Control of industrial robots

Control with vision sensors

Prof. Paolo Rocco (paolo.rocco@polimi.it)
Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Visual measurements

Artificial vision devices are useful sensors for robotics because they mimic the human sense of sight and allow to gather information from the environment without contact. Nowadays several robotic controllers integrate vision systems.

The typical use of vision in industrial robotics is to detect an object in the robot’s scene, whose position (and orientation) is then used for online path planning in order to drive the robot to the identified object.

Online re-planning of the path can also be performed when the vision system detects some unexpected change in the path the robot is supposed to follow (for example a corner in a contouring task).

Alternatively, visual measurements can be used in a real time feedback loop in order to improve position control of the end effector: this is the concept of visual servoing.
Examples of use of vision

Picking sausages…

… a binpicking problem.
Examples of use of vision

Following a corner during a contouring task:

View from an external camera

View from the onboard camera
Examples of use of vision

A ball catching robot (example of visual servoing).
The camera is a device that can measure the intensity of light, concentrated by a lens on a plane, the image plane.
How an image is stored

The image plane contains a matrix of pixels (CCD: Charge Coupled Device). The light is “captured” in terms of:

- intensity only (gray scale)
- intensity and spectral components (RGB)

**Gray scale**: each pixel describes with a certain number of bits the scale from white to black.

**RGB**: for each pixel, and for each primary color (red, green, blue), a certain number of bits is used to express the scale in such color. If we have 8 bits for each color, we can express $256^3 = 16,777,216$ different colors.
The camera performs a 2D projection of the scene. This projection entails a loss of depth information: each point in the image plane corresponds to a ray in the 3D space.

In order to determine the 3D coordinates of a point corresponding to a 2D point in the plane additional information is needed:

- multiple views with a single camera
- multiple cameras
- knowledge of characteristic relations between relevant points of the framed objects
A point \( P \) with coordinates \((X, Y, Z)\) in the camera frame is projected into a point \( p \) with coordinates \((u, v)\) in the image plane, expressed in pixels.

From similarity of triangles:

\[
\frac{\lambda}{Z} = \frac{u}{X}, \quad \frac{\lambda}{Z} = \frac{v}{Y}
\]

\[
\xi = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{\lambda}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}
\]

\( \lambda \): focal length (in pixel)
In artificial vision we denote with *image feature* any characteristics that can be extracted from an image (e.g. an edge or a corner).

We then define a *parameter* of an image feature a quantity, expressed by a real numeric value, which can be computed from one or more image features. Parameters of an image feature can be gathered in a vector:

$$\xi = [\xi_1, \xi_2, \ldots, \xi_k].$$

Examples of parameters of image features:

- point coordinates
- length and orientation of a line connecting two points
- centroids and higher order moments
- parameters of an ellipse
Example: we want to extract pixel coordinates (features) of the corners of the top face of a cube which is on the table.

Original picture  Contour picture  Image features
(binary)  (points)

Pixels are colored white corresponding to high illumination change (gradient based edge detection)
The camera has to be calibrated before usage in a robotic vision system:

- **Internal calibration:**
  - Determination of the intrinsic parameters of the camera (like the focal length $\lambda$) as well as of some additional distortion parameters due to lens imperfections and misalignments in the optical system.

- **External calibration:**
  - Determination of the extrinsic parameters of the camera like the position and the orientation of the camera with respect to a reference frame.
3D vision

3D cameras return information on the depth as well

**Depth map**: the intensity of the pixel is proportional to the inverse of the distance.
The mostly adopted technology is based on the **time of flight**.

Phase lag is proportional to travel time (which is turn is proportional to distance).
Camera configuration

The first decision to be made when setting up a vision-based control system is where to place the camera.

The camera can be:

- mounted in a fixed location in the workspace (eye-to-hand configuration) so that it can observe the manipulator and any objects to be manipulated
- attached to the robot above the wrist (eye-in-hand configuration)
Eye-to-hand configuration

- **Advantages**
  - the field of view does not change as the manipulator moves
  - the geometric relationship between the camera and the workspace is fixed and can be calibrated offline

- **Disadvantages**
  - as the manipulator moves through the workspace it can occlude the camera’s field of view
Eye-in-hand configuration

- **Advantages**
  - the camera can observe the motion of the end effector at a fixed resolution and without occlusion as the manipulator moves through the workspace

- **Disadvantages**
  - the geometric relationship between the camera and the workspace changes as the manipulator moves
  - the field of view can change dramatically for even small motions of the manipulator
Robotic vision control systems can be classified based on various criteria. Basically, we have four options:

<table>
<thead>
<tr>
<th>Position based</th>
<th>Dynamic look and move</th>
<th>Visual servoing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot is position controlled, vision at a higher hierarchical level, image features are translated into 3D world coordinates</td>
<td>Vision is within the low-level control loop, image features are translated into 3D world coordinates</td>
<td></td>
</tr>
</tbody>
</table>

| Image based | | |
| Robot is position controlled, vision at a higher hierarchical level, image features are used directly | Vision is within the low-level control loop, image features are used directly |
Control architectures: classification

