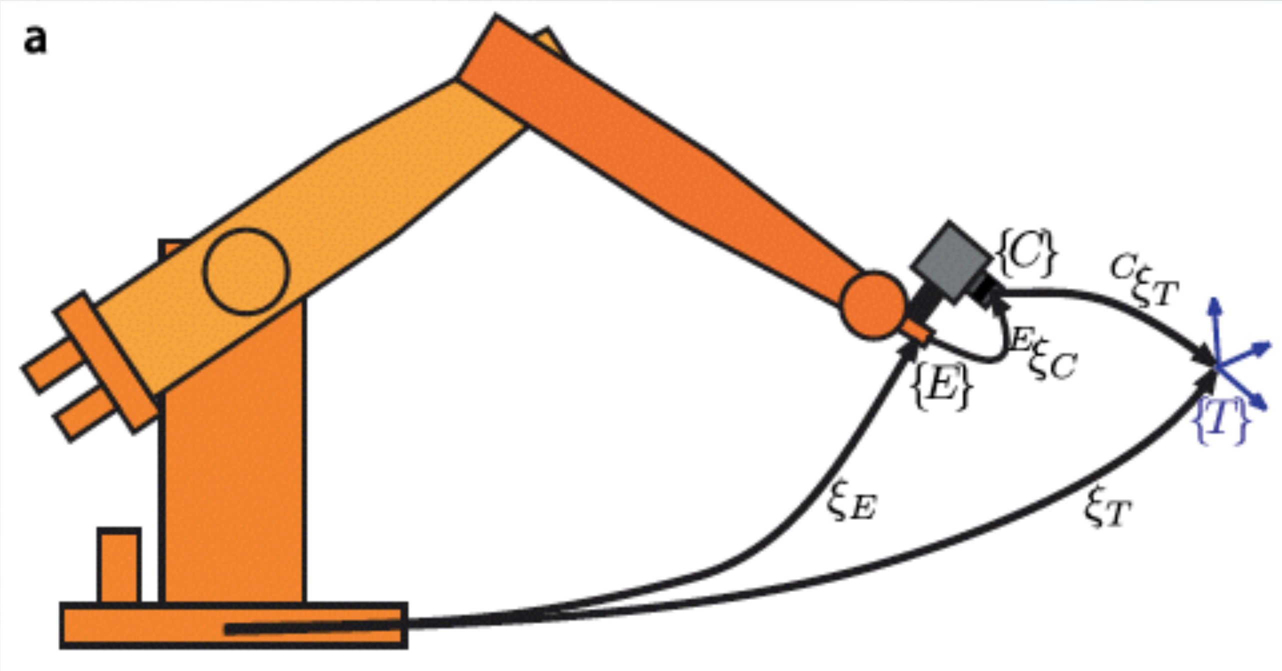
VISUAL SERVOING

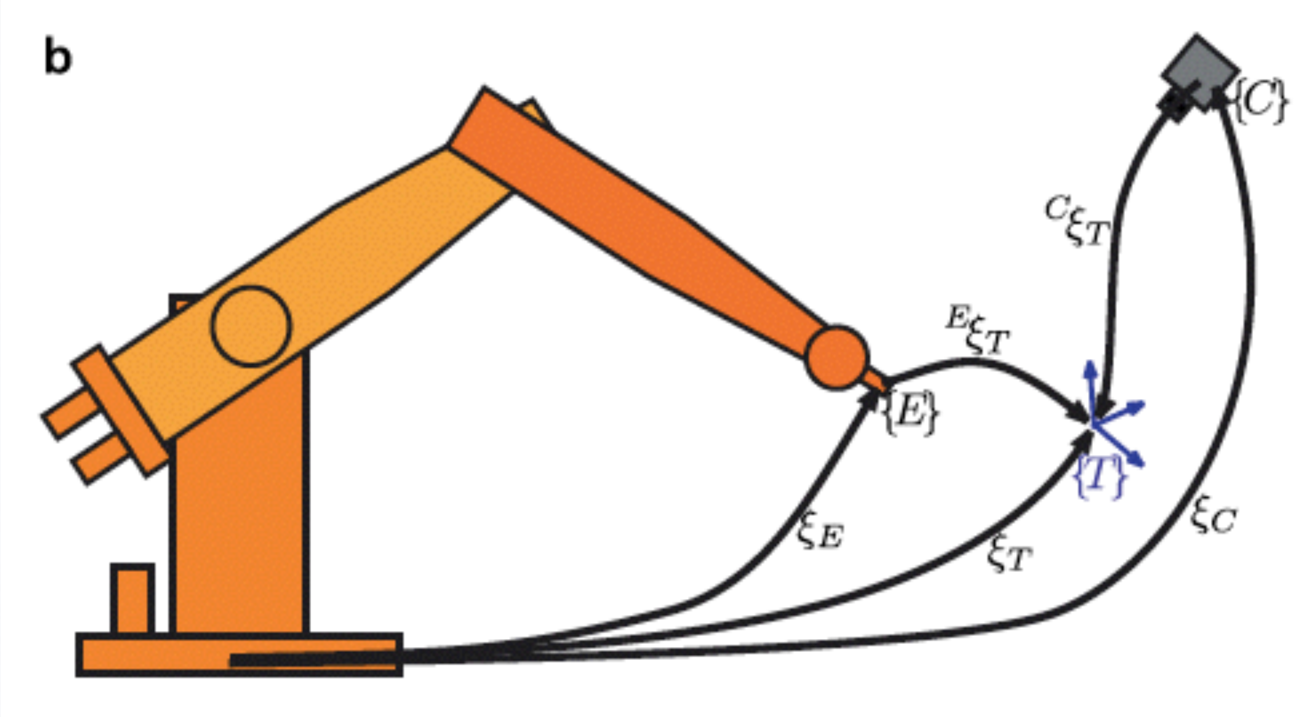
CS 3630 Introduction to Robotics and Perception

