error is computed directly on the values of the features extracted on the 2D image plane, without going through a 3D reconstruction
  - the robot should move so as to bring the current image features (what it “sees” with the camera) to their desired values
- some mixed schemes are possible (e.g., 2½D methods)

Comparison between the two schemes

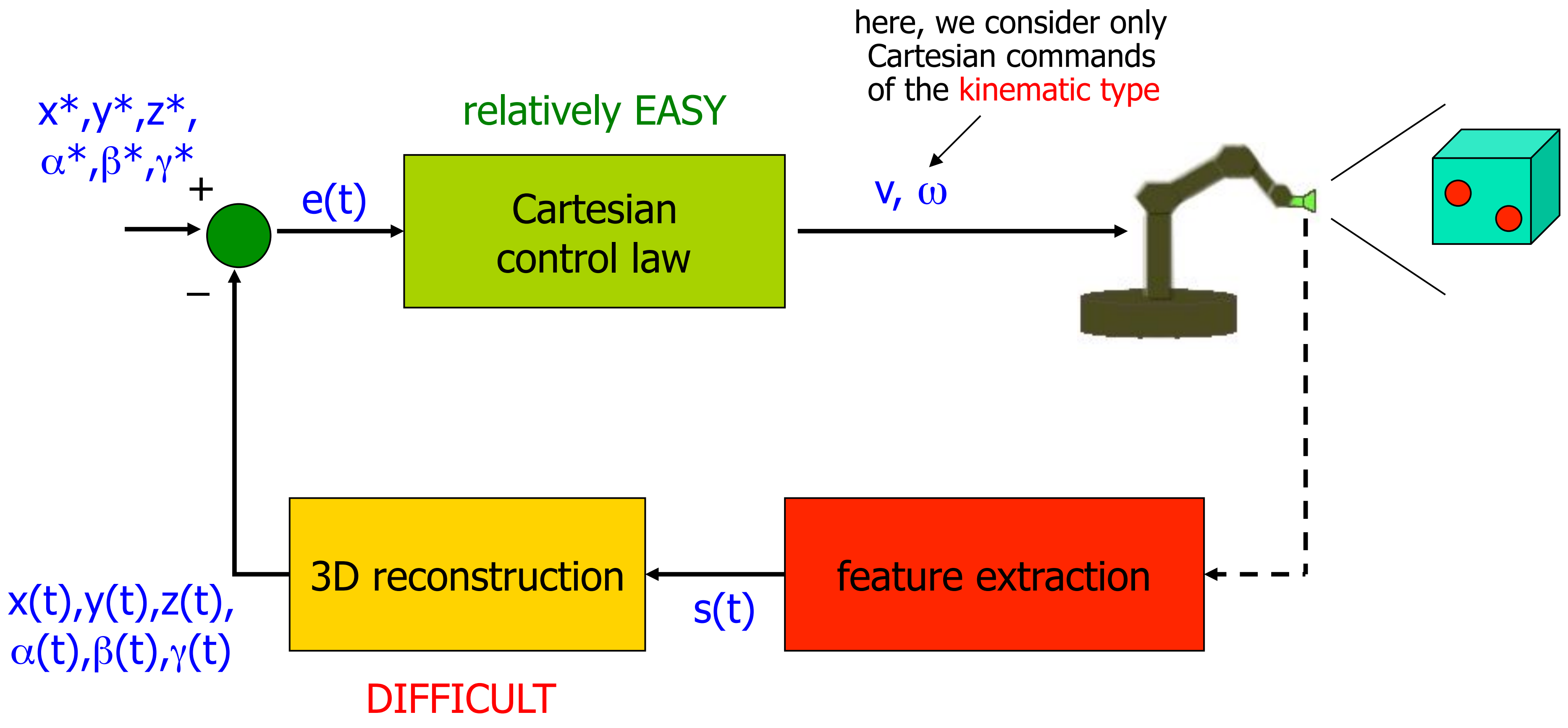
- position-based visual servoing (**PBVS**)



- image-based visual servoing (**IBVS**)



PBVS architecture



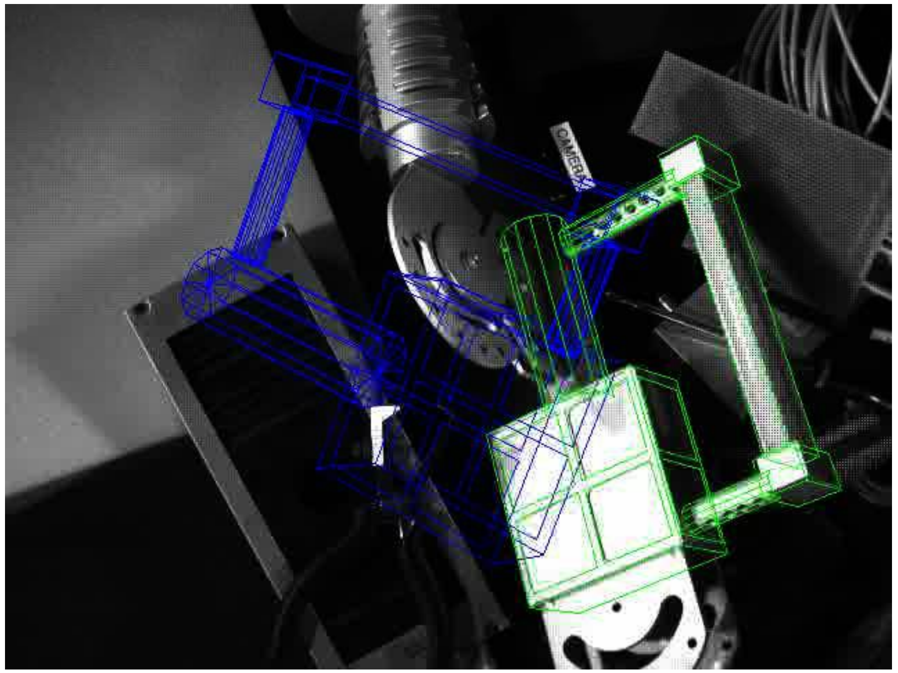
highly sensitive to camera calibration parameters

Examples of PBVS

video

eye-to-"robot" (SuperMario)

video

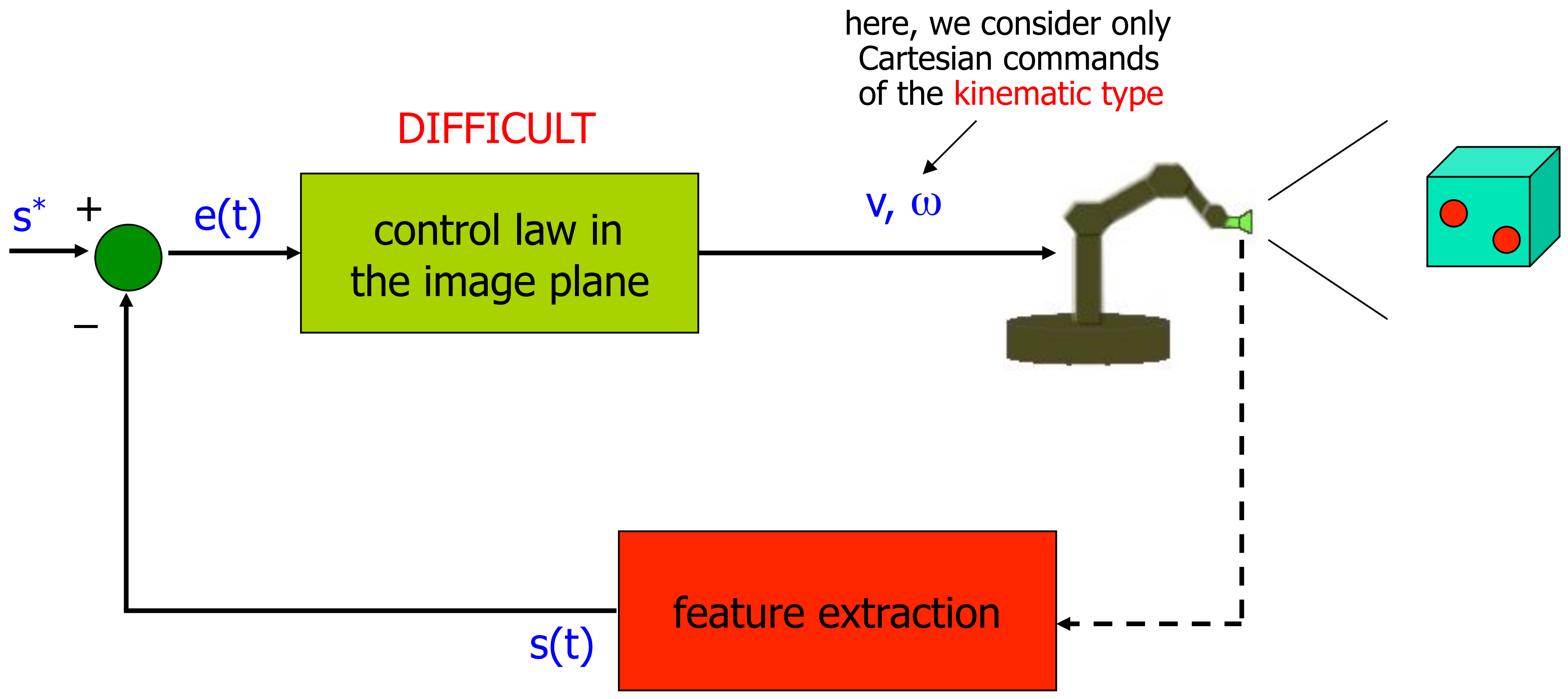


eye-in-hand (Puma robot)

position/orientation of the camera and scene geometry → position and orientation of the robot (with mobile or fixed base)

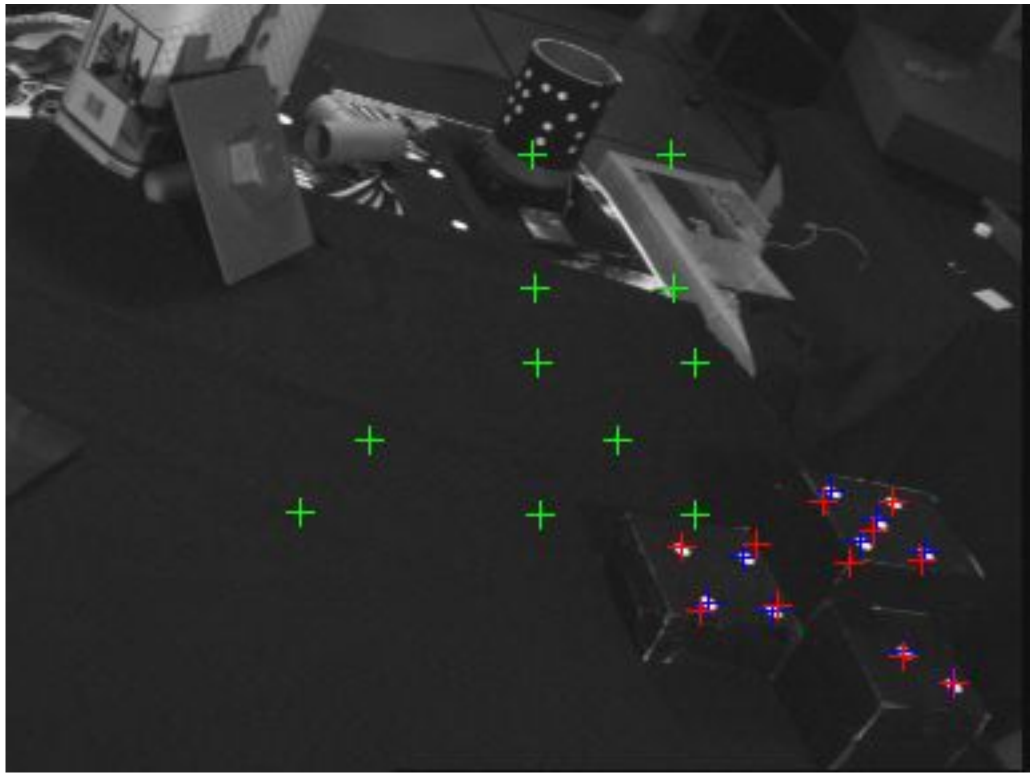
known a priori!

