GdR Robotics Winter School: Robotics Principia

Sensor-based Control

François Chaumette

Inria

Univ Rennes, CNRS, IRISA, Rennes

Preliminary note

- In this talk, focus on vision-based control / visual servoing
- But all concepts are valid for any exteroceptive sensor providing measurements related to the relative pose between the robot and its environment (proximity sensors, laser, depth map, RGB-D camera, 2D US probe, omnidirectional camera, audio sensor, …)

Overview

1. Introduction
2. Modeling
3. Task specification
4. Control

References

[1] S. Hutchinson, G. Hager, P. Corke: A tutorial on visual servo control, *IEEE. Trans. on Robotics and Automation*, 12(5):651-670, October 1996.

[2] F. Chaumette, S. Hutchinson: Visual servoing and visual tracking, In *The Handbook of Robotics*, B. Siciliano, O. Khatib (eds.), Chap 24, pp. 563-583, Springer, 2008.

[3] P. Corke: Robotics, Vision and Control, Springer, 2011.

[4] F. Chaumette: Visual servoing. In *Robot Manipulators: Modeling, Performance Analysis and Control*, E. Dombre, W. Khalil (eds.), Chap. 6, pp. 279-336, ISTE, 2007.

Software

- ViSP: *http://visp.inria.fr*
  C++ open source library for real time visual tracking and visual servoing

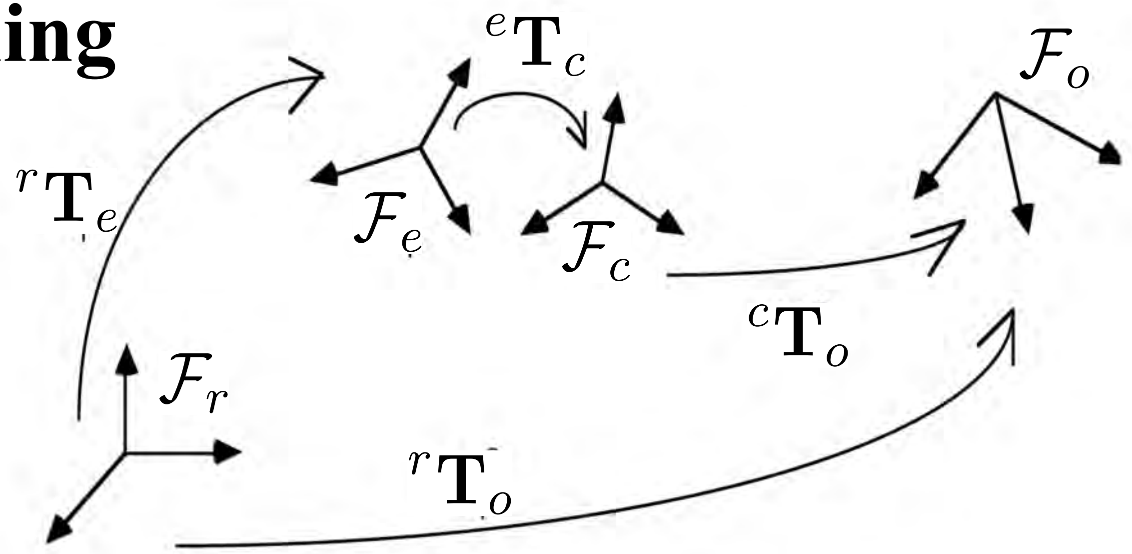
Open loop robot positioning

For a given desired pose ${}^{c^*}\mathbf{T}_o$

- Compute only once the displacement to be done

$$ {}^{c}\mathbf{T}_{c^*} = {}^{c}\mathbf{T}_o \, {}^{c^*}\mathbf{T}_o^{-1} $$

or $$ {}^{e}\mathbf{T}_{e^*} = {}^{e}\mathbf{T}_c \, {}^{c}\mathbf{T}_o \, {}^{c^*}\mathbf{T}_o^{-1} \, {}^{e}\mathbf{T}_c^{-1} $$



Advantages:

- Only one image to be processed and one very fast displacement to be achieved if the full system is perfectly calibrated

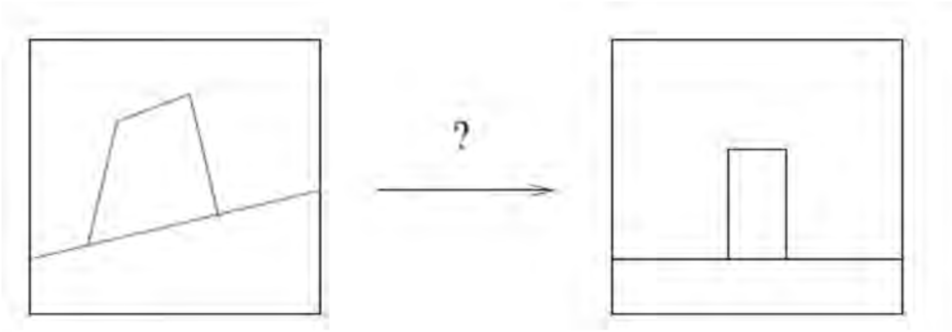
Drawback:

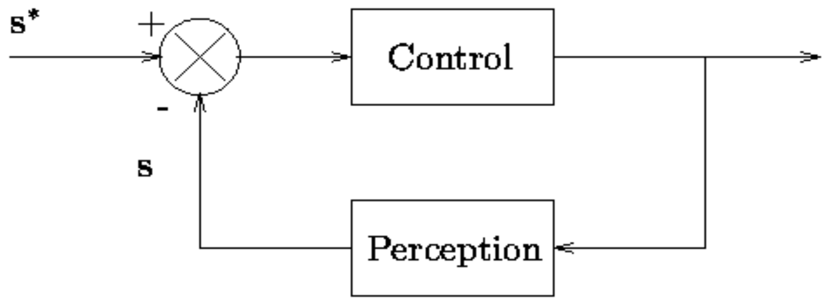
- Not robust to modeling and calibration errors

Better approach: closed-loop sensor-based control: visual servoing

What is visual servoing?

Vision-based closed loop control of a dynamic system





Advantages:

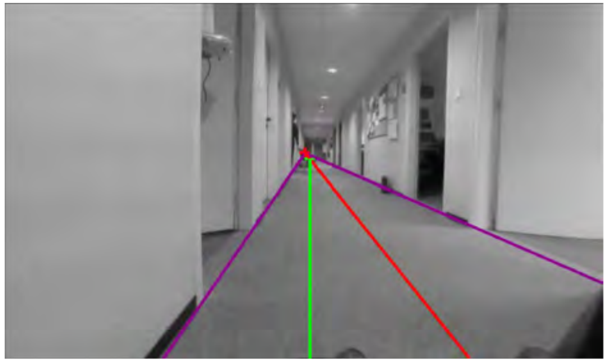
- Positioning accuracy
- Robustness with respect to modeling/calibration errors
- Reactive to changes (target tracking)
- Alternative to SLAM: achieve a task with the minimal information required

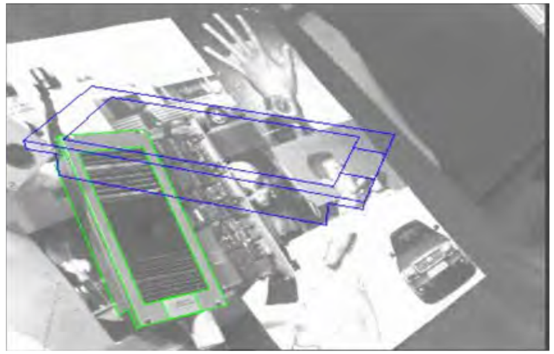
Drawback:

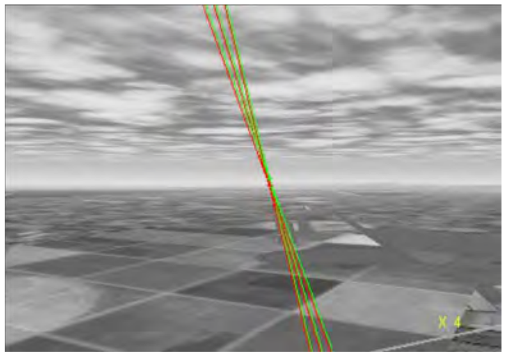
- Need many images to be processed

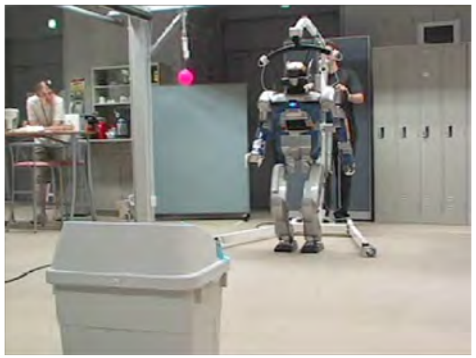
A wide spectrum of applications

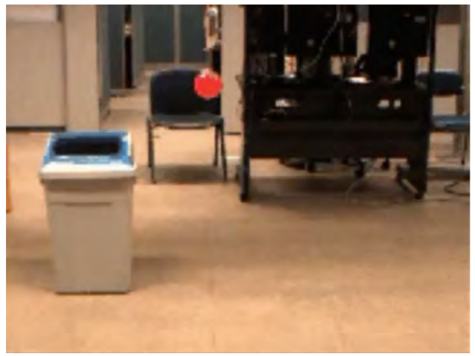
Just need a camera and a robot



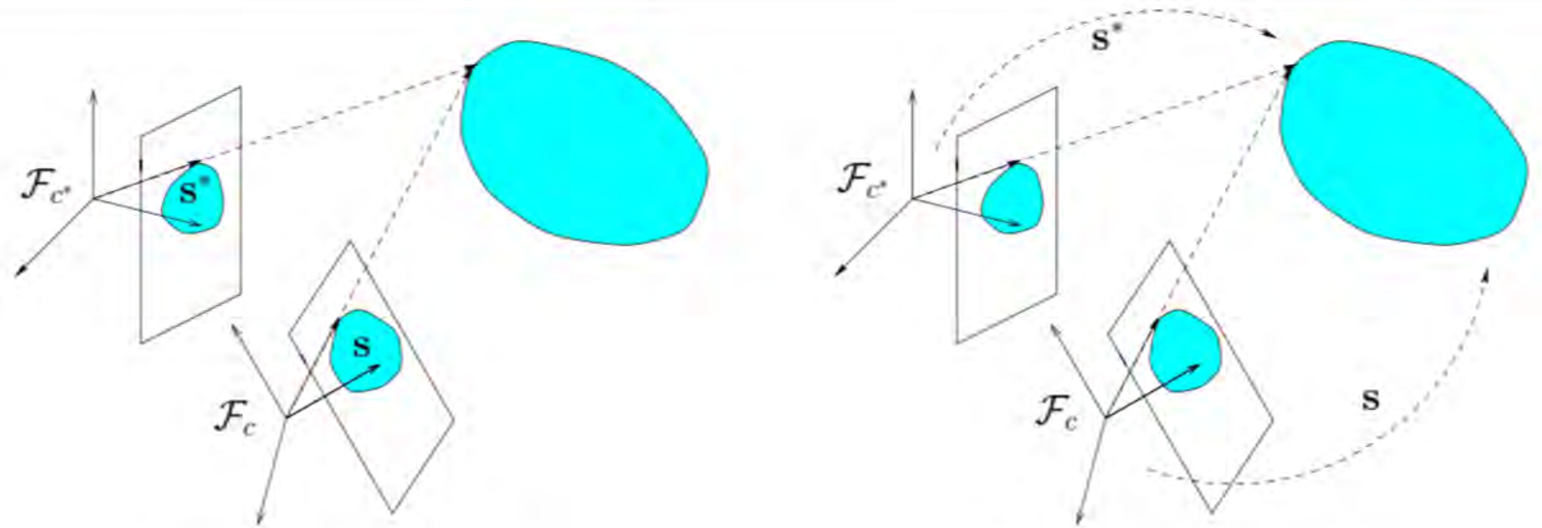








What are the best features? IBVS / PBVS



Modeling: same principle in all cases (but not same properties)

Visual features: $\mathbf{s} = \mathbf{s}(\mathbf{p}(t)) \quad \Rightarrow \quad \dot{\mathbf{s}} = \mathbf{L_s}\,\mathbf{v}$ where:

- $\mathbf{L_s}$ = interaction matrix (similar to a Jacobian)
- $\mathbf{v} = (\boldsymbol{v}, \boldsymbol{\omega}) \in se_3$ = instantaneous camera velocity
  with 3 translational and 3 rotational components

(Basic) control law / stability analysis

If we would like $\dot{\mathbf{s}} = -\lambda(\mathbf{s} - \mathbf{s}^*)$ (exponential decoupled decrease)

