UiO : Department of Technology Systems
University of Oslo

10. Visual Servoing

Kim Mathiassen

Lecture Overview

- Vision for control (10.1)
- Interaction matrix (10.3.2-10.3.3)
- The visual servoing problem (10.6)
- Position-based visual servoing (10.7)
- Image-based visual servoing (10.8)

What this lecture not will cover

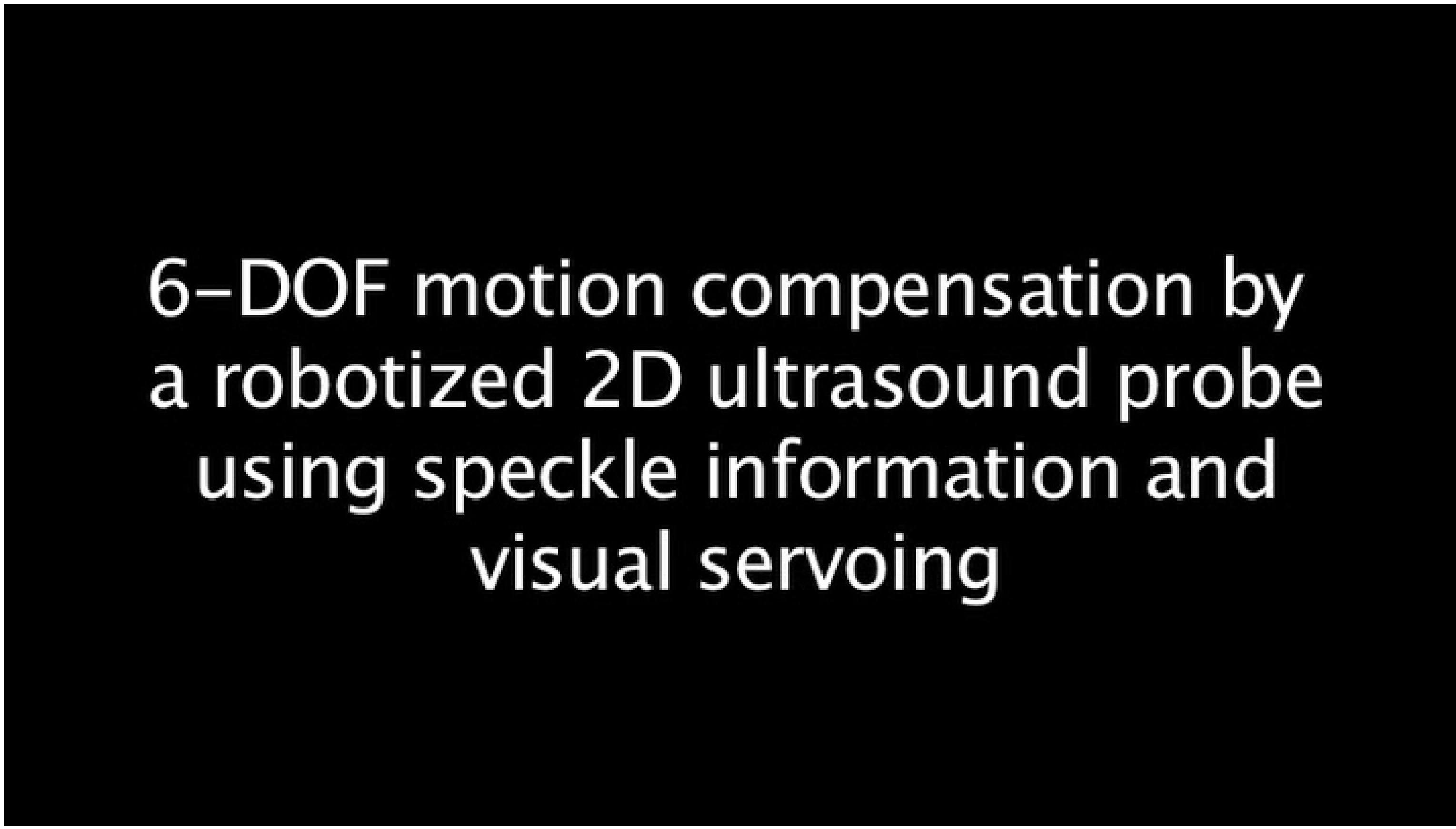
- Image processing (10.2)
- Pose estimation (10.3.1)
- Stereo vision (10.4)
- Camera calibration (10.5)

- For those subjects take UNIK4690 Machine vision

Motivation

- Eyes are one of the primary sensors for humans
- Find object in the environment and plan to grasp them (open loop)
- Visual servoing
  - Use visual measurement directly in a feedback loop
- Two main approaches
  - Position-based visual servoing
    - Uses operation space
    - Reconstucts target relative pose
  - Image-based visual servoing
    - Uses image space
    - Compare current and desired image features