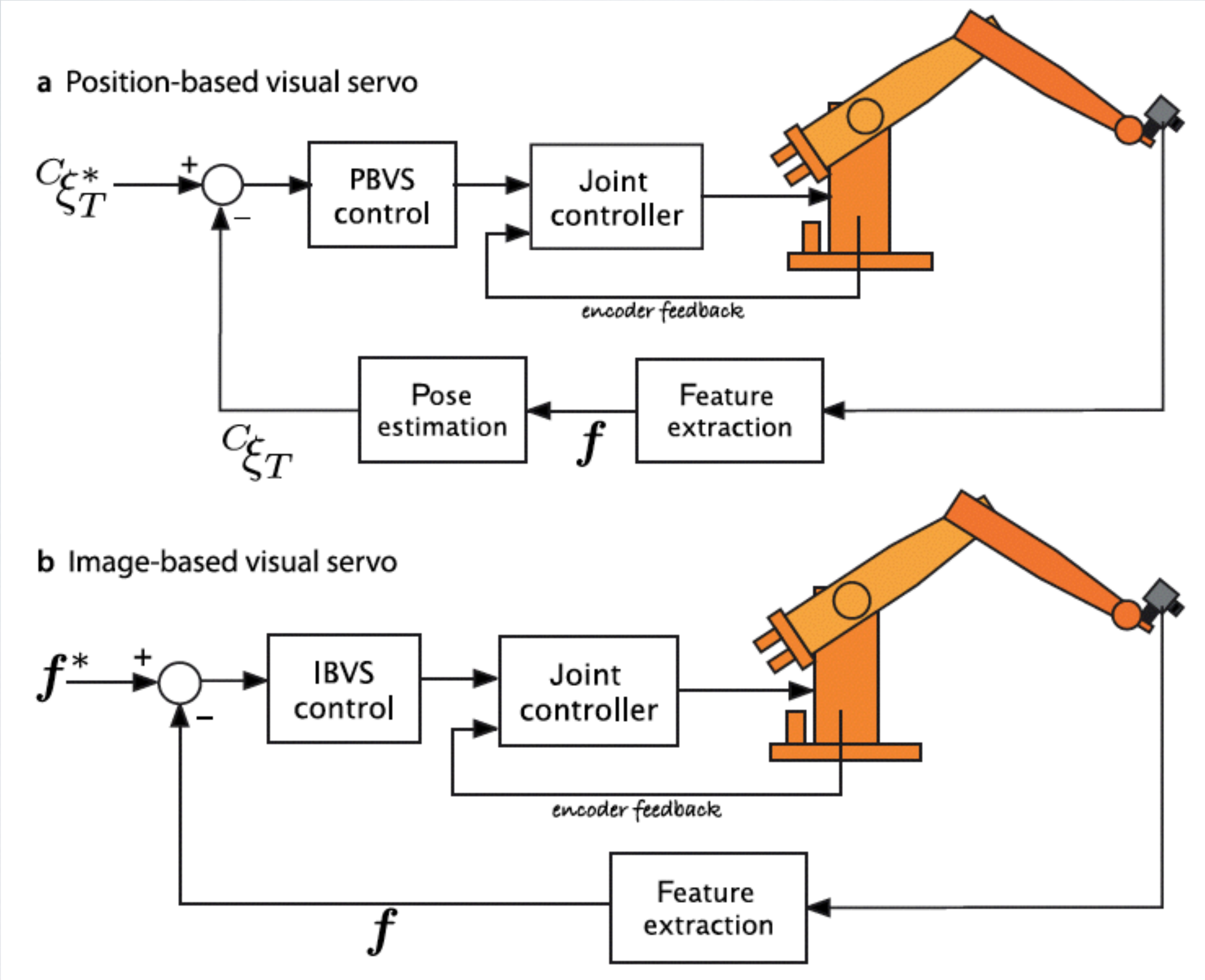
Frank Dellaert

INTRO (READ CORKE 15!)

- Visual Servoing:
  Control End-effector using visual features!
  - end-point closed-loop or **eye-in-hand**
  - end-point open-loop



