Models and Control Strategies for Visual Servoing

Nils T Siebel, Dennis Peters and Gerald Sommer
Christian-Albrechts-University of Kiel
Germany

1. Introduction

Visual servoing is the process of steering a robot towards a goal using visual feedback in a closed control loop as shown in Figure 1. The *output* u_n of the controller is a *robot movement* which steers the robot towards the goal. The state x_n of the system cannot be directly observed. Instead a visual measurement process provides feedback data, the vector of *current image features* y_n. The *input* to the controller is usually the difference between desired ($y^\star$) and actual values of this vector—the *image error* vector Δy_n.

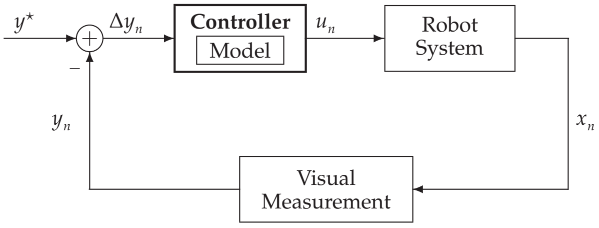


Fig. 1. Closed-loop image-based visual servoing control

In order for the controller to calculate the necessary robot movement it needs two main components:

1. a *model* of the environment—that is, a model of how the robot/scene will change after issuing a certain control commmand; and
2. a *control law* that governs how the next robot command is determined given current image measurements and model.

In this chapter we will look in detail on the effects different models and control laws have on the properties of a visual servoing controller. Theoretical considerations are combined with experiments to demonstrate the effects of popular models and control strategies on the behaviour of the controller, including convergence speed and robustness to measurement errors.

2. Building Models for Visual Servoing

2.1 Task Description

The aim of a visual servoing controller is to move the end-effector of one or more robot arms such that their configuration in relation to each other and/or to an object fulfils certain task-specific conditions. The feedback used in the controller stems from visual data, usually taken

Source: Visual Servoing, Book edited by: Rong-Fong Fung,
ISBN 978-953-307-095-7, pp. 234, April 2010, INTECH, Croatia, downloaded from SCIYO.COM

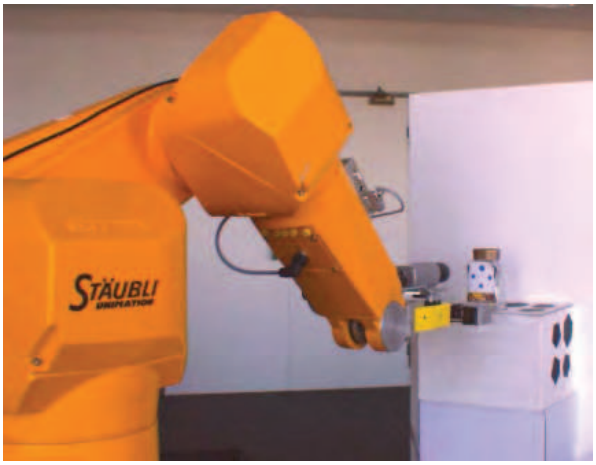


Fig. 2. Robot Arm with Camera and Object

from one or more cameras mounted to the robot arm and/or placed in the environment. A typical configuration is shown in Figure 2. Here a camera is mounted to the robot's gripper ("eye-in-hand" setup), looking towards a glass jar. The controller's task in this case is to move the robot arm such that the jar can be picked up using the gripper. This is the case whenever the visual appearance of the object in the image has certain properties. In order to detect whether these properties are currently fulfilled a camera image can be taken and image processing techniques applied to extract the image positions of object markings. These image positions make up the *image feature vector*.
Since the control loop uses visual data the goal configuration can also be defined in the image. This can be achieved by moving the robot and/or the object in a suitable position and then acquiring a camera image. The image features measured in this image can act as *desired image features*, and a comparison of actual values at a later time to these desired values ("*image error*") can be used to determine the degree of agreement with the desired configuration. This way of acquiring desired image features is sometimes called "*teaching by showing*".
From a mathematical point of view, a successful visual servoing control process is equivalent to solving an optimisation problem. In this case a measure of the image error is minimised by moving the robot arm in the space of possible configurations. Visual servoing can also be regarded as practical feedback stabilisation of a dynamical system.

2.2 Modelling the Camera-Robot System

2.2.1 Preliminaries

The *pose* of an object is defined as its position and orientation. The *position* in 3D Euclidean space is given by the 3 Cartesian coordinates. The *orientation* is usually expressed by 3 angles, i.e. the rotation around the 3 coordinate axes. Figure 3 shows the notation used in this chapter, where *yaw*, *pitch* and *roll* angles are defined as the mathematically positive rotation around the x, y and z axis. In this chapter we will use the $\{\cdot\}$-notation for a *coordinate system*, for example $\{W\}$ will stand for the world coordinate system. A variable coordinate system—one which changes its pose to over time—will sometimes be indexed by the *time index* $n \in \mathbb{N} =$

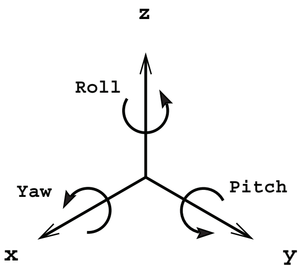


Fig. 3. Yaw, pitch and roll

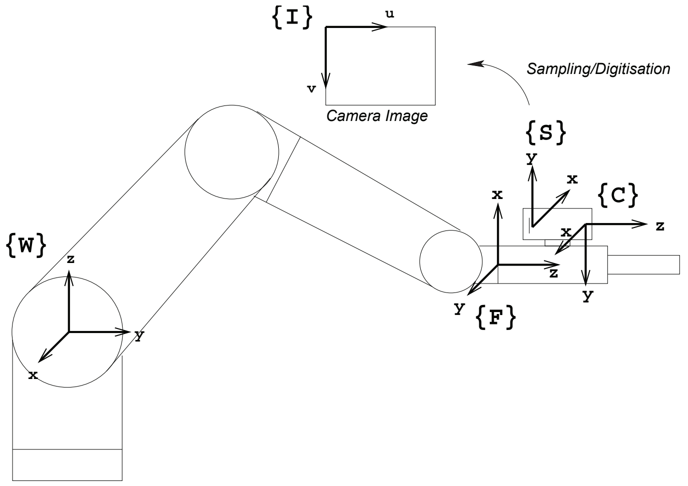


Fig. 4. World, Flange, Camera, Sensor and Image coordinate systems

$0, 1, 2, \ldots$. An example is the camera coordinate system $\{C_n\}$, which moves relative to $\{W\}$ as the robot moves since the camera is mounted to its hand.