Motivation

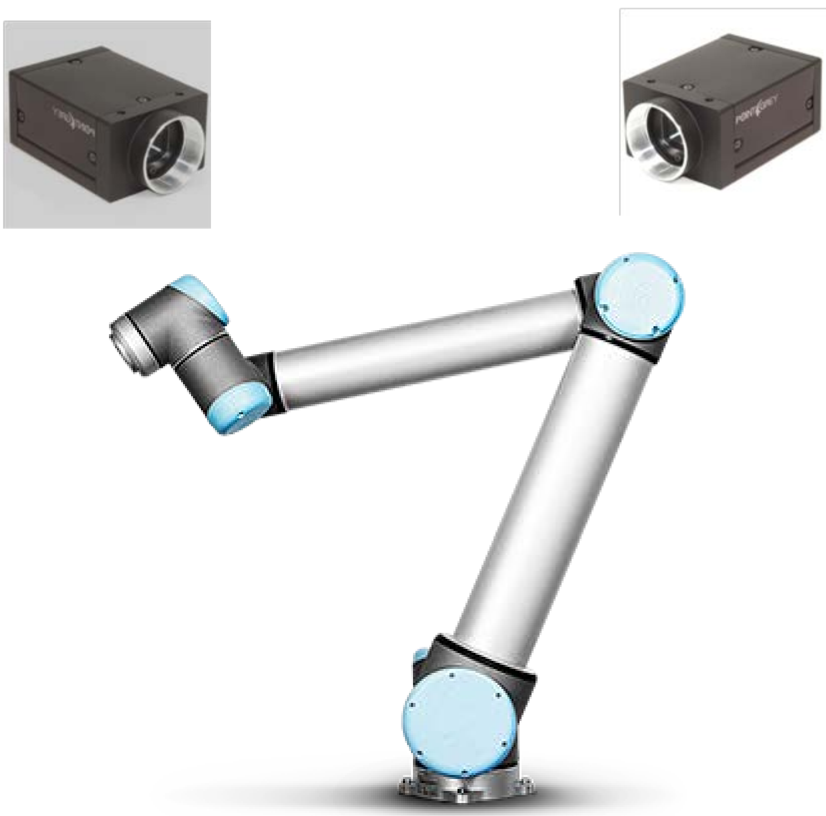

6–DOF motion compensation by
a robotized 2D ultrasound probe
using speckle information and
visual servoing

Motivation

- https://www.youtube.com/watch?v=7A5cqUEKXHg
- https://www.youtube.com/watch?v=97M3xUYo-tc

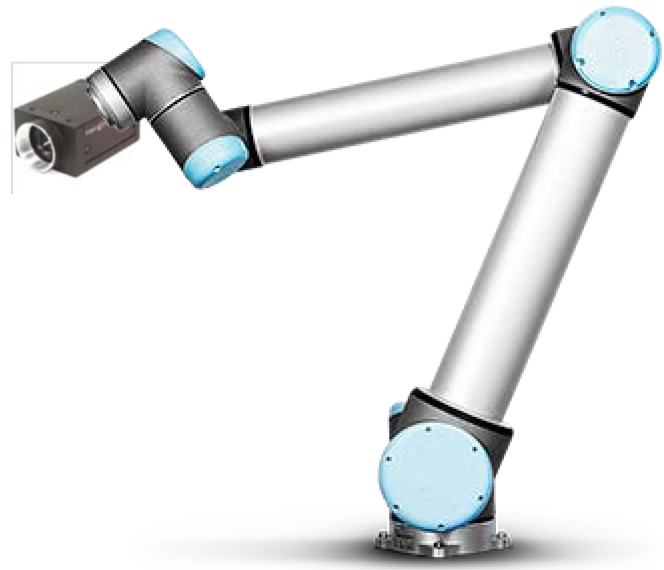
Configurations

Eye-to-hand



One or more cameras observe the robot

Eye-in-hand



The robot holds the camera

Configurations

- Eye-to-hand (fixed configuration)
  - Advantage: camera view does not change during the task
  - Disadvantage: robot may occlude the view
- Eye-in-hand (mobile configuration)
  - Advantage: No occlusions
  - Disadvantage: Varying measurement accuracy (distance to object varies) and camera field of view changes significantly
- Hybrid configuration
  - Mixing the two above
- Stereo vision
  - Using two or more cameras to retrieve depth information from the images

10.3.2 Interaction matrix

- The interaction matrix describes the relationship between the velocity of the camera (held by the robot) and the image features
- A feature is given as

$$s = \begin{bmatrix} X \\ Y \end{bmatrix}, \qquad (10.3)$$

- Where X and Y are normalized coordinates (not pixel coordinates)

