© 2022 The University of Sheffield

COM2009-3009
Robotics

Lecture 7

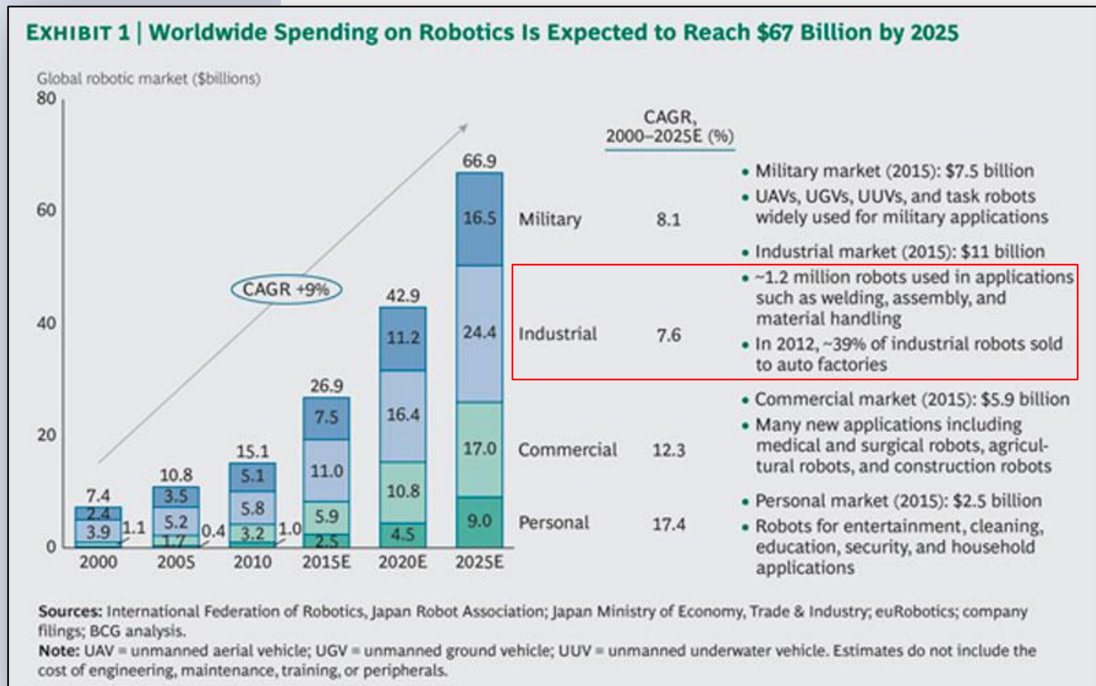
Reaching and Grasping

Dr Tom Howard

Multidisciplinary Engineering Education (MEE)

The University Of Sheffield.

Industrial Robots Dominate

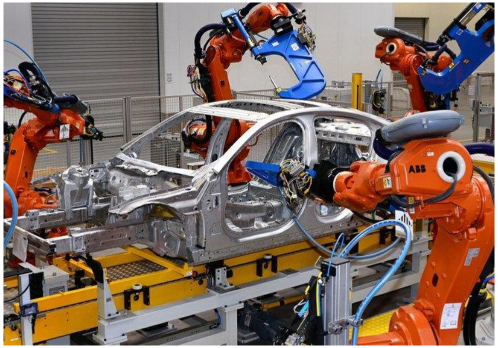
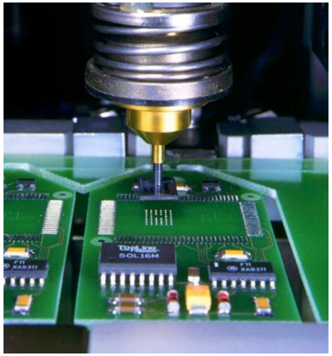


https://www.therobotreport.com/latest-research-report-shows-10-4-cagr-for-robotics-to-2025/

© 2022 The University of Sheffield

Moving from A to B



© 2022 The University of Sheffield

This lecture will cover

1. Using kinematics for robot arm reaching
2. Image-Based Visual Servoing
3. Grasping

© 2022 The University of Sheffield

Kinematic Control of Robot Arms

Task: control the position of the end effector by the coordinated action of the robot's joints and linkages.

Data: Joint angles & Linkage lengths

Forward kinematics is the process of using the measured joint angles, and specific kinematic equations of a given robot to compute the position of the end-effector.

joint angles + K eq'ns -> end eff position

Inverse kinematics uses the specific kinematic equations of the robot and computes the joint angles (and linkages) to obtain a desired end-effector position.

des end eff position + K eq'ns -> joint angles

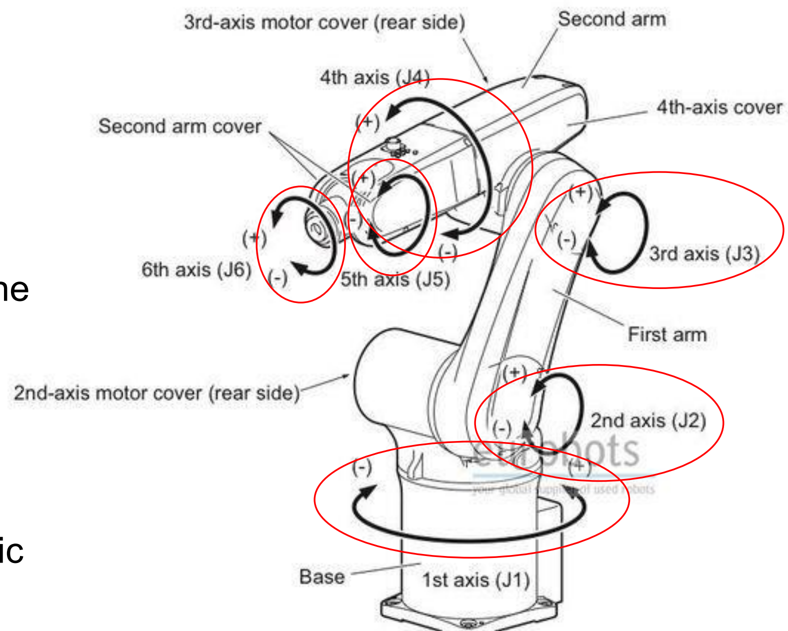


Diagram from www.Thespod.com

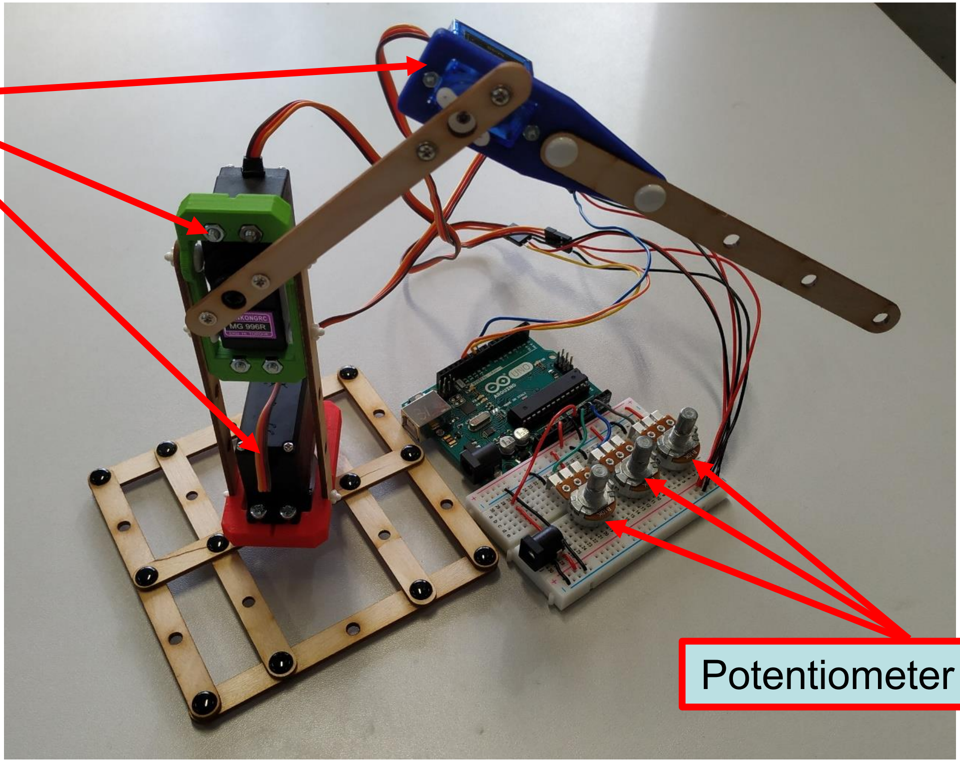
The University Of Sheffield.

