3

Vision Guided Robot Gripping Systems

Zdzislaw Kowalczuk and Daniel Wesierski
Gdansk University of Technology
Poland

1. Description of the past and recent trends in robot positioning systems

Industrial robots are used customary without any embedded sensors. They rely on a predictable pose of an object (position and orientation in 6 degrees of freedom, 6DOF) when performing the task of gripping parts located for instance on palettes or assembly lines. In practice though, a part can easily deviate from its ideal nominal location and a robot having no embedded sensors can miss or crash into the object. This would lead to damages and downtime of such an assembly line.

1.1 Manual and automated part acquisition

Manual part acquisition involves human employment. Clearly, it is not a good solution because humans are exposed to possible injuries, what increasing medical and social costs. Parts are often sharp and heavy. Yet, they are not sterile. Contamination (for instance, dust, oil, hair etc.) transferred to critical areas of the object leads to reduction in the quality of assembly (inevitably followed by product recalls).
Conventionally, automated gripping relied on intricate mechanical and electromechanical devices known as precision fixtures, which were utilized to ensure that the part was always at the programmed pose with respect to the robot. The design of such fixtures is though expensive, imposes design constraints, requires frequent maintenance, and has a reduced flexibility.

1.2 2D and 3D robot positioning

Over the years a variety of techniques have been developed to automate the process of gripping parts as an alternative to the existing manual part acquisition. Due to the rapidly evolving machine vision technology, vision sensors are playing today a key role in the three-dimensional robot positioning systems. They are not only cheaper but also far more effective.
A robot with an embedded vision sensor can have greater 'awareness' of the scene. It can grip objects, which can be non-fixtured, stacked or loosely located. Thus, it enables the robot to grip objects that are provided in racks, bins, or on pallets. Regardless of the presentation, a vision-guided robot can locate an object for further processing. This generic application of robotic guidance is applied in industries such as automotive for the location of power train components, sheet metal body parts, complete car bodies, and other parts used in assembly. Other industries such as food, pharmaceutical, glass and daily products apply vision guided robotic technology to their applications, as well.

As a response to the industry needs two major techniques have emerged: 2D and 3D machine vision. Two-dimensional machine vision is a well-developed technique and has been successfully implemented in the past years. 2D robotic vision systems locate the object in 3 degrees of freedom (x, y, and roll angle) based on one image. Consequently, the main limitation of 2D vision is its inability to compute part's rotation outside of a single plane. Unfortunately, this does not suffice in many applications that aim to eliminate, for instance, the precision fixtures in order to achieve greater versatility. 2D vision systems have proved to be very useful in picking objects from moving conveyors. Calibration of such robotic systems requires relatively simple methods.

The problem of creating a vision-guided robot positioning system for 3D part acquisition has apparently been studied before. 3D machine vision systems locate the object in 6 degrees of freedom (x, y, z and yaw, pitch, roll). We can distinguish here single-image systems which compute the object's pose iteratively using only one image, stereo systems which compute the pose analytically based on two overlapping images, and multi-vision systems, which combine the stereo-systems in a conventional manner to increase robustness and precision.

The 3D vision applications, which can position the robot to grip a rigid object using information derived only from one image, are gaining an increasing attention. The distances between the object features have to be known to the system beforehand for the purpose of computing the object's pose iteratively based on some minimized criteria. This information can be taken from a CAD model of the object in a model-based approach. Since only one camera is required, the cost of the whole plant is reduced, the cycle time is decreased, and the calibration process is made easier. Yet, finding features in one image (and not in multiple images) is simpler for image processing applications (IPAs). However, one-image methods have several drawbacks. One of them is that there are some critical configurations of points in 3D space, which could limit the number of potential features of the object for IPA. Another disadvantage is that these methods give good results if more than 5 points are found on the object what increases the processing time of IPA, and, more importantly, it increases the risk that not all points are found by IPA what can bring about stopping the plant and the entire assembly line.

Stereovision is thus far more often used in 3D positioning systems as it is simple to be implemented due to its analytical form. It computes the distance between the object features and the vision sensors, and derives all 3 coordinates of a feature. Having computed at least 3 features, the pose of the object can be determined. Commonly, more points are used to provide a certain degree of redundancy. This method has several disadvantages though: it is relatively sensitive to noise, identification of the corresponding features in two images can be very difficult (although the epipolar geometry of stereo cameras is very helpful here), and its application is confined to small objects due to a relatively small field of view. Multi-stereo-systems are used to compute the pose of bigger objects as they can examine them from opposite sides.

1.3 Retrieving information based on laser vision

Laser vision plays a vital role in 3D part acquisition tasks, as well. By painting a part's surface with a laser beam (coherent light), a laser triangulation sensor can determine the depth and the orientation of the surface observed. Although such measurements are very precise, the use of lasers has several drawbacks, such as long process of relating the features to the 'point cloud' data, shadowing/occlusion, as well as ergonomic issues when deployed near human operators. Moreover, lasers require using sophisticated interlock mechanisms, protective curtains, and goggles, which is very expensive.

1.4 Flexible assembly systems

Apart from integrating robots with machine vision, the assembly technology takes yet another interesting course. It aims to develop intelligent systems supporting human workers instead of replacing them. Such an effect can be gained by combining human skills (in particular, flexibility and intelligence) with the advantages of machine systems. It allows for creating a next generation of flexible assembly and technology processes. Their objectives cover the development of concepts, control algorithms and prototypes of intelligent assist robotic systems that allow workplace sharing (assistant robots), time-sharing with human workers, and pure collaboration with human workers in assembly processes. In order to fulfill these objectives new intelligent prototype robots are to be developed that integrate power assistance, motion guidance, advanced interaction control through sophisticated human-machine interfaces as well as multi-arm robotic systems, which integrate human skillfulness and manipulation capabilities.

Taking into account the above remarks, an analytical robot positioning system (Kowalczuk & Wesierski, 2007) guided by stereovision has been developed achieving the repeatability of ±1 mm and ±1 deg as a response to rising demands for safe, cost-effective, versatile, precise, and automated gripping of rigid objects, deviated in three-dimensional space (in 6DOF). After calibration, the system can be assessed for gripping parts during depalletizing, racking and un-racking, picking from assembly lines or even from bins, in which the parts are placed randomly. Such an effect is not possible to be obtained by robots without vision guidance. The Matlab Calibration Toolbox (MCT) software can be used for calibrating the system. Mathematical formulas for robot positioning and calibration developed here can be implemented in industrial tracking algorithms.

2. 3D object pose estimation based on single and stereo images

The entire vision-guided robot positioning system for object picking shall consist of three essential software modules: image processing application to retrieve object's features, mathematics involving calibration and transformations between CSs to grip the object, and communication interface to control the automatic process of gripping.

2.1 Camera model

In this chapter we explain how to map a point from a 3D scene onto the 2D image plane of the camera. In particular, we distinguish several parameters of the camera to determine the point mapping mathematically. These parameters comprise a model of the camera applied.

In particular, such a model represents a mathematical description of how the light reflected or emitted at points in a 3D scene is projected onto the image plane of the camera. In this Section we will be concerned with a projective camera model often referred to as a pinhole camera model. It is a model of a pinhole camera having its aperture infinitely small (reduced to a single point). With such a model, a point in space, represented by a vector characterized by three coordinates $\vec{r}^C = \begin{bmatrix} x^C & y^C & z^C \end{bmatrix}^{\mathrm{T}}$, is mapped to a point $\vec{r}^S = \begin{bmatrix} x^S & y^S \end{bmatrix}^{\mathrm{T}}$ in the sensor plane, where the line joining the point $\vec{r}^C$ with a center of projection O_C meets the sensor plane, as shown in Fig.1. The center of projection O_C, also called the camera center, is the origin of a coordinate system (CS) $\{\hat{X}_c, \hat{Y}_c, \hat{Z}_c\}$ in which the point $\vec{r}^C$ is defined (later on, this system we will be referred to as the Camera CS). By using the *triangle similarity rule* (confer Fig.1) one can easily see that the point $\vec{r}^C$ is mapped to the following point:

$$\vec{r}^C = \begin{bmatrix} -f_C \frac{x_C}{z^C} & -f_C \frac{y_C}{z^C} \end{bmatrix}^T$$

that means that

$$\vec{r}^C = \begin{bmatrix} -f_C \frac{x^C}{z^C} & -f_C \frac{y^C}{z^C} \end{bmatrix}^T \tag{1}$$

which describes the central projection mapping from Euclidean space $\mathbf{R}^3$ to $\mathbf{R}^2$. As the coordinate z^C cannot be reconstructed, the depth information is lost.

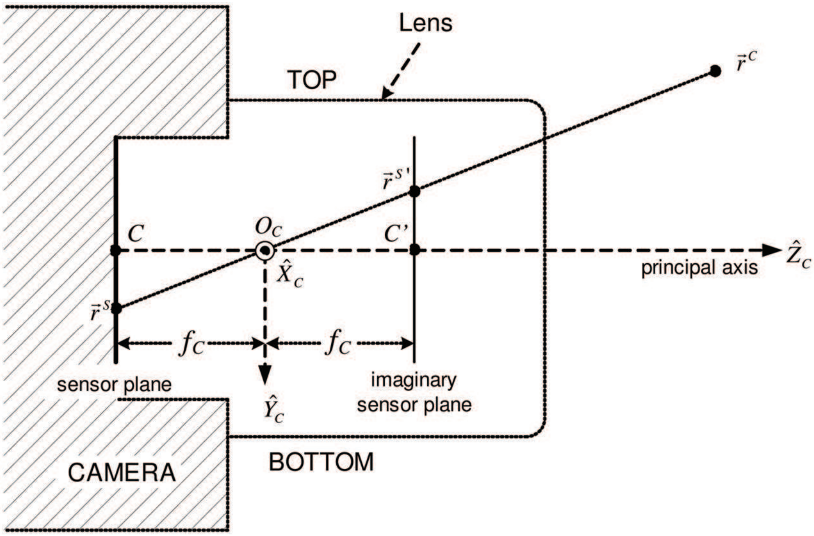


Fig. 1. Right side view of the camera-lens system

The line passing through the camera center O_C and perpendicular to the sensor plane is called the *principal axis* of the camera. The point where the principal axis meets the sensor plane is called a *principal point,* which is denoted in Fig. 1 as *C*.

The projected point $\vec{r}^S$ has negative coordinates with respect to the positive coordinates of the point $\vec{r}^C$ due to the fact that the projection inverts the image. Let us consider, for instance, the coordinate y^C of the point $\vec{r}^C$. It has a negative value in space because the axis $\hat{Y}_C$ points downwards. However, after projecting it onto the sensor plane it gains a positive value. The same concerns the coordinate x^C. In order to omit introducing negative coordinates to point $\vec{r}^S$, we can rotate the image plane by 180 deg around the axes $\hat{X}_C$ and

$\hat{Y}_C$ obtaining a non-existent plane, called an *imaginary sensor plane*. As can be seen in Fig. 1, the coordinates of the point $\vec{r}^{\,S'}$ directly correspond to the coordinates of point $\vec{r}^{\,C}$, and the projection law holds as well. In this Chapter we shall thus refer to the imaginary sensor plane.
Consequently, the central projection can be written in terms of matrix multiplication:

$$\begin{bmatrix} x^C \\ y^C \\ z^C \end{bmatrix} \rightarrow \begin{bmatrix} f_C \dfrac{x^C}{z^C} \\ f_C \dfrac{y^C}{z^C} \\ 1 \end{bmatrix} = \begin{bmatrix} f_C & 0 & 0 \\ 0 & f_C & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dfrac{x^C}{z^C} \\ \dfrac{y^C}{z^C} \\ 1 \end{bmatrix} \tag{2}$$

where $M = \begin{bmatrix} fc & 0 & 0 \\ 0 & fc & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is called a *camera matrix*.

The pinhole camera describes the ideal projection. As we use CCD cameras with lens, the above model is not sufficient enough for precise measurements because factors like rectangular pixels and lens distortions can easily occur. In order to describe the point mapping more accurately, *i.e.* from the 3D scene measured in millimeters onto the image plane measured in pixels, we extend our pinhole model by introducing additional parameters into both the camera matrix *M* and the projection equation (2). These parameters will be referred to as *intern camera parameters*.
Intern camera parameters The list of intern camera parameters contains the following components:

- distortion
- focal length (also known as a camera constant)
- principal point offset
- skew coefficient.

