Visual Servoing

- Visual measures give geometrical and qualitative information on the environment without physical interaction
  - Employed for high-level control
    - e.g. task planning
  - Employed for low-level control
    - e.g. feedback control

- Look-and-move approach (e.g. visual grasping)
  - Trajectory planned on the visual information
  - Simple motion controller
  - Open loop control → very sensitive to the uncertainties

- Vision-based control approach (also termed visual servoing approach)
  - Visual measurements used in the control loop → robust!
  - Error between the current estimated pose and the end-effector pose

- Camera provides one or more 2D matrices of values of light intensity
  - No direct measures!
  - → Extraction of image feature parameters
  - → Pose estimation (geometric relationships)
  - → Calibration (intrinsic and extrinsic parameters)

- Visual-based control schemes can be divided in the following categories:
  - Position-based visual servoing
  - Image-based visual servoing
  - Hybrid visual servoing

Configuration of the visual system

Number of cameras

- 3D vision or stereo vision
  - More cameras are used to observe the same object
  - Depth can be directly measured
    - Triangulation (e.g. human sight)
- Monocular vision
  - Only one camera is used to observe the object
  - Depth can be estimated
    - The same object is observed by the same camera from different angles
    - Estimation on the basis of certain geometrical characteristics of the object known in advance

Placement of cameras

- Fixed configuration (eye-to-hand)
  - Field of view does not change during the execution of the task
  - Problems about vision occlusions
- Mobile configuration (eye-in-hand)
  - Field of view changes during task execution
  - Occlusions are virtually absent
- Hybrid configuration
- Robotics heads
  - Typically, two cameras mounted on motorized mechanism (pan-tilt cameras)

Image Processing

- Visual information is very rich and varied
  - Complex and computational expensive elaboration
  - Extraction of numerical synthetic information
    - Image feature parameters
      → Image segmentation and interpretation

- Image function
  - Vector function: $\boldsymbol{I}(X, Y)$
  - Defined on the set of pixels of the stored image
  - It represents physical quantities related to the pixel in a sampled and quantized form
    - Light intensity (red, green and blue or shades of gray)

- Color images
  - The image function has three components corresponding to the light intensity in the wavelengths of red, green and blue

- Monochrome black-and-white image
  - The image function is scalar and coincides with the light intensity in shades of gray (gray-level)
  - The number of gray levels depends on the adopted grey-scale resolution
    - e.g. 256 levels → 1 byte of memory

- Gray-level histogram
  - Synthetic representation of the monochrome image
  - It provides the frequency of occurrence of each gray level in the image
  - Normalized histogram (frequencies divided by the total number of pixels)

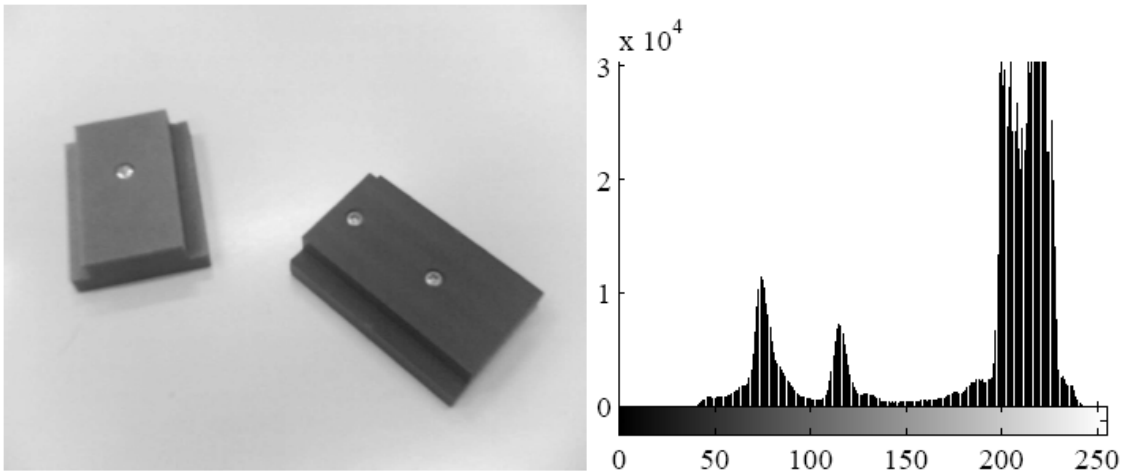


Image segmentation

- Goal: division of the image in groups (referred to as “segments”)
  - Components of each group are similar with respect to one or more characteristics
  - Distant segments correspond to distinct objects or homogenous object parts

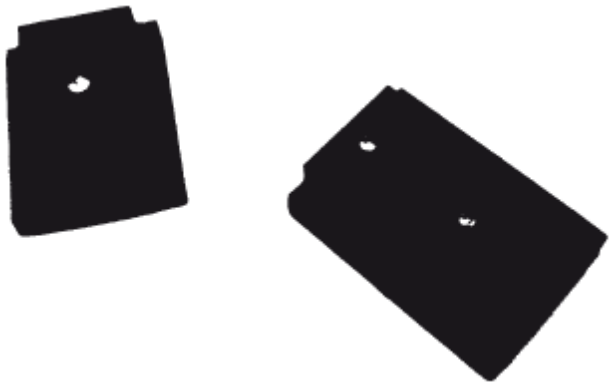
- Segmentation approaches
  - Region-based segmentation
    - Grouping set of pixels sharing common features into 2D connected areas
    - High memory usage
    - Low computational load
  - Boundary-based segmentation
    - Identifying the pixels corresponding to object contours and isolating them
    - Low memory usage
    - High computational load

    → Complementary approaches

Region-based segmentation

- Connected regions obtained by continuous merging of initially small groups of adjacent pixels into larger ones
    - Check of the “uniformity predicate” for adjacent regions
        - e.g. Check on the gray level

- Binarization (or binary segmentation)
    - Binary light intensity scale
        - Binary image. Only two possible values: 0 black, 1 white
        - Obtained by comparing the gray level of each pixel with a threshold
        - Separation of objects from the background → labeling
    - Selection of the threshold
        - Use of the gray-level histogram
        - Threshold corresponds to the minimum among two peaks (named *modes*), one of which corresponding to the background
        - Appropriate filtering of noisy images



Boundary-based segmentation:

- Boundary obtained by grouping many single “local” edges, corresponding to local discontinuities of image gray level
  - Derivation of an intermediate image based on local edges
  - Construction of short-curve segments by edge linking
  - Achievement of the boundaries by joining short-curve segments through geometric primitive known in advance
  - Effectiveness depends on reliability and authenticity of local edges

- Local edges detection
  - Filtering process (e.g. convolution)
    - Implemented via hardware
  - Computation of the spatial gradient of the image function
    - Information about rate of change of the gray level and about the direction of the maximum variation of the gray level
    - Different solutions for:
      - Direction for derivatives computation
      - Operators for the derivatives approximation
        - First differences, Roberts, Sobel, ...
    - Threshold to detect maximum (threshold sensitivity!)
  - Computation of the image function Laplacian
    - Second derivatives of the image function along two orthogonal directions (discrete operators)
    - Threshold to detect minimum
    - Sensitive to noise
    - Not provide directional information

Image interpretation

- Goal: computation of the image feature parameters
  - It works on segmented image (in terms of boundaries or in terms of regions)
- Feature parameters often used in visual servoing: *Moments*
  - Defined on a region of the image
  - Can be used to characterize easily position, orientation and shape of the 2D object correspondent to the region of definition
  - General definition of *moment* of i, j order

$$m_{i,j} = \sum_{X_I, Y_I \in \mathcal{R}} I(X_I, Y_I) X_I^i Y_I^j$$

  - In the case of binary image: $I(X_I, Y_I) = 1$ for $X_I, Y_I \in \mathcal{R}$
  - $m_{0,0}$ coincides with the area ("mass") of the region
  - *Centroid (position)* coordinates of the region

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}}, \qquad \bar{y} = \frac{m_{0,1}}{m_{0,0}}$$

    correspondent to the center of $\mathcal{R}$

  - General definition of the *central moments* of i, j order

$$\mu_{i,j} = \sum_{X_I, Y_I \in \mathcal{R}} (X_I - \bar{x})^i (Y_I - \bar{y})^j$$

    that are invariant with respect to translation

- According to a mechanical analogy, $\mu_{2,0}$ e $\mu_{0,2}$ represent the inertia moments with respect to X and Y axes, while $\mu_{1,1}$ is an inertia product, then the matrix

$$I = \begin{bmatrix} \mu_{2,0} & \mu_{1,1} \\ \mu_{1,1} & \mu_{0,2} \end{bmatrix}$$

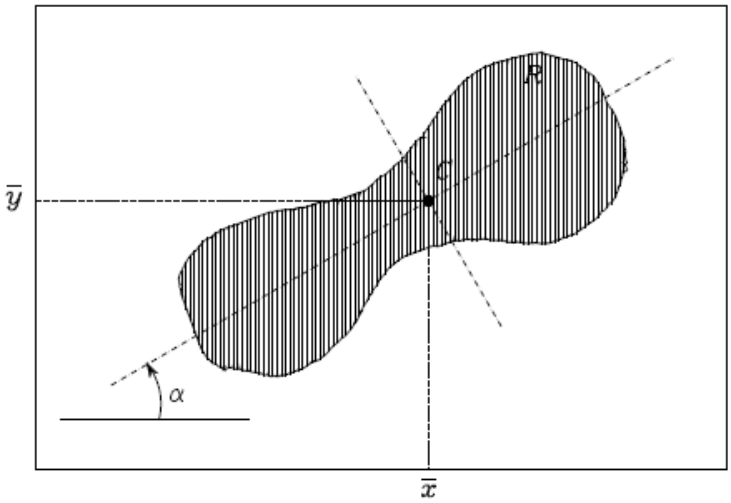
has the meaning of inertia tensor relative to the center of mass

- The eigenvalues of the matrix I define the *principal moments of inertia*, while the corresponding eigenvectors define the *principal axes of inertia* of the region
- If the region is asymmetric, the angle between the principal axis corresponding to the maximum moment and the axis X

$$\alpha = \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}\right)$$

can be used to characterize the *orientation* of the region

- The moments and the corresponding parameters can be computed also from the boundaries



- Feature points
  - Relevant points of the object (e.g. corners)
- Geometric primitives
  - Lines and lines segment, which are projections of linear edges or solids of revolution
  - Ellipses, obtained as projections of circles of spheres
  - These primitives can be characterized on the image plane in terms of a minimum set of parameters

Pose Estimation

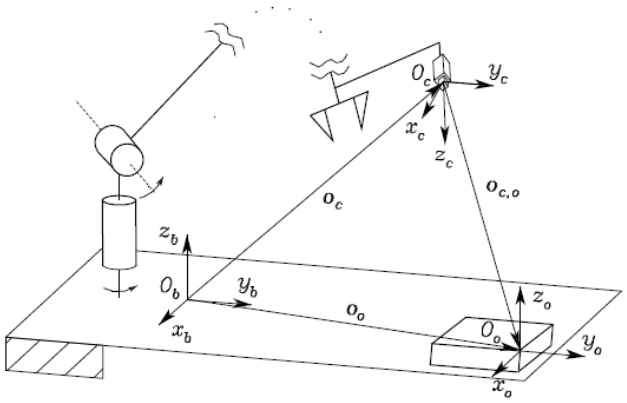
- For visual servoing purposes, a mapping between the feature parameters of an object measured in the image plane and the operational space variables is needed, defining the relative pose of the object with respect to the camera (or vice versa)
- Differential mapping in terms of velocity
  - Linear mapping!
  - Numerical integration algorithms can be used
- Feature parameters vector
  - Set of feature parameters of an image
  - Vector $\boldsymbol{s}$ (k x 1), termed *feature vector*
  - Normalized coordinates will be used
    - Intrinsic parameters of the camera are needed!
  - The feature vector of a point is defined as:

$$\boldsymbol{s} = \begin{bmatrix} X \\ Y \end{bmatrix} \qquad \tilde{\boldsymbol{s}} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

- Object pose with respect to the camera is defined as

$$\boldsymbol{T}_o^c = \begin{bmatrix} \boldsymbol{R}_o^c & \boldsymbol{o}_{c,o}^c \\ \boldsymbol{0}^T & 1 \end{bmatrix}$$

with $\boldsymbol{o}_{c,o}^c = \boldsymbol{o}_o^c - \boldsymbol{o}_c^c$



Analytic solution

- Problem: computation of elements of matrix $\boldsymbol{T}_o^c$ from the feature vector $\boldsymbol{s}$ containing the projections of n points of the object (e.g. eye-in-hand camera)

$$\boldsymbol{s}_i = \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \qquad \boldsymbol{s} = \begin{bmatrix} \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_n \end{bmatrix}$$

  Knowing the object model, the homogenous coordinates of the points of the object with respect to the camera frame can be expressed as

$$\boldsymbol{r}_{o,i}^o = \boldsymbol{p}_i^o - \boldsymbol{o}_o^o \qquad \tilde{\boldsymbol{r}}_{o,i}^c = \boldsymbol{T}_o^c \tilde{\boldsymbol{r}}_{o,i}^o$$

  and using the *perspective transformation* yields

$$\lambda_i \tilde{\boldsymbol{s}}_i = \boldsymbol{\Pi} \boldsymbol{T}_o^c \tilde{\boldsymbol{r}}_{o,i}^o$$

  with $\lambda_i > 0$ and

$$\boldsymbol{\Pi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- n correspondences → PnP Problem (Perspective-n-Points)
  - System of n equations for the unknown elements of $\boldsymbol{T}_o^c$
  - Multiple solutions may exist!
- P3P problem has four solutions, in the case of three non-collinear points
- P4P and P5P problems each have:
  - at least two solutions, in the case of non-coplanar points
  - unique solution, in the case of at least four coplanar points and no triplets of collinear points

- PnP problem, with $n \geq 6$ non-coplanar points has only one solution
- In the particular case of coplanar points, the derivation of the solution to the PnP problem becomes simpler
  - Assuming that the points lie in the plane of equation $z_o = 0$
  - Multiplying both sides of equations system by the skew-symmetric matrix $\boldsymbol{S}(\tilde{\boldsymbol{s}}_i)$, the following homogeneous equation is obtained

$$\boldsymbol{S}(\tilde{\boldsymbol{s}}_i)\boldsymbol{H}\begin{bmatrix} r_{x,i} & r_{y,i} & 1 \end{bmatrix}^{\mathrm{T}} = \boldsymbol{0}$$

where $r_{x,i}$ and $r_{y,i}$ are the two non-null components of vector $\boldsymbol{r}_{o,i}^{o}$, and where $\boldsymbol{H}$ is known as *planar homography* that is a (3x3) matrix

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{o}_{c,o}^{c} \end{bmatrix}$$

with $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ the first and second column of rotation matrix $\boldsymbol{R}_o^c$ respectively

  - Notice that the planar homography is linear with respect to $\boldsymbol{H}$, and then it can be rewritten in the form

$$\boldsymbol{A}_i(\boldsymbol{s}_i)\boldsymbol{h} = \boldsymbol{0}$$

where $\boldsymbol{h}$ is the (9x1) column vector obtained by stacking the columns of the matrix $\boldsymbol{H}$, while $\boldsymbol{A}_i$ is the (3x9) matrix

$$\boldsymbol{A}_i(\boldsymbol{s}_i) = \begin{bmatrix} r_{x,i}\boldsymbol{S}(\tilde{\boldsymbol{s}}_i) & r_{y,i}\boldsymbol{S}(\tilde{\boldsymbol{s}}_i) & \boldsymbol{S}(\tilde{\boldsymbol{s}}_i) \end{bmatrix}$$