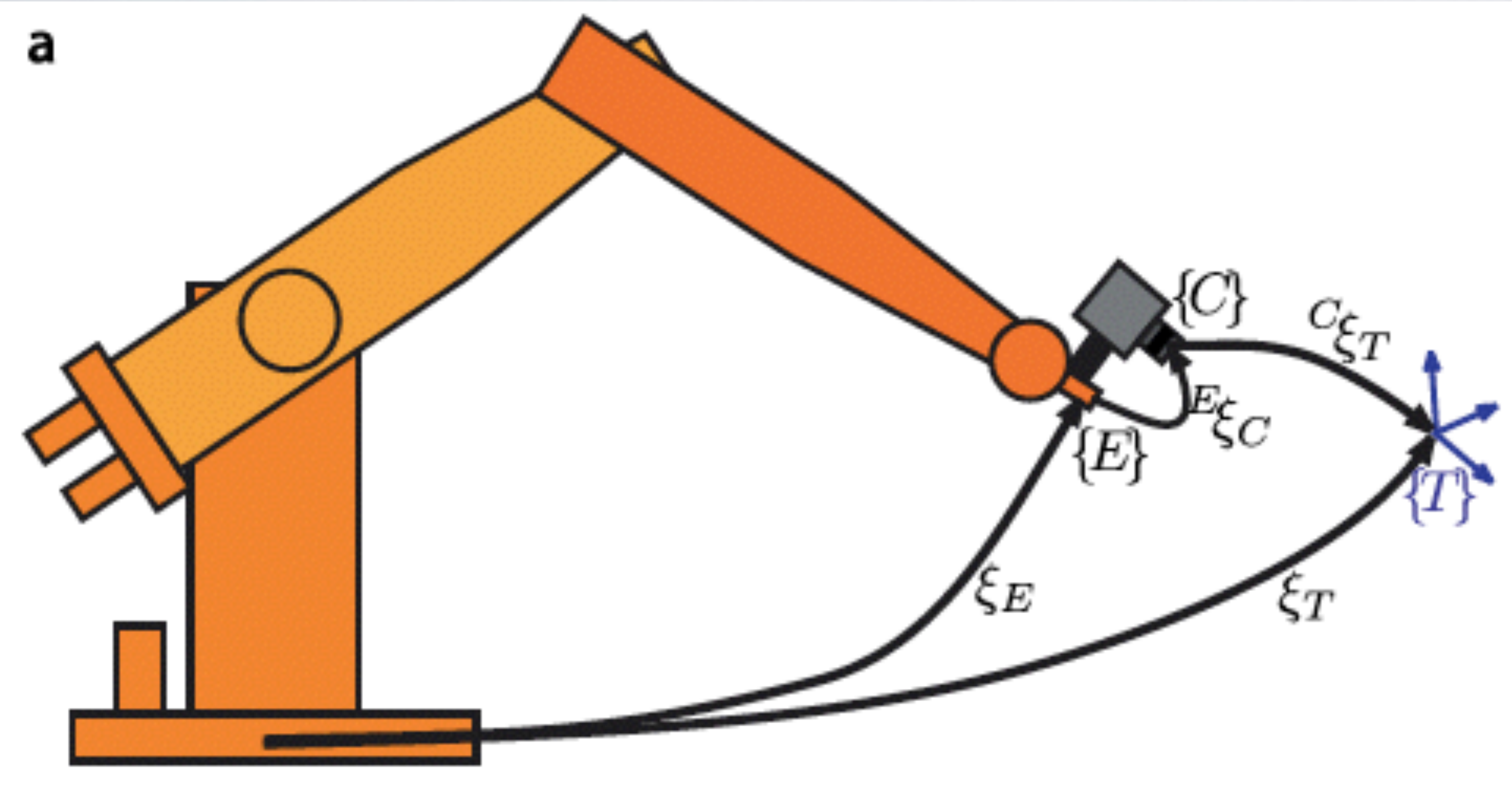


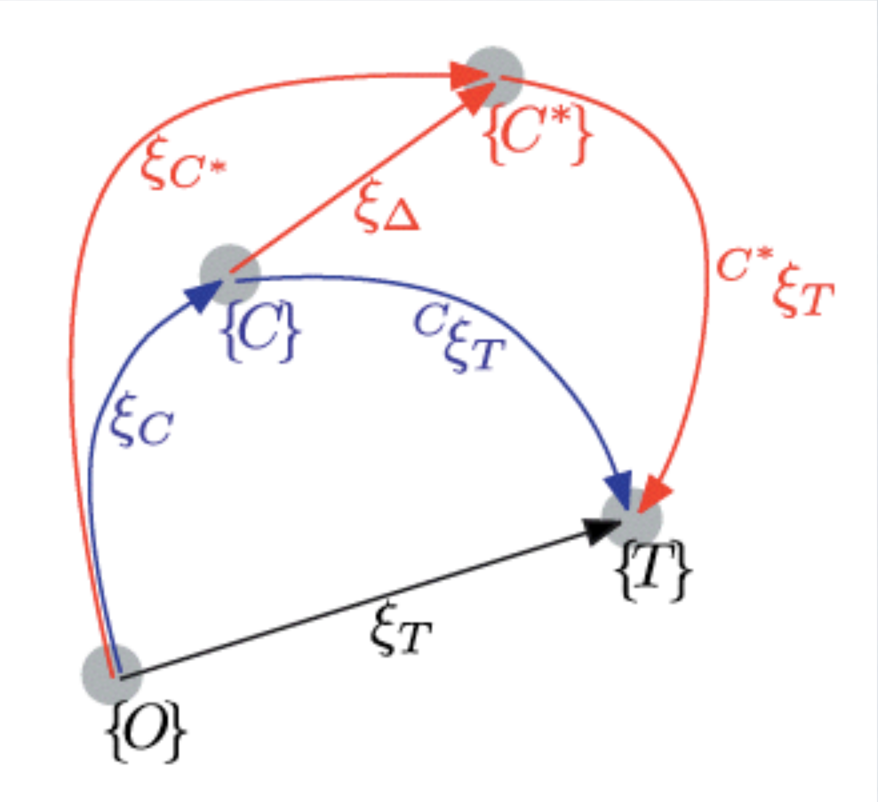


Expensive, requires good calibration

Challenging as non-linear projection involved

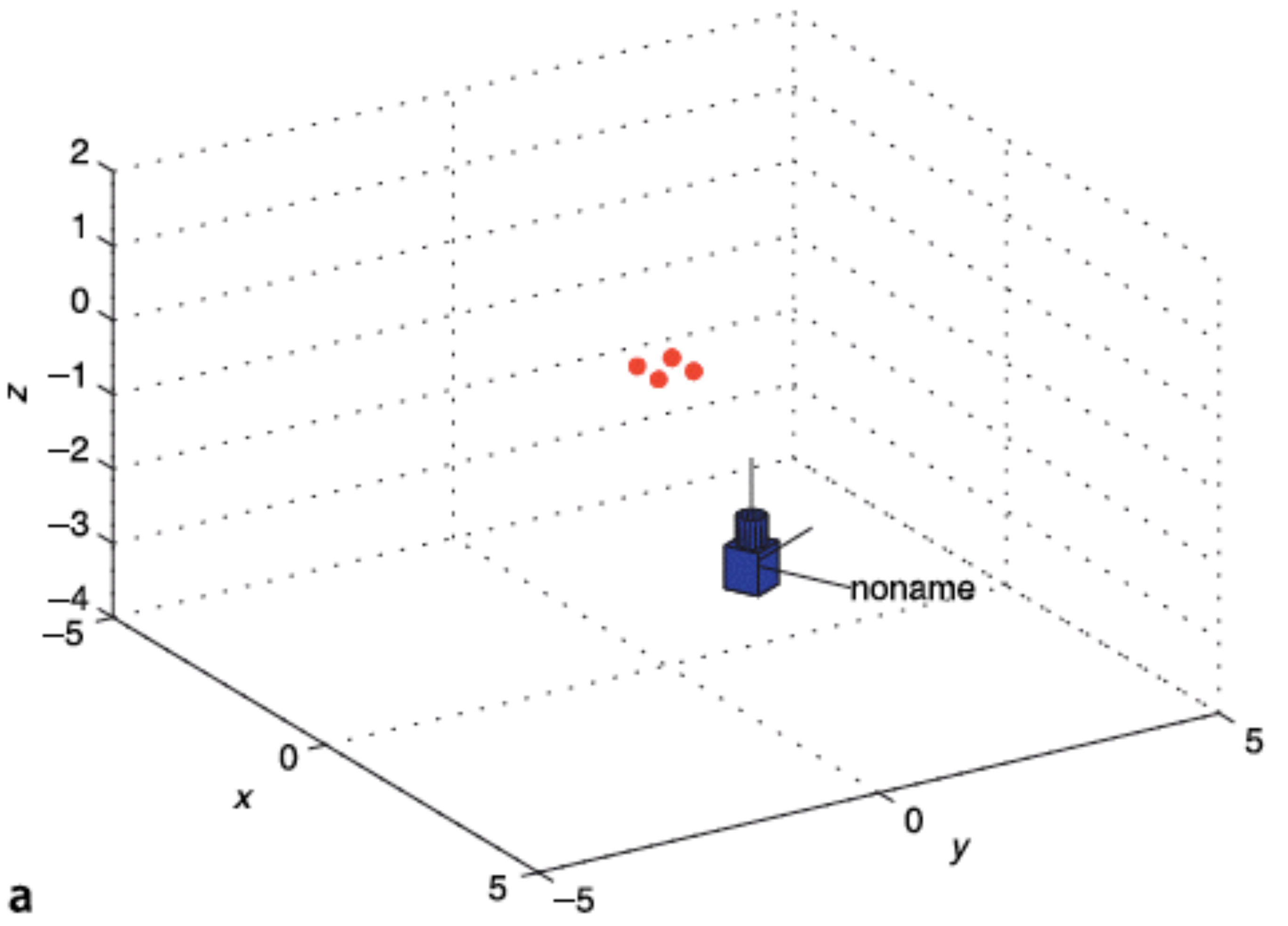
- Position-Based Visual Servo (PBVS) vs Image-Based Visual Servo (IBVS)

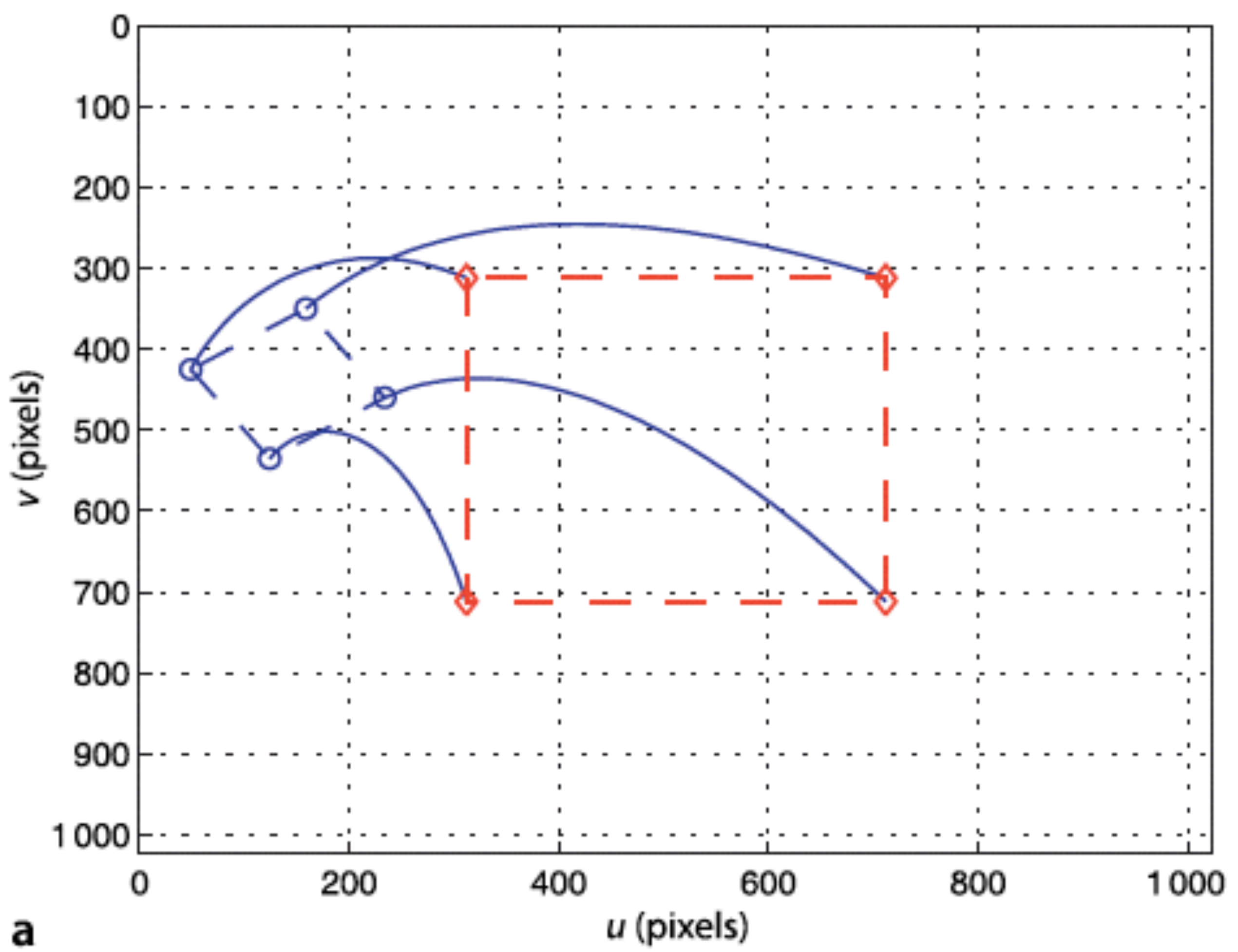




POSITION-BASED VISUAL SERVO

Estimate Pose, calculate delta pose $\xi_\Delta = {}^C\xi_T \ominus {}^{C^*}\hat{\xi}_T$