Distortion In optics the phenomenon of distortion refers to lens and is called *lens distortion*. It is an abnormal rendering of lines of an image, which most commonly appear to be bending inward (*pincushion distortion*) or outward (*barrel distortion*), as shown in Fig. 2.

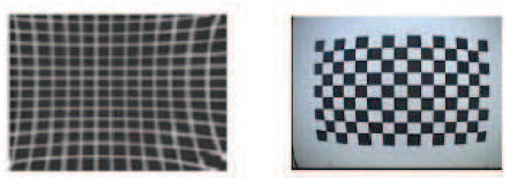


Fig. 2. Distortion: lines forming pincushion (left image) and lines forming a barrel (right image)

Since distortion is a principal phenomenon that affects the light rays producing an image, initially we have to apply the distortion parameters to the following normalized camera coordinates

$$\vec{r}^{\,C}_{Normalized} = \begin{bmatrix} \dfrac{x^C}{z^C} & \dfrac{y^C}{z^C} \end{bmatrix}^T = \begin{bmatrix} x_{norm} & y_{norm} \end{bmatrix}^T$$

Using the above and letting $h = x^2_{norm} + y^2_{norm}$, we can include the effect of distortion as follows:

$$\begin{aligned} x_d &= \left(1 + k_1 h^2 + k_2 h^4 + k_5 h^6\right) x_{norm} + dx_1 \\ y_d &= \left(1 + k_1 h^2 + k_2 h^4 + k_5 h^6\right) y_{norm} + dy_1 \end{aligned} \tag{3}$$

where x_d and y_d stand for normalized distorted coordinates and dx_1 and dx_2 are tangential distortion parameters defined as:

$$\begin{aligned} dx_1 &= 2k_3 x_{norm} y_{norm} + k_4\left(h^2 + 2x^2_{norm}\right) \\ dx_2 &= 2k_3\left(h^2 + 2y^2_{norm}\right) + 2k_4 x_{norm} y_{norm} \end{aligned} \tag{4}$$

The distortion parameters k_1 through k_5 describe both radial and tangential distortion. Such a model introduced by Brown in 1966 and called a "Plumb Bob" model is used in the MCT tool.

Focal length Each camera has an intern parameter called focal length f_c, also called a camera constant. It is the distance from the center of projection O_C to the sensor plane and is directly related to the focal length of the lens, as shown in Fig. 3. Lens focal length f is the distance in air from the center of projection O_C to the focus, also known as focal point.

In Fig. 3 the light rays coming from one point of the object converge onto the sensor plane creating a sharp image. Obviously, the distance d from the camera to an object can vary. Hence, the camera constant f_c has to be adjusted to different positions of the object by moving the lens to the right or left along the principal axis (here $\hat{Z}_c$ -axis), which changes the distance $|OC|$. Certainly, the lens focal length always remains the same, that is $|OF| = const.$

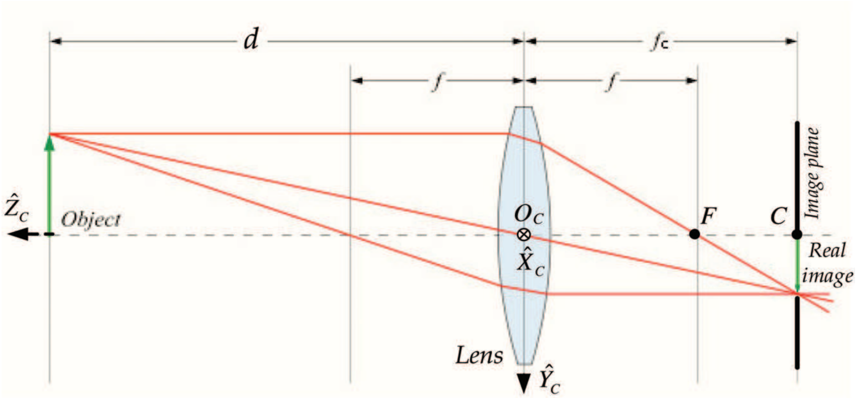


Fig. 3. Left side view of the camera-lens system

The camera focal length f_c might be roughly derived from the *thin lens formula*:

$$\frac{1}{f_C} + \frac{1}{d} = \frac{1}{f} \Rightarrow f_C = \frac{f\,d}{d - f} \tag{5}$$

Without loss of generality, let us assume that a lens has its focal length of f = 16 mm. The graph below represents the camera constant $f_C(d)$ as a function of the distance d.

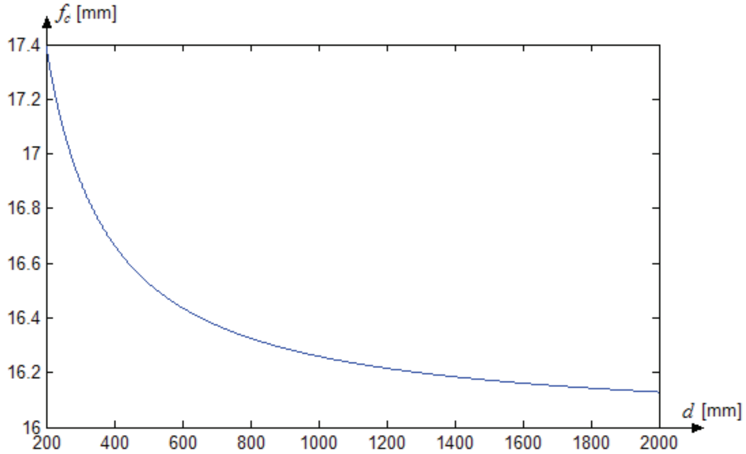


Fig. 4. Camera constant f_c in terms of the distance d

As can be seen from equation (5), when the distance goes to infinity, the camera constant equals to the focal length of the lens, what can be inferred from Fig. 4, as well. Since in industrial applications the distance ranges from 200 to 5000 mm, it is clear that the camera constant is always greater than the focal length of the lens. Because physical measurement of the distance is overly erroneous, it is generally recommended to use calibrating algorithms, like MCT, to extract this parameter. Let us assume for the time being that the camera matrix is represented by

$$K = \begin{bmatrix} f_C & 0 & 0 \\ 0 & f_C & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Principal point offset The location of the principal point C on the sensor plane is most important since it strongly influences the precision of measurements. As has already been mentioned above, the principal point is the place where the principal axis meets the sensor plane. In CCD camera systems the term principal axis refers to the lens, as shown in both Fig. 1 and Fig. 3. Thus it is not the camera but the lens mounted on the camera that determines this point and the camera's coordinate system.

In (1) it is assumed that the origin of the sensor plane is at the principal point, so that the Sensor Coordinate System is parallel to the Camera CS and their origins are only the camera constant away from each other. It is, however, not truthful in reality. Thus we have to compute a principal point offset $\begin{bmatrix} C_{0x} & C_{0y} \end{bmatrix}^{\mathrm{T}}$ from the sensor center, and extend the camera matrix by this parameter so that the projected point can be correctly determined in the Sensor CS (shifted parallel to the Camera CS). Consequently, we have the following mapping:

$$\begin{bmatrix} x^C & y^C & z^C \end{bmatrix}^{\mathrm{T}} \rightarrow \begin{bmatrix} f_C \dfrac{x^C}{z^C} + C_{Ox} & f_C \dfrac{x^C}{z^C} + C_{Oy} \end{bmatrix}^{\mathrm{T}}$$

Introducing this parameter to the camera matrix results in

$$K = \begin{bmatrix} f_C & 0 & C_{Ox} \\ 0 & f_C & C_{Oy} \\ 0 & 0 & 1 \end{bmatrix}$$

As CCD cameras are never perfect, it is most likely that CCD chips have pixels, which are not of the shape of a square. The image coordinates, however, are measured in square pixels. This has certainly an extra effect of introducing unequal scale factors in each direction. In particular, if the number of pixels per unit distance (per millimeter) in image coordinates are m_x and m_y in the directions x and y , respectively, then the camera transformation from space coordinates measured in millimeters to pixel coordinates can be gained by pre-multiplying the camera matrix M by a matrix factor diag(m_x, m_y, 1). The camera matrix can then be estimated as

$$K = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_c & 0 & C_{Ox} \\ 0 & f_c & C_{Oy} \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow K = \begin{bmatrix} f_{cp1} & 0 & C_{Oxp} \\ 0 & f_{cp2} & C_{Oyp} \\ 0 & 0 & 1 \end{bmatrix}$$

where $f_{cp1} = f_c m_x$ and $f_{cp2} = f_c m_y$ represent the focal length of the camera in terms of pixels in the x and y directions, respectively. The ratio f_{cp1} / f_{cp2} , called an *aspect ratio*, gives a simple measure of regularity meaning that the closer it is to 1 the nearer to squares are the pixels. It is very convenient to express the matrix M in terms of pixels because the data forming an image are determined in pixels and there is no need to re-compute the intern camera parameters into millimeters.

Skew coefficient Skewing does not exist in most regular cameras. However, in certain unusual instances it can be present. A skew parameter, which in CCD cameras relates to pixels, determines how pixels in a CCD array are skewed, that is to what extent the x and y axes of a pixel are not perpendicular. Principally, the CCD camera model assumes that the image has been stretched by some factor in the two axial directions. If it is stretched in a non-axial direction, then skewing results. Taking the skew parameter into considerations yields the following form of the camera matrix:

$$K = \begin{bmatrix} f_{Cp1} & 0 & C_{Oxp} \\ 0 & f_{Cp2} & C_{Oyp} \\ 0 & 0 & 1 \end{bmatrix}$$

This form of the camera matrix (M) allows us to calculate the pixel coordinates of a point $\vec{r}^C$ cast from a 3D scene into the sensor plane (assuming that we know the original coordinates):

$$\begin{bmatrix} x^S \\ y^S \\ 1 \end{bmatrix} = \begin{bmatrix} f_{Cp1} & 0 & C_{Oxp} \\ 0 & f_{Cp2} & C_{Oyp} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{6}$$

Since images are recorded through the CCD sensor, we have to consider closely the image plane, too. The origin of the sensor plane lies exactly in the middle, while the origin of the Image CS is always located in the upper left corner of the image. Let us assume that the principal point offset is known and the resolution of the camera is $\mathrm{N}_x \times \mathrm{N}_y$ pixels. As the center of the sensor plane lies intuitively in the middle of the image, the principal point offset, denoted as $[cc_x \quad cc_y]^{\mathrm{T}}$, with respect to the Image CS is $\begin{bmatrix} \frac{\mathrm{N}_x}{2} + C_{Oxp} & \frac{\mathrm{N}_y}{2} + C_{Oyp} \end{bmatrix}^{\mathrm{T}}$.
Hence the full form of the camera matrix suitable for the pinhole camera model is

$$M = \begin{bmatrix} f_{cp1} & s & cc_x \\ 0 & f_{cp2} & cc_y \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Consequently, a complete equation describing the projection of the point $\vec{r}^C = \begin{bmatrix} x^C & y^C & z^C \end{bmatrix}^{\mathrm{T}}$ from the camera's three-dimensional scene to the point $\vec{r}^I = \begin{bmatrix} x^I & y^I \end{bmatrix}^{\mathrm{T}}$ in the camera's Image CS has the following form:

$$\begin{bmatrix} x^I \\ y^I \\ 1 \end{bmatrix} = \begin{bmatrix} f_{cp1} & 0 & cc_x \\ 0 & f_{cp2} & cc_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{8}$$

where x_d and y_d stand for the normalized distorted camera coordinates as in (3).