  - Since the rank of $\boldsymbol{S}(\tilde{\boldsymbol{s}}_{\mathrm{i}})$ is at most 2, then the rank of matrix $\boldsymbol{A}_{\mathrm{i}}$ is also at most 2; therefore it is necessary to consider at least 4 equations written for 4 points of the plane, leading to the following system of 12 equations with 9 unknowns

$$\begin{bmatrix} A_1(s_1) \\ A_2(s_2) \\ A_3(s_3) \\ A_4(s_4) \end{bmatrix} h = A(s)h = 0$$

- It can be shown that, considering four points with no triplets of collinear points, matrix $\boldsymbol{A}$ has rank 8 and the system admits a non-null solution $\zeta\boldsymbol{h}$, defined up to a scaling factor ζ, from which it can be obtained

$$\begin{aligned} \boldsymbol{r}_1 &= \zeta \boldsymbol{h}_1 \\ \boldsymbol{r}_2 &= \zeta \boldsymbol{h}_2 \\ \boldsymbol{o}_{c,o}^{c} &= \zeta \boldsymbol{h}_3 \end{aligned}$$

  where $\boldsymbol{h}_i$ denotes the i-th column of matrix $\boldsymbol{H}$

- The value of the constant ζ can be computed by imposing the unit norm constraint to vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$

$$|\zeta| = \frac{1}{\|\boldsymbol{h}_1\|} = \frac{1}{\|\boldsymbol{h}_2\|}$$

  while the sign can be determined by choosing the solution corresponding to the object in front of the camera. Finally, the vector $\boldsymbol{r}_3$ can be computed as $\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$

- In a real situation, to reduce *noise* effects, a number $n > 4$ of correspondences can be considered and the solution can be computed using a *least-squares* technique
  - In general, the resulting matrix $\boldsymbol{Q} = [\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3]$ does not guarantee the property of being a rotation matrix
  - A possible solution consists of computing the rotation matrix "closest" to $\boldsymbol{Q}$ with respect to a given norm (e.g. Frobenius norm)

$$\|\boldsymbol{R}_o^c - \boldsymbol{Q}\|_F = \mathrm{Tr}\left((\boldsymbol{R}_o^c - \boldsymbol{Q})^{\mathrm{T}}(\boldsymbol{R}_o^c - \boldsymbol{Q})\right)$$

with the constraint that $\boldsymbol{R}_o^c$ is a rotation matrix

- The problem of minimizing the Frobenius norm is equivalent to that of maximizing the trace of matrix $\boldsymbol{R}_o^{c\,\mathrm{T}}\boldsymbol{Q}$, and the problem has the following solution

$$\boldsymbol{R}_o^c = \boldsymbol{U}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sigma \end{bmatrix}\boldsymbol{V}^{\mathrm{T}}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}^{\mathrm{T}}$ are, respectively, the left and the right orthogonal matrices of the singular value decomposition of $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ and $\sigma = \det(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}})$ ensures that the determinant of $\boldsymbol{R}_o^c$ is equal to 1

- The above derivation is a particular case of a method termed *direct linear transformation*
  - It allows the computation of the elements of $\boldsymbol{T}_o^c$ by solving a system of linear equations obtained using n correspondences relative to points in generic configuration
  - The system is obtained from the equalities

$$\boldsymbol{S}(\tilde{\boldsymbol{s}}_i)\begin{bmatrix}\boldsymbol{R}_o^c & \boldsymbol{o}_{c,o}^c\end{bmatrix}\tilde{\boldsymbol{r}}_{o,i}^o = \boldsymbol{0}$$

    from which two independent linear equations with 12 unknowns are obtained (n correspondences $\rightarrow 2n$ equations)
  - It can be shown that, by considering a set of 6 points not all coplanar, the corresponding system of 12 equations with 12 unknowns has rank 11 and it admits a solution defined up to a scaling factor

- In practical applications, due to the presence of noise, the system of equations has rank 12 and admits only the null solution
  - It is necessary to consider $n > 6$ of correspondences and a least-square technique

- In conclusion, it is necessary to know:
  - Object geometry (positions of the points with respect to the object frame)
  - Camera intrinsic parameters, to compute normalized coordinates from pixel ones
  - Camera extrinsic parameters, to compute the object pose with respect to the base frame, both with fixed and mobile cameras:
    - Camera eye-to-hand: $\boldsymbol{T}_o^b = \boldsymbol{T}_c^b \boldsymbol{T}_o^c$
    - Camera eye-in-hand: $\boldsymbol{T}_o^e = \boldsymbol{T}_c^e \boldsymbol{T}_o^c$

Interaction matrix

- If the object is in motion with respect to the camera, the feature vector $\boldsymbol{s}$ is time-varying; hence it is possible to define a (kx1) velocity vector in the image plane $\dot{\boldsymbol{s}}$
- The motion of the object with respect to the camera is characterized by the relative velocity

$$\boldsymbol{v}_{c,o}^{c} = \begin{bmatrix} \dot{\boldsymbol{o}}_{c,o}^{c} \\ \boldsymbol{R}_{c}^{\mathrm{T}}(\boldsymbol{\omega}_{o} - \boldsymbol{\omega}_{c}) \end{bmatrix}$$

  where $\dot{\boldsymbol{o}}_{c,o}^{c}$ is the time derivative of vector $\boldsymbol{o}_{c,o}^{c} = \boldsymbol{R}_{c}^{\mathrm{T}}(\boldsymbol{o}_{o} - \boldsymbol{o}_{c})$
- The equation relating these two velocities is

$$\dot{\boldsymbol{s}} = \boldsymbol{J}_{s}(\boldsymbol{s}, \boldsymbol{T}_{o}^{c})\boldsymbol{v}_{c,o}^{c}$$

  where $\boldsymbol{J}_{s}$ is a (kx6) matrix, termed *image Jacobian*
- It is useful to consider the mapping between the image plane velocity $\dot{\boldsymbol{s}}$, the absolute velocity of the camera frame

$$\boldsymbol{v}_{c}^{c} = \begin{bmatrix} \boldsymbol{R}_{c}^{\mathrm{T}}\dot{\boldsymbol{o}}_{c} \\ \boldsymbol{R}_{c}^{\mathrm{T}}\boldsymbol{\omega}_{c} \end{bmatrix}$$

  and the absolute velocity of the object frame

$$\boldsymbol{v}_{o}^{c} = \begin{bmatrix} \boldsymbol{R}_{c}^{\mathrm{T}}\dot{\boldsymbol{o}}_{o} \\ \boldsymbol{R}_{c}^{\mathrm{T}}\boldsymbol{\omega}_{o} \end{bmatrix}$$

  - The vector $\dot{\boldsymbol{o}}_{c,o}^{c}$ can be expressed as

$$\dot{\boldsymbol{o}}_{c,o}^{c} = \boldsymbol{R}_{c}^{\mathrm{T}}(\dot{\boldsymbol{o}}_{o} - \dot{\boldsymbol{o}}_{c}) + \boldsymbol{S}(\boldsymbol{o}_{c,o}^{c})\boldsymbol{R}_{c}^{\mathrm{T}}\boldsymbol{\omega}_{c}$$

    from which

$$\boldsymbol{v}_{c,o}^{c} = \boldsymbol{v}_{o}^{c} + \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^{c})\boldsymbol{v}_{c}^{c} \qquad \boldsymbol{\Gamma}(\cdot) = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{S}(\cdot) \\ \boldsymbol{O} & -\boldsymbol{I} \end{bmatrix}$$

    and the differential mapping can be rewritten as

$$\dot{\boldsymbol{s}} = \boldsymbol{J}_{s}\boldsymbol{v}_{o}^{c} + \boldsymbol{L}_{s}\boldsymbol{v}_{c}^{c}$$

    where the (kx6) matrix

$$\boldsymbol{L}_{s} = \boldsymbol{J}_{s}(\boldsymbol{s}, \boldsymbol{T}_{o}^{c})\boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^{c})$$

is termed *interaction matrix*

- The interaction matrix defines the linear mapping between the absolute velocity of the camera and the corresponding image plane velocity, in the case the object is fixed with respect to the base frame ($\boldsymbol{v}_o^c = \mathbf{0}$)

- The analytic expression of the interaction matrix is, in general, easier than that of image Jacobian, so that the latter can be computed from the interaction matrix using the following equation

$$\boldsymbol{J}_s(\boldsymbol{s}, \boldsymbol{T}_o^c) = \boldsymbol{L}_s \boldsymbol{\Gamma}(-\boldsymbol{o}_{c,o}^c)$$

obtained from the inversion of the previous equation defining the interaction matrix

Interaction matrix of a point

- Consider a point P of the object characterized by the vector

$$r_c^c = \boldsymbol{R}_c^{\mathrm{T}}(\boldsymbol{p} - \boldsymbol{o}_c)$$

- As the feature vector $\boldsymbol{s}$ of the point, choose the vector of normalized coordinates

$$\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{r}_c^c) = \frac{1}{z_c}\begin{bmatrix} x_c \\ y_c \end{bmatrix} \qquad \boldsymbol{r}_c^c = [x_c \quad y_c \quad z_c]^T$$

- Computing the time derivative yields

$$\dot{\boldsymbol{s}} = \frac{\partial \boldsymbol{s}(\boldsymbol{r}_c^c)}{\partial \boldsymbol{r}_c^c}\dot{\boldsymbol{r}}_c^c$$

with

$$\frac{\partial \boldsymbol{s}(\boldsymbol{r}_c^c)}{\partial \boldsymbol{r}_c^c} = \frac{1}{z_c}\begin{bmatrix} 1 & 0 & -x_c/z_c \\ 0 & 1 & -y_c/z_c \end{bmatrix} = \frac{1}{z_c}\begin{bmatrix} 1 & 0 & -X \\ 0 & 1 & -Y \end{bmatrix}$$

- Computing the time derivative of the vector $\boldsymbol{r}_c^c$, under the assumption that $\boldsymbol{p}$ is constant ($\boldsymbol{v}_o^c = \boldsymbol{0}$)

$$\dot{\boldsymbol{r}}_c^c = -\boldsymbol{R}_c^{\mathrm{T}}\dot{\boldsymbol{o}}_c + \boldsymbol{S}(\boldsymbol{r}_c^c)\boldsymbol{R}_c^{\mathrm{T}}\boldsymbol{\omega}_c = [\,-\boldsymbol{I} \quad \boldsymbol{S}(\boldsymbol{r}_c^c)\,]\,\boldsymbol{v}_c^c$$

- Combining the previous two equations, the following expression of interaction matrix of a point can be found

$$\boldsymbol{L}_s(\boldsymbol{s}, z_c) = \begin{bmatrix} -\frac{1}{z_c} & 0 & \frac{X}{z_c} & XY & -(1+X^2) & Y \\ 0 & -\frac{1}{z_c} & \frac{Y}{z_c} & 1+Y^2 & -XY & -X \end{bmatrix}$$

- The image Jacobian can be computed from the interaction matrix

$$\boldsymbol{J}_s(\boldsymbol{s}, \boldsymbol{T}_o^c) = \frac{1}{z_c}\begin{bmatrix} 1 & 0 & -X & -r_{o,y}^c X & r_{o,z}^c + r_{o,x}^c X & -r_{o,y}^c \\ 0 & 1 & -Y & -(r_{o,z}^c + r_{o,y}^c Y) & r_{o,x}^c Y & r_{o,x}^c \end{bmatrix}$$

where $r_{o,x}^c$, $r_{o,y}^c$ e $r_{o,z}^c$ are the components of the vector

$$\boldsymbol{r}_o^c = \boldsymbol{r}_c^c - \boldsymbol{o}_{c,o}^c = \boldsymbol{R}_o^c \boldsymbol{r}_o^o$$

Interaction matrix of a set of points

- The interaction matrix of a set of n points of the object $P_1, P_2, \dots, P_n$ can be built by considering the ($2n$x1) feature vector
- By denoting with $\boldsymbol{L}_{s_i}(\boldsymbol{s}_i, z_{c,i})$ the interaction matrix corresponding to point P_i, the interaction matrix of the set of points will be the ($2n$x6) matrix

$$\boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c) = \begin{bmatrix} \boldsymbol{L}_{s_1}(\boldsymbol{s}_1, z_{c,1}) \\ \vdots \\ \boldsymbol{L}_{s_n}(\boldsymbol{s}_n, z_{c,n}) \end{bmatrix}$$

with $\boldsymbol{z}_c = [z_{c,1}, z_{c,2,\dots}, z_{c,n}]^{\mathrm{T}}$

Interaction matrix of a line segment

- A line segment is the part of the line connecting two points P_1,P_2
- The projection on the image plane is still a line segment that can be characterized by the middle point coordinates $\bar{x}$, $\bar{y}$, the length L and the angle α formed by the line with respect to axis X:

$$\boldsymbol{s} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ L \\ \alpha \end{bmatrix} = \begin{bmatrix} (X_1 + X_2)/2 \\ (Y_1 + Y_2)/2 \\ \sqrt{\Delta X^2 + \Delta Y^2} \\ \tan^{-1}(\Delta Y / \Delta X) \end{bmatrix} = \boldsymbol{s}(\boldsymbol{s}_1, \boldsymbol{s}_2)$$

  where $\Delta X = X_2 - X_1$, $\Delta Y = Y_2 - Y_1$ and $\boldsymbol{s}_i = [X_i \quad Y_i]^{\mathrm{T}}$, with $i = 1, 2$
- Computing the time derivative of the feature vector yields

$$\begin{aligned} \dot{\boldsymbol{s}} &= \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_1}\dot{\boldsymbol{s}}_1 + \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_2}\dot{\boldsymbol{s}}_2 \\ &= \left(\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_1}\boldsymbol{L}_{s_1}(\boldsymbol{s}_1, z_{c,1}) + \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_2}\boldsymbol{L}_{s_2}(\boldsymbol{s}_2, z_{c,2}) \right) \boldsymbol{v}_c^c \end{aligned}$$

  where $\boldsymbol{L}_{s_i}$ is the interaction matrix of point P_i (under the assumption that the line segment is fixed with respect to the base frame)
- Then, the interaction matrix of a line segment is

$$\boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c) = \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_1}\boldsymbol{L}_{s_1}(\boldsymbol{s}_1, z_{c,1}) + \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_2}\boldsymbol{L}_{s_2}(\boldsymbol{s}_2, z_{c,2})$$

  with

$$\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_1} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \\ -\Delta X/L & -\Delta Y/L \\ \Delta Y/L^2 & -\Delta X/L^2 \end{bmatrix}, \quad \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_2} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \\ \Delta X/L & \Delta Y/L \\ -\Delta Y/L^2 & \Delta X/L^2 \end{bmatrix}$$

- The vectors $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ can be computed as functions of the parameters $\bar{x}$, $\bar{y}$, L and α; therefore the interaction matrix can be expressed as a function of the feature vector $\boldsymbol{s}$, besides the components $z_{c,1}$ and $z_{c,2}$ of the endpoints P_1 and P_2 of the line segment

Algorithmic solution

- In general, the interaction matrix $\boldsymbol{L}_s$ is a matrix of dimension (kxm)
  - k is the number of feature parameters of the image
  - m is dimension of the velocity vector $\boldsymbol{v}_c^c$
    - $m < 6$ when the relative motion of the object with respect to the camera is constrained
    - $m = 6$ in the other cases
- The image Jacobian $\boldsymbol{J}_s$ is also of dimension (kxm) and its rank coincides with that of the interaction matrix
- In the case that $\boldsymbol{L}_s$ is full rank, it is possible to compute $\boldsymbol{v}_{c,o}^c$ from $\dot{\boldsymbol{s}}$
- If $k = m$

$$\boldsymbol{v}_{c,o}^c = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^c)\boldsymbol{L}_s^{-1}\dot{\boldsymbol{s}}$$