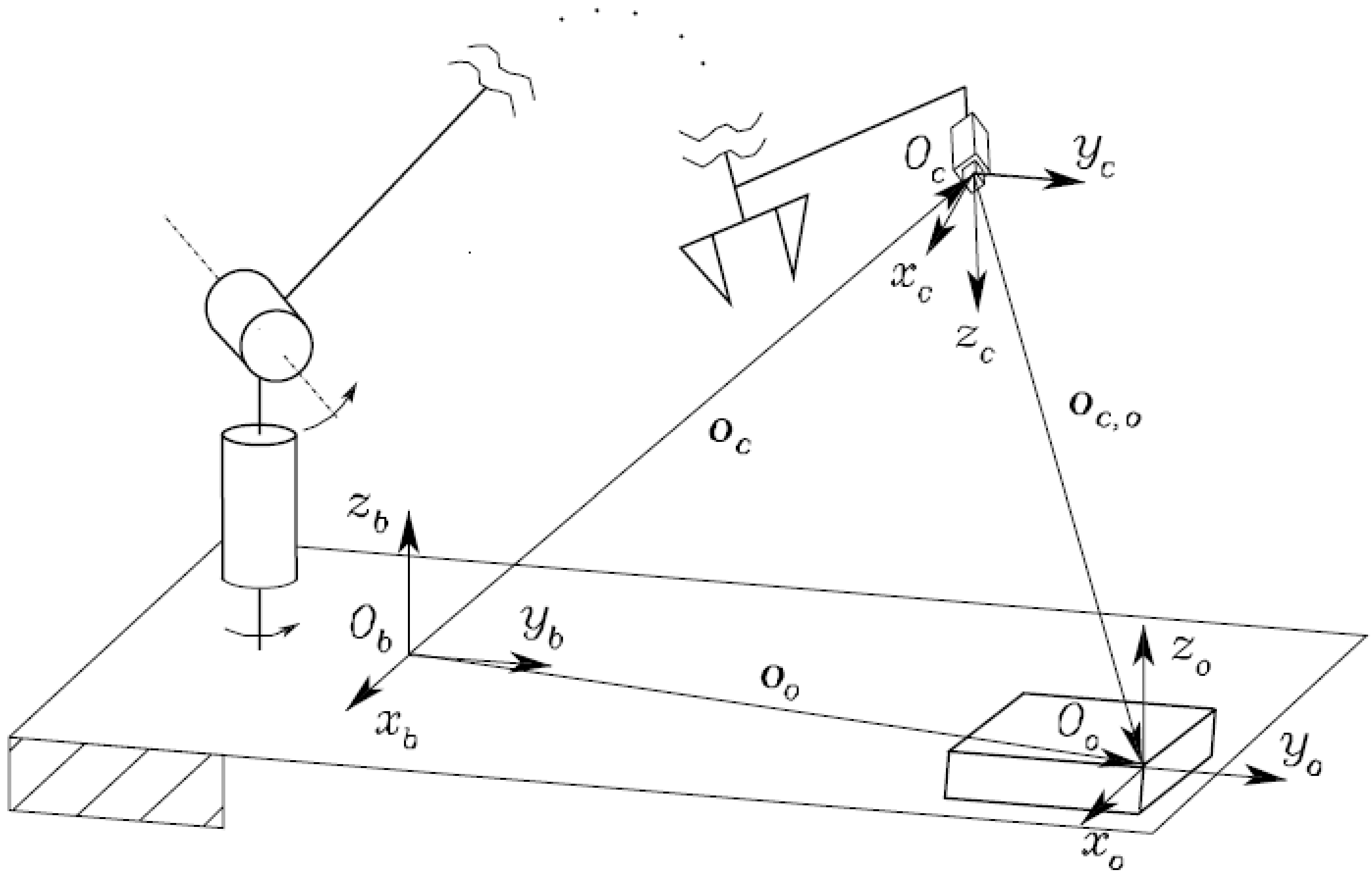


Fig. 10.5. Reference frames for an eye-in-hand camera

Interaction matrix

- If there is relative motion between the camera and the object it is possible to find a velocity vector $\dot{s}$
- The motion of the object relative to the camera is defined as

$$\boldsymbol{v}^c_{c,o} = \begin{bmatrix} \dot{\boldsymbol{o}}^c_{c,o} \\ \boldsymbol{R}^T_c(\boldsymbol{\omega}_o - \boldsymbol{\omega}_c) \end{bmatrix}, \qquad (10.16)$$

- Where $\dot{\boldsymbol{o}}^c_{c,o}$ is the time derivative of $\boldsymbol{o}^c_{c,o} = \boldsymbol{R}^T_c(\boldsymbol{o}_o - \boldsymbol{o}_c)$ and $\boldsymbol{\omega}_o$ and $\boldsymbol{\omega}_c$ are the angular velocity of the object and camera respectively

Interaction matrix

- The relationship between the image features velocity and the object velocity is

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

- $\boldsymbol{J}_s$ is termed the image Jacobian
- This generally depends on the feature vector and the relative pose between the camera and object

$$\boldsymbol{\Gamma}(\cdot) = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{S}(\cdot) \\ \boldsymbol{O} & -\boldsymbol{I} \end{bmatrix}$$

Interaction matrix

$$\boldsymbol{S} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix},$$

- It is useful to transform this equation from the relative velocity between the object and camera frame to the absolute velocity between to two frames

$$\boldsymbol{v}_c^c = \begin{bmatrix} \boldsymbol{R}_c^T \dot{\boldsymbol{o}}_c \\ \boldsymbol{R}_c^T \boldsymbol{\omega}_c, \end{bmatrix} \qquad \boldsymbol{v}_o^c = \begin{bmatrix} \boldsymbol{R}_c^T \dot{\boldsymbol{o}}_o \\ \boldsymbol{R}_c^T \boldsymbol{\omega}_o \end{bmatrix}$$

- We then get

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

- And we can rewrite the velocity as

$$\boldsymbol{v}_{c,o}^c = \boldsymbol{v}_o^c + \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^c) \boldsymbol{v}_c^c, \qquad (10.18)$$

Interaction matrix

- We then get

$$\dot{\boldsymbol{s}} = \boldsymbol{J}_s \boldsymbol{v}_o^c + \boldsymbol{L}_s \boldsymbol{v}_c^c, \tag{10.19}$$

- Where

$$\boldsymbol{L}_s = \boldsymbol{J}_s(\boldsymbol{s}, \boldsymbol{T}_o^c)\boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^c) \tag{10.20}$$

- $\boldsymbol{L}_s$ is termed the interaction matrix
- For fixed objects this matrix defines the linear mapping between the camera velocity and the change in image features
- The interaction matrix is in general simpler than the image Jacobian. The later may be found using

$$\boldsymbol{J}_s(\boldsymbol{s}, \boldsymbol{T}_o^c) = \boldsymbol{L}_s \boldsymbol{\Gamma}(-\boldsymbol{o}_{c,o}^c), \tag{10.21}$$

Interaction matrix of a point

- Consider a point P which can be represented in the camera frame with the following vector

$$r_c^c = R_c^T(p - o_c), \tag{10.22}$$

- Where p is the position of point P in the base frame
- We choose the coordinates as the feature vector for the point

$$s(r_c^c) = \frac{1}{z_c}\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix}, \tag{10.24}$$

- Where $r_c^c = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$

$$\dot{R} = S(\omega)R.$$

$$(R\mathbf{a}) \times (R\mathbf{b}) = R(\mathbf{a} \times \mathbf{b})$$

Interaction matrix of a point

- Computing the time derivative of the feature vector yields

$$\dot{\boldsymbol{s}} = \frac{\partial \boldsymbol{s}(\boldsymbol{r}_c^c)}{\partial \boldsymbol{r}_c^c} \dot{\boldsymbol{r}}_c^c, \qquad (10.25)$$

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

- Then taking the time derivative of $\boldsymbol{r}_c^c$ under the assumption of constant $\boldsymbol{p}$ yields

$$\dot{\boldsymbol{r}}_c^c = -\boldsymbol{R}_c^T \dot{\boldsymbol{o}}_c + \boldsymbol{S}(\boldsymbol{r}_c^c)\boldsymbol{R}_c^T \boldsymbol{\omega}_c = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{S}(\boldsymbol{r}_c^c) \end{bmatrix} \boldsymbol{v}_c^c. \qquad (10.26)$$

Interaction matrix of a point

- Combining the equations together yields

$$\boldsymbol{L}_s(\boldsymbol{s}, z_c) = \begin{bmatrix} -\frac{1}{z_c} & 0 & \frac{X}{z_c} & XY & -(1+X^2) & Y \\ 0 & -\frac{1}{z_c} & \frac{Y}{z_c} & 1+Y^2 & -XY & -X \end{bmatrix}, \qquad (10.27)$$

