Lectorial A3: Camera and image motion

Peter Corke

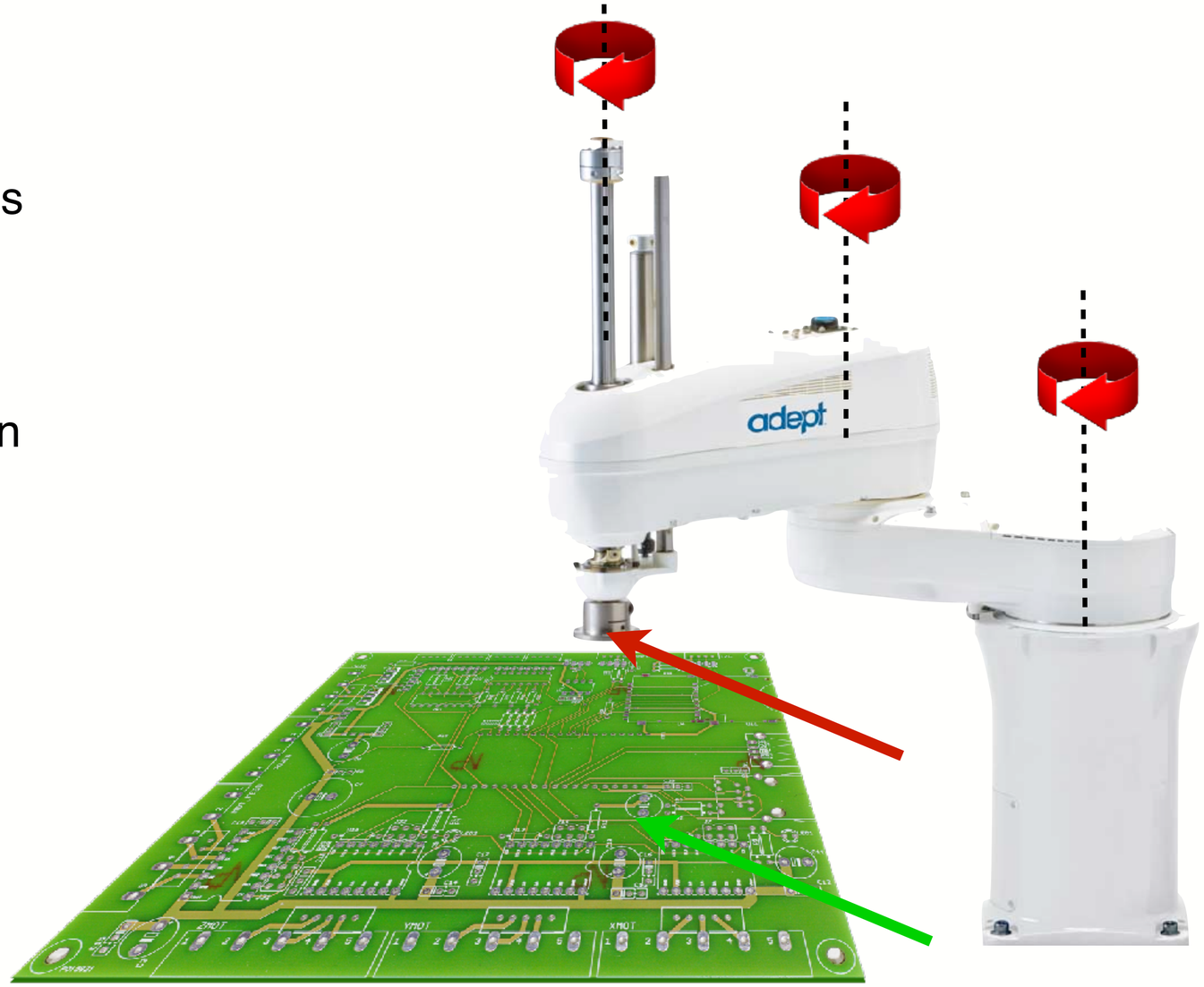
Section 1 Motivation


Staub

A precision robotic task

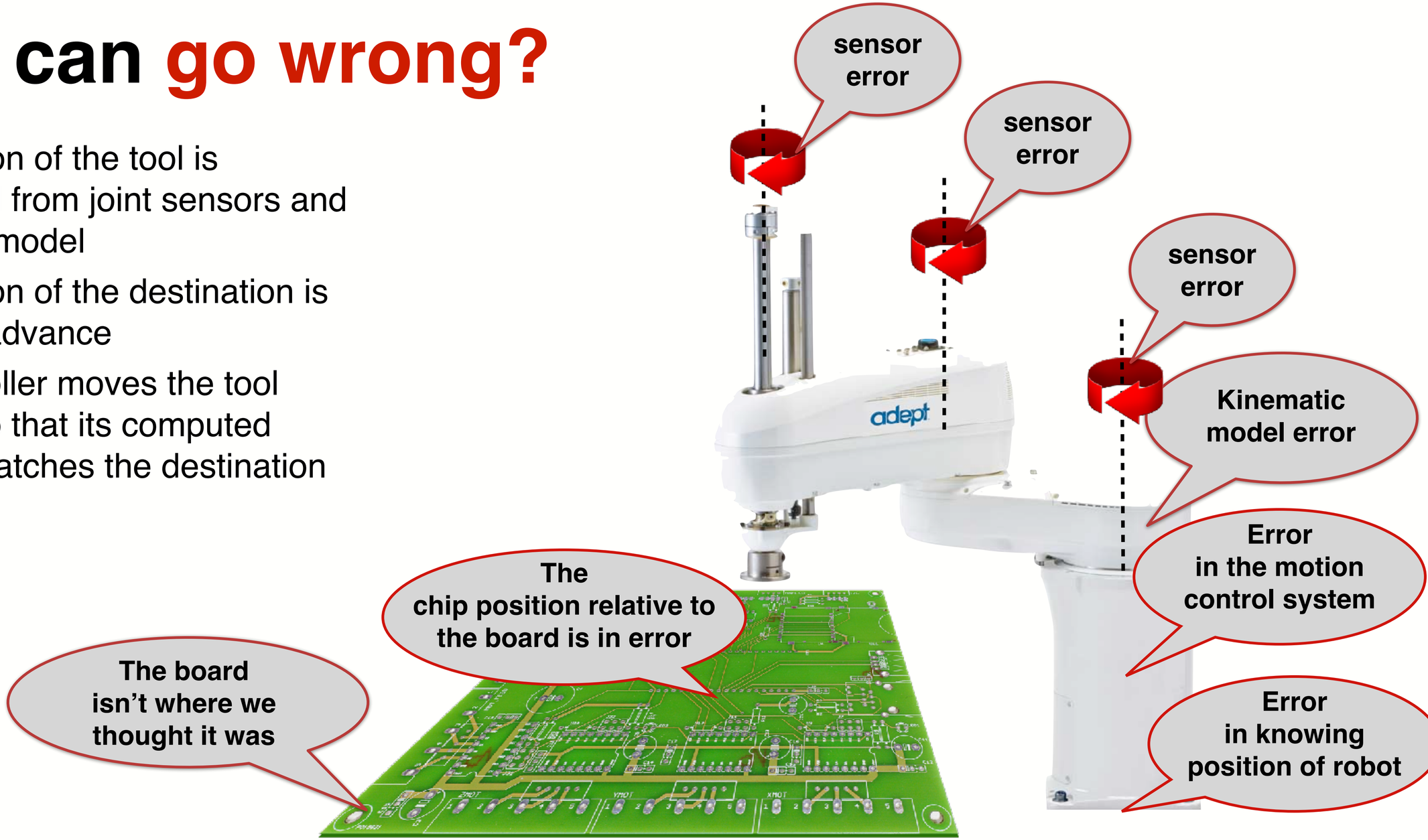
How does the robot know where the chip is?

- The position of the tool is **computed** from joint sensors
- The position of the destination is known in advance
- The controller moves the tool position so that its computed position matches the destination



What can go wrong?

- The position of the tool is **computed** from joint sensors and kinematic model
- The position of the destination is known in advance
- The controller moves the tool position so that its computed position matches the destination



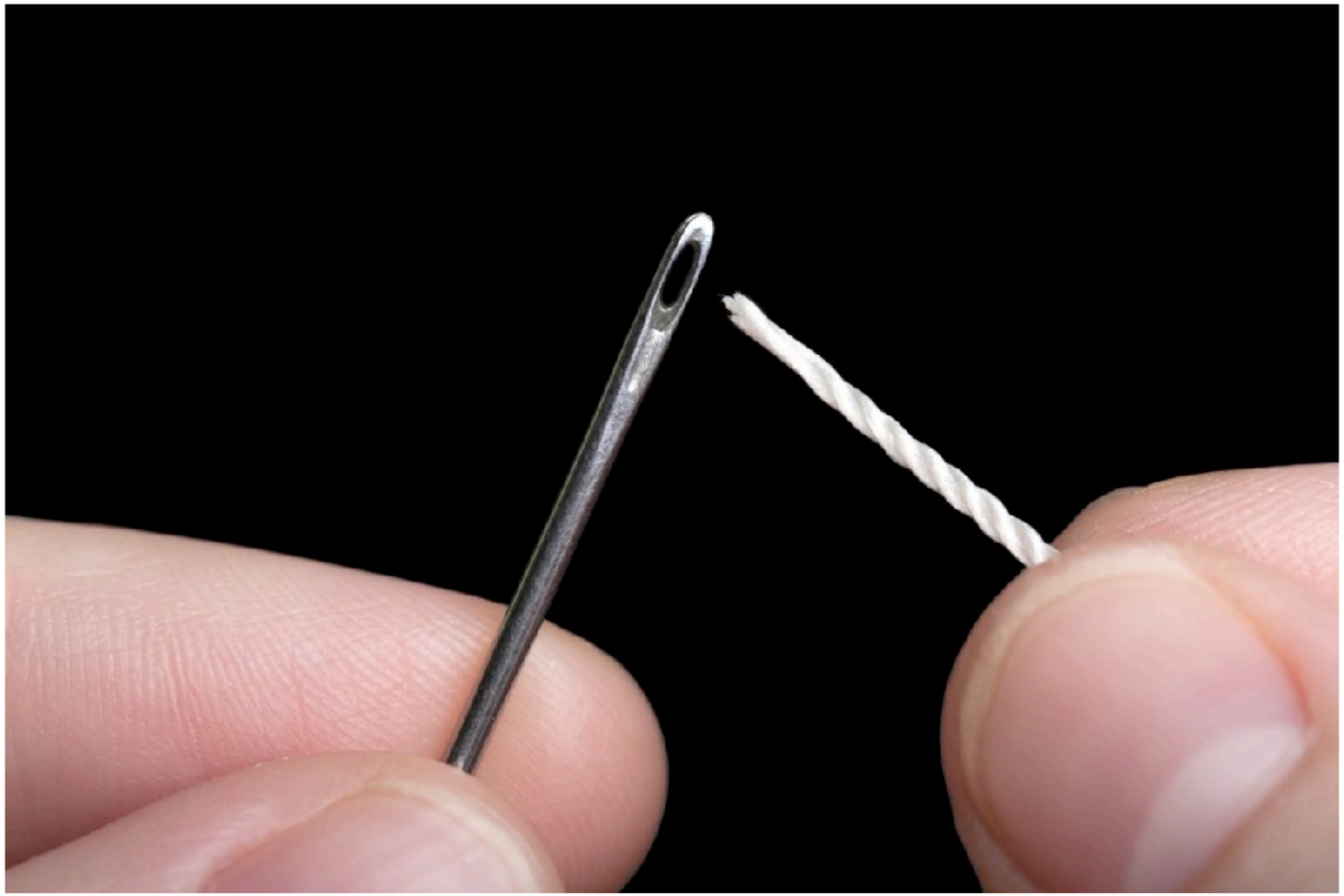
Cost of **precision**

- Accurate kinematic model requires
  - Precision in manufacture
  - No static/dynamic deformation
    - stiff and heavy links
- To move quickly and accurately we need
  - Precise encoders
  - Big motors
- But we still need to know where the goal object is…

The bottom line

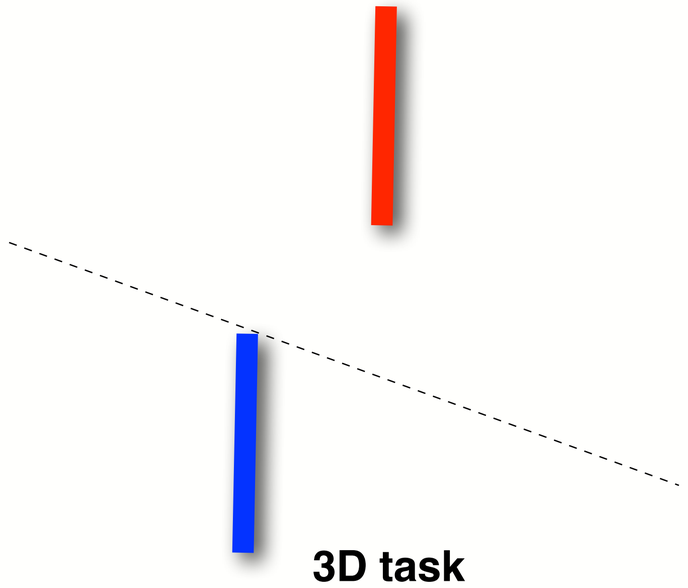
- Robot's are incredibly precise
- But for robots to do useful things
  - ➥ the object to be manipulated must be precisely positioned
    - – requires jigs and fixtures
  - ➥ the robot must know the position of the object
    - – often requires "teaching"

Contrast with how we do it

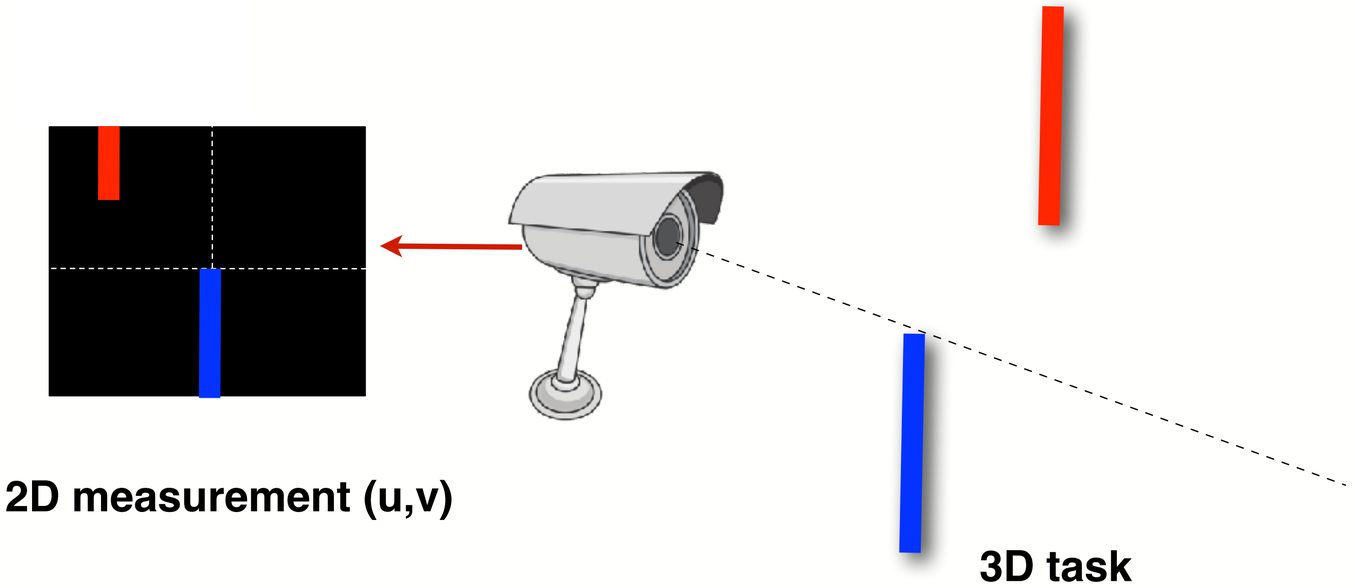


Section 2 Specifying tasks in the image plane

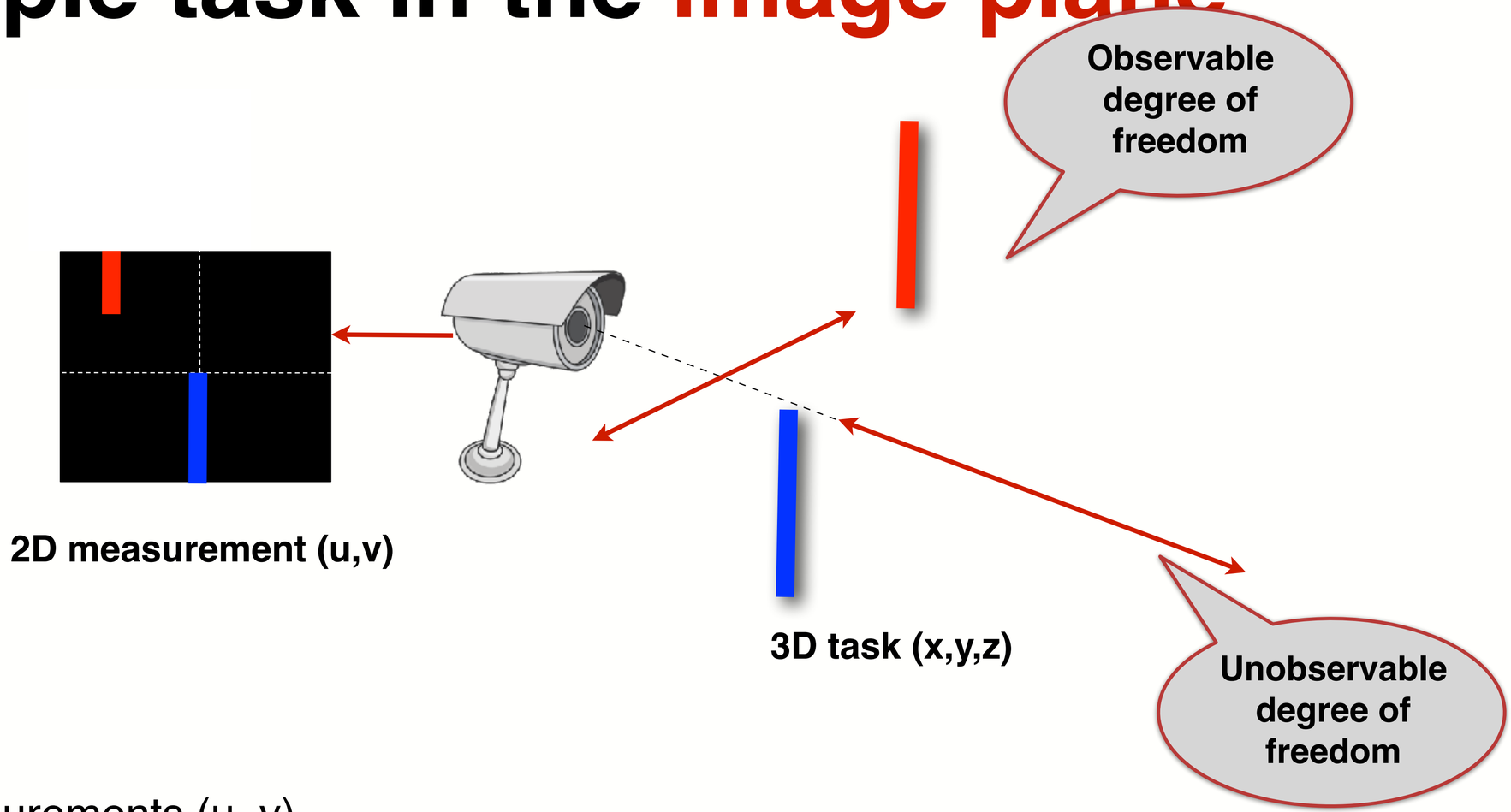
A simple 3D task



Simple task in the image plane

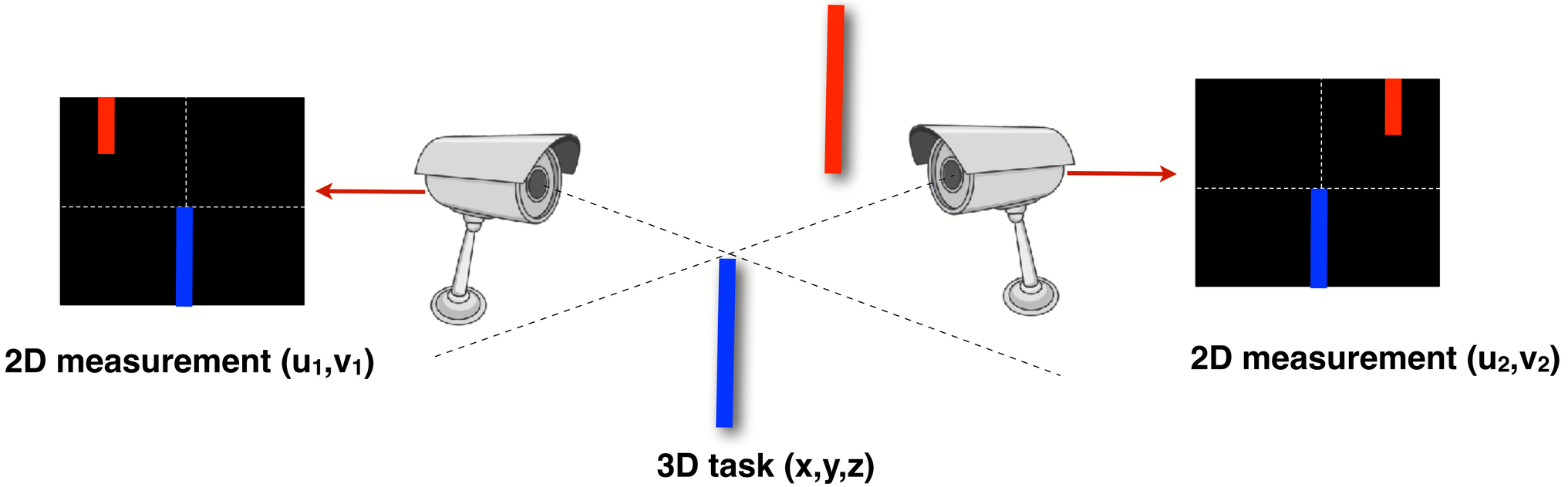


Simple task in the image plane



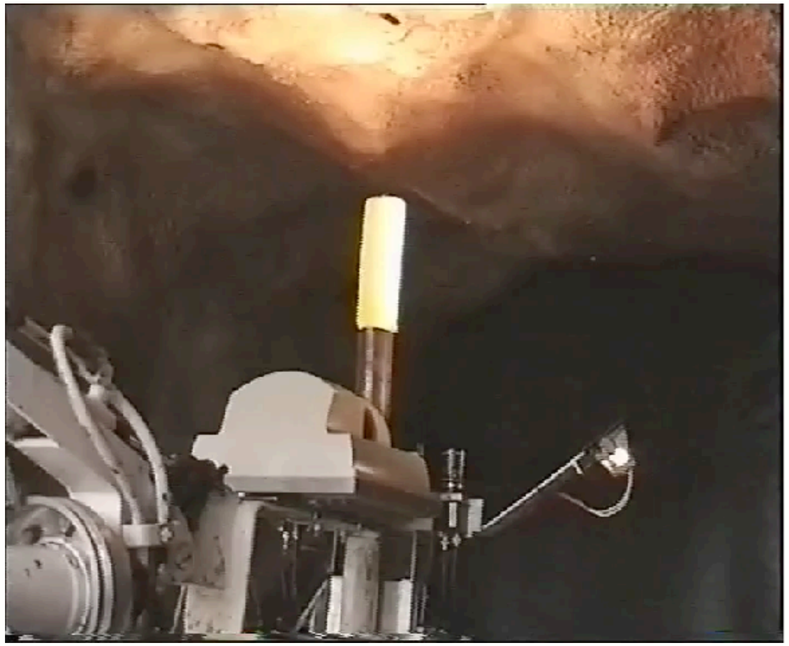
- 2 measurements (u, v)
- **Insufficient** for a task with 3 degrees of freedom (DOF)