- If $k > m$, the interaction matrix has more rows than columns → least-squares solution

$$\boldsymbol{v}_{c,o}^c = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^c)(\boldsymbol{L}_s^{\mathrm{T}}\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^{\mathrm{T}}\dot{\boldsymbol{s}}$$

  where the left pseudo-inverse of $\boldsymbol{L}_s$ has been used
- If $k < m$, the interaction matrix has more columns than rows → infinite solutions
  - Number of parameters is not sufficient to determine uniquely the relative motion of the object with respect to the camera
  - Relative motions of the object with respect to the camera (or vice versa) do not produce variations of the image feature parameters→ velocities belong to the null subspace of $\boldsymbol{J}_s$
  - This case has no interest if the problem is that of computing uniquely the relative pose of the object with respect to the camera from feature parameters in the image plane

- The pose estimation problem may be cast in a form analogous to that of inverse kinematics algorithms for robot manipulators
- To represent the relative pose of the object with respect to the camera using a minimum number of coordinates, consider the (mx1) vector

$$\boldsymbol{x}_{c,o} = \begin{bmatrix} \boldsymbol{o}_{c,o}^{c} \\ \boldsymbol{\phi}_{c,o} \end{bmatrix}$$

- If Euler angles are used to represent orientation, the transformation matrix between $\boldsymbol{v}_{c,o}^{c}$ and $\dot{\boldsymbol{x}}_{c,o}$ is

$$\boldsymbol{v}_{c,o}^{c} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{T}(\boldsymbol{\phi}_{c,o}) \end{bmatrix} \dot{\boldsymbol{x}}_{c,o} = \boldsymbol{T}_{A}(\boldsymbol{\phi}_{c,o})\dot{\boldsymbol{x}}_{c,o}$$

- From which the mapping between $\dot{\boldsymbol{s}}$ and $\dot{\boldsymbol{x}}_{c,o}$ is given by

$$\dot{\boldsymbol{s}} = \boldsymbol{J}_{A_s}(\boldsymbol{s}, \boldsymbol{x}_{c,o})\dot{\boldsymbol{x}}_{c,o}$$

  where the matrix

$$\boldsymbol{J}_{A_s}(\boldsymbol{s}, \boldsymbol{x}_{c,o}) = \boldsymbol{L}_{s}\boldsymbol{\Gamma}(-\boldsymbol{o}_{c,o}^{c})\boldsymbol{T}_{A}(\boldsymbol{\phi}_{c,o})$$

  has a meaning analogous to that of the analytic Jacobian of a manipulator
- By denoting with $\hat{\boldsymbol{s}} = \boldsymbol{s}(\hat{\boldsymbol{x}}_{c,o})$ the current estimate of feature vector $\boldsymbol{s}$, computed from the pose specified by $\hat{\boldsymbol{x}}_{c,o}$ (current estimate of vector $\boldsymbol{x}_{c,o}$), it can be defined an algorithm to minimize the error

$$\boldsymbol{e}_{s} = \boldsymbol{s} - \hat{\boldsymbol{s}}$$

- Computing the time derivative of the previous equation yields

$$\dot{\boldsymbol{e}}_{s} = -\dot{\hat{\boldsymbol{s}}} = -\boldsymbol{J}_{A_s}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{x}}_{c,o})\dot{\hat{\boldsymbol{x}}}_{c,o}$$

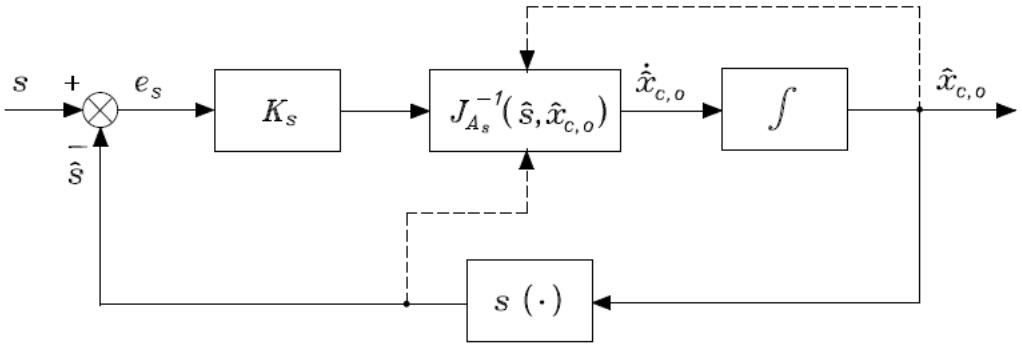
- Assumed that matrix $\boldsymbol{J}_{A_s}$ is square and nonsingular, the choice

$$\dot{\hat{\boldsymbol{x}}}_{c,o} = \boldsymbol{J}_{A_s}^{-1}(\hat{\boldsymbol{s}}, \hat{\boldsymbol{x}}_{c,o})\boldsymbol{K}_{s}\boldsymbol{e}_{s}$$

  leads to the equivalent linear system

$$\dot{\boldsymbol{e}}_{s} + \boldsymbol{K}_{s}\boldsymbol{e}_{s} = \boldsymbol{0}$$

If $\boldsymbol{K}_s$ is a positive definite matrix (usually diagonal), the previous system is asymptotically stable and the error tends to zero with a convergence speed that depends on the eigenvalues of the matrix $\boldsymbol{K}_s$



- The function $\boldsymbol{s}(\cdot)$ denotes the function computing the feature vector of the "virtual" image corresponding to the current estimate $\hat{\boldsymbol{x}}_{c,o}$ of the object pose with respect to the camera
- The convergence properties depend on the choice of the image feature parameters, on the initial condition $\hat{\boldsymbol{x}}_{c,o}(0)$ and on instability problems related to singularities of matrix $\boldsymbol{J}_{A_s}$
- Singularities of the matrix $\boldsymbol{J}_{A_s}$
  - Representation singularities of the orientation
  - Singularities of the interaction matrix (!)
- To separate the effects of the two types of singularities, it is convenient to realize the computation in two steps

$$\hat{\boldsymbol{v}}_{c,o}^{c} = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^{c})\boldsymbol{L}_s^{-1}\boldsymbol{K}_s\boldsymbol{e}_s \qquad\qquad \dot{\hat{\boldsymbol{x}}}_{c,o} = \boldsymbol{T}_A^{-1}(\boldsymbol{\phi}_{c,o})\hat{\boldsymbol{v}}_{c,o}^{c}$$

- Under the assumption of working far from representation singularities, the problem of singularities of the interaction matrix can be overcome by using a number k of feature parameters greater than the minimum required m ($k > m$)
  - Reduction of the effects of measurement noises
  - The resulting estimation algorithm requires the use of the left pseudo-inverse of interaction matrix

$$\hat{\boldsymbol{v}}_{c,o}^{c} = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^{c})(\boldsymbol{L}_s^{\mathrm{T}}\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^{\mathrm{T}}\boldsymbol{K}_s\boldsymbol{e}_s$$

- The convergence can be shown using the direct Lyapunov method based on the following positive definite function

$$V(\boldsymbol{e}_s) = \frac{1}{2}\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{K}_s\boldsymbol{e}_s$$

Computing the time derivative follows

$$\dot{V} = -\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{K}_s\boldsymbol{L}_s(\boldsymbol{L}_s^{\mathrm{T}}\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^{\mathrm{T}}\boldsymbol{K}_s\boldsymbol{e}_s$$

which is negative semi-definite, because $\mathcal{N}\left(\boldsymbol{L}_s{}^{\mathrm{T}}\right) \neq \emptyset$ being $\boldsymbol{L}_s{}^{\mathrm{T}}$ a matrix with more columns than row → the system is stable, but not asymptotically stable**!**

- The algorithm can get stuck with $\boldsymbol{e}_s \neq 0$ and $\boldsymbol{K}_s\boldsymbol{e}_s \in \mathcal{N}\left(\boldsymbol{L}_s{}^{\mathrm{T}}\right)$

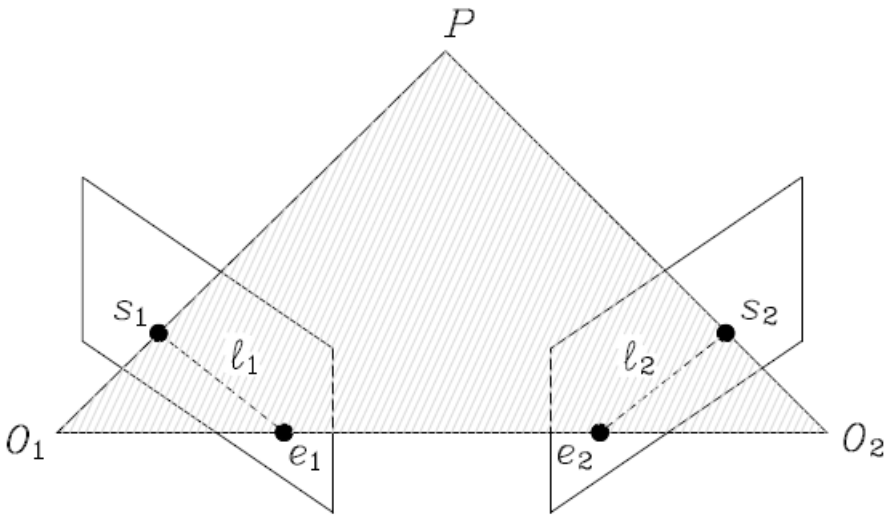
- These methods for their performance are mainly adopted for real-time "visual-tracking" applications
  - The estimate has as initial value the estimate computed at the previous step time

Stereo Vision

- 2D Image → Does not give any explicit information on depth
- This information can be recovered:
  - Indirect way from the geometric model of the object, assumed to be known
  - Direct way from two or more images from different points of view (Stereo Vision)
    - 2 or more cameras
    - One moving camera (fixed object!)
- Fundamental problems in the framework of stereo vision
  - Correspondence problem
    - *Definition*: identification of the points of the two images that are projections of the same point of the scene (conjugate or corresponding points)
    - Existence of geometric constraints between two images of the same point
    - Existence of details of the scene which appear to be similar in the different images
  - 3D reconstruction
    - *Definition*: computation of the relative pose of the cameras (calibrated and not) and thus, starting from this pose, the position in the 3D space of the points of the observed object

Epipolar geometry

- It has the basis on the *epipolar constraint*
  - Consider two cameras, noted as 1 and 2



  - The coordinates of a point P expressed in the two frames are related by the following equation

$$p^1 = o_{1,2}^1 + R_2^1 p^2$$

  - Let s_1 and s_2 be the coordinates of the projections of P on the image planes

$$\lambda_i \tilde{s}_i = \Pi \tilde{p}^i = p^i, \qquad i = 1, 2$$

  from which, substituting the previous equation in the first one

$$\lambda_1 \tilde{s}_1 = o_{1,2}^1 + \lambda_2 R_2^1 \tilde{s}_2$$

  Premultiplying both sides by $S(o_{1,2}^1)$ gives

$$\lambda_1 S(o_{1,2}^1)\tilde{s}_1 = \lambda_2 S(o_{1,2}^1) R_2^1 \tilde{s}_2$$

  Premultiplying both sides by $\tilde{s}_1^{\mathrm{T}}$

$$\lambda_2 \tilde{s}_1^{\mathrm{T}} S(o_{1,2}^1) R_2^1 \tilde{s}_2 = 0$$

which has to be satisfied for any value of the scalar λ_2; therefore this equality is equivalent to the so-called *epipolar constraint*

$$\tilde{\boldsymbol{s}}_1^{\mathrm{T}} \boldsymbol{E} \tilde{\boldsymbol{s}}_2 = 0$$

with $\boldsymbol{E} = \boldsymbol{S}\left(\boldsymbol{o}_{1,2}^1\right)\boldsymbol{R}_2^1$ known as a (3x3) matrix termed *essential matrix*

- The epipolar constraint expresses in analytic form the geometric constraint existing between the projections of the same point on the image planes of two different cameras
  - The segments O_1P, O_2P and O_1O_2 (base line) must lie on the same plane (epipolar plane)
- The projections of the optical centers O_1, O_2 of each camera on the image plane of the other camera $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ are termed *epipoles*
  - The epipoles and the base line do not change if the point P varies
- The segments ℓ_1 and ℓ_2 are termed *epipolar lines* (intersection of the epipolar plane with the image planes of the two cameras)
- For the purpose of *computing the correspondences*
  - Finding correspondences reduces to searching for a point along the epipolar line and not on the whole image plane

Triangulation

- Purely geometric method of 3D reconstruction which can be used in the case that both intrinsic and extrinsic parameters of the camera are known (calibrated cameras)
- This method allows the computation of the coordinates $\boldsymbol{p} = [p_x \quad p_y \quad p_z]^{\mathrm{T}}$ of a point P with respect to the base frame, starting from the normalized coordinates $\boldsymbol{s}_i = [X_i \quad Y_i]^{\mathrm{T}}$ of the projections of P on the image planes
  - If the base frame coincides with Frame 1, it can be easily shown
$$\boldsymbol{p} = \lambda_1 \tilde{\boldsymbol{s}}_1$$
$$\boldsymbol{p} = \boldsymbol{o} + \lambda_2 \boldsymbol{R} \tilde{\boldsymbol{s}}_2$$
    which can be resolved with respect to $\boldsymbol{p}$
  - Computing λ_1 from the third scalar equation of the first vectorial equation and replacing its value into the other two yields
$$\begin{bmatrix} 1 & 0 & -X_1 \\ 0 & 1 & -Y_1 \end{bmatrix} \boldsymbol{p} = \boldsymbol{0}$$
  - Using a similar derivation for the second vectorial equation obtained by premultiplying both sides by $\boldsymbol{R}^{\mathrm{T}}$, the following system is obtained
$$\begin{bmatrix} \boldsymbol{r}_1^{\mathrm{T}} - X_2 \boldsymbol{r}_3^{\mathrm{T}} \\ \boldsymbol{r}_2^{\mathrm{T}} - Y_2 \boldsymbol{r}_3^{\mathrm{T}} \end{bmatrix} \boldsymbol{p} = \begin{bmatrix} o_x - o_z X_2 \\ o_y - o_z Y_2 \end{bmatrix}$$
    with $\boldsymbol{R} = [\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3]$ and $\boldsymbol{R}^{\mathrm{T}} \boldsymbol{o} = [o_x \quad o_y \quad o_z]^{\mathrm{T}}$
  - A system of 4 linear equations with respect to $\boldsymbol{p}$ with 3 unknowns has been obtained, of which only 3 equations are linear independent
  - In practical applications, because of noise, these 4 equations are independent and the system has only one approximated solution: suitable algorithms based on least-squares techniques have to be adopted to compute it

- In the case of cameras with parallel and aligned image planes, the solution can be greatly simplified
  - $\boldsymbol{R} = \boldsymbol{I}$ and $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{o} = [b \quad 0 \quad 0]^{\mathrm{T}}$, with $b > 0$ being the distance between the origins of the two cameras frame
  - The system solution is

$$p_x = \frac{X_1 b}{X_1 - X_2}$$
$$p_y = \frac{Y_1 b}{X_1 - X_2} = \frac{Y_2 b}{X_1 - X_2}$$
$$p_z = \frac{b}{X_1 - X_2}.$$