2.2 Conventions on the orientation matrix of the rigid body transformation

There are various industrial tasks in which a robotic plant can be utilized. For example, a robot with its tool mounted on a robotic flange can be used for welding, body painting or gripping objects. To automate this process, an object, a tool, and a complete mechanism itself have their own fixed coordinate systems assigned. These CSs are rotated and translated w.r.t. each other. Their relations are determined in the form of certain mathematical transformations T.
Let us assume that we have two coordinate systems $\{F1\}$ and $\{F2\}$ shifted and rotated w.r.t. to each other. The mapping ${}_{F1}T^{F2} = \left({}_{F1}R^{F2}, {}_{F1}K^{F2}\right)$ in a three-dimensional space can be represented by the following 4×4 homogenous coordinate transformation matrix:

$${}_{F1}T^{F2} = \begin{bmatrix} {}_{F1}R^{F2} & {}_{F1}K^{F2} \\ 0_{[1\times3]} & 1 \end{bmatrix} \tag{9a}$$

where ${}_{F1}R^{F2}$ is a 3×3 orthogonal *rotation* matrix determining the orientation of the {*F*2} CS with respect to the {*F*1} CS and ${}_{F1}K^{F2}$ is a 3×1 translation vector determining the position of the origin of the {*F*2} CS shifted with respect to the origin of the {*F*1} CS.
The matrix ${}_{F1}T^{F2}$ can be divided into two sub-matrices:

$$ {}_{F1}R^{F2} = \begin{bmatrix} r11_{F1F2} & r12_{F1F2} & r13_{F1F2} \\ r21_{F1F2} & r22_{F1F2} & r23_{F1F2} \\ r31_{F1F2} & r32_{F1F2} & r33_{F1F2} \end{bmatrix}, \quad {}_{F1}K^{F2} = \begin{bmatrix} kx_{F1F2} \\ ky_{F1F2} \\ kz_{F1F2} \end{bmatrix} \tag{9b} $$

Due to its orthogonality, the rotation matrix *R* fulfills the condition $R^{\mathrm{T}}R = I$, where *I* is a 3×3 identity matrix.
It is worth noticing that there are a great number (about 24) of conventions of determining the rotation matrix *R*. We describe here two most common conventions, which are utilized by leading robot-producing companies, i.e. the ZYX-Euler-angles and the unit-quaternion notations.
Euler angles notation The ZYX Euler angles representation can be described as follows. Let us first assume that two CS, {*F*1} and {*F*2}, coincide with each other. Then we rotate the {*F*2} CS by an angle *A* around the $\hat{Z}_{F2}$ axis, then by an angle *B* around the $\hat{Y}'_{F2}$ axis, and finally by an angle *C* around the $\hat{X}''_{F2}$ axis. The rotations refer to the rotation axes of the {*F*2} CS instead of the fixed {*F*1} CS. In other words, each rotation is carried out with respect to an axis whose position depends on the previous rotation, as shown in Fig. 5.

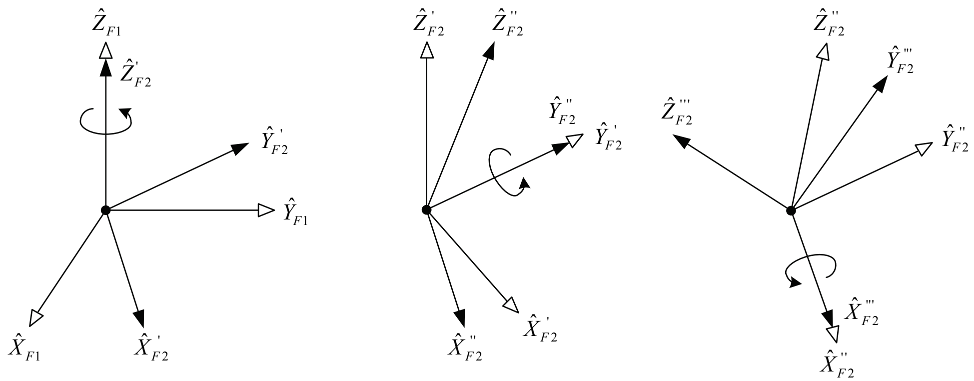


Fig. 5. Representation of the rotations in terms of the ZYX Euler angles

In order to find the rotation matrix ${}_{F1}R^{F2}$ from the {*F*1} CS to the {*F*2} CS, we introduce indirect {*F*2′} and {*F*2″} CSs. Taking the rotations as descriptions of these coordinate systems (CSs), we write:

$$ {}_{F1}R^{F2} = {}_{F1}R^{F2'}\,{}_{F2'}R^{F2''}\,{}_{F2''}R^{F2} $$

In general, the rotations around the $\hat{Z}, \hat{Y}, \hat{X}$ axes are given as follows, respectively:

$$R_{\hat{Z}} = \begin{bmatrix} \cos(A) & -\sin(A) & 0 \\ \sin(A) & \cos(A) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_{\hat{Y}} = \begin{bmatrix} \cos(B) & 0 & \sin(B) \\ 0 & 1 & 0 \\ -\sin(B) & 0 & \cos(B) \end{bmatrix} \quad R_{\hat{X}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(C) & -\sin(C) \\ 0 & \sin(C) & \cos(C) \end{bmatrix}$$

By multiplying these matrices we get a compose formula for the rotation matrix $R_{\hat{Z}\hat{Y}\hat{X}}$:

$$R_{\hat{Z}\hat{Y}\hat{X}} = \begin{bmatrix} \cos(A)\cos(B) & \cos(A)\sin(B)\sin(C)-\sin(A)\cos(C) & \cos(A)\sin(B)\cos(C)+\sin(A)\sin(C) \\ \sin(A)\cos(B) & \sin(A)\sin(B)\sin(C)+\cos(A)\cos(C) & \sin(A)\sin(B)\cos(C)-\cos(A)\sin(C) \\ -\sin(B) & \cos(B)\sin(C) & \cos(B)\cos(C) \end{bmatrix} \quad (10)$$

As the above formula implies, the rotation matrix is actually described by only 3 parameters, *i.e.* the Euler angles A, B and C of each rotation, and not by 9 parameters, as suggested (9b). Hence the transformation matrix T is described by 6 parameters overall, also referred to as a *frame*.

Let us now describe the transformation between points in a three-dimensional space, by assuming that the $\{F2\}$ CS is moved by a vector $K = [kx_{F1F2} \quad ky_{F1F2} \quad kz_{F1F2}]^{\mathrm{T}}$ w.r.t. the $\{F1\}$ CS in three dimensions and rotated by the angles A, B and C following the ZYX Euler angles convention. Given a point $\vec{r}^{\,F2} = [x^{F2} \quad y^{F2} \quad z^{F2}]^{\mathrm{T}}$, a point $\vec{r}^{\,F1} = [x^{F1} \quad y^{F1} \quad z^{F1}]^{\mathrm{T}}$ is computed in the following way:

$$\begin{bmatrix} x^{F1} \\ y^{F1} \\ z^{F1} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(A)\cos(B) & \cos(A)\sin(B)\sin(C)-\sin(A)\cos(C) & \cos(A)\sin(B)\cos(C)+\sin(A)\sin(C) & kx_{F1F2} \\ \sin(A)\cos(B) & \sin(A)\sin(B)\sin(C)+\cos(A)\cos(C) & \sin(A)\sin(B)\cos(C)-\cos(A)\sin(C) & ky_{F1F2} \\ -\sin(B) & \cos(B)\sin(C) & \cos(B)\cos(C) & kz_{F1F2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^{F2} \\ y^{F2} \\ z^{F2} \\ 1 \end{bmatrix} \quad (11)$$

Using (9) we can also represent the above in a concise way:

$$\begin{bmatrix} x^{F1} \\ y^{F1} \\ z^{F1} \\ 1 \end{bmatrix} = \begin{bmatrix} r11_{F1F2} & r12_{F1F2} & r13_{F1F2} & kx_{F1F2} \\ r21_{F1F2} & r22_{F1F2} & r23_{F1F2} & ky_{F1F2} \\ r31_{F1F2} & r32_{F1F2} & r33_{F1F2} & kz_{F1F2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^{F2} \\ y^{F2} \\ z^{F2} \\ 1 \end{bmatrix} = {}_{F1}T^{F2} \begin{bmatrix} x^{F2} \\ y^{F2} \\ z^{F2} \\ 1 \end{bmatrix} \quad (12)$$

After decomposing this transformation into rotation and translation matrices, we have:

$$\begin{bmatrix} x^{F1} \\ y^{F1} \\ z^{F1} \end{bmatrix} = \begin{bmatrix} r11_{F1F2} & r12_{F1F2} & r13_{F1F2} \\ r21_{F1F2} & r22_{F1F2} & r23_{F1F2} \\ r31_{F1F2} & r32_{F1F2} & r33_{F1F2} \end{bmatrix} \begin{bmatrix} x^{F2} \\ y^{F2} \\ z^{F2} \end{bmatrix} + \begin{bmatrix} kx_{F1F2} \\ ky_{F1F2} \\ kz_{F1F2} \end{bmatrix} = {}_{F1}R^{F2} \begin{bmatrix} x^{F2} \\ y^{F2} \\ z^{F2} \end{bmatrix} + {}_{F1}K^{F2} \quad (13)$$

There from, knowing the rotation R and the translation K from the first CS to the second CS in the three-dimensional space and having the coordinates of a point defined in the second CS, we can compute its coordinates in the first CS.

Unit quaternion notation Another notation for rotation, widely utilized in machine vision industry and computer graphics, refers to unit quaternions. A quaternion, $p = (p_0, p_1, p_2, p_3) = (p_0, \vec{p})$, is a collection of four components, first of which is taken as a scalar and the other three form a vector. Such an entity can thus be treated in terms of complex numbers what allows us to re-write it in the following form:

$$p = p_0 + i \cdot p_1 + j \cdot p_2 + k \cdot p_3$$

where i, j, k are imaginary numbers. This means that a real number (scalar) can be represented by a purely real quaternion and a three-dimensional vector by a purely imaginary quaternion. The conjugate and the magnitude of a quaternion can be determined in a way similar to the complex numbers calculus:

$$p^* = p_0 - i \cdot p_1 - j \cdot p_2 - k \cdot p_3 \quad , \quad \|p\| = \sqrt{p_0^{\ 2} + p_1^{\ 2} + p_2^{\ 2} + p_3^{\ 2}}$$

With another quaternion $q = (q_0, q_1, q_2, q_3) = (q_0, \vec{q})$ in use, the sum of them is

$$p + q = (p_0 + q_0, \vec{p} + \vec{q})$$

and their (non-commutative) product can be defined as

$$p \cdot q = (p_0 q_0 - \vec{p}\vec{q}, p_0 \vec{q} + q_0 \vec{p} + \vec{p}\vec{q})$$

The latter can also be written in a matrix form as

$$p \cdot q = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \cdot q = P \cdot q$$

or

$$q \cdot p = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{bmatrix} \cdot q = \overline{P} \cdot q$$

where P and $\overline{P}$ are 4×4 orthogonal matrices.
Dot product of two quaternions is the sum of products of corresponding elements:

$$p \circ q = p_0 q_0 + p_1 q_1 + p_2 q_2 + p_3 q_3$$

A unit quaternion $\|p\| = 1$ has its inverse equal its conjugate:

$$p^{-1} = \left(\frac{1}{p \circ p} \right) p^* = p^*$$

as the square of the magnitude of a quaternion is a dot product of the quaternion with itself:

$$\|p\|^2 = p \circ p$$

It is clear that the vector's length and angles relative to the coordinate axes remain constant after rotation. Hence rotation also preserves dot products. Therefore it is possible to represent the rotation in terms of quaternions. However, simple multiplication of a vector by a quaternion would yield a quaternion with a real part (vectors are quaternions with imaginary parts only). Namely, if we express a vector $\vec{q}$ from a three-dimensional space as a unit quaternion $q = \left(0, \vec{q}\right)$ and perform the operation with another unit quaternion p

$$\vec{q}\,' = p \cdot q = \left(q_0{}', q_1{}', q_2{}', q_3{}'\right)$$

then we attain a quaternion which is not a vector. Thus we use composite product in order to rotate a vector into another one while preserving its length and angles:

$$\vec{q}\,' = p \cdot q \cdot p^{-1} = p \cdot q \cdot p^{*} = \left(0, q_1{}', q_2{}', q_3{}'\right)$$

We can prove this by the following expansion:

$$p \cdot q \cdot p^{*} = (Pq)p^{*} = \overline{P}^{\mathrm{T}}(Pq) = \left(\overline{P}^{\mathrm{T}} P\right)q$$

where

$$\overline{P}^{\mathrm{T}} P = \begin{bmatrix} p \circ p & 0 & 0 & 0 \\ 0 & \left(p_0^2 + p_1^2 - p_2^2 - p_3^2\right) & 2\left(p_1 p_2 - p_0 p_3\right) & 2\left(p_1 p_3 - p_0 p_2\right) \\ 0 & 2\left(p_1 p_2 - p_0 p_3\right) & \left(p_0^2 - p_1^2 + p_2^2 - p_3^2\right) & 2\left(p_2 p_3 - p_0 p_1\right) \\ 0 & 2\left(p_3 p_1 - p_0 p_2\right) & 2\left(p_3 p_2 - p_0 p_1\right) & \left(p_0^2 - p_1^2 - p_2^2 + p_3^2\right) \end{bmatrix}$$