From $\dot{\mathbf{s}} = \mathbf{L_s}\,\mathbf{v}$, we get

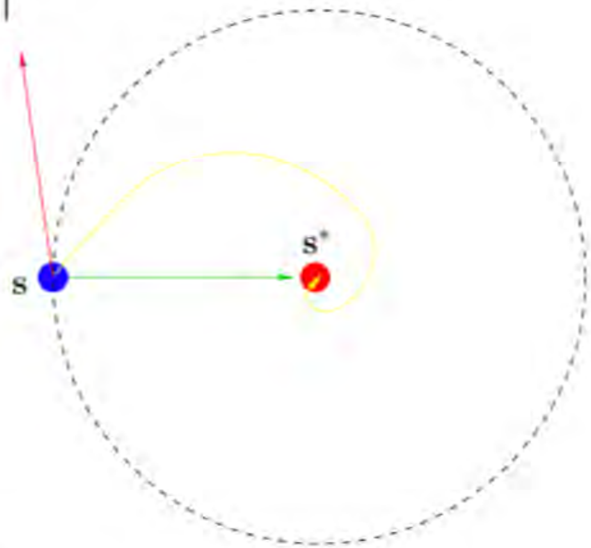
$$\mathbf{v} = -\lambda\,\widehat{\mathbf{L_s}}^{+}(\mathbf{s} - \mathbf{s}^*) \text{ with } \widehat{\mathbf{L_s}}_{(\mathbf{s},\mathbf{p},\mathbf{a})}$$

Closed-loop system: $\dot{\mathbf{s}} = \mathbf{L_s}\,\mathbf{v} = -\lambda\,\mathbf{L_s}\widehat{\mathbf{L_s}}^{+}(\mathbf{s} - \mathbf{s}^*)$

Lyapunov stability analysis: $\mathcal{L} = \frac{1}{2}\|\mathbf{s} - \mathbf{s}^*\|^2$

$$\dot{\mathcal{L}} = -\lambda(\mathbf{s} - \mathbf{s}^*)^{\top}\mathbf{L_s}\widehat{\mathbf{L_s}}^{+}(\mathbf{s} - \mathbf{s}^*)$$

- if $\mathbf{L_s}\widehat{\mathbf{L_s}}^{+} = \mathbf{I}$, perfect behavior
- if $\mathbf{L_s}\widehat{\mathbf{L_s}}^{+} > 0$, $\|\mathbf{s} - \mathbf{s}^*\|$ decreases
- if $\mathbf{L_s}\widehat{\mathbf{L_s}}^{+} < 0$, $\|\mathbf{s} - \mathbf{s}^*\|$ increases...



Sufficient condition for stability: $\mathbf{L_s}\widehat{\mathbf{L_s}}^{+} > 0$

Stability analysis with $\mathbf{v} = -\lambda\, \widehat{\mathbf{L_s}}^+ (\mathbf{s} - \mathbf{s}^*)$**:** $\mathbf{L_s}\widehat{\mathbf{L_s}}^+ > 0$ **?**

- if $k = 6$ (usual case in PBVS), dim $\mathbf{L_s} = 6 \times 6$
  $\mathbf{L_s}\widehat{\mathbf{L_s}}^+ = \mathbf{L_s}\widehat{\mathbf{L_s}}^{-1} > 0$ allows the system to be GAS

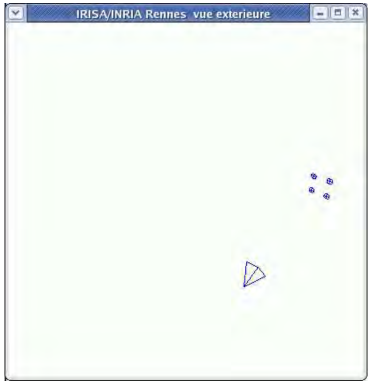
- if $k > 6$ (usual case in IBVS), dim $\mathbf{L_s} = k \times 6$
  $\mathbf{L_s}\widehat{\mathbf{L_s}}^+ > 0$ impossible (rank $\mathbf{L_s} = 6 < k$ at max)

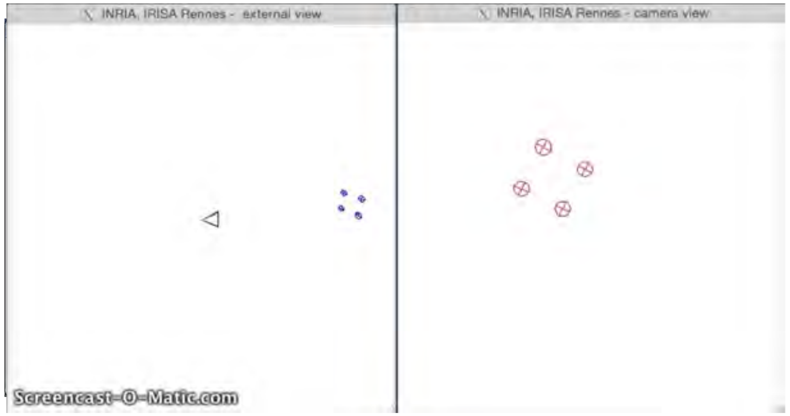
By looking at $\mathbf{e} = \widehat{\mathbf{L_s}}^+(\mathbf{s} - \mathbf{s}^*)$ and $\mathcal{L} = \frac{1}{2}\|\mathbf{e}\|^2$

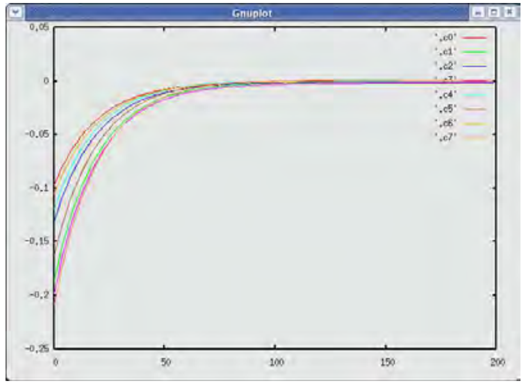
$$\dot{\mathcal{L}} = \mathbf{e}^\top \dot{\mathbf{e}} \approx \mathbf{e}^\top \widehat{\mathbf{L_s}}^+ \mathbf{L_s} \mathbf{v} = -\lambda \mathbf{e}^\top \widehat{\mathbf{L_s}}^+ \mathbf{L_s} \widehat{\mathbf{L_s}}^+ (\mathbf{s} - \mathbf{s}^*) = -\lambda \mathbf{e}^\top \widehat{\mathbf{L_s}}^+ \mathbf{L_s}\, \mathbf{e}$$

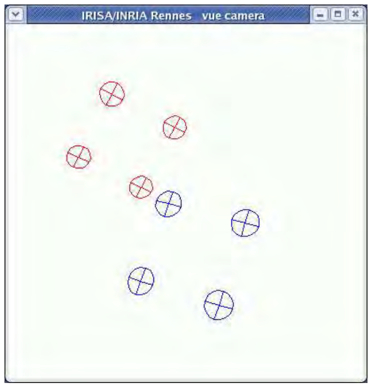
$\widehat{\mathbf{L_s}}^+ \mathbf{L_s} > 0$ allows the system to be LAS (because of $\approx$)

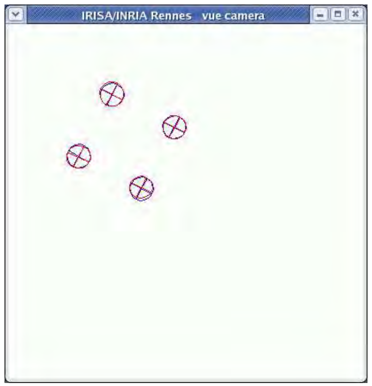
Example 1: reaching a local minimum using $\widehat{\mathbf{L_s}}^{+} = \mathbf{L_s}^{+}$

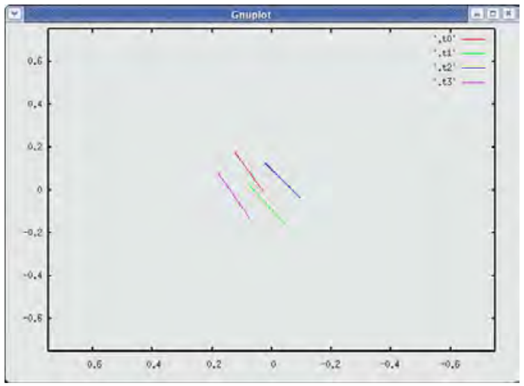


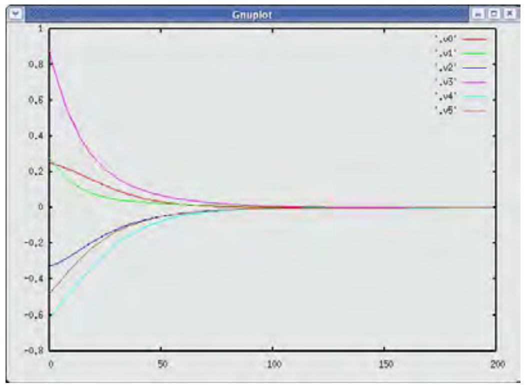




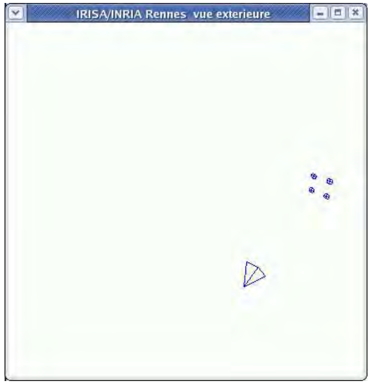


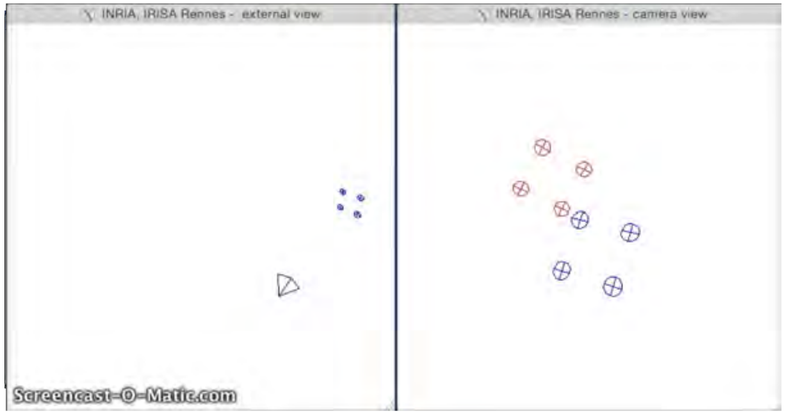


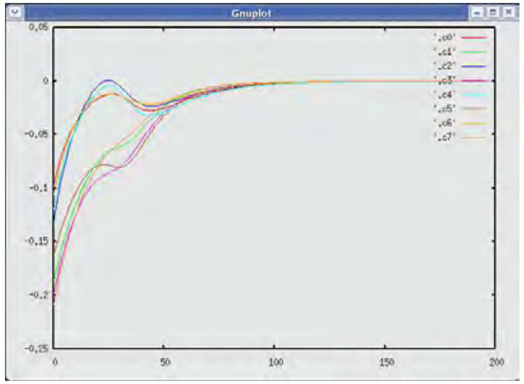


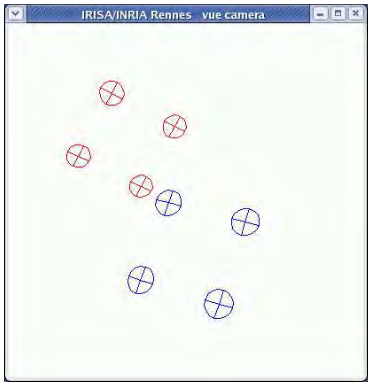


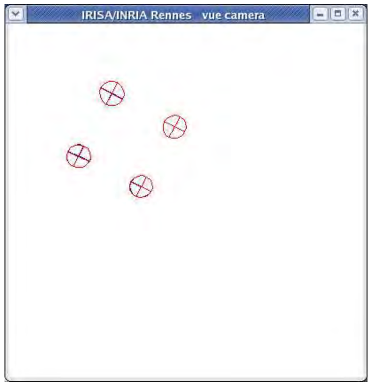
Example 2: reaching the global minimum using $\widehat{\mathbf{L_s}}^{+} = \mathbf{L_s}^{+}{}_{|\mathbf{s}=\mathbf{s}^*}$