SHEFFIELD ROBOTICS

© 2022 The University of Sheffield

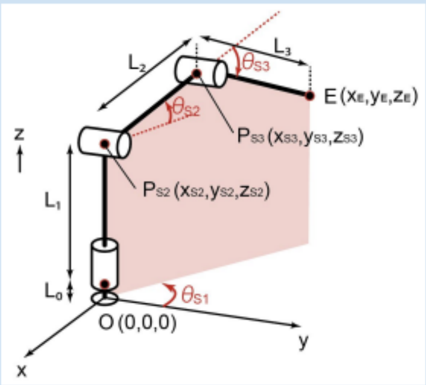
Kinematic Control of Robot Arms

A simple example:



Kinematic Control of Robot Arms

Inverse kinematics



Given position E (x_E, y_E, z_E), derive θ_{S1}, θ_{S2}, and θ_{S3}.

$$
E \begin{cases}
x_E = -L_2 \cos(\theta_{S2} - \pi/2)\sin(\theta_{S1} - \pi/2) - L_3\cos(\theta_{S3} - \theta_{S2})\sin(\theta_{S1} - \pi/2) \\
y_E = L_2 \cos(\theta_{S2} - \pi/2)\cos(\theta_{S1} - \pi/2) + L_3\cos(\theta_{S3} - \theta_{S2})\cos(\theta_{S1} - \pi/2) \\
z_E = L_0 + L_1 + L_2\sin(\theta_{S2} - \pi/2) - L_3\sin(\theta_{S3} - \theta_{S2})
\end{cases}
$$

$\cos(\theta - \pi/2) = \sin(\theta)$

$\sin(\theta - \pi/2) = -\cos(\theta)$

Sum and Difference Identities

$\sin(a+b) = \sin a\cos b + \cos a\sin b$

$\sin(a-b) = \sin a\cos b - \cos a\sin b$

$\cos(a+b) = \cos a\cos b - \sin a\sin b$

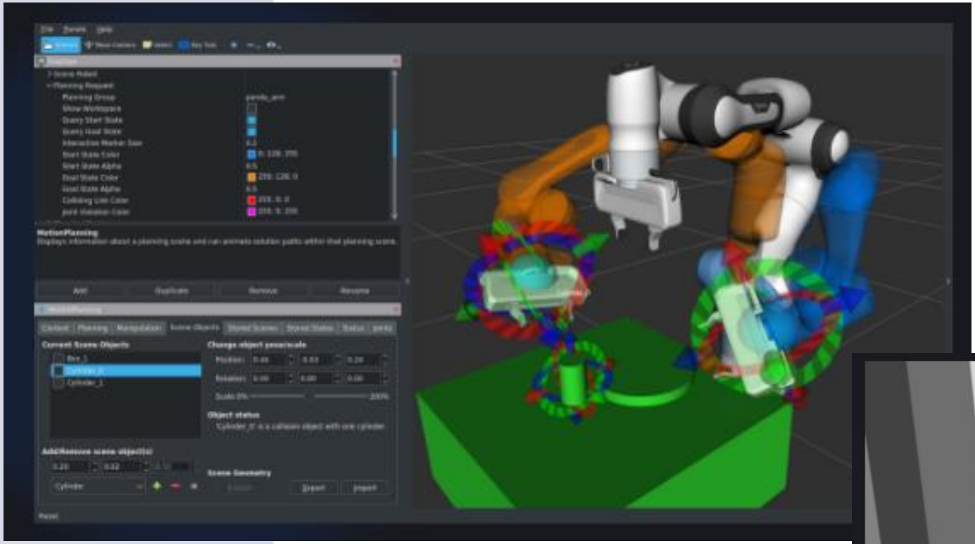
$\cos(a-b) = \cos a\cos b + \sin a\sin b$

$$
E \begin{cases}
x_E = +L_2 \sin(\theta_{S2})\cos(\theta_{S1}) + L_3\cos(\theta_{S3} - \theta_{S2})\cos(\theta_{S1}) \\
y_E = L_2 \sin(\theta_{S2})\sin(\theta_{S1}) + L_3\cos(\theta_{S3} - \theta_{S2})\sin(\theta_{S1}) \\
z_E = L_0 + L_1 - L_2\cos(\theta_{S2}) - L_3\sin(\theta_{S3} - \theta_{S2})
\end{cases}
$$

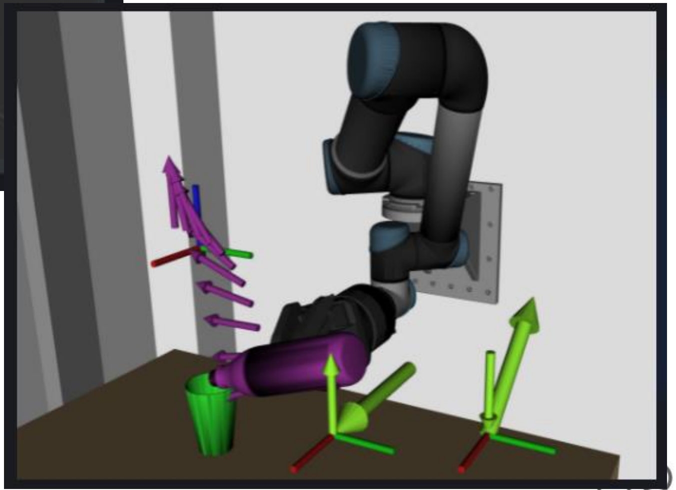
Courtesy of Dr Shuhei Miyashita (ACS231 "Mechatronics")