Figure 4 lists the coordinate systems used for modelling the camera-robot system. The *world coordinate system* $\{W\}$ is fixed at the robot base, the *flange coordinate system* $\{F\}$ (sometimes called "*tool coordinate system*", but this can be ambiguous) at the flange where the hand is mounted. The *camera coordinate system* $\{C\}$ (or $\{C_n\}$ at a specific time n) is located at the optical centre of the camera, the *sensor coordinate system* $\{S\}$ in the corner of its CCD/CMOS chip (sensor); their orientation and placement is shown in the figure. The *image coordinate system* which is used to describe positions in the digital image is called $\{I\}$. It is the only system to use pixel as its unit; all other systems use the same length unit, e.g. mm.

Variables that contain coordinates in a particular coordinate system will be marked by a superscript left of the variable, e.g. ${}^{A}x$ for a vector $x \in \mathbb{R}^n$ in $\{A\}$-coordinates. The coordinate transform which transforms a variable from a coordinate system $\{A\}$ to another one, $\{B\}$, will be written ${}^{B}_{A}\mathrm{T}$. If ${}^{A}x$ and ${}^{B}x$ express the pose of the same object then

$$ {}^{A}x = {}^{A}_{B}\mathrm{T}\ {}^{B}x, \qquad \text{and always} \qquad {}^{A}_{B}\mathrm{T} = \left({}^{B}_{A}\mathrm{T}\right)^{-1}. \tag{1} $$

The *robot's pose* is defined as the pose of $\{F\}$ in $\{W\}$.

2.2.2 Cylindrical Coordinates

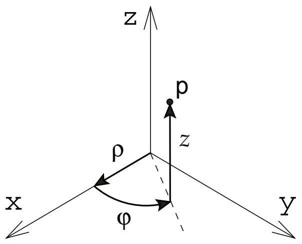


Fig. 5. A point $p = (\rho, \varphi, z)$ in cylindrical coordinates.

An alternative way to describe point positions is by using a cylindrical coordinate system as the one in Figure 5. Here the position of the point p is defined by the distance ρ from a fixed axis (here aligned with the Cartesian z axis), an angle φ around the axis (here $\varphi = 0$ is aligned with the Cartesian x axis) and a height z from a plane normal to the z axis (here the plane spanned by x and y). Using the commonly used alignment with the Cartesian axes as in Figure 5 converting to and from cylindrical coordinates is easy. Given a point $p = (x, y, z)$ in Cartesian coordinates, its cylindrical coordinates $p = (\rho, \varphi, z) \in \mathbb{R} \times]-\pi, \pi] \times \mathbb{R}$ are as follows:

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2} \\ \varphi &= \operatorname{atan2}(y, x) \\ &\overset{\star}{=} \begin{cases} 0 & \text{if } x = 0 \text{ and } y = 0 \\ \arcsin(\frac{y}{\rho}) & \text{if } x \geq 0 \\ \arcsin(\frac{y}{\rho}) + \pi & \text{if } x < 0 \end{cases} \\ z &= z, \end{aligned} \tag{2}$$

($\star$ up to multiples of 2π), and, given a point $p = (\rho, \varphi, z)$ in cylindrical coordinates:

$$\begin{aligned} x &= \rho \cos \varphi \\ y &= \rho \sin \varphi \\ z &= z. \end{aligned} \tag{3}$$

2.2.3 Modelling the Camera

A simple and popular approximation to the way images are taken with a camera is the *pinhole camera model* (from the pinhole camera/camera obscura models by Ibn al-Haytham "Alhacen", 965–1039 and later by Gérard Desargues, 1591–1662), shown in Figure 6. A light ray from an object point passes an aperture plate through a very small hole ("pinhole") and arrives at the sensor plane, where the camera's CCD/CMOS chip (or a photo-sensitive film in the 17th century) is placed. In the digital camera case the sensor elements correspond to picture elements ("pixels"), and are mapped to the image plane. Since pixel positions are stored in the computer as unsigned integers the centre of the $\{I\}$ coordinate system in the image plane is shifted to the upper left corner (looking towards the object/monitor). Therefore the centre ${}^{I}c \neq (0, 0)^T$.

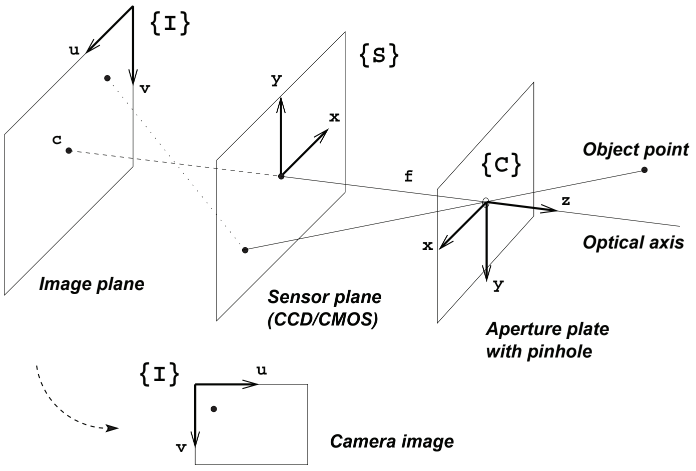


Fig. 6. Pinhole camera model

Sometimes the sensor plane is positioned in front of the aperture plate in the literature (e.g. in Hutchinson et al., 1996). This has the advantage that the x- and y-axis of $\{S\}$ can be (directionally) aligned with the ones in $\{C\}$ and $\{I\}$ while giving identical coordinates. However, since this alternative notation has also the disadvantage of being less intuitive, we use the one defined above.

Due to the simple model of the way the light travels through the camera the object point's position in $\{C\}$ and the coordinates of its projection in $\{S\}$ and $\{I\}$ are proportional, with a shift towards the new centre in $\{I\}$. In particular, the *sensor coordinates* ${}^{S}p = ({}^{S}x, {}^{S}y)^T$ of the image of an object point ${}^{C}p = ({}^{C}x, {}^{C}y, {}^{C}z)^T$ are given as

$$ {}^{S}x = \frac{{}^{C}x \cdot f}{{}^{C}z} \qquad \text{and} \qquad {}^{S}y = \frac{{}^{C}y \cdot f}{{}^{C}z}, \tag{4} $$

where f is the distance the aperture plate and the sensor plane, also called the "*focal length*" of the camera/lens.

The pinhole camera model's so-called "*perspective projection*" is not an exact model of the projection taking place in a modern camera. In particular, lens distortion and irregularities in the manufacturing (e.g. slightly tilted CCD chip or positioning of the lenses) introduce deviations. These modelling errors may need to be considered (or, corrected by a lens distortion model) by the visual servoing algorithm.