- The interaction matrix only depends on the feature vector and z_c

Interaction matrix of a set of points

- An interaction matrix can be built using a set of points

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

- With $\boldsymbol{z}_c = [\, z_{\ c,1} \ldots z_{c,n}]^T$

Interaction matrix of a line segment

- A line segment is the part of a line connecting the two points P_1 and P_2
- The line can be characterized by
  – Middle point coordinates $\bar{x},\ \bar{y}$
  – Length L
  – Angle α
- The feature vector can be defined by

$$\boldsymbol{s} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ L \\ \alpha \end{bmatrix} = \begin{bmatrix} (X_1 + X_2)/2 \\ (Y_1 + Y_2)/2 \\ \sqrt{\Delta X^2 + \Delta Y^2} \\ \tan^{-1}(\Delta Y / \Delta X) \end{bmatrix} = \boldsymbol{s}(\boldsymbol{s}_1, \boldsymbol{s}_2) \qquad (10.28)$$

with $\Delta X = X_2 - X_1$, $\Delta Y = Y_2 - Y_1$ and $\boldsymbol{s}_i = \begin{bmatrix} X_i & Y_i \end{bmatrix}^T$

Interaction matrix of a line segment

- Computing the time derivative 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 P_i
- This yields the interaction matrix

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

Interaction matrix of line segment

- With

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

Inverting the interaction matrix (10.3.3)

- The interaction matrix has generally a dimension of k x m
  - k is the number of features parameters
  - m is the dimension of the velocity vector (usually 6)
- If $\boldsymbol{L}_s$ has full rank (k=m)
  - The interaction matrix can be inverted

$$\boldsymbol{v}_{c,o}^{c} = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^{c})\boldsymbol{L}_s^{-1}\dot{\boldsymbol{s}}, \tag{10.29}$$

- If k > m we have an over-defined system and can use the pseudo-inverse

$$\boldsymbol{v}_{c,o}^{c} = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^{c})(\boldsymbol{L}_s^{T}\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^{T}\dot{\boldsymbol{s}}, \tag{10.30}$$

Inverting the interaction matrix

- If $k < m$ we have an infinite number of solutions (under-defined)
- This means that the camera can move without changing the features
- We must add more features in order to determine the relative motion between the camera and the object

10.6 The visual servoing problem

- Position-based visual servoing
  - Using pose estimation to find the object pose
  - Use operational space parameters
  - Absence of direct control of image features may cause the object to exit the camera view
  - Calibration error as disturbance in feedback path
- Image-based visual servoing
  - Using image features to control the robot
  - No pose estimation is required
  - Keeps the object within the camera field of view
  - Nonlinear mapping between image features parameters and operational space variables
  - Calibration error as disturbance in forward path