© 2022 The University of Sheffield

Kinematic Control of Robot Arms

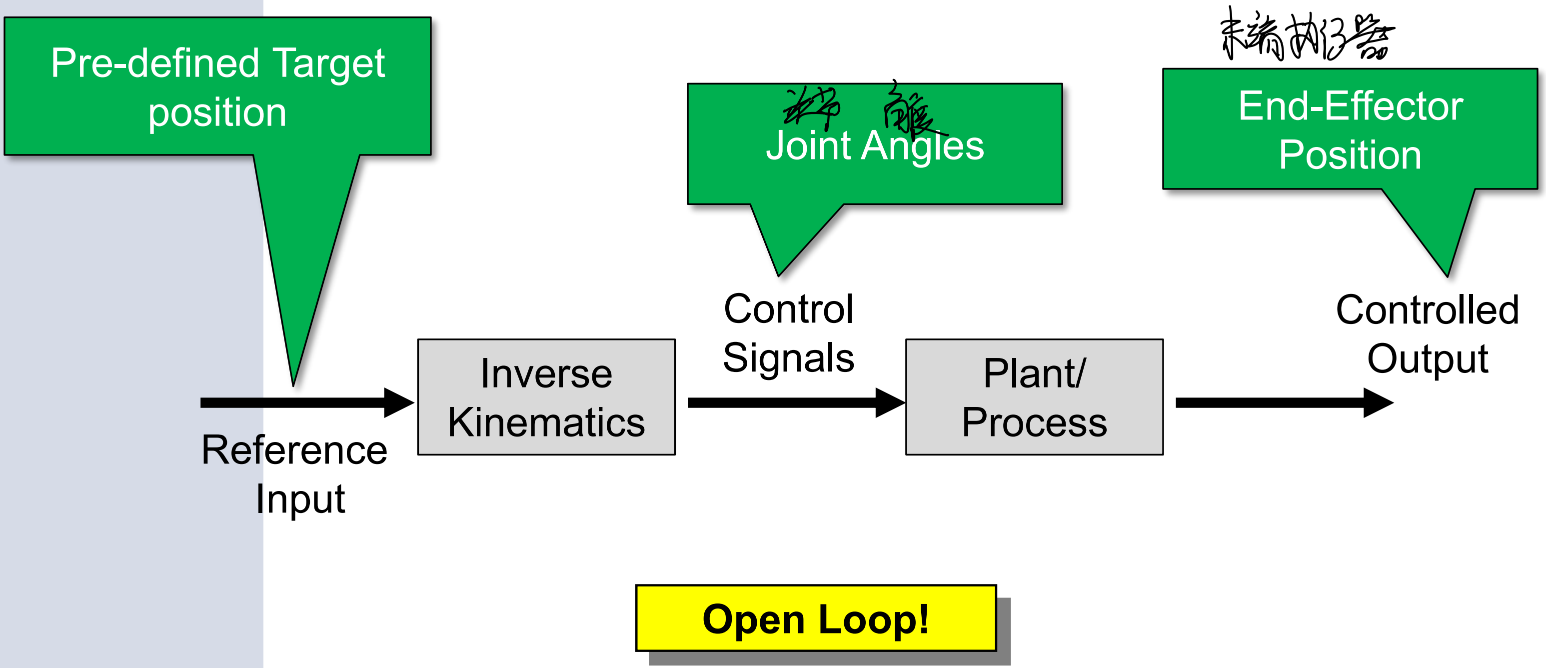


moveit.ros.org

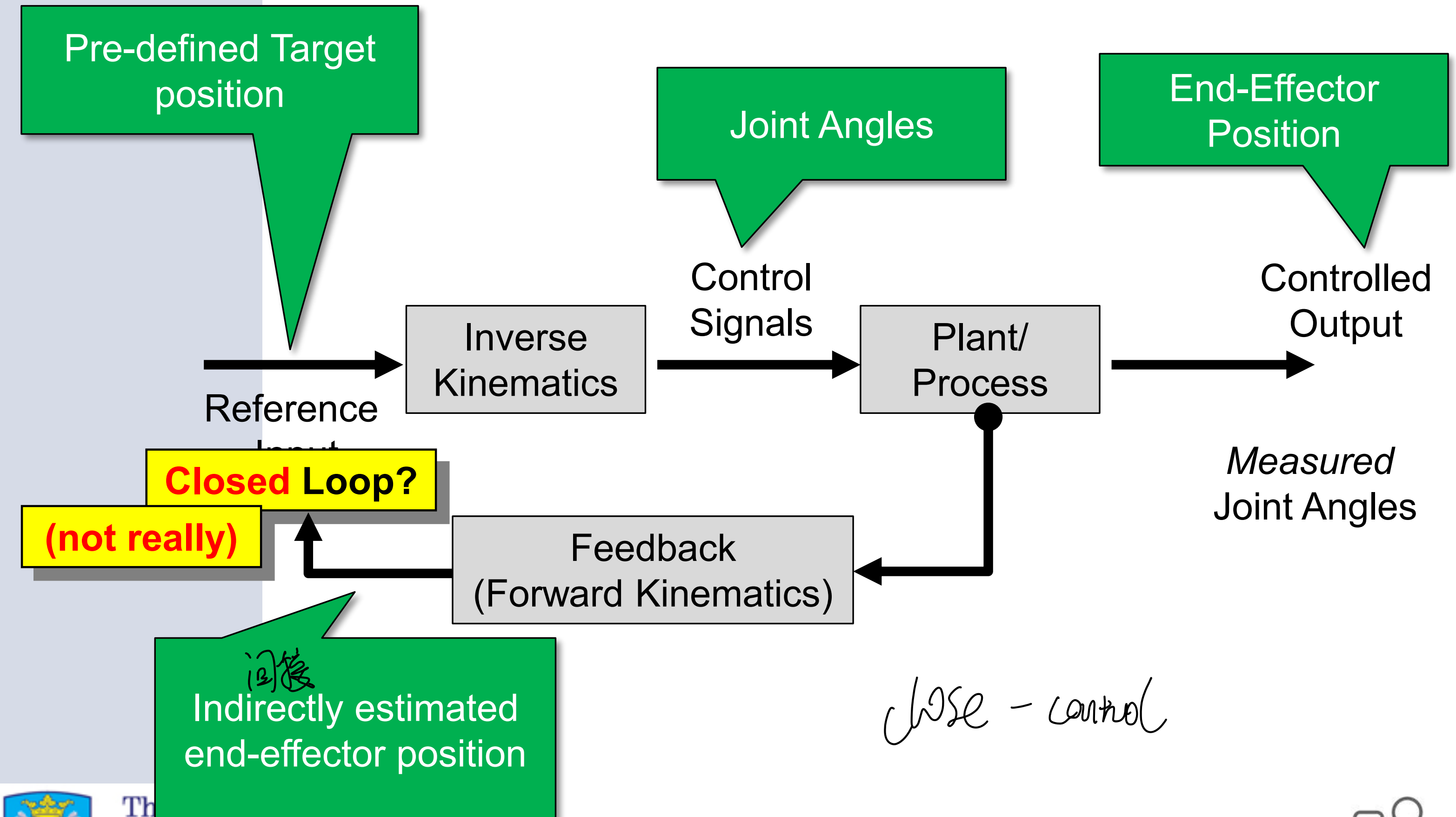


© 2022 The University of Sheffield

Kinematic Control of Robot Arms



Kinematic Control of Robot Arms



© 2022 The University of Sheffield

Kinematic Control of Robot Arms

- Common errors:
  - **Sensor** Errors:
    - Sensors drift, fail or are simply not able to directly monitor the output directly (end effector position)
  - **Positional** Errors:
    - Target position ≠ actual position
  - **Controller** Errors:
    - System performance degrades over time or system models aren't accurate

The University Of Sheffield.

SHEFFIELD ROBOTICS

© 2022 The University of Sheffield

Which *can* be overcome by:

$$$!

High precision sensors

Stiff metal linkages

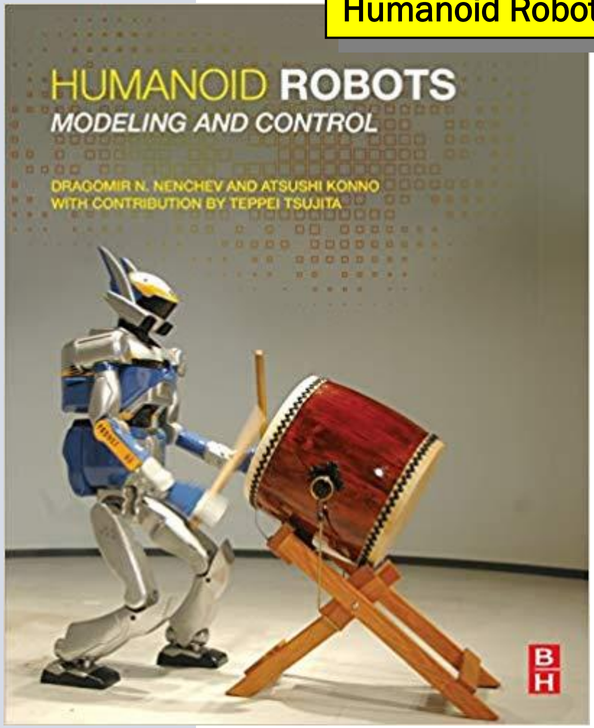
Highly engineered workspace

High precision motors

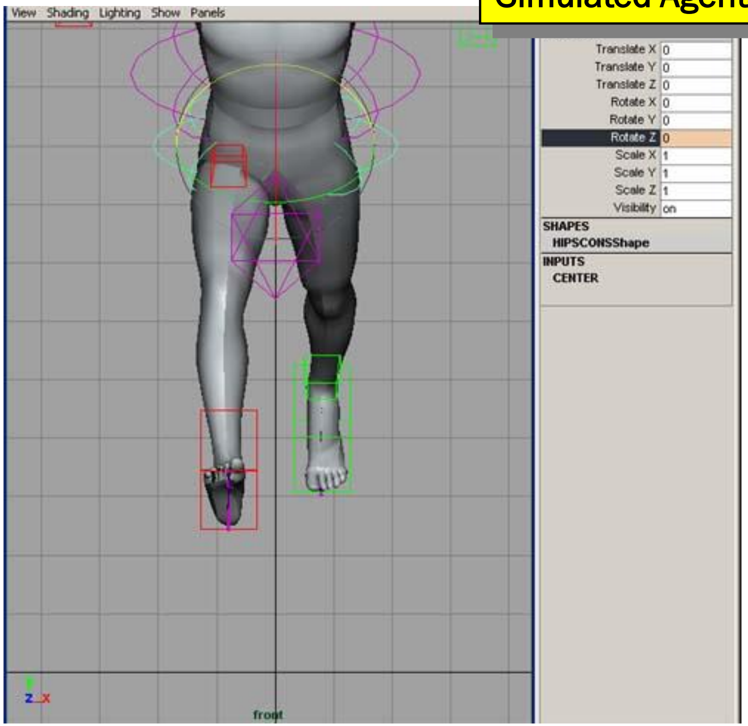
Maintenance

Beyond robot arms

Humanoid Robots



Simulated Agents



© 2022 The University of Sheffield

Inverse Kinematics define:

A. The robot's joint angles

B. The end-effector position

C. The error between current and desired positions

D. The transformation between frame of references

© 2022 The University of Sheffield

Summary of Kinematic Control

1. **Forward kinematics** uses the defined kinematic equations of a system and measured joint angles to define the position of the end effector.