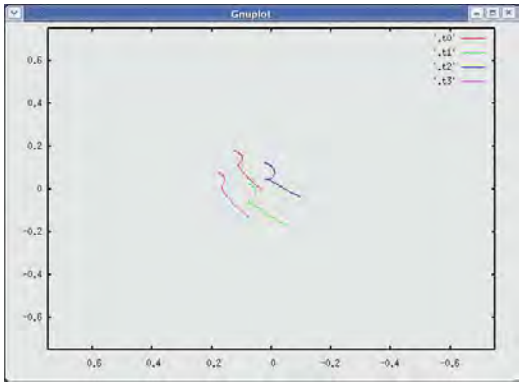


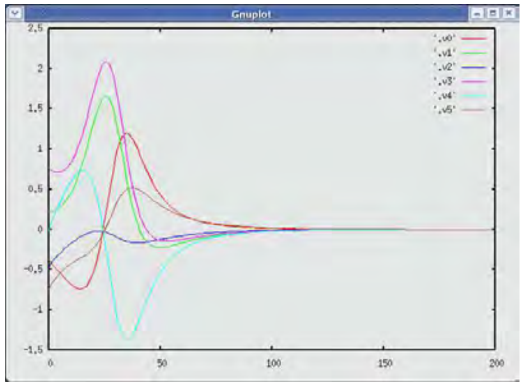








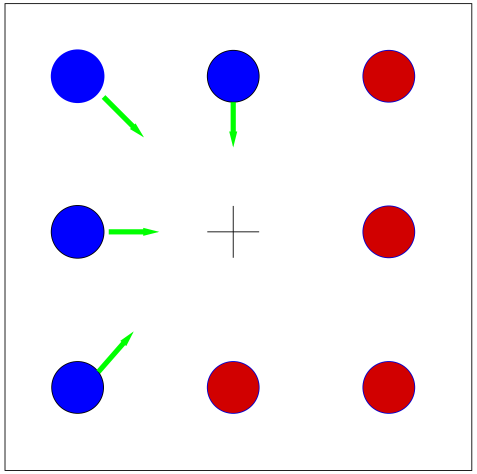




Example 3: reaching a singularity of $\mathbf{L_s}$

Example : rotation of 180° around the optical axis
s composed of image points

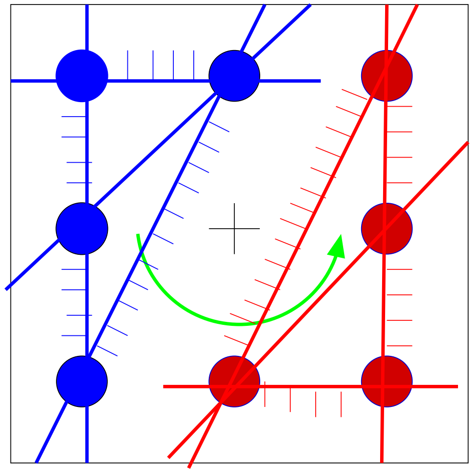
Cartesian coordinates



Using $\mathbf{L_s^+}$

At singularity, rank $\mathbf{L_s}$ = 2

Cylindrical coordinates

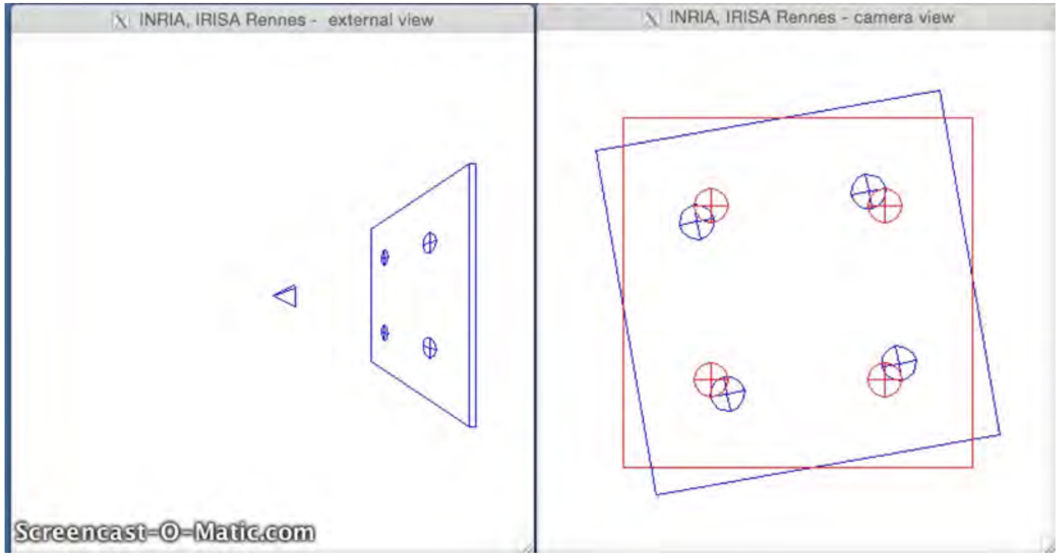


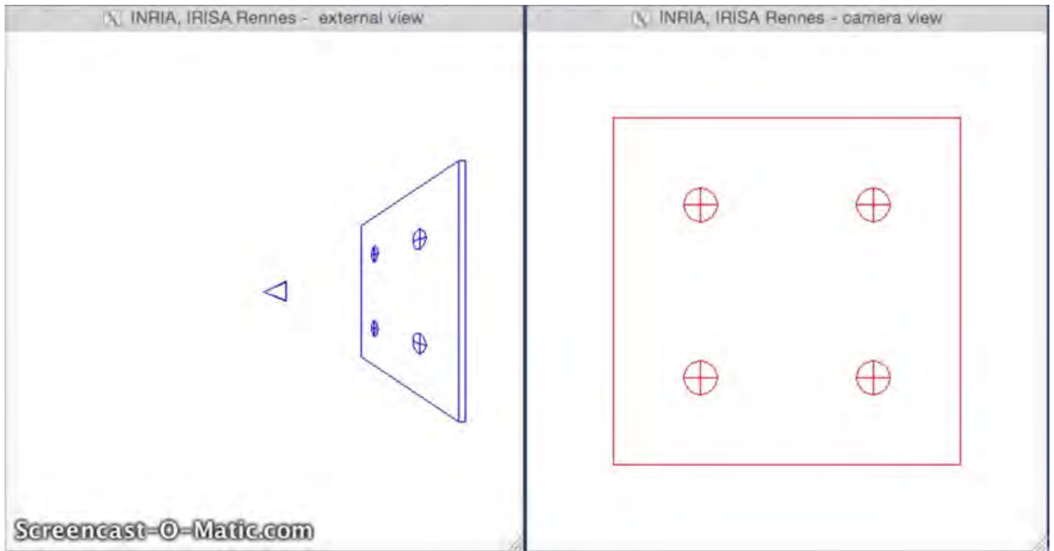
Using $\mathbf{L_s^+}$ or $\mathbf{L_s^+}_{|\mathbf{s}=\mathbf{s}^*}$

Perfect behavior

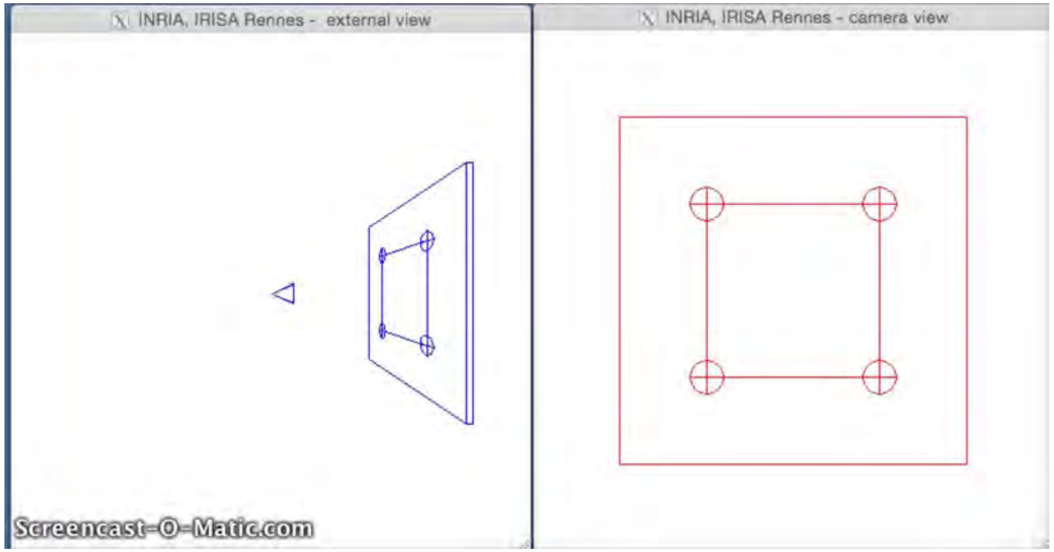
What are the best features?

Bad choice





Perfect choice (for this configuration)



Modeling issues

1. Basics
2. 3D visual features
3. 2D visual features
4. Numerical methods
5. Photometric features

Kinematic screw (instantaneous velocity)

$\mathbf{v} = (\boldsymbol{\upsilon}, \boldsymbol{\omega})$: kinematic screw between the camera and the scene expressed at C in $\mathcal{F}_c$

$\boldsymbol{\omega}$: rotational velocity : $[\boldsymbol{\omega}]_\times = {}^o\mathbf{R}_c^\top\, {}^o\dot{\mathbf{R}}_c = -{}^o\dot{\mathbf{R}}_c^\top\, {}^o\mathbf{R}_c$

$\boldsymbol{\upsilon}$: translational velocity at C : $\boldsymbol{\upsilon}(\mathrm{O}) = -\boldsymbol{\upsilon}(\mathrm{C}) - \boldsymbol{\omega} \times \mathbf{CO}$

To express $\mathbf{v}$ at $\mathbf{O}$ in $\mathcal{F}_o$: ${}^o\mathbf{v} = {}^o\mathbf{V}_c\, \mathbf{v}$ with ${}^o\mathbf{V}_c = \begin{bmatrix} {}^o\mathbf{R}_c & [{}^o\mathbf{t}_c]_\times {}^o\mathbf{R}_c \\ \mathbf{0}_3 & {}^o\mathbf{R}_c \end{bmatrix}$

We can decompose $\mathbf{v}$ as $\mathbf{v} = \mathbf{v}_c - \mathbf{v}_o$

where $\mathbf{v}_c$: camera kinematic screw, expressed at C in $\mathcal{F}_c$

$\mathbf{v}_o$: object kinematic screw, expressed at C in $\mathcal{F}_c$

The interaction matrix

A set $\mathbf{s}$ of k visual features is given by a function from SE_3 to $\mathbb{R}^k$:

$$\mathbf{s} = \mathbf{s}(\mathbf{p}(t))$$

where $\mathbf{p}(t)$ is the pose between the camera and the scene.
We get

$$\dot{\mathbf{s}} = \frac{\partial \mathbf{s}}{\partial \mathbf{p}} \dot{\mathbf{p}} = \mathbf{L_s}\, \mathbf{v}$$

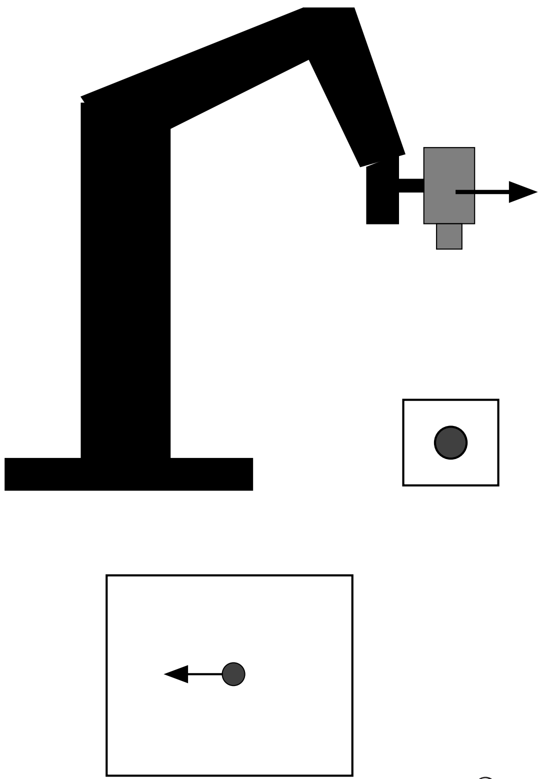
where $\mathbf{L_s}$ is the **interaction matrix** related to $\mathbf{s}$
(Jacobian $\frac{\partial \mathbf{s}}{\partial \mathbf{p}} \approx \mathbf{L_s}$ since $\dot{\mathbf{p}} = \mathbf{L_p}\, \mathbf{v}$)