Therefore, if q is purely imaginary then q' is purely imaginary, as well. Moreover, if p is a unit quaternion, then $p \circ p = 1$, and P and $\overline{P}$ are orthonormal. Consequently, the 3×3 lower right-hand sub-matrix is also orthonormal and represents the rotation matrix as in (9b).
The quaternion notation is closely related to the axis-angle representation of the rotation matrix. A rotation by an angle θ about a unit vector $\hat{\omega} = \left[\omega_x \quad \omega_y \quad \omega_z\right]^{\mathrm{T}}$ can be determined in terms of a unit quaternion as:

$$p = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\left(i\omega_x + j\omega_y + k\omega_z\right)$$

In other words, the imaginary part of the quaternion represents the vector of rotation and the real part along with the magnitude of the imaginary part provides the angle of rotation.
There are several important advantages of unit quaternions over other conventions. Firstly, it is much simpler to enforce the constraint on the quaternion to have a unit magnitude than to implement the orthogonality of the rotation matrix based on Euler angles. Secondly, quaternions avoid the *gimbal lock* phenomenon occurring when the pitch angle is 90°. Then yaw and roll angles refer to the same motion what results in losing one degree of freedom. We postpone this issue until Section 3.3.
Finally, let us study the following example. In Fig. 6 there are four CSs: {A}, {B}, {C} and {D}. Assuming that the transformations $_AT^B$, $_BT^C$ and $_AT^D$ are known, we want to find the

other two, ${}_{A}T^{C}$ and ${}_{D}T^{C}$. Note that there are 5 loops altogether, *ABCD, ABC, ACD, ABD* and *BCD,* that connect the origins of all CSs. Thus there are several ways to find the unknown transformations. We find ${}_{A}T^{C}$ by means of the loop *ABC,* and ${}_{D}T^{C}$ by following the loop *ABCD*. Writing the matrix equation for the first loop we immediately obtain:

$${}_{A}T^{C} = {}_{A}T^{B}\,{}_{B}T^{C}$$

Writing the equation for the other loop we have:

$${}_{A}T^{B}\,{}_{B}T^{C} = {}_{A}T^{D}\,{}_{D}T^{C} \Rightarrow {}_{D}T^{C} = \left({}_{A}T^{D}\right)^{-1}\,{}_{A}T^{B}\,{}_{B}T^{C}$$

To conclude, given that the transformations can be placed in a closed loop and only one of them is unknown, we can compute the latter transformation based on the known ones. This is a principal property of transformations in vision-guided robot positioning applications.

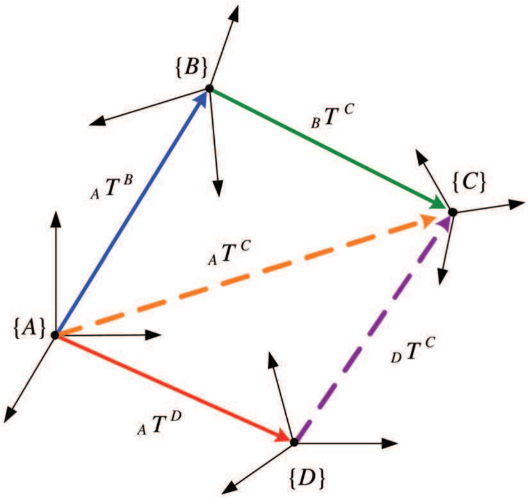


Fig. 6. Transformations based on closed loops

2.3 Pose estimation – problem statement

There are many methods in the machine vision literature suitable for retrieving the information from a three-dimensional scene with the use of a single image or multiple images. Most common cases include single and stereo imaging, though recently developed applications in robotic guidance use 4 or even more images at a time. In this Section we characterize few methods of pose estimation to give the general idea of how they can be utilized in robot positioning systems.

Why do we compute the pose of the object relative to the camera? Let us suppose that we have a robot-camera-gripper positioning system, which has already been calibrated. In robot positioning applications the vision sensor acts somewhat as a medium only. It determines the pose of the object that is then transformed to the Gripper CS. This means that the pose of the object is estimated with respect to the gripper and the robot 'knows' how to grip the object.

In another approach we do not compute the pose of the object relative to the camera and then to the gripper. Single or multi camera systems calculate the coordinates of points at the calibration stage, and then perform the calculation at each position while the system is running. Based on the computed coordinates, a geometrical motion of a given camera from the calibrated position to its actual position is processed. Knowing this motion and the geometrical relation between the camera and the gripper, the gripping motion can then be computed so that the robot 'learns' where its gripper is located w.r.t to the object, and then the gripping motion can follow.

2.3.1 Computing 3D points using stereovision

When a point in a 3D scene is projected onto a 2D image plane, the depth information is lost. The simplest method to render this information is stereovision. The 3D coordinates of any point can be computed provided that this point is visible in two images (1 and 2) and the intern camera parameters together with the geometrical relation between stereo cameras are known.

Rendering 3D point coordinates based on image data is called *inverse point mapping*. It is a very important issue in machine vision because it allows us to compute the camera motion from one position to another. We shall now derive a mathematical formula for rendering the 3D point coordinates using stereovision.

Let us denote the 3D point $\vec{r}$ in the Camera 1 CS as $\vec{r}^{\,C1} = \begin{bmatrix} x^{C1} & y^{C1} & z^{C1} & 1 \end{bmatrix}^{\mathrm{T}}$. The same point in the Camera 2 CS will be represented by $\vec{r}^{\,C2} = \begin{bmatrix} x^{C2} & y^{C2} & z^{C2} & 1 \end{bmatrix}^{\mathrm{T}}$. Moreover, let the geometrical relation between these two cameras be given as the transformation from Camera 1 to Camera 2 ${}_{C1}T^{C2} = \left({}_{C1}R^{C2}, {}_{C1}K^{C2}\right)$, their calibration matrices be M_{C1} and M_{C2}, and the projected image points be $\vec{r}^{\,I1} = \begin{bmatrix} x^{I1} & y^{I1} & 1 \end{bmatrix}^{\mathrm{T}}$ and $\vec{r}^{\,I2} = \begin{bmatrix} x^{I2} & y^{I2} & 1 \end{bmatrix}^{\mathrm{T}}$, respectively.

There is no direct way to transform distorted image coordinates into undistorted ones because (3) and (4) are not linear. Hence, the first step would be to solve these equations iteratively. For the sake of simplicity, however, let us assume that our camera model is free of distortion. In Section 5 we will verify how these parameters affect the precision of measurements. In the considered case, the normalized distorted coordinates match the normalized undistorted ones: $x_d = x_{norm}$ and $y_d = y_{norm}$. As the stereo images are related with each other through the transformation ${}_{C1}T^{C2}$, the pixel coordinates of Image 2 can be transformed to the plane of Image 1. Thus combining (8) and (13), and eliminating the coordinates x and y yields:

$$M_{C1}^{-1}\vec{r}^{\,I1}z^{C1} = {}_{C1}R^{C2}M_{C2}^{-1}\vec{r}^{\,I2}z^{C2} + {}_{C1}K^{C2} \tag{14}$$

This overconstrained system is solved by the linear least squares method (LS) and computation of the remaining coordinates in {C1} and {C2} comes straightforward. Such an approach based on (14) is called *triangulation*.

It is worth mentioning that the stereo camera configuration has several interesting geometrical properties, which can be used, for instance, to inform the operator that the system needs recalibration and/or to simplify the implementation of the image processing application (IPA) used to retrieve object features from the images. Namely, the only constraint of the stereovision systems is imposed by their epipolar geometry. An epipolar plane and an epipolar line represent epipolar geometry. The epipolar plane is defined by a

3D point in the scene and the origins of the two Camera CSs. On the basis of the projection of this point onto the first image, it is possible to derive the equation of the epipolar plane (characterized by a *fundamental matrix*) which has also to be satisfied by the projection of this point onto the second image plane. If such a plane equation condition is not satisfied, then an error offset can be estimated. When, for instance, the frequency of the appearance of such errors exceeds an *a priori* defined threshold, it can be treated as a warning sign of the necessity for recalibration. The epipolar line is also quite useful. It is the straight line of intersection of the epipolar plane with the image plane. Consequently, a 3D point projected onto one image generates a line in the other image on which its corresponding projection point must lie. This feature is extremely important when creating an IPA. Having found one feature in the image reduces the scope of the search for its corresponding projection in the other image from a region to a line. Since the main problem of stereovision IPAs lies in locating the corresponding image features (which are projections of the same 3D point), this greatly improves the efficiency of IPAs and yet eases the process of creating them.

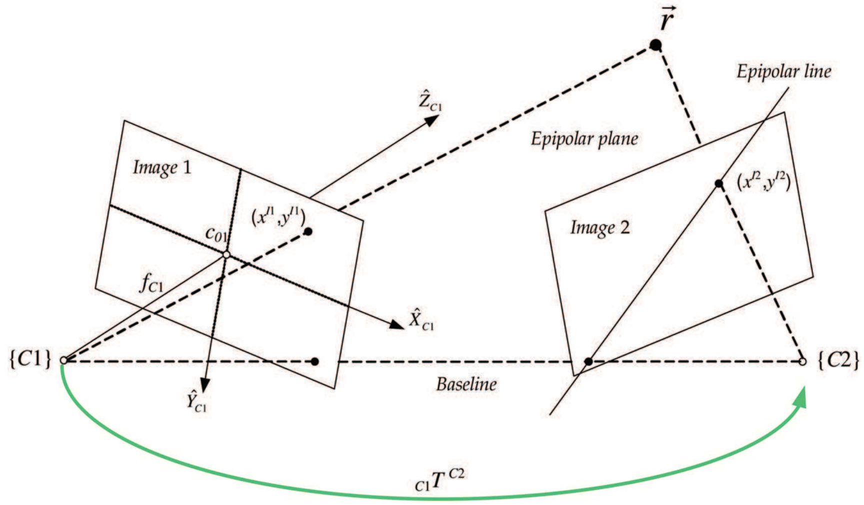


Fig. 7. Stereo–image configuration with epipolar geometry

2.3.2 Single image pose estimation

There are two methods of pose estimation utilized in 3D robot positioning applications. A first one, designated as *3D-3D estimation*, refers to computing the actual pose of the camera either w.r.t. the camera at the calibrated position or w.r.t. the actual position of the object. In the first case, the 3D point coordinates have to be known in both camera positions. In the latter, the points have to be known in the Camera CS as well as in the Object CS. Points defined in the Object CS can be taken from its CAD model (therefore called *model points*).

The second type of pose estimation is called *2D-3D estimation* and is used only by the gripping systems equipped with a single camera. It consists in computing the pose of the object with respect to the actual position of the camera given the 3D model points and their projected pixel coordinates. The main advantage of this approach over the first one is that it does not need to calculate the 3D points in the Camera CS to find the pose. Its disadvantage lies in only iterative implementations of the computations. Nevertheless, it is widely utilized in camera calibration procedures.

The assessment of camera motions or else the poses of the camera at the actual position relative to the pose of the camera at the calibration position are also known as *relative orientation*. The estimation of the transformation between the camera and the object is identified as *exterior orientation*.