2.3 Defining the Camera-Robot System as a Dynamical System

As mentioned before, the camera-robot system can be regarded as a dynamical system. We define the *state* x_n of the robot system at a *time step* $n \in \mathbb{N}$ as the current robot pose, i.e. the pose of the flange coordinate system $\{F\}$ in world coordinates $\{W\}$. $x_n \in \mathbb{R}^6$ will contain the position and orientation in the x, y, z, yaw, pitch, roll notation defined above. The set of possible robot poses is $\mathcal{X} \subset \mathbb{R}^6$. The *output* of the system is the *image feature vector* y_n. It contains pairs of image coordinates of object markings viewed by the camera, i.e. $({}^{S}x_1, {}^{S}y_1, \ldots, {}^{S}x_M, {}^{S}y_M)^T$ for $M = \frac{m}{2}$ object markings (in our case $M = 4$, so $y_n \in \mathbb{R}^8$).

Let $\mathcal{Y} \subset \mathbb{R}^m$ be the set of possible output values. The *output (measurement) function* is $\eta : \mathcal{X} \to \mathcal{Y}$, $x_n \mapsto y_n$. It contains the whole measurement process, including projection onto the sensor, digitisation and image processing steps.
The *input (control) variable* $u_n \in \mathcal{U} \subset \mathbb{R}^6$ shall contain the desired pose change of the camera coordinate system. This robot movement can be easily transformed to a new robot pose $\tilde{u}_n$ in $\{W\}$, which is given to the robot in a move command. Using this definition of u_n an input of $(0,0,0,0,0,0)^T$ corresponds to no robot movement, which has advantages, as we shall see later. Let $\varphi : \mathcal{X} \times \mathcal{U} \to \mathcal{X}, (x_n, u_n) \mapsto x_{n+1}$ be the corresponding *state transition (next-state) function*.

With these definitions the camera-robot system can be defined as a time invariant, time discrete input-output system:

$$\begin{aligned} x_{n+1} &= \varphi\left(x_n, u_n\right) \\ y_n &= \eta\left(x_n\right). \end{aligned} \tag{5}$$

When making some mild assumptions, e.g. that the camera does not move relative to $\{F\}$ during the whole time, the state transition function φ can be calculated as follows:

$$\begin{aligned} \varphi(x_n, u_n) = x_{n+1} = {}^{W}x_{n+1} = {}^{W}\tilde{u}_n &\mathrel{\hat{=}} {}^{\phantom{F_{n+1}}W}_{F_{n+1}}\mathrm{T} \\ &= \underbrace{{}^{W}_{F_n}\mathrm{T}}_{\hat{=} x_n} \circ \underbrace{{}^{F_n}_{C_n}\mathrm{T}}_{\star} \circ \underbrace{{}^{C_n}_{C_{n+1}}\mathrm{T}}_{\hat{=} u_n} \circ \underbrace{{}^{C_{n+1}}_{F_{n+1}}\mathrm{T}}_{\star}, \end{aligned} \tag{6}$$

where $\{F_n\}$ is the flange coordinate system at time step n, etc., and the $\hat{=}$ operator expresses the equivalence of a pose with its corresponding coordinate transform.
$\star$ = *external ("extrinsic") camera parameters*; ${}^{T_n}_{C_n}\mathrm{T} = {}^{T_{n+1}}_{C_{n+1}}\mathrm{T} = \left({}^{C_{n+1}}_{T_{n+1}}\mathrm{T}\right)^{-1} \quad \forall n \in \mathbb{N}$.

For $m = 2$ image features corresponding to coordinates $({}^{S}x, {}^{S}y)$ of a projected object point ${}^{W}p$ the equation for η follows analogously:

$$\begin{aligned} \eta(x) = y = {}^{S}y &= {}^{S}_{C}\mathrm{T}\ {}^{C}p \\ &= {}^{S}_{C}\mathrm{T} \circ {}^{C}_{T}\mathrm{T} \circ {}^{T}_{W}\mathrm{T}\ {}^{W}p, \end{aligned} \tag{7}$$

where ${}^{S}_{C}\mathrm{T}$ is the mapping of the object point ${}^{C}p$ depending on the focal length f according to the pinhole camera model / perspective projection defined in (4).

2.4 The Forward Model—Mapping Robot Movements to Image Changes

In order to calculate necessary movements for a given desired change in visual appearance the relation between a robot movement and the resulting change in the image needs to be modelled. In this section we will analytically derive a *forward model*, i.e. one that expresses image changes as a function of robot movements, for the eye-in-hand setup described above. This forward model can then be used to predict changes effected by controller outputs, or (as it is usually done) simplified and then inverted. An *inverse model* can be directly used to determine the controller output given actual image measurements.
Let $\Phi : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ the function that expresses the system output y depending on the state x and the input u:

$$\Phi(x, u) := \eta \circ \varphi(x, u) = \eta(\varphi(x, u)). \tag{8}$$

For simplicity we also define the function which expresses the behaviour of $\Phi(x_n, \cdot)$ at a time index n, i.e. the dependence of image features on the camera movement u:

$$\Phi_n(u) := \Phi(x_n, u) = \eta(\varphi(x_n, u)). \tag{9}$$

This is the forward model we wish to derive.
Φ_n depends on the camera movement u and the current system state, the robot pose x_n. In particular it depends on the position of all object markings in the current camera coordinate system. In the following we need assume the knowledge of the camera's focal length f and the ${}^{C}z$ component of the positions of image markings in $\{C\}$, which cannot be derived from their image position $({}^{S}x, {}^{S}y)$. Then with the help of f and the image coordinates $({}^{S}x, {}^{S}y)$ the complete position of the object markings in $\{C\}$ can be derived with the pinhole camera model (4).
We will first construct the model Φ_n for the case of a single object marking, $M = \frac{m}{2} = 1$. According to equations (6) and (7) we have for an object point ${}^{W}p$:

$$\begin{aligned}\Phi_n(u) &= \eta \circ \varphi(x_n, u) \\ &= {}^{S}_{C_{n+1}}\mathrm{T} \circ {}^{C_{n+1}}_{C_n}\mathrm{T} \circ {}^{C_n}_{T}\mathrm{T} \circ {}^{T}_{W}\mathrm{T}\ {}^{W}p \\ &= {}^{S}_{C_{n+1}}\mathrm{T} \circ {}^{C_{n+1}}_{C_n}\mathrm{T}\ {}^{C_n}x,\end{aligned} \tag{10}$$

where ${}^{C_n}x$ are the coordinates of the object point in $\{C_n\}$.
In the system state x_n the position of an object point ${}^{C_n}x =: p = (p_1, p_2, p_3)^T$ can be derived with $({}^{S}x, {}^{S}y)^T$, assuming the knowledge of f and ${}^{C}z$, via (4). Then the camera changes its pose by ${}^{C}u =: u = (u_1, u_2, u_3, u_4, u_5, u_6)^T$; we wish to know the new coordinates $({}^{S}\tilde{x}, {}^{S}\tilde{y})^T$ of p in the image. The new position $\tilde{p}$ of the point in new camera coordinates is given by a translation by u_1 through u_3 and a rotation of the camera by u_4 through u_6. We have