The visual servoing problem - position-based

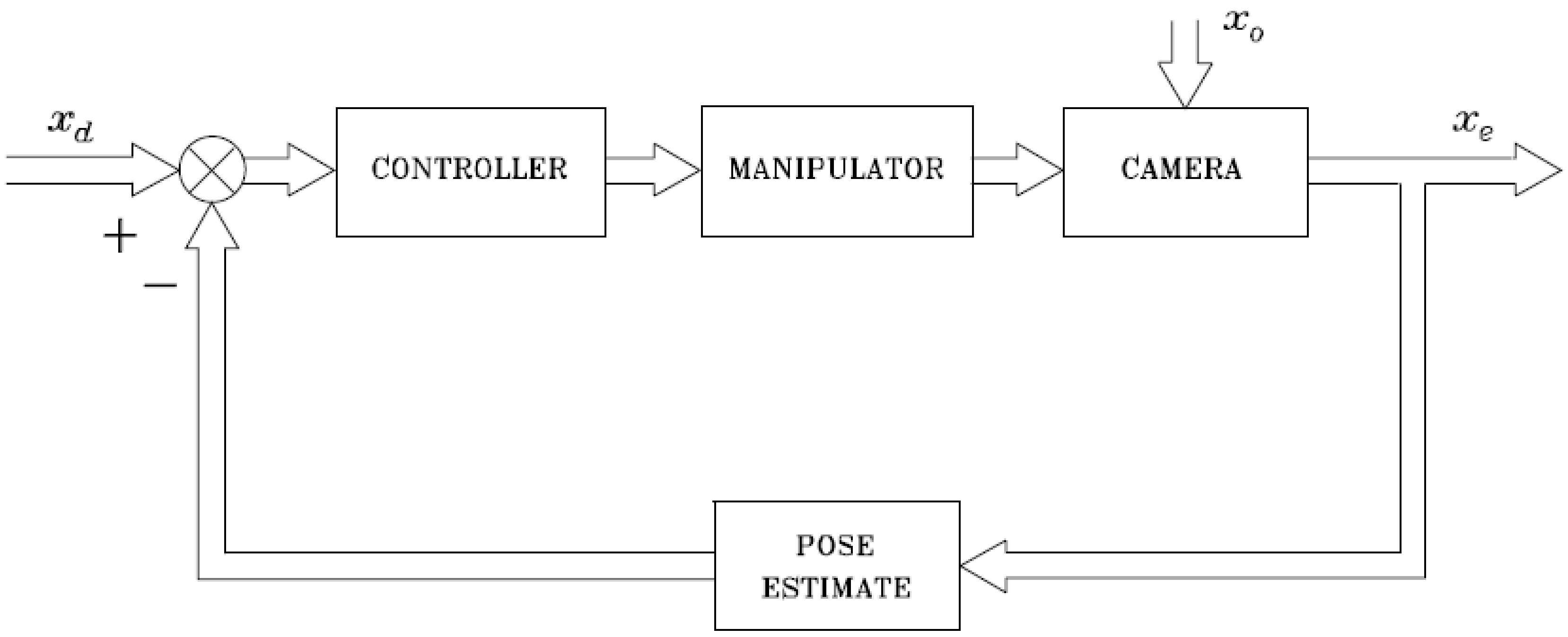


Fig. 10.12. General block scheme of position-based visual servoing

The visual servoing problem - image-based

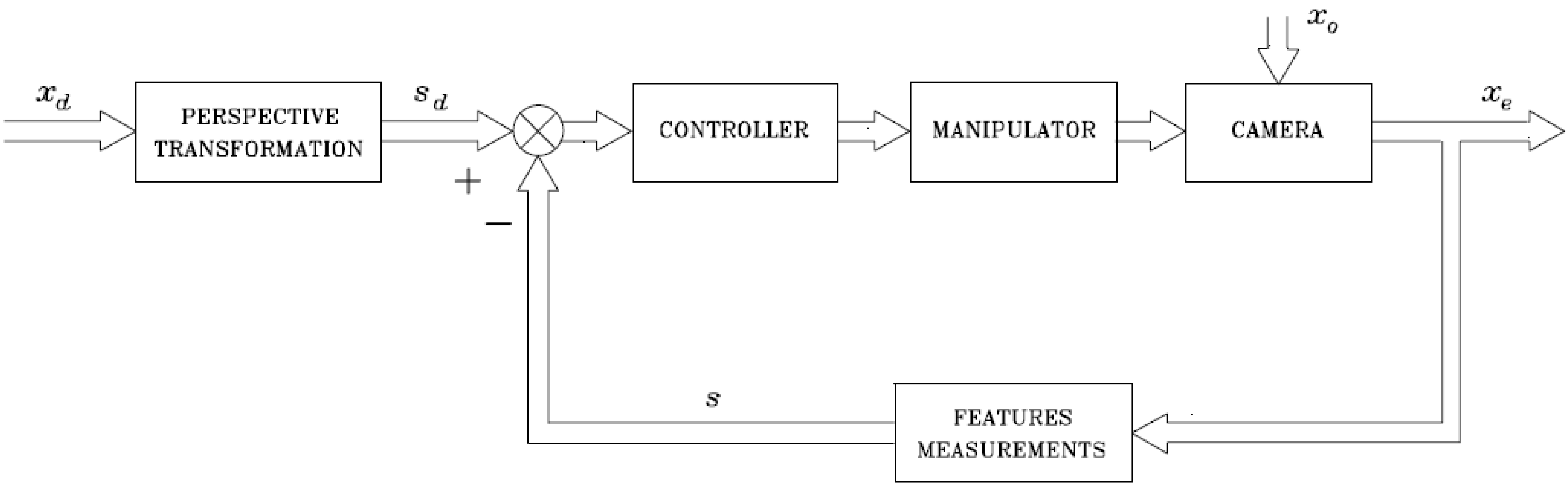


Fig. 10.13. General block scheme of image-based visual servoing

10.7 Position-based visual servoing

- Visual measurements estimate the transform $\boldsymbol{T}_o^c$
- Assuming a fixed object with respect to the base frame
- Specifies a desired pose of the object frame with regards to the camera frame $\boldsymbol{T}_o^d$:

- The displacement of the camera frame in the current pose with the respect to the desired pose

$$\boldsymbol{T}_c^d = \boldsymbol{T}_o^d(\boldsymbol{T}_o^c)^{-1} = \begin{bmatrix} \boldsymbol{R}_c^d & \boldsymbol{o}_{d,c}^d \\ \boldsymbol{0}^T & 1 \end{bmatrix}, \tag{10.68}$$

Position-based visual servoing

- Based on the matrix $\boldsymbol{T}_c^d$ the following error vector can be defined

$$\widetilde{\boldsymbol{x}} = -\begin{bmatrix} \boldsymbol{o}_{d,c}^d \\ \phi_{d,c} \end{bmatrix}, \qquad (10.69)$$

10.7.1 PD control with gravity compensation

- Now we will find the error change in time
- Computing the time derivative of (10.69) yields

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

$$\dot{\boldsymbol{\phi}}_{d,c} = \boldsymbol{T}^{-1}(\boldsymbol{\phi}_{d,c})\boldsymbol{\omega}_{d,c}^{d} = \boldsymbol{T}^{-1}(\boldsymbol{\phi}_{d,c})\boldsymbol{R}_d^T \boldsymbol{\omega}_c.$$

- Using $\dot{\boldsymbol{o}}_d^d = \mathbf{0}$ and $\boldsymbol{\omega}_d^d = \mathbf{0}$ since $\boldsymbol{o}_d$ and $\boldsymbol{R}_d$ are constant
- This yields

$$\dot{\tilde{\boldsymbol{x}}} = -\boldsymbol{T}_A^{-1}(\boldsymbol{\phi}_{d,c}) \begin{bmatrix} \boldsymbol{R}_d^T & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_d^T \end{bmatrix} \boldsymbol{v}_c. \qquad (10.71)$$

PD control with gravity compensation

- Mapping the error to joint space

$$\dot{\widetilde{\boldsymbol{x}}} = -\boldsymbol{J}_{A_d}(\boldsymbol{q}, \widetilde{\boldsymbol{x}})\dot{\boldsymbol{q}}, \tag{10.72}$$

- Where

$$\boldsymbol{J}_{A_d}(\boldsymbol{q}, \widetilde{\boldsymbol{x}}) = \boldsymbol{T}_A^{-1}(\phi_{d,c}) \begin{bmatrix} \boldsymbol{R}_d^T & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_d^T \end{bmatrix} \boldsymbol{J}(\boldsymbol{q}) \tag{10.73}$$