Relative orientation

Let us consider the following situation. During the calibration process we have positioned the cameras, measured *n* 3D object points ($n \geq 3$) in a chosen Camera CS {*Y*}, and taught the robot how to grip the object from that particular camera position. We could measure the points using, for instance, stereovision, linear *n*-point algorithms, or structure-from-motion algorithms. Let us denote these points as $\vec{r}_1^{\,Y},...,\vec{r}_n^{\,Y}$. Now, we move the camera-robot system to another (actual) position in order to get another measurement of the same points (in the Camera CS {*X*}). This time they have different coordinates as the Camera CS has been moved. We denote these points as $\vec{r}_1^{\,X},...,\vec{r}_n^{\,X}$, where for an *i*-th point we have: $\vec{r}_i^{\,Y} \leftrightarrow \vec{r}_i^{\,X}$, meaning that the points correspond to each other. From Section 2.2 we know that there exists a mapping which transforms points $\vec{r}^{\,X}$ to points $\vec{r}^{\,Y}$. Note that this transformation implies the rigid motion of the camera from the calibrated position to the actual position. As will be shown in Section 3.2, knowing it, the robot is able to grip the object from the actual position. We can also consider these pairs of points as defined in the Object CS ($\vec{r}_1^{\,X},...,\vec{r}_n^{\,X}$) and in the Camera CS ($\vec{r}_1^{\,Y},...,\vec{r}_n^{\,Y}$). In such a case the mapping between these points describes the relation between the Object and the Camera CSs. Therefore, in general, given the points in these two CSs, we can infer the transformation between them from the following equation:

$$\vec{r}^{\,Y}_{[4\times n]} = T_{[4\times 4]}\vec{r}^{\,X}_{[4\times n]}$$

After rearranging and adding noise η to the measurements, we obtain:

$$\vec{r}_n^{\,Y} = R \cdot \vec{r}_n^{\,X} + K + \eta_n$$

One of the ways of solving the above equation consists in setting up a least squares equation and minimizing it, taking into account the constraint of orthogonality of the rotation matrix. For example, Haralick *et al.* (1989) describe iterative and non-iterative solutions to this problem. Another method, developed by Weinstein (1998), minimizes the summed-squared-distance between three pairs of corresponding points. He derives an analytic least squares fitting method for computing the transformation between these points. Horn (1987) approaches this problem using unit quaternions and giving a closed-form solution for any number of corresponding points.

Exterior orientation

The problem of determining the pose of an object relative to the camera based on a single-image has found many relevant applications in machine vision for object gripping, camera calibration, hand-eye calibration, cartography, etc. It can be easily stated more formally: given a set of (model) points that are described in the Object CS, the projections of these points onto an image plane, and the intern camera parameters, determine the rotation R and translation K between the object centered and the camera centered coordinate system.

As has been mentioned, this problem is labeled as the *exterior orientation* problem (in the photogrammetry literature, for instance). The dissertation by Szczepanski (1958) surveys

nearly 80 different solutions beginning with the one given by Schrieber of Karlsruhe in the year 1879. A first robust solution, identified a *RANSAC paradigm*, has been delivered by Fischler and Bolles (1981), while Wrobel (1992) and Thomson (1966) discuss configurations of points for which the solution is unstable. Haralick *et al.* (1989) introduced three iterative algorithms, which simultaneously compute both object pose w.r.t. the camera and the depths values of the points observed by the camera. A subsequent method represents rotation using Euler angles, where the equations are linearized by a Newton's first-order approximation. Yet another approach solves linearized equations using M-estimators.
It has to be emphasized that there exist more algorithms for solving the 2D-3D estimation problem. Some of them are based on minimizing the error functions derived from the collinearity condition of both the object-space and the image-space error vector. Certain papers (Schweighofer & Pinz, 2006; Lu *et al.*, 1998; Phong *et al.*, 1995) provide us with the derivation of these functions and propose iterative algorithms for solving them.

3. 3D robot positioning system

The calibrated vision guided three-dimensional robot positioning system, able to adjust the robot to grip the object deviated in 6DOF, comprises the following three consecutive fundamental steps:

1. Identification of object features in single or multi images using a custom image processing application (IPA).
2. Estimation of the relative or exterior orientation of the camera.
3. Computation of the transformation determining the gripping motion.

The calibration of the vision guided gripping systems involves three steps, as well. In the first stage the image processing software is taught some specific features of the object in order to detect them at other object/robot positions later on. The second step performs derivation of the camera matrix and hand-eye transformations through calibration relating the camera with the flange (end-effector) of the robot. This is a crucial stage, because though the camera can estimate algorithmically the actual pose of the object relative to itself, the object's pose has to be transformed to the gripper (also calibrated against the end-effector) in order to adjust the robot gripper to the object. This adjustment means a motion of the gripper from the position where the object features are determined in the images to the position where the object is gripped. The robot knows how to move its gripper along the motion trajectory because it is calibrated beforehand, what constitutes the third step.

3.1 Coordinate systems

In order do derive the transformations relating each component of the positioning system it is necessary to fix definite coordinate systems to these components. The robot positioning system (Kowalczuk & Wesierski, 2007) presented in this chapter is guided by stereovision and consists of the following coordinate systems (CS):

1. Robot CS, $\{R\}$
2. Flange CS, $\{F\}$
3. Gripper CS, $\{G\}$
4. Camera 1 CS, $\{C1\}$
5. Camera 2 CS, $\{C2\}$
6. Sensor 1 CS of Camera 1, $\{S1\}$
7. Sensor 2 CS of Camera 2, $\{S2\}$
8. Image 1 CS of Camera 1, $\{I1\}$

9. Image 2 CS of Camera 2, {*I*2}
10. Object CS, {*W*}.

The above CSs, except for the Sensor and Image CSs (discussed in Section 2.1), are three-dimensional Cartesian CSs translated and rotated with respect to each other, as depicted in Fig. 8. The Robot CS has its origin in the root of the robot. The Flange CS is placed in the middle of the robotic end-effector. The Gripper CS is located on the gripper within its origin, called Tool Center Point (TCP), defined during the calibration process. The center of the Camera CS is placed in the camera projection center O_C. As has been shown in Fig.1, the Camera principal axis determines the $\hat{Z}_C$-axis of the Camera CS pointing out of the camera in positive direction, the $\hat{Y}_C$-axis pointing downward and the $\hat{X}_C$-axis pointing to the left as one looks from the front. Apart from the intern parameters, the camera has extern parameters as well. They are the translation vector *K* and the three Euler angles *A*, *B*, *C*. The extern parameters describe translation and rotation of the camera with respect to any CS, and, in Fig. 8, with respect to the Flange CS, thus forming the hand-eye transformation. The Object CS has its origin at an arbitrary point/feature defined on the object. The other points determine the object's axes and orientation.

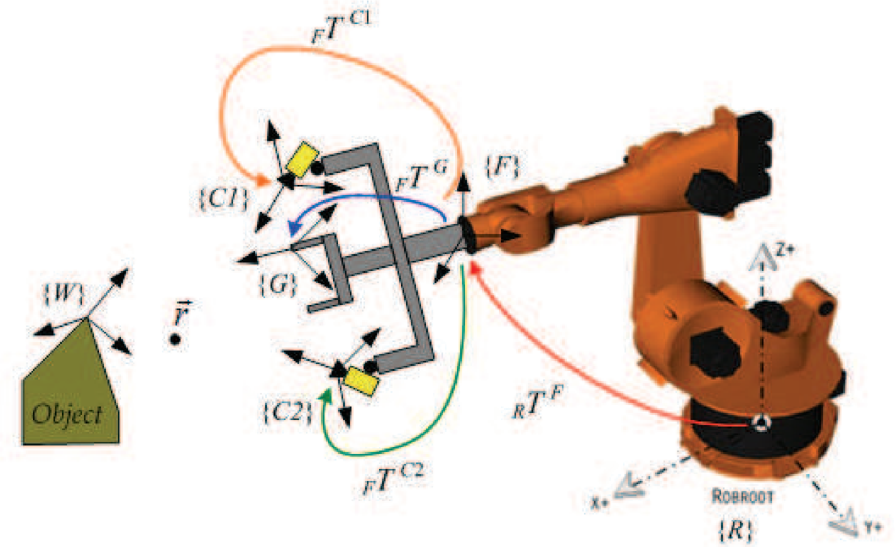


Fig. 8. Coordinate systems of the robot positioning system

3.2 Realization of gripping

In Section 2.3.2 we have shortly described two methods for gripping the object. We refer to them as the exterior and the relative orientation methods. In this section we explain how these methods are utilized in vision guided robot positioning systems and derive certain mathematical equations of concatenated transformations.

In order to grip an object at any position the robot has to be first taught the gripping motion from a position at which it can identify object features. This motion embraces three positions and two partial motions, first, a point-to-point movement (PTP), and then a linear

movement (LIN). The point-to-point movement means a possibly quickest way of moving the tip of the tool (TCP) from a current position to a programmed end position. In the case of linear motion, the robot always follows the programmed straight line from one point to another.
The robot is jogged to the first position in such a way that it can determine at least 3 features of the object in two image planes {*I*1} and {*I*2}. This position is called Position 1 or a *'Look-Position'*. Then, the robot is jogged to the second position called a *'Before-Gripping-Position'*, denoted as *Gb*. Finally, it is moved to the third position called an *'After-Gripping-Position'*, symbolized by *Ga,* meaning that the gripper has gripped the object. Although the motion from the *'Look-Position'* to the *Gb* is programmed with a PTP command, the motion from *Gb* to *Ga* has to be programmed with a LIN command because the robot follows then the programmed linear path avoiding possible collisions. After saving all these three calibrated positions, the robot 'knows' that moving the gripper from the calibrated *'Look-Position'* to *Ga* means gripping the object (assuming that the object is static during this gripping motion).
For the sake of conceptual clarity let us assume that the positioning system has been fully calibrated and the following data are known:

- transformation from the Flange to the Camera 1 CS: ${}_{F}T^{C1}$
- transformation from the Flange to the Camera 2 CS: ${}_{F}T^{C2}$
- transformation from the Flange to the Gripper CS: ${}_{F}T^{G}$
- transformation from the Gripper CS at Position 1 (*'Look-Position'*) to the *'Before-Gripping-Position'*: ${}_{G}T^{Gb}$
- transformation from the *'Before-Gripping-Position'* to the *'After-Gripping-Position'*: ${}_{Gb}T^{Ga}$
- the pixel coordinates of the object features in stereo images when the system is positioned at the *'Look-Position'*.

Having calibrated the whole system allows us to compute the transformation from the Camera 1 to the Gripper CS ${}_{C1}T^{G}$ and from the Camera 1 to the Camera 2 CS. We find the first transformation using the equation below:

$${}_{C1}T^{G} = \left({}_{F}T^{C1}\right)^{-1} {}_{F}T^{G}$$

To find the latter transformation, we write:

$${}_{C1}T^{C2} = \left({}_{F}T^{C1}\right)^{-1} {}_{F}T^{C2}$$

Based on the transformation ${}_{C1}T^{C2}$ and on the pixel coordinates of the projected points, the system uses the triangulation method to calculate the 3D points in the Camera 1 CS at Position 1.
We propose now two methods to grip the object, assuming that the robot has changed its position from Position 1 to Position N, as depicted in Fig. 9.
Exterior orientation method for robot positioning This method is based on computing the transformation ${}_{C1}T^{W}$ from the camera to the object using the 3D model points determined in the Object CS {*W*1} and the pixel coordinates of these points projected onto the image. The exterior orientation methods described in Section 2.3.2 are used to obtain ${}_{C1}T^{W}$.

The movement of the positioning system, shown in Fig. 9, from Position 1 to an arbitrary Position N can be presented in three ways:

- the system changes its position relative to a constant object position
- the object changes its position w.r.t. a constant position of the system
- the system and the object both change their positions.

Note that, as the motion of the gripper between the *Gb* to the *Ga* Positions is programmed by a LIN command, the transformation ${}_{Gb}T^{Ga}$ remains constant.