2. **Inverse kinematics** uses the defined kinematic equations of a system and a desired end-effector position to derive the desired joint angles which can be used to define a motion plan

3. **Applications** extend beyond robot arms to humanoid robots and games

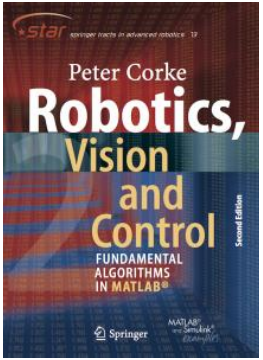
4. **Sensing, positional and controller errors** limit the applicability of these methods to simpler systems.

© 2022 The University of Sheffield

Where to find out more

1. **Advanced level course in University of Sheffield:**
   ACS329 Robotics Course

2. **Reading Materials:**
   Textbook: Peter Corke, Robotics, Vision and Control

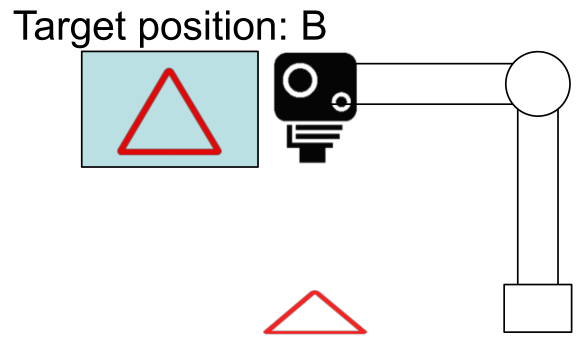


3. **Online Materials:**
   Robot Academy MOOC

© 2022 The University of Sheffield

Visual Servoing Concept

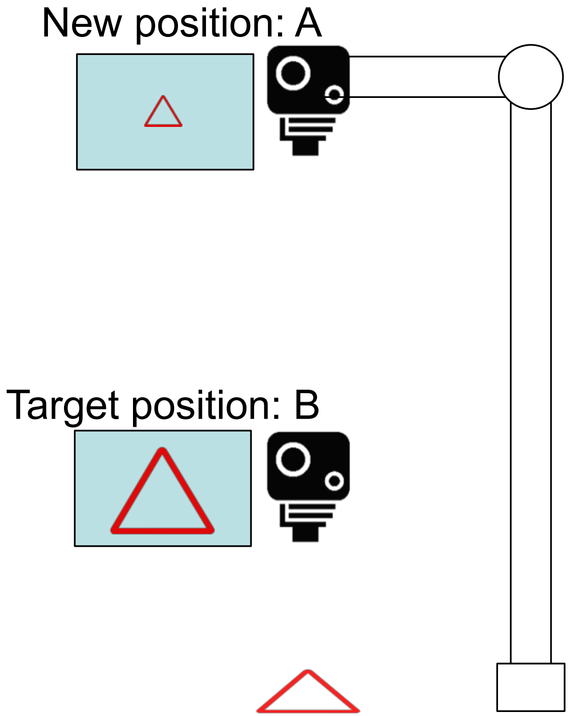
1. **Add a camera to the end-effector of the robot to *directly* observe the target**



© 2022 The University of Sheffield

Visual Servoing Concept

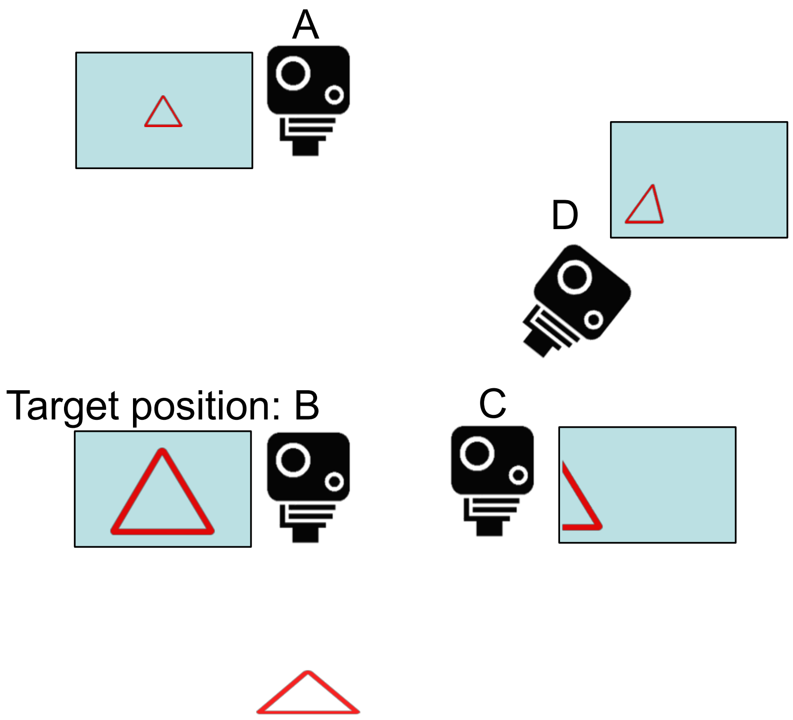
1. Add a camera to the end-effector of the robot to ***directly*** observe the target



Visual Servoing Concept

1. **Add a camera to the end-effector of the robot to *directly* observe the target**

2. **Minimizing the difference between stored and current views will solve positioning problem**



© 2022 The University of Sheffield

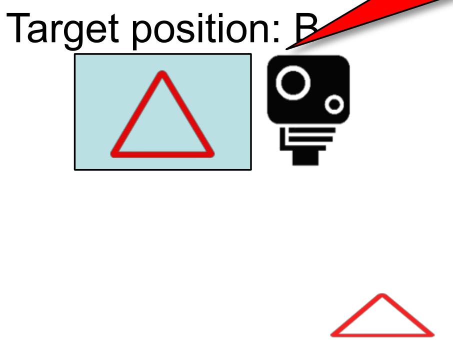
Visual Servoing Concept

1. **Add a camera to the end-effector of the robot to *directly* observe the target**

2. **Minimizing the difference between stored and current views will solve positioning problem**

3. **Bypasses many positioning, sensing and controller issues.**

Challenge:
How do we convert errors that are measured in 'pixel space' into corrective movement in 'joint space'?