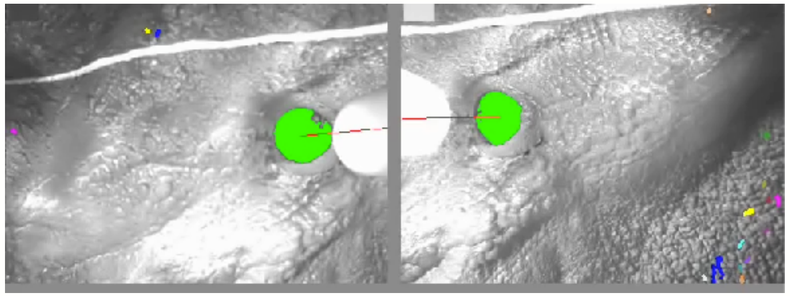
Simple task in the image plane



- 4 measurements (u_1,v_1, u_2,v_2)
- Sufficient for a task with 3 degrees of freedom (DOF)

Dual-camera visual servoing



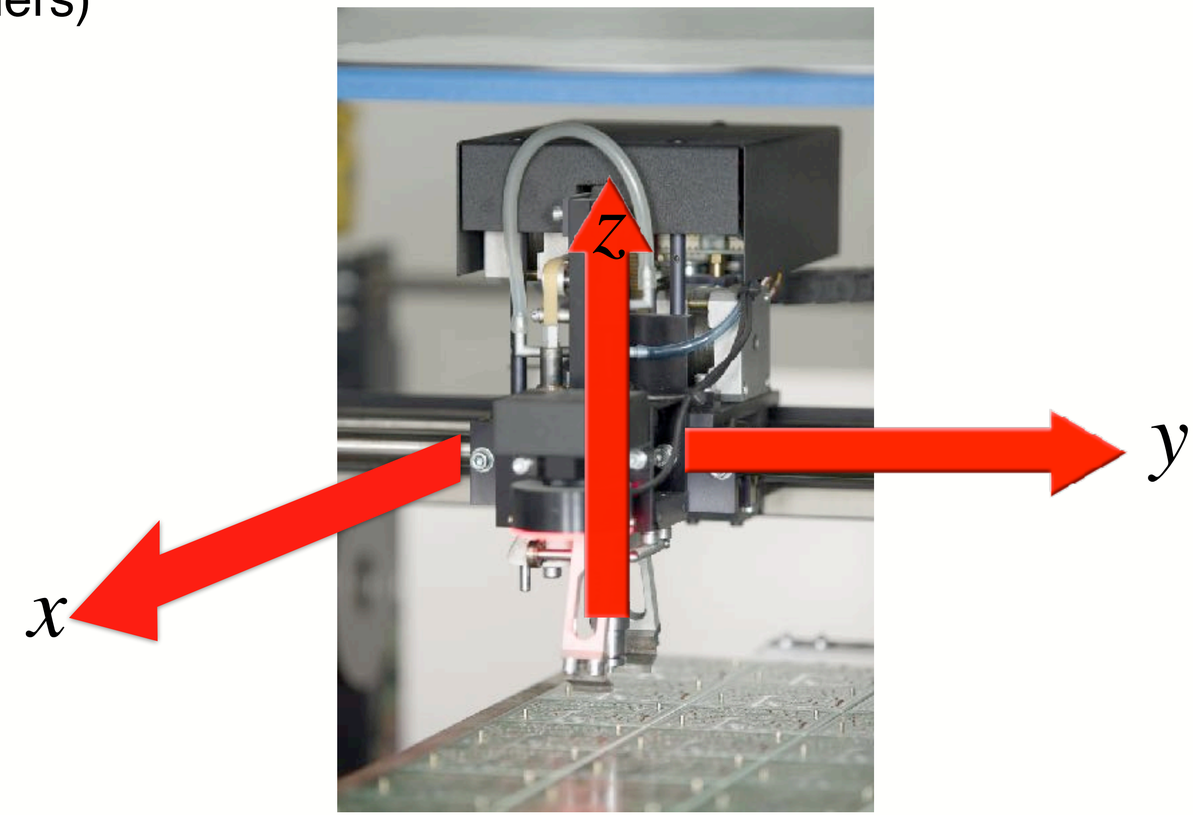


Dual-camera image-based visual servo

all images & video
CSIRO | used with permission

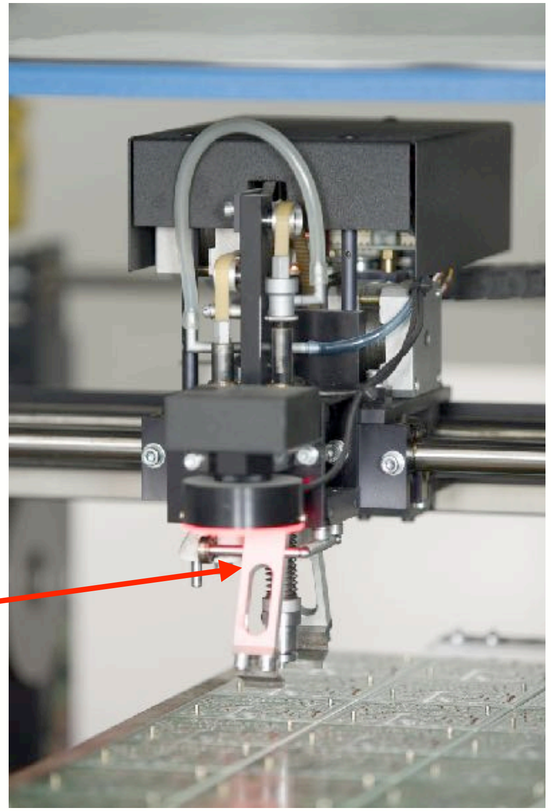
Conventional robot control

- We use information from joint sensors (encoders)
- Plus a kinematic model
  - ➥ To **infer** the end-point pose
- To control the motion of a mechanism



Vision-based control

- The use of visual information
  - ➥ from one or more cameras
  - ➥ making many measurements per second
- To directly observe the error between the tool and the object
- To control the motion of a mechanism



Vision-based vs conventional control

Commonly called

visual servoing

- Direct observation of the
  - ➥ tool
  - ➥ object
  - ➥ error

- Indirect observation
- + kinematic model
- Need to be told the position of the object

Some possible uses of visual servoing

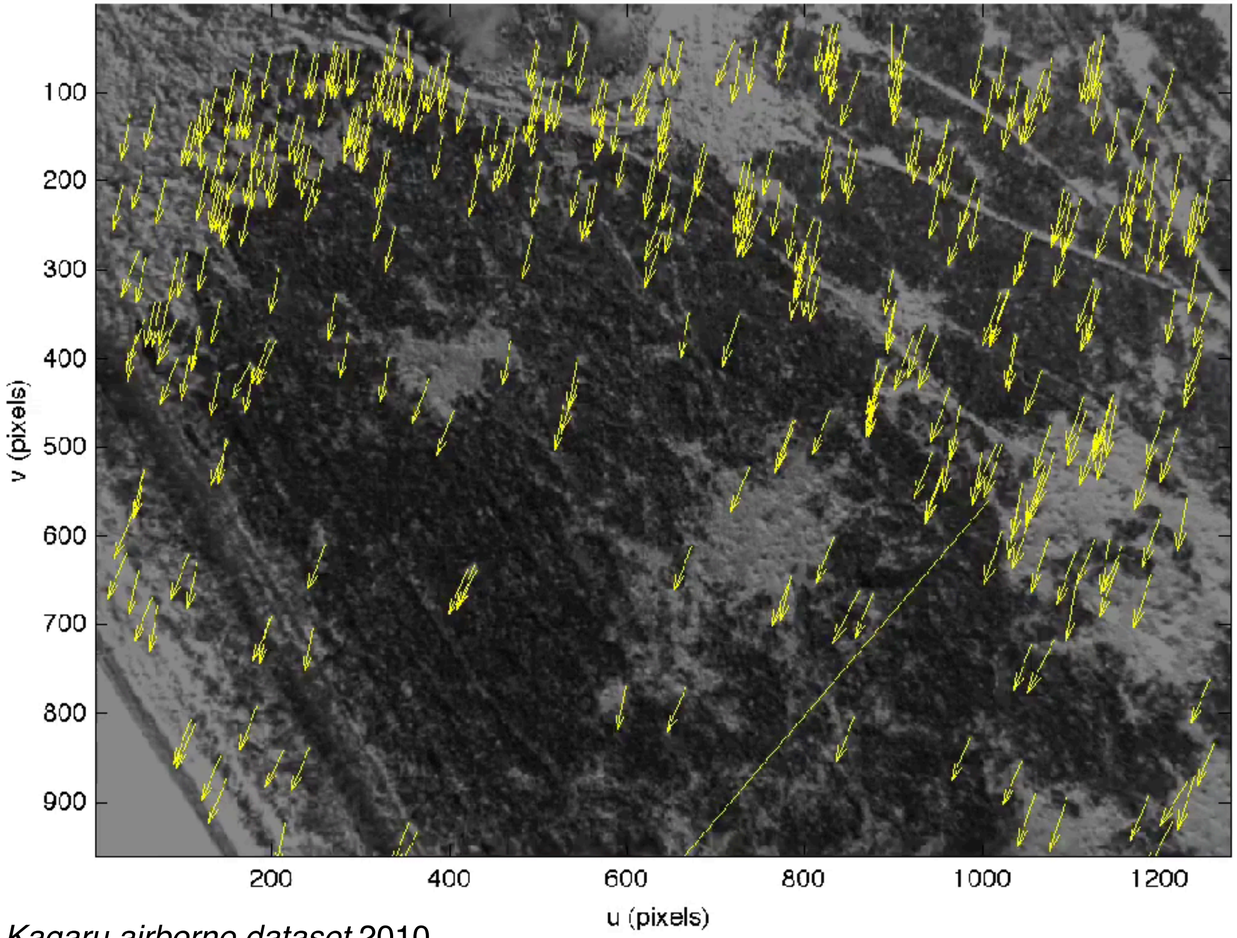
- Use the visual information to control the motion of a mechanism to complete a *task*, such as:
  - ➥ Manufacturing:
    - – grabbing parts moving on conveyor belts
    - – grasping swinging objects
  - ➥ Mating a refuelling nozzle (car, aircraft, space shuttle)

Some possible uses of visual servoing

- Use the visual information to control the motion of a mechanism to complete a *task*, such as:
  - Manufacturing:
    - grasping swinging objects
    - grabbing parts moving on conveyor belts
  - Mating a refuelling nozzle (car, aircraft, space shuttle)
  - Fruit picking
  - Aircraft landing by observing the runway
  - Submarine station keeping
  - Submarine following an underwater pipeline
  - Juggling, etc.

Section 3 View from a moving camera

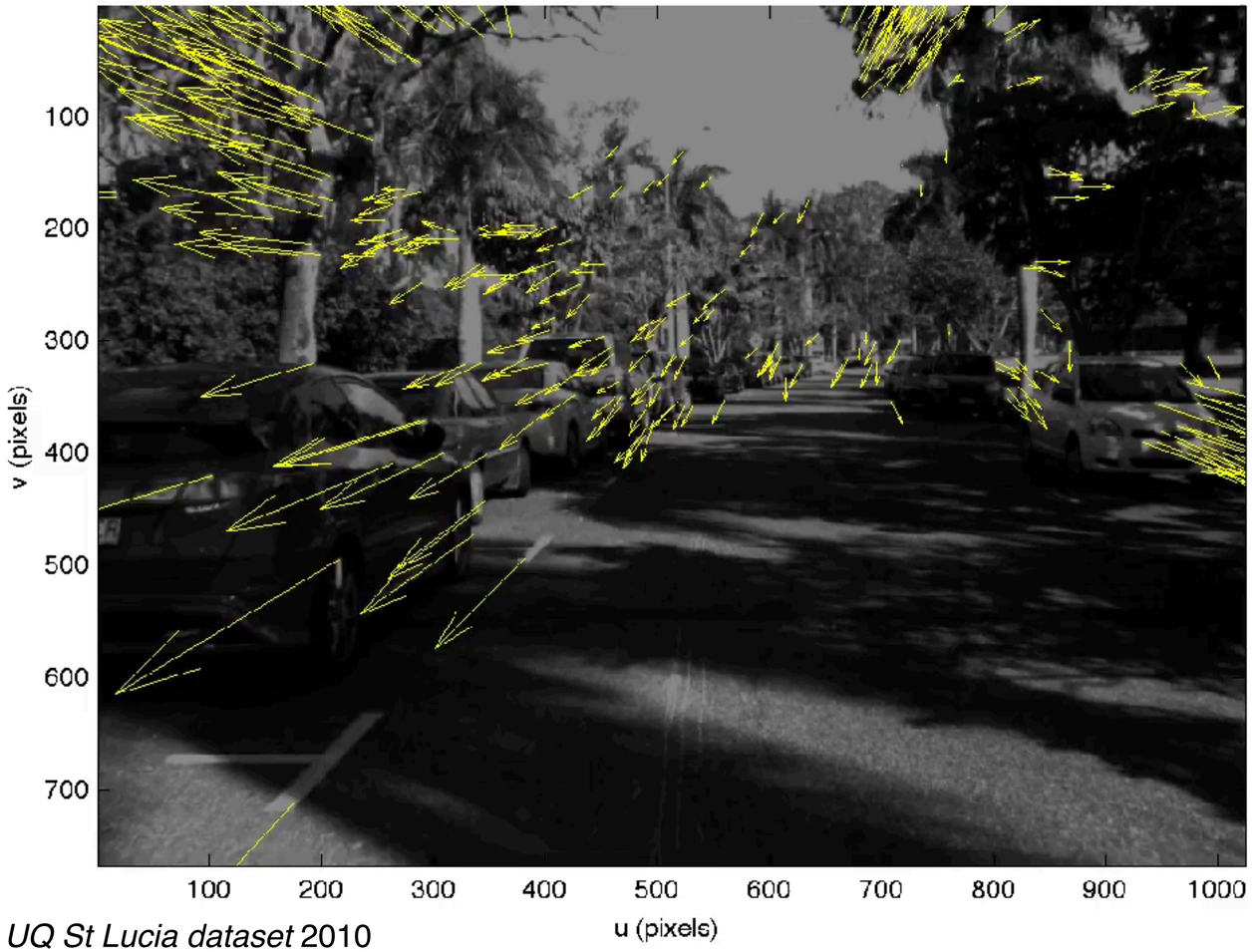
Real world optical flow



Kagaru airborne dataset 2010

Michael Warren | used with permission

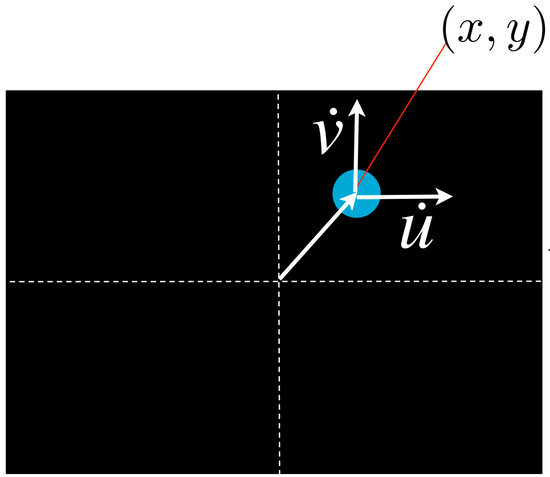
Real world optical flow



UQ St Lucia dataset 2010

Michael Warren I used with permission

Section 4 The image Jacobian



normalized/retinal coordinates

$\mathbf{P} = (X, Y, Z)$

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

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

$$X = xZ, Y = yZ$$

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

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

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

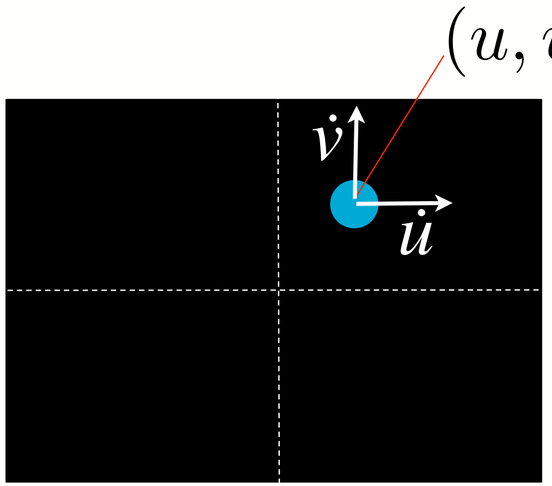
$$\bar{u} = \hat{f}x, \bar{v} = \hat{f}y$$

$$\hat{f} = \frac{f}{\rho}$$

$$\begin{pmatrix}\dot{\bar{u}} \\ \dot{\bar{v}}\end{pmatrix} = \begin{pmatrix}-\frac{\hat{f}}{Z} & 0 & \frac{\bar{u}}{Z} & \frac{\bar{u}\bar{v}}{\hat{f}} & -(\hat{f} + \frac{\bar{u}^2}{\hat{f}}) & \bar{v} \\ 0 & -\frac{\hat{f}}{Z} & \frac{\bar{v}}{Z} & \hat{f} + \frac{\bar{v}^2}{\hat{f}} & -\frac{\bar{u}\bar{v}}{\hat{f}} & -\bar{u}\end{pmatrix} \begin{pmatrix}v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z\end{pmatrix}$$

pixel coordinates relative to the principal point

Optical flow for translation



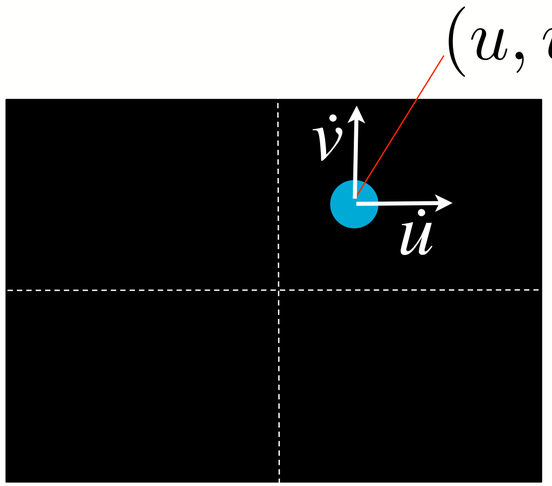
pixel velocity

image Jacobian

camera velocity

$$\begin{pmatrix} \dot{\bar{u}} \\ \dot{\bar{v}} \end{pmatrix} = \begin{pmatrix} -\frac{\hat{f}}{Z} & 0 & \frac{\bar{u}}{Z} & \frac{\bar{u}\bar{v}}{\hat{f}} & -(\hat{f} + \frac{\bar{u}^2}{\hat{f}}) & \bar{v} \\ 0 & -\frac{\hat{f}}{Z} & \frac{\bar{v}}{Z} & \hat{f} + \frac{\bar{v}^2}{\hat{f}} & -\frac{\bar{u}\bar{v}}{\hat{f}} & -\bar{u} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