IBVS architecture



almost insensitive to intrinsic/extrinsic camera calibration parameters

An example of IBVS



video

here, the features are points (selected from the given set, or in suitable combinations)

desired feature positions

current feature positions

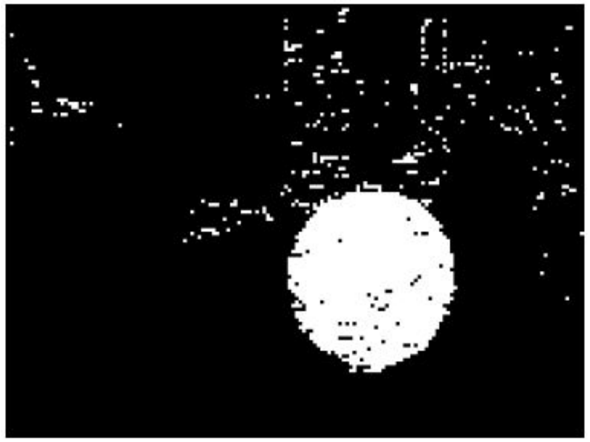
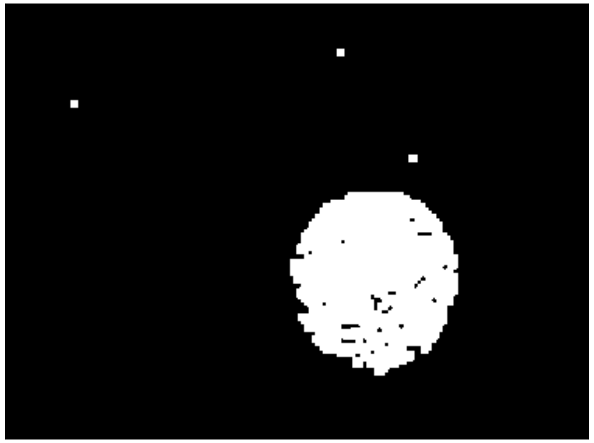
$\longrightarrow$

the error in the image plane (task space!) drives/controls the motion of the robot

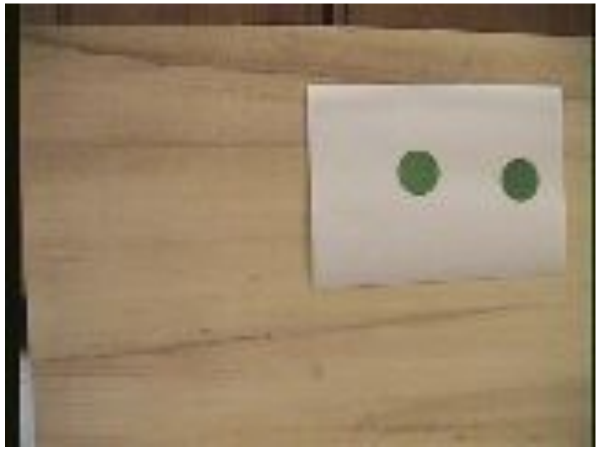
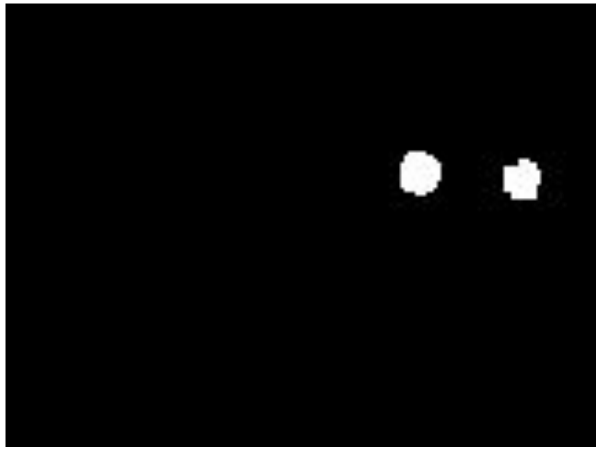
Steps in an IBVS scheme

- image acquisition
  - frame rate, delay, noise, ...
- feature extraction
  - with image processing techniques (it could be a difficult and time consuming step!)
- comparison with “desired” feature values
  - definition of an error signal in the image plane
- generation of motion of the camera/robot
  - perspective transformations
  - differential kinematics of the manipulator
  - control laws of kinematic (most often) or dynamic type (e.g., PD + gravity cancelation --- see reference textbook)

Image processing techniques

binarization in RGB space → erosion and dilation

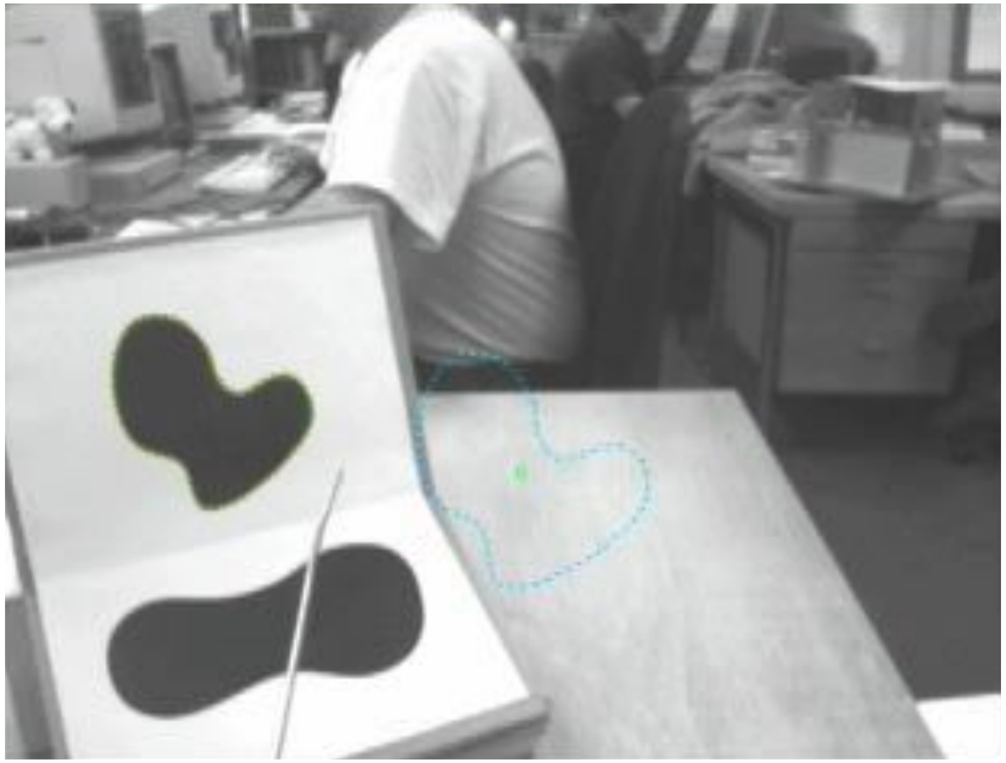



binarization in HSV space → dilation

What is a feature?

- image feature: any interesting characteristic or geometric structure extracted from the image/scene
  - points
  - lines
  - ellipses (or any other 2D contour)
- feature parameter(s): any numerical quantity associated to the selected feature in the image plane
  - coordinates of a point
  - angular coefficient and offset of a line
  - center and radius of a circle
  - area and center of a 2D contour
  - generalized moments of an object in the image
    - can also be defined so as to be scale- and rotation-invariant

Example of IBVS using moments



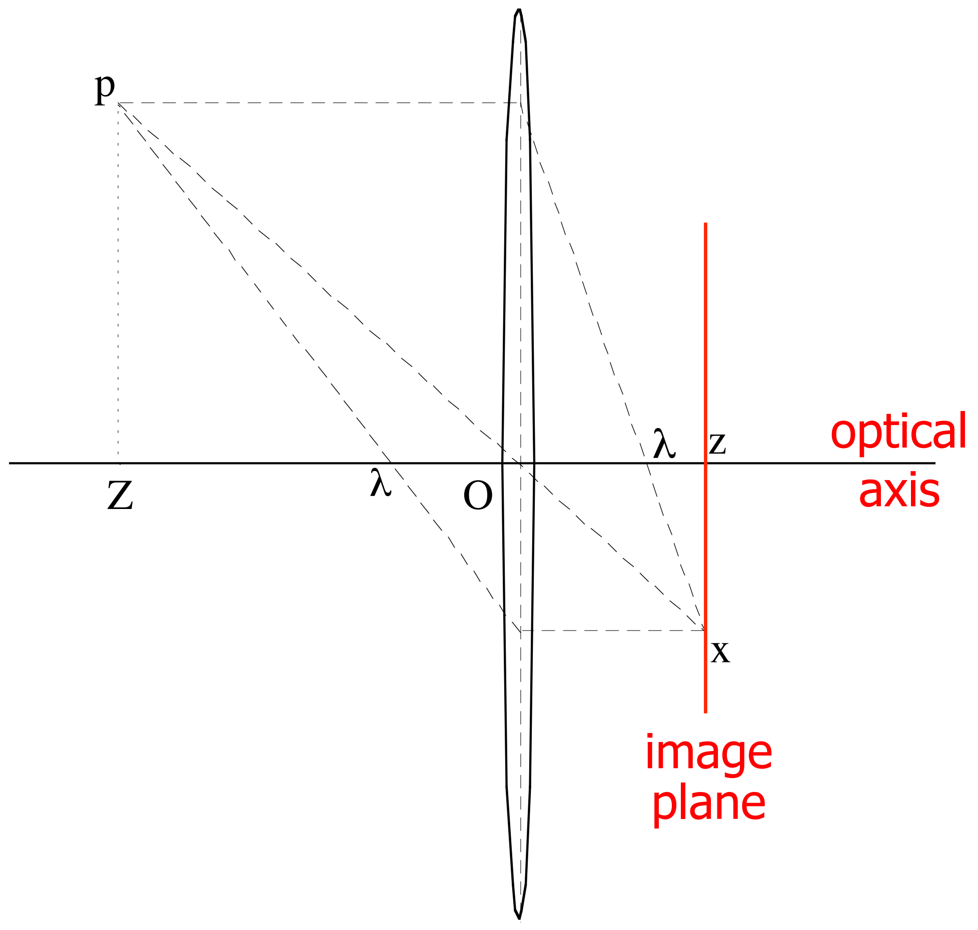
video

- avoids the problem of “finding correspondences” between points
- however, it is not easy to control the motion of all 6 dofs of the camera when using only moments as features

Camera model

- set of lenses to focus the incoming light
  - (converging) thin lenses
  - pinhole approximation
  - catadioptric lens or systems (combined with mirrors)
- matrix of light sensitive elements (pixels in image plane), possibly selective to RGB colors ~ human eye
- frame grabber "takes shots": an electronic device that captures individual, digital still frames as output from an analog video signal or a digital video stream
  - board + software on PC
  - frame rate = output frequency of new digital frames
  - it is useful to randomly access a subset (area) of pixels

Thin lens camera model



- geometric/projection optics
- rays parallel to the optical axis are deflected through a point at distance λ (focal length)
- rays passing through the optical center O are not deflected

fundamental equation of a thin lens

$$\underbrace{\frac{1}{Z}+\frac{1}{z}=\frac{1}{\lambda}}_{\text{dioptric (optical) power}}$$

dioptric (optical) power

proof? left as exercise ...