- This is similar to what was done in compliance and impedance control

PD control with gravity compensation

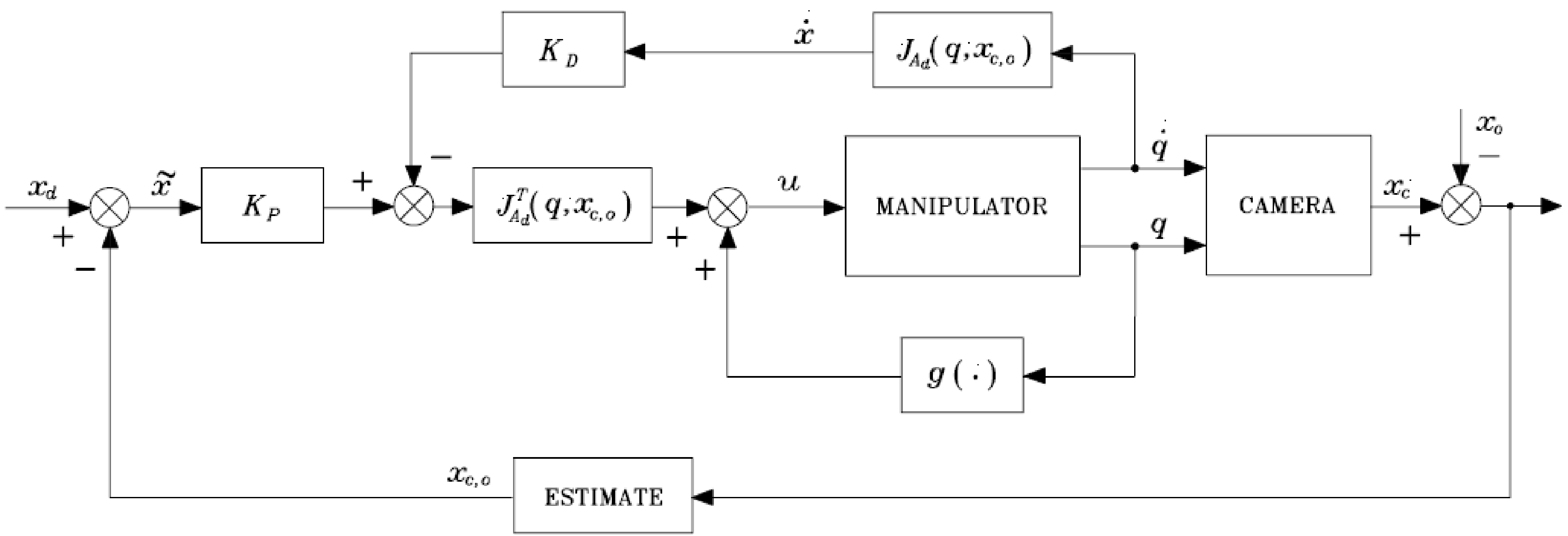
- Using the control law

$$u = g(q) + J_{A_d}^T(q, \widetilde{x})(K_P \widetilde{x} - K_D J_{A_d}(q, \widetilde{x}) \dot{q}), \qquad (10.74)$$

- Same control law as operational space motion control using a PD controller with gravity compensation
- Different definition of the error
- Asymptotic stability can be proven using

$$V(\dot{q}, \widetilde{x}) = \frac{1}{2} \dot{q}^T B(q) \dot{q} + \frac{1}{2} \widetilde{x}^T K_P \widetilde{x} > 0 \qquad \forall \dot{q}, \widetilde{x} \neq 0,$$

PD control with gravity compensation



- Sum block finding $\tilde{x}$ has conceptual meaning

10.7.2 Resolved velocity control

- Visual measurements have lower update rates than used in motion control of robot manipulators
- As a result the control in (10.74) must have low control gain values to preserve stability
- A solution is to have two levels of control
  - A high-gain motion controller in joint space or operational space
  - A visual servoing controller with lower update frequency

Resolved velocity control

- The high-gain motion controller is considered an ideal positioning device

$$\boldsymbol{q}(t) \approx \boldsymbol{q}_r(t), \qquad (10.75)$$

- Then the visual servoing control can be achieved by computing the trajectory $\boldsymbol{q}_r(t)$
- The equation

$$\dot{\widetilde{\boldsymbol{x}}} = -\boldsymbol{J}_{A_d}(\boldsymbol{q}, \widetilde{\boldsymbol{x}})\dot{\boldsymbol{q}}, \qquad (10.72)$$

- Suggests the following control law

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

Resolved velocity control

- Inserting into (10.72), and taking (10.75) into account, yields

$$\dot{\widetilde{\boldsymbol{x}}} + \boldsymbol{K}\widetilde{\boldsymbol{x}} = \boldsymbol{0}. \tag{10.77}$$

- Assuming a positive definite K ensures the error to go to zero asymptotically
- The scheme is termed resolved-velocity control as it is based on the computation of $\dot{\boldsymbol{q}}_r$ from the operational space error
- The trajectory $\boldsymbol{q}_r(t)$ is found by integration

Resolved velocity control

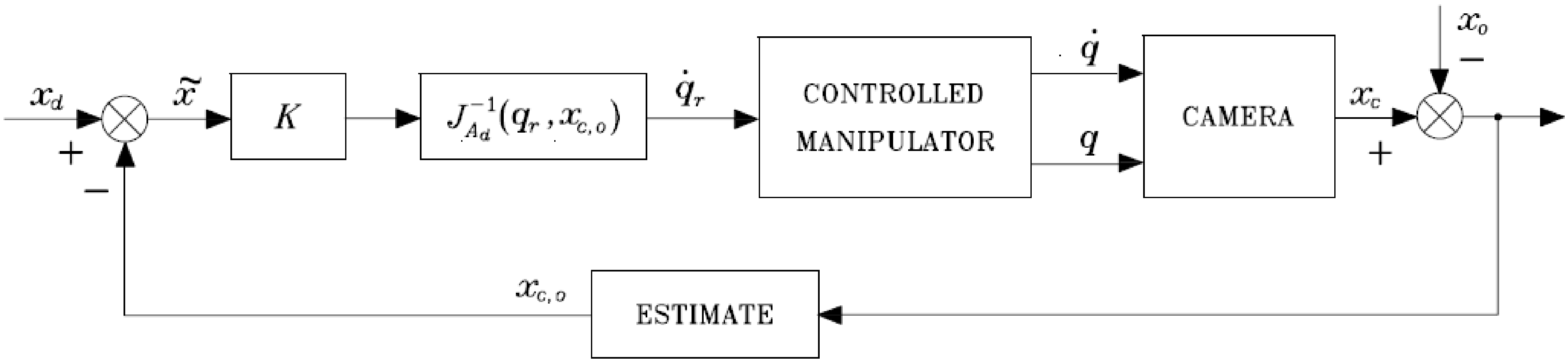


Fig. 10.15. Block scheme of resolved-velocity position-based visual servoing

10.8 Image-based visual servoing

- Assumptions
  - Object fixed with respect to base frame
  - Constant desired feature parameter $\boldsymbol{s}_d$
- Find a control law that get the error asymptotically to zero

$$\boldsymbol{e}_s = \boldsymbol{s}_d - \boldsymbol{s} \tag{10.79}$$

10.8.1 PD control with gravity compensation

- Consider the following Lyapunov function candidate

$$\boldsymbol{V}(\dot{\boldsymbol{q}}, \boldsymbol{e}_s) = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{B}(\boldsymbol{q})\boldsymbol{q} + \frac{1}{2}\boldsymbol{e}_s^T \boldsymbol{K}_{Ps}\boldsymbol{e}_s > 0 \qquad \forall \dot{\boldsymbol{q}}, \boldsymbol{e}_s \neq \boldsymbol{0}, \tag{10.80}$$