Optical flow for rotation



pixel velocity

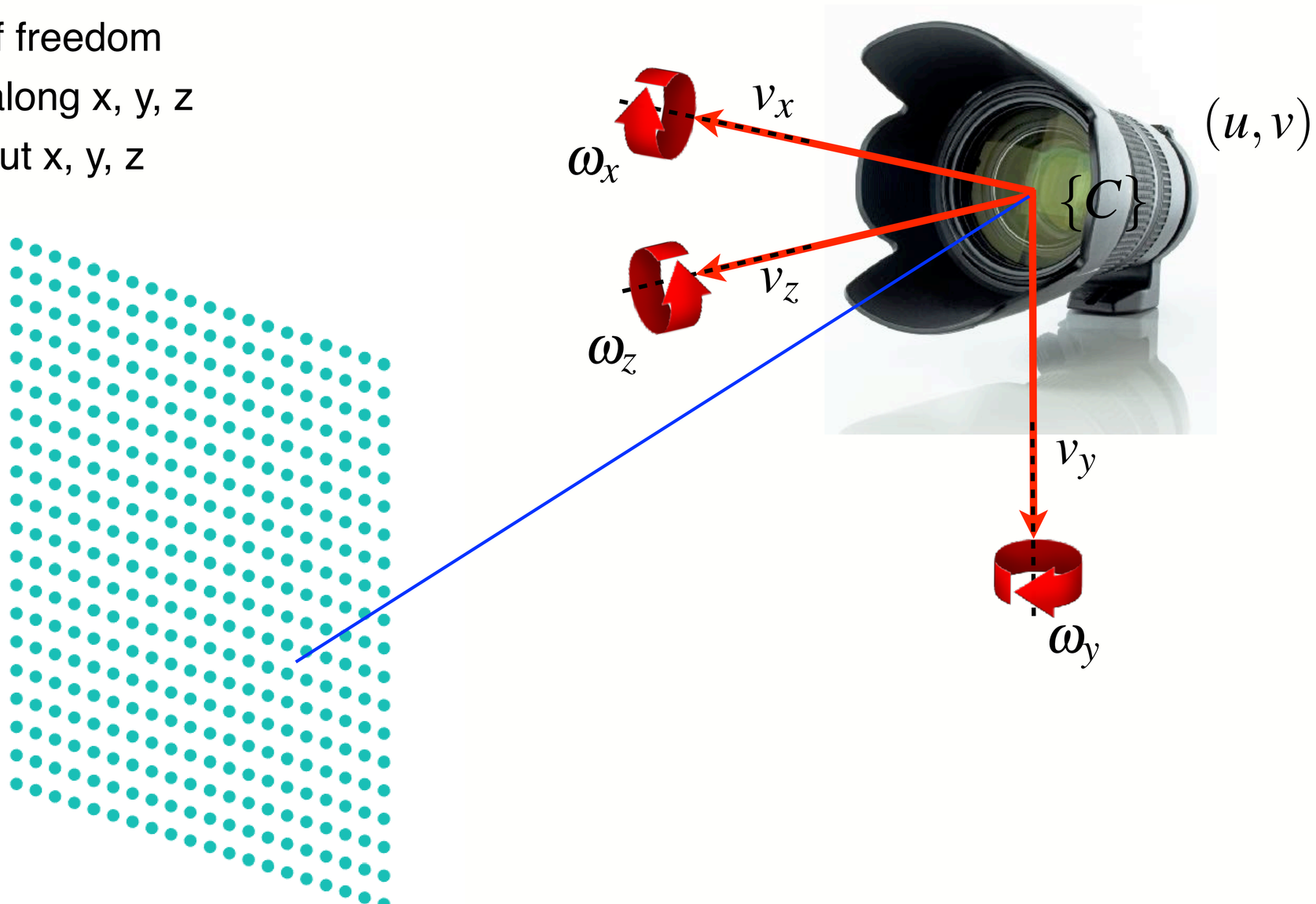
image Jacobian

camera velocity

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} -\frac{\hat{f}}{Z} & 0 & \frac{\bar{u}}{Z} & \frac{\bar{u}\bar{v}}{\hat{f}} & -(\hat{f} + \frac{\bar{u}^2}{\hat{f}}) & \bar{v} \\ 0 & -\frac{\hat{f}}{Z} & \frac{\bar{v}}{Z} & \hat{f} + \frac{\bar{v}^2}{\hat{f}} & -\frac{\bar{u}\bar{v}}{\hat{f}} & -\bar{u} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