Pinhole camera model

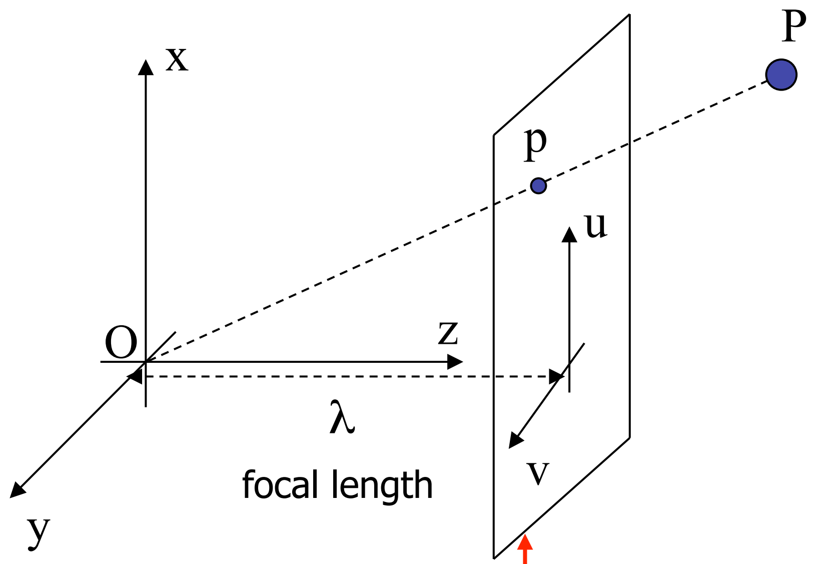


image plane with an array of light sensitive elements (actually located symmetrically to the optical center O, but this model avoids a change of signs...)

- when thin lens wideness is neglected
- all rays pass through optical center
- from the relations on similar triangles one gets the perspective equations

$$u = \lambda \frac{X}{Z} \qquad v = \lambda \frac{Y}{Z}$$

to obtain these from discrete pixel values, an offset and a scaling factor should be included in each direction

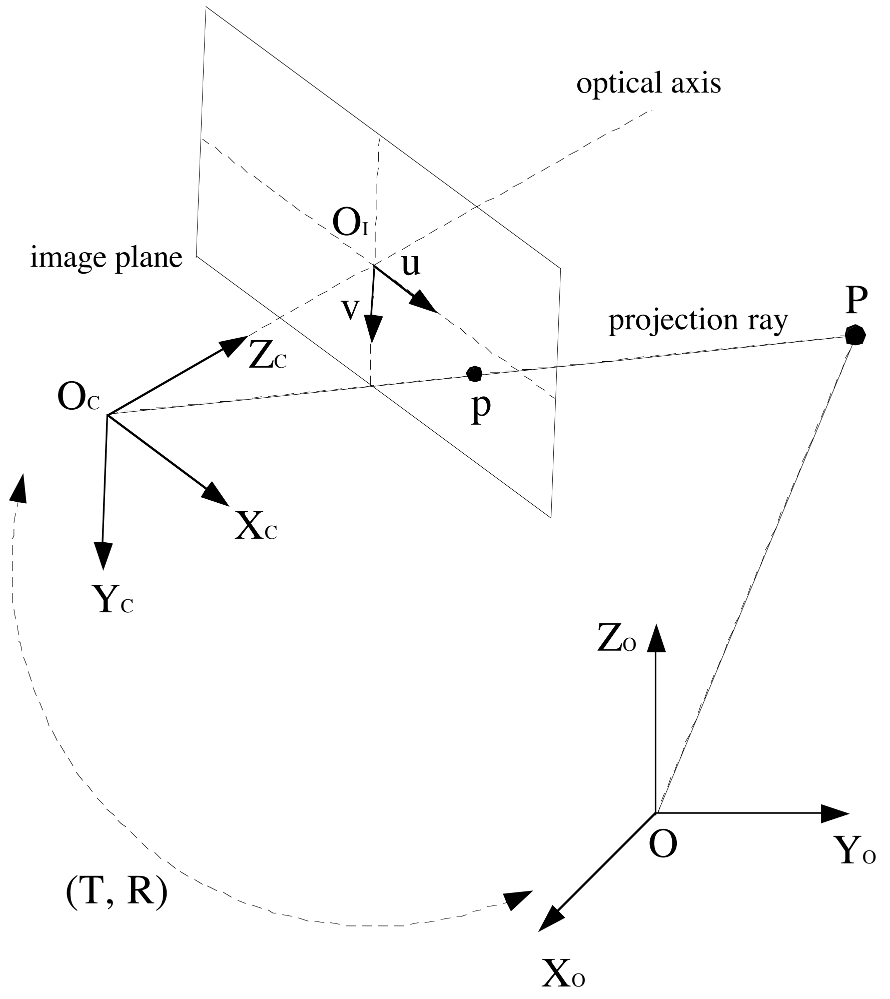
sometimes normalized values (u′,v′) are used (dividing by the focal length)

with

$$P = (X,\, Y,\, Z)$$ Cartesian point (in camera frame)

$$p = (u, v, \lambda)$$ representative point on the image plane

Reference frames of interest



- absolute reference frame

$$\mathcal{F}_0 : \left\{O,\, \overrightarrow{X}_0,\, \overrightarrow{Y}_0,\, \overrightarrow{Z}_0\right\}$$

- camera reference frame

$$\mathcal{F}_C : \left\{O_C,\, \overrightarrow{X}_C,\, \overrightarrow{Y}_C,\, \overrightarrow{Z}_C\right\}$$

- image plane reference frame

$$\mathcal{F}_I : \{O_I,\, \overrightarrow{u},\, \overrightarrow{v}\}$$

- position/orientation of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_0$

$$(T,\, R)$$

(translation, rotation)

Interaction matrix

- given a set of feature(s) parameters $f = [f_1 \dots f_k]^T \in \mathbb{R}^k$
- we look for the (kinematic) differential relation between motion imposed to the camera and motion of features on the image plane

$$\dot{f} = J(\cdot) \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

- $(V, \Omega) \in \mathbb{R}^6$ is the camera linear/angular velocity, a vector expressed in $\mathcal{F}_C$
- $J(\cdot)$ is a k*6 matrix, known as interaction matrix
- in the following, we consider mainly point features
  - other instances (areas, lines, …) can be treated as extensions of the presented analysis

Computing the interaction matrix
point feature, pinhole model

- from the perspective equations $u = \lambda \frac{X}{Z},\ v = \lambda \frac{Y}{Z}$, we have

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{Z} & 0 & -\frac{u}{Z} \\ 0 & \frac{\lambda}{Z} & -\frac{v}{Z} \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = J_1(u, v, \lambda) \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix}$$

- the velocity $\left(\dot{X}, \dot{Y}, \dot{Z}\right)$ of a point P in frame $\mathcal{F}_C$ is actually due to the roto-translation $(V,\ \Omega)$ of the camera (P is assumed fixed in $\mathcal{F}_0$)
- kinematic relation between $\left(\dot{X}, \dot{Y}, \dot{Z}\right)$ and $(V,\ \Omega)$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = -V - \Omega \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Note: ALL quantities are expressed in the camera frame $\mathcal{F}_C$
without explicit indication of subscripts

Computing the interaction matrix (cont)
point feature, pinhole model

- the last equation can be expressed in matrix form

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 & -Z & Y \\ 0 & -1 & 0 & Z & 0 & -X \\ 0 & 0 & -1 & -Y & X & 0 \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix} = J_2(X,Y,Z) \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

- combining things, the interaction matrix for a point feature is

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = J_1 J_2 \begin{bmatrix} V \\ \Omega \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{Z} & 0 & \frac{u}{Z} & \frac{uv}{\lambda} & -\left(\lambda + \frac{u^2}{\lambda}\right) & v \\ 0 & -\frac{\lambda}{Z} & \frac{v}{Z} & \lambda + \frac{v^2}{\lambda} & -\frac{uv}{\lambda} & -u \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

$$= \underset{\uparrow}{J_p}(u,v,Z) \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

here, λ is assumed to be known

p = point (feature)

Comments

- the interaction matrix in the map

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \boxed{J_p(u, v, Z)} \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

  - depends on the actual value of the feature and on its depth Z
  - since $\dim \ker J_p = 4$, there exist ∞^4 motions of the camera that are unobservable (for this feature) on the image
    - for instance, a translation along the projection ray
  - when more point features are considered, the relative single interaction matrices are stacked one on the other
    - p point features: the interaction matrix is k × 6, with k = 2p

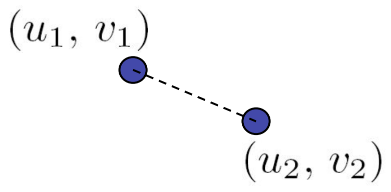
Other examples of interaction matrices

- distance between two point features

$$d = \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}$$

$$\dot{d} = \frac{1}{d} \begin{bmatrix} u_1 - u_2 & v_1 - v_2 & u_2 - u_1 & v_2 - v_1 \end{bmatrix} \begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \end{bmatrix}$$

$$= J_d(u_1, u_2, v_1, v_2) \begin{bmatrix} J_{p1}(u_1, v_1, Z_1) \\ J_{p2}(u_2, v_2, Z_2) \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix}$$