**Dynamic look and move:**
- the reduced sampling rate of the visual signal does not compromise the overall performance of the position control system
- in several industrial robot controllers it is only allowed to operate at the position setpoints level
- the robot can be seen as an ideal positioner in the Cartesian space, simplifying the design of the vision control system

**Visual servoing:**
- needs cameras with high sampling rate
- high performance can be achieved, but needs to directly access the robot actuators

**Position based control:**
- vision data are used to build a partial 3D representation of the world
- pose estimation algorithms are computationally intensive (a real-time implementation is required) and sensitive to errors in camera calibration

**Image based control:**
- uses the image data directly to control the robot motion
- an error function is defined in terms of quantities that can be directly measured in an image, and a control law is constructed that maps this error directly to robot motion
Position based vs. Image based

Two examples:

Position based

Image based
Position-based look-and-move

\[ \mathbf{c} \mathbf{x}_d + \quad \text{Cartesian control law} \]

\[ \mathbf{x} \]

\[ \mathbf{x} \]

\[ \xi \]

\[ \mathbf{c} \]

\[ \text{Pose estimation} \]

\[ \text{Image feature extraction} \]

Drives

Video

DIFFICULT
Image-based look-and-move

\[ \xi_d \]

Control law in the image feature space

Axis controllers

Drives

Difficult

Image feature extraction

Video
Position-based visual servoing

Cartesian control law

Drives

Pose estimation

Image feature extraction

DIFFICULT

Video
Image-based visual servoing

\[ \xi_d \]

**DIFFICULT**

Control law in the image feature space

Drives

\[ \xi \]

Image feature extraction

Video

Control of industrial robots – Control with vision sensors – Paolo Rocco
Examples (position-based look-and-move)

Adept feeder mechanism prepares pick surface
Image-based schemes

To solve an image-based scheme, we need to relate motion of the camera with motion of the features in the image plane:

\[
\begin{bmatrix}
\dot{O}_c \\
\omega_c
\end{bmatrix}
\quad \leftrightarrow 
\begin{bmatrix}
\dot{u} \\
\dot{v}
\end{bmatrix}
\]

Linear and angular velocities of the camera frame
Linear velocity of the point feature in the image

We will end up with the notion of interaction matrix (and of image Jacobian)
Consider a moving camera observing a point fixed in space:

We have:

\[ P^w = R_c^w(t)P^c(t) + O_c^w(t) \]

or:

\[ P^c(t) = R_c^w(t)^T\left(P^w - O_c^w(t)\right) \]

Differentiating wrt time:

\[ \dot{P}^c = -\omega_c^c \times P^c - \dot{O}_c^c \]

(can be easily proven…)

Kinematic relations
We have then obtained this fundamental relation:

\[ \dot{P} = -\omega_c \times P - \dot{O}_c \]

where all vectors are expressed in the camera frame.
Kinematic relations

Define:

\[
P = \begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}, \quad \omega_c = \begin{bmatrix}
\omega_x \\
\omega_y \\
\omega_z
\end{bmatrix}, \quad \dot{O}_c = \begin{bmatrix}
\dot{O}_x \\
\dot{O}_y \\
\dot{O}_z
\end{bmatrix}
\]

The previous relation can be expressed in terms of scalar equations:

\[
\dot{X} = Y\omega_z - Z\omega_y - \dot{O}_x
\]
\[
\dot{Y} = Z\omega_x - X\omega_z - \dot{O}_y
\]
\[
\dot{Z} = X\omega_y - Y\omega_x - \dot{O}_z
\]

From the perspective geometry:

\[
X = \frac{u}{\lambda} Z, \quad Y = \frac{v}{\lambda} Z
\]
Kinematic relations

By substitution we obtain:

\[ \dot{X} = \frac{v}{\lambda} Z\omega_z - Z\omega_y - \dot{O}_x \]

\[ \dot{Y} = Z\omega_x - \frac{u}{\lambda} Z\omega_z - \dot{O}_y \]

\[ \dot{Z} = \frac{u}{\lambda} Z\omega_y - \frac{v}{\lambda} Z\omega_x - \dot{O}_z \]

We can also take the derivative of the equations of the perspective geometry:

\[ \dot{u} = \lambda \frac{d}{dt} \frac{X}{Z} = \lambda \frac{Z\dot{X} - X\dot{Z}}{Z^2} \]

\[ \dot{v} = \lambda \frac{d}{dt} \frac{Y}{Z} = \lambda \frac{Z\dot{Y} - Y\dot{Z}}{Z^2} \]
Combining the previous equations we finally obtain:

$$
\begin{bmatrix}
\dot{u} \\
\dot{v}
\end{bmatrix} = L(\lambda, u, v, Z)
\begin{bmatrix}
\dot{O}_c \\
\omega_c
\end{bmatrix}
$$

where the matrix:

$$
L(\lambda, u, v, Z) = \begin{bmatrix}
-\frac{\lambda}{Z} & 0 & u & \frac{uv}{\lambda} & -\frac{\lambda^2 + u^2}{\lambda} & v \\
0 & -\frac{\lambda}{Z} & \frac{v}{Z} & \frac{\lambda^2 + v^2}{\lambda} & -\frac{uv}{\lambda} & -u
\end{bmatrix}
$$

is called the interaction matrix.