Absolute orientation

- *Definition*: Computation of the relative motion between a system of two calibrated cameras and a known rigid object
  - Under the assumption of rigid motion, let us consider the base frame coincident with Frame 1
  - If the stereo camera system is moving and the object is fixed, let $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n$ be the position vectors of n points measured at time t, and let $\boldsymbol{p}'_1, \dots, \boldsymbol{p}'_n$ be the position vectors of the same points measured at time t', using triangulation
  - With the assumption of rigid motion

$$p_i = o + Rp'_i \qquad i = 1, \dots, n$$

    where $\boldsymbol{o}$ and $\boldsymbol{R}$ are the unknowns
- From rigid body mechanics it is known that this problem has a unique solution in the case of three non-collinear points
  - Since points are obtained using triangulation, the measurements are affected by error and the system may have no solution
  - It is convenient to consider a number $n > 3$, and the computed solution is that which minimizes the function

$$\sum_{i=1}^{n} \|p_i - o - Rp'_i\|^2$$

    with the constraint that $\boldsymbol{R}$ is a rotation matrix

- The problem can be separated in two parts
  - Observe that the value $\boldsymbol{o} = \bar{\boldsymbol{p}} - \boldsymbol{R}\bar{\boldsymbol{p}}'$ minimizes the function if $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}'$ are chosen as the centroids of the set of points $\{\boldsymbol{p}_i\}$ and $\{\boldsymbol{p}_i'\}$, and then defined as

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i, \qquad \bar{p}' = \frac{1}{n}\sum_{i=1}^{n} p_i'$$

  - The problem becomes that of computing $\boldsymbol{R}$ which minimizes the linear quadratic function

$$\sum_{i=1}^{n} \|\bar{p}_i - R\bar{p}_i'\|^2 \qquad \begin{aligned} \bar{p}_i &= p_i - \bar{p} \\ \bar{p}_i' &= p_i' - \bar{p}' \end{aligned}$$

  - The solution is the matrix $\boldsymbol{R}$ which minimizes the trace of the matrix $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{K}$, with

$$K = \sum_{i=1}^{n} \bar{p}_i \bar{p}_i'^{\mathrm{T}}$$

    already presented in the analytic solution for the pose estimation

3D reconstruction from planar homography

- *Goal*: evaluate the motion of a cameras system which observes an unknown planar object
  - Rotation and displacement defined up to a scaling factor
- *Assumption*: feature points of the observed object lie on the same plane
  - Additional constraint to the epipolar constraint → planar homography
- By denoting with $d_2 > 0$ the distance of the plane from the origin of Frame 2, with $\boldsymbol{n}^2$ the unit vector orthogonal to the plane containing the feature points and with $\boldsymbol{p}^2$ the position vector of a point P expressed with respect to Frame 2, it is

$$\frac{1}{d_2}\boldsymbol{n}^{2\mathrm{T}}\boldsymbol{p}^2 = 1$$

From this relation and from the relation of the representations between the two cameras, it follows

$$\boldsymbol{p}^1 = \boldsymbol{H}\boldsymbol{p}^2 \qquad \boldsymbol{H} = \boldsymbol{R}_2^1 + \frac{1}{d_2}\boldsymbol{o}_{1,2}^1\boldsymbol{n}^{2\mathrm{T}}$$

from which

$$\tilde{\boldsymbol{s}}_1 = \lambda\boldsymbol{H}\tilde{\boldsymbol{s}}_2 \qquad \lambda = \lambda_2/\lambda_1 > 0$$

By premultiplying both sides by $\boldsymbol{S}(\tilde{\boldsymbol{s}}_1)$, the following equality yields

$$\boldsymbol{S}(\tilde{\boldsymbol{s}}_1)\boldsymbol{H}\tilde{\boldsymbol{s}}_2 = \boldsymbol{0}$$

representing a planar homography defined by matrix $\boldsymbol{H}$

- The system admits solution up to a scaling factor, if coordinates of 4 points in the plane are known
  - The constant value can be computed using a numerical algorithm based on the expression of the matrix $\boldsymbol{H}$
  - Once $\boldsymbol{H}$ is known, it is possible to compute $\boldsymbol{n}^2$, $\boldsymbol{R}_2^1$ and $\boldsymbol{o}_{1,2}^1/d_2$
    - The solution is defined up to a scalar factor d_2
    - Two admissible solutions exist!

Camera Calibration

- Camera calibration
  - Estimation of the intrinsic parameters (matrix $\boldsymbol{\Omega}$)
  - Estimation of the extrinsic parameters (pose of the camera frame with respect to the base frame or the end-effector frame)
- Various calibration techniques exist which are based on algorithms similar to those used for the pose estimation of an object with respect to the camera
  - If the intrinsic parameters are known, the solution to the PnP problem with n coplanar points can be directly used to compute the extrinsic parameters
    - With a fixed camera: $\boldsymbol{T}_c^b = \boldsymbol{T}_o^b (\boldsymbol{T}_o^c)^{-1}$
    - With a non-fixed camera: $\boldsymbol{T}_c^e = \boldsymbol{T}_o^e (\boldsymbol{T}_o^c)^{-1}$
- In a general case where the intrinsic parameters are not known, a suitable method in three phases can be adopted
  - **Phase 1: planar homography computation**
    - A planar homography can be computed starting from the pixel coordinates

$$\boldsymbol{c}_i = \begin{bmatrix} X_{Ii} \\ Y_{Ii} \end{bmatrix}$$

from which the following equations system is obtained

$$\boldsymbol{S}(\boldsymbol{c}_i)\boldsymbol{H}' \begin{bmatrix} r_{x,i} & r_{y,i} & 1 \end{bmatrix}^{\mathrm{T}} = \boldsymbol{0}$$

$$\boldsymbol{H}' = \boldsymbol{\Omega}\boldsymbol{H}$$

where $\boldsymbol{\Omega}$ is the intrinsic parameters matrix and $\boldsymbol{H}$ is the matrix

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{o}_{c,o}^c \end{bmatrix}$$

    - With the method exposed in the case of analytic solution, $\zeta \boldsymbol{H}'$ can be computed in the case of $n \geq 4$

- **Phase 2: intrinsic parameters $\boldsymbol{\Omega}$ computation**
  - Matrix $\boldsymbol{\Omega}$ can be computed from the elements of matrix $\zeta \boldsymbol{H}'$

$$\zeta \begin{bmatrix} \boldsymbol{h}'_1 & \boldsymbol{h}'_2 & \boldsymbol{h}'_3 \end{bmatrix} = \boldsymbol{\Omega} \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{o}^c_{c,o} \end{bmatrix}$$

    where $\boldsymbol{h}'_i$ denotes the i-th column of $\boldsymbol{H}'$
  - Computing $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ and imposing the orthogonality and unit norm constraints on these vectors yields

$$\boldsymbol{h}'^{\mathrm{T}}_1 \boldsymbol{\Omega}^{-\mathrm{T}} \boldsymbol{\Omega}^{-1} \boldsymbol{h}'_2 = 0$$
$$\boldsymbol{h}'^{\mathrm{T}}_1 \boldsymbol{\Omega}^{-\mathrm{T}} \boldsymbol{\Omega}^{-1} \boldsymbol{h}'_1 = \boldsymbol{h}'^{\mathrm{T}}_2 \boldsymbol{\Omega}^{-\mathrm{T}} \boldsymbol{\Omega}^{-1} \boldsymbol{h}'_2$$

    which, being linear, can be rewritten in the form

$$\boldsymbol{A}' \boldsymbol{b} = \boldsymbol{0}$$

    where $\boldsymbol{A}'$ is a (2x6) matrix depending on $\boldsymbol{h}'_1$ and $\boldsymbol{h}'_2$, while

$$\boldsymbol{b} = \begin{bmatrix} b_{11} & b_{12} & b_{22} & b_{13} & b_{23} & b_{33} \end{bmatrix}^{\mathrm{T}}$$

    with $b_{i,j}$ the generic element of the symmetric matrix

$$B = \boldsymbol{\Omega}^{-\mathrm{T}} \boldsymbol{\Omega}^{-1} = \begin{bmatrix} 1/\alpha_x^2 & 0 & -X_0/\alpha_x^2 \\ 0 & 1/\alpha_y^2 & -Y_0/\alpha_y^2 \\ -X_0/\alpha_x^2 & -Y_0/\alpha_y^2 & 1 + X_0^2/\alpha_x^2 + Y_0^2/\alpha_y^2 \end{bmatrix}$$

  - By repeating the Phase 1 k times, with the same plane placed in different pose, the system $\boldsymbol{A}' \boldsymbol{b} = \boldsymbol{0}$ is obtained, where $\boldsymbol{A}'$ is a $2k$x6 matrix
  - In the case $k \geq 3$ a unique solution $\gamma \boldsymbol{b}$ exists up to a scaling factor γ
  - From matrix $\gamma \boldsymbol{B}$ the following expression for the intrinsic parameters can be found

$$X_0 = -b'_{13}/b'_{11}$$
$$Y_0 = -b'_{23}/b'_{22}$$
$$\alpha_x = \sqrt{\gamma/b'_{11}}$$
$$\alpha_y = \sqrt{\gamma/b'_{22}}$$

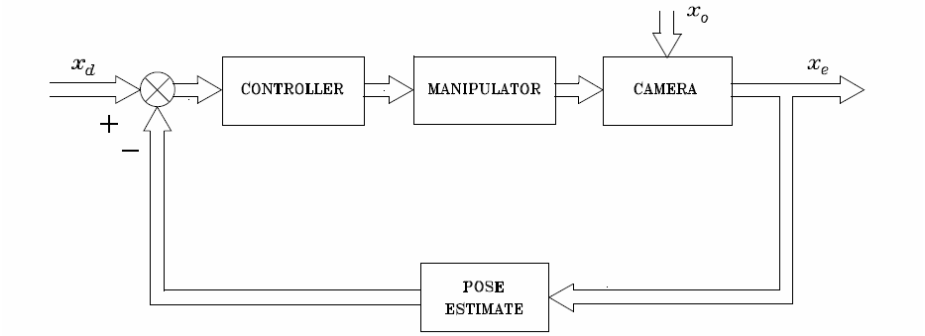
    with $b'_{i,j} = \gamma b_{i,j}$ and $\gamma = b_{13} + b_{23} + b_{33}$

- **Phase 3: extrinsic parameters computation**
  - Once matrix $\boldsymbol{\Omega}$ of intrinsic parameters has been computed, it is possible to evaluate the matrix $\boldsymbol{H}$ (up to a constant ζ) from matrix $\boldsymbol{H}'$ for one of the k poses of the plane
  - Having evaluated $\boldsymbol{H}$, the computation of matrix $\boldsymbol{T}_o^c$ can be performed as for the case of the solution of a $\mathrm{P}n\mathrm{P}$ problem
  - It is then possible to compute
    - In eye-to-hand case: $\boldsymbol{T}_c^b = \boldsymbol{T}_o^b(\boldsymbol{T}_o^c)^{-1}$
    - In eye-in-hand case: $\boldsymbol{T}_c^e = \boldsymbol{T}_o^e(\boldsymbol{T}_o^c)^{-1}$

- The method illustrated above is merely conceptual
  - Does not provide satisfactory solutions in the presence of measurement noise
  - Lens distortion is not considered
  - The accuracy of the results can be improved using models that take in account the distortion phenomena of the lenses, together with nonlinear optimization techniques
    - A technique similar to that proposed with the assumption of absence of lens distortion and of a linear model can be used
    - Then, nonlinear techniques for optimization can be used for the estimation of the parameters of the complete nonlinear model
- Calibration plane with a “chessboard” pattern
- A calibration method can also be set up starting from the solution of a $\mathrm{P}n\mathrm{P}$ nonplanar problem using the direct linear transformation method

The Visual Servoing Problem

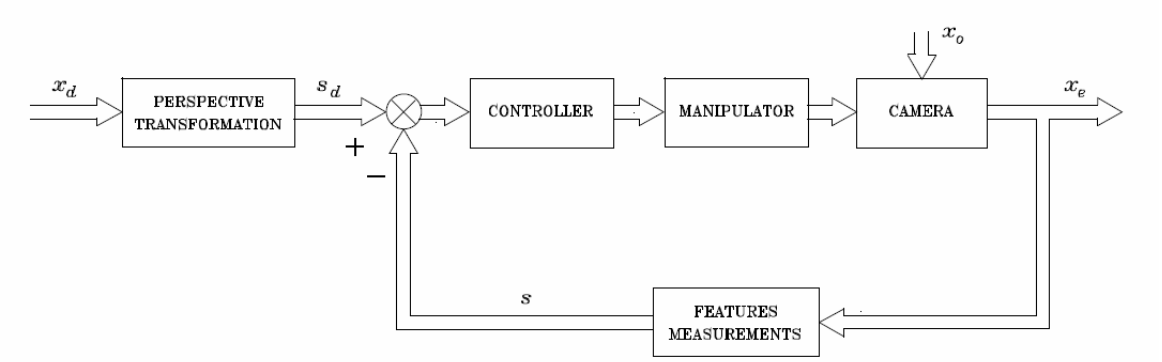
- *Goal*: The objective of visual servoing is to ensure that the end-effector, on the basis of the visual measures elaborated in real-time, reaches and keeps a (constant or time-varying) desired pose with respect to the observed object
  - Only regulation problem will be considered
- The visual system provides feature parameters in the image plane as direct measures, while the robotic task is defined in the operational space, in terms of the relative pose of the end-effector with respect to the object
- Two kinds of control approaches can be defined
  - Visual Servoing in the Operational Space (Position Based Visual Control)
  - Visual Servoing in the Image Space (Image Based Visual Control)
- Visual servoing in the operational space



  - Real-time estimation of the relative pose between the object and the camera
    - Estimation can be performed analytically or using iterative numerical algorithms
  - Direct control of the operational space variables
  - The absence of a direct control of the image features can lead the object to exit from the camera field of view → open loop!
  - Very sensitive with respect to calibration errors

- If one camera is used, the object model is needed

- Visual servoing in the image space



- The control action is computed on the basis of the error defined as the difference between the values of image features in the desired configuration and in the current one
- The object pose estimation with respect to the camera is not required
- It is possible to keep the object within the camera field of view, but singular configurations may occur
- The end-effector trajectories are not easily predictable → collisions, joints limits violations
- Robust with respect to calibration errors
- The observed object model is not needed