- Taking the time derivative yields

$$\dot{V} = -\dot{\boldsymbol{q}}^T \boldsymbol{F}\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T(\boldsymbol{u} - \boldsymbol{g}(\boldsymbol{q})) + \dot{\boldsymbol{e}}_s^T \boldsymbol{K}_{Ps}\boldsymbol{e}_s. \tag{10.81}$$

PD control with gravity compensation

- Since $\dot{s}_d = \mathbf{0}$ and the object is fixed to the base frame

$$\dot{\boldsymbol{e}}_s = -\dot{\boldsymbol{s}} = -\boldsymbol{J}_L(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q})\dot{\boldsymbol{q}}, \tag{10.82}$$

- Where

$$\boldsymbol{J}_L(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q}) = \boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c) \begin{bmatrix} \boldsymbol{R}_c^T & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_c^T \end{bmatrix} \boldsymbol{J}(\boldsymbol{q}), \tag{10.83}$$

- Choosing

$$\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{J}_L^T(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q}) \left(\boldsymbol{K}_{Ps}\boldsymbol{e}_s - \boldsymbol{K}_{Ds}\boldsymbol{J}_L(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q})\dot{\boldsymbol{q}} \right), \tag{10.84}$$

- Yields

$$\dot{V} = -\dot{q}^T \boldsymbol{F}\dot{\boldsymbol{q}} - \dot{q}^T \boldsymbol{J}_L^T \boldsymbol{K}_{Ds}\boldsymbol{J}_L\dot{\boldsymbol{q}}. \tag{10.85}$$

PD control with gravity compensation

$$\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{J}_L^T(\boldsymbol{s}, z_c, \boldsymbol{q}) \left(\boldsymbol{K}_{Ps} \boldsymbol{e}_s - \boldsymbol{K}_{Ds} \boldsymbol{J}_L(\boldsymbol{s}, z_c, \boldsymbol{q}) \dot{\boldsymbol{q}} \right), \tag{10.84}$$

- The control law consist of
  - Nonlinear gravity compensation action
  - Linear PD action in the image space
- The last term corresponds to a derivative action in image space, used to increase damping
- $-\boldsymbol{K}_{Ds}\dot{\boldsymbol{s}}$ is not measured, therefore using $-\boldsymbol{K}_{Ds}\boldsymbol{J}_L(\boldsymbol{s}, z_c, \boldsymbol{q})\dot{\boldsymbol{q}}$
- The system reaches the equilibrium state

$$\boldsymbol{J}_L^T(\boldsymbol{s}, z_c, \boldsymbol{q}) \boldsymbol{K}_{Ps} \boldsymbol{e}_s = \boldsymbol{0}. \tag{10.86}$$

- If the geometric Jacobian and the interaction matrix is full rank the error goes to zero

PD control with gravity compensation

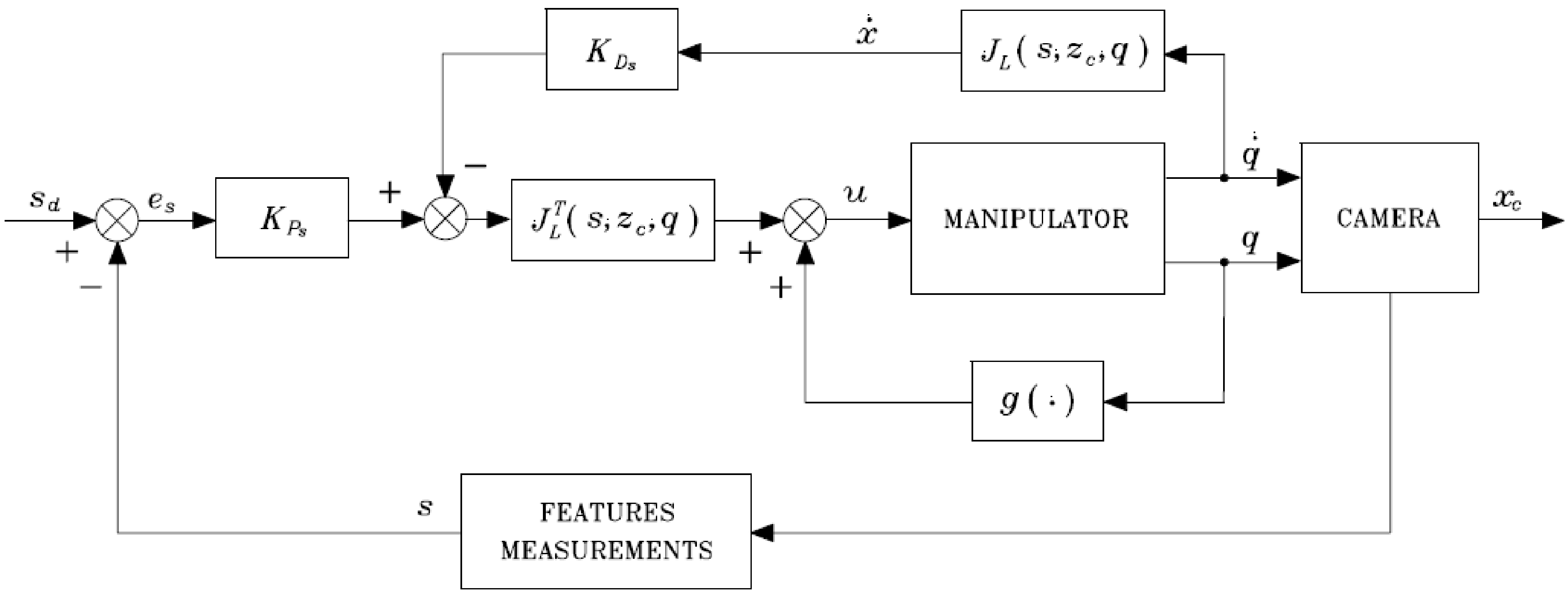


Fig. 10.16. Block scheme of image-based visual servoing of PD type with gravity compensation