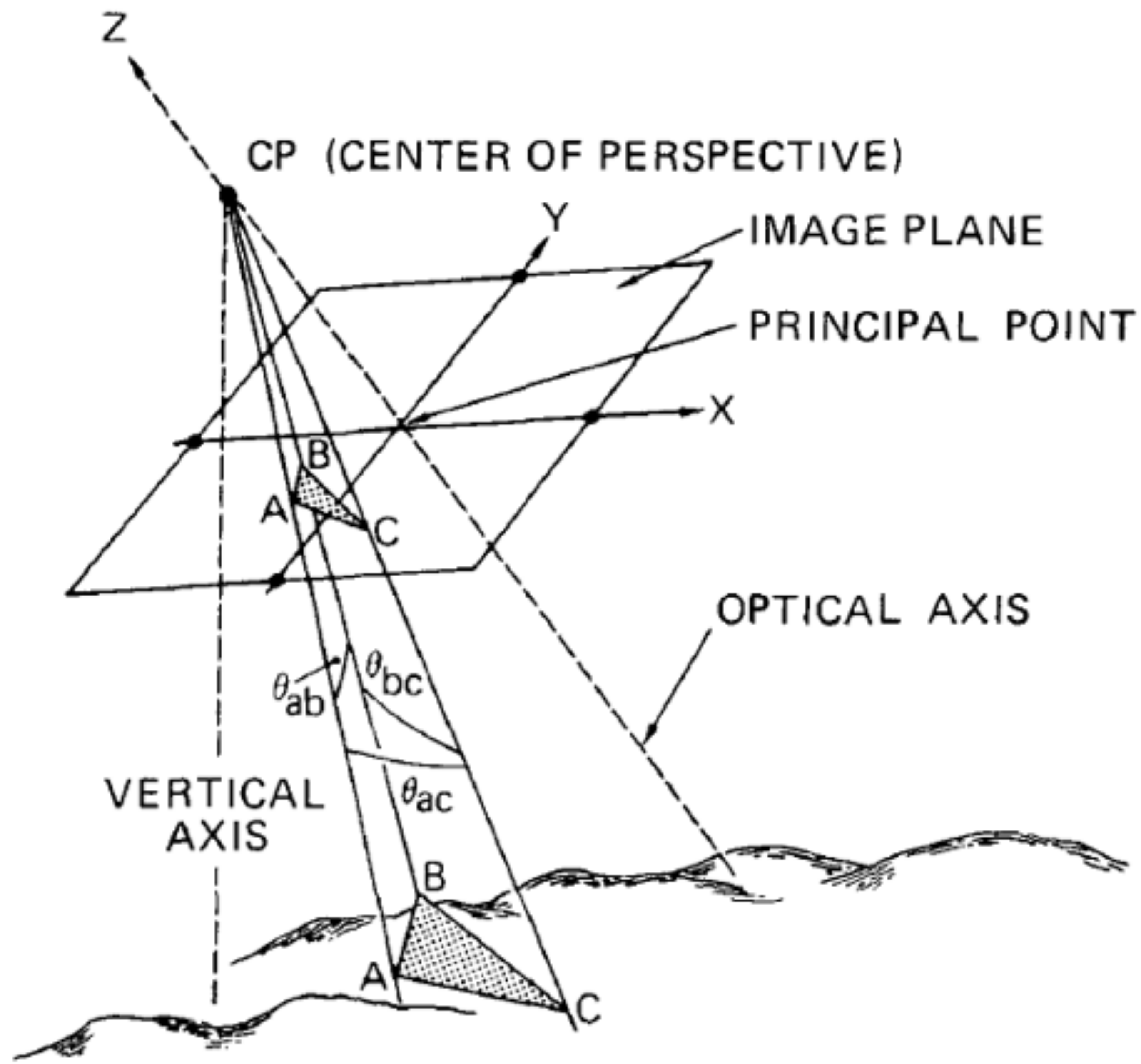


POSITION-BASED VISUAL SERVO

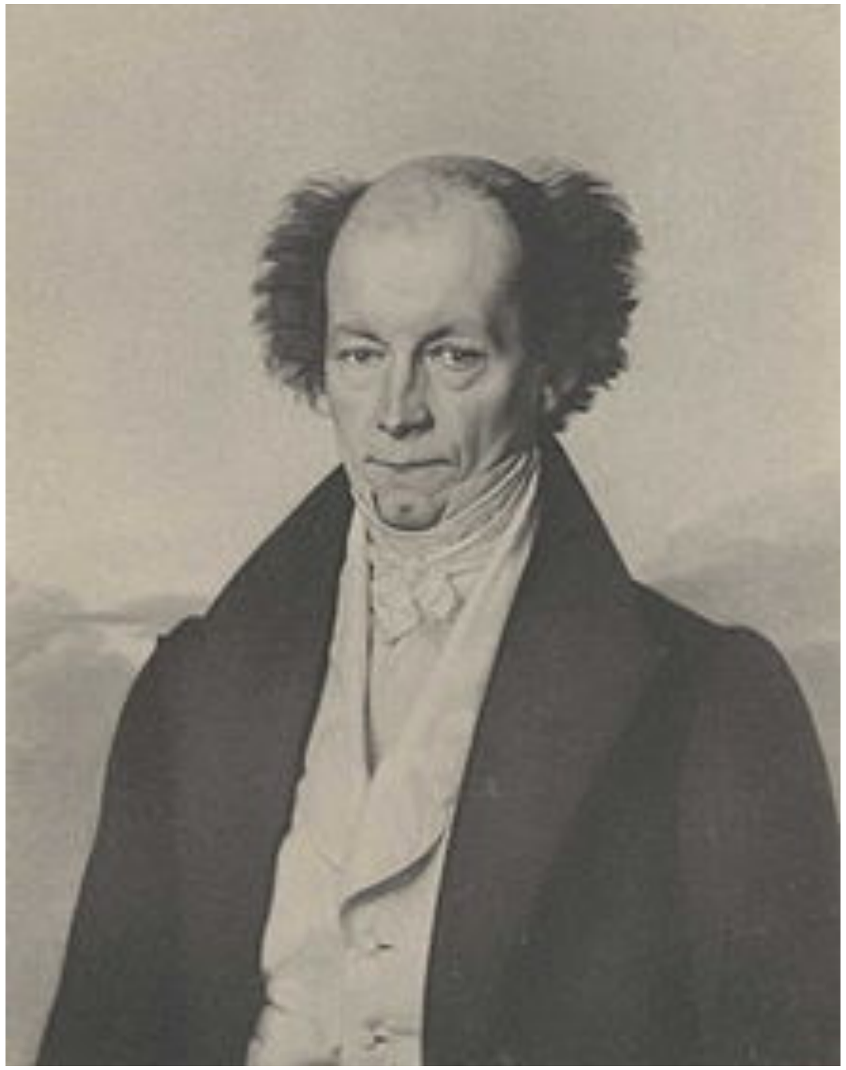
Estimate Pose, calculate delta pose

Three-point Algorithm

- 3-point perspective pose estimation
- Applications:
  - Camera calibration
  - Object recognition
  - Robot picking
  - Visual odometry
  - Photogrammetry



History



- Grunert 1841
- Church 45: Iterative
  - Needs good estimate
- Fischler & Bolles
  - Seminal RANSAC paper
- Haralick 94: review
- Nister 04: generalized
- Moreno 07:

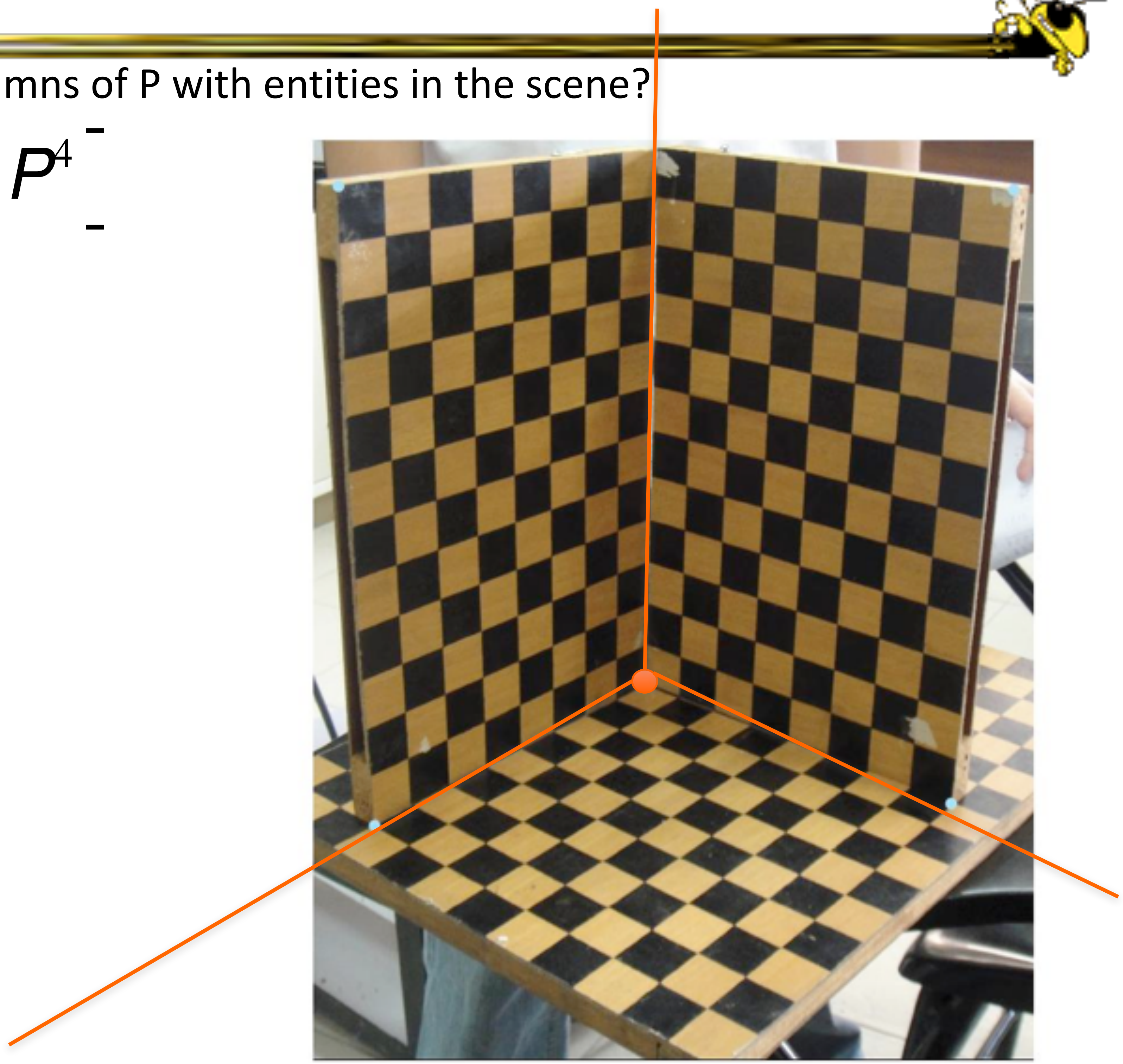
Resectioning
Finding a Camera Given Known points