Position-based Visual Servoing

- Only regulation problem will be considered (constant reference)
- Assumptions:
  - Calibrated camera
  - End-effector frame ≡ Camera frame
  - Fixed object with respect to the base frame
- Features measurements for on-line estimation of matrix $\boldsymbol{T}_o^c \rightarrow \boldsymbol{x}_{c,o}$
- The problem can be formulated by imposing a desired value to the relative pose of the object frame with respect to the camera $\boldsymbol{T}_o^d \rightarrow \boldsymbol{x}_{d,o}$
- The displacement matrix of the camera frame in the current pose with respect to the desired pose can be evaluated

$$\boldsymbol{T}_c^d = \boldsymbol{T}_o^d (\boldsymbol{T}_o^c)^{-1} = \begin{bmatrix} \boldsymbol{R}_c^d & \boldsymbol{o}_{d,c}^d \\ \boldsymbol{0}^T & 1 \end{bmatrix}$$

  and a suitable error vector in operational space can be computed

$$\tilde{\boldsymbol{x}} = - \begin{bmatrix} \boldsymbol{o}_{d,c}^d \\ \boldsymbol{\phi}_{d,c} \end{bmatrix}$$

  which does not depend on the object pose and does not coincide with the difference between $\boldsymbol{x}_{d,o} - \boldsymbol{x}_{c,o}$
- The control has to be designed so that the operational space error $\tilde{\boldsymbol{x}}$ tends to zero asymptotically
- The control objective can be satisfied provided that $\boldsymbol{x}_{d,o}$, corresponding to the homogeneous transformation matrix

$$\boldsymbol{T}_d = \boldsymbol{T}_c (\boldsymbol{T}_c^d)^{-1} = \begin{bmatrix} \boldsymbol{R}_d & \boldsymbol{o}_d \\ \boldsymbol{0}^T & 1 \end{bmatrix}$$

  belongs to the dexterous workspace of the manipulator

PD control with gravity compensation

- Computation of the time derivative of $\tilde{\boldsymbol{x}}$ yields

$$\dot{o}_{d,c}^{d} = \dot{o}_{c}^{d} - \dot{o}_{d}^{d} = \boldsymbol{R}_{d}^{\mathrm{T}} \dot{o}_{c}$$

$$\dot{\phi}_{d,c} = \boldsymbol{T}(\phi_{d,c})^{-1} \omega_{d,c}^{d} = \boldsymbol{T}(\phi_{d,c})^{-1} \boldsymbol{R}_{d}^{\mathrm{T}} \omega_{c}$$

then taking in account that $\dot{\boldsymbol{o}}_{d}^{d} = \boldsymbol{0}$ and $\omega_{d}^{d} = \boldsymbol{0}$, being $\boldsymbol{o}_{d}$ and $\boldsymbol{R}_{d}$ costant yields

$$\dot{\tilde{\boldsymbol{x}}} = -\boldsymbol{T}_{A}(\phi_{d,c})^{-1} \begin{bmatrix} \boldsymbol{R}_{d}^{\mathrm{T}} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_{d}^{\mathrm{T}} \end{bmatrix} \boldsymbol{v}_{c}$$

- Since the end-effector frame and the camera frame coincide

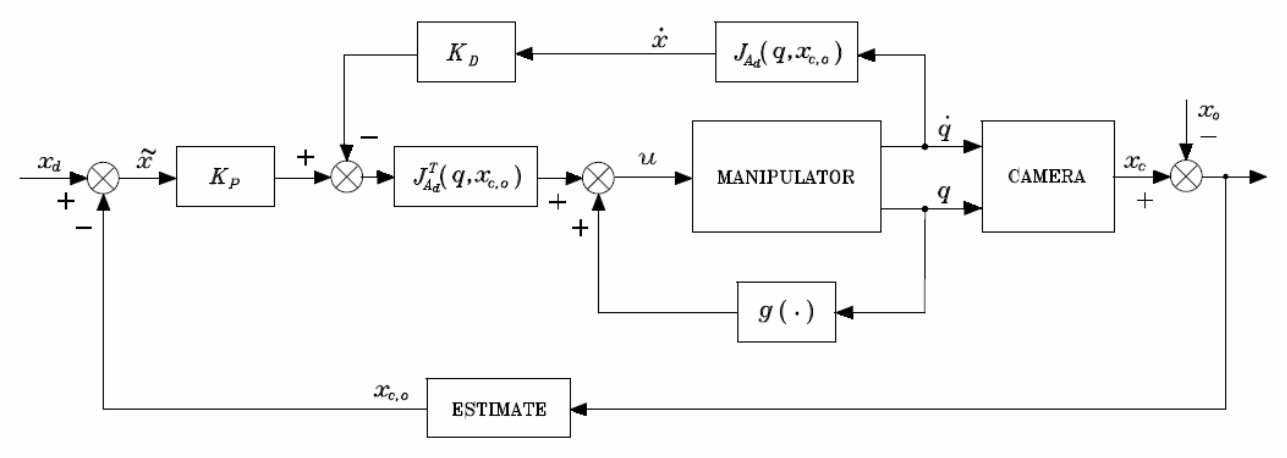
$$\dot{\tilde{\boldsymbol{x}}} = -\boldsymbol{J}_{A_q}(\boldsymbol{q}, \tilde{\boldsymbol{x}}) \dot{\boldsymbol{q}}$$

$$\boldsymbol{J}_{A_q}(\boldsymbol{q}, \tilde{\boldsymbol{x}}) = \boldsymbol{T}_{A}(\phi_{d,c})^{-1} \begin{bmatrix} \boldsymbol{R}_{d}^{\mathrm{T}} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_{d}^{\mathrm{T}} \end{bmatrix} \boldsymbol{J}(\boldsymbol{q})$$

- Position-based visual servoing of PD type with gravity compensation has the expression

$$\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{J}_{A_q}^{\mathrm{T}}(\boldsymbol{q}, \tilde{\boldsymbol{x}}) \boldsymbol{K}_{P} \tilde{\boldsymbol{x}} - \boldsymbol{J}_{A_q}^{\mathrm{T}} \boldsymbol{K}_{D} \boldsymbol{J}_{A_q}(\boldsymbol{q}, \tilde{\boldsymbol{x}}) \dot{\boldsymbol{q}}$$

which has the following block scheme (the sum blocks have a purely conceptual meaning!)



- The asymptotic stability of the equilibrium pose corresponding to $\tilde{\boldsymbol{x}} = \boldsymbol{0}$, under the assumption of symmetric and positive definite matrices $\boldsymbol{K}_{P}$ and $\boldsymbol{K}_{D}$, can be proven using the following Lyapunov function

$$V = \frac{1}{2}\dot{q}^{\mathrm{T}} B(q)\dot{q} + \frac{1}{2}\tilde{x}^{\mathrm{T}} K_P \tilde{x}$$

- The derivative term can also be chosen as $-K_D \dot{q}$

Resolved-velocity control

- Performance limits due to
  - Information deriving from visual measurements is computed at frequency lower or equal to the camera frame rate
  - Frequencies are at least of one order of magnitude lower than the typical frequencies used for motion control of robot manipulators
  - The control gains must be set to values much lower than those typically used for motion control to preserve stability of the closed-loop system
    - Low speed of convergence
    - Poor disturbance rejection capability
- This problem can be avoided assuming that the manipulator is equipped with a high-gain motion controller
  - Effects on the tracking errors deriving from manipulator dynamics and disturbances are neglected (manipulator as an ideal positioning device)

$$\boldsymbol{q}(t) \simeq \boldsymbol{q}_r(t)$$

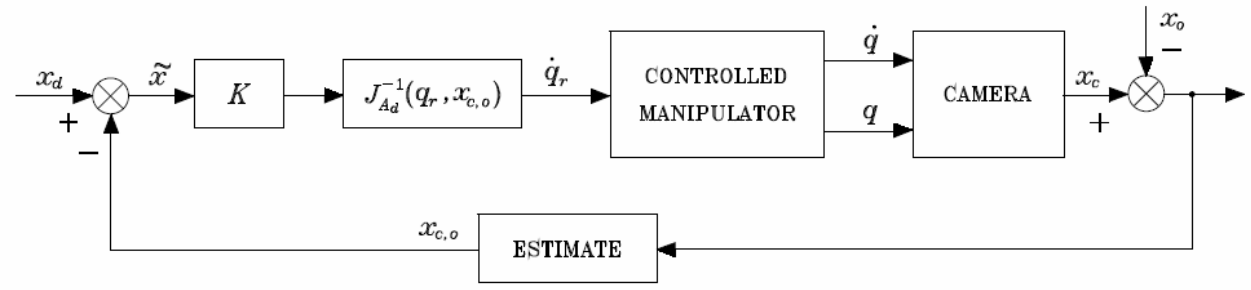
  - $\boldsymbol{q}_r$ is computed from visual measurements so that the operational space tracking error goes asymptotically to zero
  - From the equation about the error time derivative, it can be chosen

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}_{A_q}^{-1}(\boldsymbol{q}_r, \tilde{\boldsymbol{x}})\boldsymbol{K}\tilde{\boldsymbol{x}}$$

    which yields the linear equation

$$\dot{\tilde{\boldsymbol{x}}} + \boldsymbol{K}\tilde{\boldsymbol{x}} = \boldsymbol{0}$$

    which is asymptotically stable under the assumption that $\boldsymbol{K}$ is a positive definite matrix



- It is a *resolved-velocity control*, because it is based on the computation of velocity from the operational space error
- The choice of $\boldsymbol{K}$ influences the transient behavior of the trajectory of the camera frame
  - If $\boldsymbol{K}$ is a diagonal matrix with the same gains for the positional part, the origin of the camera frame follows the line segment connecting the initial position to the desired position
  - The orientation trajectory depends on the choice of $\boldsymbol{K}$ and on the particular choice of Euler angles
  - The possibility of knowing in advance the trajectory of the camera is important to avoid that object may exit from the camera field of view, making visual measurements unavailable

Image-based Visual Servoing

- A desired constant value $\boldsymbol{s}_d$ of the feature parameters is imposed, under the assumption that the object is fixed with respect to the base frame
  - It is assumed that the desired pose $\boldsymbol{s}_d = \boldsymbol{s}(\boldsymbol{x}_{d,o})$ belongs to the dexterous workspace of the manipulator and univocally determined
    - $n \geq 4$ points for coplanar points (no triplet of collinear points)
    - $n \geq 6$ points for non-coplanar points
    - In the case of $m < 6$, a reduced number of points can be used
  - $\boldsymbol{s}_d$ can be directly computed by measuring the feature parameters when the object is in the desired pose with respect to the camera
    - $\boldsymbol{x}_{d,o}$ does not need to be known
  - It is worth recalling that $\boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c)$ depends on $\boldsymbol{z}_c = [z_{c,1}, \dots, z_{c,n}]$
- The control law must be designed so as to guarantee that the image space error

$$\boldsymbol{e}_s = \boldsymbol{s}_d - \boldsymbol{s}$$

tends asymptotically to zero

PD control with gravity compensation

- Consider the following Lyapunov function

$$V(\dot{q}, e_s) = \frac{1}{2}\dot{q}^{\mathrm{T}} B(q)q + \frac{1}{2}e_s^{\mathrm{T}} K_{Ps} e_s > 0 \quad \forall \dot{q}, e_s \neq \mathbf{0}$$

with $\boldsymbol{K}_{P_S}$ symmetric and positive definite (kxk) matrix

Computing the time derivative and taking into account the expression of the joint space dynamic model of the manipulator gives

$$\dot{V} = -\dot{q}^{\mathrm{T}} F \dot{q} + \dot{q}^{\mathrm{T}}(u - g(q)) - \dot{e}_s^{\mathrm{T}} K_{Ps} e_s$$

Since $\dot{\boldsymbol{s}}_d = \mathbf{0}$ and the object is fixed with respect to the base frame, under the assumption that the camera frame coincides with the end-effector frame, it follows

$$\dot{e}_s = -\dot{s} = -\boldsymbol{J}_L(\boldsymbol{s}, z_c, q)\dot{q}$$

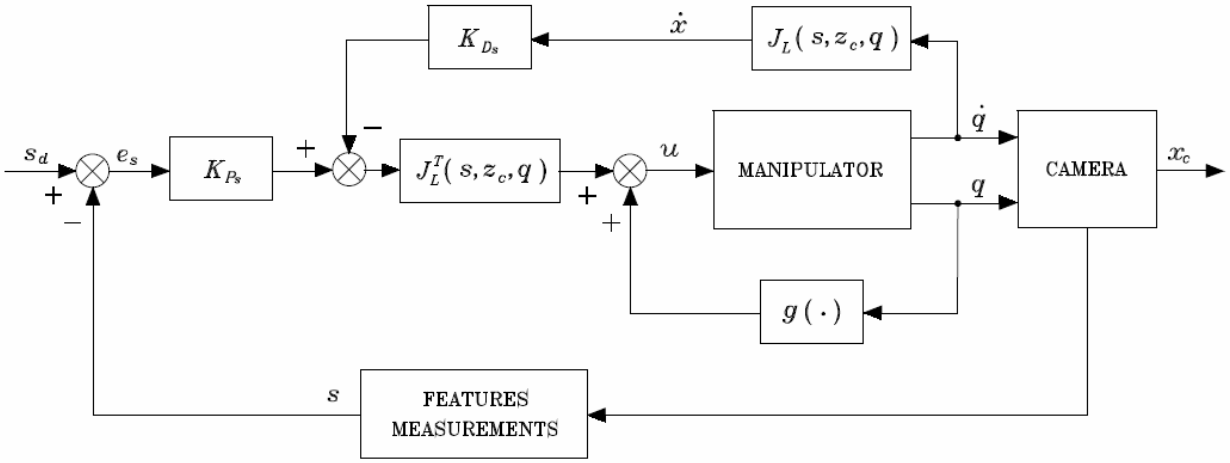
$$\boldsymbol{J}_L(\boldsymbol{s}, z_c, q) = \boldsymbol{L}_s(\boldsymbol{s}, z_c) \begin{bmatrix} R_c^{\mathrm{T}} & O \\ O & R_c^{\mathrm{T}} \end{bmatrix} J(q)$$

With the choice of the following control law

$$u = g(q) + J_L^{\mathrm{T}}(\boldsymbol{s}, z_c, q) K_{Ps} e_s - J_L^{\mathrm{T}} K_{Ds} J_L(\boldsymbol{s}, z_c, q)\dot{q}$$

with $\boldsymbol{K}_{D_S}$ symmetric and positive definite (kxk) matrix, it is

$$\dot{V} = -\dot{q}^{\mathrm{T}} F \dot{q} - \dot{q}^{\mathrm{T}} J_L^T K_{Ds} J_L \dot{q}$$



- The last term corresponds to a derivative action in the image space and has to be added to increase damping (derivative action in the image space)
  - It can be replaced with $-\boldsymbol{K}_{D_s}\dot{\boldsymbol{s}}$, but being $\dot{\boldsymbol{s}}$ not available, the term $-\boldsymbol{K}_D\dot{\boldsymbol{q}}$ can be used, with $\boldsymbol{K}_D$ symmetric and positive definite (nxn) matrix
- The system Lyapunov function decreases until $\dot{\boldsymbol{q}} \neq \boldsymbol{0}$ → the system reaches the equilibrium state characterized by

$$\boldsymbol{J}_L^{\mathrm{T}}(\boldsymbol{s}, z_c, \boldsymbol{q})\boldsymbol{K}_{Ps}\boldsymbol{e}_s = \boldsymbol{0}$$

  and if the interaction matrix and the geometric Jacobian are full rank, it will result $\boldsymbol{e}_s = \boldsymbol{0}$
- The control law requires the computation of $\boldsymbol{z}_c$, which in the image-based visual servoing philosophy should be avoided
  - In some applications $\boldsymbol{z}_c$ is known with good approximation (e.g. object belonging to a plane)
  - It is assumed $\boldsymbol{z}_c$ constant, as the value in the initial configuration or that in the desired configuration → it is equivalent to using an estimate $\hat{\boldsymbol{L}}_s$ of the interaction matrix
    - The stability proof becomes much more complex!