Using $\mathbf{v}_c$ and $\mathbf{v}_o$, we obtain :

$$\dot{\mathbf{s}} = \mathbf{L_s}\, (\mathbf{v}_c - \mathbf{v}_o)$$

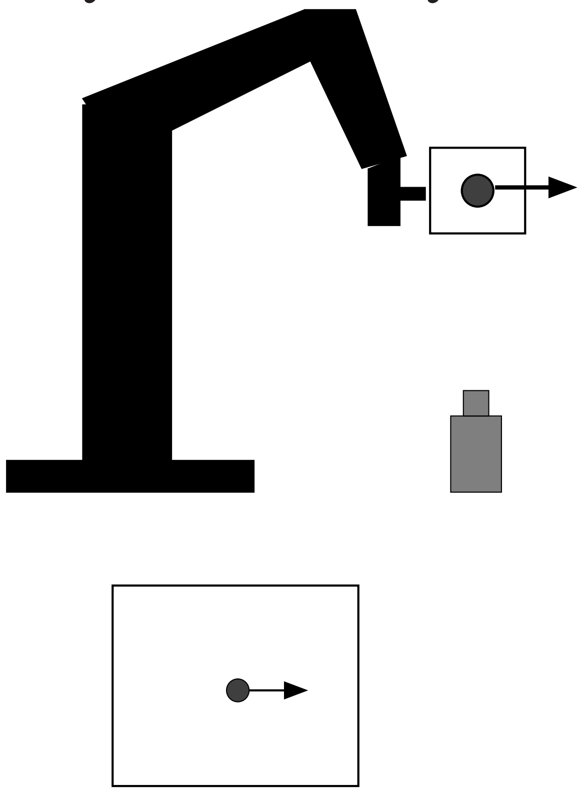
The feature Jacobian $\mathbf{J_s}$

Eye-in-Hand system



$$\dot{\mathbf{s}} = \mathbf{L_s}\,{}^c\mathbf{V}_n\,{}^n\mathbf{J}_n(\mathbf{q})\,\dot{\mathbf{q}} + \frac{\partial \mathbf{s}}{\partial t}$$
$$= \mathbf{J_s}\,\dot{\mathbf{q}} + \frac{\partial \mathbf{s}}{\partial t}$$

Eye-to-Hand system



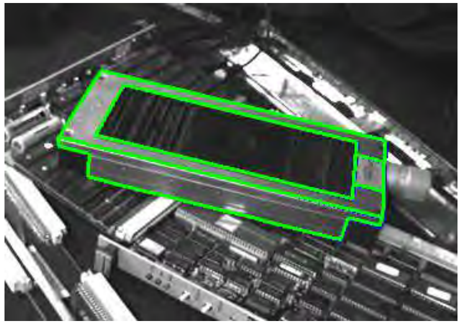
$$\dot{\mathbf{s}} = -\mathbf{L_s}\,{}^c\mathbf{V}_n\,{}^n\mathbf{J}_n(\mathbf{q})\,\dot{\mathbf{q}} + \frac{\partial \mathbf{s}}{\partial t}$$
$$= -\mathbf{L_s}\,{}^c\mathbf{V}_\emptyset\,{}^\emptyset\mathbf{V}_n\,{}^n\mathbf{J}_n(\mathbf{q})\,\dot{\mathbf{q}} + \frac{\partial \mathbf{s}}{\partial t}$$

Modeling issues

1. Basics
2. 3D visual features
3. 2D visual features
4. Numerical methods
5. Photometric features

3D visual features with one camera

Based on pose estimation $\hat{\mathbf{p}}(t)$ from $\mathcal{F}_c$ to $\mathcal{F}_o$ using



- an image of the object: $\mathbf{x}(t)$
- the knowledge of the object 3D model: $\mathbf{X}$
- an estimation of the camera intrinsic parameters: x_c, y_c, f_x, f_y

$$\hat{\mathbf{p}}(t) = \hat{\mathbf{p}}(\mathbf{x}(t), \mathbf{X}, x_c, y_c, f_x, f_y)$$

Pose estimation problem $\sim$ camera calibration problem

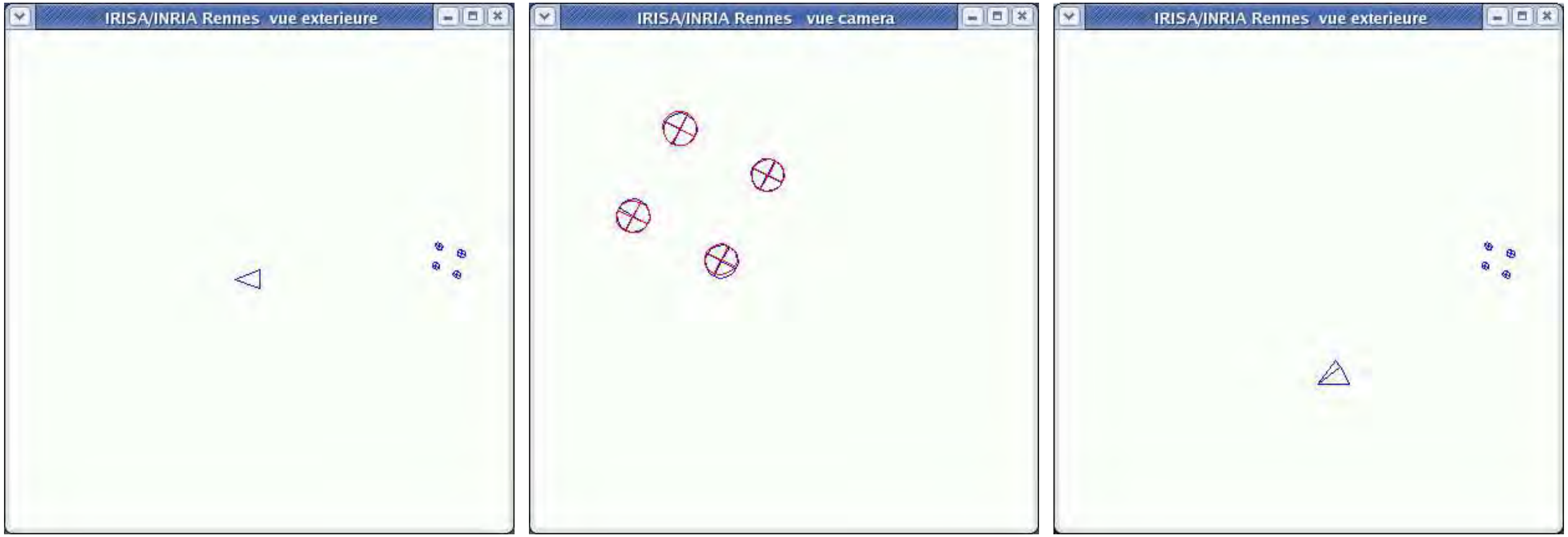
(intrinsic camera parameters already known)

3D visual features with one camera

Estimated pose $\hat{\mathbf{p}}(t) = \hat{\mathbf{p}}(\mathbf{x}(t), \mathbf{X}, x_c, y_c, f_x, f_y)$

$$\Rightarrow \quad \dot{\hat{\mathbf{p}}}(t) = \frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{x}} \dot{\mathbf{x}} = \frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{x}} \mathbf{L_x} \mathbf{v} \quad \Rightarrow \quad \mathbf{L}_{\hat{\mathbf{p}}} = \frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{x}} \mathbf{L_x}$$

where $\mathbf{L_x}$ is known but $\frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{x}}$ is unknown (and sometimes unstable)



3D visual features

Under the strong hypothesis that 3D estimation is perfect:

$$\frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{p}} = \mathbf{I}_6 \Rightarrow \dot{\hat{\mathbf{p}}} = \dot{\mathbf{p}} = \mathbf{L}_{\mathbf{p}}\mathbf{v}$$

- parameters $\theta\mathbf{u}$ that represent rotation ${}^{c^*}\mathbf{R}_c$

$$\mathbf{L}_{\theta\mathbf{u}} = \left[\, \mathbf{0}_3 \;\; \mathbf{L}_{\boldsymbol{\omega}} \,\right] \text{ where } \mathbf{L}_{\boldsymbol{\omega}} = \mathbf{I}_3 + \frac{\theta}{2}\,[\mathbf{u}]_\times + (1 - \frac{\mathrm{sinc}\theta}{\mathrm{sinc}^2\frac{\theta}{2}})\,[\mathbf{u}]^2_\times$$

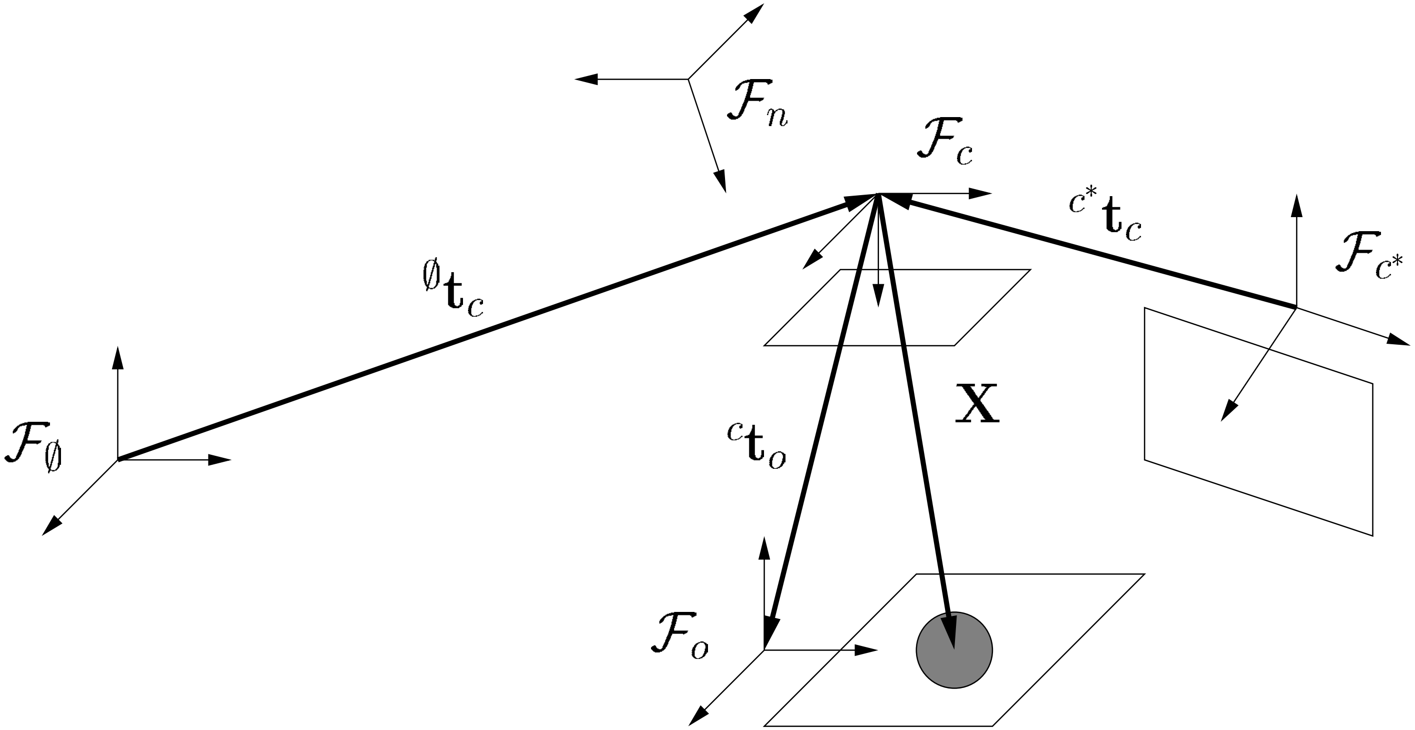
$$\mathbf{L}_{\boldsymbol{\omega}} \text{ such that } \mathbf{L}_{\boldsymbol{\omega}}\theta\mathbf{u} = \mathbf{L}_{\boldsymbol{\omega}}^{-1}\,\theta\mathbf{u} = \theta\mathbf{u}$$

- coordinates of a 3D point $\mathbf{X}$:

$$\dot{\mathbf{X}} = \boldsymbol{v}(\mathbf{X}) = -\boldsymbol{v}(\mathbf{C}) - \boldsymbol{\omega} \times \mathbf{CX} = -\boldsymbol{v} + \mathbf{CX} \times \boldsymbol{\omega} = -\boldsymbol{v} + [\mathbf{X}]_\times \boldsymbol{\omega}$$

$$\Rightarrow \mathbf{L}_{\mathbf{X}} = \left[\, -\mathbf{I}_3 \;\; [\mathbf{X}]_\times \,\right] = \begin{bmatrix} -1 & 0 & 0 & 0 & -Z & Y \\ 0 & -1 & 0 & Z & 0 & -X \\ 0 & 0 & -1 & -Y & X & 0 \end{bmatrix}$$