$$\begin{aligned}\tilde{p} &= \mathrm{rot}_x(-u_4)\ \mathrm{rot}_y(-u_5)\ \mathrm{rot}_z(-u_6) \begin{pmatrix} p_1 - u_1 \\ p_2 - u_2 \\ p_3 - u_3 \end{pmatrix} \\ &= \begin{pmatrix} c_5c_6 & c_5s_6 & -s_5 \\ s_4s_5c_6 - c_4s_6 & s_4s_5s_6 + c_4c_6 & s_4c_5 \\ c_4s_5c_6 + s_4s_6 & c_4s_5s_6 - s_4c_6 & c_4c_5 \end{pmatrix} \begin{pmatrix} p_1 - u_1 \\ p_2 - u_2 \\ p_3 - u_3 \end{pmatrix}\end{aligned} \tag{11}$$

using the short notation

$$s_i := \sin u_i,\ c_i := \cos u_i \qquad \text{for } i = 4, 5, 6. \tag{12}$$

Again with the help of the pinhole camera model (4) we can calculate the $\{S\}$ coordinates of the projection of the new point, which finally yields the model Φ_n:

$$\begin{aligned}\begin{bmatrix} {}^{S}\tilde{x} \\ {}^{S}\tilde{y} \end{bmatrix} &= \Phi(x_n, u) \\ &= \Phi_n(u) \\ &= f \cdot \begin{bmatrix} \dfrac{c_5\, c_6\, (p_1 - u_1) + c_5\, s_6\, (p_2 - u_2) - s_5\, (p_3 - u_3)}{(c_4\, s_5\, c_6 + s_4\, s_6)\, (p_1 - u_1) + (c_4\, s_5\, s_6 - s_4\, c_6)\, (p_2 - u_2) + c_4\, c_5\, (p_3 - u_3)} \\ \\ \dfrac{(s_4\, s_5\, c_6 - c_4\, s_6)\, (p_1 - u_1) + (s_4\, s_5\, s_6 + c_4\, c_6)\, (p_2 - u_2) + s_4\, c_5\, (p_3 - u_3)}{(c_4\, s_5\, c_6 + s_4\, s_6)\, (p_1 - u_1) + (c_4\, s_5\, s_6 - s_4\, c_6)\, (p_2 - u_2) + c_4\, c_5\, (p_3 - u_3)} \end{bmatrix}.\end{aligned} \tag{13}$$

2.5 Simplified and Inverse Models

As mentioned before, the controller needs to derive necessary movements from given desired image changes, for which an inverse model is beneficial. However, $\Phi_n(u)$ is too complicated to invert. Therefore in practice usually a linear approximation $\hat{\Phi}_n(u)$ of $\Phi_n(u)$ is calculated and then inverted. This can be done in a number of ways.

2.5.1 The Standard Image Jacobian

The simplest and most common linear model is the *Image Jacobian*. It is obtained by Taylor expansion of (13) around $u = 0$:

$$\begin{aligned} y_{n+1} &= \eta(\varphi(x_n, u)) \\ &= \Phi(x_n, u) \\ &= \Phi_n(u) \\ &= \Phi_n(0 + u) \\ &= \Phi_n(0) + J_{\Phi_n}(0)\, u + \mathcal{O}(\|u\|^2). \end{aligned} \tag{14}$$

With $\Phi_n(0) = y_n$ and the definition $J_n := J_{\Phi_n}(0)$ the image change can be approximated

$$y_{n+1} - y_n \approx J_n\, u \tag{15}$$

for sufficiently small $\|u\|_2$.

The Taylor expansion of the two components of (13) around $u = 0$ yields the Image Jacobian J_n for one object marking ($m = 2$):

$$J_n = \begin{pmatrix} -\frac{f}{{}^{C}z} & 0 & \frac{{}^{S}x}{{}^{C}z} & \frac{{}^{S}x\,{}^{S}y}{f} & -f - \frac{{}^{S}x^2}{f} & {}^{S}y \\ 0 & -\frac{f}{{}^{C}z} & \frac{{}^{S}y}{{}^{C}z} & f + \frac{{}^{S}y^2}{f} & -\frac{{}^{S}x\,{}^{S}y}{f} & -{}^{S}x \end{pmatrix} \tag{16}$$

where again image positions where converted back to sensor coordinates.

The Image Jacobian for M object markings, $M \in \mathbb{N}_{>1}$, can be derived analogously; the change of the $m = 2M$ image features can be approximated by

$$
\begin{aligned}
y_{n+1} - y_n &\approx J_n\, u \\
&= \begin{pmatrix}
-\frac{f}{{}^{C}z_1} & 0 & \frac{{}^{S}x_1}{{}^{C}z_1} & \frac{{}^{S}x_1\,{}^{S}y_1}{f} & -f - \frac{{}^{S}x_1^2}{f} & {}^{S}y_1 \\
0 & -\frac{f}{{}^{C}z_1} & \frac{{}^{S}y_1}{{}^{C}z_1} & f + \frac{{}^{S}y_1^2}{f} & -\frac{{}^{S}x_1\,{}^{S}y_1}{f} & -{}^{S}x_1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-\frac{f}{{}^{C}z_M} & 0 & \frac{{}^{S}x_M}{{}^{C}z_M} & \frac{{}^{S}x_M\,{}^{S}y_M}{f} & -f - \frac{{}^{S}x_M^2}{f} & {}^{S}y_M \\
0 & -\frac{f}{{}^{C}z_M} & \frac{{}^{S}y_M}{{}^{C}z_M} & f + \frac{{}^{S}y_M^2}{f} & -\frac{{}^{S}x_M\,{}^{S}y_M}{f} & -{}^{S}x_M
\end{pmatrix}
\begin{pmatrix} u_1 \\ \vdots \\ u_6 \end{pmatrix},
\end{aligned}
\tag{17}
$$

for small $\|u\|_2$, where $({}^{S}x_i, {}^{S}y_i)$ are the sensor coordinates of the ith projected object marking and ${}^{C}z_i$ their distances from the camera, $i = 1, \ldots, M$.