Resolved-velocity control

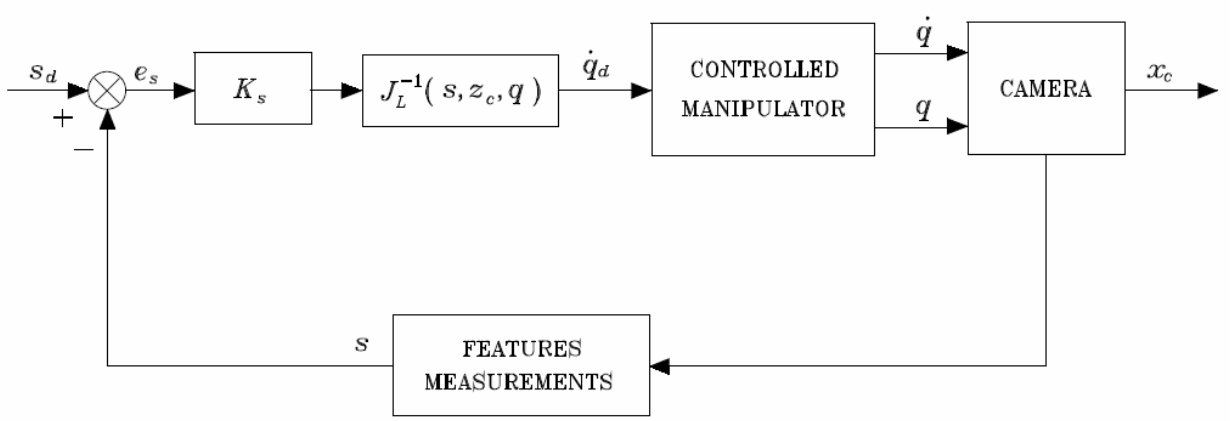
- If the following choice of the reference velocity in the joint space is assumed

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}_L^{-1}(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q}_r)\boldsymbol{K}_s\boldsymbol{e}_s$$

under the assumption of invertible matrix $\boldsymbol{J}_L$, the following linear equation of the error holds

$$\dot{\boldsymbol{e}}_s + \boldsymbol{K}_s\boldsymbol{e}_s = \boldsymbol{0}$$

If $\boldsymbol{K}_s$ is a positive definite matrix, the error tends asymptotically to zero



- The system is affected by problems related to the singularities of $\boldsymbol{J}_L$
  - The singularities are both those of the geometric Jacobian and those of the interaction matrix (!)
  - The control law is computed in two steps

$$\boldsymbol{v}_r^c = \boldsymbol{L}_s^{-1}(\boldsymbol{s}, \boldsymbol{z}_c)\boldsymbol{K}_s\boldsymbol{e}_s$$

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}^{-1}(\boldsymbol{q}) \begin{bmatrix} \boldsymbol{R}_c & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_c \end{bmatrix} \boldsymbol{v}_r^c$$

  - Using a number k of feature parameters greater than the minimum required m, the left pseudo-inverse can be used

$$\boldsymbol{v}_r^c = (\boldsymbol{L}_s^{\mathrm{T}}\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^{\mathrm{T}}\boldsymbol{K}_s\boldsymbol{e}_s$$

- Stability of the closed-loop system can be shown using the Lyapunov function

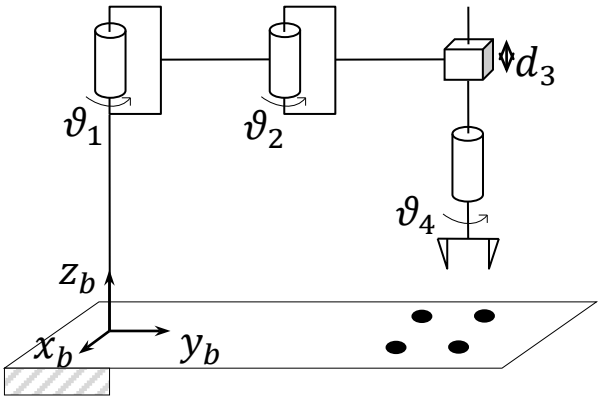
$$V(\boldsymbol{e}_s) = \frac{1}{2}\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{K}_s\boldsymbol{e}_s$$

$$\dot{V} = -\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{K}_s\boldsymbol{L}_s(\boldsymbol{L}_s^{\mathrm{T}}\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^{\mathrm{T}}\boldsymbol{K}_s\boldsymbol{e}_s$$

  which is negative semi-definite because $\mathcal{N}\left(\boldsymbol{L}_s{}^{\mathrm{T}}\right) \neq \emptyset$, being $\boldsymbol{L}_s{}^{\mathrm{T}}$ a matrix with more columns than rows → the system is stable but not asymptotically stable
    - The error is bounded, but in some cases the system may reach an equilibrium with $\boldsymbol{e}_s \neq 0$ and $\boldsymbol{K}_s\boldsymbol{e}_s \in \mathcal{N}\left(\boldsymbol{L}_s{}^{\mathrm{T}}\right)$
- The computation of $\boldsymbol{L}_s$ requires the knowledge of z_c
    - An estimate of matrix $\hat{\boldsymbol{L}}_s^{-1}$ (or of the pseudo-inverse) can be used
    - The Lyapunov method can be used to prove that the control scheme remains stable provided that matrix $\boldsymbol{L}_s\hat{\boldsymbol{L}}_s^{-1}$ is positive definite
    - Notice that $\boldsymbol{z}_c$ is the only information depending on the object geometry
- The choice of $\boldsymbol{K}_s$ influences the trajectories of the feature parameters
    - In the case of features points, if a scalar matrix $\boldsymbol{K}_s$ is set, the projections of these points on the image plane will follow line segments
        - The corresponding camera motion cannot be easily predicted!

Comparison Among Various Control Schemes

- SCARA manipulator ($m = 4$) with planar object (4 points)



- A dynamic model is considered for the manipulator
- The control is implemented with a sample time of 0.04s (25Hz), which coincides with the minimum frame rate of analog cameras
- Visual servoing in the operational space: $\boldsymbol{x}_{c,o}$ estimated with a numerical algorithm using only P_1 and P_2 ($\boldsymbol{J}_{A_s}$ is a (4x4) square matrix), integration step of 1ms and $\boldsymbol{K}_s = 160\boldsymbol{I}_4$ (converges in 0.03s)
- Visual servoing in the image space: only points P_1 and P_2 ($\boldsymbol{J}_L$ (4x4) square matrix)
- Initial and desired values:

$$\boldsymbol{x}_c(0) = [1 \quad 1 \quad 1 \quad \pi/4]^T$$
$$\boldsymbol{x}_{c,o}(0) = [0 \quad 0 \quad 0.5 \quad 0]^T$$
$$\boldsymbol{x}_{d,o} = [-0.1 \quad 0.1 \quad 0.6 \quad -\pi/3]^T$$
$$\boldsymbol{s}_d = [-0.1667 \quad 0.1667 \quad -0.0833 \quad 0.0223]^T$$

- The dynamics of the velocity-controlled manipulator has been neglected
- $\hat{\boldsymbol{L}}_s = \boldsymbol{L}_s(\boldsymbol{s}, z_d)$, with z_d the third component of $\boldsymbol{x}_{d,o}$

- **Scheme A**: Position-based visual servoing of PD type with gravity compensation

$$\boldsymbol{K}_P = \mathrm{diag}\{500, 500, 10, 10\}$$
$$\boldsymbol{K}_D = \mathrm{diag}\{500, 500, 10, 10\}$$

- **Scheme B**: Resolved-velocity position-based visual servoing

$$\boldsymbol{K} = \mathrm{diag}\{1, 1, 1, 2\}$$

corresponding to a time constant of 1s for the three position variables and of 0.5s for the orientation variable

- **Scheme C:** Image-based visual servoing of PD type with gravity compensation

$$\boldsymbol{K}_{Ps} = 300\boldsymbol{I}_4 \qquad \boldsymbol{K}_{Ds} = 330\boldsymbol{I}_4$$

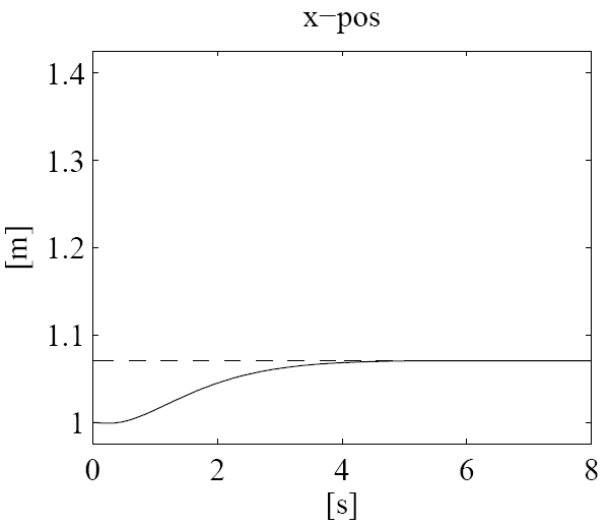
- **Scheme D**: Resolved-velocity image-based visual servoing

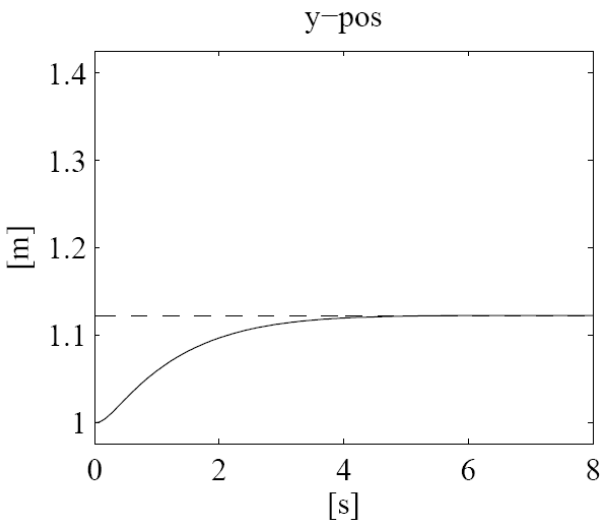
$$\boldsymbol{K}_s = \boldsymbol{I}_4$$

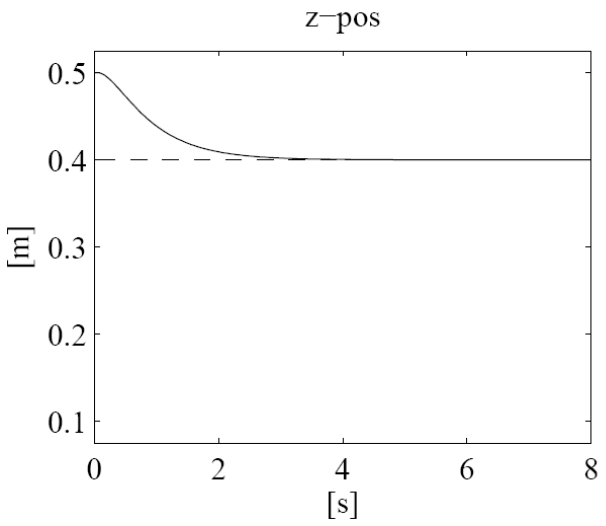
corresponding to a time constant of 1s for the feature parameters

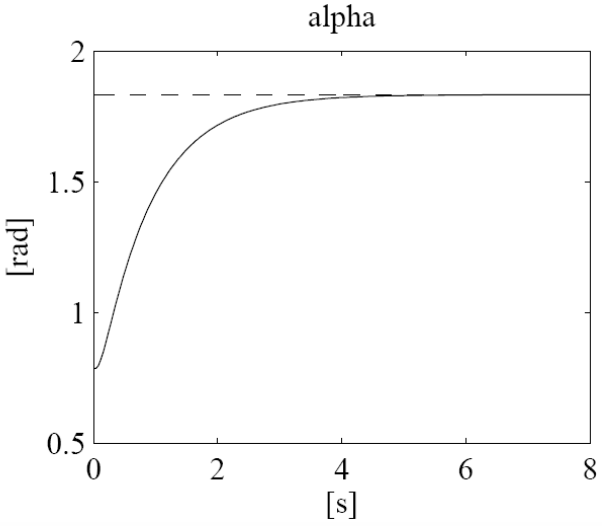
The parameters of the various controllers have been chosen in such a way as to show the particular features of the different control laws and, at the same time, to allow a significant comparison of the performance of each scheme