PD control with gravity compensation

- The control law requires the measurement of s and the computation of z_c
- The dependency of z_c is undesired
- Strategies
  - z_c may in some cases be known with reasonably accuracy
  - Estimated or constant values may be used, for instance the value in the initial or the desired pose
- Find an estimate $\widehat{L}_s$ of the interaction matrix

10.8.2 Resolved-velocity control

- Resolved-velocity control can be extended to image-based visual servoing
- Based on

$$\dot{\boldsymbol{e}}_s = -\dot{\boldsymbol{s}} = -\boldsymbol{J}_L(\boldsymbol{s}, z_c, \boldsymbol{q})\dot{\boldsymbol{q}}, \tag{10.82}$$

- We select

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}_L^{-1}(\boldsymbol{s}, z_c, \boldsymbol{q}_r)\boldsymbol{K}_s\boldsymbol{e}_s, \tag{10.87}$$

- This yields the error dynamics

$$\dot{\boldsymbol{e}}_s + \boldsymbol{K}_s\boldsymbol{e}_s = \boldsymbol{0}. \tag{10.88}$$

- Which shows that the system is asymptotically stable

Resolved-velocity control

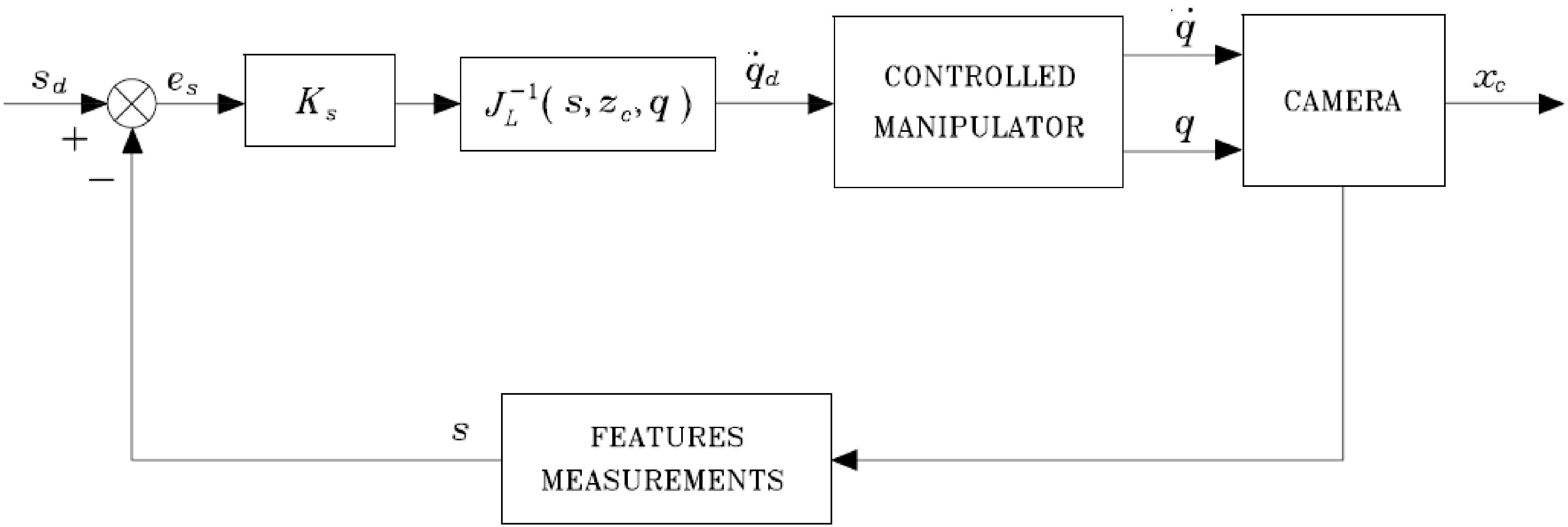


Fig. 10.17. Block scheme of resolved-velocity image-based visual servoing

Resolved-velocity control

- The control law depends on computing the inverse of $\boldsymbol{J}_L$
- This might cause problems with singularities
- Can partition the control law into two steps

$$\boldsymbol{v}_r^c = \boldsymbol{L}_s^{-1}(\boldsymbol{s}, z_c)\boldsymbol{K}_s\boldsymbol{e}_s. \tag{10.89}$$

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}^{-1}(\boldsymbol{q})\begin{bmatrix} \boldsymbol{R}_c & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_c \end{bmatrix}\boldsymbol{v}_r^c. \tag{10.90}$$

- Selecting more features and using the pseudo-inverse can overcome singularities in $\boldsymbol{L}_s$

$$\boldsymbol{v}_r^c = (\boldsymbol{L}_s^T\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^T\boldsymbol{K}_s\boldsymbol{e}_s \tag{10.91}$$

Resolved-velocity control

- Using the Lyapunov candidate function

$$V(\boldsymbol{e}_s) = \frac{1}{2}\boldsymbol{e}_s^T \boldsymbol{K}_s \boldsymbol{e}_s > 0 \qquad \forall \boldsymbol{e}_s \neq \mathbf{0}.$$

- Yields

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

- Which is negative semi-definite, since $\mathcal{N}(\boldsymbol{L}_s^T) \neq \emptyset$, i.e. it does not have full rank
- The controller is stable, but not asymptotically stable
- Bounded error, but not necessarily $\boldsymbol{e}_s = 0$

Summary

- Interaction matrix
  - Point
  - Line
- Position-based visual servoing
  - PD control with gravity compensation
  - Resolved-velocity control
- Image-based visual servoing
  - PD control with gravity compensation
  - Resolved-velocity control