Regardless of the current presentation, the two transformations ${}_{C1}T^{W}$ and ${}_{G}T^{Gb}$ change into ${}_{C1p}T^{W}$ and ${}_{Gp}T^{Gb}$, respectively, and they have to be calculated. Having computed ${}_{C1p}T^{W}$ by using exterior orientation algorithms, we write a *loop equation* for the concatenating transformations at Position N:

$$
{}_{C1}T^{W} = {}_{C1}T^{G}\,{}_{G}T^{Gb}\left({}_{Gp}T^{Gb}\right)^{-1}\left({}_{C1}T^{G}\right)^{-1}{}_{C1p}T^{W}
$$

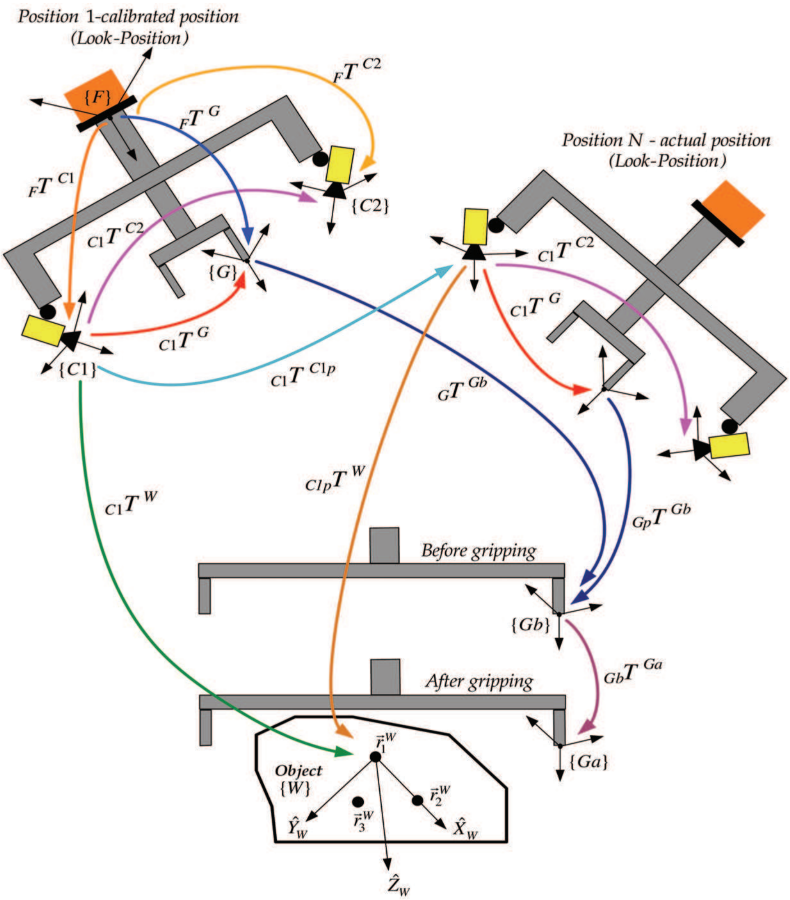


Fig. 9. Gripping the object

After rearranging, a new transformation from the Gripper CS at Position N to the Gripper CS at Position *Gb* can be shown as:

$$ {}_{Gp}T^{Gb} = \left({}_{C1}T^{G}\right)^{-1} {}_{C1p}T^{W} \left({}_{C1}T^{W}\right)^{-1} {}_{C1}T^{G}\ {}_{G}T^{Gb} \tag{15a} $$

Relative orientation method for robot positioning After measuring at least three 3D points in the Camera 1 CS at Position 1 and at Position N, we can calculate the transformation ${}_{C1}T^{C1p}$ between these two positions of the camera (confer Fig. 9), using the methods mentioned in Section 2.3.2. A straightforward approach is to use 4 points to derive ${}_{C1}T^{C1p}$ analytically. It is possible to do so based on only 3 points (which cannot be collinear) since the fourth one can be taken from the (vector) cross product of two vectors representing the 3 points hooked at one of the primary points. Though we sacrifice here the orthogonality constraint of the rotation matrix.
We write the following *loop equation* relating the camera motion, constant camera-gripper transformation, and the gripping motions:

$$ {}_{C1}T^{G}\ {}_{G}T^{Gb} = {}_{C1}T^{C1p}\ {}_{C1}T^{G}\ {}_{Gp}T^{Gb} $$

And after a useful rearrangement,

$$ {}_{Gp}T^{Gb} = \left({}_{C1}T^{G}\right)^{-1} \left({}_{C1}T^{C1p}\right)^{-1} {}_{C1}T^{G}\ {}_{G}T^{Gb} \tag{15b} $$

The new transformation ${}_{Gp}T^{Gb}$ determines a new PTP movement at Position N from *Gp* to *Gb*, while a final gripping motion LIN is determined from the constant transformation ${}_{Gb}T^{Ga}$. Consequently, equations (15a) and (15b) determine the sought motion trajectory which the robot has to follow in order to grip the object.
Furthermore, the transformations described by (15a, b) can be used to position the gripper while the object is being tracked. In order to predict the 3D image coordinates of at least three features one or two sampling steps ahead, a tracking algorithm can be implemented. With the use of such a tracking computation and based on the predicted points, the transformations ${}_{C1}T^{W}$ or ${}_{C1}T^{C1p}$ can be developed and substituted directly into equations (15a, b) so that the gripper could adjust its position relative to the object in the next sampling step.

3.3 Singularities

In systems using the Euler angles representation of orientation the movement ${}_{Gp}T^{Gb}$ has to be programmed in a robot encoder using the frame representation of the transformation ${}_{Gp}T^{Gb}$. The last column of the transformation matrix is the translation vector, directly indicating the first three parameters of the frame (*X*, *Y* and *Z*). The last three parameters *A*, *B* and *C* have to be computed based on the rotation matrix of the transformation. Let us assume that the rotation matrix has the form of (10). First, the angle *B* is computed in radians as

$$ B_1 = \pi + \arcsin(r31) \quad \vee \quad B_2 = -\arcsin(r31) \tag{16a} $$

Then, the angles A and C, based on the angle B, can be computed from the following recipes:

$$A_1 = \operatorname{a\,tan2}\left(\frac{r21}{\cos(B_1)}, \frac{r11}{\cos(B_1)}\right) \quad \vee \quad A_2 = \operatorname{a\,tan2}\left(\frac{r21}{\cos(B_2)}, \frac{r11}{\cos(B_2)}\right) \tag{16b}$$

$$C_1 = \operatorname{a\,tan2}\left(\frac{r32}{\cos(B_1)}, \frac{r33}{\cos(B_1)}\right) \quad \vee \quad C_2 = \operatorname{a\,tan2}\left(\frac{r32}{\cos(B_2)}, \frac{r33}{\cos(B_2)}\right) \tag{16c}$$

The above solutions results from solving the sine/cosine equations of the rotation matrix in (10). As the sine/cosine function is a multi-value function over the interval $(-\pi,+\pi)$, the equations (16a-16c) have two sets of solutions: $\{A_1, B_1, C_1\}$ and $\{A_2, B_2, C_2\}$. These two sets give the very same transformation matrix when substituted into (9b). Another common method of rendering these angles from the rotation matrix represents the Nonlinear Least Squares Fitting algorithm. Although its accuracy is higher than that of the technique (16a-16c), applying the NLSF algorithm to the positioning system guided by stereovision obviously deprives the system of its fully analytical development.

As (16a-16c) imply, the singularity of the system occurs in the case when the pitch angle equals ±90 deg, that is $r31$ equals ±1, since it results in zero values of the denominators. This case is called a *gimbal lock* and is a well-known problem in aerospace navigation systems. That is also why unit quaternions notation is preferred against the Euler angles notation.

Another singularity refers to the configuration of object features. Considering the relative orientation algorithms, the transformation between camera positions can only be computed under the condition that at least three not collinear object features are found (as has been discussed above the points have to span a plane in order to render the orientation). The exterior orientation algorithms have drawbacks, as well. Namely, there exist certain critical configurations of points for which the solution is unstable, as already mentioned in Section 2.3.2.

4. Calibration of the system – outline of the algorithms

There are many calibration methods able to find the transformation from the flange of the robot (hand) to the camera (eye). This calibration is called a hand-eye calibration. We demonstrate a classical approach initially introduced by Tsai & Lenz (1989). It states that when the camera undergoes a motion from Position i to Position i+1, described by the transformation ${}_{Ci}T^{Ci+1} = \left({}_{Ci}R^{Ci+1}, {}_{Ci}K^{Ci+1}\right)$, and the corresponding flange motion is ${}_{Fi}T^{Fi+1} = \left({}_{Fi}R^{Fi+1}, {}_{Fi}K^{Fi+1}\right)$, then they are coupled by the following hand-eye transformation ${}_{F}T^{C} = \left({}_{F}R^{C}, {}_{F}K^{C}\right)$, depicted in Fig. 10. This approach yields the subsequent equation:

$${}_{Fi}T^{Fi+1}\,{}_{F}T^{C} = {}_{F}T^{C}\,{}_{Ci}T^{Ci+1} \tag{17}$$

where ${}_{Ci}T^{Ci+1}$ is estimated from the images of the calibration rig using the MCT software, for instance, ${}_{Fi}T^{Fi+1}$ is known with the robot precision from the robot encoder, and ${}_{F}T^{C}$ is the unknown. This equation is also known as the Sylvester equation in systems theory. Since each transformation can be split into rotation and translation matrices, we easily land at

$${}_{Fi}R^{Fi+1}\,{}_{F}R^{C} = {}_{F}R^{C}\,{}_{Ci}R^{Ci+1} \tag{18a}$$

$$_{Fi}R^{Fi+1}{}_{F}K^{C}+_{Fi}K^{Fi+1}=_{F}R^{C}{}_{Ci}K^{Ci+1}+_{F}R^{C} \quad (18b)$$

Tsai and Lenz proposed a two-step method to solve the problem resented by (18a) and (18b). At first, they solve (18a) by least-square minimization of a linear system, obtained by using the axis-angle representation of the rotation matrix. Then, once $_{F}R^{C}$ is known, the solution for (18b) follows using the linear least squares method.

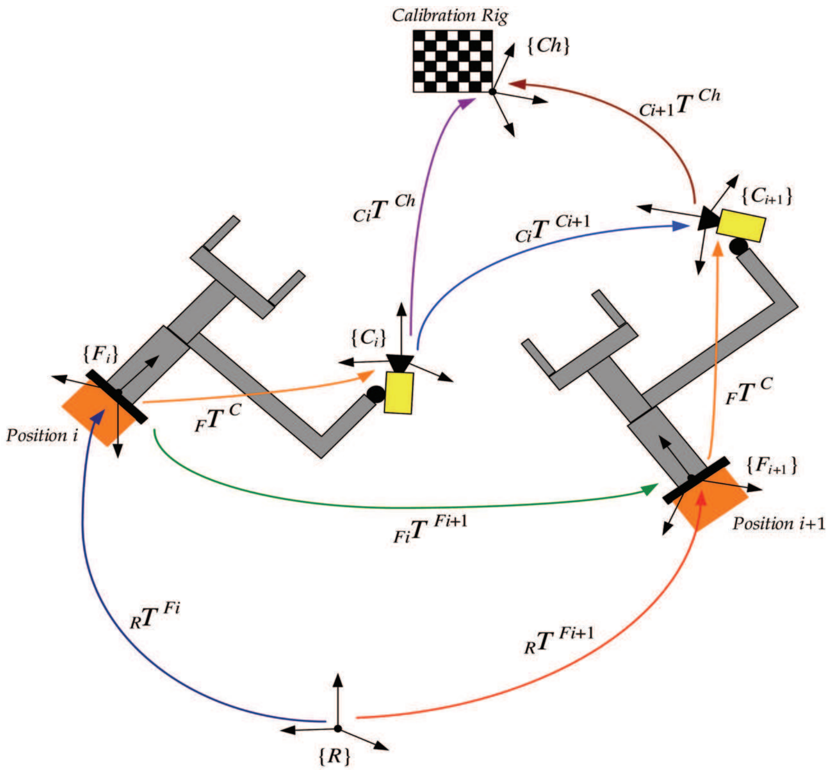


Fig. 10. Hand-Eye Calibration

In order to obtain a unique solution, there have to be at least two motions of the flange-camera system giving accordingly two pairs $\left({}_{F1}T^{F2},{}_{C1}T^{C2}\right)\left({}_{F2}T^{F3},{}_{C2}T^{C3}\right)$. Unfortunately, noise is inevitable in the measurement-based transformations $_{Fi}T^{Fi+1}$ and $_{Ci}T^{Ci+1}$. Hence it is useful to make more measurements and form a number of the transformations pairs $\left\{\left({}_{F1}T^{F2},{}_{C1}T^{C2}\right)\left({}_{F2}T^{F3},{}_{C2}T^{C3}\right)\ldots,\left({}_{Fi}T^{Fi+1},{}_{Ci}T^{Ci+1}\right)\ldots,\left({}_{Fk-1}T^{Fk},{}_{Ck-1}T^{Ck}\right)\right\}$, and, consequently, to find a transformation $_{F}T^{C}$ that minimizes an error criterion:

$$\varepsilon=\sum_{i=1}^{k} d\left({}_{Fi+1}T^{Fi}\,{}_{C}T^{F},{}_{C}T^{F}\,{}_{Ci+1}T^{Ci}\right)$$

where $d(\cdot,\cdot)$ stands for some distance metric on the Euclidean group. With the use of the Lie algebra the above minimization problem can be recast into a least squares fitting problem that admits an explicit solution. Specifically, given vectors $x_1,\ldots,x_k$, $y_1,\ldots,y_k$ in a Euclidean n-space, there exist explicit expressions for the orthogonal matrix R and translation K that minimize:

$$\varepsilon = \sum_{i=1}^{k} \left\| Rx_i + K - y_i \right\|^2$$

The best values of R and K turn out to depend on only the matrix $M = \sum_{i=1}^{k} x_i y_i^T$, while the rotation matrix R is then given by the following formula:

$$R = \left(M^T M\right)^{-1/2} M^T$$

Thus ${}_F R^C = \left(M^T M\right)^{-1/2} M^T$ represents in that case the computed rotation matrix of the hand-eye transformation ${}_F T^C$. After straightforward matrix operations on (18b), we acquire the following matrix equation for the translation vector ${}_F K^C$:

$$\begin{bmatrix} {}_{F1}K^{F2} - I \\ {}_{F2}K^{F3} - I \\ \vdots \\ {}_{Fk-1}K^{Fk} - I \end{bmatrix} {}_F K^C = \begin{bmatrix} {}_F R^C \, {}_{C1}K^{C2} - {}_{F1}K^{F2} \\ {}_F R^C \, {}_{C2}K^{C3} - {}_{F2}K^{F3} \\ \vdots \\ {}_F R^C \, {}_{Ck-1}K^{Ck} - {}_{Fk-1}K^{Fk} \end{bmatrix}$$

Using the least-squares method we obtain the solution for ${}_F K^C$.