- SVD: 6-point algorithm
- Apply cross-product trick
- Take Notes

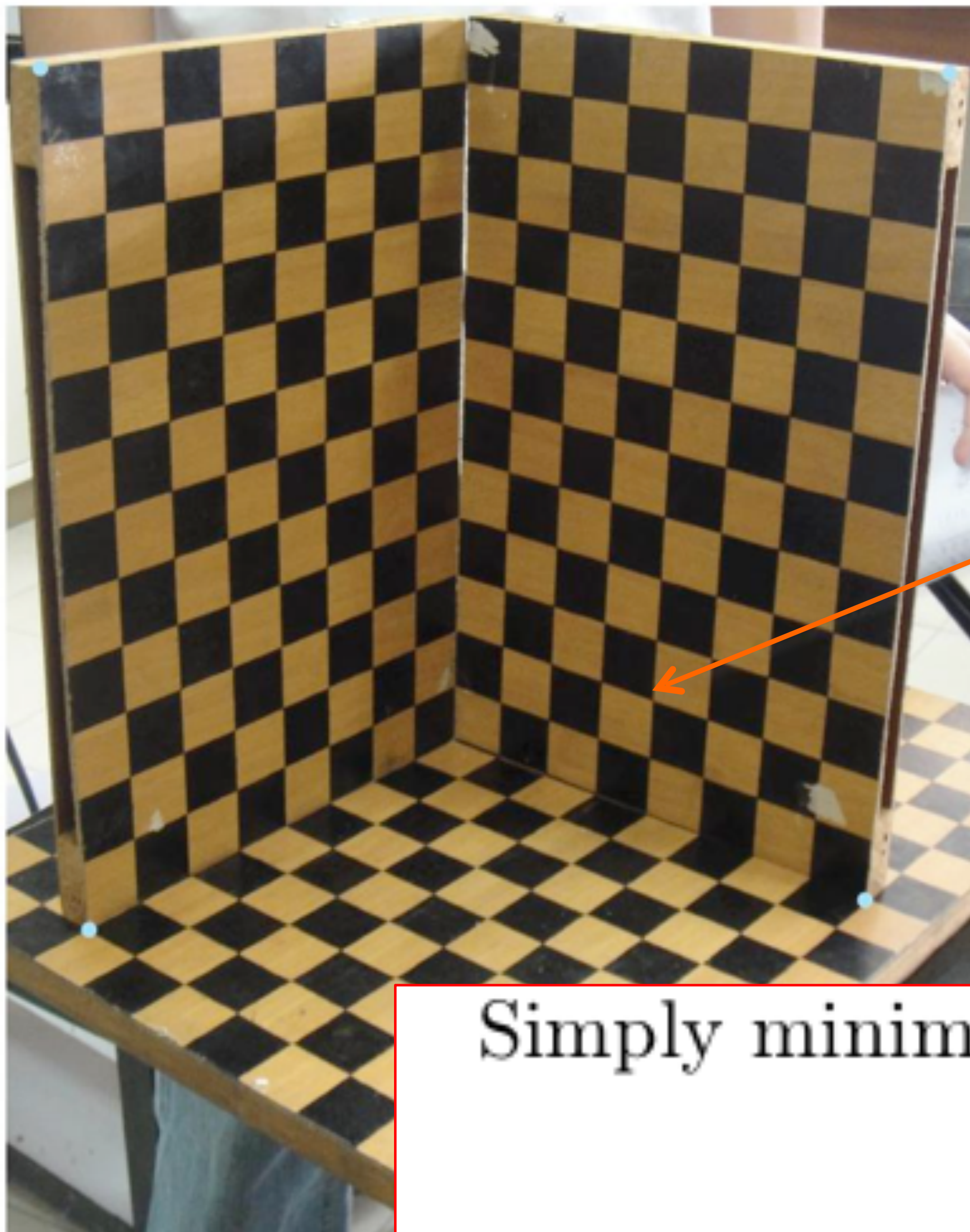
3D Target and Vanishing Points

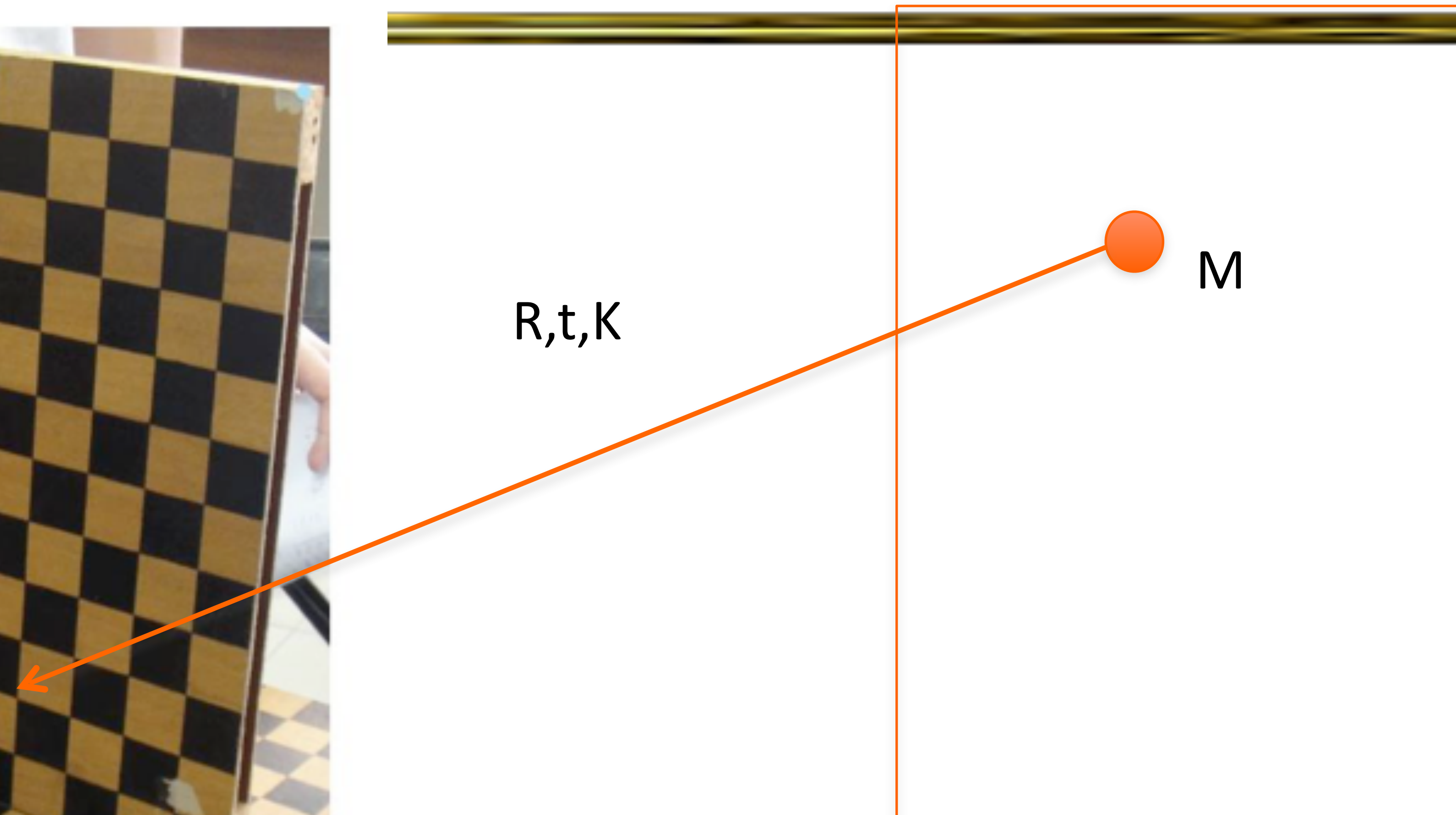
Can we you interpret the columns of P with entities in the scene?

$$P = \begin{bmatrix} P^1 & P^2 & P^3 & P^4 \end{bmatrix}$$



3D Target, Non-linear Minimization





Simply minimize the following sum-squared error::

$$\sum_j \|m_j - \hat{m}(A, R, t, M_j)\|^2$$

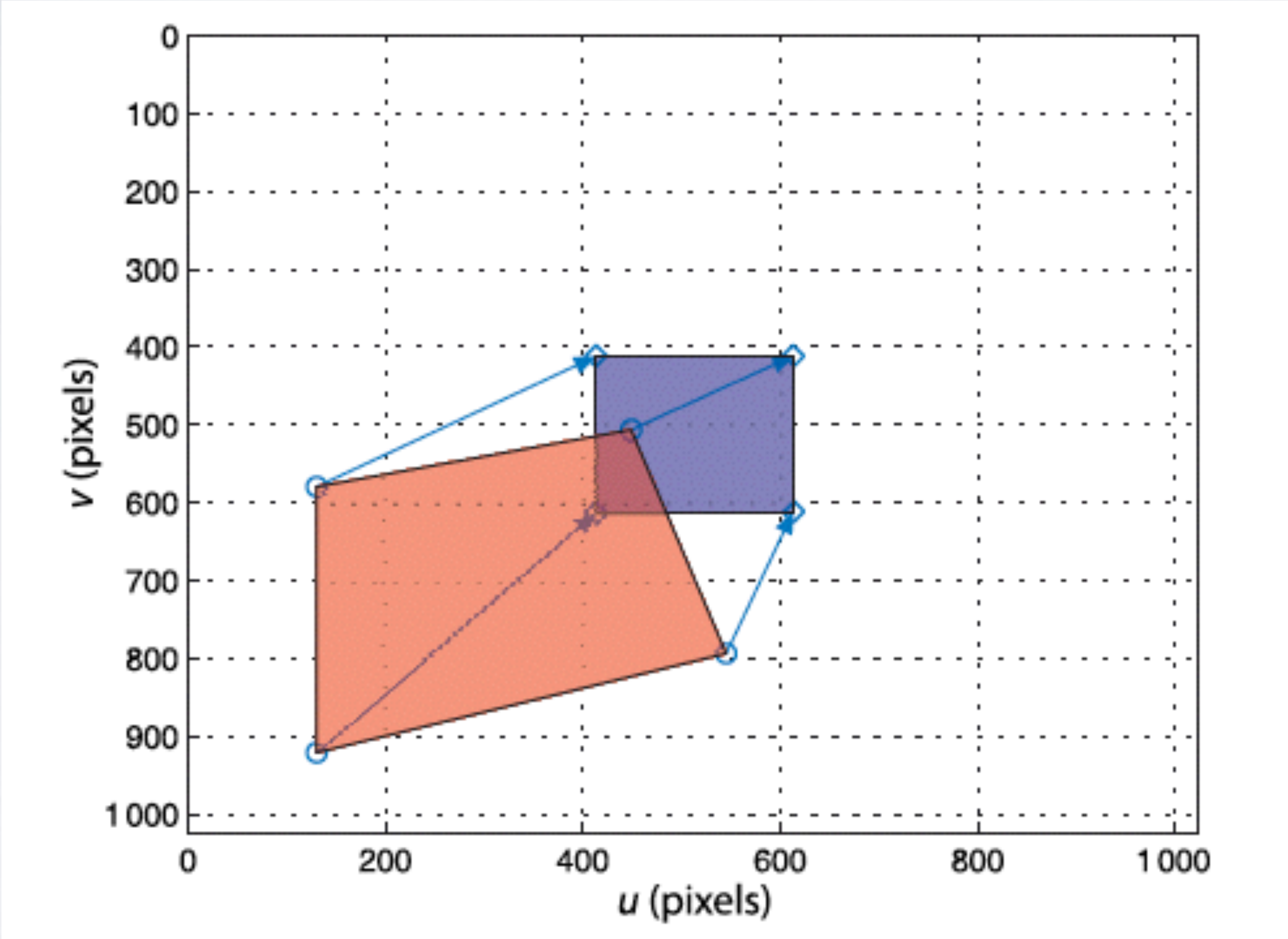


IMAGE-BASED VISUAL SERVO

Estimate Pose, calculate delta pose $\xi_\Delta = {}^{C}\xi_T \ominus {}^{C^*}\hat{\xi}_T$

