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:

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

that means that

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

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.

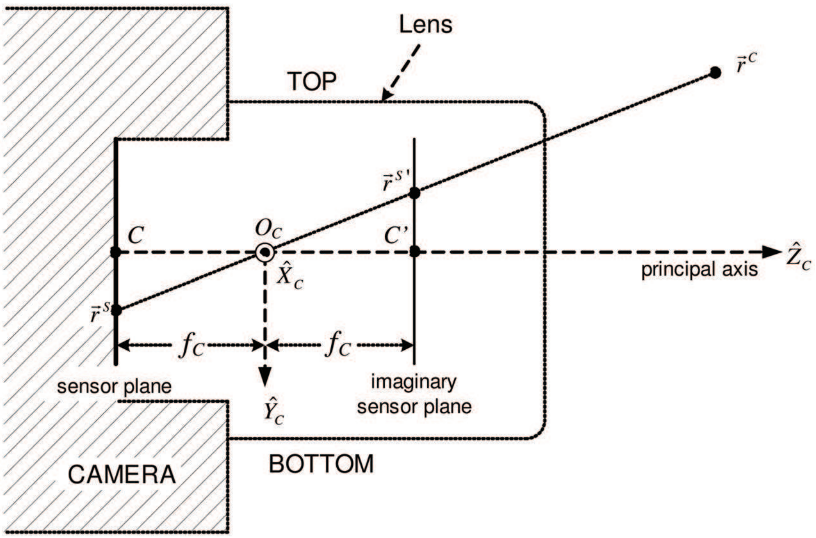

Fig. 1. Right side view of the camera-lens system

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:

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

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.

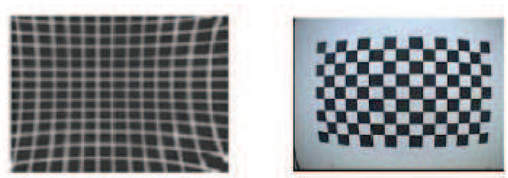

Fig. 2. Distortion: lines forming pincushion (left image) and lines forming a barrel (right image)

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 $h = x_{norm}^2 + y_{norm}^2$, we can include the effect of distortion as follows:

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

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

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

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 $|OF| = const.$

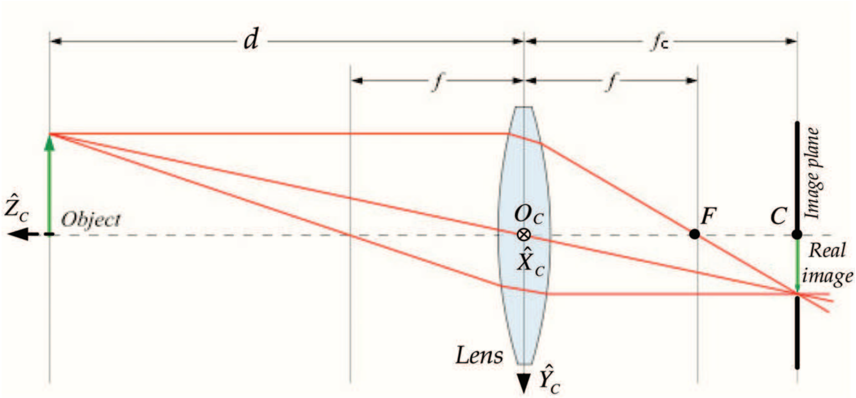

Fig. 3. Left side view of the camera-lens system

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.

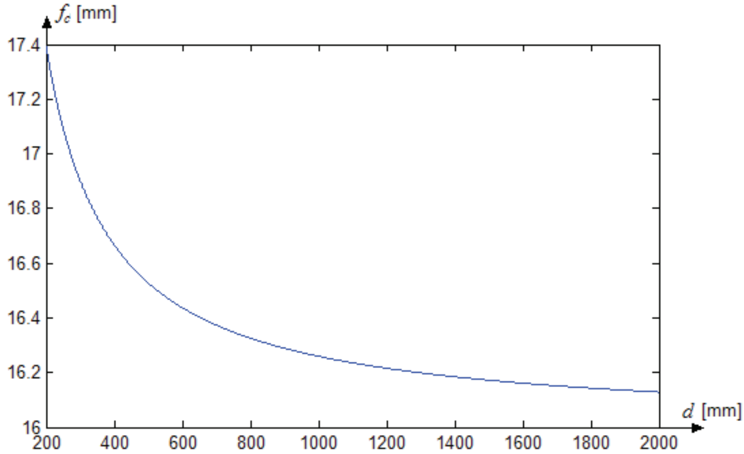

Fig. 4. Camera constant f_c in terms of the distance d

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.