- image moments

$$m_{ij} = \iint_{\mathcal{R}(t)} x^i y^j \, dx dy$$

$\mathcal{R}(t)$ region of the image plane occupied by the object

  - useful for representing also qualitative geometric information (area, center, orientation of an approximating ellipse, …)
  - by using Green formulas and the interaction matrix of a generic point feature

➡ $$\dot{m}_{ij} = L_{ij} \begin{bmatrix} V \\ \Omega \end{bmatrix}$$

(see F. Chaumette: “Image moments: A general and useful set of features for visual servoing”, IEEE Trans. on Robotics, August 2004)

Robot differential kinematics

- eye-in-hand case: camera is mounted on the end-effector of a manipulator arm (with fixed base or on a wheeled mobile base)
- the motion (V, Ω) to be imposed to the camera coincides with the desired end-effector linear/angular velocity and is realized by choosing the manipulator joint velocity (or, more in general, the feasible velocity commands of a mobile manipulator)

$$\begin{bmatrix} V \\ \Omega \end{bmatrix} = J_m(q)\dot{q} = J_m(q)\boldsymbol{u} \longleftarrow \text{velocity control input}$$

for consistency with the previous interaction matrix, also these Jacobians must be expressed in the camera frame $\mathcal{F}_C$ $\left\{ \right.$ Geometric Jacobian of a manipulator ... or the NMM Jacobian of a mobile manipulator

with $q \in \mathbb{R}^n$ being the robot configuration variables

- in general, an hand-eye calibration is needed for this Jacobian

Image Jacobian

- combining the step involved in the passage from the velocity of point features on the image plane to the joint velocity/feasible velocity commands of the robot, we finally obtain

$$\dot{f} = J_p(f,\, Z)\, J_m(q)\, \boldsymbol{u} \;= J(f,\, Z,\, q)\, \boldsymbol{u}$$

- matrix $J(f,\, Z,\, q)$ is called the Image Jacobian and plays the same role of a classical robot Jacobian
- we can thus apply one of the many standard kinematic (or even dynamics-based) control techniques for robot motion
- the (controlled) output is given by the vector of features in the image plane (the task space!)
- the error driving the control law is defined in this task space

Kinematic control for IBVS

- defining the error vector as $e = f_d - f$, the general choice

$$\boldsymbol{u} = \underbrace{J^{\#}}_{\text{minimum norm solution}}(\dot{f}_d + k\,e) + \underbrace{(I - J^{\#}J)\,\boldsymbol{u}_0}_{\text{term in } \ker J \text{ does not ``move'' the features}}$$

  will exponentially stabilize the task error to zero (up to singularities, limit joint range/field of view, features exiting the image plane, ...)
- in the redundant case, vector $\boldsymbol{u}_0$ (projected in the image Jacobian null space) can be used for optimizing behavior/criteria
- the error feedback signal depends only on feature values
- there is still a dependence of J on the depths Z of the features
  - use the constant and "known" values at the final desired pose

$$J(f,\, Z^*,\, q)$$

  - or, estimate on line their current values using a dynamic observer

Example with NMM

- mobile base (unicycle) + 3R manipulator
- eye-in-hand configuration

- 5 commands
  - linear and angular velocity of mobile base
  - velocities of the three manipulator joints
- task specified by 2 point features $\longrightarrow$ 4 output variables

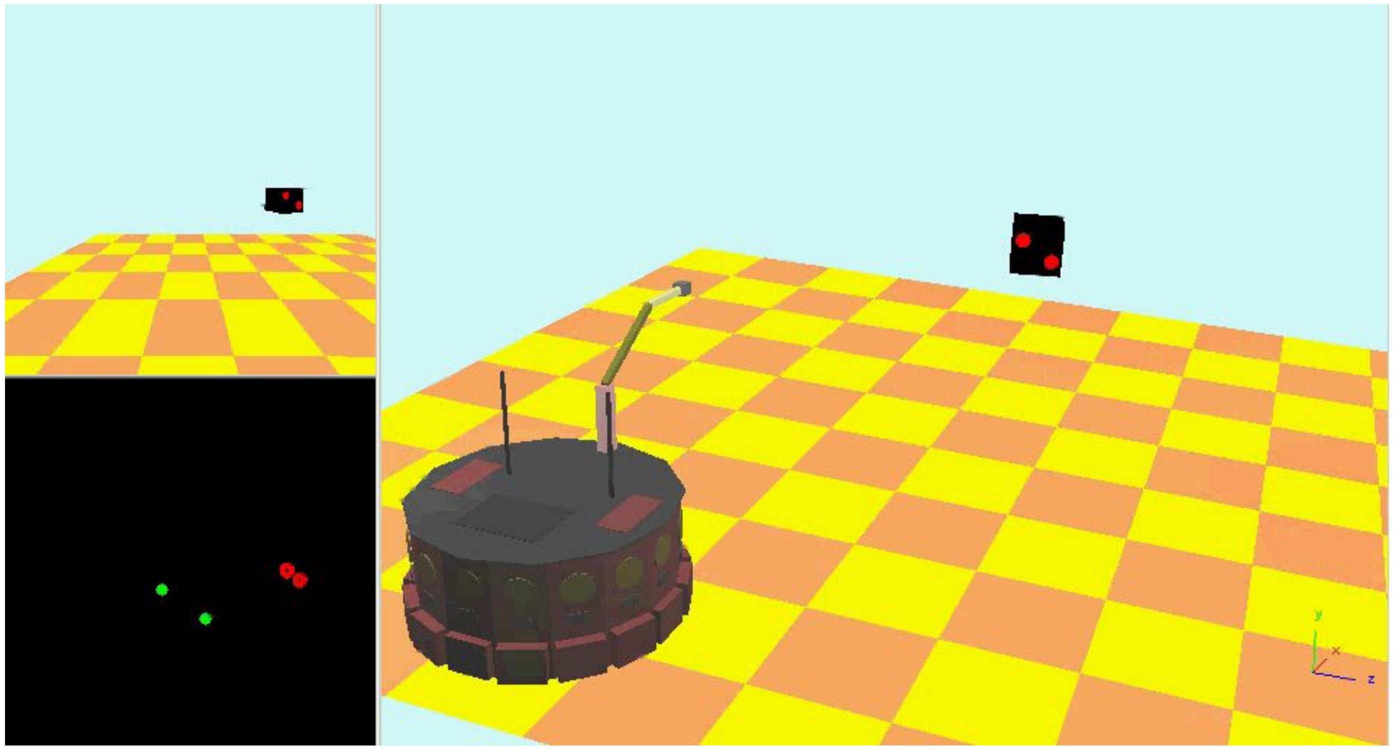
$$f = [f_{u1}\ f_{v1}\ f_{u2}\ f_{v2}]^T$$

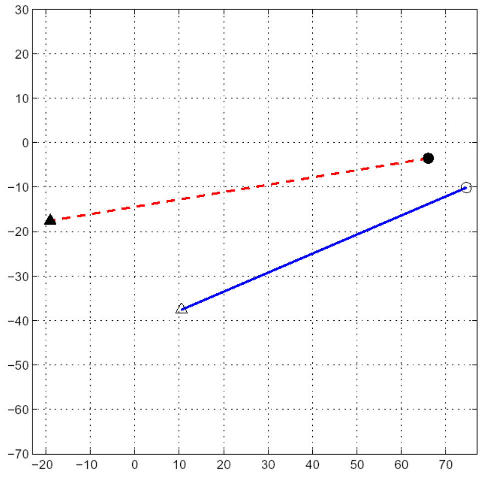
- 5 - 4 = 1 degree of redundancy (w.r.t. the task)

Simulation

view from camera

processed image





behavior of the 2 features

video

- simulation in Webots
- current value of Z is supposed to be known
- diagonal task gain matrix k (decoupling!)
- "linear paths" of features motion on the image plane

IBVS control with Task Sequencing

- approach: regulate only one (some) feature at the time, while keeping "fixed" the others by unobservable motions in the image plane
- Image Jacobians of single point features (e.g., p=2)

$$\dot{f}_1 = J_1 \boldsymbol{u}, \quad \dot{f}_2 = J_2 \boldsymbol{u}$$

- the first feature f_1 is regulated during a first phase by using

$$\boldsymbol{u} = J_1^{\#} k_1 e_1 + (I - J_1^{\#} J_1) \boldsymbol{u_0}$$

- feature f_2 is then regulated by a command in the null-space of the first task

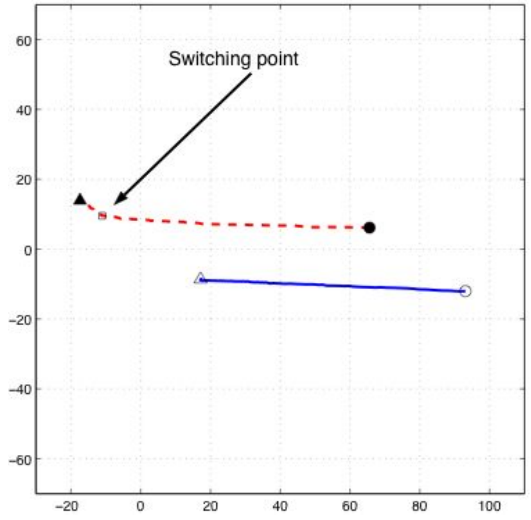
$$\boldsymbol{u} = (I - J_1^{\#} J_1)\, J_2^T k_2 e_2$$