Although simple to implement, the idea has a disadvantage as it solves (17) in two steps. Namely, the rotation matrix derived from (18a) propagates errors onto the translation vector derived from (18b). In the literature there is a large collection of hand-eye calibration methods, which have proved to be more accurate than the one discussed here. For instance, Daniilidis (1998) solves equation (17) simultaneously using dual quaternions. Andreff *et al.* (2001) uses the structure-from-motion algorithm to find the camera motion ${}_{Ci}T^{Ci+1}$ based on unknown scene parameters, and not by finding the transformations ${}_{Ci}T^{Ch}$ relating the scene (the calibration rig, here) with the camera. This is an interesting approach as it allows for a fully automatic calibration and thus reduces human supervision.

4.1 Manual hand-eye calibration – an evolutionary approach

After having derived the hand-eye transformations for both cameras (using MCT and Tsai method, for instance), it is essential to test their measurement accuracy. Based on images of a checkerboard, the MCT computed the transformation ${}_{Ci}T^{Ch}$ for each robot position with the estimation errors of ±2 mm. This has proved to be too large, as the point measurements resulted then in the repeatability error of even ±6 mm, which was unacceptable. Therefore, a genetic algorithm (GA) was utilized to correct the hand-eye parameters of both cameras, as they have a major influence on the entire accuracy of the system. We aimed to obtain the repeatability error of ±1 mm for each coordinate of all 3D points when compared to the points measured at the first vantage point.

Correcting the values of the hand-eye frames involves the following calibration steps: jogging the camera-robot system to K positions and saving pixel coordinates of N features seen in stereo images. Assuming that the accuracy of K measurements of N points

$\left(P_{1,1},\ldots,P_{N,1},\ldots,P_{1,K},\ldots,P_{N,K}\right)$ depends only on the hand-eye parameters (actually it depends also on the robot accuracy), the estimated values of both frames have to be modified by some yet unknown corrections:

$$\left(kx_{FC1}+\Delta kx_{FC1},ky_{FC1}+\Delta ky_{FC1},kz_{FC1}+\Delta kz_{FC1},A_{FC1}+\Delta A_{FC1},B_{FC1}+\Delta B_{FC1},C_{FC1}+\Delta C_{FC1}\right)$$

and

$$\left(kx_{FC2}+\Delta kx_{FC2},ky_{FC2}+\Delta ky_{FC2},kz_{FC2}+\Delta kz_{FC2},A_{FC2}+\Delta A_{FC2},B_{FC2}+\Delta B_{FC2},C_{FC2}+\Delta C_{FC2}\right).$$

The corrections, indicated here by Δ, have to be found based on a certain criterion. Thus, a sum of all repeatability errors $f(\Delta)$ of each coordinate of N=3 points has been chosen as a criterion to be minimized. The robot was jogged to K=10 positions. It is clear that the smaller the sum of the errors, the better the repeatability. Consequently, we seek for such corrections, which minimize the following function of the error sum:

$$\varepsilon = f(\Delta)=\sum_{n=1}^{N}\sum_{k=2}^{K}\left|P_{n1}(\Delta)-P_{nk}(\Delta)\right|, \qquad N=3, K=10 \tag{19}$$

As genetic algorithms effectively maximize the criterion function, while we wish to minimize (19), we transform it to:

$$g(\Delta)=C-f(\Delta)$$

The fitness function $g(\Delta)$ can then be maximized, with C being a constant scale factor ensuring that $g(\Delta)>0$,

$$\min f(\Delta)=\max g(\Delta)=\max\{-f(\Delta)\}$$

The function $g(\Delta)$ has 12 variables (6 corrections for each frame). Let us assume that the corrections for both translation vectors $dK=\{\Delta kx_{FC1},\Delta ky_{FC1},\Delta kz_{FC1},\Delta kx_{FC2},\Delta ky_{FC2},\Delta kz_{FC2}\}$ are within a searched interval $D_{dK}=[k_1,k_2]\subseteq R$ and the corrections for the Euler angles of both frames $dR=\{\Delta A_{FC1},\Delta B_{FC1},\Delta C_{FC1},\Delta A_{FC2},\Delta B_{FC2},\Delta C_{FC2}\}$ are within the interval $D_{dR}=[r_1,r_2]\subseteq R$, where $\Delta=\{dK,dR\}$ and $\underset{dK\in D_{dK}}{\forall}\ \underset{dR\in D_{dR}}{\forall}\ g(\Delta)>0$. Our desire is to maximize $g(\Delta)$ with a certain precision for dK and dR, say 10^{-n} and 10^{-m}, respectively. It means that we have to divide D_{dK} and D_{dR} into $(k_2-k_1)\cdot 10^n$ and $(r_2-r_1)\cdot 10^m$ equal intervals, respectively. Denoting a and b as the least numbers satisfying $(k_2-k_1)\cdot 10^n\le 2^a$ and $(r_2-r_1)\cdot 10^m\le 2^b$ implies that when the values $dK_i, i=1,\ldots,6$ and $dR_i, i=1,\ldots,6$ are coded as binary chains $(ch_i)_{bin},\quad i=1,\ldots,6$ and $(ch_j)_{bin},\quad j=1,\ldots,6$ of length a and b, respectively, then the binary representation of these values will satisfy the precision constraints. The decimal value of such binary chains can then be expressed as

$$dK_i = k_1+\frac{decimal\ \left[(ch_i)_{bin}\right]\cdot(k_2-k_1)}{2^a} \quad \text{and} \quad dR_i = r_1+\frac{decimal\ \left[(ch_j)_{bin}\right]\cdot(r_2-r_1)}{2^b} \tag{20}$$

Putting binary representations of the corrections $dK_i, i = 1,\ldots,6$ and $dR_i, i = 1,\ldots,6$ into one binary chain leads to a chromosome:

$$v = \{ch_i, ch_j\}, \qquad i, j = 1,\ldots,6$$

A reasonable number of chromosomes, forming a population, have to be defined to guarantee the effectiveness of a GA. The population is initiated completely randomly, *i.e.* bit by bit for each chromosome. In each generation we evaluate all chromosomes by first separating the chains ch_i and ch_j, then computing their decimal values using (20), and finally substituting the final results into $g(\Delta)$. The error function producing a sum of measurement errors for each chromosome, is used to compute the suitability of each chromosome in terms of the fitness function (in effect, by minimizing the repeatability error both frames are optimized). After evaluation, we select a new population according to the probability distribution based on suitability of each chromosome and with the use of recombination and mutation.

The most challenging part of creating a GA lies in determining the fitness function. Suitable selection, recombination and mutation processes are also very important as they form the GA structure and affect convergence to the right results. In spite of a wealth of GA modifications (Kowalczuk & Bialaszewski, 2006), we have implemented classical forms of the procedures of selection, recombination, and mutation (Michalewicz, 1996). Additionally, in order to increase the effectiveness of convergence, though, we did not recombine the five best chromosomes at each selection step (elitism).

After these steps the new population is ready for another evaluation, which is used to determine the distribution of the probability for new selection. The algorithm terminates when the number of generations reaches a certain/given epoch (number). Then the final, sought result is represented by one chromosome characterized by a minimal value of $f(\Delta)$. The chromosome is then divided into 12 binary chains, which are transformed into their decimal values. They represent the computed phenotype, or the optimized corrections, which are then added to the hand-eye frames.

Technical values of the parameters of the genetic algorithm have been as follows:

- generation epoch (number of populations): 300
- population of chromosomes: 40
- recombination probability: 0.5
- mutation probability: 0.05
- precision of corrections: 10^{-4}
- interval for corrections of the translation vectors: [-5, +5] mm
- interval for corrections of the Euler angles: [-0.5, +0.5] deg.

Our genetic algorithm might not converge to the desired error bounds of ±1 mm in the first trial. If this is the case, one has to run the algorithm few times with changed or unchanged parameters.

4.2 Automated calibration

Apart from the pose calibration methods (like the one of Tsai and Lenz), there are also structure-from-motion algorithms that can be applied to calibrate the system without any calibration rig (self hand-eye calibration). Andreff *et al.* (2001) have developed one of such effective calibration methods. They allow for a mobile autonomic robot or robot space applications equipped with vision sensors to recalibrate themselves based on only the actual surrounding scene. By performing programmed movements and using 2D image matching algorithms, the translation vector between the origins of the moving Camera CSs can be estimated and used as a further input to the main algorithm. One of such image matching techniques, based on an adaptive least squares correlation, has been proposed by Gruen (1985).

A fully automatic hand-eye calibration algorithm is a great challenge (with or without a calibration rig) because it would certainly simplify and speed up the calibration process. The time factor is of high importance because it happens quite often that the robotic systems with vision guidance have to be recalibrated. The hand-eye relations change easily during the assembly process due to vibrations during a permanent gripping process. The measurement accuracy does then decrease and the assembly line needs to be stopped (declining efficiency). The faster the system is recalibrated, the sooner the assembly line resumes its work.

5. Experimental results

The experimental system consisted of the following hardware and optical components:

- Kuka robot with 6DOF, model KR 60-3
- Siemens camera, model SIMATIC VS723, resolution of 640×480 pixels
- Cognex camera, model In-Sight 1100, resolution of 640×480 pixels
- Fujinon TV lens, focal length of 16 mm
- Pentax TV lens, focal length of 16 mm

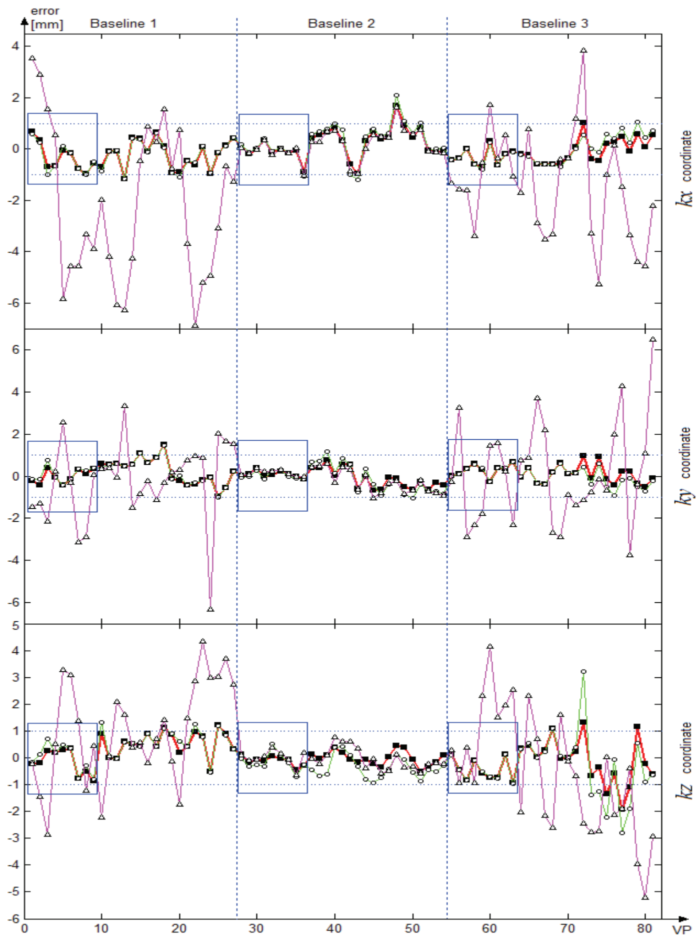
In this research a KUKA robot was utilized which is a six-axis robot with a maximum payload of 60 kg at a maximum range of 2429 mm and a repeatability of ±0.25 mm.