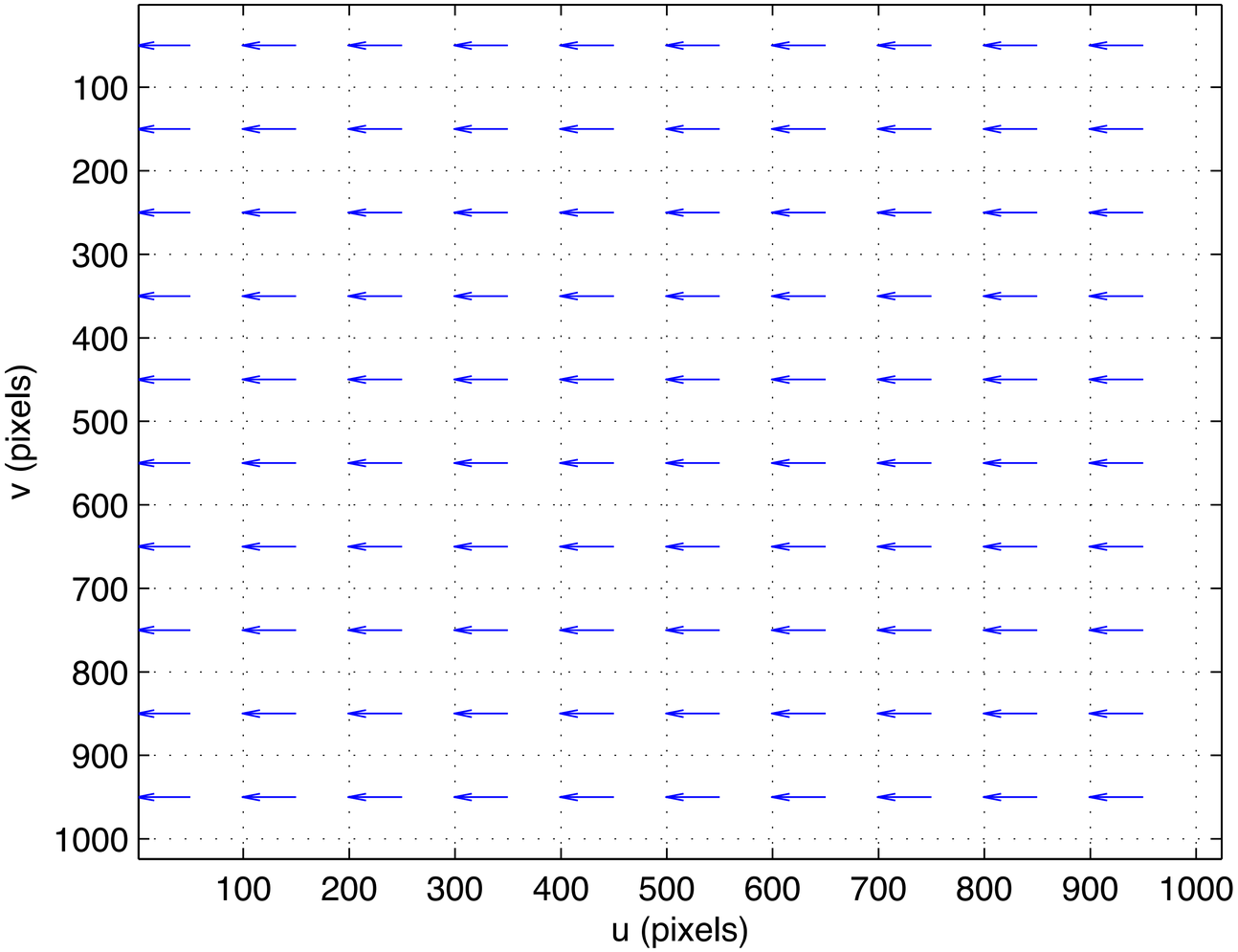
Motion of a camera

- 6 degrees of freedom
  - translate along x, y, z
  - rotate about x, y, z



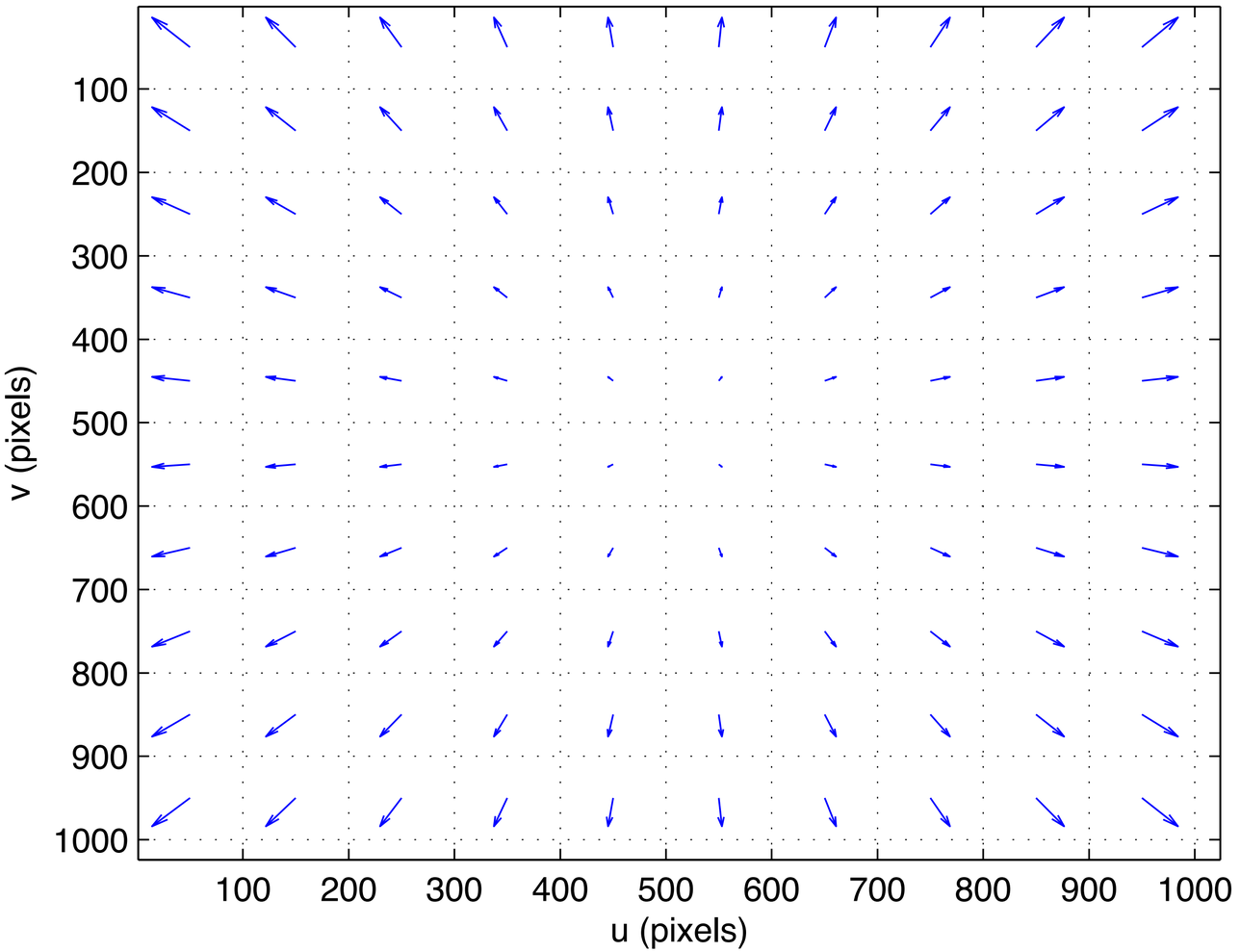
Optical flow patterns

v_x



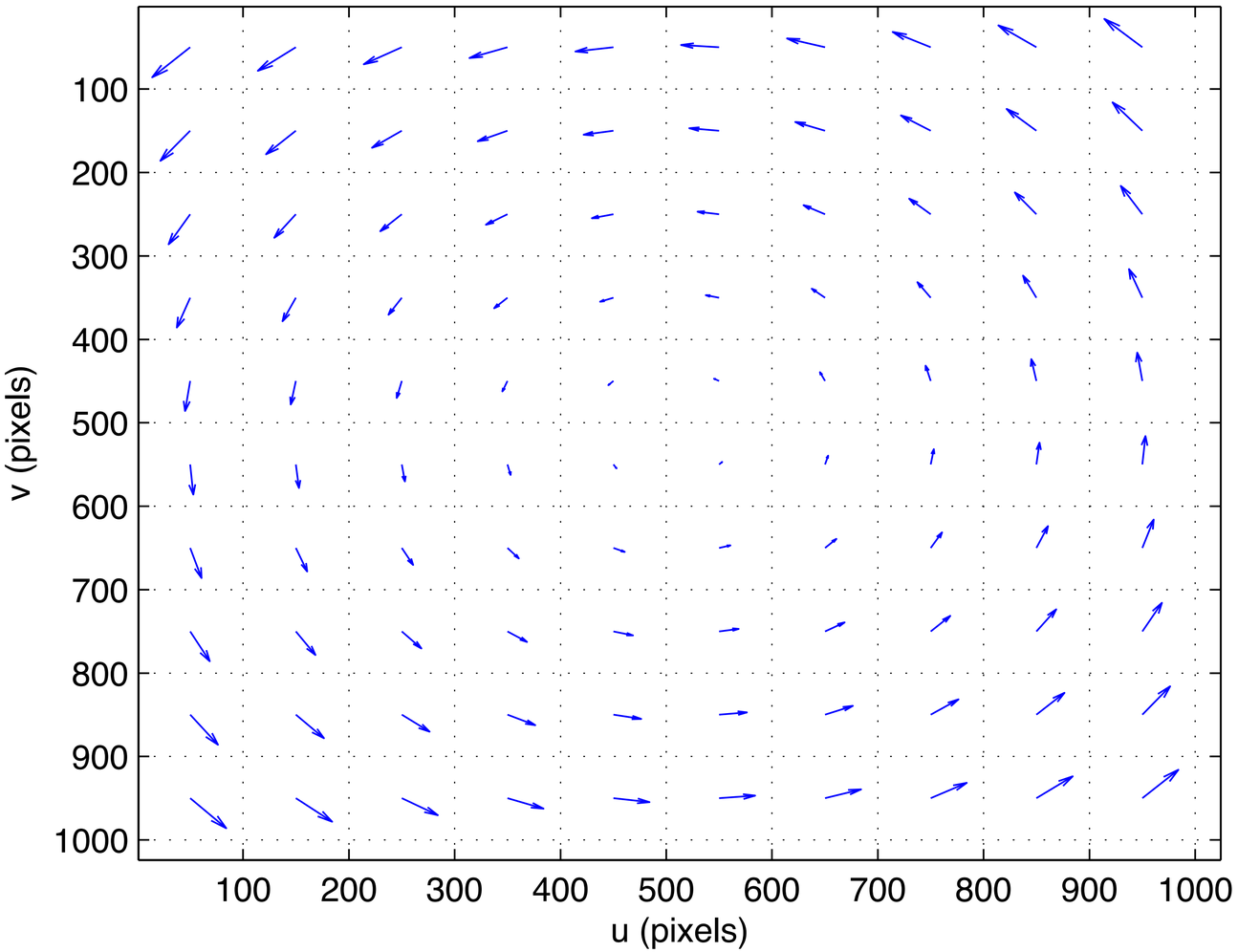
Optical flow patterns

v_z



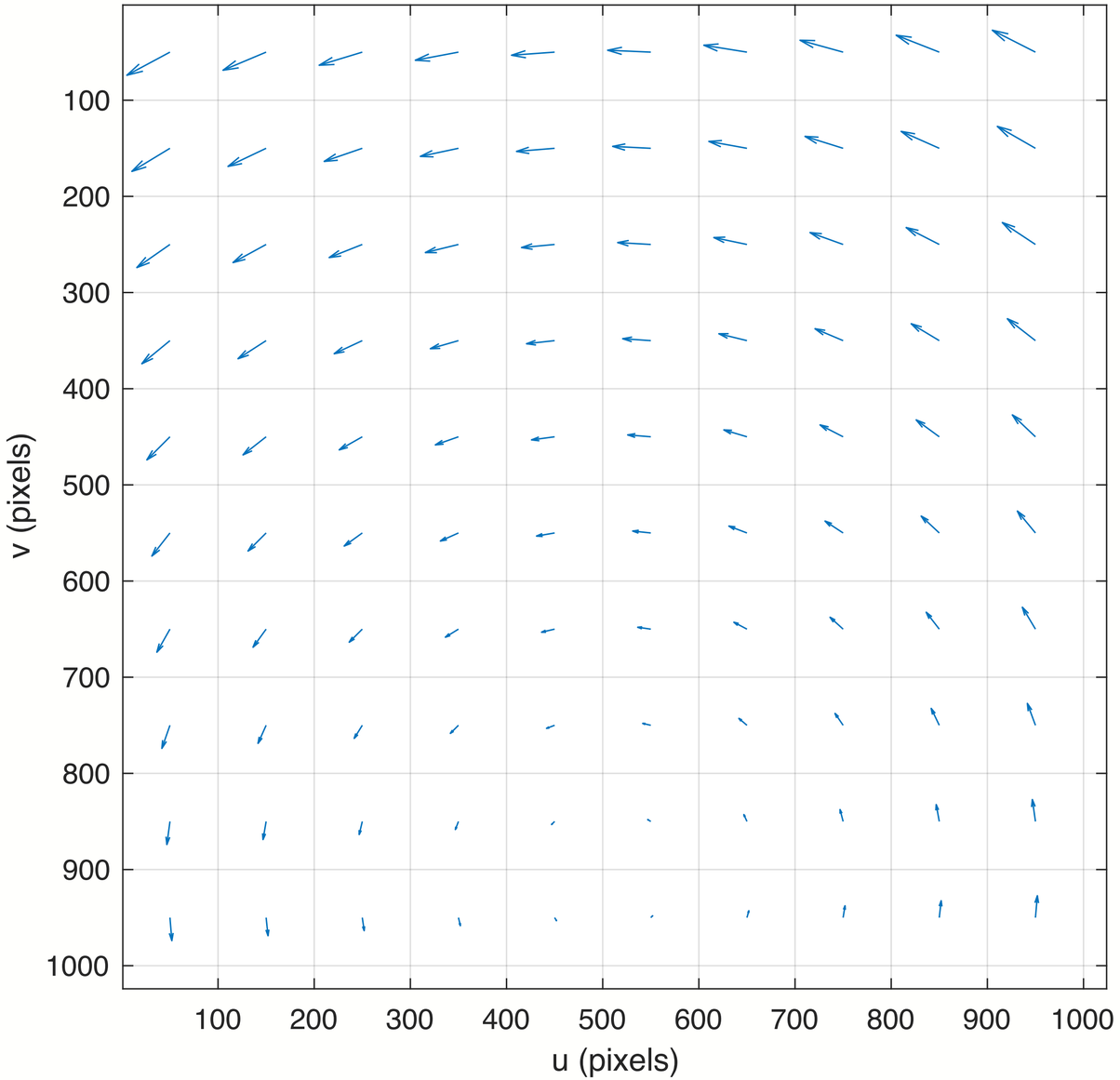
Optical flow patterns

ω_z



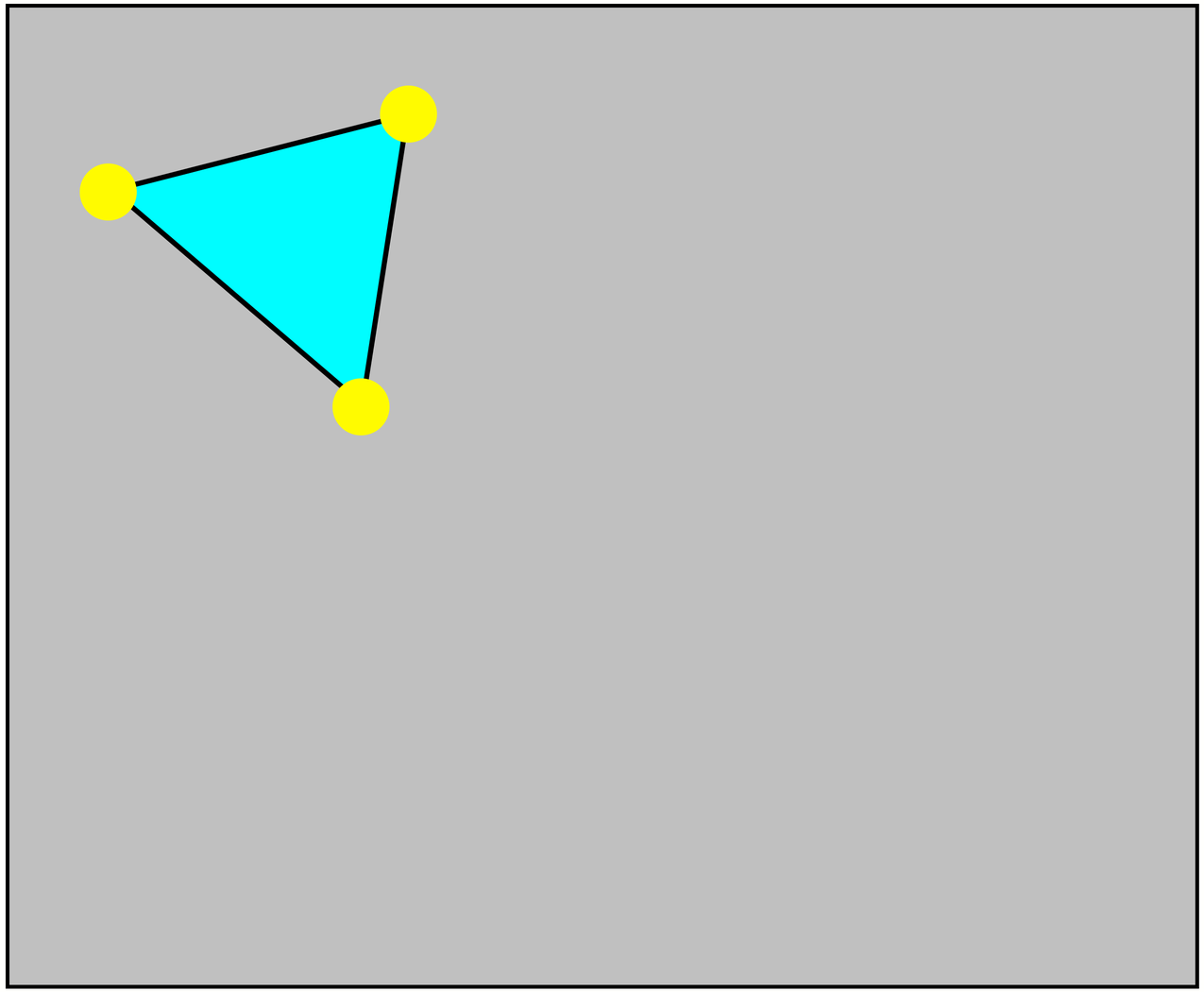
Optical flow patterns

$$v_x + \omega_z$$



Section 6 Image-based visual servoing

Initial view



Desired view

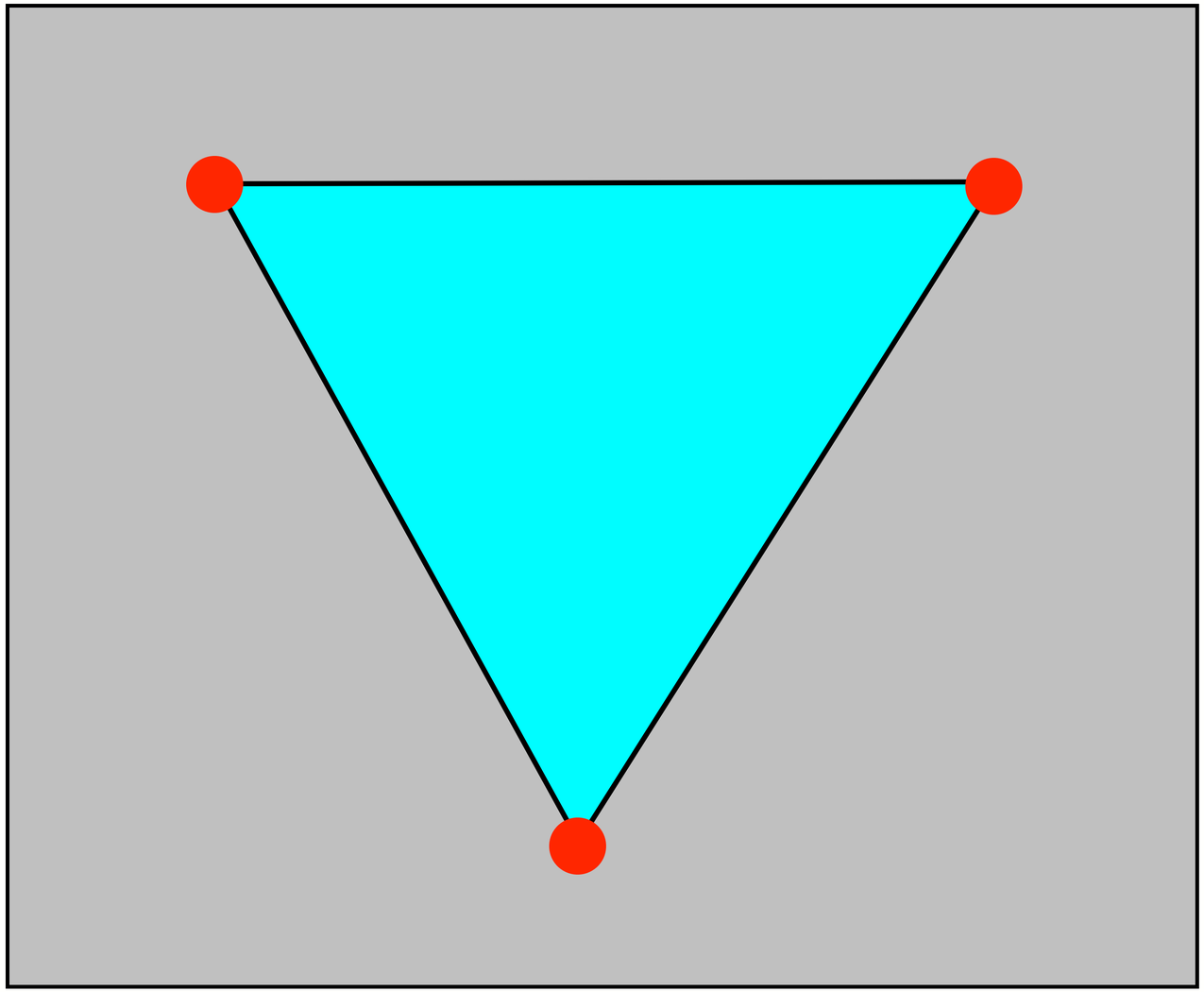


Image plane motion

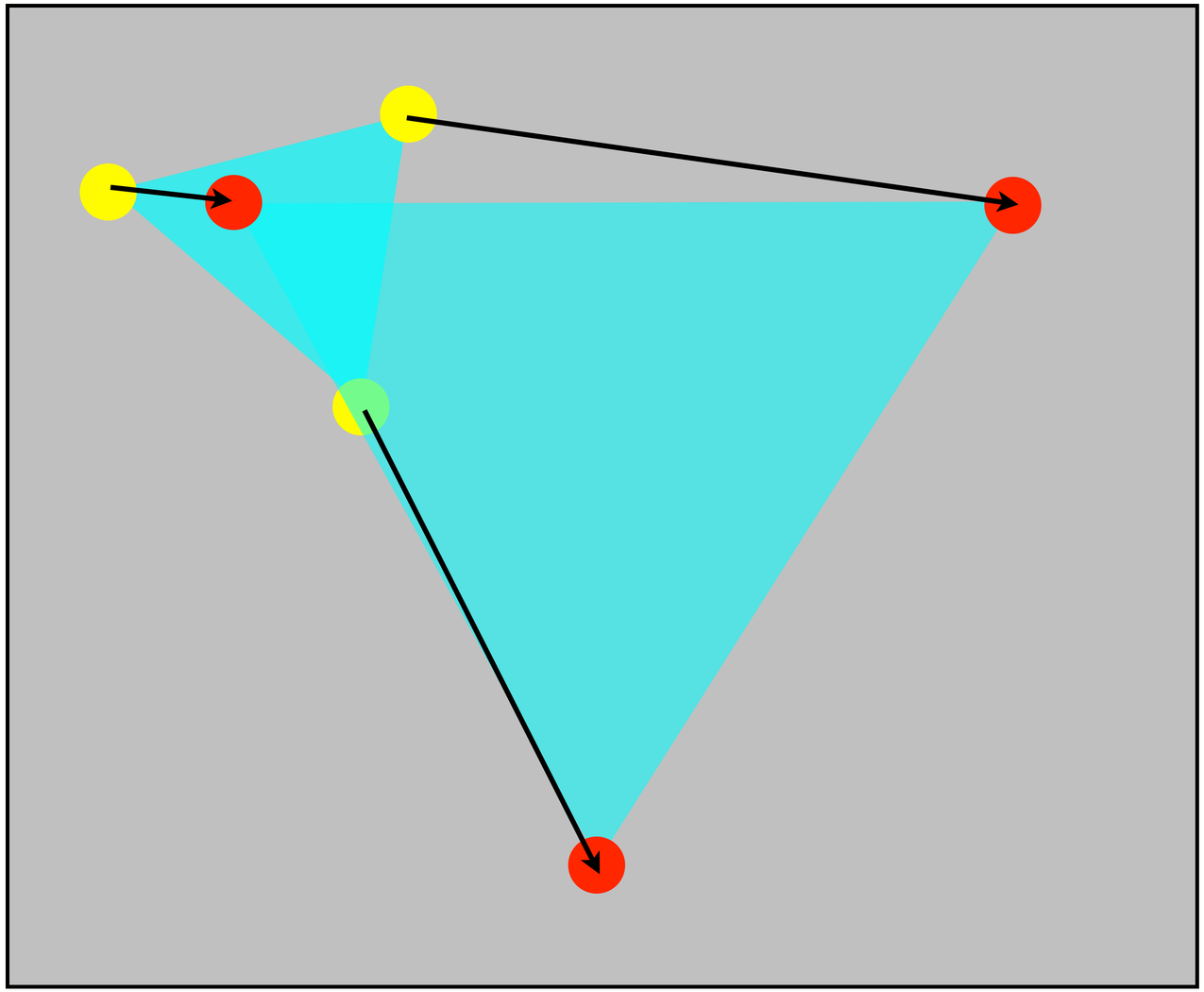
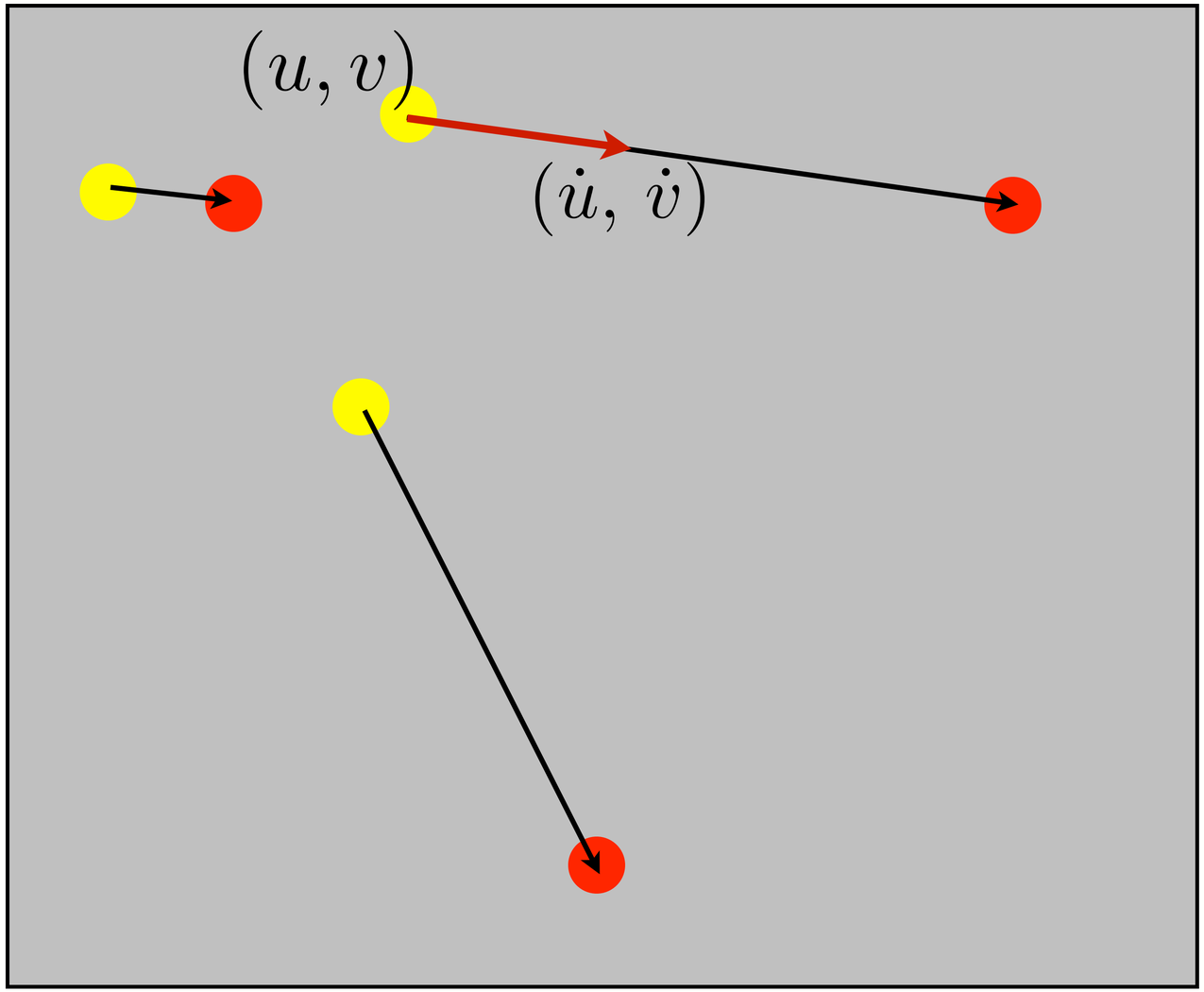


Image plane motion



Motion for three points

pixel velocity

image Jacobian

camera velocity

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \underbrace{\begin{pmatrix} -\hat{f}/Z & 0 & u/Z & uv/\hat{f} & -(\hat{f}+u^2/\hat{f}) & v \\ 0 & -\hat{f}/Z & v/Z & \hat{f}+v^2/\hat{f} & -uv/\hat{f} & -u \end{pmatrix}}_{\mathbf{J}_p(u,v,Z)} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

u, v here are assumed to be relative to the principal point, earlier I used a bar

$$\begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{J}_p(u_1, v_1, Z_1) \\ \mathbf{J}_p(u_2, v_2, Z_2) \\ \mathbf{J}_p(u_3, v_3, Z_3) \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

Motion for three points

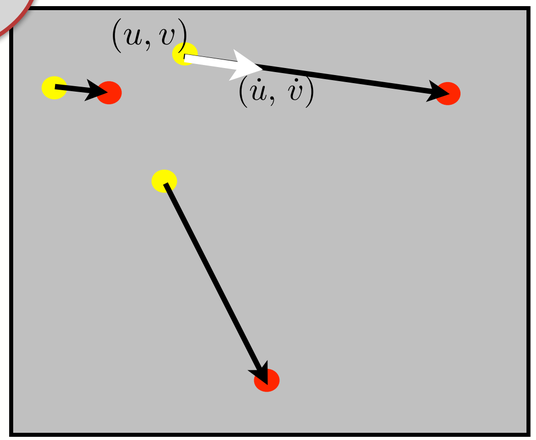
6 x 6

$$\begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{J}_p(u_1, v_1, Z_1) \\ \mathbf{J}_p(u_2, v_2, Z_2) \\ \mathbf{J}_p(u_3, v_3, Z_3) \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

camera velocity

$$\begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} \mathbf{J}_p(u_1, v_1, Z_1) \\ \mathbf{J}_p(u_2, v_2, Z_2) \\ \mathbf{J}_p(u_3, v_3, Z_3) \end{pmatrix}^{-1} \begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \end{pmatrix}$$

pixel velocity



Desired pixel velocity

desired pixel

desired point

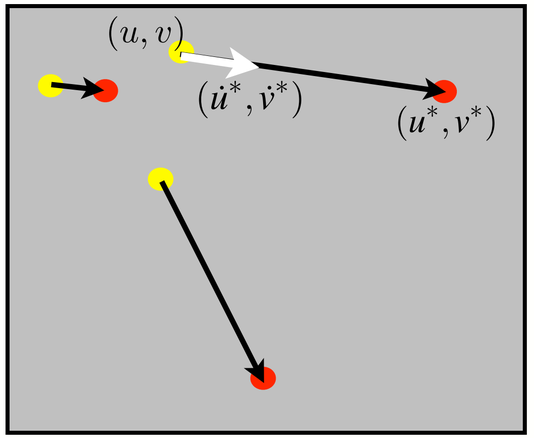
current point

$$\begin{pmatrix} \dot{u}_i^* \\ \dot{v}_i^* \end{pmatrix} = \lambda \left\{ \begin{pmatrix} u_i^* \\ v_i^* \end{pmatrix} - \begin{pmatrix} u_i \\ v_i \end{pmatrix} \right\}$$

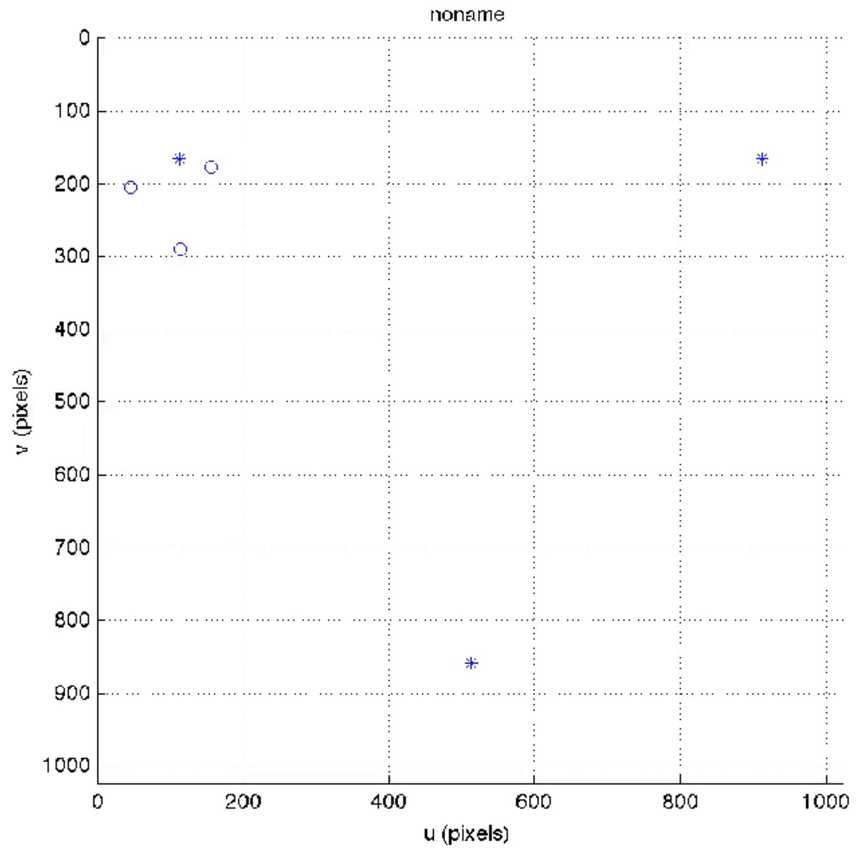
$$\dot{\boldsymbol{p}}_i^* = \lambda (\boldsymbol{p}_i^* - \boldsymbol{p}_i)$$

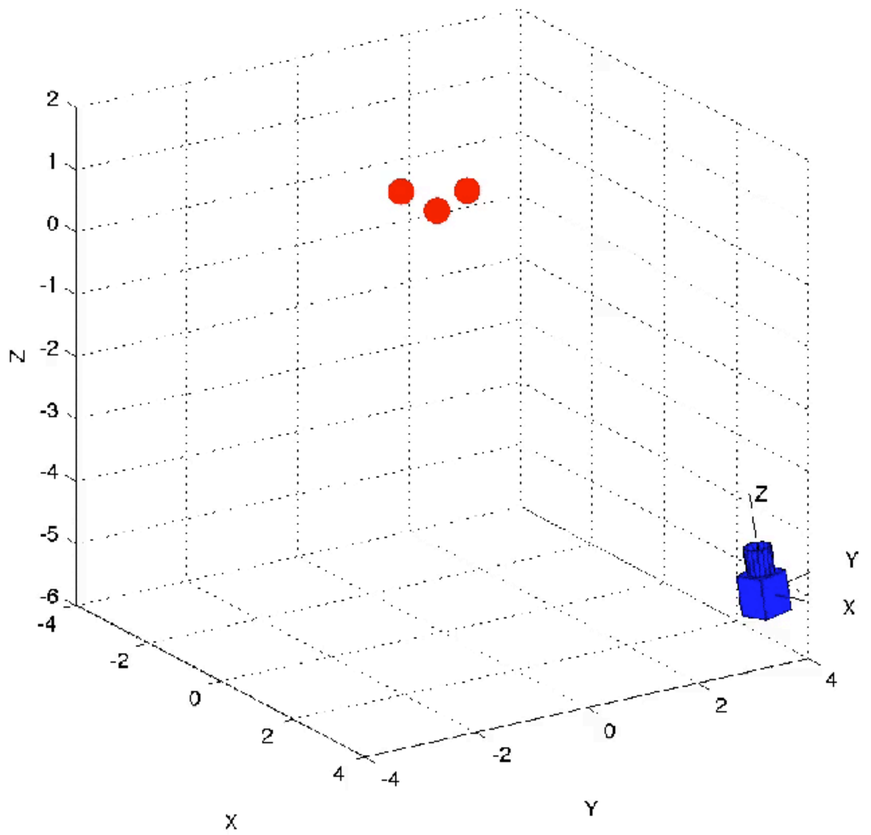
$$\boldsymbol{p}_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix}$$