2.5.2 A Linear Model in the Cylindrical Coordinate System

Iwatsuki and Okiyama (2005) suggest a formulation of the problem in cylindrical coordinates. This means that positions of markings on the sensor are given in polar coordinates, $(\rho, \varphi)^T$ where ρ and φ are defined as in Figure 5 ($z = 0$). The Image Jacobian J_n for one image point is given in this case by

$$
J_n = \begin{pmatrix}
-\frac{f c_\varphi}{{}^{C}z} & -\frac{f s_\varphi}{{}^{C}z} & \frac{{}^{C}y s_\varphi + {}^{C}x c_\varphi}{{}^{C}z} & \left(f + \frac{{}^{C}y^2}{f}\right) s_\varphi + \frac{{}^{C}x\,{}^{C}y c_\varphi}{f} & \left(-f - \frac{{}^{C}x^2}{f}\right) c_\varphi - \frac{{}^{C}x\,{}^{C}y s_\varphi}{f} & {}^{C}y c_\varphi - {}^{C}x s_\varphi \\
\frac{f s_\varphi}{{}^{C}z} & -\frac{f c_\varphi}{{}^{C}z} & \frac{{}^{C}y c_\varphi + {}^{C}x s_\varphi}{{}^{C}z} & \left(f + \frac{{}^{C}y^2}{f}\right) c_\varphi - \frac{{}^{C}x\,{}^{C}y s_\varphi}{f} & \left(f + \frac{{}^{C}x^2}{f}\right) s_\varphi - \frac{{}^{C}x\,{}^{C}y c_\varphi}{f} & -{}^{C}y s_\varphi - {}^{C}x c_\varphi
\end{pmatrix}
\tag{18}
$$

with the short notation

$$
s_\varphi := \sin\varphi \quad \text{and} \quad c_\varphi := \cos\varphi. \tag{19}
$$

and analogously for $M > 1$ object markings.

2.5.3 Quadratic Models

A quadratic model, e.g. a quadratic approximation of the system model (13), can be obtained by a Taylor expansion; a resulting approximation for $M = 1$ marking is

$$
y_{n+1} = \begin{bmatrix} {}^{S}\tilde{x} \\ {}^{S}\tilde{y} \end{bmatrix} = \Phi_n(0) + J_{\Phi_n}(0)\, u + \frac{1}{2} \begin{bmatrix} u^T H_{S_x} u \\ u^T H_{S_y} u \end{bmatrix} + \mathcal{O}(\|u\|^3). \tag{20}
$$

where again $\Phi_n(0) = y_n$ and $J_{\Phi_n}(0) = J_n$ from (16), and the Hessian matrices are

$$H_{S_x} = \begin{pmatrix} 0 & 0 & \frac{-f}{{}^{C}z^2} & \frac{-{}^{S}y}{{}^{C}z} & \frac{2\,{}^{S}x}{{}^{C}z} & 0 \\ 0 & 0 & 0 & \frac{-{}^{S}x}{{}^{C}z} & 0 & \frac{-f}{{}^{C}z} \\ \frac{-f}{{}^{C}z^2} & 0 & \frac{2\,{}^{S}x}{{}^{C}z^2} & \frac{2\,{}^{S}x\,{}^{S}y}{f\,{}^{C}z} & \frac{-2\,{}^{S}x^2}{f\,{}^{C}z} & \frac{{}^{S}y}{{}^{C}z} \\ \frac{-{}^{S}y}{{}^{C}z} & \frac{-{}^{S}x}{{}^{C}z} & \frac{2\,{}^{S}x\,{}^{S}y}{f\,{}^{C}z} & {}^{S}x\left(1+2\left(\frac{{}^{S}y}{f}\right)^2\right) & -{}^{S}y\left(1+2\left(\frac{{}^{S}x}{f}\right)^2\right) & \frac{{}^{S}y^2-{}^{S}x^2}{f} \\ \frac{2\,{}^{S}x}{{}^{C}z} & 0 & \frac{-2\,{}^{S}x^2}{f\,{}^{C}z} & -{}^{S}y\left(1+2\left(\frac{{}^{S}x}{f}\right)^2\right) & 2\,{}^{S}x\left(1+\frac{{}^{S}x^2}{f}\right)^2 & -2\,{}^{S}x\frac{{}^{S}y}{f} \\ 0 & \frac{-f}{{}^{C}z} & \frac{{}^{S}y}{{}^{C}z} & \frac{{}^{S}y^2-{}^{S}x^2}{f} & \frac{-2\,{}^{S}x\,{}^{S}y}{f} & -{}^{S}x \end{pmatrix} \tag{21}$$

as well as

$$H_{S_y} = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{{}^{S}y}{{}^{C}z} & \frac{f}{{}^{C}z} \\ 0 & 0 & \frac{-f}{{}^{C}z^2} & \frac{-2\,{}^{S}y}{{}^{C}z} & \frac{{}^{S}x}{{}^{C}z} & 0 \\ 0 & \frac{-f}{{}^{C}z^2} & \frac{2\,{}^{S}y}{{}^{C}z^2} & \frac{2\,{}^{S}y^2}{f\,{}^{C}z} & \frac{-2\,{}^{S}x\,{}^{S}y}{f\,{}^{C}z} & \frac{-{}^{S}x}{{}^{C}z} \\ 0 & \frac{-2\,{}^{S}y}{{}^{C}z} & \frac{2\,{}^{S}y^2}{f\,{}^{C}z} & 2\,{}^{S}y\left(1+\frac{{}^{S}y^2}{f}\right)^2 & \left(\frac{{}^{S}y}{f}\right)\left(-2\,{}^{S}x\frac{{}^{S}y}{f}\right) & \frac{-2\,{}^{S}x\,{}^{S}y}{f} \\ \frac{{}^{S}y}{{}^{C}z} & \frac{{}^{S}x}{{}^{C}z} & \frac{-2\,{}^{S}x\,{}^{S}y}{f\,{}^{C}z} & \left(\frac{{}^{S}y}{f}\right)\left(-2\,{}^{S}x\frac{{}^{S}y}{f}\right) & {}^{S}y\left(1+2\left(\frac{{}^{S}x}{f}\right)^2\right) & \frac{{}^{S}x^2-{}^{S}y^2}{f} \\ \frac{f}{{}^{C}z} & 0 & \frac{-{}^{S}x}{{}^{C}z} & \frac{-2\,{}^{S}x\,{}^{S}y}{f} & \frac{{}^{S}x^2-{}^{S}y^2}{f} & -{}^{S}y \end{pmatrix}. \tag{22}$$

2.5.4 A Mixed Model

Malis (2004) proposes a way of constructing a mixed model which consists of different linear approximations of the target function Φ. Let x_n again be the current robot pose and $x^\star$ the teach pose. For a given robot command u we set again $\Phi_n(u) := \Phi(x_n, u)$ and now also $\Phi^\star(u) := \Phi(x^\star, u)$ such that $\Phi_n(0) = y_n$ und $\Phi^\star(0) = y^\star$. Then Taylor expansions of Φ_n and $\Phi^\star$ at $u = 0$ yield

$$y_{n+1} = y_n + J_{\Phi_n}(0)u + \mathcal{O}(\|u\|^2) \tag{23}$$

and

$$y_{n+1} = y_n + J_{\Phi^\star}(0)u + \mathcal{O}(\|u\|^2). \tag{24}$$

In other words, both Image Jacobians, $J_n := J_{\Phi_n}(0)$ and $J^\star := J_{\Phi^\star}(0)$ can be used as linear approximations of the behaviour of the robot system. One of these models has its best validity at the current pose, the other at the teach pose. Since we are moving the robot from one towards the other it may be useful to consider both models. Malis proposes to use a mixture of these two models, i.e.

$$y_{n+1} - y_n \approx \frac{1}{2}(J_n + J^\star)\, u. \tag{25}$$

In his control law (see Section 3 below) he calculates the pseudoinverse of the Jacobians, and therefore calls this approach "Pseudo-inverse of the Mean of the Jacobians", or short "PMJ". In a variation of this approach the computation of mean and pseudo-inverse is exchanged, which results in the "MPJ" method. See Section 3 for details.