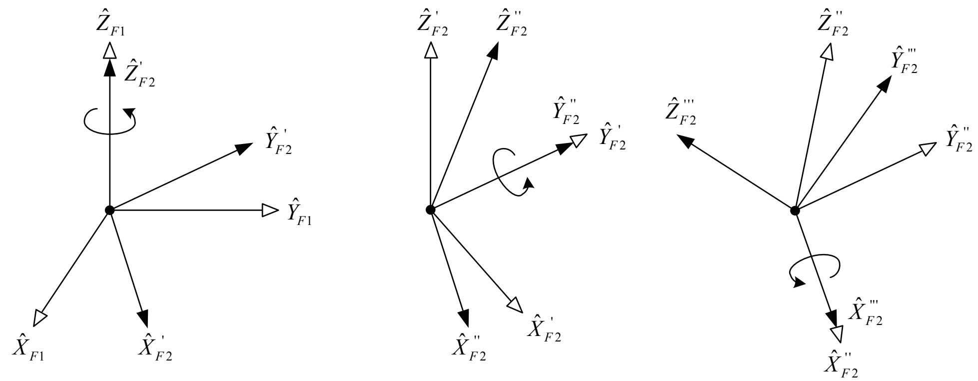

Fig. 5. Representation of the rotations in terms of the ZYX Euler angles

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 {*F*2} CS is moved by a vector $K = [kx_{F1F2} \quad ky_{F1F2} \quad kz_{F1F2}]^{\mathrm{T}}$ w.r.t. the {*F*1} 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:

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

(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 $q = (0, \vec{q})$ and perform the operation with another unit quaternion p

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

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:

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

We can prove this by the following expansion:

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

where

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

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 $\hat{\omega} = [\omega_x \quad \omega_y \quad \omega_z]^{\mathrm{T}}$ can be determined in terms of a unit quaternion as:

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

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.

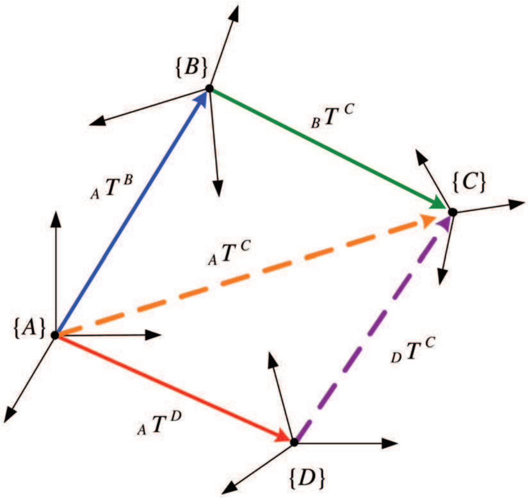

Fig. 6. Transformations based on closed loops

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.

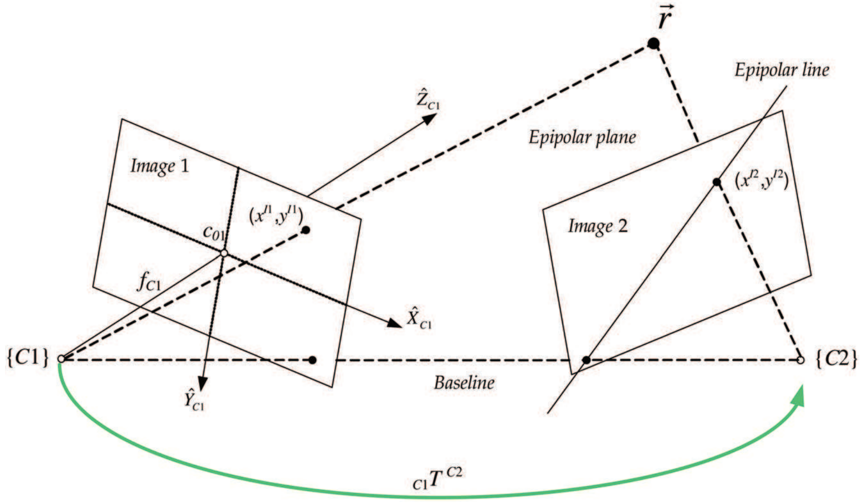

Fig. 7. Stereo–image configuration with epipolar geometry

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