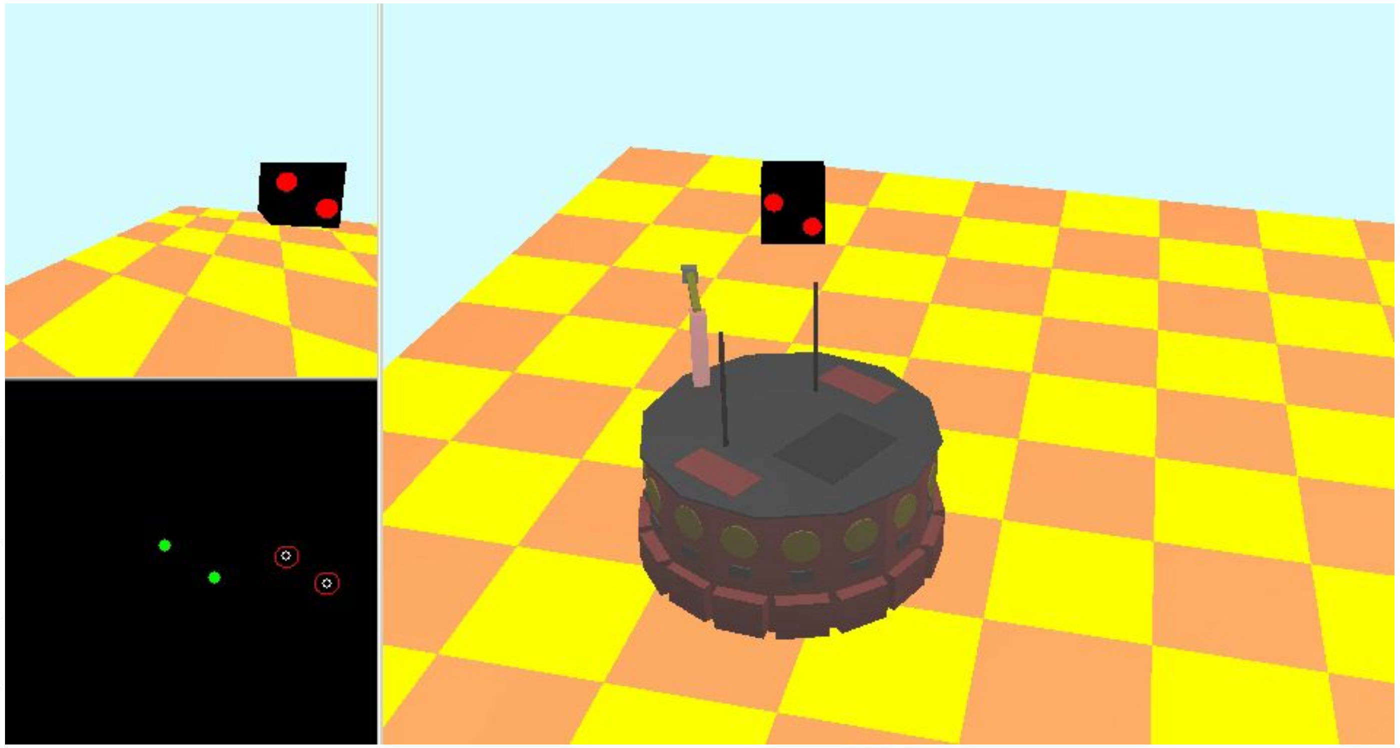
- in the first phase there are two (more) degrees of redundancy w.r.t. the "complete" case, which can be used, e.g., for improving robot alignment to the target
- if the complete case is not redundant: singularities are typically met without Task Sequencing, but possibly prevented with TS ...

Simulation with TS scheme

mobile base (unicycle) + polar 2R arm: Image Jacobian is square (4×4)

point feature 1
(regulated in
first phase)
point feature 2
(in second phase)





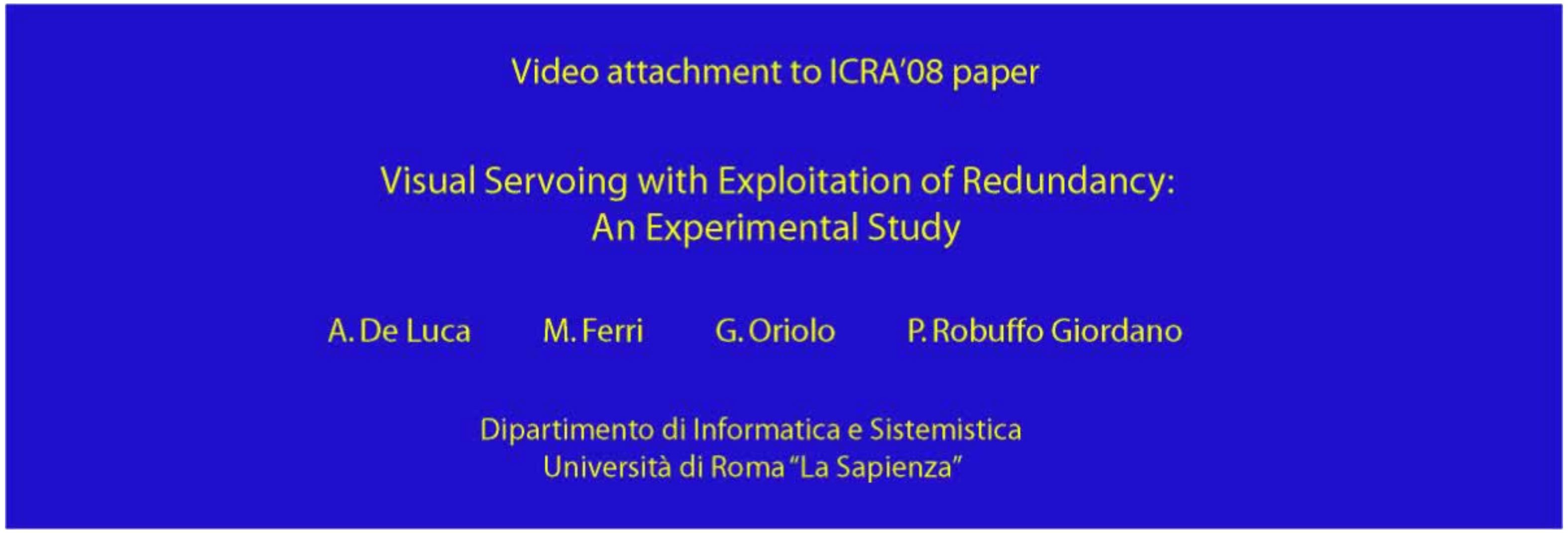
video

behavior of
the 2 features

- simulation in Webots
- current value of *Z* is supposed to be known

Experiments

Magellan (unicycle) + pan-tilt camera (same mobility of a polar 2R robot)



Video attachment to ICRA'08 paper

Visual Servoing with Exploitation of Redundancy:
An Experimental Study

A. De Luca M. Ferri G. Oriolo P. Robuffo Giordano

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"

video

- comparison between Task Priority (TP) and Task Sequencing (TS) schemes
- both can handle singularities (of Image Jacobian) that are possibly encountered
- camera frame rate = 7 Hz

In many practical cases…

- the uncertainty in a number of relevant data may be large
  - focal length λ (intrinsic camera parameters)
  - hand-eye calibration (extrinsic camera parameters)
  - depth Z of point features
  - ….
- one can only compute an approximation of the Image Jacobian (both in its interaction matrix part, as well as in the robot Jacobian part)
- in the closed loop, error dynamics on features becomes $\dot{e} = -J\hat{J}^{\#}Ke$
  - ideal case: $JJ^{\#} = I$ real case: $J\hat{J}^{\#} \neq I$
- it is possible to show that a sufficient condition for local convergence of the error to zero is

$$\boxed{J\hat{J}^{\#} > 0}$$

An observer of the depth Z

- it is possible to estimate on line the current value (possibly time-varying) of the depth Z, for each considered point feature, using a dynamic observer
- define $x = \begin{bmatrix} u & v & 1/Z \end{bmatrix}^T$, $\hat{x} = \begin{bmatrix} \hat{u} & \hat{v} & 1/\hat{Z} \end{bmatrix}^T$ as current state and estimated state, and $y = \begin{bmatrix} u & v \end{bmatrix}^T$ as measured output
- a (nonlinear) observer of x

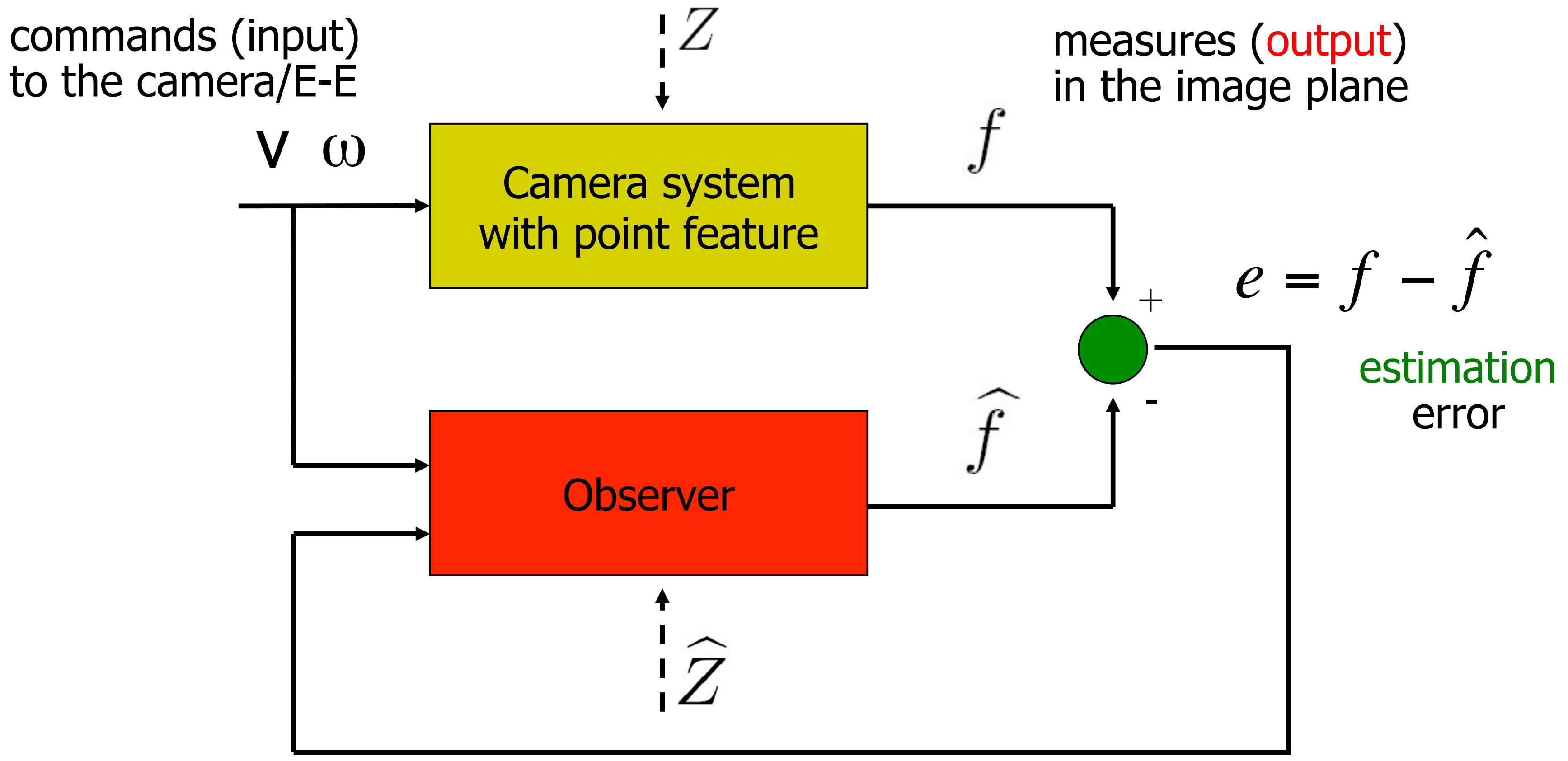
$$\dot{\hat{x}} = \alpha(\hat{x}, y)\,\boldsymbol{u} + \beta(\hat{x}, y, \boldsymbol{u})$$

guarantees $\lim_{t \to \infty} \left\| x(t) - \hat{x}(t) \right\| = 0$ provided that

  - linear velocity of the camera is not zero
  - the linear velocity vector is not aligned with the projection ray of the considered point feature

  $\Rightarrow$ these are persistent excitation conditions (~ observability conditions)

Block diagram of the observer



commands (input) to the camera/E-E
V ω
Z
Camera system with point feature
f
measures (output) in the image plane
+
-
$e = f - \hat{f}$
estimation error
$\hat{f}$
Observer
$\hat{Z}$