2.5.5 Estimating Models

Considering the fact that models can only ever approximate the real system behaviour it may be beneficial to use measurements obtained during the visual servoing process to update the model "online". While even the standard models proposed above use current measurements to estimate the distance ${}^{C}z$ from the object to use this estimate in the Image Jacobian, there are also approaches that estimate more variables, or construct a complete model from scratch. This is most useful when no certain data about the system state or setup are available. The following aspects need to be considered when estimating the Image Jacobian—or other models:

- How precise are the measurements used for model estimation, and how large is the sensitivity of the model to measurement errors?
- How many measurements are needed to construct the model? For example, some methods use 6 robot movements to measure the 6-dimensional data within the Image Jacobian. In a static look-and-move visual servoing setup which may reach its goal in 10-20 movements with a given Jacobian the resulting increase in necessary movements, as well as possible mis-directed movements until the estimation process converges, need to be weighed against the flexibility achieved by the automatic model tuning.

The most prominent approach to estimation methods of the whole Jacobian is the *Broyden approach* which has been used by Jägersand (1996). The Jacobian estimation uses the following update formula for the current estimate $\hat{J}_n$:

$$\hat{J}_n := {}^{C_n}_{C_{n-1}}\mathrm{T} \left(\hat{J}_{n-1} + \frac{(y_n - y_{n-1} - \hat{J}_{n-1}\, u_n)\, u_n^T}{u_n^T u_n} \right), \tag{26}$$

with an additional weighting of the correction term

$$J_n := \gamma\, \hat{J}_{n-1} + (1-\gamma)\, \hat{J}_n, \quad 0 \leq \gamma < 1 \tag{27}$$

to reduce the sensitivity of the estimate to measurement noise.

In the case of Jägersand's system using an estimation like this makes sense since he worked with a dynamic visual servoing setup where many more measurements are made over time compared to our setup ("static look-and-move", see below).

In combination with a model-based measurement a non-linear model could also make sense. A number of methods for the estimation of quadratic models are available in the optimisation literature. More on this subject can be found e.g. in Fletcher (1987, chapter 3) and Sage and White (1977, chapter 9).

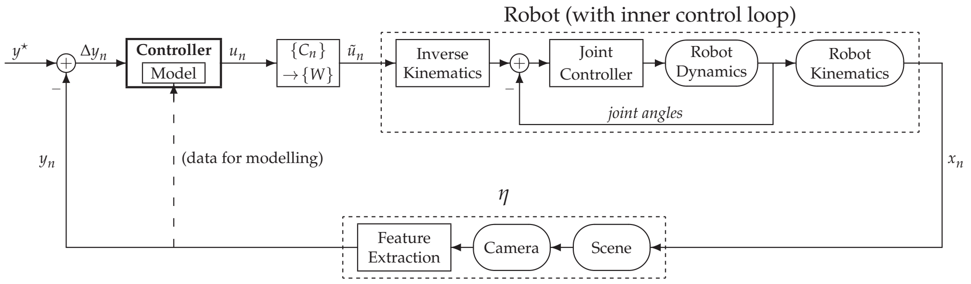


Fig. 7. Typical closed-loop image-based visual servoing controller

3. Designing a Visual Servoing Controller

Using one of the models defined above we wish to design a controller which steers the robot arm towards an object of unknown pose. This is to be realised in the visual feedback loop depicted in Figure 7. Using the terminology defined by Weiss et al. (1987) the visual servoing controller is of the type "*Static Image-based Look-and-Move*". "Image-based" means that goal and error are defined in image coordinates instead of using positions in normal space (that would be "position-based"). "Static Look-and-Move" means that the controller is a sampled data feedback controller and the robot does not move while a measurement is taken. This traditionally implies that the robot is controlled by giving world coordinates to the controller instead of directly manipulating robot joint angles (Chaumette and Hutchinson, 2008; Hutchinson et al., 1996).

The object has 4 circular, identifiable markings. Its appearance in the image is described by the *image feature vector* $y_n \in \mathbb{R}^8$ that contains the 4 pairs of image coordinates of these markings in a fixed order. The desired pose relative to the object is defined by the object's appearance in that pose by measuring the corresponding *desired image features* $y^\star \in \mathbb{R}^8$ ("*teaching by showing*"). Object and robot are then moved so that no Euclidean position of the object or robot is known to the controller. The *input* to the controller is the *image error* $\Delta y_n := y^\star - y_n$. The current image measurements y_n are also given to the controller for adapting its internal model to the current situation. The *output* of the controller is a relative movement of the robot in the camera coordinate system, a 6-dimensional vector $(x, y, z, yaw, pitch, roll)$ for a 6 DOF movement.

Controllers can be classified into approaches where the control law (or its parameters) are adapted over time, and approaches where they are fixed. Since these types of controllers can exhibit very different controlling behaviour we will split our considerations of controllers into these two parts, after some general considerations.

3.1 General Approach

Generally, in order to calculate the necessary camera movement u_n for a given desired image change $\Delta\tilde{y}_n := \tilde{y}_{n+1} - y_n$ we again use an approximation $\hat{\Phi}_n$ of Φ_n, for example the image Jacobian J_n. Then we select

$$u_n \in \operatorname*{argmin}_{u \in \mathcal{U}(x_n)} \left\| \Delta\tilde{y}_n - \hat{\Phi}_n(u) \right\|_2^2 . \tag{28}$$

where a given algorithm may or may not enforce a restriction $u \in \mathcal{U}(x_n)$ on the admissible movements when determining u. If this restriction is inactive and we are using a Jacobian, $\hat{\Phi}_n = J_n$, then the solution to (28) with minimum norm $\|u_n\|_2$ is given by

$$u_n = J_n^+ \, \Delta\tilde{y}_n \tag{29}$$

where J_n^+ is the *pseudo-inverse* of J_n.
With 4 coplanar object markings $m = 8$ and thereby $J_n \in \mathbb{R}^{8\times 6}$. One can show that J_n has maximum rank[1], so rk $J_n = 6$. Then the pseudo-inverse $J_n^+ \in \mathbb{R}^{6\times 8}$ of J_n is given by:

$$J_n^+ = (J_n^T J_n)^{-1} J_n^T \tag{30}$$

(see e.g. Deuflhard and Hohmann, 2003, chapter 3).