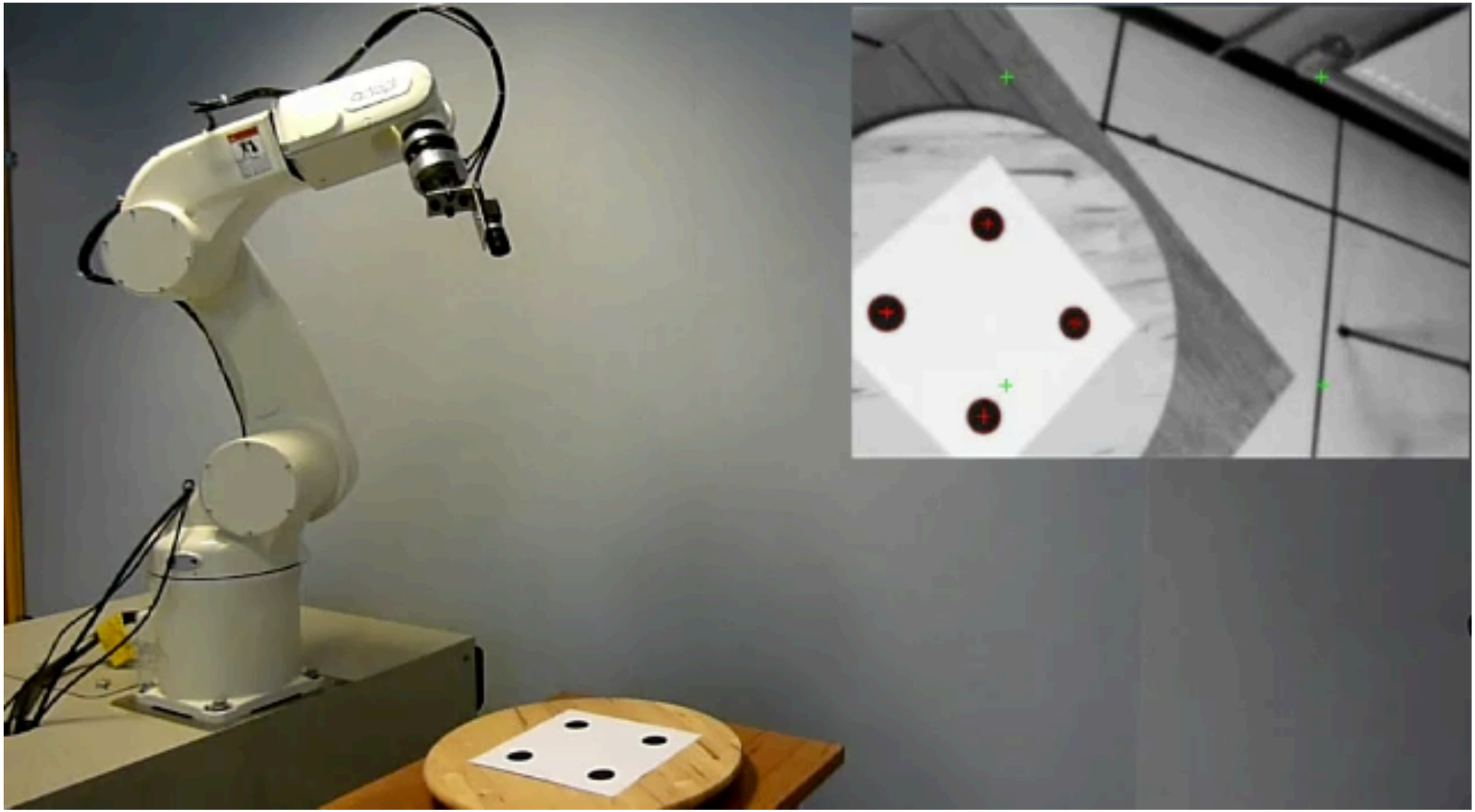
$$\begin{pmatrix} v_x^* \\ v_y^* \\ v_z^* \\ \omega_x^* \\ \omega_y^* \\ \omega_z^* \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{J}_p(u_1, v_1, Z_1) \\ \mathbf{J}_p(u_2, v_2, Z_2) \\ \mathbf{J}_p(u_3, v_3, Z_3) \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{p}_1^* - \boldsymbol{p}_1 \\ \boldsymbol{p}_2^* - \boldsymbol{p}_2 \\ \boldsymbol{p}_3^* - \boldsymbol{p}_3 \end{pmatrix}$$



Simulation


noname
v (pixels)
u (pixels)
0
100
200
300
400
500
600
700
800
900
1000
0
200
400
600
800
1000


Z
Y
X
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
X
-4
-2
0
2
4
Y



Courtesy Fabian Spindler, INRIA Rennes

Practical considerations

$$\begin{pmatrix} v_x^* \\ v_y^* \\ v_z^* \\ \omega_x^* \\ \omega_y^* \\ \omega_z^* \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{J}_p(u_1, v_1, Z_1) \\ \mathbf{J}_p(u_2, v_2, Z_2) \\ \mathbf{J}_p(u_3, v_3, Z_3) \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{p}_1^* - \boldsymbol{p}_1 \\ \boldsymbol{p}_2^* - \boldsymbol{p}_2 \\ \boldsymbol{p}_3^* - \boldsymbol{p}_3 \end{pmatrix}$$

It must actually be possible for $\mathbf{p}_i$ to move to $\mathbf{p}_i^*$

Must get the correspondence correct between the observed point and its desired point

Can be scalar or diagonal matrix

IBVS is complete when the norm of this vector is small

More than 3 points

$$\begin{pmatrix} v_x^* \\ v_y^* \\ v_z^* \\ \omega_x^* \\ \omega_y^* \\ \omega_z^* \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{J}_p(u_1, v_1, Z_1) \\ \mathbf{J}_p(u_2, v_2, Z_2) \\ \vdots \\ \mathbf{J}_p(u_n, v_n, Z_n) \end{pmatrix}^+ \begin{pmatrix} \boldsymbol{p}_1^* - \boldsymbol{p}_1 \\ \boldsymbol{p}_2^* - \boldsymbol{p}_2 \\ \vdots \\ \boldsymbol{p}_n^* - \boldsymbol{p}_n \end{pmatrix}$$

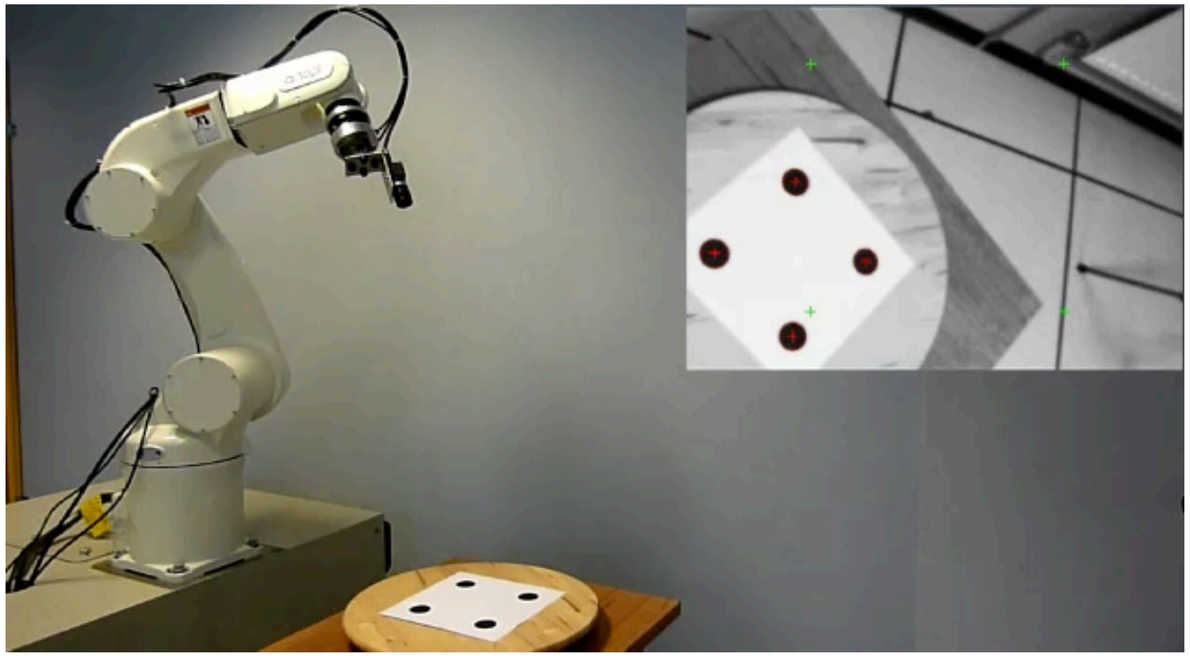
- We can use more than 3 points
- Take the pseudo inverse, or
- Stack the Jacobians of the 3 points that maximise the condition number of the stack

What about Z?

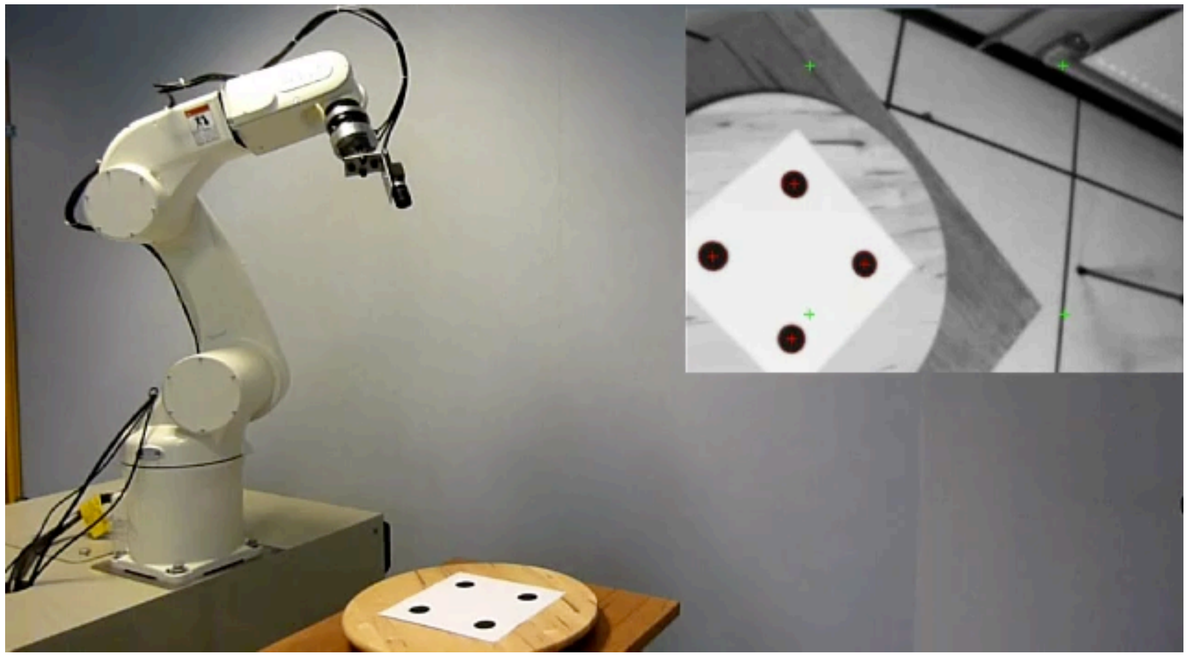
$$\begin{pmatrix} v_x^* \\ v_y^* \\ v_z^* \\ \omega_x^* \\ \omega_y^* \\ \omega_z^* \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{J}_p(u_1, v_1, Z_1) \\ \mathbf{J}_p(u_2, v_2, Z_2) \\ \mathbf{J}_p(u_3, v_3, Z_3) \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{p}_1^* - \boldsymbol{p}_1 \\ \boldsymbol{p}_2^* - \boldsymbol{p}_2 \\ \boldsymbol{p}_3^* - \boldsymbol{p}_3 \end{pmatrix}$$

- Z can be:
  - some reasonable guess: initial value, final value
  - estimated online from successive views if camera is calibrated
- IBVS is remarkably tolerant to errors in Z

The effect of different Z values

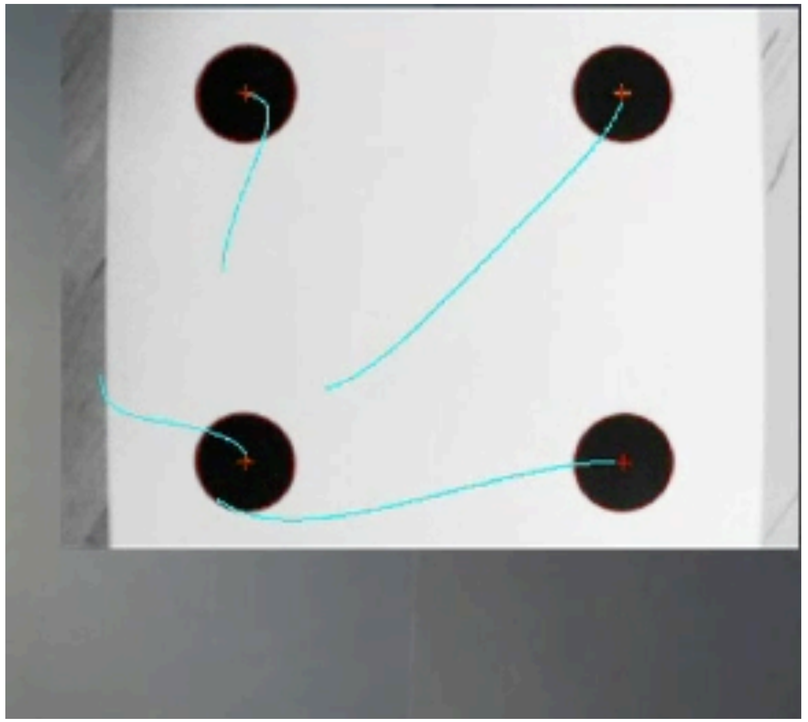


final Z

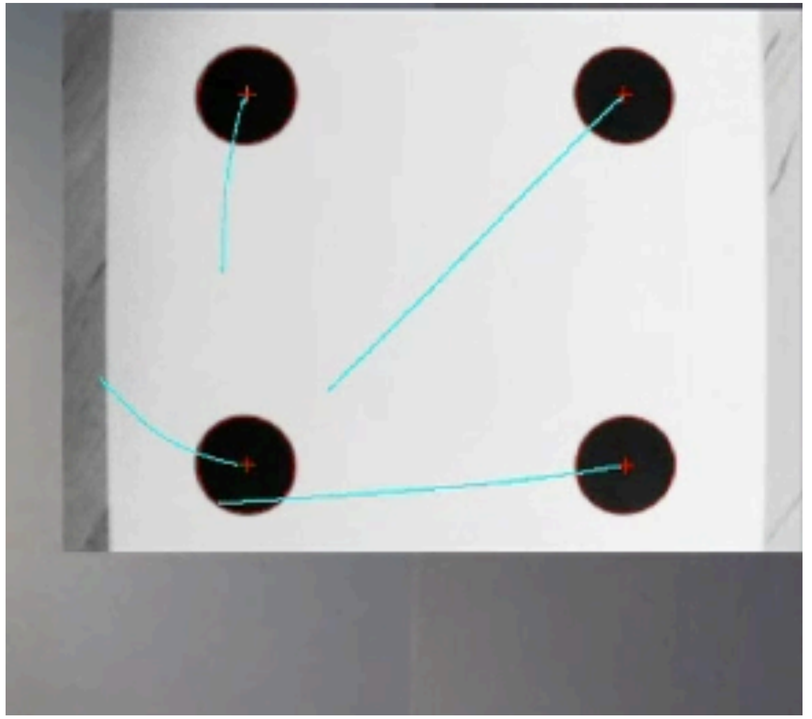


current Z

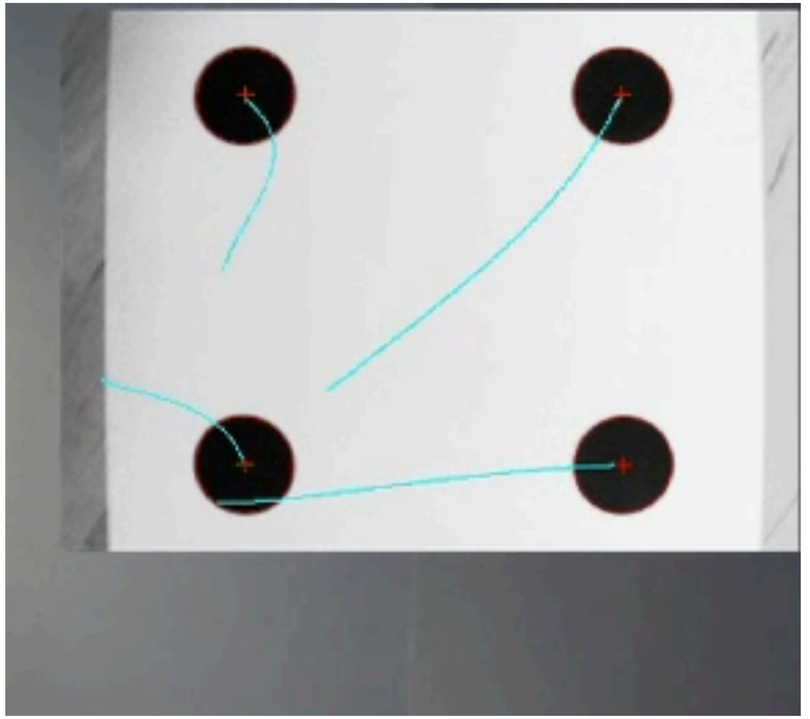
The effect of Z choice



final Z

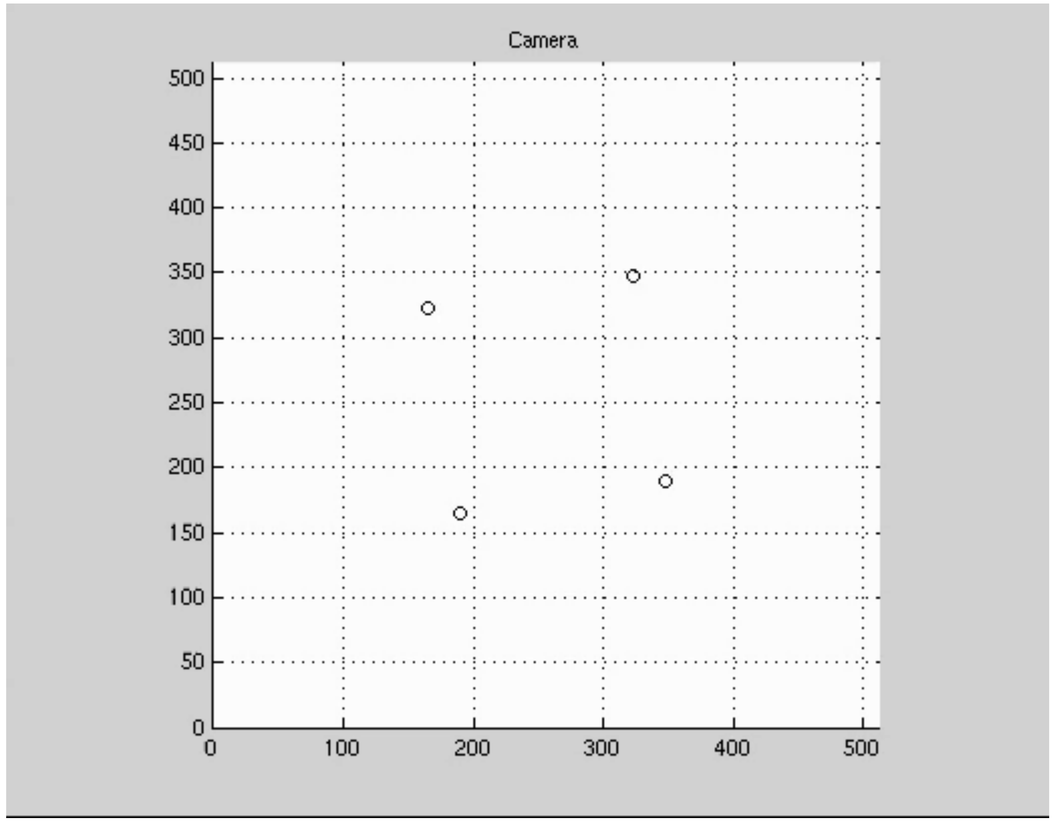


current Z

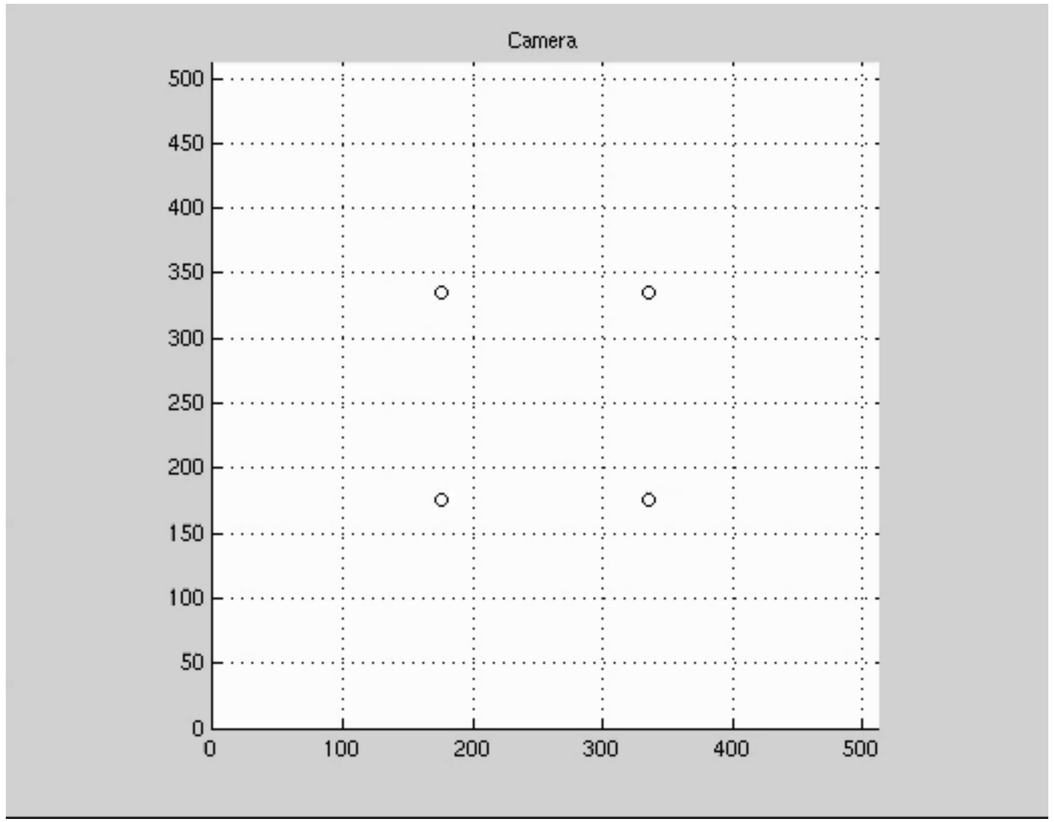


mean Z (initial, desired)