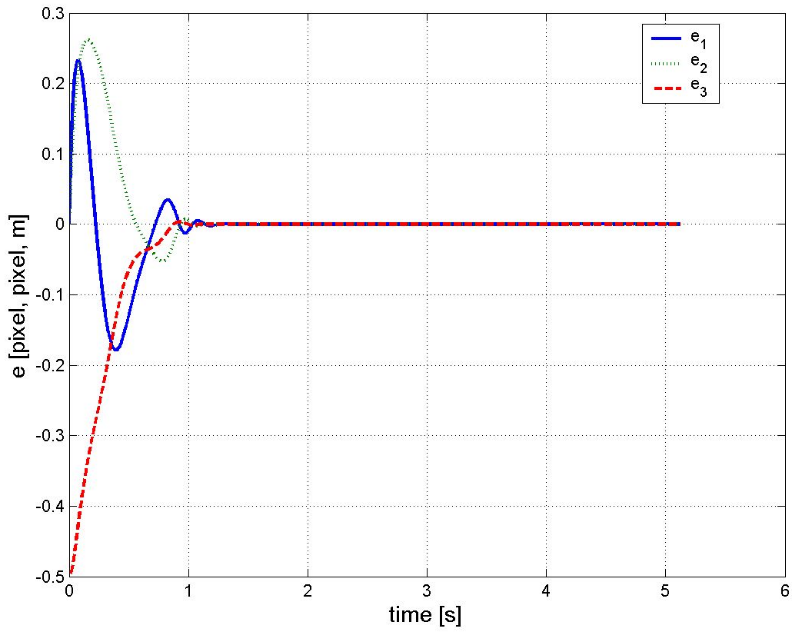
Results on the estimation of Z

real and estimated initial state

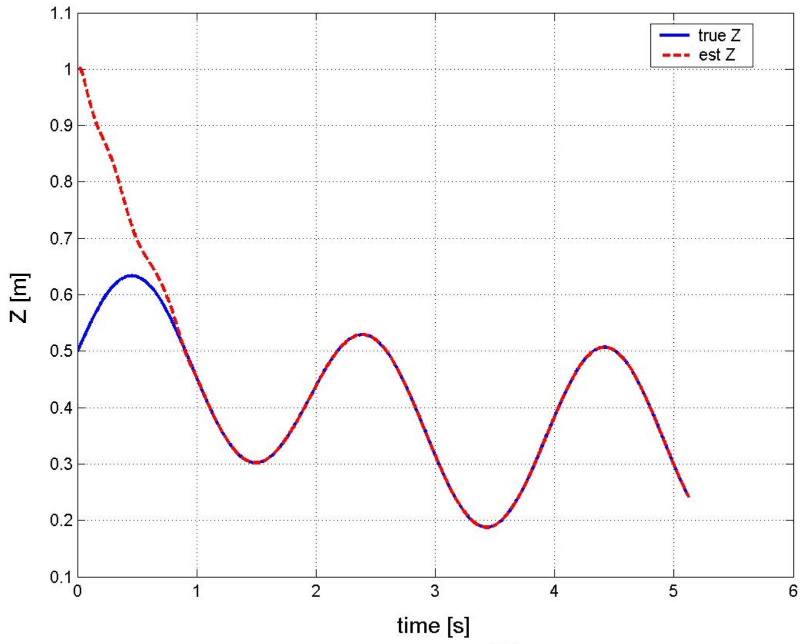
$$\begin{aligned} x(t_0) &= [10 \quad -10 \quad 2]^T \\ \widehat{x}(t_0) &= [10 \quad -10 \quad 1]^T \end{aligned}$$

$$\begin{aligned} v_x(t) &= 0.1\cos 2\pi t \\ v_z(t) &= 0.5\cos \pi t \\ \omega_x(t) &= 0.6\cos \pi/2\, t \\ \omega_z(t) &= 1 \end{aligned}$$

open-loop commands

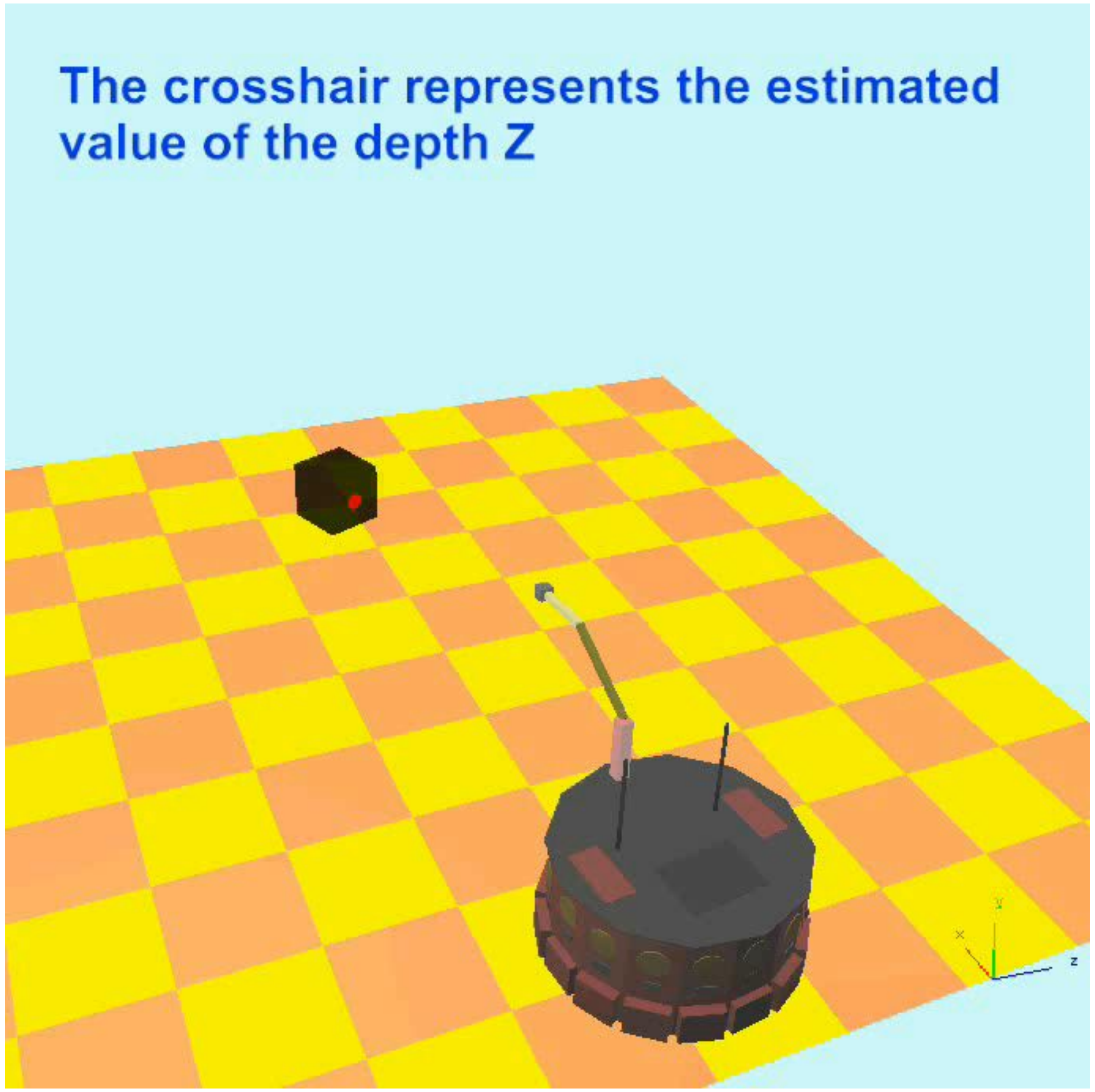


estimation errors $e(t)$



$Z(t)$ and $\widehat{Z}(t)$

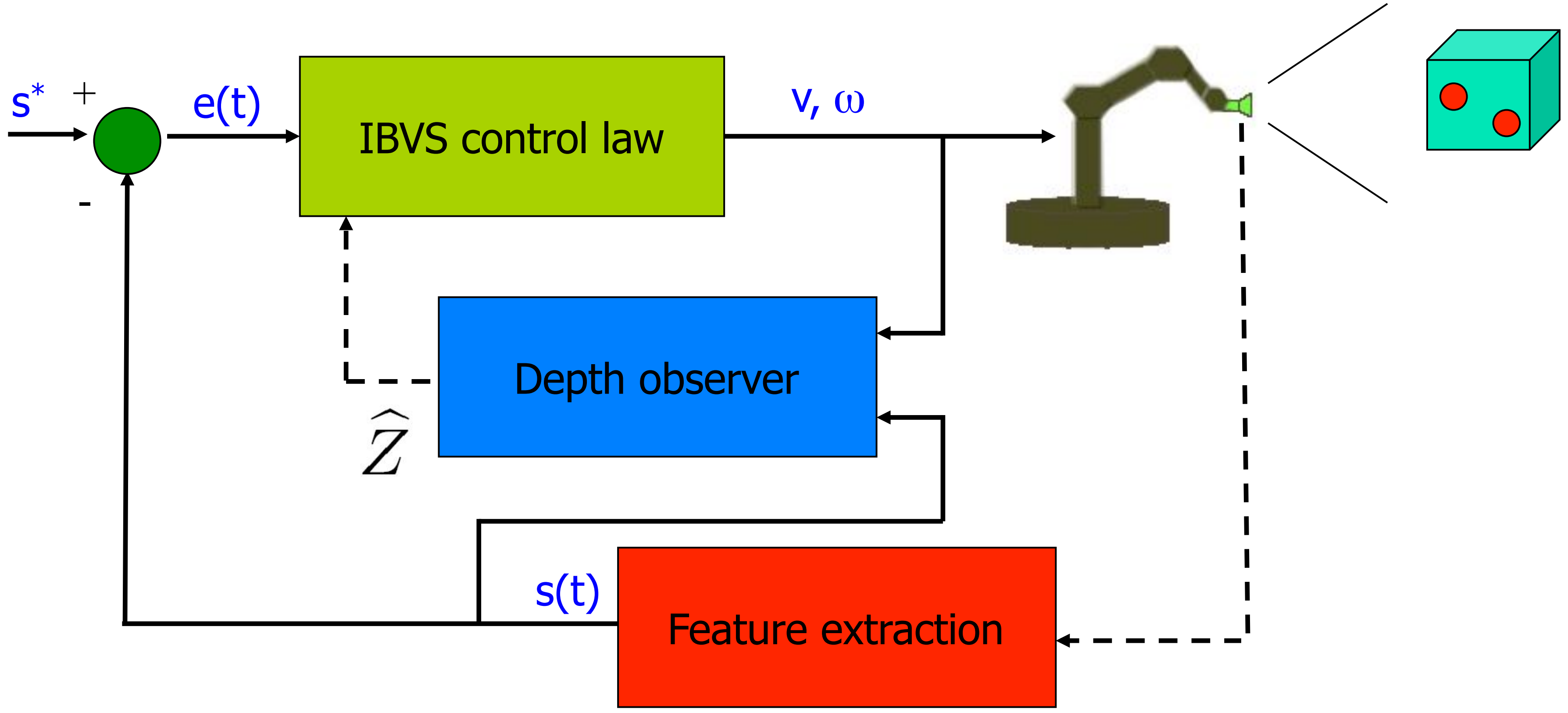
Simulation of the observer
with stepwise displacements of the target



video

IBVS control with observer

- the depth observer can be easily integrated on line with any IBVS control scheme