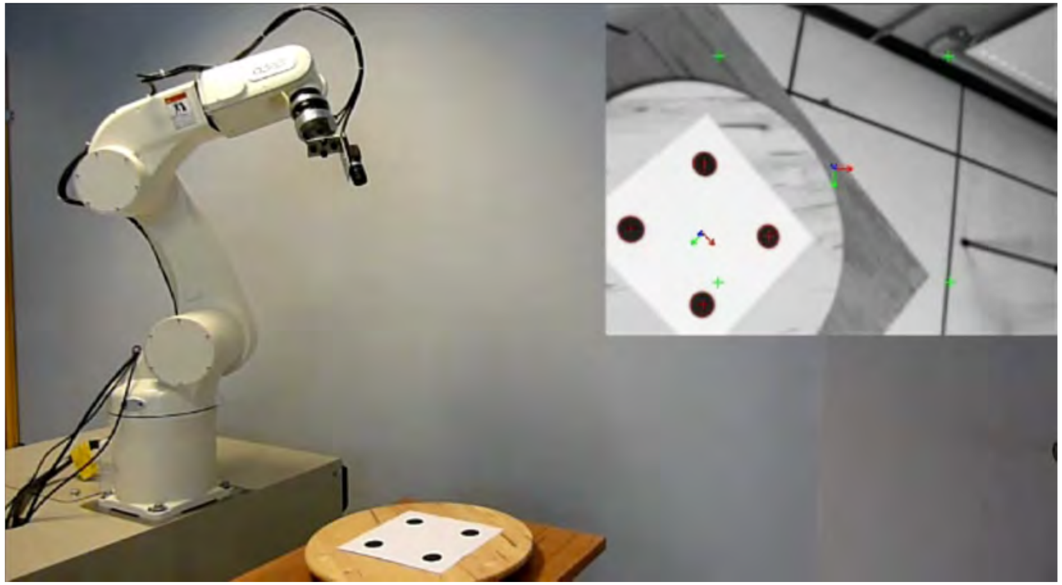
3D visual features for an eye-in-hand system



$$\mathbf{L}_{{}^c\mathbf{t}_o} = [\ -\mathbf{I}_3 \quad [{}^{\mathbf{c}}\mathbf{t}_{\mathbf{o}}]_\times\ ] \qquad \mathbf{L}_{{}^o\mathbf{t}_c} = [\ {}^{o}\mathbf{R}_c \quad \mathbf{0}_3\ ]$$

$$\mathbf{L}_{{}^\emptyset\mathbf{t}_c} = [\ {}^{\emptyset}\mathbf{R}_c \quad \mathbf{0}_3\ ] \qquad \mathbf{L}_{{}^{c^*}\mathbf{t}_c} = [\ {}^{c^*}\mathbf{R}_c \quad \mathbf{0}_3\ ]$$

PBVS



$$\mathbf{s} = ({}^{c}\mathbf{t}_o, \theta\mathbf{u})$$

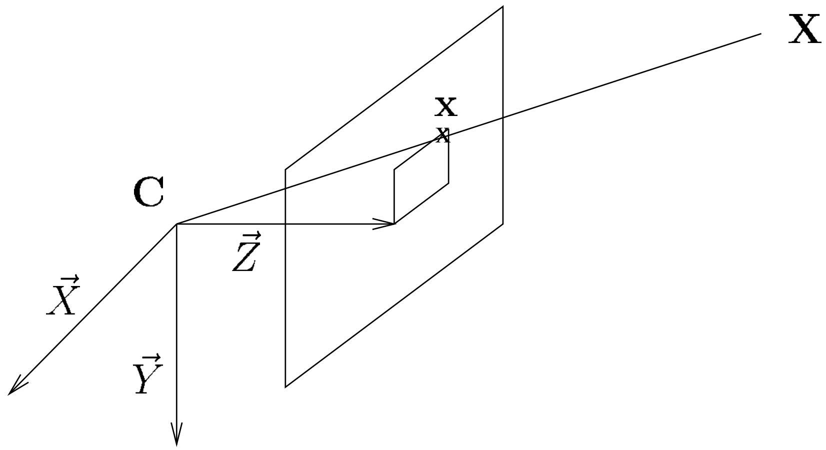
$$\mathbf{s} = \left({}^{c^*}\mathbf{t}_c, \theta\mathbf{u}\right)$$

Modeling issues

1. Basics
2. 3D visual features
3. 2D visual features
4. Numerical methods
5. Photometric features

2D visual features: image point coordinates

Perspective projection : $\mathbf{x} = (x, y)$



$$x = X/Z \ , y = Y/Z$$

$$\Rightarrow \begin{cases} \dot{x} = [\ 1/Z \quad 0 \quad -X/Z^2\ ]\ \dot{\mathbf{X}} \\ \dot{y} = [\ 0 \quad 1/Z \ -Y/Z^2\ ]\ \dot{\mathbf{X}} \end{cases}$$

Using a mobile camera and a fixed point:

$$\dot{\mathbf{X}} = \boldsymbol{v}(\mathbf{X}) = -\boldsymbol{v}(\mathbf{C}) - [\boldsymbol{\omega}]_\times \mathbf{CX} = \begin{bmatrix} -\mathbf{I}_3 & [\mathbf{X}]_\times \end{bmatrix} \mathbf{v}$$

We obtain:

$$\dot{\mathbf{x}} = \mathbf{L_x}\ \mathbf{v} \text{ where } \mathbf{L_x} = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{bmatrix}$$

2D visual features: image point coordinates

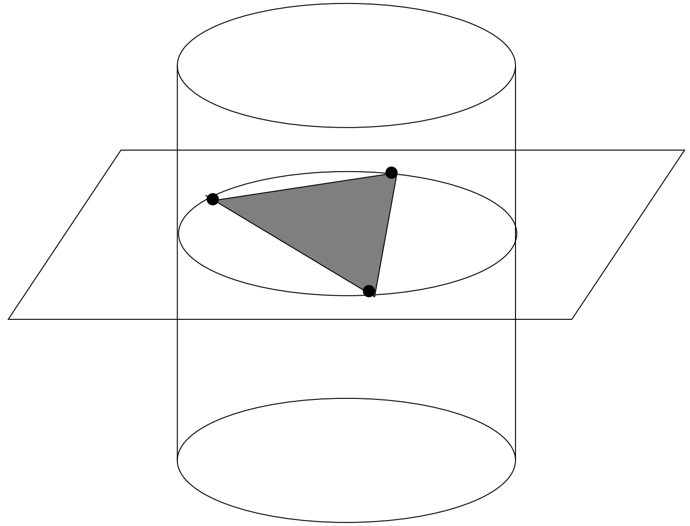
When $x = y = 0$ (principal point):

$$\mathbf{L_x} = \begin{bmatrix} -1/Z & 0 & 0 & 0 & -1 & 0 \\ 0 & -1/Z & 0 & 1 & 0 & 0 \end{bmatrix}$$

A single point is adequate to control v_x or ω_y and v_y or ω_x

Using several points (at least 3) allows to control the 6 dof.

$$\mathbf{s} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \Rightarrow \mathbf{L_x} = \begin{bmatrix} \mathbf{L}_{\mathbf{x}_1} \\ \vdots \\ \mathbf{L}_{\mathbf{x}_n} \end{bmatrix}$$



Be careful to singularities in the interaction matrix ($\Rightarrow n > 4$)

What's about 3D information

The depth Z_i of each point appears for the 3 translational dof (true $\forall \mathbf{s} \in 2\mathrm{D}$)

- Can be approximated:
- Can be estimated:
  - by triangulation with stereovision
  - from pose if 3D object model available
  - up to a scale factor from epipolar geometry/homography with current & desired images
  - from structure from known motion

Note:

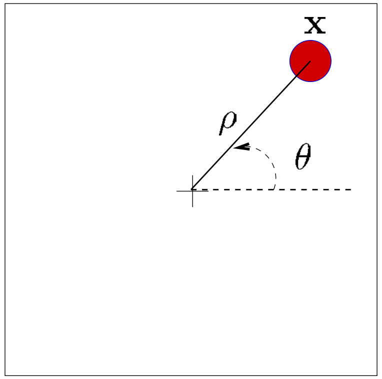
- For IBVS, the depth has an effect on the transient phase, not on the final accuracy (when the system is stable)
- For PBVS, 3D is involved for both the transient phase and the final accuracy, so problem in case of 3D noise

IBVS with points

Using $\mathbf{v} = -\lambda \mathbf{L}^{+}_{\mathbf{s}(\mathbf{s},\mathbf{Z})}(\mathbf{s} - \mathbf{s}^*)$

Using $\mathbf{v} = -\lambda \mathbf{L}^{+}_{\mathbf{s}(\mathbf{s}^*,\mathbf{Z}^*)}(\mathbf{s} - \mathbf{s}^*)$

Image point in cylindrical coordinates [Iwatsuki 02]



Use of (ρ, θ) for an image points instead of (x, y):

$$\rho = \sqrt{x^2 + y^2}\ , \ \ \theta = \arctan \frac{y}{x}$$

Corresponding interaction matrix:

$$\mathbf{L}_\rho = \left[\ \frac{-\cos\theta}{Z} \ \ \frac{-\sin\theta}{Z} \ \ \frac{\rho}{Z} \ \ (1+\rho^2)\sin\theta \ \ -(1+\rho^2)\cos\theta \ \ \ 0 \ \right]$$

$$\mathbf{L}_\theta = \left[\ \frac{\sin\theta}{\rho Z} \ \ \frac{-\cos\theta}{\rho Z} \ \ 0 \ \ \ \frac{\cos\theta}{\rho} \ \ \ \frac{\sin\theta}{\rho} \ \ \ -1 \ \right]$$

Better decoupling between v_z and ω_z

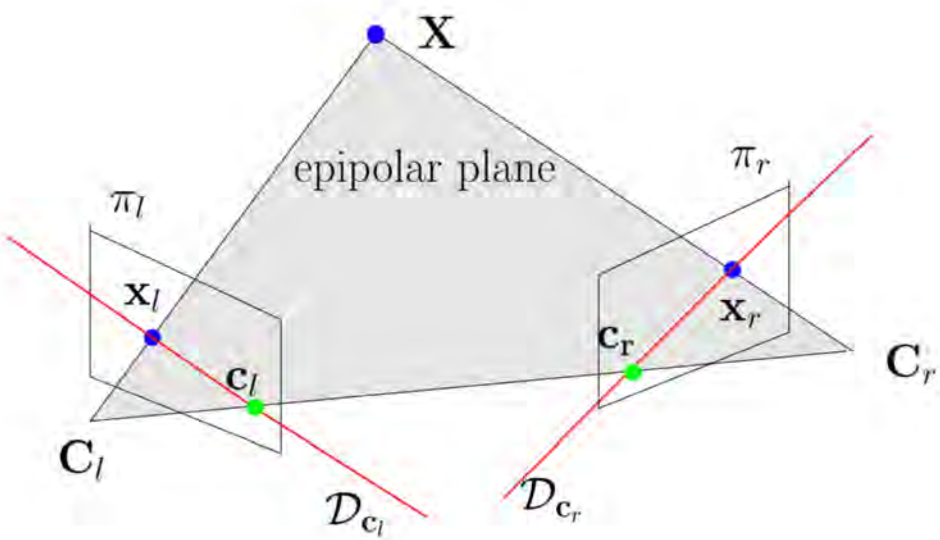
Be careful for the principal point ($x = y = \rho = 0,\ \theta$ undefined)

Image point for a stereovision system

$$\dot{\mathbf{x}}_l = \mathbf{L}_{\mathbf{x}_l} \, \mathbf{v}_l$$
$$\dot{\mathbf{x}}_r = \mathbf{L}_{\mathbf{x}_r} \, \mathbf{v}_r$$

$$\Rightarrow \begin{bmatrix} \dot{\mathbf{x}}_l \\ \dot{\mathbf{x}}_r \end{bmatrix} = \mathbf{L}_{\mathbf{x}_l \mathbf{x}_r} \, \mathbf{v}_c$$

$$\text{where } \mathbf{L}_{\mathbf{x}_l \mathbf{x}_r} = \begin{bmatrix} \mathbf{L}_{\mathbf{x}_l} {}^l\mathbf{V}_c \\ \mathbf{L}_{\mathbf{x}_r} {}^r\mathbf{V}_c \end{bmatrix}$$