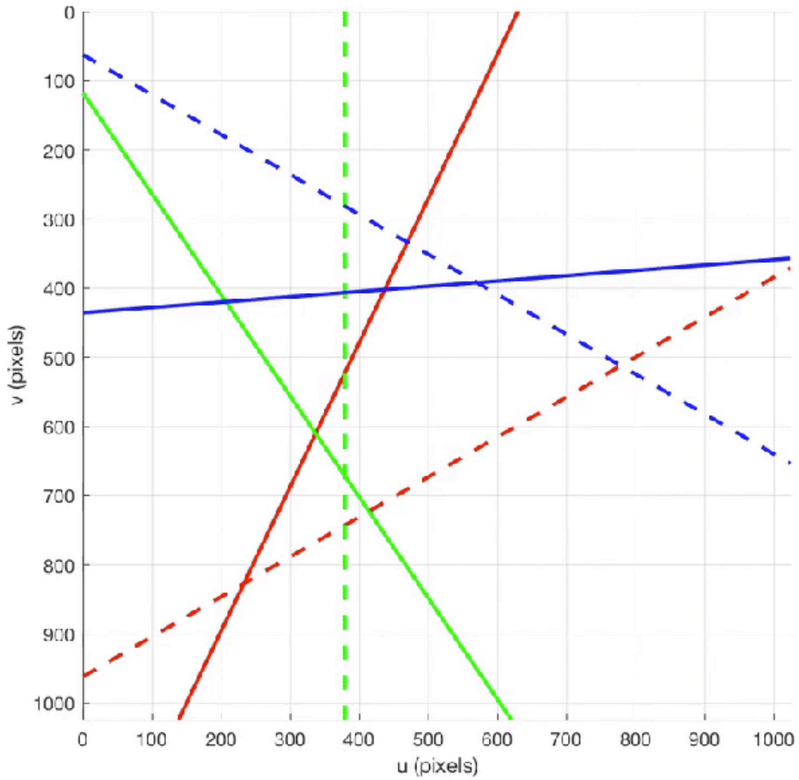
IBVS problems with rotation

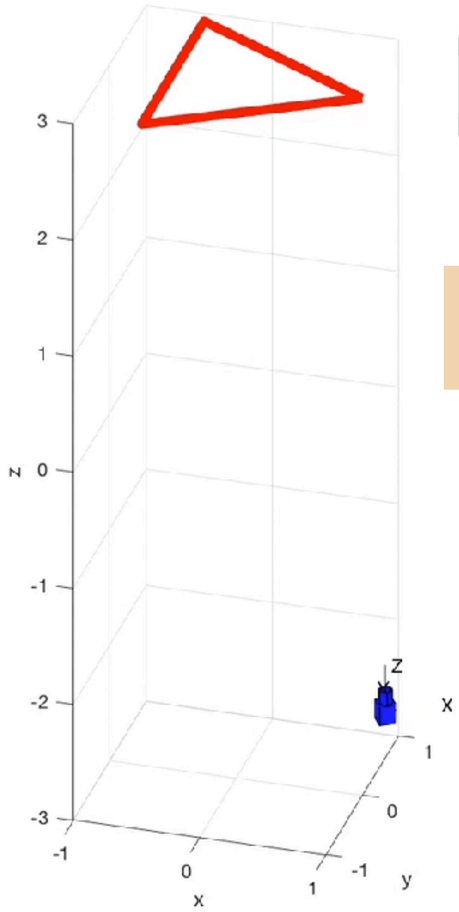


IBVS problems with rotation (2)



Servo to lines

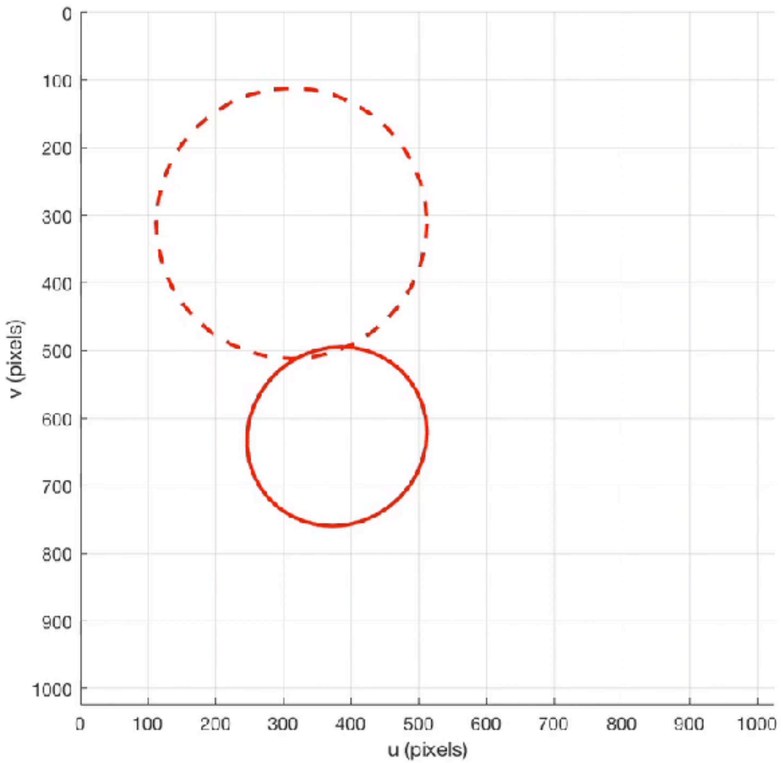


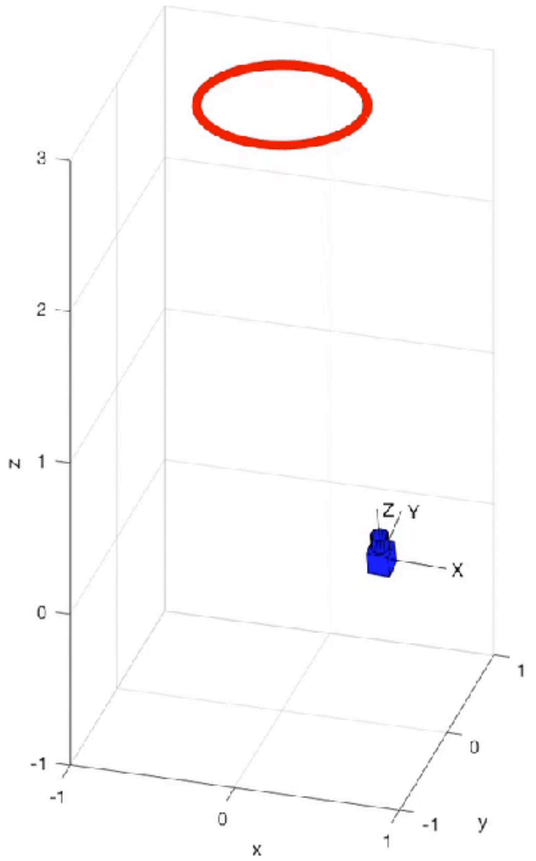


$$\begin{pmatrix}\dot{\theta}\\ \dot{\rho}\end{pmatrix} = \boldsymbol{J}_l \boldsymbol{\nu}$$

$$\boldsymbol{J}_l = \begin{pmatrix}\lambda_\theta \sin\theta & \lambda_\theta \cos\theta & -\rho\lambda_\theta & -\rho\sin\theta & -\rho\cos\theta & -1\\ \lambda_\rho \sin\theta & \lambda_\rho \cos\theta & -\lambda_\rho\rho & -\cos\theta\left(1+\rho^2\right) & \sin\theta\left(1+\rho^2\right) & 0\end{pmatrix}$$

Servo to an ellipse





$$u^2 + E_1 v^2 - 2E_2 uv + 2E_3 u + 2E_4 v + E_5 = 0$$

$$\begin{pmatrix} \dot{E}_1 \\ \dot{E}_2 \\ \vdots \\ \dot{E}_5 \end{pmatrix} = \boldsymbol{J}_e(\boldsymbol{E}, \rho)\boldsymbol{\nu}$$

$$\boldsymbol{J}_e(\boldsymbol{E},\rho) = \begin{pmatrix} 2\beta E_2 - 2\alpha E_1 & 2E_1(\beta - \alpha E_2) & 2\beta E_4 - 2\alpha E_1 E_3 & 2E_4 & 2E_1 E_3 & -2E_2(E_1+1) \\ \beta - \alpha E_2 & \beta E_2 - \alpha\left(2E_2^2 - E_1\right) & \alpha(E_4 - 2E_2 E_3) + \beta E_3 & E_3 & 2E_2 E_3 - E_4 & E_1 - 2E_2^2 - 1 \\ \gamma - \alpha E_3 & \alpha(E_4 - 2E_2 E_3) + \gamma E_2 & \gamma E_3 - \alpha\left(2E_3^2 - E_5\right) & -E_2 & 1 + 2E_3^2 - E_5 & E_4 - 2E_2 E_3 \\ E_3\beta + E_2\gamma - 2\alpha E_4 & E_4\beta + E_1\gamma - 2\alpha E_2 E_4 & \beta E_5 + \gamma E_4 - 2\alpha E_3 E_4 & E_5 - E_1 & 2E_3 E_4 + E_2 & -2E_2 E_4 - E_3 \\ 2\gamma E_3 - 2\alpha E_5 & 2\gamma E_4 - 2\alpha E_2 E_5 & 2\gamma A5 - 2\alpha E_3 E_5 & -2E_4 & 2E_3 E_5 + 2E_3 & -2E_2 E_5 \end{pmatrix}$$

Photometric IBVS

current view

error (desired image - current)

$$\begin{pmatrix} \dot{I}_1 \\ \dot{I}_2 \\ \vdots \\ \dot{I}_N \end{pmatrix} = \boldsymbol{J}_I(\boldsymbol{I})\nu \qquad \boldsymbol{J}_I(\boldsymbol{I}) = \begin{pmatrix} \nabla_I(p_1)\boldsymbol{J}_p(p_1) \\ \nabla_I(p_2)\boldsymbol{J}_p(p_2) \\ \vdots \\ \nabla_I(p_N)\boldsymbol{J}_p(p_N) \end{pmatrix} \in \mathbb{R}^{N\times 6} \qquad \nabla_I(\boldsymbol{p}) = (\nabla_u(\boldsymbol{p}), \nabla_v(\boldsymbol{p}))$$

I_u and I_v from Monday's lecture

Marchand INRIA

Photometric IBVS

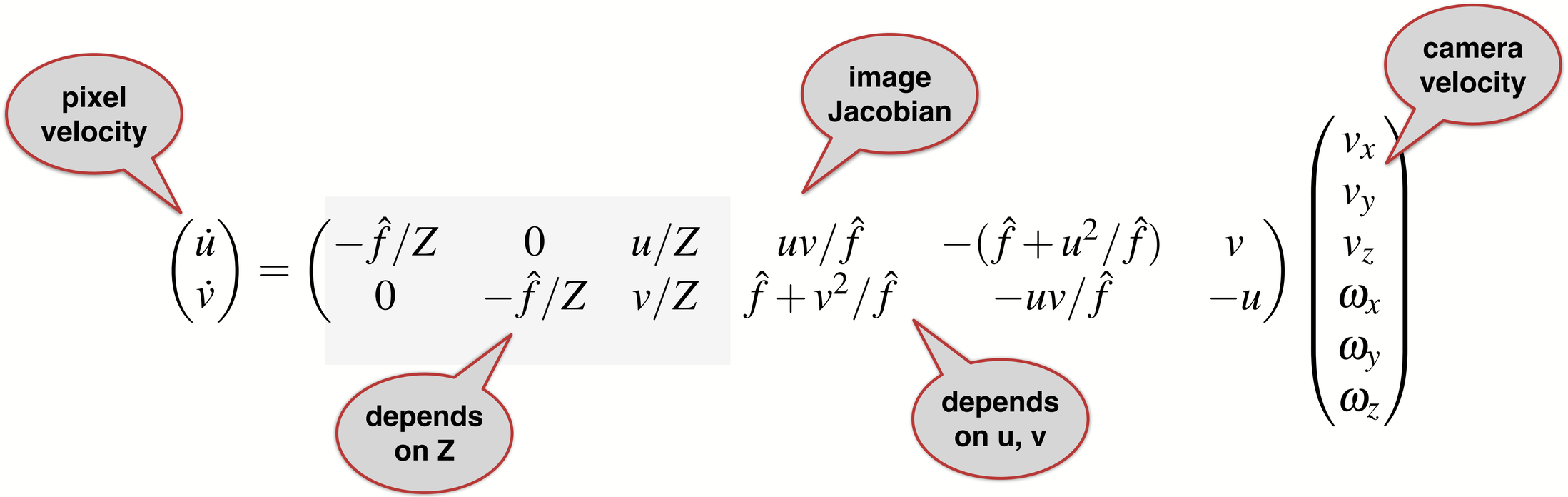
- quite tolerant to occlusions

Summary

- This approach is called Image-based visual servoing
- We describe how we want points to move in the image
- Then we construct a controller that moves the camera so as to create that point motion
- We determine the relative motion of the camera, not absolute. We *steer* the camera to the right pose.
- Works well for the case of 3 points in the scene
  - ➥ cannot work for 1 or 2 points
  - ➥ more complex for 4 or more points
- Remarkably tolerant to errors in the Jacobian
- Converges from a wide range of initial poses (but not global)
- The points never leave the field of view of the camera

Section 8 Flow, distance and camera velocity

Image plane and camera motion



- These quantities are all inter-related
  - ➥ If we know some, we can estimate the other

Estimating distance to a point

pixel velocity

image Jacobian

camera velocity

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} -\hat{f}/Z & 0 & u/Z & uv/\hat{f} & -(\hat{f}+u^2/\hat{f}) & v \\ 0 & -\hat{f}/Z & v/Z & \hat{f}+v^2/\hat{f} & -uv/\hat{f} & -u \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

depends on Z

depends on u, v

Estimating camera speed

pixel velocity

image Jacobian

camera velocity

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} -\hat{f}/Z & 0 & u/Z & uv/\hat{f} & -(\hat{f}+u^2/\hat{f}) & v \\ 0 & -\hat{f}/Z & v/Z & \hat{f}+v^2/\hat{f} & -uv/\hat{f} & -u \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

depends on Z

depends on u, v

CSIRO | used with permission

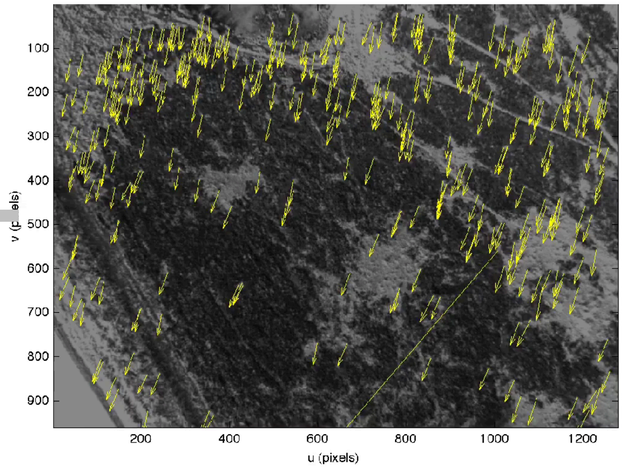
This is known as

visual odometry

Simple visual odometry

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} -\hat{f}/Z & 0 & u/Z & uv/\hat{f} & -(\hat{f}+u^2/\hat{f}) & v \\ 0 & -\hat{f}/Z & v/Z & \hat{f}+v^2/\hat{f} & -uv/\hat{f} & -u \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \approx 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

$\mathbf{J}_u$ $\mathbf{J}_v$ ω



$$\dot{u} = -\hat{f}v_x/Z + \mathbf{J}_u\omega$$

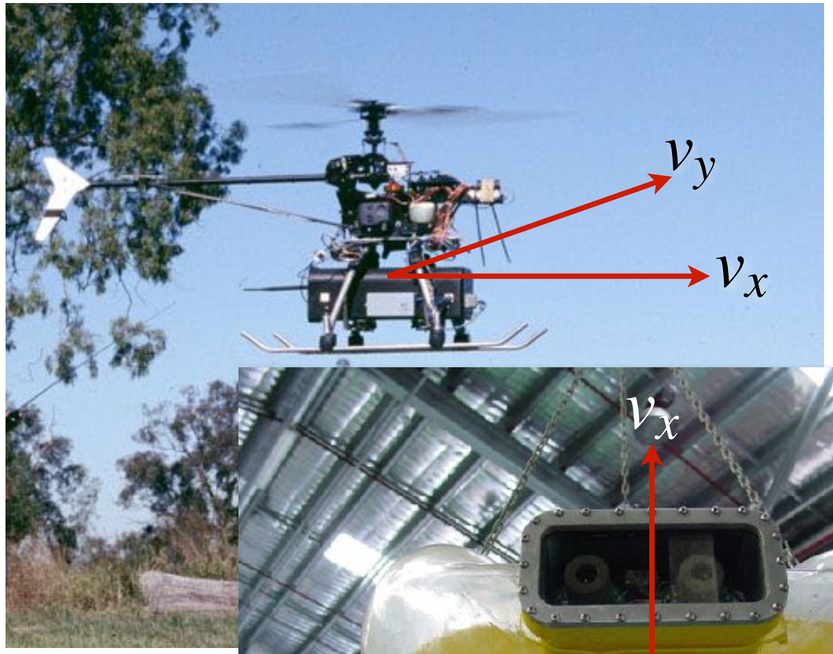
$$\dot{v} = -\hat{f}v_y/Z + \mathbf{J}_v\omega$$

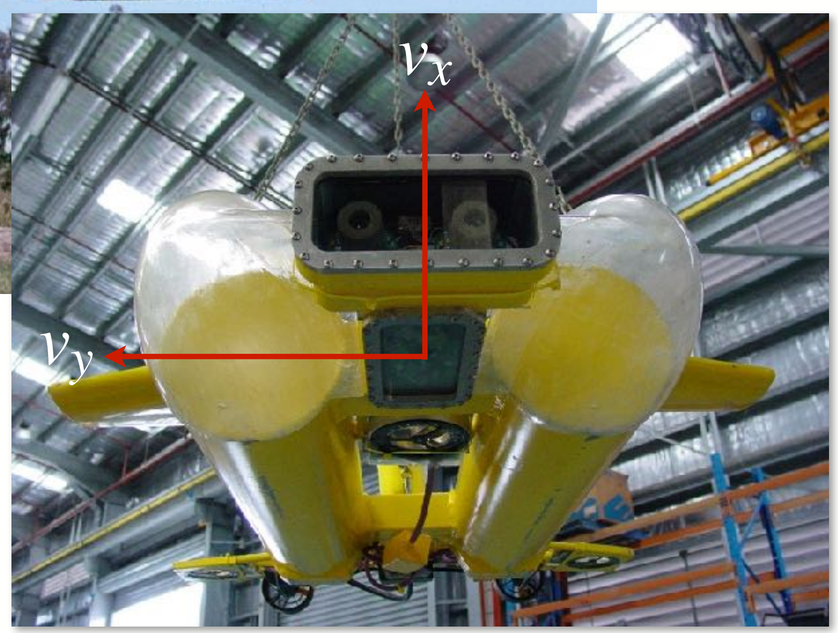
$$v_x = Z(\mathbf{J}_u\omega - \dot{u})/\hat{f}$$