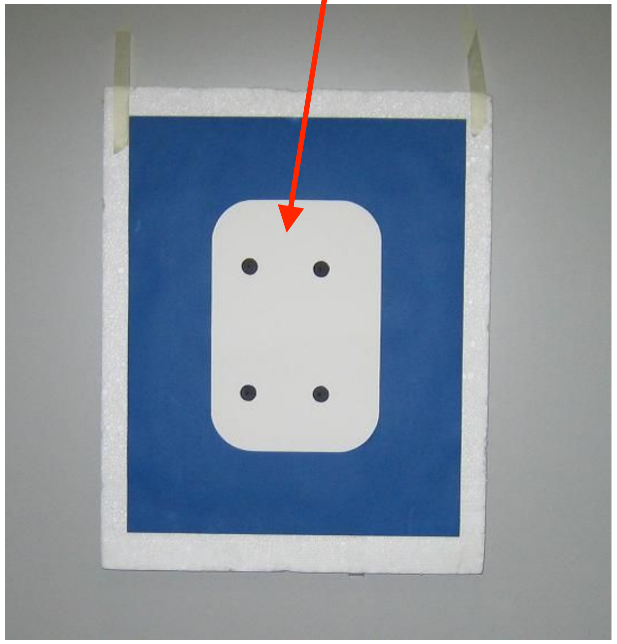


Experimental set-up

- visual servoing with fixed camera on a skid-steering mobile robot
- very rough eye-robot calibration
- unknown depths of target points (estimated on line)

camera

4 target points (on a plane)



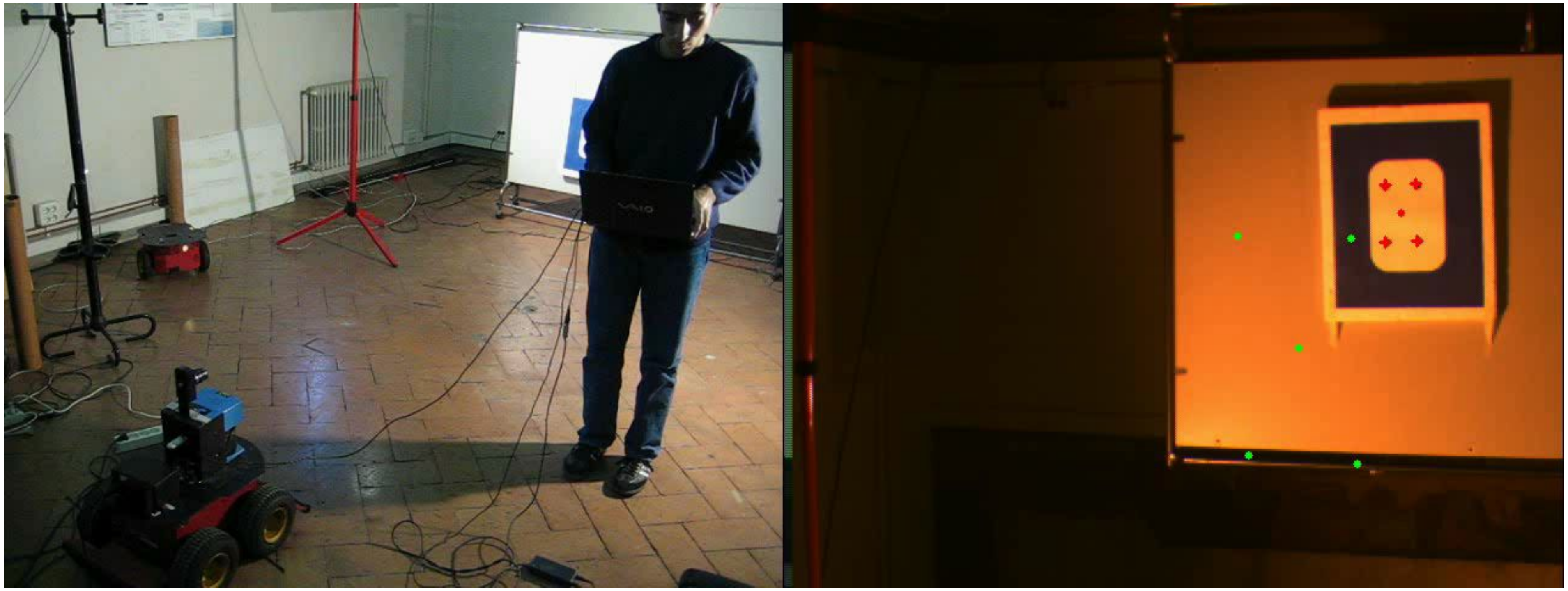
a "virtual" 5th feature is also used
as the average of the four point features

Experiments

- motion of features on the image plane is not perfect...
- the visual regulation task is however correctly realized

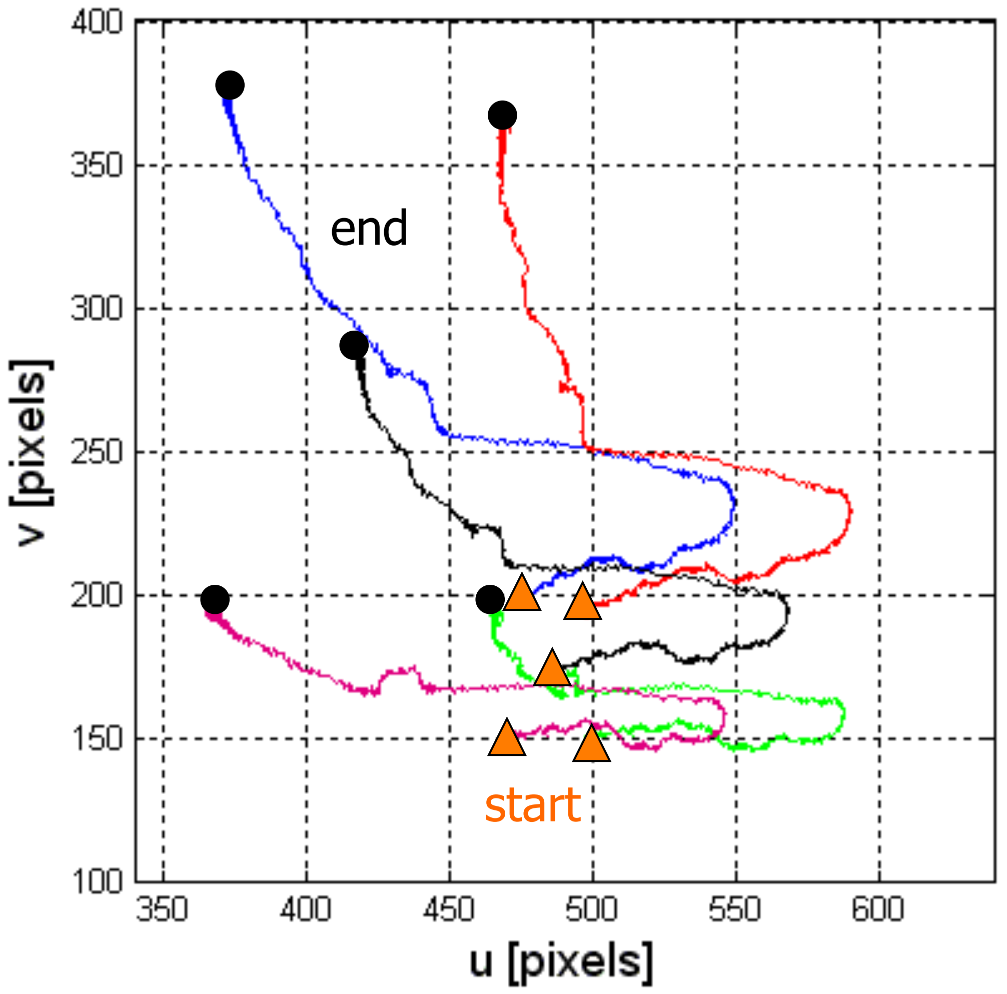
external view

camera view



video (c/o Università di Siena)

Evolution of features



- motion of the 5 features (including the average point) in the image plane
- toward end, motion is ≈ linear since the depth observers have already converged to the true values
- computed Image Jacobian is close to the actual one

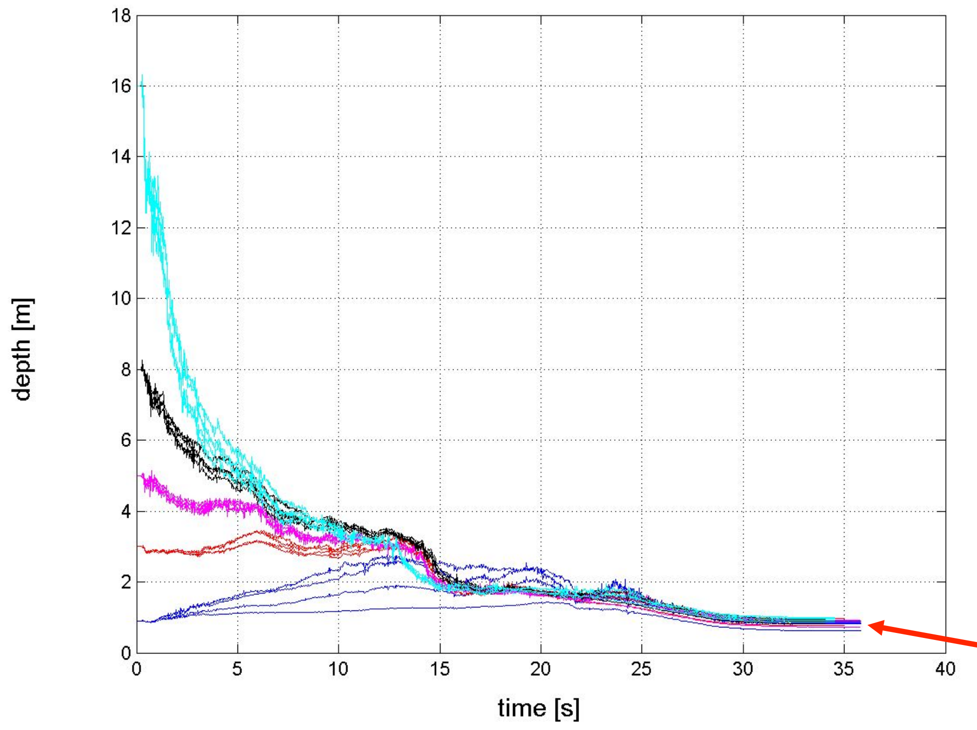
$$\widehat{Z} \rightarrow Z \Rightarrow \widehat{J} \rightarrow J$$