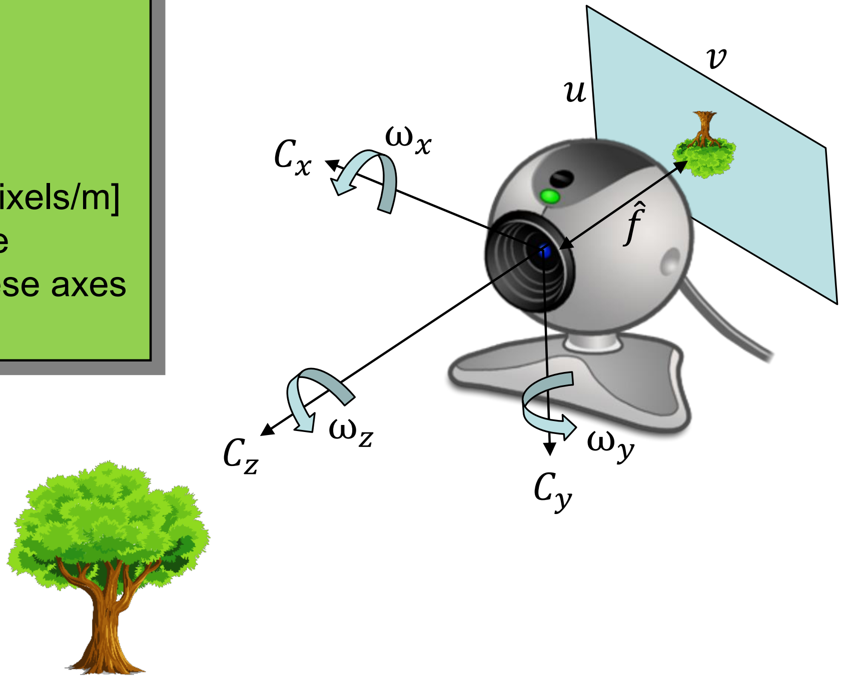


© 2022 The University of Sheffield

Machine Vision 101

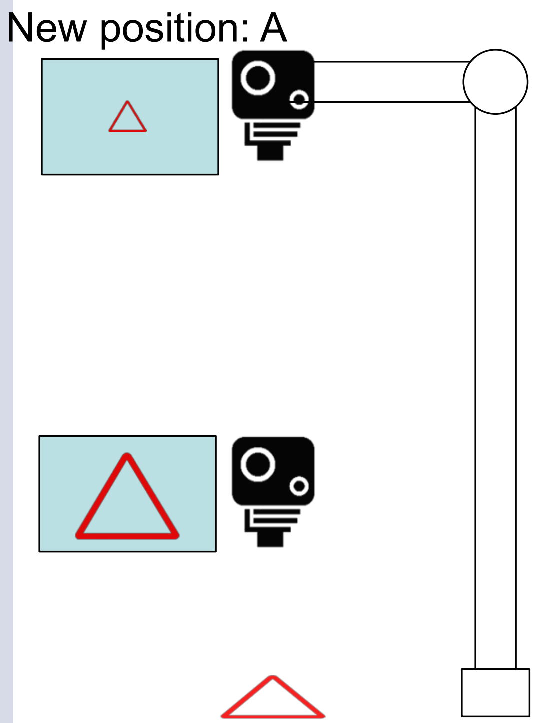
A pinhole camera will form an inverted image on a 2-D surface mounted behind the aperture. We define:

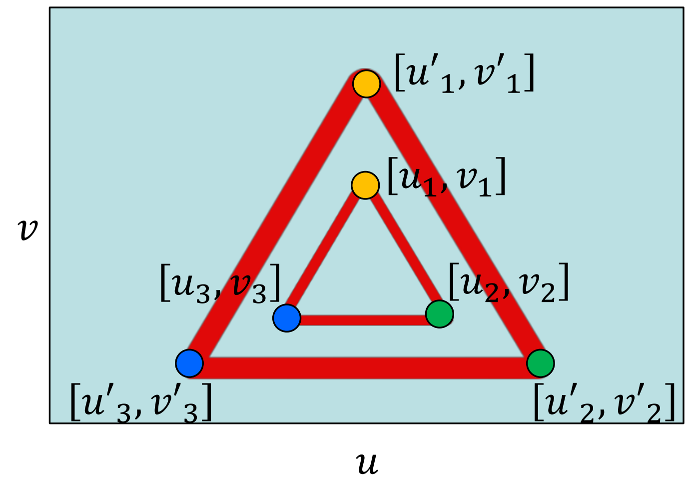
[u,v] - position of the feature in pixel space
$\hat{\boldsymbol{f}}$ - focal length in pixel space ($\hat{f} = {}^{f}/_{\rho}$)
[f=focal distance in m, ρ=pixel distance in pixels/m]
[$\boldsymbol{C_x} \quad \boldsymbol{C_y} \quad \boldsymbol{C_z}$**]** - camera position in 3D space
[$\boldsymbol{\omega_x} \quad \boldsymbol{\omega_y} \quad \boldsymbol{\omega_z}$**]** - angular rotations about these axes



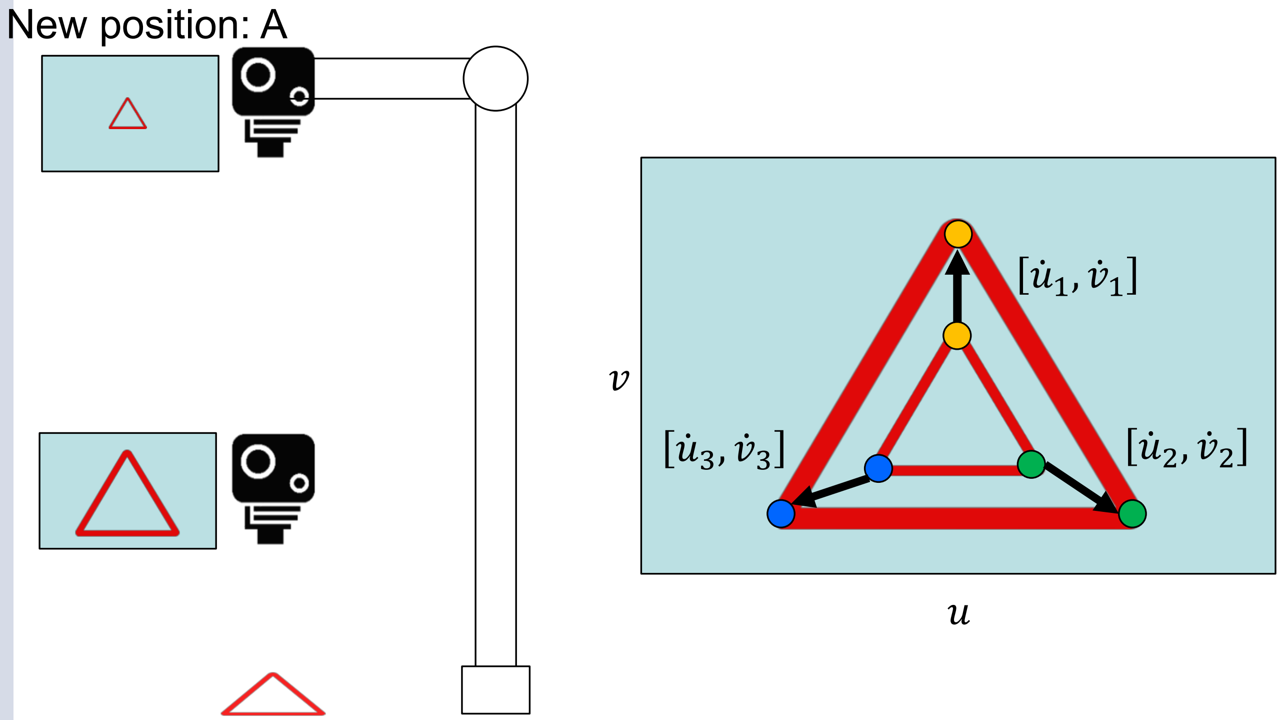
© 2022 The University of Sheffield

Visual Servoing Formalisation





Visual Servoing Formalisation

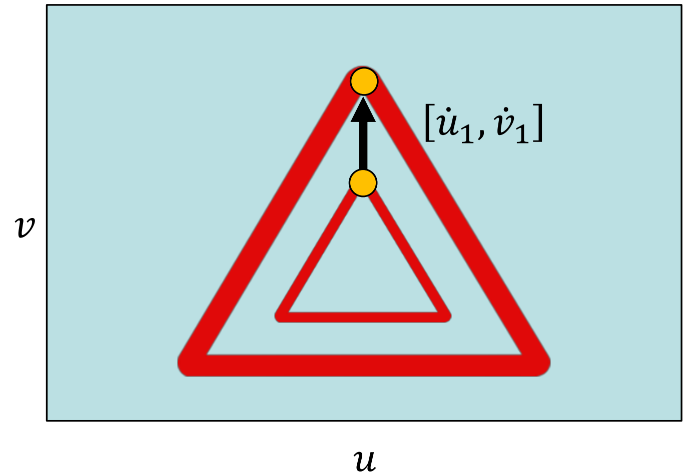


© 2022 The University of Sheffield

The Image Jacobian

The image Jacobian relates the velocity of the camera in 3D space to the velocity of the pixels in the image plane.

The relationship between pixel and camera velocities for 1 point is:



Camera velocities (v) along the camera x, y, and z axis: and rotational velocities (w) around those axis

Pixel velocity

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = J_p(u, v, Z) \begin{pmatrix} v_x \\ v_y \\ v_z \\ \varpi_x \\ \varpi_y \\ \varpi_z \end{pmatrix}$$

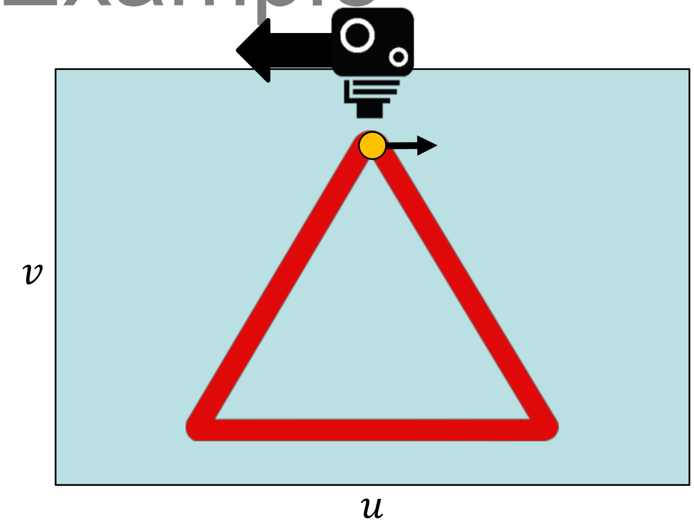
Image Jacobian

where $J_p(u, v, Z) =$

$$\begin{pmatrix} -\hat{f}/Z & 0 & u/Z & -uv/\hat{f} & -(\hat{f} + \frac{u^2}{\hat{f}}) & v \\ 0 & -\hat{f}/Z & v/Z & \hat{f} + v^2/\hat{f} & -uv/\hat{f} & -u \end{pmatrix}$$

© 2022 The University of Sheffield

Worked Example



Task:
For a camera with $\hat{f}$ = 1 what is the velocity of a pixel at [u,v]=[0,0], Z = 100cm when the camera moves left at 5 cm/s?