When realising a control loop given such a controller one usually sets a fixed error threshold $\varepsilon > 0$ and repeats the steps

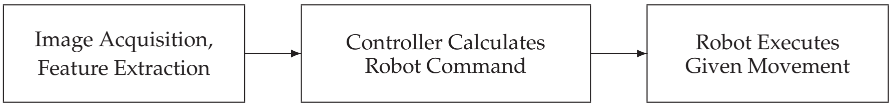


Image Acquisition, Feature Extraction
Controller Calculates Robot Command
Robot Executes Given Movement

until

$$\|\Delta y_n\|_2 = \|y^\star - y_n\|_2 < \varepsilon, \tag{31}$$

or until

$$\|\Delta y_n\|_\infty = \|y^\star - y_n\|_\infty < \varepsilon \tag{32}$$

if one wants to stop only when the maximum deviation in *any* component of the image feature vector is below ε. Setting $\varepsilon := 0$ is not useful in practice since measurements even in the same pose tend to vary a little due to small movements of the robot arm or object as well as measurement errors and fluctuations.

3.2 Non-Adaptive Controllers

3.2.1 The Traditional Controller

The most simple controller, which we will call the "*Traditional Controller*" due to its heritage, is a straightforward proportional controller as known in engineering, or a dampened Gauss-Newton algorithm as it is known in mathematics.
Given an Image Jacobian J_n we first calculates the full Gauss-Newton step Δu_n for a complete movement to the goal in one step (desired image change $\Delta\tilde{y}_n := \Delta y_n$):

$$\Delta u_n := J_n^+ \, \Delta y_n \tag{33}$$

without enforcing a restriction $u \in \mathcal{U}(x_n)$ for the admissibility of a control command.
In order to ensure a convergence of the controller the resulting vector is then scaled with a dampening factor $0 < \lambda_n \leq 1$ to get the controller output u_n. In the traditional controller the factor λ_n is constant over time and the most important parameter of this algorithm. A typical value is $\lambda_n = \lambda = 0.1$; higher values may hinder convergence, while lower values also significantly slow down convergence. The resulting controller output u_n is given by

[1] One uses the fact that no 3 object markings are on a straight line, ${}^{C}z_i > 0$ for $i = 1, \ldots, 4$ and all markings are visible (in particular, neither all four ${}^{C}x_i$ nor all four ${}^{C}y_i$ are 0).

$$u_n := \lambda \cdot J_n^+ \, \Delta y_n. \tag{34}$$

3.2.2 Dynamical and Constant Image Jacobians

As mentioned in the previous section there are different ways of defining the Image Jacobian. It can be defined in the current pose, and is then calculated using the current distances to the object, ${}^{C}z_i$ for marking i, and the current image features. This is the *Dynamical Image Jacobian* J_n. An alternative is to define the Jacobian in the teach (goal) pose $x^\star$, with the image data $y^\star$ and distances at that pose. We call this the *Constant Image Jacobian* $J^\star$. Unlike J_n, $J^\star$ is constant over time and does not require image measurements for its adaptation to the current pose.
From a mathematical point of view the model J_n has a better validity in the current system state and should therefore yield better results. We shall later see whether this is the case in practice.

3.2.3 The Retreat-Advance Problem

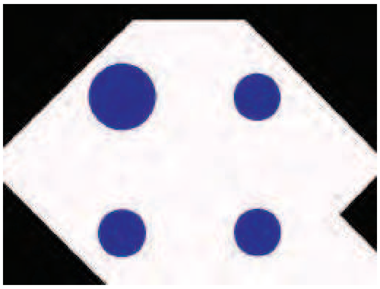


Fig. 8. Camera view in the start pose with a pure rotation around the ${}^{C}z$ axis

When the robot's necessary movement to the goal pose is a pure rotation around the optical axis (${}^{C}z$, approach direction) there can be difficulties when using the standard Image Jacobian approach (Chaumette, 1998). The reason is that the linear approximation J_n models the relevant properties of Φ_n badly in these cases. This is also the case with $J^\star$ if this Jacobian is used. The former will cause an unnecessary movement away from the object, the latter a movement towards the goal. The larger the roll angle, the more pronounced is this phenomenon, an extreme case being a roll error of $\pm\pi$ (all other pose elements already equal to the teach pose) where the Jacobians suggest a pure movement along the ${}^{C}z$ axis. Corke and Hutchinson (2001) call this the "*Retreat-Advance Problem*" or the "*Chaumette Conundrum*".

3.2.4 Controllers using the PMJ and MPJ Models

In order to overcome the Retreat-Advance Problem the so-called "*PMJ Controller*" (Malis, 2004) uses the pseudo-inverse of the mean of the two Jacobians J_n and $J^\star$. Using again a dampening factor $0 < \lambda \leq 1$ the controller output is given by

$$u_n = \lambda \cdot \left(\frac{1}{2} \left(J_n + J^\star \right) \right)^+ \Delta y_n. \tag{35}$$

Analogously, the "*MPJ Controller*" works with the mean of the pseudo-inverse of the Jacobians:

$$u_n = \lambda \cdot \left(\frac{1}{2} \left(J_n^+ + J^{\star +} \right) \right) \Delta y_n. \tag{36}$$

Otherwise, these controllers work like the traditional approach, with a constant dampening λ.

3.2.5 Defining the Controller in the Cylindrical Coordinate System

Using the linear model by Iwatsuki and Okiyama (2005) in the cylindrical coordinate system as discussed in Section 2.5.2 a special controller can also be defined. The authors define the image error for the ith object marking as follows:

$$e_i := \begin{pmatrix} \rho^\star - \rho \\ \rho(\varphi^\star - \varphi) \end{pmatrix} \tag{37}$$

where $(\rho, \varphi)^T$ is the current position and $(\rho^\star, \varphi^\star)$ the teach position. The control command u is then given by

$$u = \lambda \tilde{J}^+ e, \tag{38}$$

$\tilde{J}^+$ being the pseudo-inverse of the Image Jacobian in cylindrical coordinates from equation (18). e is the vector of pairs of image errors in the markings, i.e. a concatenation of the e_i vectors.
It should be noted that even if e is given in cylindrical coordinates, the output u of the controller is in Cartesian coordinates.
Due to the special properties of cylindrical coordinates, the calculation of the error and control command is very much dependent on the definition of the origin of the coordinate system. Iwatsuki and Okiyama (2005) therefore present a way to shift the origin of the coordinate system such that numerical difficulties are avoided.

One approach to select the origin of the cylindrical coordinate system is such that the current pose can be transformed to the desired (teach) pose with a pure rotation around the axis normal to the sensor plane, through the origin. For example, the general method given by Kanatani (1996) can be applied to this problem.
Let $l = (l_x, l_y, l_z)^T$ be the unit vector which defines this rotation axis, and $o = (o_x, o_y)^T$ the new origin, obtained by shifting the original origin $(0,0)^T$ in $\{S\}$ by $(\eta, \xi)^T$.
If $|l_z|$ is very small then the rotation axis l is almost parallel to the sensor. Then η and ξ are very large, which can create numerical difficulties. Since the resulting cylindrical coordinate system approximates a Cartesian coordinate system as $\eta, \xi \to \infty$, the standard Cartesian Image Jacobian J_n from (17) can therefore used if $|l_z| < \delta$ for a given lower limit δ.