$\mathbf{L}_{\mathbf{x}_l \mathbf{x}_r}$ is of rank 3 because of the epipolar constraint

- Generalization to multi-cameras systems immediate
- Probably better to use the coordinates of the 3D point

2D visual features: geometrical primitives

P_o : configuration of an *object feature* parameterized by $\mathbf{P_o}$
$p_i = \pi(P_o)$: configuration of an *image feature* parameterized by $\mathbf{p_i}$

Noting $\mathbf{P_o} = \varphi(P_o)$ and $\mathbf{p_i} = \psi(p_i)$, we get

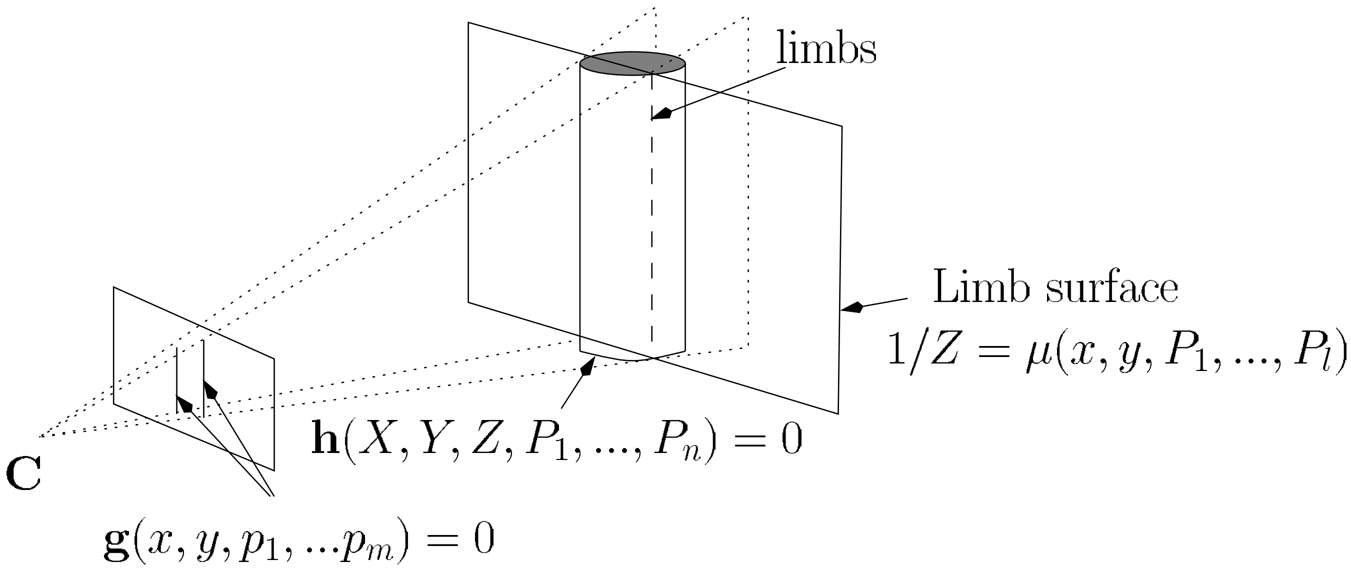
$$\mathbf{p_i} = \nu(\mathbf{P_o}) = \psi \circ \pi \circ \varphi^{-1}(\mathbf{P_o})$$

We also have $\mathbf{P_o} = \varphi \circ \delta(\mathbf{p}) \Rightarrow \mathbf{p_i} = \psi \circ \pi \circ \delta(\mathbf{p}) = \nu \circ \varphi \circ \delta(\mathbf{p})$

$$\begin{tikzcd}
\underset{(\mathbf{p})}{W \subseteq SE_3} \arrow[r, "\delta"'] & \underset{(P_o)}{U \subseteq \mathcal{P}_o} \arrow[r, "\pi"'] \arrow[d, "\varphi"] & \underset{(p_i)}{V \subseteq \mathcal{P}_i} \arrow[d, "\psi"] & \\
 & \underset{(\mathbf{P_o})}{\mathbb{R}^n} \arrow[r, "\nu = \psi \circ \pi \circ \varphi^{-1}"'] & \underset{(\mathbf{p_i})}{\mathbb{R}^m} \arrow[r, "\sigma"'] & \underset{(\mathbf{s})}{\mathbb{R}^k}
\end{tikzcd}$$

Finally $\mathbf{s} = \sigma(\mathbf{p_i}) \Rightarrow \mathbf{L_s} = \frac{\partial \mathbf{s}}{\partial \mathbf{p_i}} \frac{\partial \mathbf{p_i}}{\partial \mathbf{P_o}} \mathbf{L_{P_o}}$

Modeling a geometrical primitive



3D primitive: $\mathbf{h}(\mathbf{X}, \mathbf{P_o}) = 0$

2D primitive: $\mathbf{g}(\mathbf{x}, \mathbf{p_i}) = 0$

Limb surface: $\Rightarrow 1/Z = \mu(\mathbf{x}, \mathbf{P_o})$

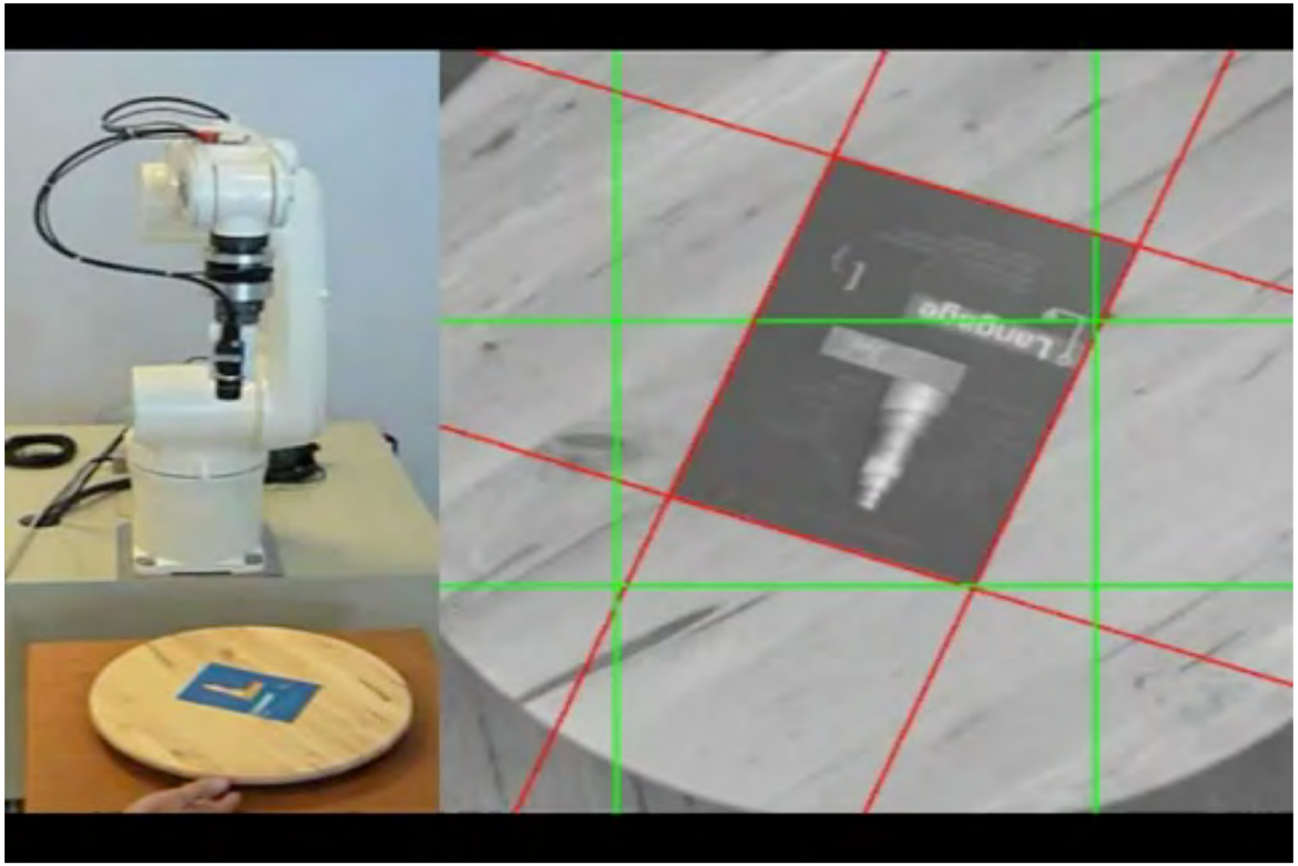
(planar limb surface $\Leftrightarrow 1/Z = Ax + By + C$)

Summary

3D primitives	2D primitives	Parameterization
point	point	(x, y) or (ρ, θ)
segment	segment	(x_1, y_1, x_2, y_2) $(x_m/l, y_m/l, 1/l, \alpha)$
straight line	straight line	(ρ, θ)
circle	ellipse	$(x_g, y_g, \mu_{20}, \mu_{11}, \mu_{02})$
sphere	ellipse	$(x_g, y_g, a = \pi r^2)$
cylinder	2 straight lines	$(\rho_1, \theta_1, \rho_2, \theta_2)$
planar object	moments	$(a, x_g, y_g, \theta, ...)$

$\mathbf{L_s}$ also available for distance from a point to a straight line, angle between two straight lines, etc.

IBVS with straight lines



Modeling issues

1. Basics
2. 3D visual features
3. 2D visual features
4. Numerical methods
5. Photometric features

Other approach: direct numerical estimation

Using N measurements of $\mathbf{v_c}$ and corresponding $\dot{\mathbf{s}}$ around $\mathbf{s}^*$

- Off-line learning of $\mathbf{L_s}$:

  With 1 measurement, $\mathbf{L_s}\,\mathbf{v_c} = \dot{\mathbf{s}}$: k equations and $k \times 6$ unknowns

  With $N (\geq 6)$, $\mathbf{L_s}\,\mathbf{A} = \mathbf{B}$ where $\mathbf{A} \in \mathbb{R}^{6\times N}$ and $\mathbf{B} \in \mathbb{R}^{k\times N}$

$$\Rightarrow \widehat{\mathbf{L_s}} = \mathbf{B}\mathbf{A}^+$$

- Off-line learning of $\mathbf{L_s^+}$ (better method):

  With 1 measurement, $\mathbf{L_s^+}\,\dot{\mathbf{s}} = \mathbf{v_c}$: 6 equations and $6 \times k$ unknowns

  With $N (\geq k)$, $\mathbf{L_s^+}\,\mathbf{B} = \mathbf{A} \Rightarrow \widehat{\mathbf{L_s^+}} = \mathbf{A}\mathbf{B}^+$

- Other methods: neural networks,...

Methods valid locally around $\mathbf{s}^*$ only since $\mathbf{L_s}$ is not constant.