- "true" initial and final depths

$$Z(t_0) \simeq 4\,m \quad Z_d \simeq 0.9\,m$$

Experiments with the observer

- the same task was executed with five different initializations for the depth observers, ranging between 0.9 m (= true depth in the final desired pose) and 16 m (much larger than in the true initial pose)



the evolutions of the estimated depths for the 4 point features

- initial values of depth estimates in the five tests

$$\begin{cases} \widehat{Z}_1(t_0) = 16\,m & 🟦 \\ \widehat{Z}_2(t_0) = 8\,m & ⬛ \\ \widehat{Z}_3(t_0) = 5\,m & 🟪 \\ \widehat{Z}_4(t_0) = 3\,m & 🟥 \\ \widehat{Z}_5(t_0) = 0.9\,m & 🟦 \end{cases}$$

- true depths in initial pose

$$Z(t_0) \simeq 4\,m$$

- true depths in final pose

$$Z_d \simeq 0.9\,m$$

Visual servoing with Kuka robot

set-up

- Kuka KR5 sixx R650 manipulator (6 revolute joints, spherical wrist)
- Point Grey Flea©2 CCD camera, eye-in-hand (mounted on a support)
- Kuka KR C2sr low-level controller (RTAI Linux and Sensor Interface)
- image processing and visual servoing on PC (Intel Core2 @2.66GHz)
- high-level control data exchange every 12ms (via UDP protocol)

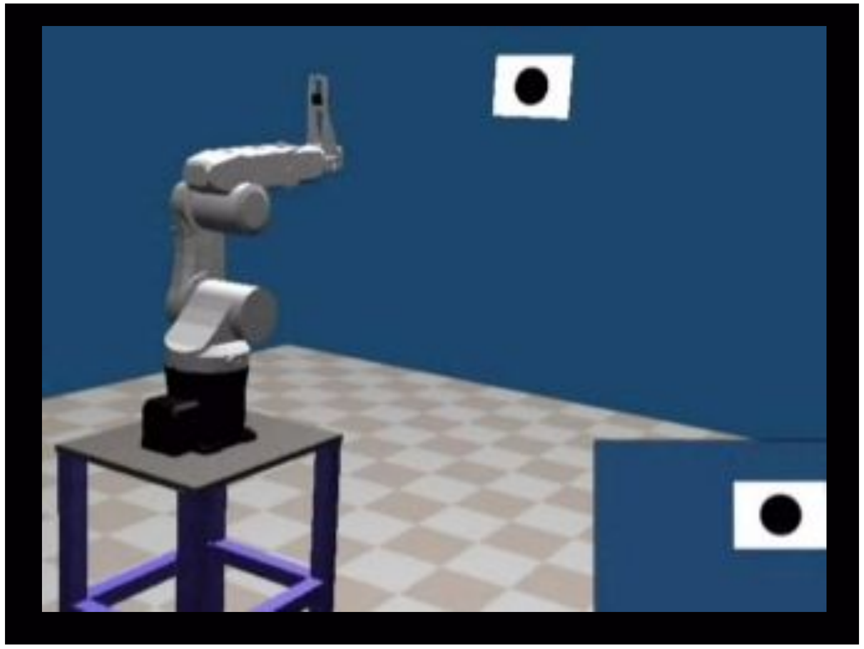
@ DIAG Robotics Laboratory

Visual servoing with Kuka robot

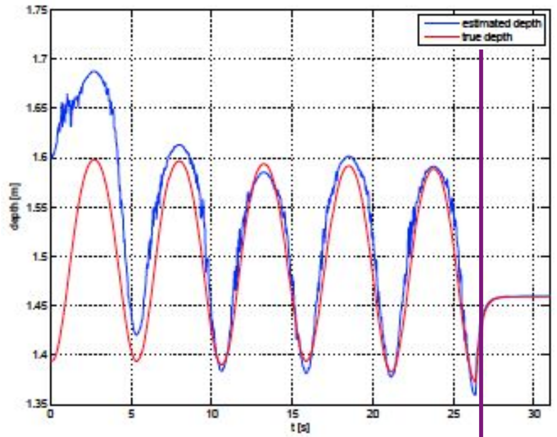
simulation and experiment with the observer

- depth estimation for a fixed target (single point feature)
- simulation using Webots first, then experimental validation
- sinusoidal motion of four robot joints so as to provide sufficient excitation

video

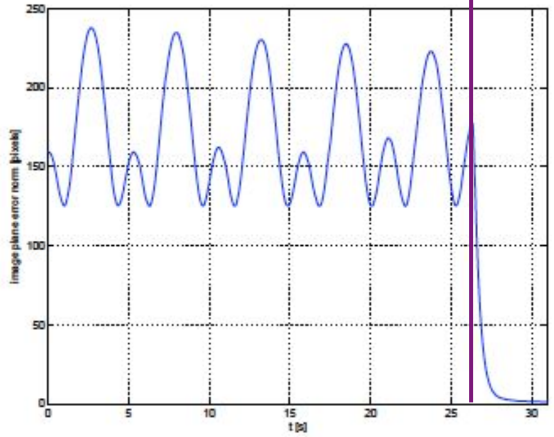


Webots simulation



true and estimated depth

Kuka experiment

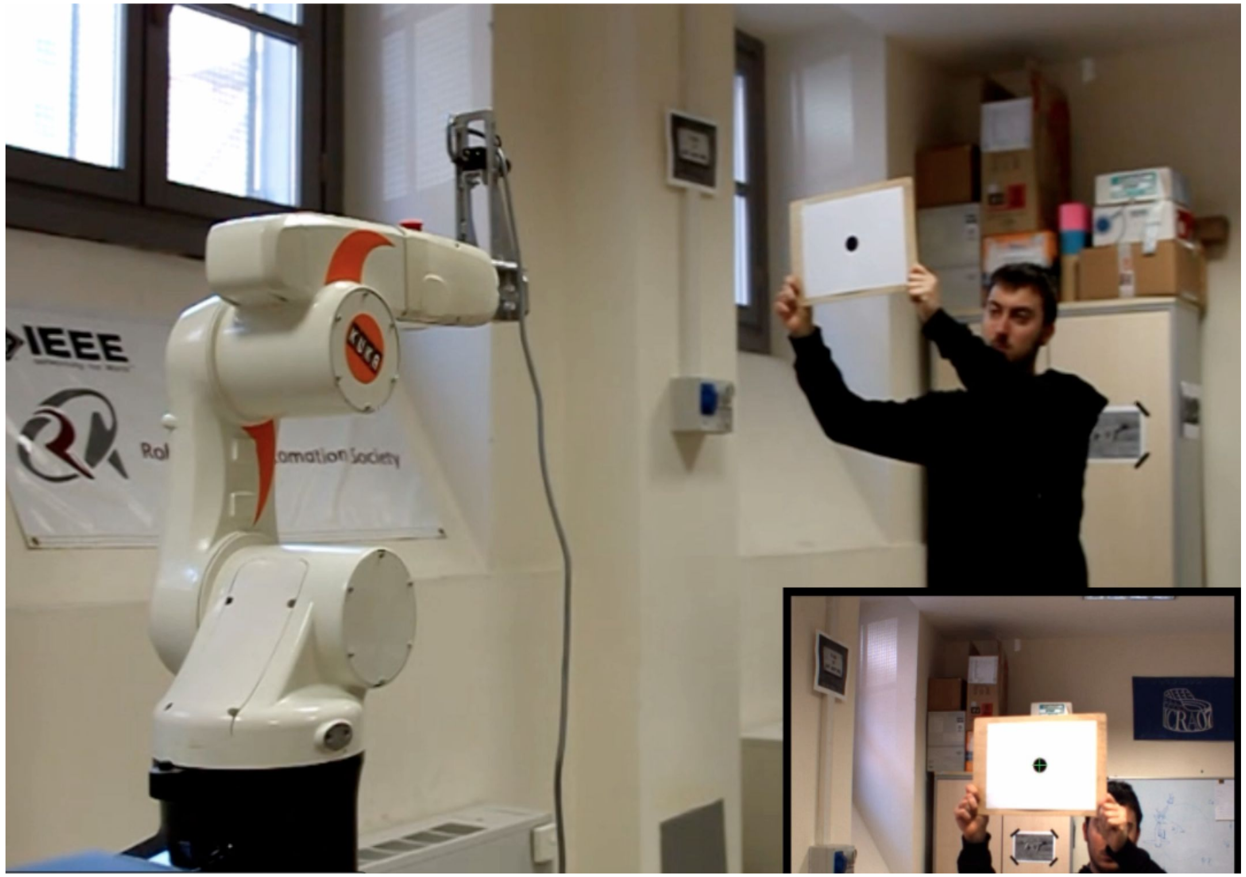


norm of centering error of feature point [in pixels]

goes to zero once control is activated

Visual servoing with Kuka robot
tracking experiment

- tracking a (slowly) time-varying target by visual servoing, including the depth observer (after convergence)



video

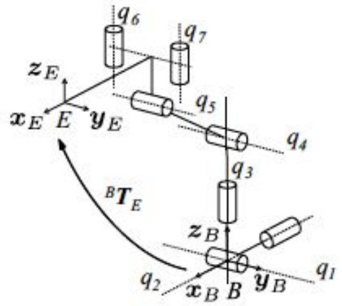
Gazing with humanoid head

stereo vision experiment

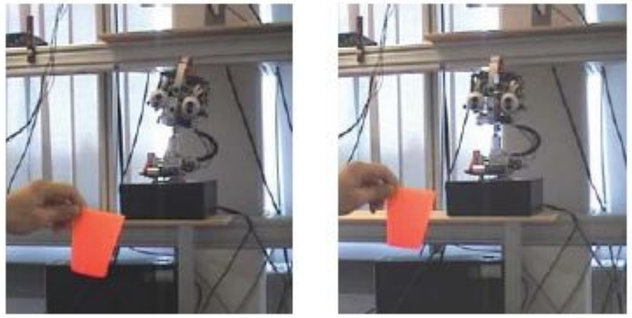
- gazing with a robotized head (7 joints, with n=6 independently controlled) at a moving target (visual task dimension m=2), using redundancy (to keep a better posture and avoid joint limits) and a predictive feedforward

Adaptive Predictive Gaze Control
of a Redundant Humanoid Robot Head
IROS 2011
Giulio Milighetti
Luca Vallone
Alessandro De Luca
Fraunhofer IOSB
SAPIENZA UNIVERSITÀ DI ROMA

video (c/o Fraunhofer IOSB, Karlsruhe)



$$\dot{q} = J_W^{\dagger}(q)\left(\dot{\theta}_{\rm fb} + \dot{\theta}_{\rm ff}\right) - k_0\left(I - J_W^{\dagger}(q)J(q)\right)\nabla_q H_0(q)$$



final head posture
without and with self-motions