We calibrated the entire stereo system using the classical Tsai and Lenz method and corrected the hand-eye parameters (translation plus rotation based on the Euler angles) with the GA algorithm. We then performed three tests to verify the system's ability to compute the object's pose with repeatable results w.r.t. the static Robot CS based on measured 3 object features. The verification procedure was performed as follows. The robot-stereo system was calibrated for 3 baselines (Baseline 1 was 450 mm, Baseline 2 was 330 mm, and Baseline 3 was 220 mm) and for each one the object was placed at 3 object positions (OPs). The robot was jogged to 10 vantage points (VPs) for each OP. The Euler angles were computed using the NLSF algorithm. As our aim was to create the positioning system with the pose repeatability error of ±1 mm and ±1 deg, the GA was run three times for each camera baseline until it obtained satisfactory results. Each time the computation took about 5 min. The values of the GA parameters listed in Section 4.1 were applied.

Furthermore, we analyzed the influence of the distortion parameters on the system's performance. Three types of verification were used for each baseline. A first method verified the performance of the system with the distortion parameters and with the corrected hand-eye frames of both cameras, a second approach was without the distortion parameters and with the hand-eye corrections, while a third verification was without hand-eye corrections and with the distortion parameters.

As depicted in Fig. 11 after adding the corrections (computed by the GA) to the hand-eye parameters the repeatability error was significantly diminished. Satisfactory results were obtained with and without the distortion parameters. Although distortions seemed not to influence the accuracy for the lens focal length of 16 mm, the authors suggest that they should be included in the camera model when the camera is equipped with lenses of shorter focal lengths.

The figures show that the measuring accuracy of the system without the corrected hand-eye parameters is unsatisfactory for Baselines 1 and 3. The desired accuracy was achieved for Baseline 2, though here the camera-checkerboard transformations computed by MCT had very small errors as compared to those obtained for the other two baselines. Yet, the system with Baseline 2 had the best accuracy because the distances from the Camera 1 to the Object CS were relatively smaller in this configuration. Fig. 12 shows the distances between these two CSs varying from 250 mm to 600 mm, while we set the focus of the cameras at the distance of 400 mm. Although images were blurred at minimal and maximal distances, such deviations proved to be acceptable. Not surprisingly, Baseline 3, the shortest one, produced the worst accuracy.



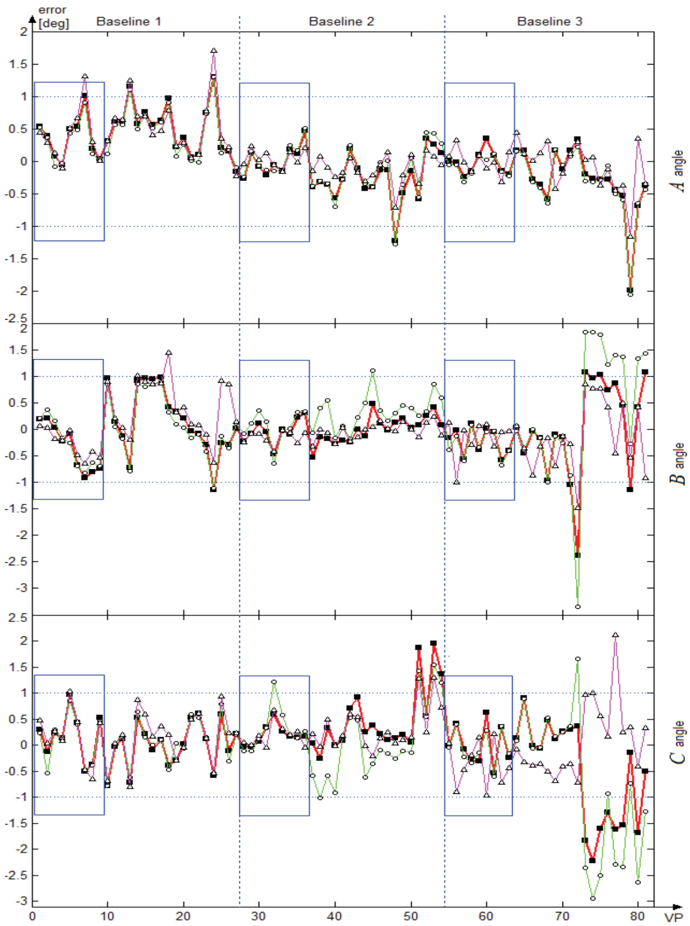


Fig. 11. Repeatability error of the *kx*, *ky*, *kz* coordinates and the *A*, *B*, *C* angles for Baselines 1, 2, 3 : *square* – with the distortion coefficients and with hand-eye corrections; *circle* – without the distortion coefficients and with the hand-eye corrections; *triangle* – with the distortion coefficients and without the hand-eye corrections

The proposed manual calibration of the stereovision system satisfied the criterions of repeatability of measurements. Although there are some errors shown in Fig. 11 that exceed the desired accuracy, it has to be noticed that some pictures were taken at very acute angles. In overall, the camera's yaw angle varied between -50 and +100 deg and the pitch angle varied between -60 and +40 deg throughout the whole test, what far exceeds the real working conditions. Moreover, the image data were collected for GA only at the first OP for each baseline (marked as blue rectangles in the figures) and they were very noisy in several cases. We suppose that noise must have decreased the GA's efficiency in searching for the best solutions, but the evolutionary approach itself allowed preserving stability and robustness of the ultimate robotic system.

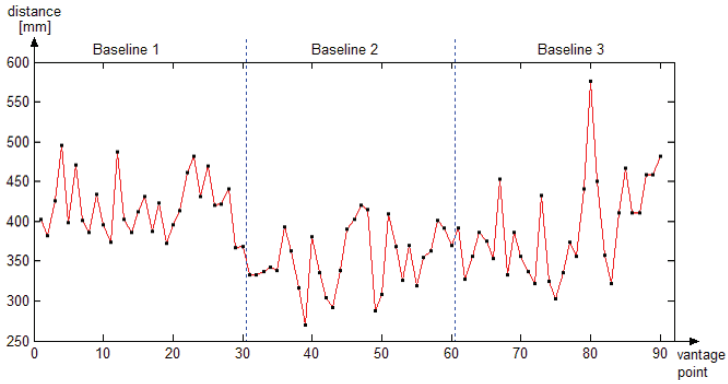


Fig. 12. Distance between the origins of the Object CS and the Camera 1 CS for each VP

6. Conclusion and future work

A manipulator equipped with vision sensors can be 'aware' of the surrounding scene, what admits of performing tasks with higher flexibility and efficiency. In this chapter a robotic system with stereo cameras has been presented the purpose of which was to release humans from handling (picking, moving, *etc.*) non-constrained objects in a three-dimensional space.
In order to utilize image data, a pinhole camera model has been introduced together with a "Plumb Bob" model for lens distortions. A precise description of all parameters has been given. Two conventions (*i.e.* the Euler-angle and the unit quaternion notations) have been presented for describing the orientation matrix of rigid-body transformations that are utilized by leading robot manufacturers. The problem of 3D object pose estimation has been explained based on retrieved information from single and stereo images. Epipolar geometry of stereo camera configurations has been analyzed to explain how it can be used to make image processing more reliable and faster. We have outlined certain pose estimation algorithms to provide the reader with a wide integrated spectrum of methods utilized in robot positioning applications when considering specific constraints (like analytical, or iterative). Moreover, we have also supplied various references to other algorithms. Two methods for a three-dimensional robot positioning system have been developed and bridged with the object pose estimation algorithms. Singularities of the robot positioning systems have been indicated, as well.
A challenging task has been to find a hand-eye transformation of the system, *i.e.* the transformation between a camera and a robot end-effector. We have explained the classic approach by Tsai and Lenz solving this problem and have used a Matlab Calibration Toolbox to perform calibration. We have extended this approach by utilizing a genetic algorithm (GA) in order to improve the system measurement precision in the sense of satisfactory repeatability of positioning the robotic gripper. We have then outlined other calibration algorithms and suggested an automated calibration as a step towards making the entire system autonomous and reliable.
The experimental results obtained have proved that our GA-based calibration method yields the system precision of ±1 mm and ±1 deg, thus satisfying the industrial demands on the accuracy of automated part acquisition. A future research effort should be placed on (•) optimization of the mathematical principles for positioning the robot through some orthogonality constraints of rotation to increase the system's accuracy, (•) development of a

method for computing 3D points using two non-overlapping images (to be utilized for large objects), (•) implementation of a hand-eye calibration method based on the structure-from-motion algorithms, and (•) implementation of algorithms for tracking objects.

7. References

Andreff, N.; Horaud, R. & Espiau, B. (2001). Robot hand-eye calibration using structure-from-motion. *The International Journal of Robotics Research*, vol. 20, no. 3, pp. 228-248

Daniilidis, K. (1998). *Hand-Eye Calibration Using Dual Quaternions*, GRASP Laboratory, University of Pennsylvania, PA (USA)

Fischler, M.A. & Bolles, R.C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, vol. 24, no. 6, pp. 381-395

Gruen, A.W. (1985). Adaptive least squares correlation: A powerful image matching technique. *South African Journal of Photogrammetry, Remote Sensing and Cartography*, vol. 14, no. 3, pp. 175-187

Haralick, R.M.; Joo, H.; Lee, C.; Zhuang, X.; Vaidya, V.G. & Kim, M.B. (1989). Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 6, pp. 1426-1446

Horn, B.K.P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, vol. 4, no.4, pp. 629-642

Kowalczuk, Z. & Bialaszewski, T. (2006) Niching mechanisms in evolutionary computations. *International Journal of Applied Mathematics and Computer Science*, vol. 16, no. 1, pp. 59-84

Kowalczuk, Z. & Wesierski, D. (2007). Three-dimensional robot positioning system with stereo vision guidance. *Proc. 13th IEEE/IFAC Int. Conf. on Methods and Models in Automation and Robotics*, Szczecin (Poland), CD-ROM, pp. 1011-1016

Lu, C.P.; Hager, G.D. & Mjolsness, E. (1998). Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 610-622

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, New York

Phong, T.Q.; Horaud, R.; Yassine, A. & Tao, P.D. (1995). Object pose from 2D to 3D point and line correspondences. *International Journal of Computer Vision*, vol. 15, no. 3, pp. 225-243

Schweighofer, G. & Pinz, A. (2006). Robust pose estimation from planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2024-2030

Szczepanski, W. (1958). Die Lösungsvorschläge für den räumlichen Rückwärtseinschnitt. *Deutsche Geodätische Komission*, Reihe C: Dissertationen-Heft, No. 29, pp. 1-144

Thompson, E. H. (1966). Space resection: Failure cases. *Photogrammetric Record*, vol. X, no. 27, pp. 201-204

Tsai, R. & Lenz, R. (1989). A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, vol. 5,, no. 3, pp. 345-358

Weinstein, D.M. (1998). The analytic 3-D transform for the least-squared fit of three pairs of corresponding points. *Technical Report*, Dept. of Computer Science, University of Utah, UT (USA)

Wrobel, B.P. (1992). Minimum solutions for orientation. *Proc. IEEE Workshop Calibration and Orientation Cameras in Computer Vision*, Washington D.C. (USA)

Automation and Robotics
Edited by Juan Manuel Ramos Arreguin

ISBN 978-3-902613-41-7
Hard cover, 388 pages
Publisher I-Tech Education and Publishing
Published online 01, May, 2008
Published in print edition May, 2008

In this book, a set of relevant, updated and selected papers in the field of automation and robotics are presented. These papers describe projects where topics of artificial intelligence, modeling and simulation process, target tracking algorithms, kinematic constraints of the closed loops, non-linear control, are used in advanced and recent research.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Zdzislaw Kowalczuk and Daniel Wesierski (2008). Vision Guided Robot Gripping Systems, Automation and Robotics, Juan Manuel Ramos Arreguin (Ed.), ISBN: 978-3-902613-41-7, InTech, Available from: http://www.intechopen.com/books/automation_and_robotics/vision_guided_robot_gripping_systems

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.