Stability impossible to demonstrate

Other approach: direct numerical estimation

On-line iterative estimation (based on Broyden update):

$$\widehat{\mathbf{L}_{\mathbf{s}}}(t+1) = \widehat{\mathbf{L}_{\mathbf{s}}}(t) + \frac{\alpha}{\mathbf{v_c}^\top \mathbf{v_c}} \left(\dot{\mathbf{s}} - \widehat{\mathbf{L}_{\mathbf{s}}}(t)\mathbf{v_c} \right) \mathbf{v_c}^\top$$

Be careful to initial value $\widehat{\mathbf{L}_{\mathbf{s}}}(t0)$

Stability impossible to demonstrate

May be useful for unknown complex objects or unmodeled systems

Modeling issues

1. Basics
2. 3D visual features
3. 2D visual features
4. Numerical methods
5. Photometric features

A new family of visual servoing: photometric VS

Remove the image processing part:

- No more extraction nor tracking visual measurements near video rate

Advantages:

- Robustness to image processing errors and noise!
- End-to-end control (here without deep learning)

Photometric/direct/dense visual servoing

Visual features: intensity of each pixel $\quad \mathbf{s} = \mathbf{I}(\mathbf{x}(t))$

$\mathbf{I}^*$ $\quad\quad$ $\mathbf{I}$ $\quad\quad$ $\mathbf{I} - \mathbf{I}^*$

Modeling: $\mathbf{L_I} = -\nabla \mathbf{I_x}\ \mathbf{L_x}$ (function of the image content)

$\mathcal{L} = \frac{1}{2}\|\mathbf{I} - \mathbf{I}^*\|$ highly non linear

Drawbacks: small convergence domain, strange robot trajectory

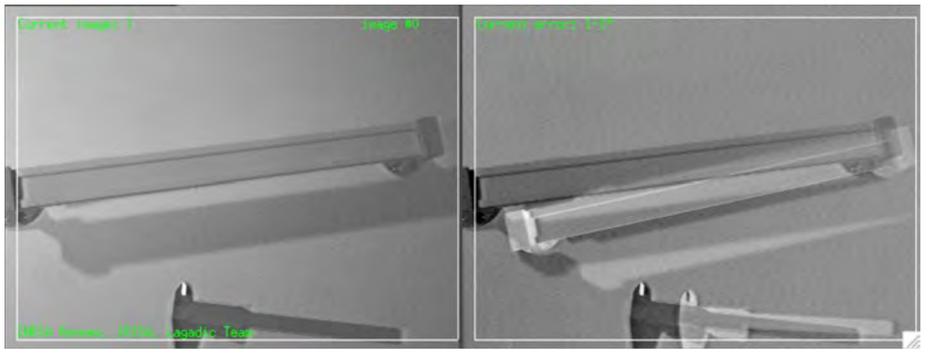
But no feature extraction, tracking nor matching

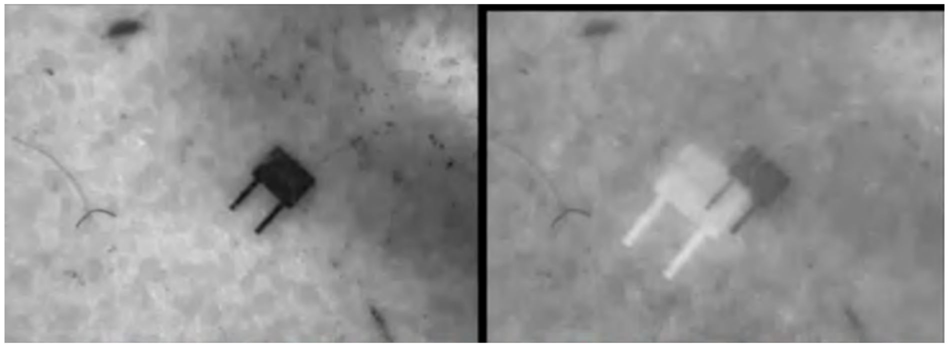
$+$ excellent positioning accuracy

Photometric visual servoing

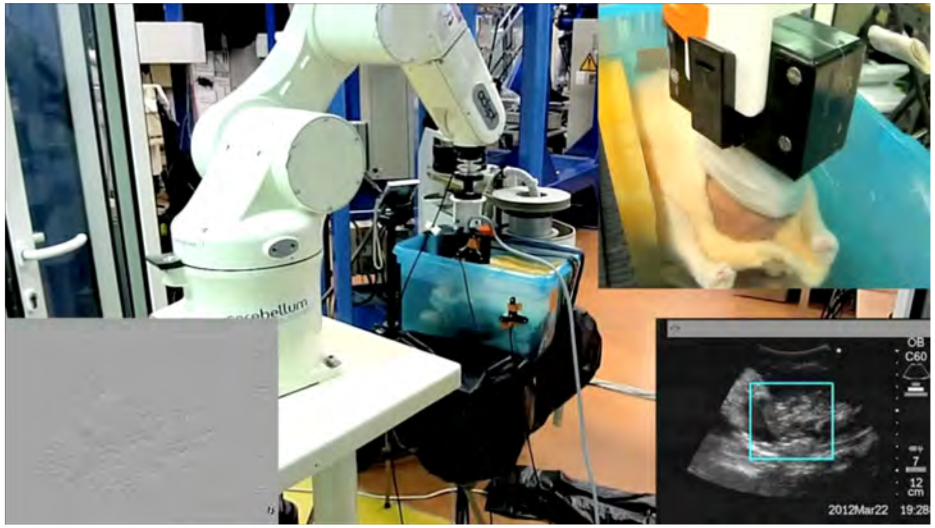
Robustness to global illumination changes by using $\mathbf{s} = (\mathbf{I} - \bar{\mathbf{I}})/\sigma_{\mathbf{I}}^2$

Robustness to outliers (occlusion) by using $\mathbf{s} = \rho_{\mathbf{I}}\ \mathbf{I}$

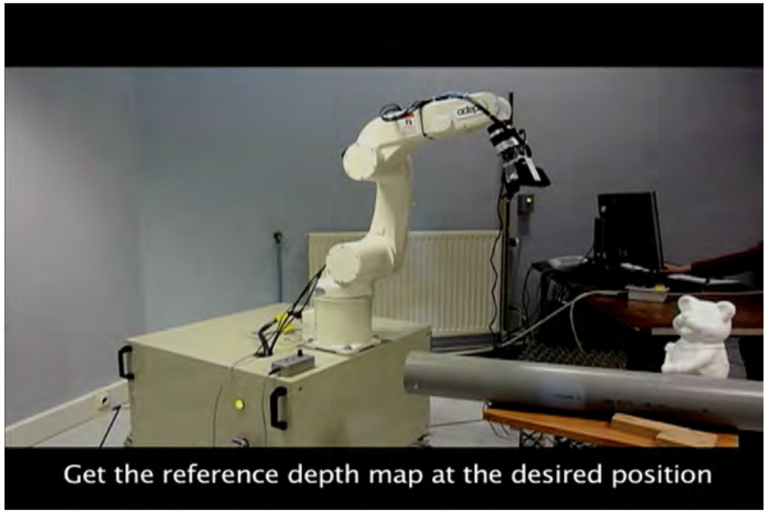




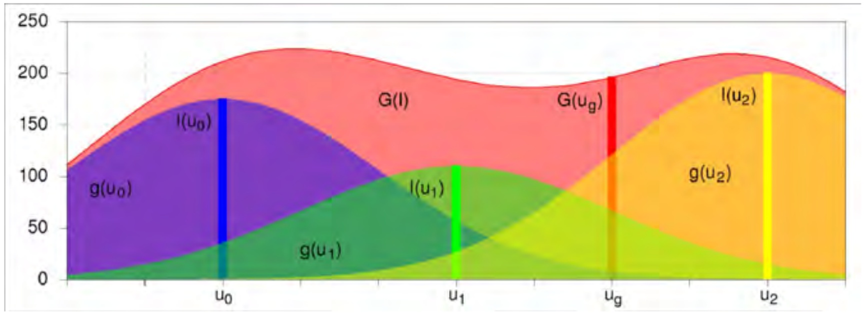
Similar on 2D ultrasound images



Similar on depth map

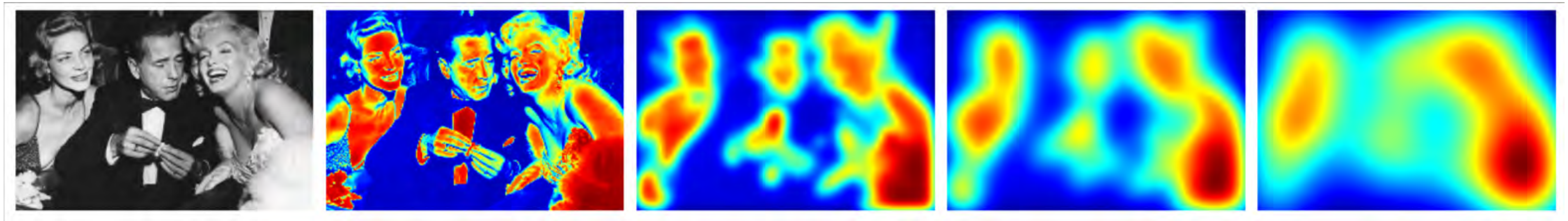


Mixture of Gaussians



Enlarge the convergence domain

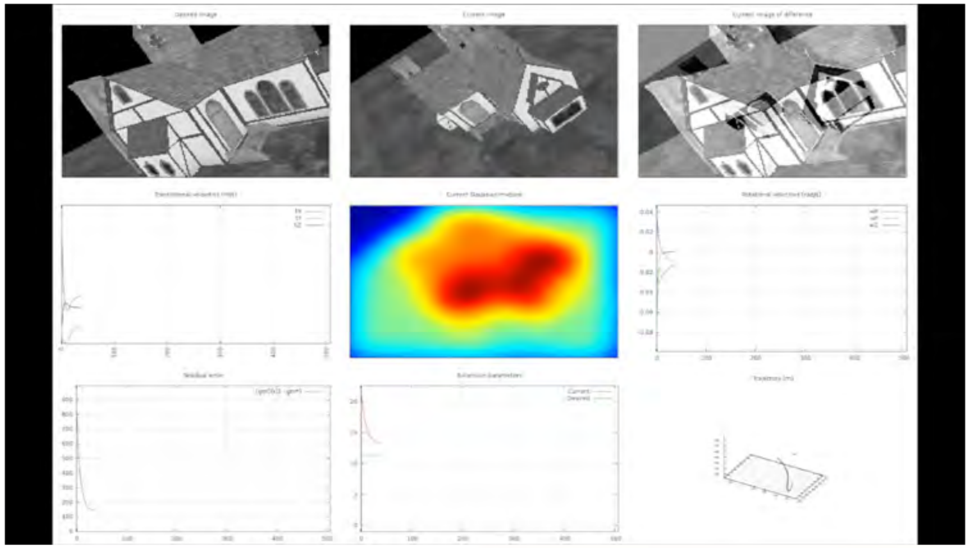
$$G(\mathbf{u_g}, \lambda_g) = \sum_{\mathbf{u_i} \in \mathbf{I}} I(\mathbf{u_i}) \exp \left(- \frac{(u_g - u_i)^2 + (v_g - v_i)^2}{2\lambda_g^2} \right)$$



$\lambda_g = 0.1$ $\lambda_g = 5$ $\lambda_g = 10$ $\lambda_g = 20$

Control simultaneously

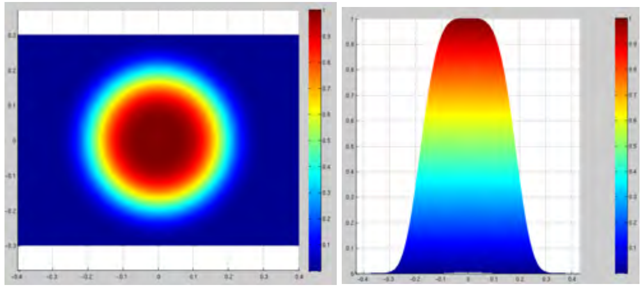
- the camera motion
- the expansion parameter (large to small)



Photometric moments

Going back to geometric features for enlarging the convergence domain and improving the robot trajectory

$$m_{pq} = \iint_{\pi} x^p y^q \, w(\mathbf{x}) \; I(\mathbf{x}, t) \; \mathrm{d}x \, \mathrm{d}y$$



Then select adequate moments (area, cog, main orientation, …)

$\mathbf{I}^*$

$\mathbf{I}$

$\mathbf{I} - \mathbf{I}^*$

Overview

1. Introduction
2. Modeling
3. Task specification
4. Control

Task specification

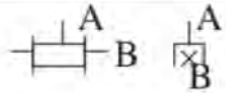
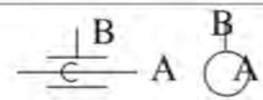
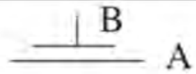
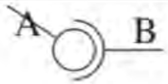
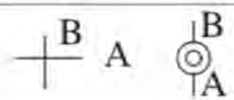
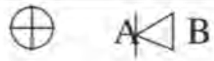
- Just specify **s*** or **s***(t)
  (such as an object has to appear at the center of the image)
- Specify a desired pose and deduce the value of **s***
  (but 3D model of the object + camera calibration needed)
- Teach by showing:
  1. go to the desired pose;
  2. acquire the corresponding image;
  3. determine **s*** in the same way as **s**(t).