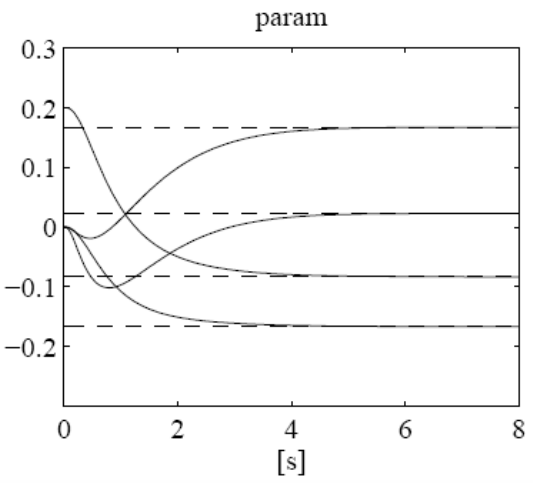
Scheme A

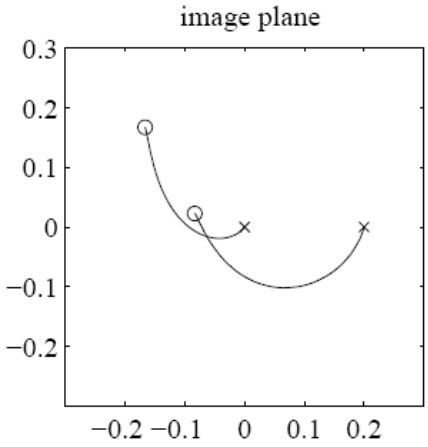




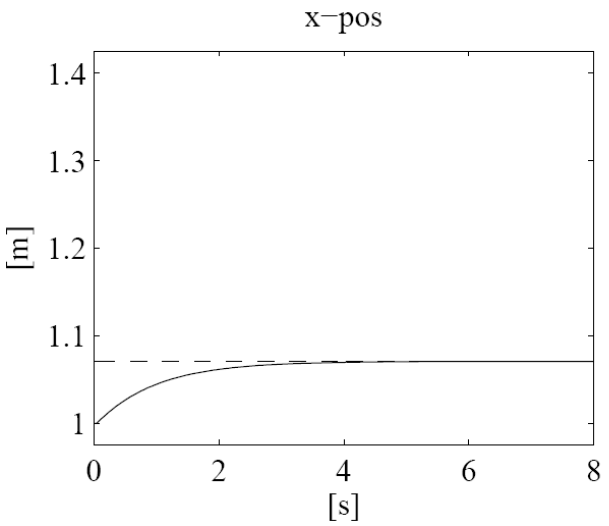


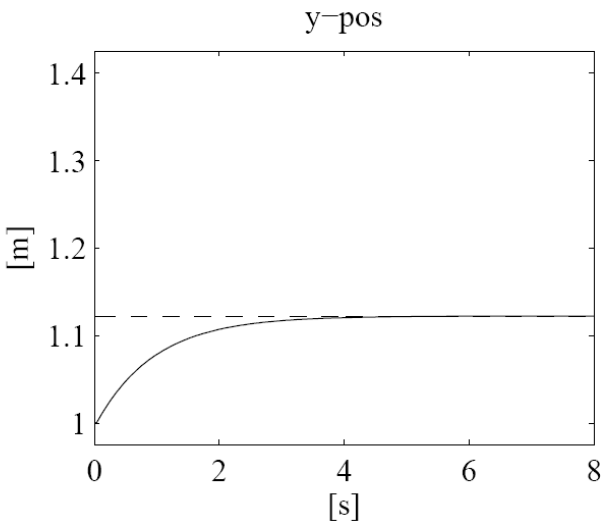


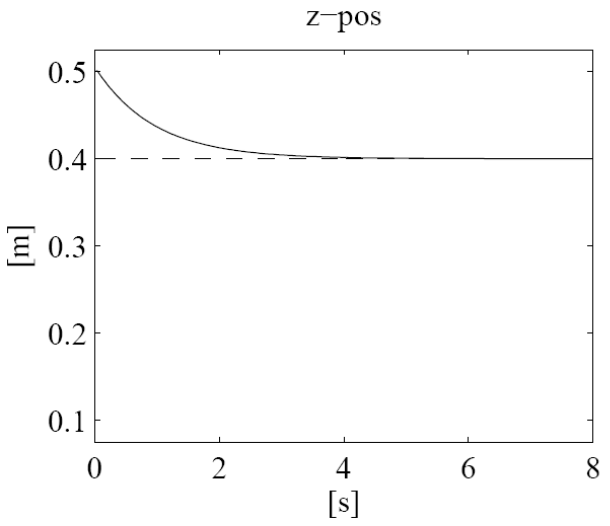


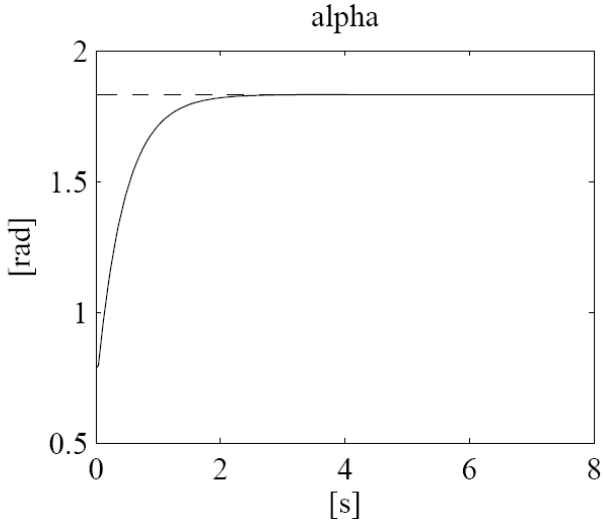


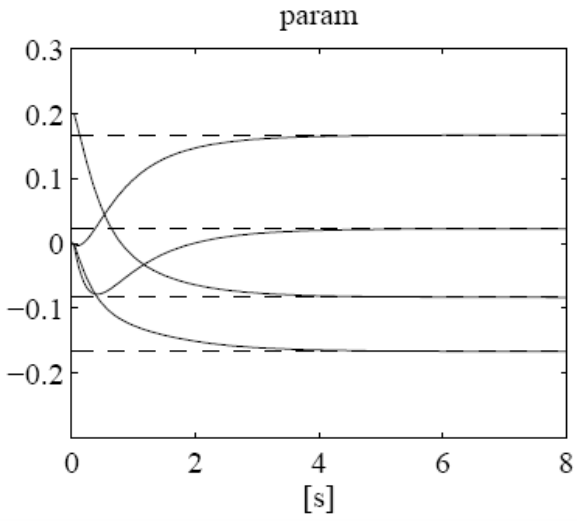
Scheme B

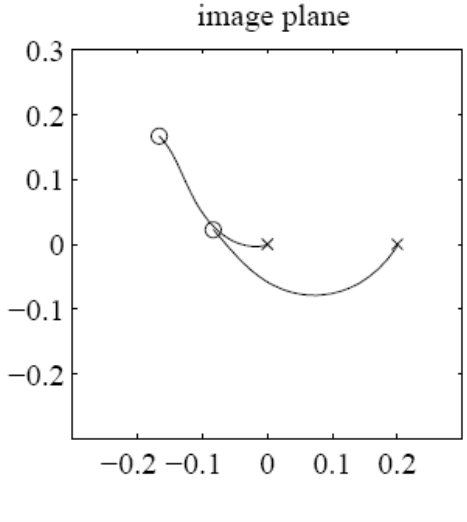




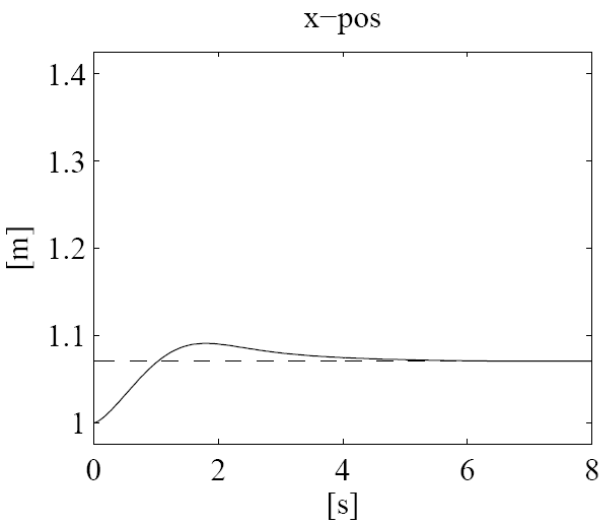


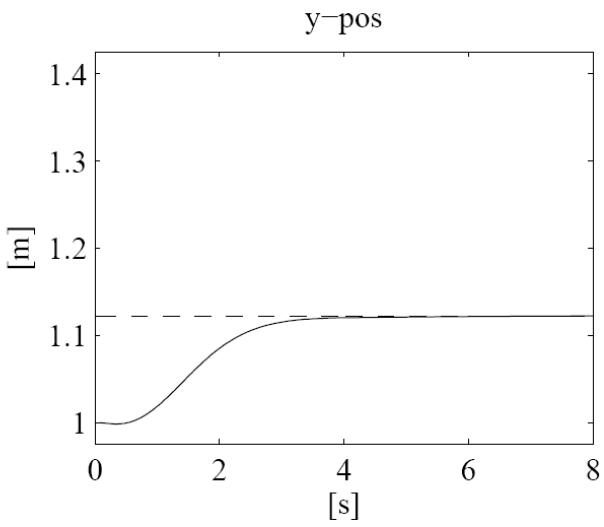


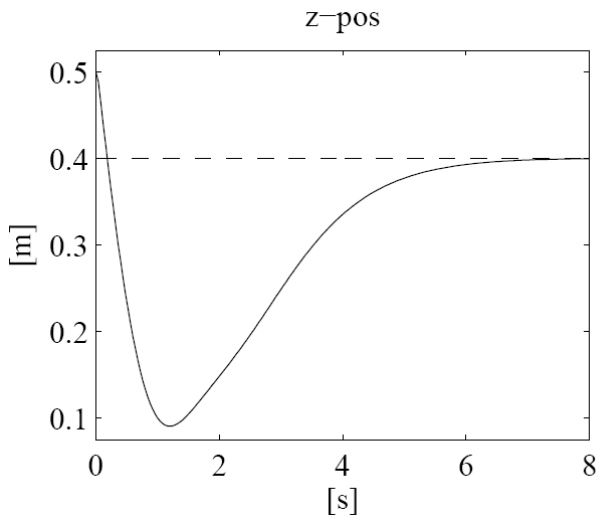


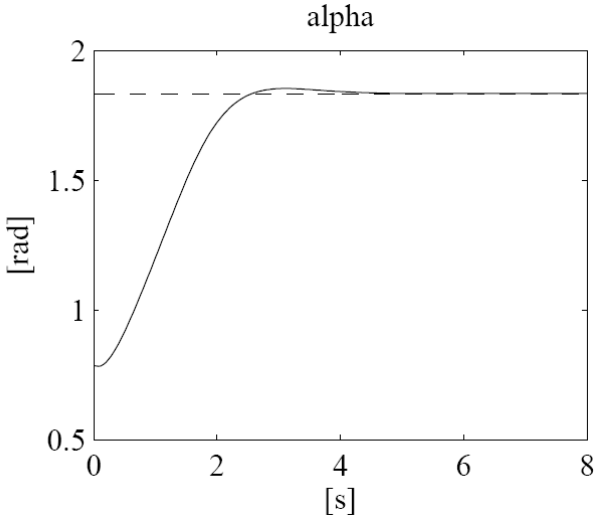


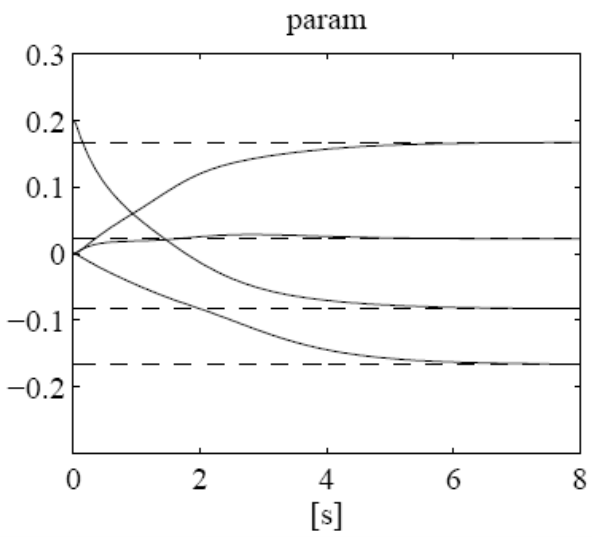
Scheme C

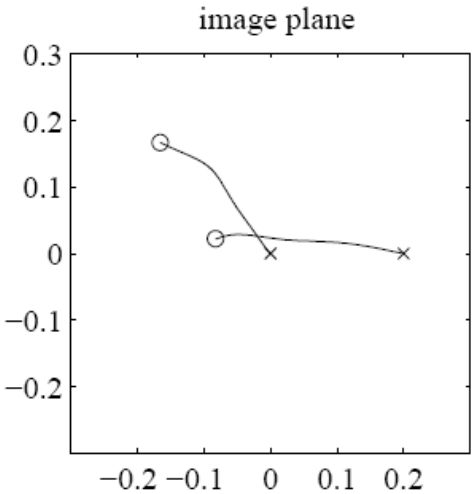




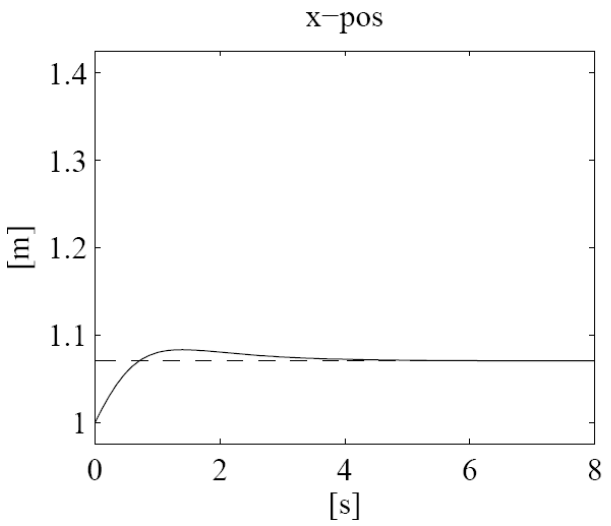


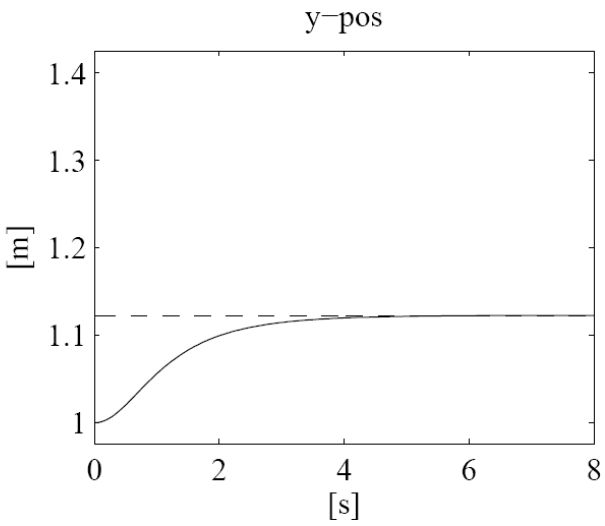


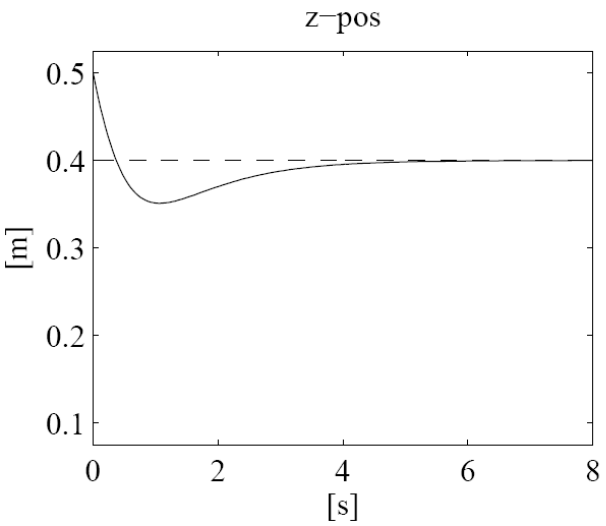


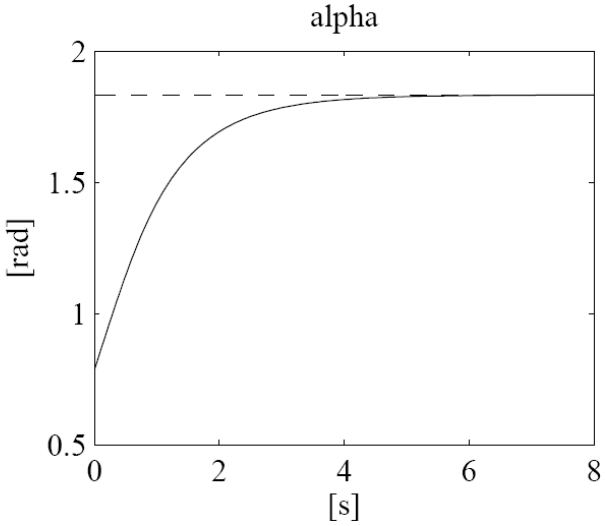


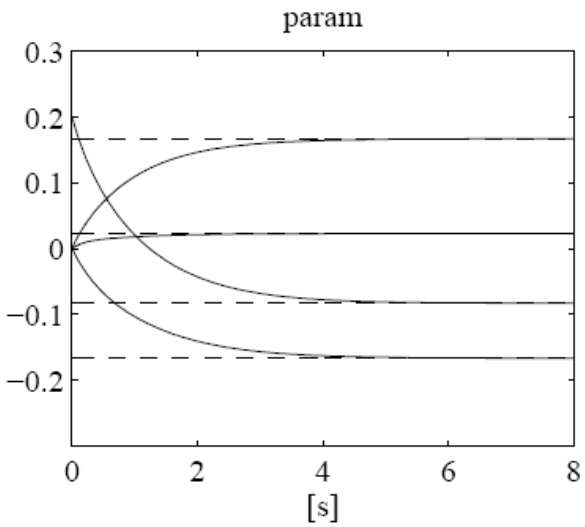
Scheme D

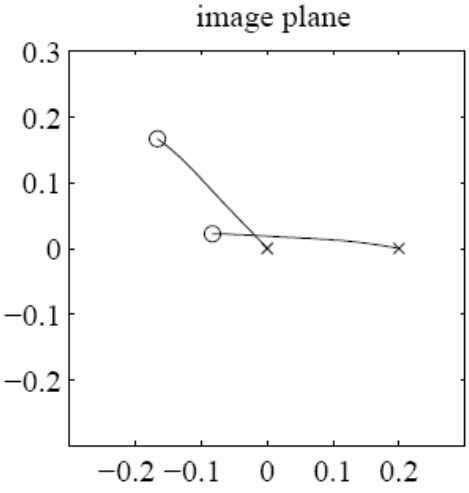




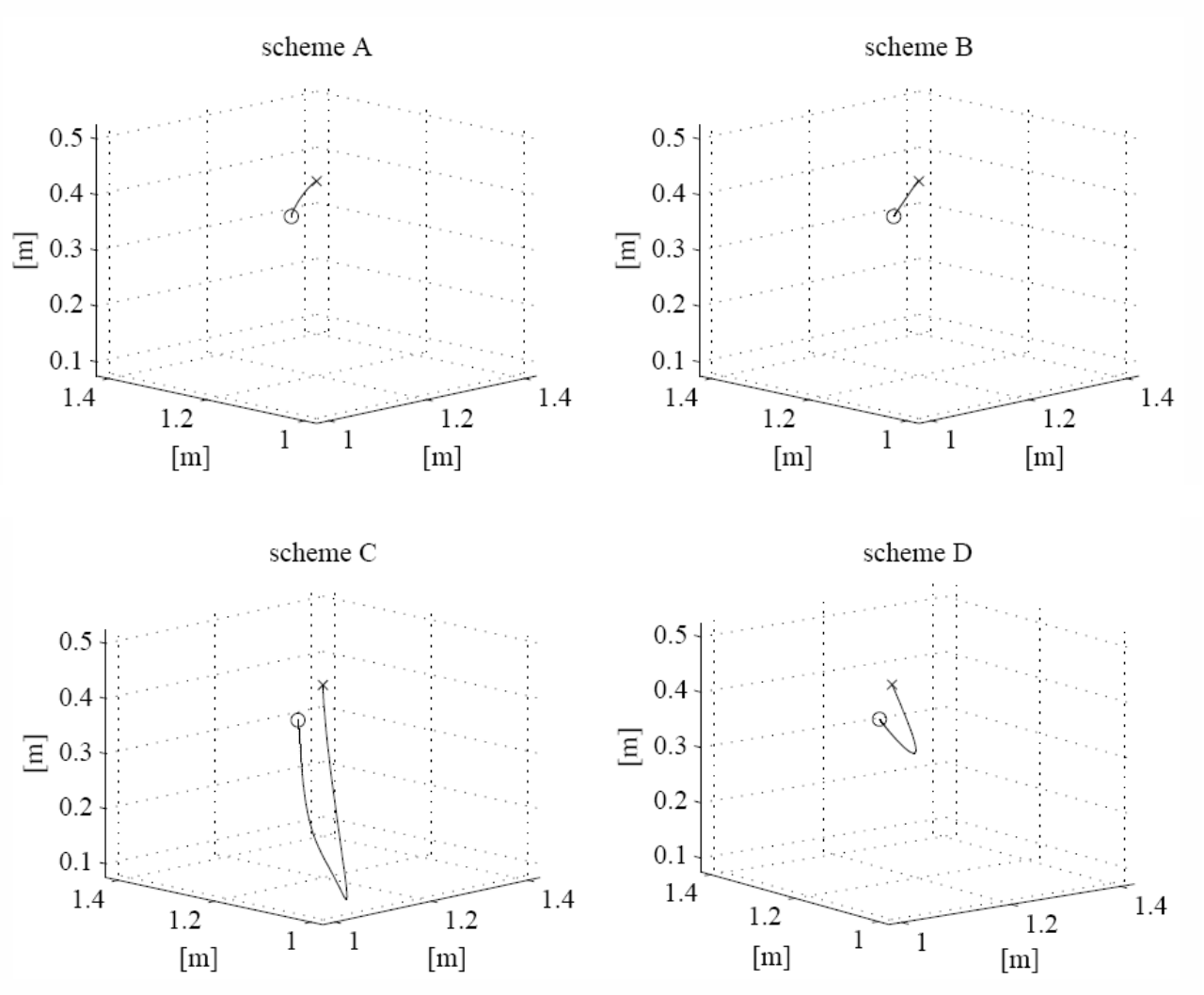








Paths of camera frame origin



Hybrid Visual Servoing

- It combines the benefits of position-based and image-based visual servoing
  - The control error is defined in the operational space for some components and in the image space for the others
    - Control on the operational space trajectories
    - Control on keeping image features in the camera field of view
- Estimation of some operational space variables is required
  - e.g. using planar homography in the case of planar object with at least 4 points (only $\boldsymbol{s}$ and $\boldsymbol{s}_d$ are necessary)
- Imposing the planar homography between the desired camera frame and the current one yields

$$\boldsymbol{H} = \boldsymbol{R}_d^c + \frac{1}{d_d}\boldsymbol{o}_{c,d}^c \boldsymbol{n}^{d\mathrm{T}}$$

  From $\boldsymbol{H}$ it is possible to compute $\boldsymbol{n}^{d\mathrm{T}}$, $\boldsymbol{R}_d^c$ e $\boldsymbol{o}_{c,d}^c/d_c$ at each sampling time
- By adopting a resolved-velocity approach, the control objective consists of computing the reference absolute velocity of the camera frame

$$\boldsymbol{v}_r^c = \begin{bmatrix} \boldsymbol{\nu}_r^c \\ \boldsymbol{\omega}_r^c \end{bmatrix}$$

- From matrix $\boldsymbol{R}_d^c$ the orientation error can be computed
  - Having defined with $\boldsymbol{\phi}_{c,d}$ the vector of the Euler angles extracted from $\boldsymbol{R}_d^c$, the control vector can be chosen as

$$\boldsymbol{\omega}_r^c = -\boldsymbol{T}(\phi_{c,d})\boldsymbol{K}_o\phi_{c,d}$$

  where $\boldsymbol{K}_o$ is a symmetric positive definite (3x3) matrix, from which

$$\dot{\phi}_{c,d} + \boldsymbol{K}_o\phi_{c,d} = \boldsymbol{0}$$

implying that the orientation error tends to zero asymptotically with convergence of exponential type

- The control vector $\boldsymbol{\nu}_r^c$ should be selected so that the positional part of the error between the desired and the current camera pose converges to zero
  - It is not possible to define the error as the difference of the coordinates of a point of the object (these cannot be directly measured)