IMAGE JACOBIAN

- Camera moves with
  - angular velocity $\boldsymbol{\omega}$
  - linear velocity $\boldsymbol{v}$
- How does image p of point P move ?
- This slide: normalized coordinates.

$$x = \frac{X}{Z},\ y = \frac{Y}{Z}$$

$$\dot{x} = \frac{\dot{X}Z - X\dot{Z}}{Z^2},\ \dot{y} = \frac{\dot{Y}Z - Y\dot{Z}}{Z^2}$$

$$\dot{\boldsymbol{P}} = -\boldsymbol{\omega} \times \boldsymbol{P} - \boldsymbol{v}$$

$$\begin{aligned} \dot{X} &= Y\omega_z - Z\omega_y - v_x \\ \dot{Y} &= Z\omega_x - X\omega_z - v_y \\ \dot{Z} &= X\omega_y - Y\omega_x - v_z \end{aligned}$$

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

IMAGE JACOBIAN

- Camera moves with:
  - angular velocity ω
  - linear velocity $\boldsymbol{v}$
- How does image p of point P move ?
- This slide: pixel coordinates. Corke notation: focal length f in metric units, ρ_u and ρ_v are pixel dimensions

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

$$u = \frac{f}{\rho_u}x + u_0, \; v = \frac{f}{\rho_v}y + v_0$$

which we rearrange as

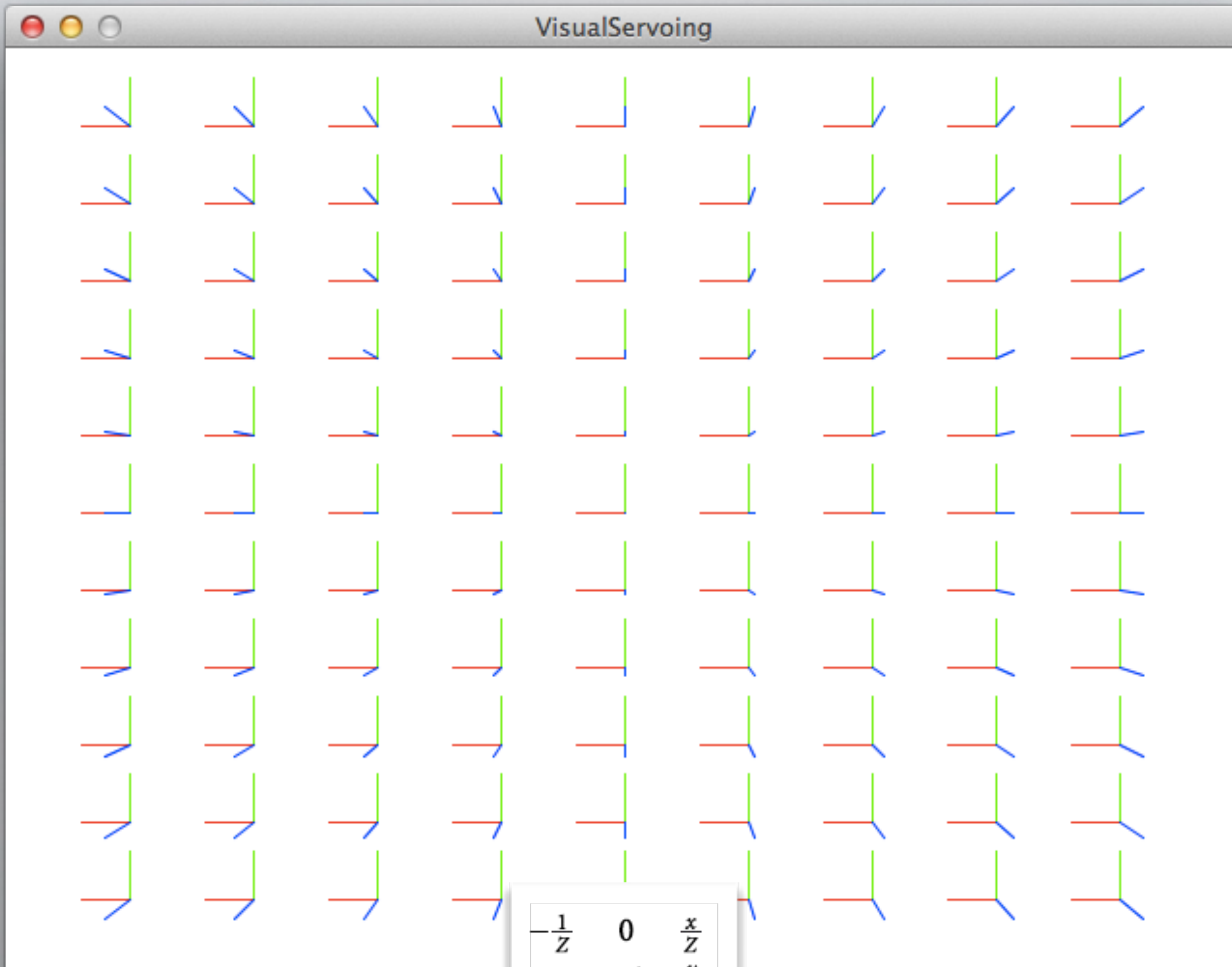
$$x = \frac{\rho_u}{f}\bar{u}, \; y = \frac{\rho_v}{f}\bar{v}$$

$$\begin{pmatrix}\dot{\bar{u}}\\ \dot{\bar{v}}\end{pmatrix} = \underbrace{\begin{pmatrix} -\frac{f}{\rho_u Z} & 0 & \frac{\bar{u}}{Z} & \frac{\rho_u \bar{u}\bar{v}}{f} & -\frac{f^2+\rho_u^2\bar{u}^2}{\rho_u f} & \bar{v} \\ 0 & -\frac{f}{\rho_v Z} & \frac{\bar{v}}{Z} & \frac{f^2+\rho_v^2\bar{v}^2}{\rho_v f} & -\frac{\rho_v \bar{u}\bar{v}}{f} & -\bar{u} \end{pmatrix}}_{J_p} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