$$v_y = Z(\mathbf{J}_v\omega - \dot{v})/\hat{f}$$

- Assume the robot moves in the x-y plane
  - ➥ Z is known and constant

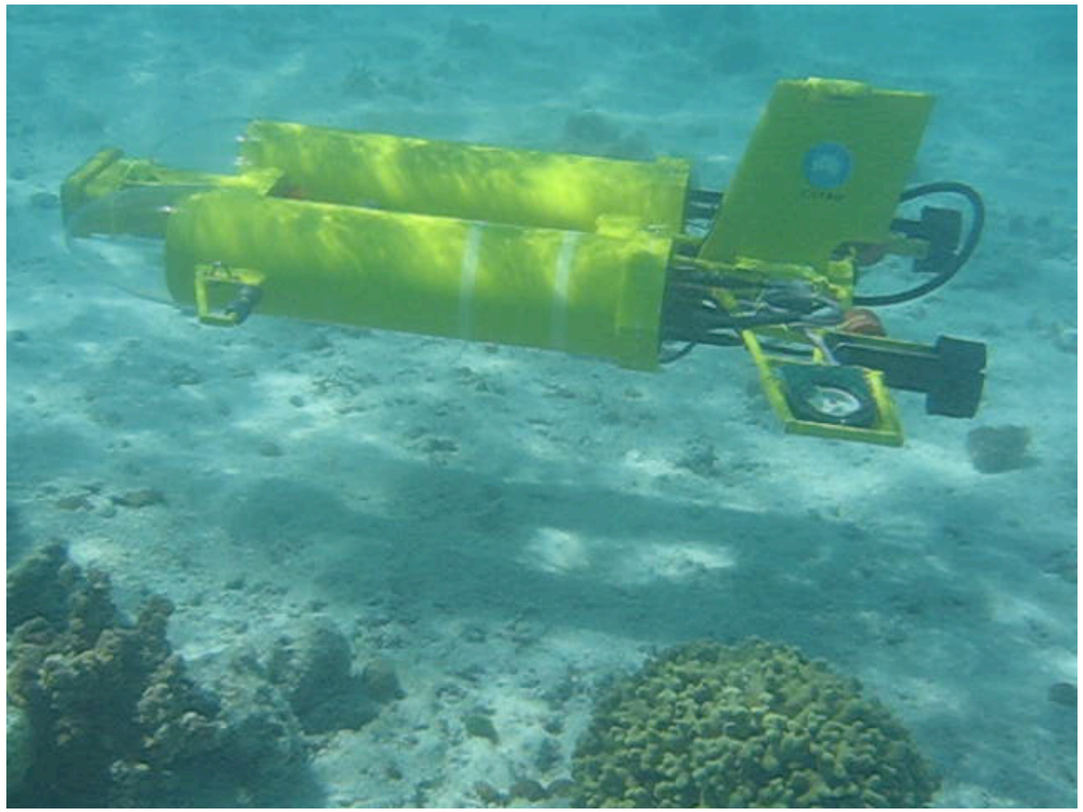
Real-world applications





Helicopter & Starbug

CSIRO | used with permission



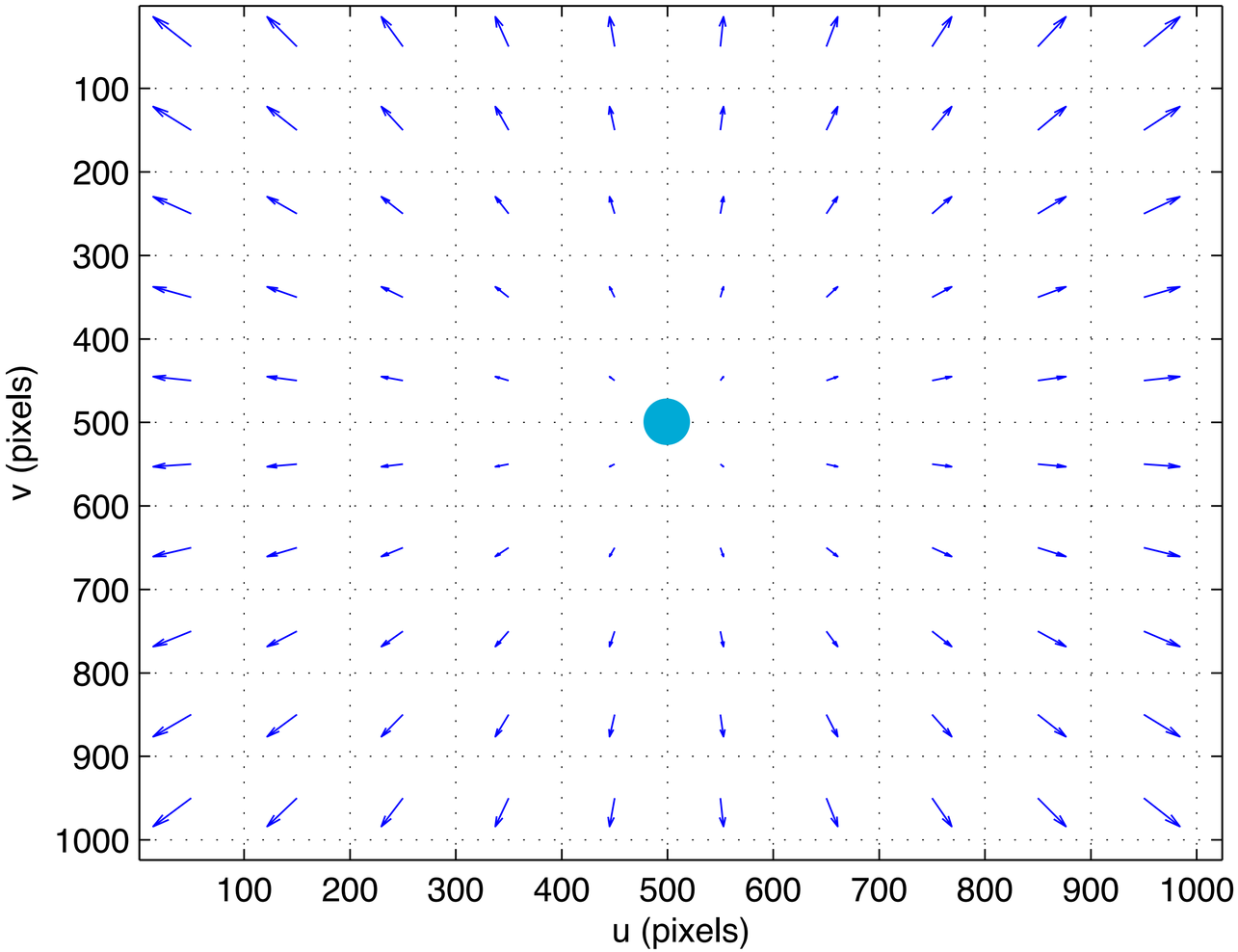
Section 5 Motion perceptibility

Perceptibility

- Perceptibility
  - ➥ Camera motion causes image points to move
  - ➥ Sometimes non-zero camera velocity causes zero change for some points
    - ➥ such motion is **imperceptible**

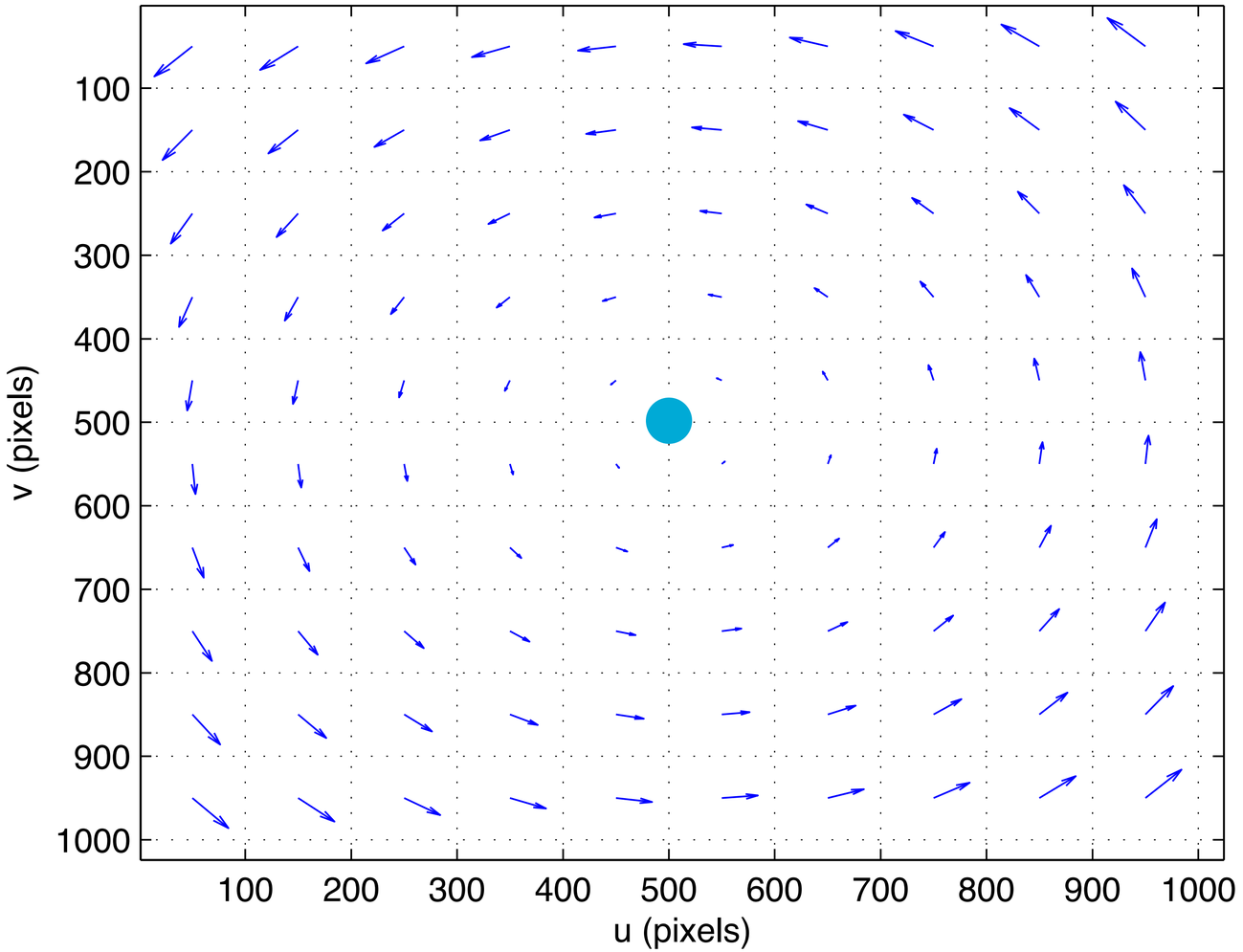
Imperceptible motion

$$v_z$$



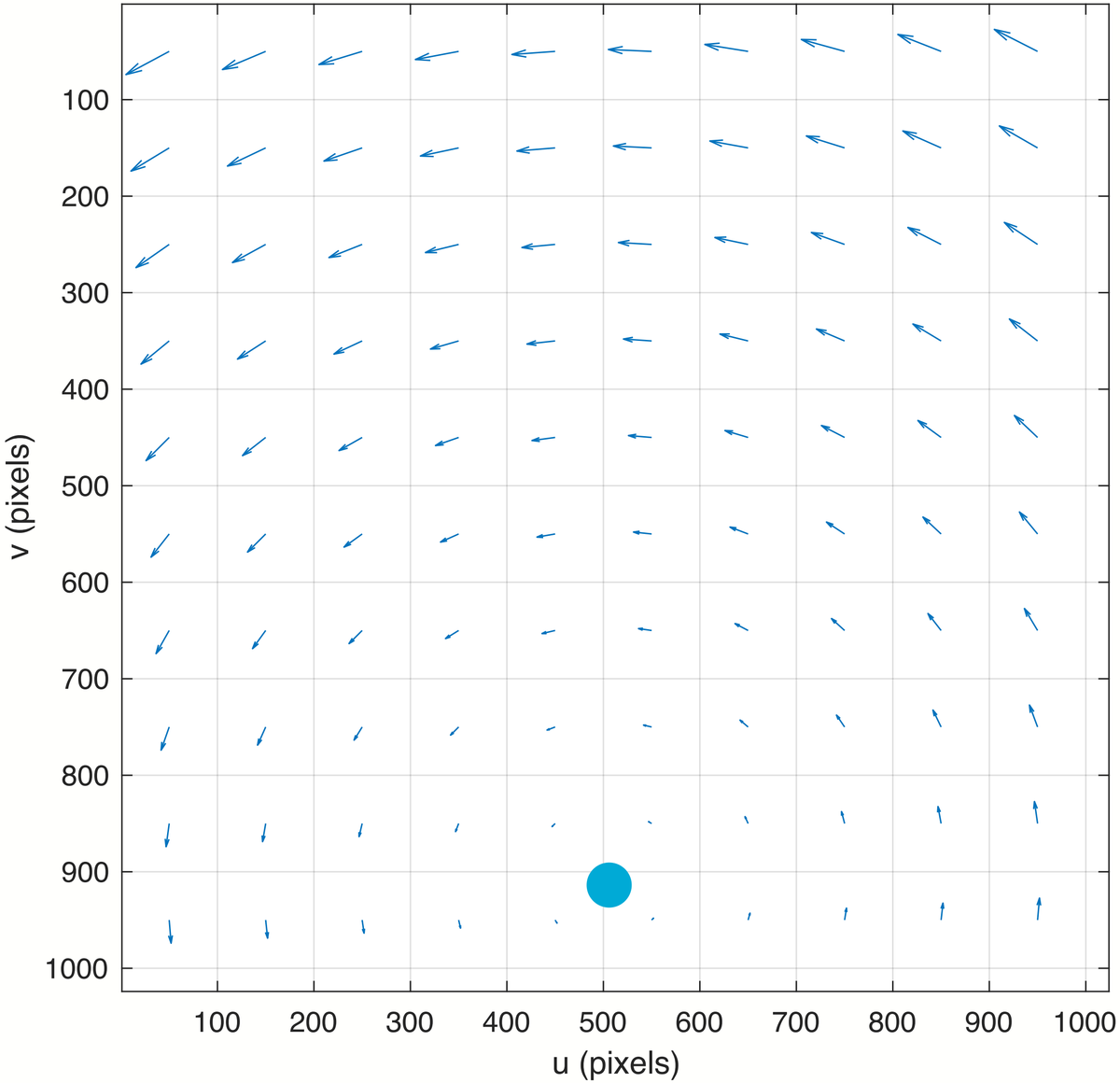
Imperceptible motion

ω_z



Imperceptible motion

$$v_x + \omega_z$$

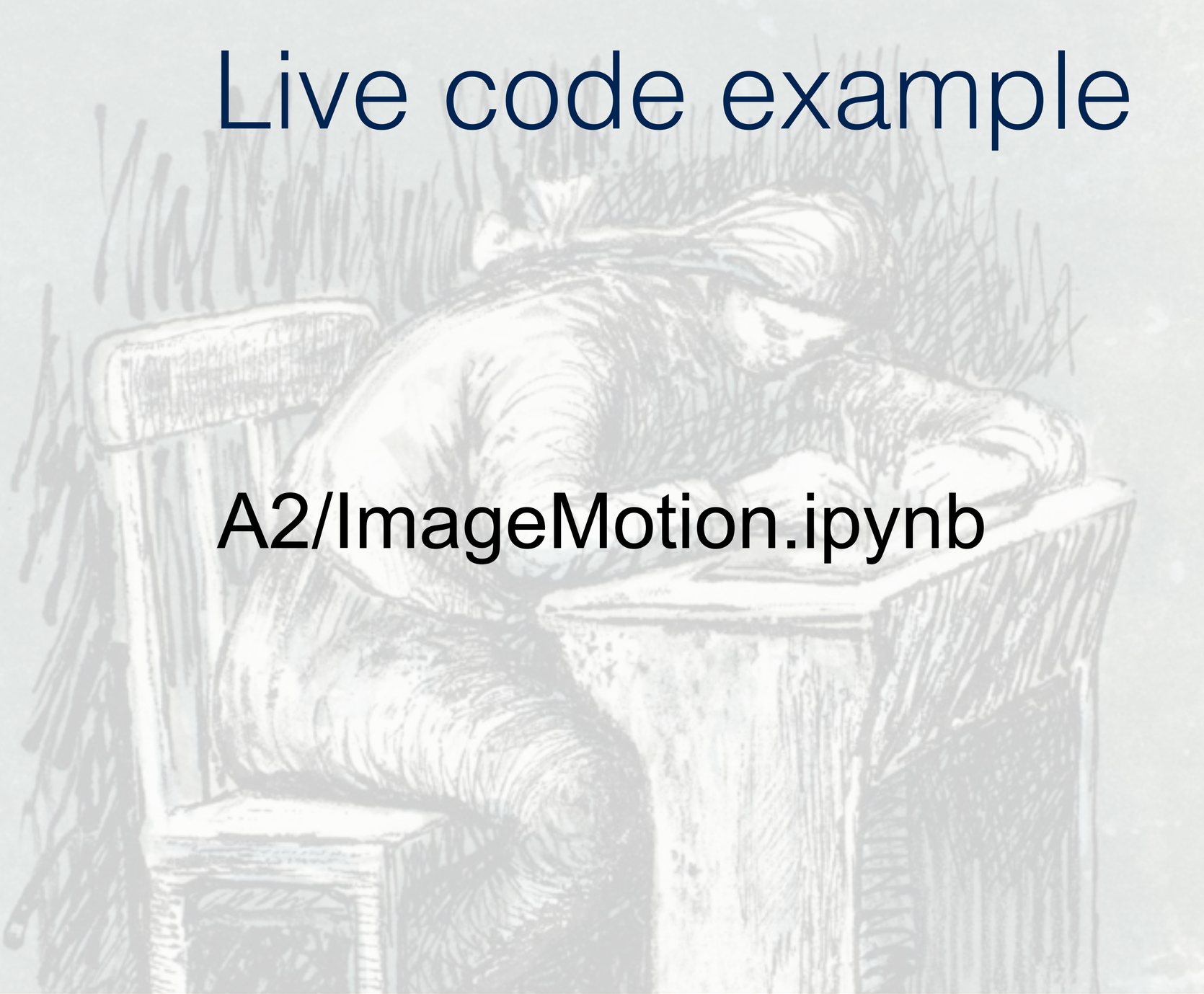


Motion perceptibility

$$\begin{pmatrix}\dot{\bar{u}}\\ \dot{\bar{v}}\end{pmatrix} = \begin{pmatrix} -\frac{\hat{f}}{Z} & 0 & \frac{\bar{u}}{Z} & \frac{\bar{u}\bar{v}}{\hat{f}} & -(\hat{f}+\frac{\bar{u}^2}{\hat{f}}) & \bar{v} \\ 0 & -\frac{\hat{f}}{Z} & \frac{\bar{v}}{Z} & \hat{f}+\frac{\bar{v}^2}{\hat{f}} & -\frac{\bar{u}\bar{v}}{\hat{f}} & -\bar{u} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

- The Jacobian is a 2x6 matrix of rank 2 if $Z \neq \infty$
- The nullity is 4
  - There are 4 possible camera velocities for which there is no optical flow, ie. the point appears not to move

Live code example



A2/ImageMotion.ipynb

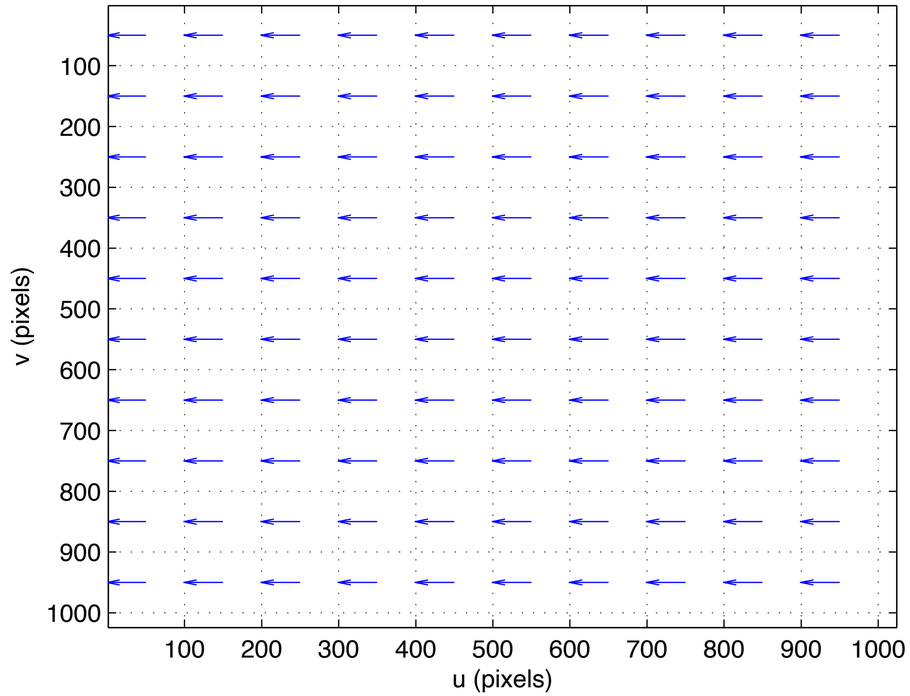
Section 5 Motion ambiguity

Ambiguity

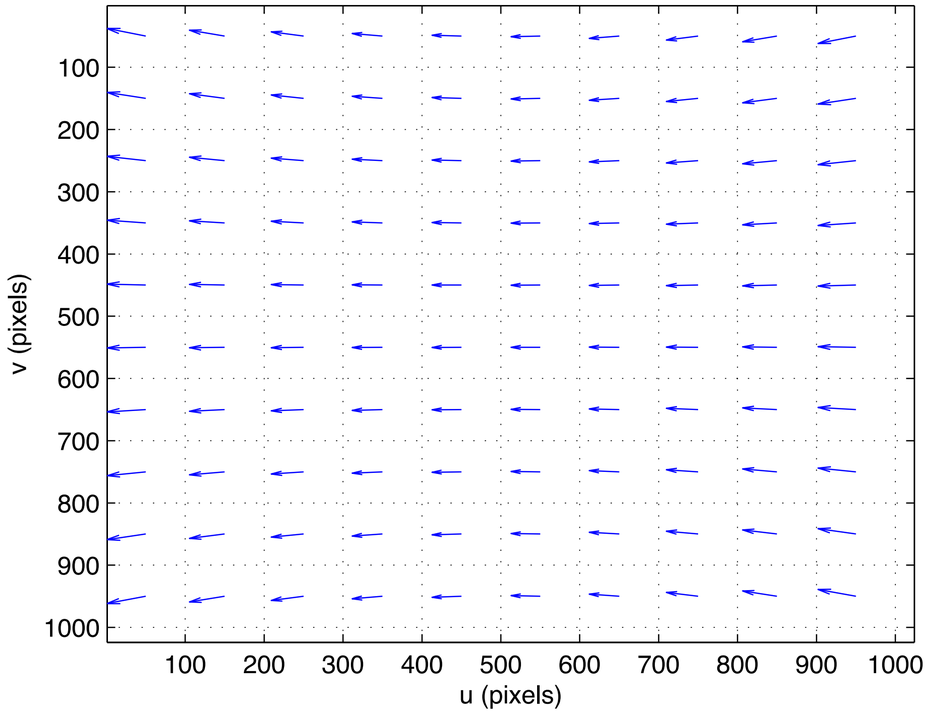
- Ambiguity
  - ➥ Several distinct camera velocities cause the same image change