Solution:
Define camera motion

$$\begin{pmatrix} v_x \\ v_y \\ v_z \\ \varpi_x \\ \varpi_y \\ \varpi_z \end{pmatrix} = \begin{pmatrix} -5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Compute Jacobian

$$J_p(u, v, Z) = \begin{pmatrix} -0.01 & 0 & 0 & 0 & -1 & 0 \\ 0 & -0.01 & 0 & 1 & 0 & 0 \end{pmatrix}$$

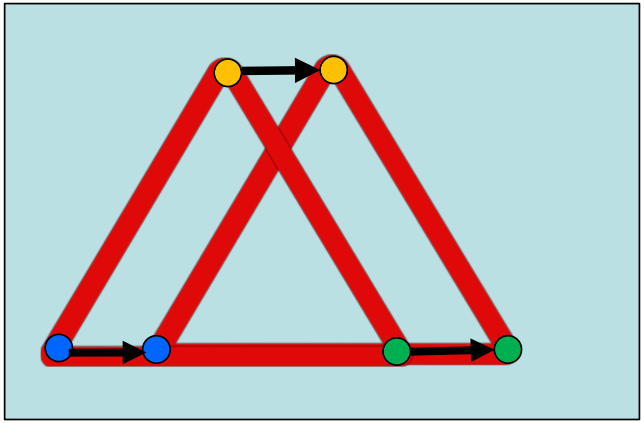
Compute pixel velocity

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} -0.01 & 0 & 0 & 0 & -1 & 0 \\ 0 & -0.01 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} -5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.05 \\ 0 \end{pmatrix}$$

Solving for camera velocity

The image Jacobian relates the velocity of the camera in 3D space to the velocity of the pixels in the image plane.

The relationship between pixel and camera velocities for 3 points is:



$$\begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \end{pmatrix} = \begin{pmatrix} \boldsymbol{J_p}(u_1, v_1, Z_1) \\ \boldsymbol{J_p}(u_2, v_2, Z_2) \\ \boldsymbol{J_p}(u_3, v_3, Z_3) \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ \varpi_x \\ \varpi_y \\ \varpi_z \end{pmatrix}$$

re-arranging for camera velocities

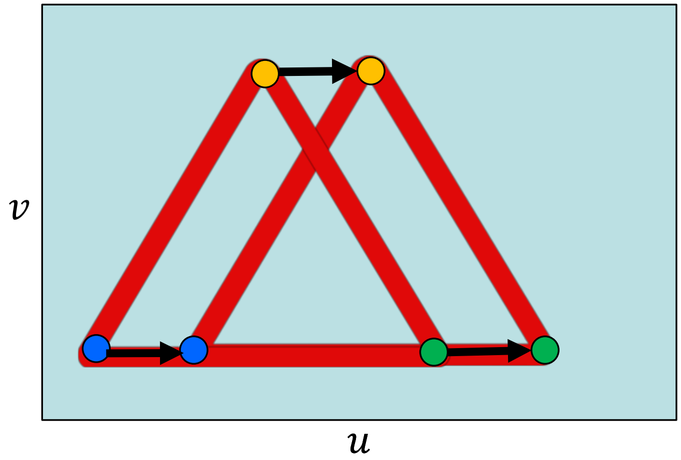
$$\begin{pmatrix} v_x \\ v_y \\ v_z \\ \varpi_x \\ \varpi_y \\ \varpi_z \end{pmatrix} = \begin{pmatrix} \boldsymbol{J_p}(u_1, v_1, Z_1) \\ \boldsymbol{J_p}(u_2, v_2, Z_2) \\ \boldsymbol{J_p}(u_3, v_3, Z_3) \end{pmatrix}^{-1} \begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \end{pmatrix}$$

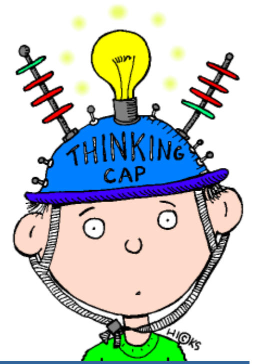
© 2022 The University of Sheffield

Homework

Task:
Your computer vision system has detected features at pixel locations:
$[u_1, v_1]$=[5,5],
$[u_2, v_2]$=[10,10],
$[u_3, v_3]$=[15,5].
The same features in the reference image are located at=:
$[u'_1, v'_1]$=[15,5],
$[u'_2, v'_2]$=[20,10],
$[u'_3, v'_3]$=[25,5].

For a camera with $\hat{f}$ = 1, and assuming all points are at Z = 100cm, use the Image Jacobian matrix to compute the camera movement required to return to the desired position.