    - The corresponding coordinates in the image plane are available

$$\boldsymbol{s}_{p,d} = [X_d \quad Y_d]^{\mathrm{T}} = [x_d/z_d \quad y_d/z_d]^{\mathrm{T}}$$
$$\boldsymbol{s}_p = [X \quad Y]^{\mathrm{T}} = [x_c/z_c \quad y_c/z_c]^{\mathrm{T}}$$

  - From the computation of the homography it can be possible to compute the values

$$\rho_z = z_c/z_d = \frac{d_c}{d_d}\frac{\boldsymbol{n}^{d\mathrm{T}}\tilde{\boldsymbol{s}}_{p,d}}{\boldsymbol{n}^{c\mathrm{T}}\tilde{\boldsymbol{s}}_p}$$

$$\frac{d_c}{d_d} = 1 + \boldsymbol{n}^{cT}\frac{\boldsymbol{o}_{c,d}^c}{d_d} = \det(\boldsymbol{H})$$

with $\boldsymbol{n}^c = \boldsymbol{R}_d^c\boldsymbol{n}^d$

  - The position error can be defined as

$$\boldsymbol{e}_p(\boldsymbol{r}_d^c, \boldsymbol{r}_c^c) = \begin{bmatrix} X_d - X \\ Y_d - Y \\ \ln\rho_z \end{bmatrix}$$

noticing that convergence to zero of $\boldsymbol{e}_p$ implies convergence to zero of $\boldsymbol{r}_d^c - \boldsymbol{r}_c^c$ and vice versa

  - Computing the time derivative of $\boldsymbol{e}_p$, being $\boldsymbol{r}_d^c$ constant, yields

$$\dot{\boldsymbol{e}}_p = \frac{\partial \boldsymbol{e}_p(\boldsymbol{r}_c^c)}{\partial \boldsymbol{r}_c^c}\dot{\boldsymbol{r}}_c^c = -\boldsymbol{J}_p\boldsymbol{\nu}_c^c - \boldsymbol{J}_o\boldsymbol{\omega}_c^c$$

where

$$\boldsymbol{J}_p = \frac{1}{z_d \rho_z}\begin{bmatrix} -1 & 0 & X \\ 0 & -1 & Y \\ 0 & 0 & -1 \end{bmatrix} \boldsymbol{J}_o = \begin{bmatrix} XY & -1-X^2 & Y \\ 1+Y^2 & -XY & -X \\ -Y & X & 0 \end{bmatrix}$$

- The previous equation suggests the following choice (being $\boldsymbol{J}_p(z_d)$ invertible)

$$\boldsymbol{\nu}_r^c = \boldsymbol{J}_p^{-1}(\boldsymbol{K}_p \boldsymbol{e}_p - \boldsymbol{J}_o \boldsymbol{\omega}_r^c)$$

  - Under the assumption that z_d is known, and $\boldsymbol{v}_c^c \approx \boldsymbol{v}_r^c$ and $\boldsymbol{\omega}_c^c \approx \boldsymbol{\omega}_r^c$, it yields

$$\dot{\boldsymbol{e}}_p + \boldsymbol{K}_p \boldsymbol{e}_p = \boldsymbol{0}$$

  - If z_d is not known, the estimate can be adopted

$$\hat{\boldsymbol{J}}_p^{-1} = \frac{\hat{z}_d}{z_d}\boldsymbol{J}_p^{-1}$$

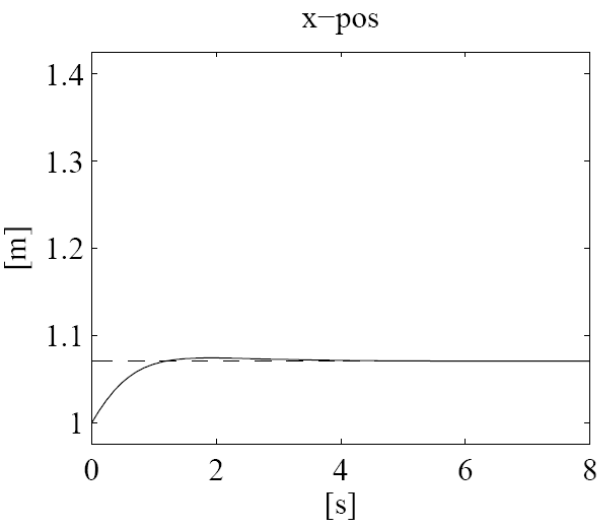
    from which the following error equation is obtained

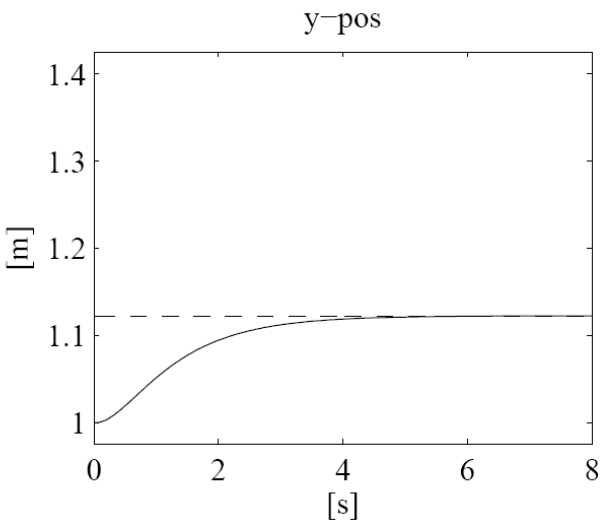
$$\dot{\boldsymbol{e}}_p + \frac{\hat{z}_d}{z_d}\boldsymbol{K}_p \boldsymbol{e}_p = \left(1 - \frac{\hat{z}_d}{z_d}\right)\boldsymbol{J}_o \boldsymbol{\omega}_r^c$$

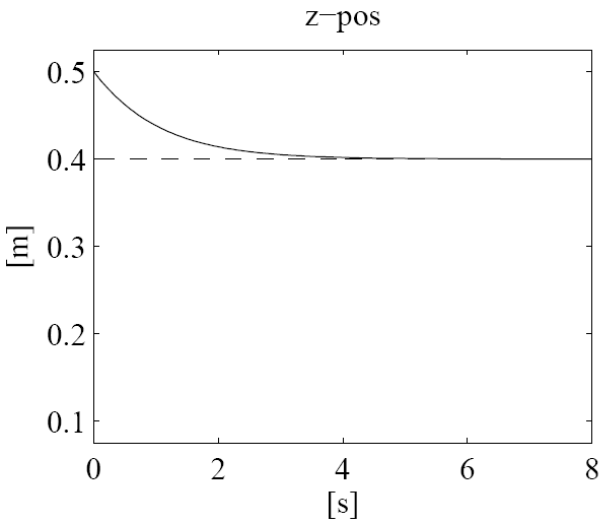
    where the gain is scaled and the time history of $\boldsymbol{e}_p$ depends on $\boldsymbol{\omega}_r^c$

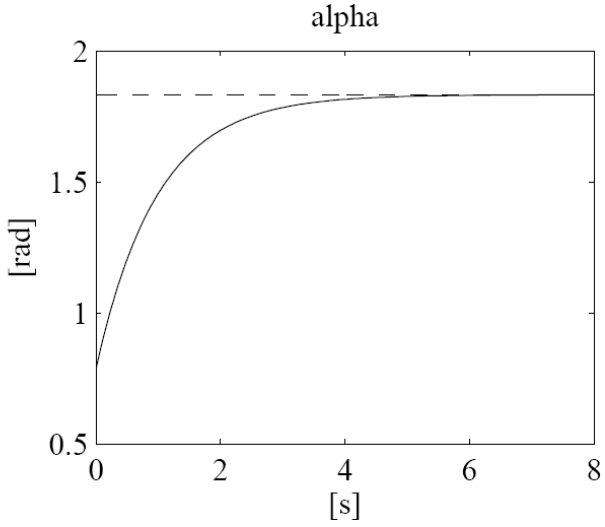
- This method is only one of the possible visual servoing approaches based on the computation of the planar homography and on its decomposition

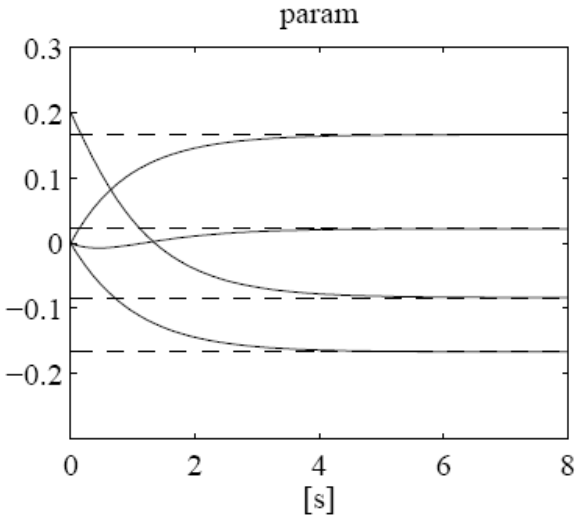
Hybrid control scheme

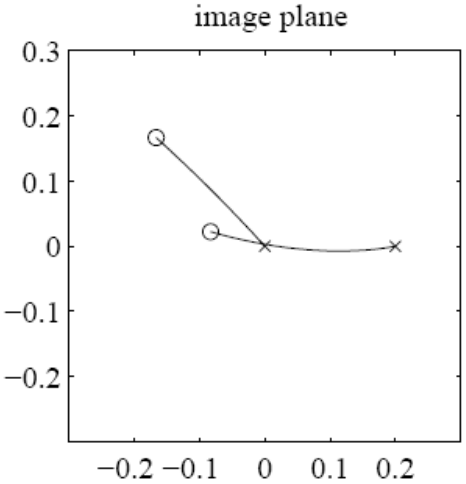












Path of camera frame origin