Task classification: virtual link

- The task $\mathbf{s}(t) = \mathbf{s}^*$ defines a virtual link between the sensor and its environment.
- This link is characterized by the set $\mathcal{S}^*$ of 3D motions such that $\dot{\mathbf{s}} = 0$

$$\mathcal{S}^* = \text{Ker } \mathbf{L_s}$$

- dim $\mathcal{S}^*$ = class of the link

Name	Class	T	R	Geometric symbol
Rigid	0	0	0	
Prismatic	1	1	0	
Rotary	1	0	1	
Sliding pivot	2	1	1	
Plane-to-plane	3	2	1	
Bearing	3	0	3	
Linear rectlinear	4	2	2	
Linear annular	4	1	3	
Point	5	2	3	

Case of a point

$$\mathbf{s} = (x, y)$$

$$\Rightarrow \mathbf{L}_{xy} = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{bmatrix}$$

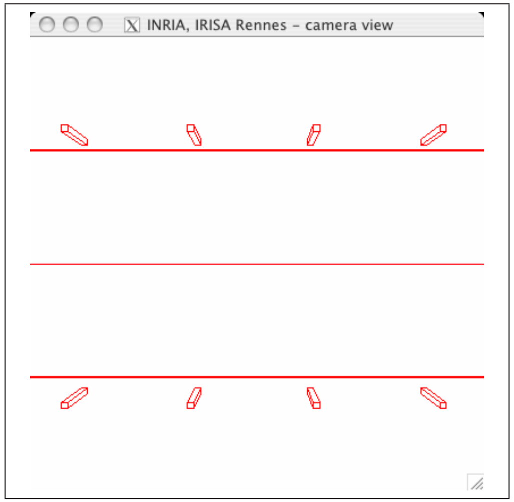
$$\Rightarrow \mathcal{S}^* = \begin{bmatrix} x & 0 & Z(1+x^2+y^2) & 0 \\ y & 0 & 0 & Z(1+x^2+y^2) \\ 1 & 0 & 0 & 0 \\ 0 & x & -xy & 1+x^2 \\ 0 & y & -(1+y^2) & xy \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$\Rightarrow$ Link of class 4

Prismatic link

$$\mathcal{S}^* = (1, 0, 0, 0, 0, 0)$$

Using 3 (horizontal) straight lines



3D straight lines :

$$\mathbf{h}_i(\mathbf{X}, \mathbf{P}) = \begin{cases} Y - \dfrac{Y_i^*}{Z_i^*} Z = 0 \\ Z - Z_i^* = 0 \end{cases}, \quad i = 1, 2, 3$$

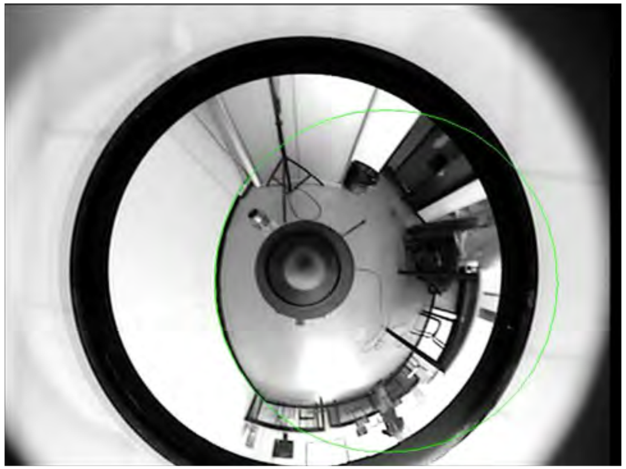
2D straight lines : $\rho_i = Y_i^*/Z_i^*, \ \theta_i = \pi/2$

$$\Rightarrow \mathbf{L}_{\rho_i \theta_i} = \begin{bmatrix} 0 & -1/Z_i^* & \rho_i/Z_i^* & (1+\rho_i^2) & 0 & 0 \\ 0 & 0 & 0 & 0 & -\rho_i & -1 \end{bmatrix}$$

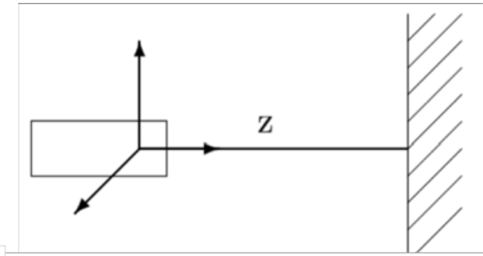
With a 3 dof mobile robot (v_x, v_z, ω_y), 1 straight line is sufficient.

Prismatic link from an omnidirectional camera

3 dof ground mobile robot so the observation of 1 straight line (here a circle) is sufficient to achieve the task



Plane-to-plane link from proximity sensors



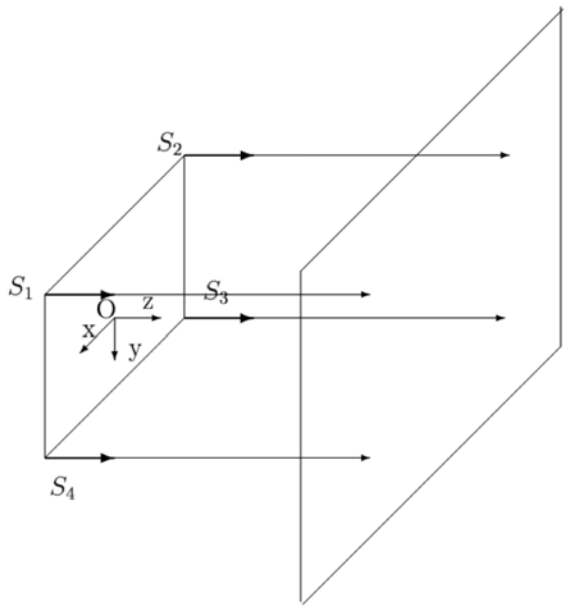
A narrow beam proximity sensor provides the range Z from the sensor to the nearest object along the sensor axis.

When the object surface is perpendicular to the sensor axis: $\mathbf{L}_Z = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix}$

$$\mathbf{L}_{Z_i|R_O} = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbb{I}_3 & -[\mathbf{S_i}]_\times \\ 0 & \mathbb{I}_3 \end{pmatrix}$$
$$= \begin{pmatrix} 0 & 0 & -1 & -Y_i & X_i & 0 \end{pmatrix}$$

where $\mathbf{S}_i = (X_i, Y_i, 0)_{|R_O}$.

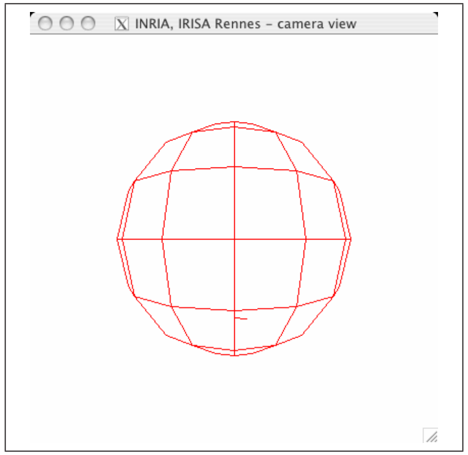
$$\Rightarrow \mathcal{S}^* = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



Bearing

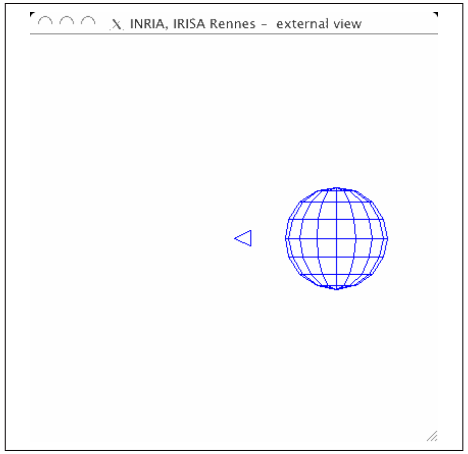
Using a sphere with center $\mathbf{O} = (0, 0, Z_0)$

$\Rightarrow$ Image of the sphere = centered circle



$$
\begin{aligned}
\mathbf{L}_{x_c} &= [\; -1/Z_c \quad 0 \quad 0 \quad 0 \quad -1-r^2 \quad 0 \;] \\
\mathbf{L}_{y_c} &= [\; 0 \quad -1/Z_c \quad 0 \quad 1+r^2 \quad 0 \quad 0 \;] \\
\mathbf{L}_{\mu} &= [\; 0 \quad 0 \quad 2r^2/Z_c \quad 0 \quad 0 \quad 0 \;]
\end{aligned}
$$

with $Z_c = (Z_0^2 - R^2)/Z_0$ and $r^2 = R^2/(Z_0^2 - R^2)$.



$$
\Rightarrow S^* = \begin{bmatrix} 0 & -z_0 & 0 \\ z_0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{|\mathcal{F}_c} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{|\mathcal{F}_o}
$$

Overview

1. Introduction
2. Modeling
3. Task specification
4. Control

Control

Basic kinematics controller: $\mathbf{v}_c = -\lambda\, \widehat{\mathbf{L}_\mathbf{s}}^+ (\mathbf{s} - \mathbf{s}^*)$

with $$\widehat{\mathbf{L}_\mathbf{s}} = \begin{cases} \mathbf{L}_\mathbf{s}(\mathbf{s}, \mathbf{Z}) \\ \mathbf{L}_\mathbf{s}(\mathbf{s}, \mathbf{Z}^*) \\ \mathbf{L}_\mathbf{s}(\mathbf{s}^*, \mathbf{Z}^*) \\ \frac{1}{2}\left(\mathbf{L}_\mathbf{s}(\mathbf{s}, \mathbf{Z}) + \mathbf{L}_\mathbf{s}(\mathbf{s}^*, \mathbf{Z}^*)\right) \end{cases}$$

Time-to-convergence improved by using an adaptive gain $\hat{\lambda}$

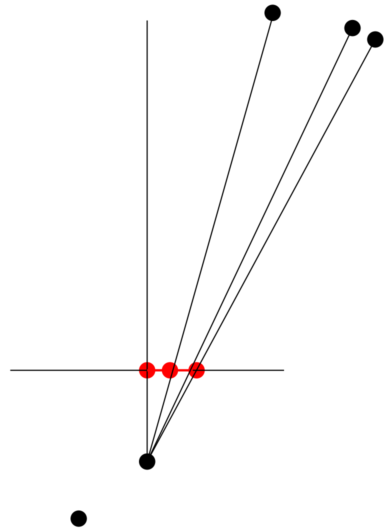
High initial velocities avoided using 2nd order behavior (see ViSP)

Note: If the robot is not able to apply $\mathbf{v}_c$, use $\dot{\mathbf{q}} = -\lambda\, \widehat{\mathbf{J}_\mathbf{s}}^+ (\mathbf{s} - \mathbf{s}^*)$

with $\widehat{\mathbf{J}_\mathbf{s}} = \widehat{\mathbf{L}_\mathbf{s}}\, {}^c\mathbf{V}_n\, {}^n\mathbf{J}_n(\mathbf{q})$ (remember slide 17)

A simple case $k = m = n = 2$

Case of a pan-tilt camera observing a point :

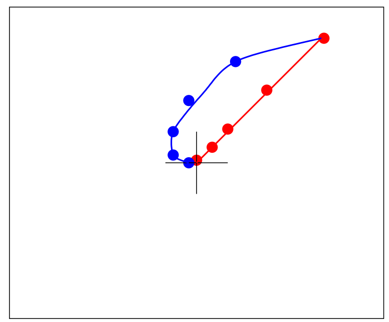


$$\mathbf{s} = (x, y)\ ,\ \mathbf{s}^* = (0, 0)$$

$$\dot{\mathbf{e}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} xy & -(1+x^2) \\ 1+y^2 & -xy \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix}$$

$$\mathbf{v}_c = -\lambda\, \widehat{\mathbf{L}_{\mathbf{s}}}^{-1} (\mathbf{s} - \mathbf{s}^*)$$

$$\Leftrightarrow \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} = -\frac{\lambda}{1 + x^2 + y^2} \begin{bmatrix} y \\ -x \end{bmatrix}$$



If no error occurs, $\dot{\mathbf{s}} = -\lambda\, \mathbf{s}$: trajectory = straight line in the image

Target tracking

1st solution: Use an integral term to compensate for the lag

$$\mathbf{v_c} = -\lambda\, \mathbf{e} + \mu \sum_{j=0}^{k} \mathbf{e}_j \qquad \text{with } \mathbf{e} = \widehat{\mathbf{L_s}}^{+} (\mathbf{s} - \mathbf{s}^*)$$

- Need to tune the gain μ
- Efficient only for target moving at constant velocity

2nd solution: Estimate, predict and compensate for the target motion

$$\mathbf{v}_c = -\lambda\, \mathbf{e} + \widehat{\frac{\partial e}{\partial t}} \quad \text{with} \quad \widehat{\frac{\partial e}{\partial t}} \quad \text{the predicted value of} \quad \widehat{\frac{\partial e}{\partial t}} = \hat{\dot{\mathbf{e}}} - \widehat{\mathbf{L_s}} \mathbf{v}_c$$

obtained for instance from a Kalman filter and with $\hat{\dot{\mathbf{e}}}_k = \dfrac{\mathbf{e}_k - \mathbf{e}_{k-1}}{\Delta t}$

- Need to measure the camera velocity $\mathbf{v}_c$

To go further

Consider constraints:

- visibility, occlusion, obstacles
- joint limits, singularities
- dynamics constraints: non holonomy, under-actuation
  - Path planning in the image, model-predictive control, optimal control (QP)
  - Redundancy (GPM), task sequencing, stack of tasks