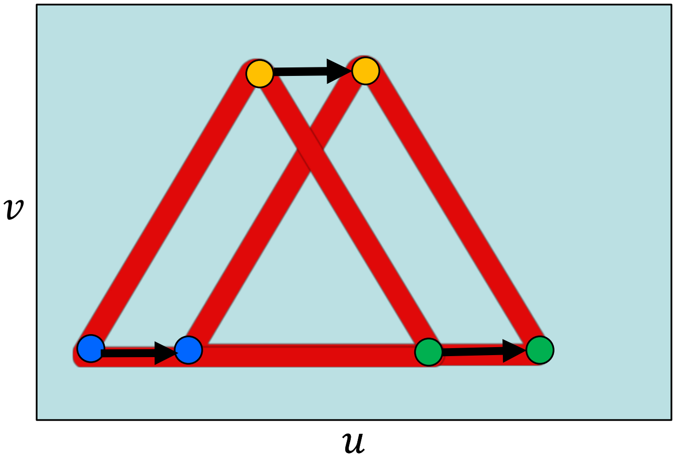




Homework

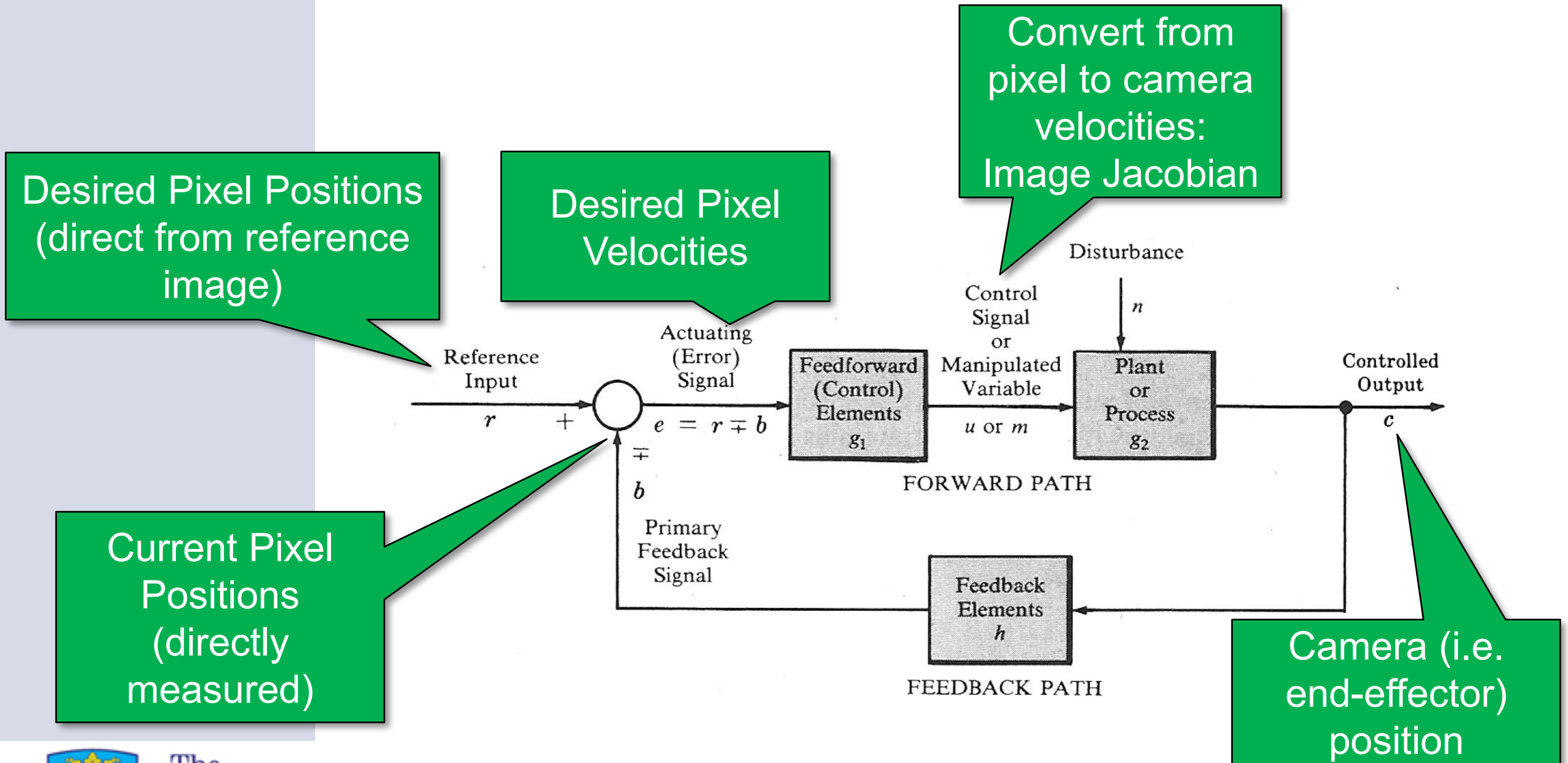
Task:
Your computer vision system has detected features at pixel locations:
$[u_1, v_1]$=[5,5],
$[u_2, v_2]$=[10,10],
$[u_3, v_3]$=[15,5].
The same features in the reference image are located at=:
$[u'_1, v'_1]$=[15,5],
$[u'_2, v'_2]$=[20,10],
$[u'_3, v'_3]$=[25,5].

For a camera with $\hat{f}$ = 1, and assuming all points are at Z = 100cm, use the Image Jacobian matrix to compute the camera movement required to return to the desired position.



Answer: $\begin{pmatrix} v_x \\ v_y \\ v_z \\ \varpi_x \\ \varpi_y \\ \varpi_z \end{pmatrix} = \begin{pmatrix} -1000 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

Visual Servoing Control Scheme



© 2022 The University of Sheffield

VisP Demo

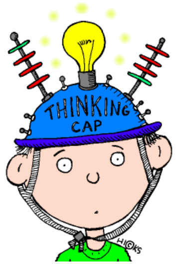
https://www.youtube.com/watch?v=4Se-_LIw51I

© 2022 The University of Sheffield

Which method of control is best suited to this robot arm?

A. Kinematic

B. Visual Servoing



Which method of control is best suited to this robot arm?

A. Kinematic

B. Visual Servoing

© 2022 The University of Sheffield

Summary of Visual Servoing

1. **Visual Servoing** is a method of robot control that seeks to minimise the error between *features* in a view stored a target position (B) and the same *features* in the current view (A).

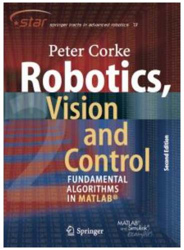
2. **The Image Jacobian** allows us to translate desired pixel velocities into camera velocities.

3. **As the camera directly measures** the position of the end effector and target position it requires no knowledge of its position and can recover sensing and positioning errors.

4. **Not perfect**: issues covered in next lecture

© 2022 The University of Sheffield

Where to find out more



1. **Reading Materials:**
   Textbook: Peter Corke, Robotics, Vision and Control

2. **3 seminal papers:**
   1. Hutchinson, Seth, Gregory D. Hager, and Peter I. Corke. "A tutorial on visual servo control." *IEEE transactions on robotics and automation* 12.5 (1996): 651-670.
   2. Chaumette, François, and Seth Hutchinson. "Visual servo control. I. Basic approaches." *IEEE Robotics & Automation Magazine* 13.4 (2006): 82-90.
   3. Chaumette, François, and Seth Hutchinson. "Visual servo control. II. Advanced approaches [Tutorial]." *IEEE Robotics & Automation Magazine* 14.1 (2007): 109-118.

3. **Online Materials:**
   [Robot Academy MOOC]

© 2022 The University of Sheffield

Visual Controlled Robot Grasping

Task:

- Move end-effector to position A
- Pick up object
- Move to position B

Method:

- Visual Control only

不一定抓得到

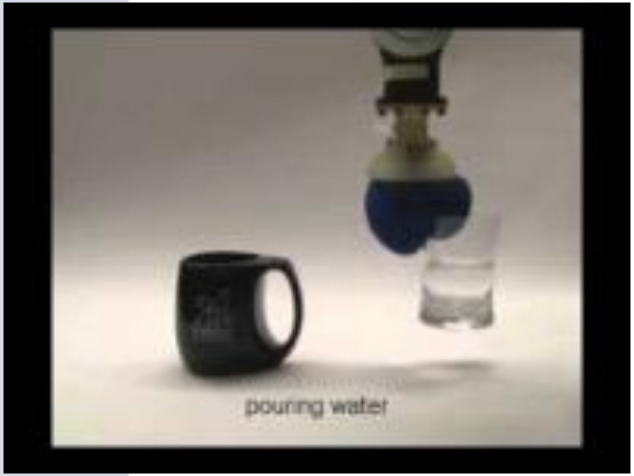
Result:

- Not good

© 2022 The University of Sheffield

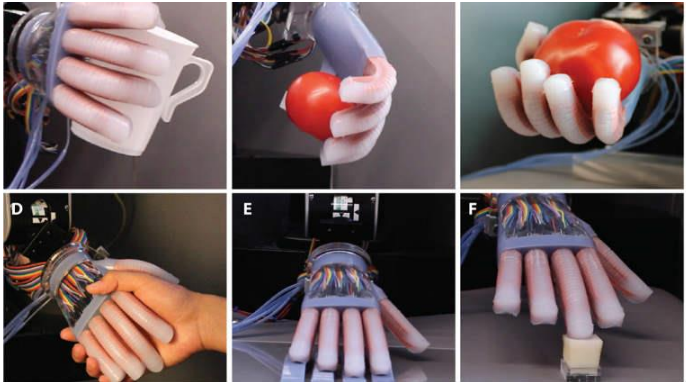
抓手

Soft grippers avoid issue



https://www.youtube.com/watch?v=86G9DLJEagw

Universal Gripper from iRobot



© 2022 The University of Sheffield

The importance of touch

Task:
- Pick up match
- Strike match
- Blow out flame

Method:
- Visual
- Touch

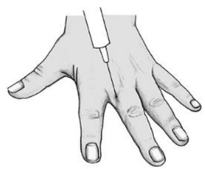
Result:
- **Good**

© 2022 The University of Sheffield

Visual Controlled Human Grasping

Task:
- Pick up match
- Strike match
- Blow out flame

Method:
- Visual Control only
- Anesthetized fingers:
  - Blocks all sense of touch from the finger
  - Does not affect motor control

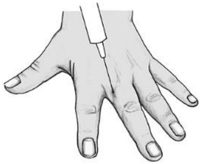


Result:
- Not good

Visual Controlled Human Grasping

Task:
- Pick up match
- Strike match
- Blow out flame

Method:
- Visual Control only
- Anesthetized fingers:
    - Blocks all sense of touch from the finger
    - Does not affect motor control