It relates the linear and angular velocities of the camera to the velocity in the image plane.
Interaction matrix

\[
L(\lambda, u, v, Z) = \begin{bmatrix}
-\frac{\lambda}{Z} & 0 & \frac{u}{Z} & \frac{uv}{\lambda} & -\frac{\lambda^2 + u^2}{\lambda} & v \\
0 & -\frac{\lambda}{Z} & \frac{v}{Z} & \frac{\lambda^2 + v^2}{\lambda} & -\frac{uv}{\lambda} & -u
\end{bmatrix}
\]

The interaction matrix:

1. is a $2 \times 6$ rectangular matrix
2. depends on the actual values of the features $u$ and $v$ and on the depth $Z$ (merely as a scale factor)
3. can be decomposed in two submatrices

\[
\begin{bmatrix}
\dot{u} \\
\dot{v}
\end{bmatrix} = L_0(\lambda, u, v, Z)\dot{\omega}_c + L_\omega(\lambda, u, v)\omega_c
\]

does not depend on the depth $Z$
Null space of the interaction matrix

Since the interaction matrix is 2 X 6, the null-space has dimensions 4, which means that there are $\infty^4$ motions of the camera that do not produce any motion of the feature in the image plane.

It can be proven that the null space is spanned by the four vectors:

$$
\begin{bmatrix}
  u \\
  v \\
  \lambda \\
  0
\end{bmatrix}
\text{ and }
\begin{bmatrix}
  0 \\
  0 \\
  0 \\
  \lambda
\end{bmatrix}
$$

$$
\begin{bmatrix}
  uvZ \\
  -\left(\lambda^2 + u^2\right)Z \\
  \lambda vZ \\
  -\lambda^2
\end{bmatrix}
\text{ and }
\begin{bmatrix}
  \lambda\left(\lambda^2 + u^2 + v^2\right)Z \\
  0 \\
  -u\left(\lambda^2 + u^2 + v^2\right)Z \\
  -\left(\lambda^2 + u^2\right)Z
\end{bmatrix}
$$

Motion of the camera frame along the projection ray that contains point $P$

Rotation of the camera frame about a projection ray that contains the point $P$
Null space of the interaction matrix

Motion of the camera frame along the projection ray that contains point P

Rotation of the camera frame about a projection ray that contains the point P
Multiple feature points

The definition of interaction matrix can be easily extended to the case of the coordinates of \( n \) image points. We define the feature vector \( \xi \) and the vector of depth values \( Z \) as:

\[
\xi = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \end{bmatrix}, \quad Z = \begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix}
\]

The interaction matrix is obtained by stacking the \( n \) interaction matrices for the \( n \) individual feature points:

\[
L_c(\lambda, \xi, Z) = \begin{bmatrix} L_1(\lambda, u_1, v_1, Z_1) \\ \vdots \\ L_n(\lambda, u_n, v_n, Z_n) \end{bmatrix}
\]
We can now relate the motion of the feature point to the motion of the robot in joint space:

\[
\begin{bmatrix}
\dot{u} \\
\dot{v}
\end{bmatrix} = L(\lambda, u, v, Z)T(q)J(q)\dot{q}
\]

We can write:

\[
\begin{bmatrix}
\dot{u} \\
\dot{v}
\end{bmatrix} = J_I(\lambda, u, v, Z, q)\dot{q}
\]

where the matrix:

\[
J_I(\lambda, u, v, Z, q) = L(\lambda, u, v, Z)T(q)J(q)
\]

is called the **image Jacobian**.
Dependence on the depth $Z$

The image Jacobian $J_i$ depends on the depth $Z$:

$$
\begin{bmatrix}
\dot{u} \\
\dot{v}
\end{bmatrix} = J_i(\lambda, u, v, Z, q) \dot{q}
$$

This information is clearly not available, but it can be estimated in several ways:

- using the desired goal position
- using geometrical knowledge of the scene
- using the geometrical knowledge of the object

Suitable observers can be setup
Kinematic control law

A control law can be now devised based on the image Jacobian:

$$\dot{\mathbf{q}} = J_1^\# \left( \dot{\xi}_d + K (\xi_d - \xi) \right)_+ \left( I - J_1^\# J_1 \right) \mathbf{q}_0$$

Minimum norm solution

Term projected in the null-space of the Jacobian (does not move the features)

\[\xi_d\]
The **Machine Vision Toolbox** by Peter Corke provides many functions that are useful in machine vision and vision-based control:

http://petercorke.com/wordpress/toolboxes/machine-vision-toolbox/

Combined with the **Robotics Toolbox** by the same author, it allows to simulate vision-based control systems for robots:

http://petercorke.com/wordpress/toolboxes/robotics-toolbox/
The robot is an ideal positioner

Four points in space are assigned: the block *camera*, based on the current position/orientation of a camera, returns the image features of the four points

These are compared to the desired features

A block is available that computes the interaction matrix. A pseudo-inverse of such matrix is then computed
An example of IBVS scheme

Time histories of the feature errors

Time histories of the camera position coordinates