A visual impression of the Image Jacobian

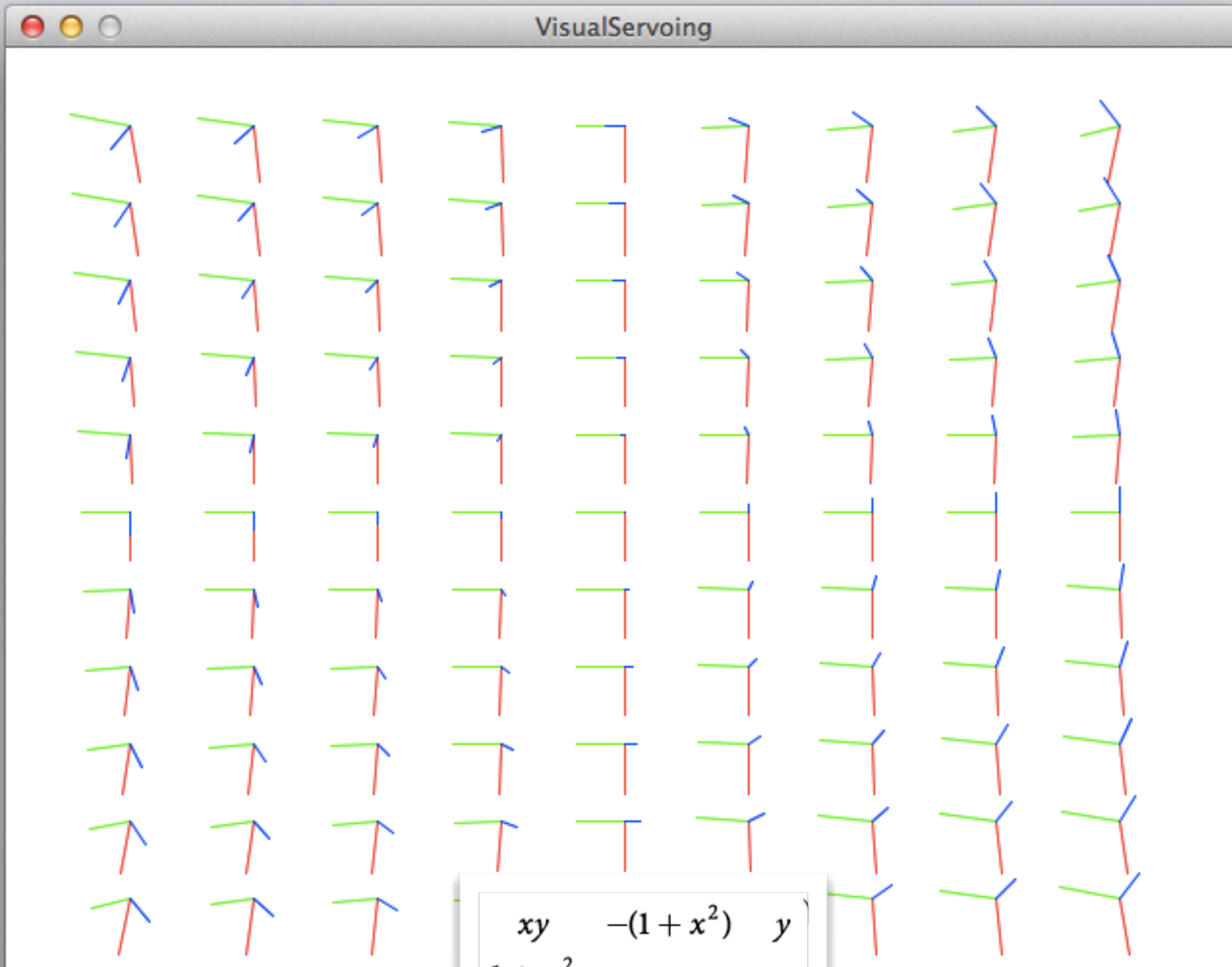
$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

LINEAR PART

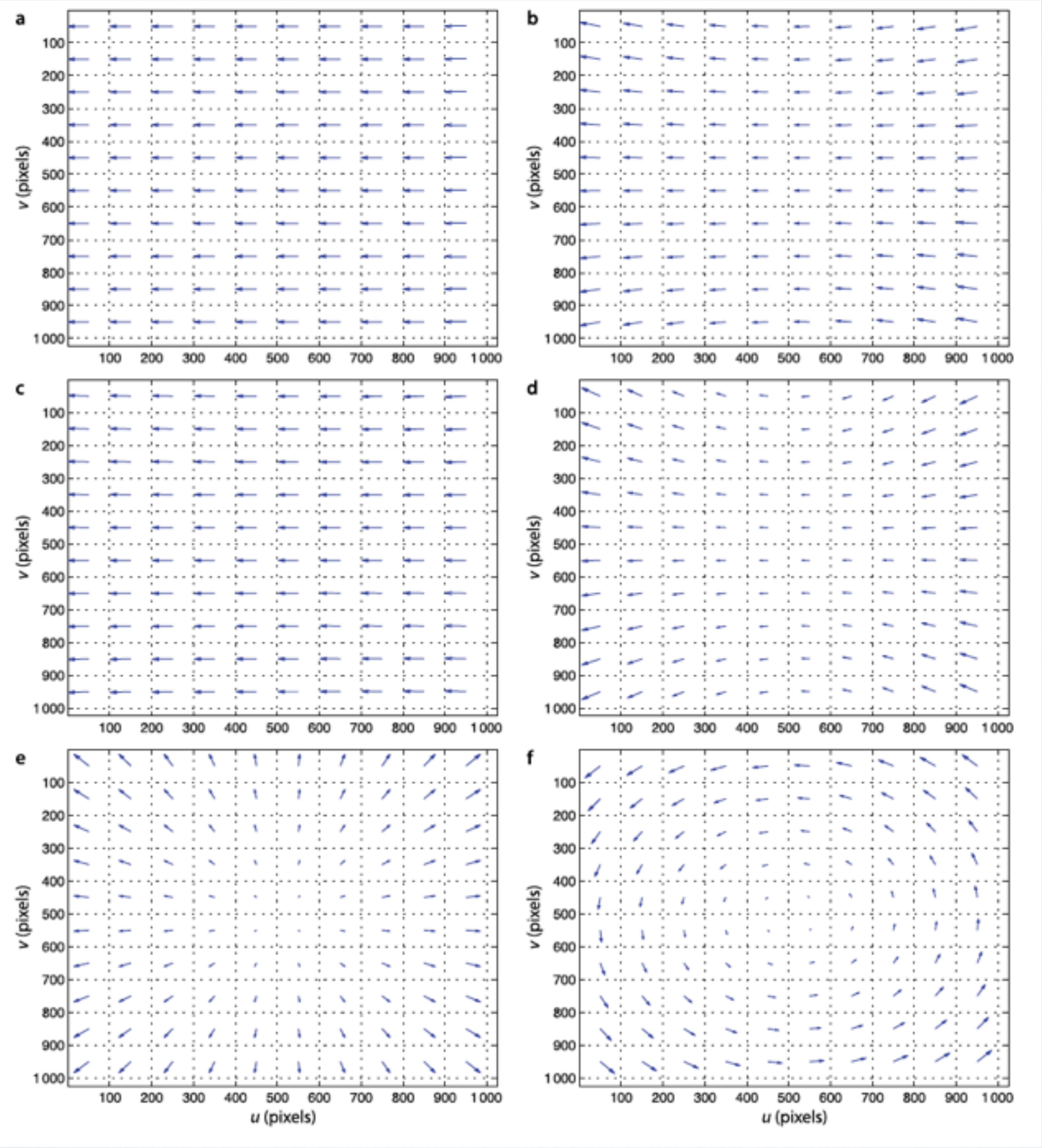


$$\begin{matrix} -\frac{1}{Z} & 0 & \frac{x}{Z} \\ 0 & -\frac{1}{Z} & \frac{y}{Z} \end{matrix}$$

ANGULAR PART



$$\begin{pmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{pmatrix}$$



Corke Figure

JACOBIAN RANK

- 2*6 matrix, rank 2
- Null-space = 4-dim
- Corresponds to 4D space of motions that leave image of a point invariant

We can consider the motion of two points by stacking their Jacobians

$$\begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \end{pmatrix} = \begin{pmatrix} J_{p_1} \\ J_{p_2} \end{pmatrix} \nu$$

For three points

$$\begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \end{pmatrix} = \begin{pmatrix} J_{p_1} \\ J_{p_2} \\ J_{p_3} \end{pmatrix} \nu$$

CONTROLLING FEATURE MOTION

- For three features: invert !
- For more: pseudo-inverse

$$\boldsymbol{\nu} = \begin{pmatrix} \boldsymbol{J}_{p_1} \\ \boldsymbol{J}_{p_2} \\ \boldsymbol{J}_{p_3} \end{pmatrix}^{-1} \begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \end{pmatrix}$$

$$\boldsymbol{\nu} = \lambda \begin{pmatrix} \boldsymbol{J}_{p_1} \\ \boldsymbol{J}_{p_2} \\ \boldsymbol{J}_{p_3} \end{pmatrix}^{-1} (\boldsymbol{p}^* - \boldsymbol{p})$$

$$\boldsymbol{\nu} = \lambda \begin{pmatrix} \boldsymbol{J}_1 \\ \vdots \\ \boldsymbol{J}_N \end{pmatrix}^{+} (\boldsymbol{p}^* - \boldsymbol{p})$$