Patient with long-term neural dystrophy of touch

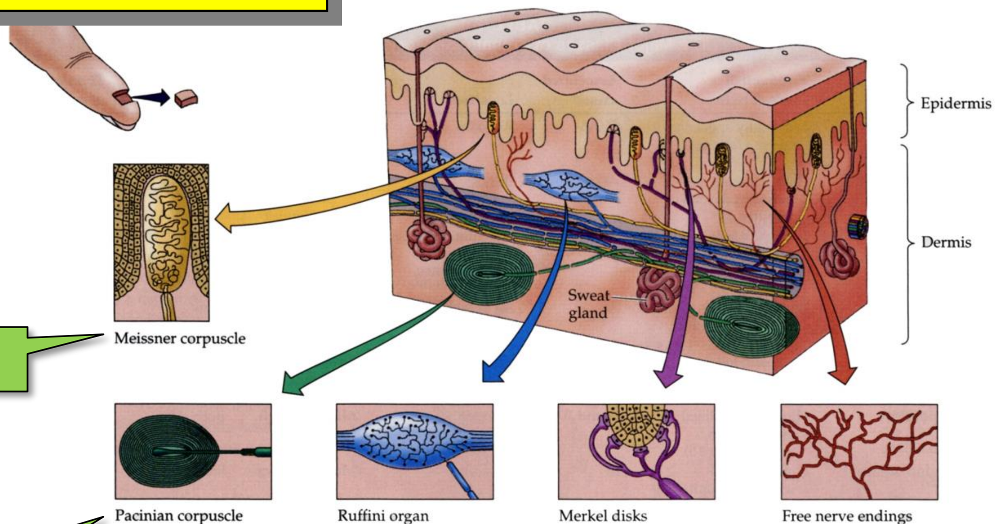
Humans don't recover over time

Result:
- Not good

© 2022 The University of Sheffield

Complicated Sensory System

Over 15,000 sensors in total in the human hand



1. Light touch

2. Sudden disturbances

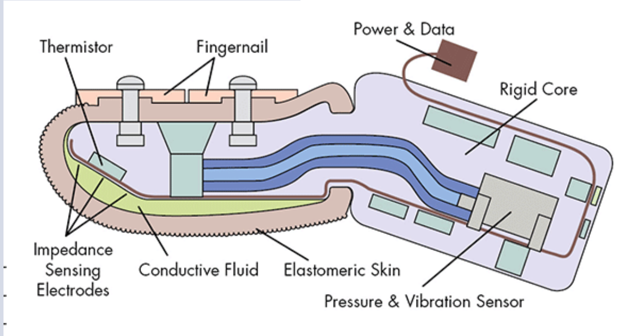
3. Stretch

4. Mechanical pressure

Neuroscience. 2nd edition.
Purves D, Augustine GJ, Fitzpatrick D, et al., editors.
Sunderland (MA): Sinauer Associates; 2001.

© 2022 The University of Sheffield

Adding touch to robot grippers



https://www.youtube.com/watch?time_continue=3&v=GJ_Zki8e8Kw

© 2022 The University of Sheffield

Which type of gripper?

An underwater salvage robot…

吸盘.

A. Suction cup 

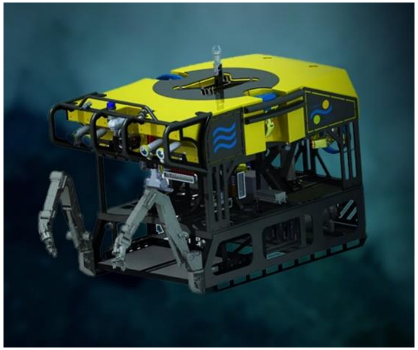
充气模具

B. Inflatable mould 

C. Standard Hook 

触觉手

D. Tactile Hand 



© 2022 The University of Sheffield

Which type of gripper?

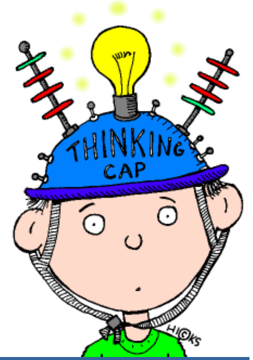
An repetitive pick and place task…

A. Suction cup 

B. Inflatable mould 

C. Standard Hook 

D. Tactile Hand 



© 2022 The University of Sheffield

Which type of gripper?

A prosthetic arm…

A. Suction cup 

B. Inflatable mould 

C. Standard Hook 

D. Tactile Hand 

© 2022 The University of Sheffield

Summary of Grasping

1. **Visual Control alone insufficient for gripping non-standardised objects**
2. **Soft-grippers can overcome some limitations**
3. **Touch is essential for human grasping but is in itself a complex system**
4. **Cutting edge grippers are adding touch to robot and prosthetic grippers**

© 2022 The University of Sheffield

Where to find out more

1. **Dr Hannes Saal, University of Sheffield:**
   Final year projects in this area
   https://www.sheffield.ac.uk/psychology/staff/academic/hannes_saal

2. **Reading Materials:**
   Dahiya RS, Metta G, Valle M, Sandini G. Tactile Sensing—From Humans to Humanoids. IEEE Trans Rob. 2010;26: 1–20.

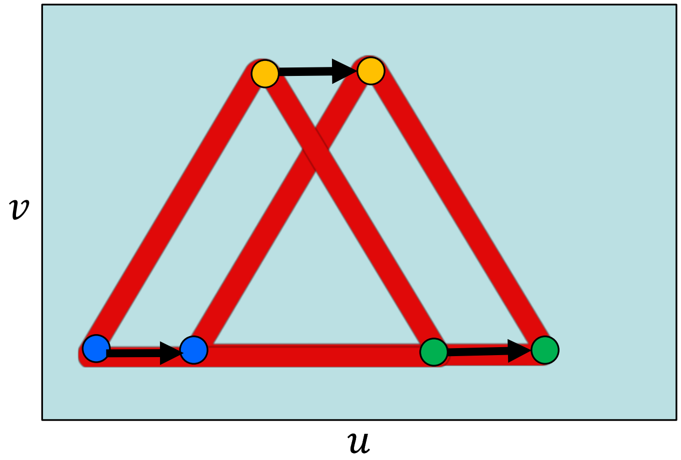
3. **Online Materials:**
   GRABlab at Yale:
   Jan Peters' lab at TU Darmstadt:

© 2022 The University of Sheffield

Homework - Solution

Task:
Your computer vision system has detected features at pixel locations:
$[u_1, v_1]$=[5,5],
$[u_2, v_2]$=[10,10],
$[u_3, v_3]$=[15,5].
The same features in the reference image are located at=:
$[u'_1, v'_1]$=[15,5],
$[u'_2, v'_2]$=[20,10],
$[u'_3, v'_3]$=[25,5].

For a camera with $\hat{f}$ = 1, and assuming all points are at Z = 100cm, use the Image Jacobian matrix to compute the camera movement required to return to the desired position.



Answer: $\begin{pmatrix} v_x \\ v_y \\ v_z \\ \varpi_x \\ \varpi_y \\ \varpi_z \end{pmatrix} = \begin{pmatrix} -1000 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

© 2022 The University of Sheffield

Homework - Solution

Step 1 – Compute the pixel velocities

$$\begin{pmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \end{pmatrix} = \begin{pmatrix} 15-5 \\ 5-5 \\ 20-10 \\ 10-10 \\ 25-15 \\ 5-5 \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \\ 10 \\ 0 \\ 10 \\ 0 \end{pmatrix}$$

Step 2 – Build the 6*6 image jacobian by concatenating the 3*[2*6] image jacobians as in Slide 29

$$Jacobian = \begin{bmatrix} -0.01 & 0.00 & 0.05 & -25 & -26 & 5 \\ 0.00 & -0.01 & 0.05 & 26 & -25 & -5 \\ -0.01 & 0.00 & 0.10 & -100 & -101 & 10 \\ 0.00 & -0.01 & 0.10 & 101 & -100 & -10 \\ -0.01 & 0.00 & 0.15 & -75 & -226 & 5 \\ 0.00 & -0.01 & 0.05 & 26 & -75 & -15 \end{bmatrix}$$

Homework - Solution

```
>> pixel_velocities

pixel_velocities =

    10
     0
    10
     0
    10
     0

>> image_jacobian

image_jacobian =

   -0.0100         0    0.0500  -25.0000  -26.0000    5.0000
         0   -0.1000    0.0500   26.0000  -25.0000   -5.0000
   -0.0100         0    0.1000 -100.0000 -101.0000   10.0000
         0   -0.0100    0.1000  101.0000 -100.0000  -10.0000
   -0.0100         0    0.1500  -75.0000 -226.0000    5.0000
         0   -0.0100    0.0500   26.0000  -75.0000  -15.0000
```

Step 3 – Inverting a 6*6 matrix manually is not the point of this exercise so solve programmatically.

```
>> camera_velocities=inv(image_jacobian)*pixel_velocities

camera_velocities =

   1.0e+03 *

   -1.0000
         0
   -0.0000
   -0.0000
    0.0000
         0
```