Motion ambiguity

v_x



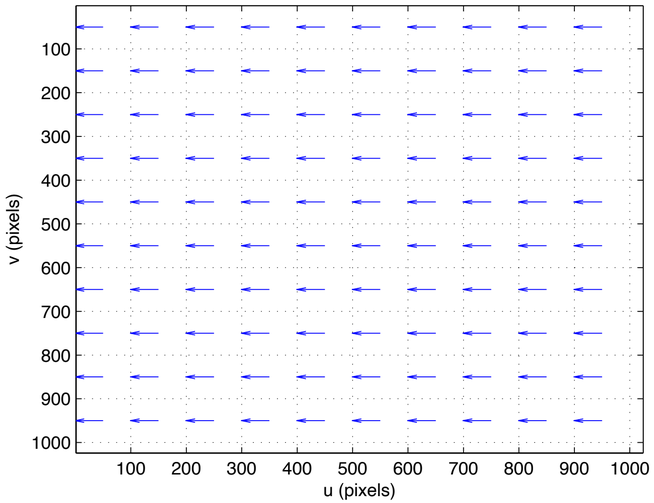
ω_y



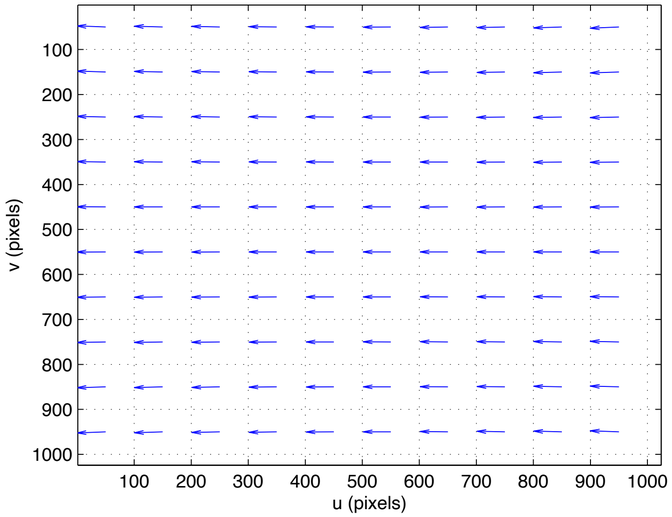
- Ambiguity between translation and rotation

Motion ambiguity

v_x



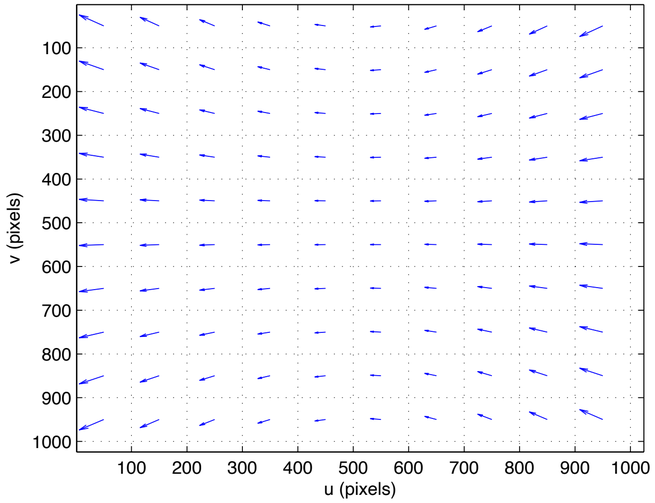
ω_y



large f

telephoto (zoom) lens

ω_y



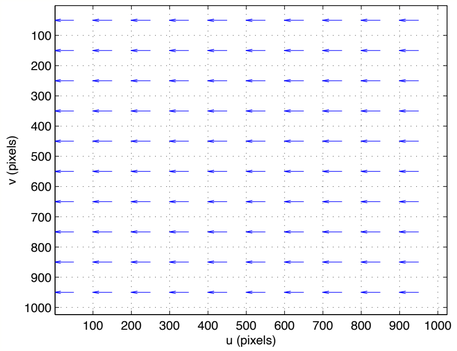
small f

wide angle lens

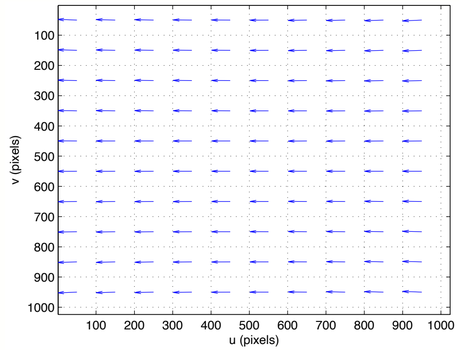
Motion ambiguity

$$\begin{pmatrix}\dot{u}\\ \dot{v}\end{pmatrix} = \begin{pmatrix}-\hat{f}/Z & 0 & u/Z & uv/\hat{f} & -(\hat{f}+u^2/\hat{f}) & v\\ 0 & -\hat{f}/Z & v/Z & \hat{f}+v^2/\hat{f} & -uv/\hat{f} & -u\end{pmatrix}\begin{pmatrix}v_x\\ v_y\\ v_z\\ \omega_x\\ \omega_y\\ \omega_z\end{pmatrix}$$

v_x



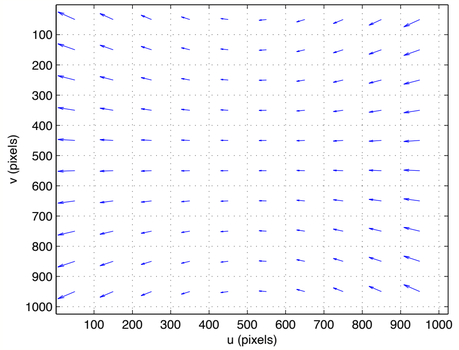
ω_y



large f

telephoto (zoom) lens

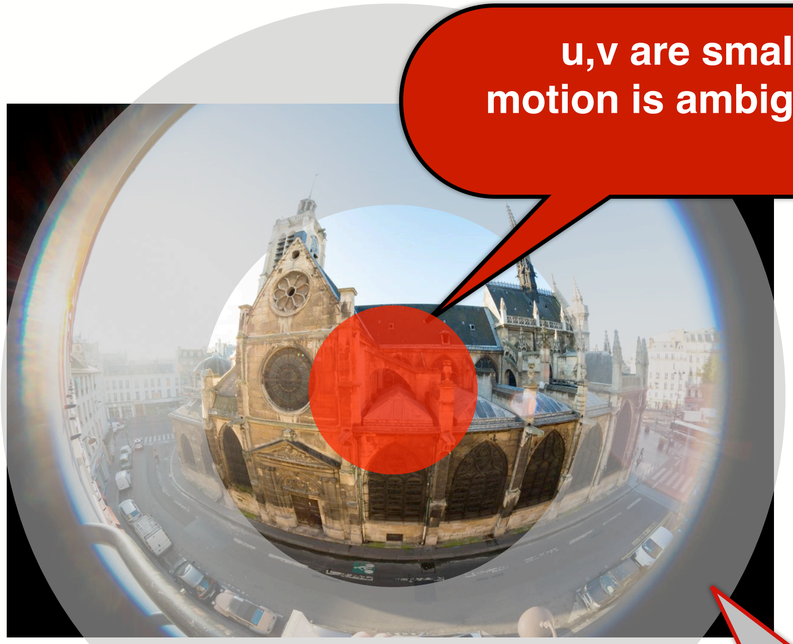
ω_y



small f

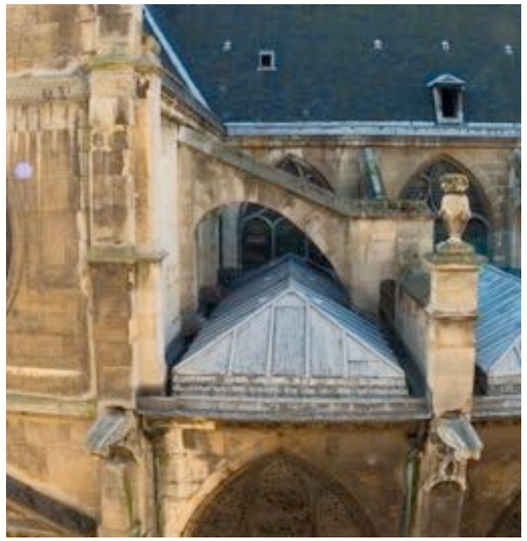
wide angle lens

Ambiguity and focal length



small f

very wide field of view



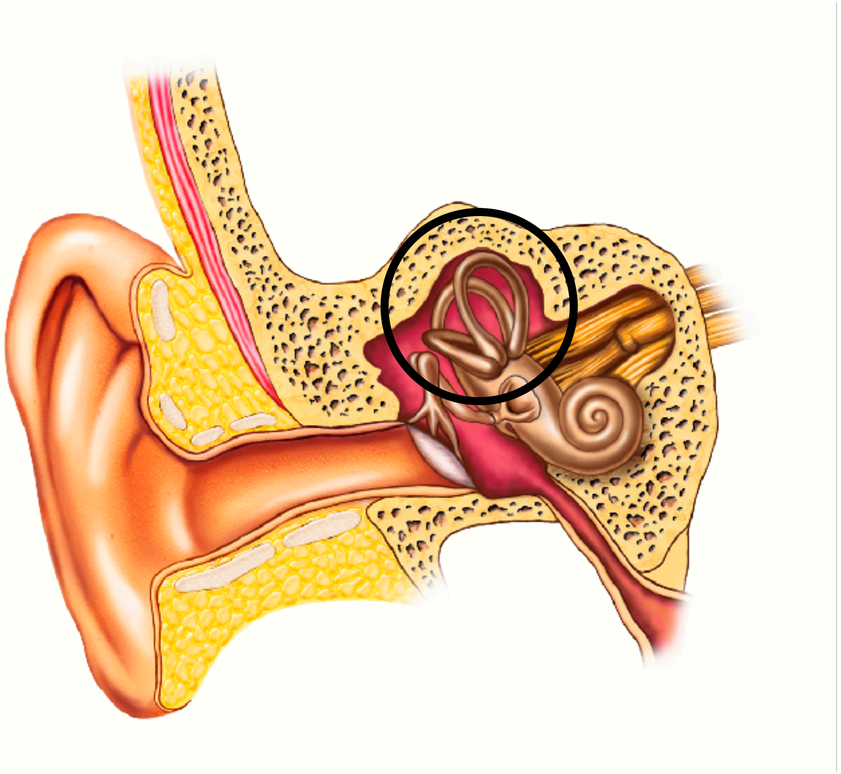
large f

very narrow field of view

**u,v are large,
motion is not ambiguous**

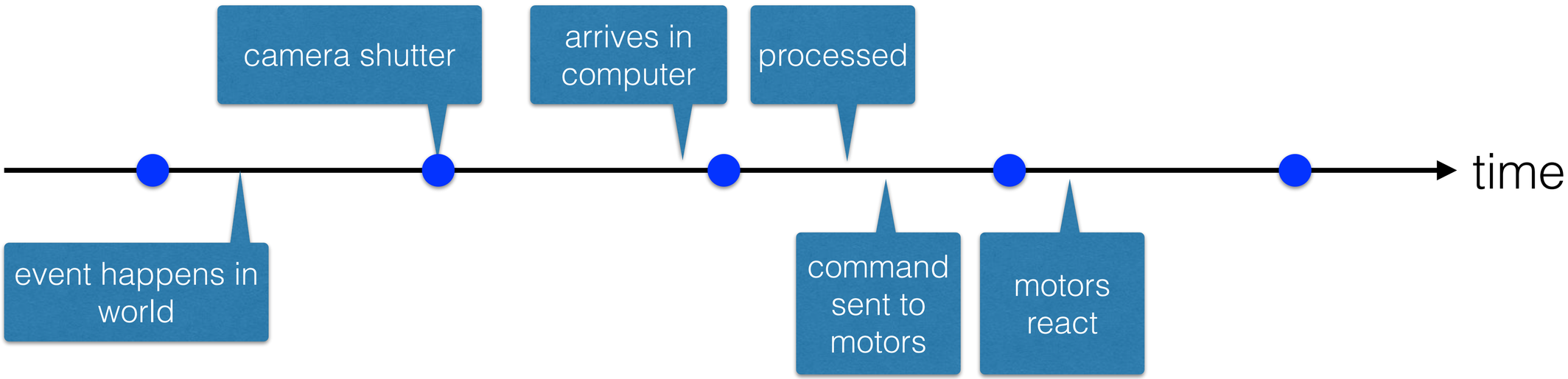
Add a sensor

- Use a separate sensor to measure rotation

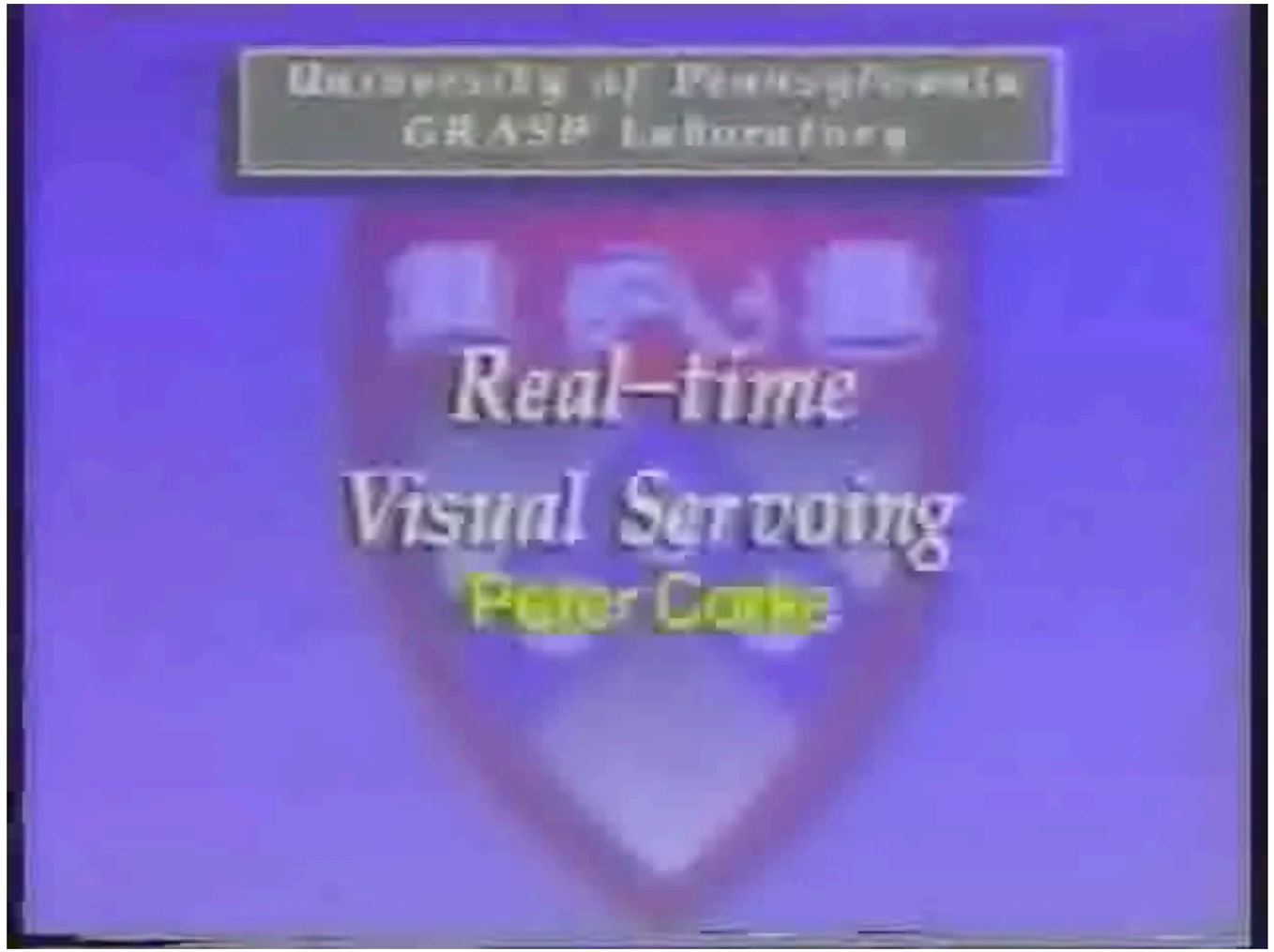


Section Perceptual dynamics

Not everything happens at once



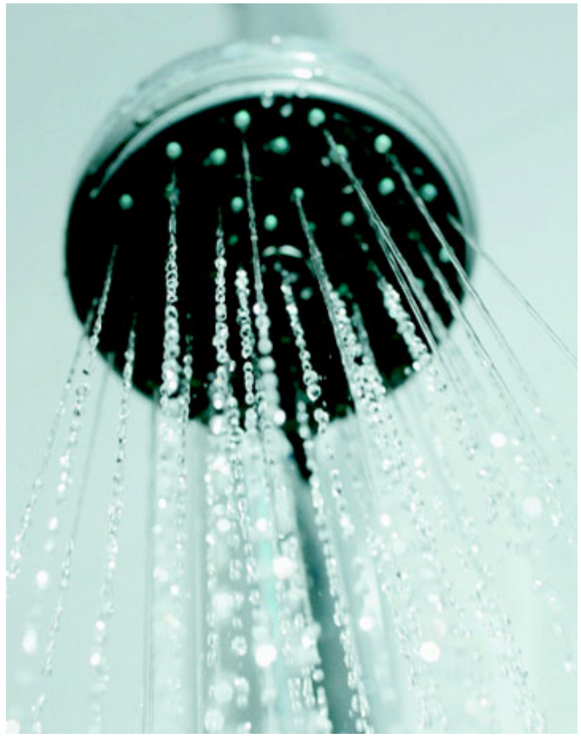
Visual servoing: simple feedback



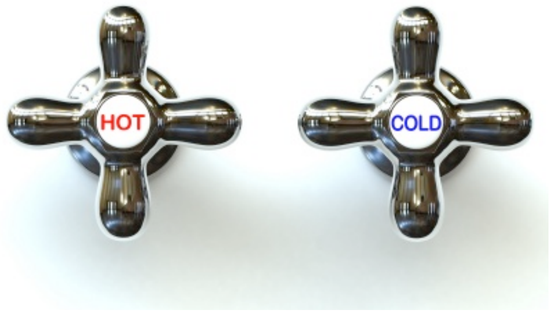
1DOF, feedback, 60Hz (1988)

Destabilizing effects of time delay

- Trying to talk on a phone/ skype call with a time delay



Pilot induced oscillation



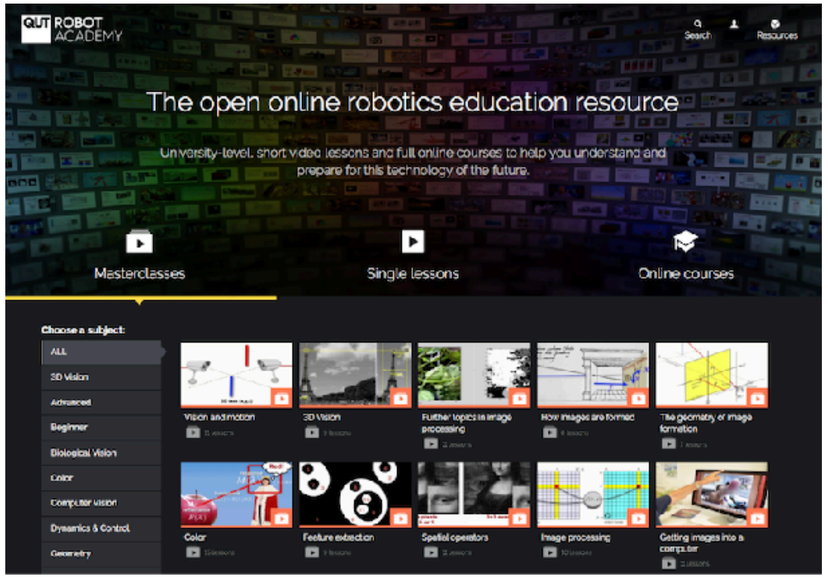
Robot watching a moving object

1993

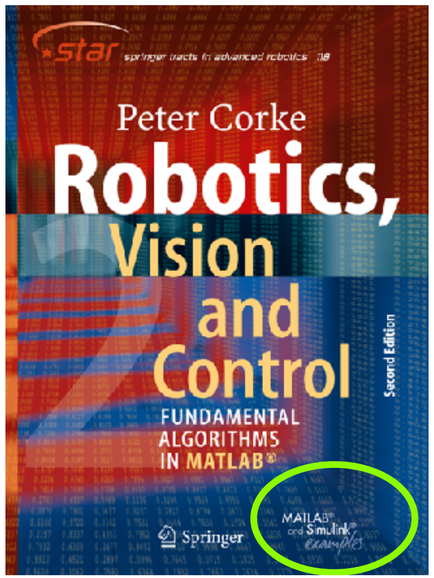
Robot following a flying object

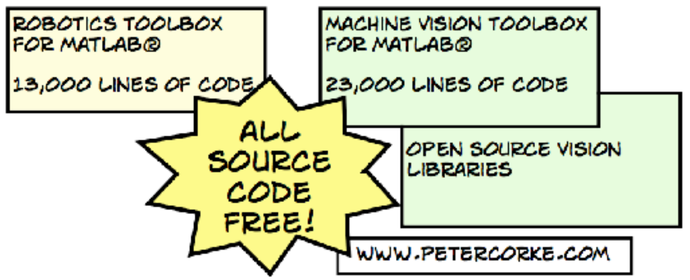
1993

Further reading

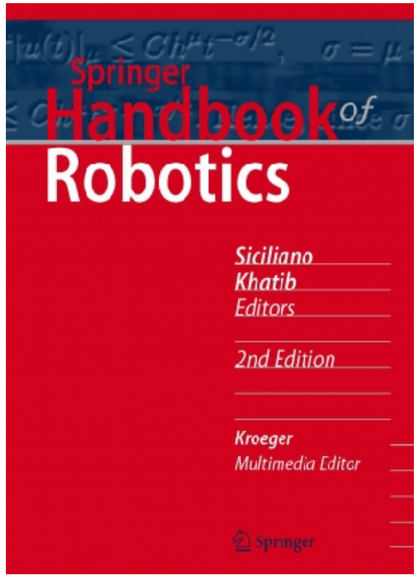


- **Robot Academy** (free)
  - How images are formed
  - The geometry of image formation
  - Vision & motion masterclass





- **Robotics, Vision & Control** **
  - Chapters 15, 16



- **Springer Handbook of Robotics** **
  - Second edition, Chapter 34
  - Videos

** Probably free through your university library