3.3 Adaptive Controllers

Using adaptive controllers is a way to deal with errors in the model, or with problems resulting from the simplification of the model (e.g. linearisation, or the assumption that the camera works like a pinhole camera). The goal is to ensure a fast convergence of the controller in spite of these errors.

3.3.1 Trust Region-based Controllers

Trust Region methods are known from mathematics as globally convergent optimisation methods (Fletcher, 1987). In order to optimise "difficult" functions one uses a model of its properties, like we do here with the Image Jacobian. This model is adapted to the current state/position in the solution space, and therefore only valid within some region around the current state. The main idea in trust region methods is to keep track of the validity of the current system model, and adapt a so-called "*Trust Region*", or "*Model Trust Region*" around the current state within which the model does not exhibit more than a certain pre-defined "acceptable error".

To our knowledge the first person to use trust region methods for a visual servoing controller was Jägersand (1996). Since the method was adapted to a particular setup and cannot be used here we have developed a different trust region-based controller for our visual servoing scenario (Siebel et al., 1999). The main idea is to replace the constant dampening λ for Δu_n with a variable dampening λ_n:

$$u_n := \lambda_n \cdot \Delta u_n = \lambda_n \cdot J_n^+ \Delta y_n. \tag{39}$$

The goal is to adapt λ_n before each step to balance the avoidance of model errors (by making small steps) and the fast movement to the goal (by making large steps).

In order to achieve this balance we define an *actual model error* e_n which is set in relation to a *desired (maximum) model error* e_{des}[2] to adapt a bound α_n for the movement of projected object points on the sensor. Using this purely image-based formulation has advantages, e.g. having a measure to avoid movements that lead to losing object markings from the camera's field of view.

Our algorithm is explained in Figure 9 for one object marking. We wish to calculate a robot command to move such that the current point position on the sensor moves to its desired position. In step ①, we calculate an undampened robot movement Δu_n to move as close to this goal as possible ($\Delta\tilde{y}_n := \Delta y_n$) according to an Image Jacobian J_n:

$$\Delta u_n := J_n^+ \Delta y_n. \tag{40}$$

This gives us a predicted movement ℓ_n on the sensor, which we define as the maximum movement on the sensor for all M markings:

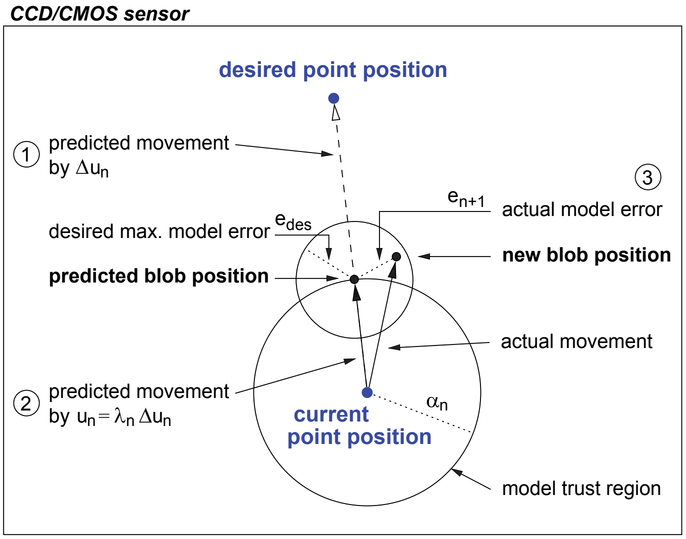
$$\ell_n := \max_{i=1,\dots,M} \left\| \begin{bmatrix} (J_n \Delta u_n)_{2i-1} \\ (J_n \Delta u_n)_{2i} \end{bmatrix} \right\|_2, \tag{41}$$

where the subscripts to the vector $J_n \Delta u_n$ signify a selection of its components.

Before executing the movement we restrict it in step ② such that the distance on the sensor is less or equal to a current limit α_n:

$$\begin{aligned} u_n &:= \lambda_n \cdot \Delta u_n \\ &= \min\left\{1, \frac{\alpha_n}{\ell_n}\right\} \cdot J_n^+ \Delta y_n. \end{aligned} \tag{42}$$

[2] While the name "desired error" may seem unintuitive the name is chosen intentionally since the α adaptation process (see below) can be regarded as a control process to have the robot system reach exactly this amount of error, by controlling the value of α_n.


CCD/CMOS sensor
desired point position
① predicted movement by Δu_n
③
e_{n+1} actual model error
desired max. model error e_{des}
new blob position
predicted blob position
actual movement
② predicted movement by $u_n = \lambda_n \Delta u_n$
α_n
current point position
model trust region
Fig. 9. Generation of a robot command by the trust region controller: view of the image sensor with a projected object marking

After this restricted movement is executed by the robot we obtain new measurements y_{n+1} and thereby the actual movement and model (prediction) error e_{n+1} ③, which we again define as the maximum deviation on the sensor for $M > 1$ markings:

$$e_{n+1} := \max_{i=1,\ldots,M} \left\| \begin{bmatrix} (\hat{y}_{n+1})_{2i-1} \\ (\hat{y}_{n+1})_{2i} \end{bmatrix} - \begin{bmatrix} (y_{n+1})_{2i-1} \\ (y_{n+1})_{2i} \end{bmatrix} \right\|_2 . \tag{43}$$

where $\hat{y}_{n+1}$ is the vector of predicted positions on the sensor,

$$\hat{y}_{n+1} := y_n + J_n\, u_n . \tag{44}$$

The next step is the adaptation of our restriction parameter α_n. This is done by comparing the model error e_{n+1} with a given desired (maximum admissible) error e_{des}:

$$r_{n+1} := \frac{e_{n+1}}{e_{\text{des}}} \tag{45}$$

where r_n is called the *relative model error*. A small value signifies a good agreement of model and reality. In order to balance model agreement and a speedy control we adjust α_n so as to achieve $r_n = 1$. Since we have a linear system model we can set

$$\alpha_{n+1} := \alpha_n \cdot \frac{e_{\text{des}}}{e_{n+1}} = \frac{\alpha_n}{r_{n+1}} \tag{46}$$

with an additional restriction on the change rate, $\frac{\alpha_{n+1}}{\alpha_n} \leq 2$. In practice, it may make sense to define minimum and maximum values α_{min} and α_{max} and set $\alpha_0 := \alpha_{\text{min}}$.
In the example shown in Figure 9 the actual model error is smaller than e_{des}, so α_{n+1} can be larger than α_n.