42. Industrial Robotics

Martin Hägele, Klas Nilsson, J. Norberto Pires

Most robots today can trace their origin to early industrial robot designs. Much of the technology that makes robots more human–friendly and adaptable for different applications has emerged from manufacturers of industrial robots. Industrial robots are by far the largest commercial application of robotics technology today. All the important foundations for robot control were initially developed with industrial applications in mind. These applications deserve special attention in order to understand the origin of robotics science and to appreciate many unsolved problems that still prevent the wider use of robots in manufacturing. In this chapter we present a brief history and descriptions of typical industrial robotics applications. We show how robots with different mechanisms fit different applications. Even though robots are well established in large-scale manufacturing, particularly in automobile and related component assembly, there are still many challenging problems to solve. The range of feasible applications could significantly increase if robots were easier to install, to integrate with other manufacturing processes,

and to program, particularly with adaptive sensing and automatic error recovery. We outline some of these remaining challenges for researchers.

Industrial robots are considered as a cornerstone of competitive manufacturing, which aims to combine high productivity, quality, and adaptability at minimal cost. In 2007 more than one million industrial robot installations were reported, with automotive industries as the predominant users with a share of more than 60% [42.1]. However, high-growth industries (in life sciences, electronics, solar cells, food, and logistics) and emerging manufacturing processes (gluing, coating, laser-based processes, precision assembly etc.) will increasingly depend on advanced robot technology. These industries' share of the number of robot installations has been growing steadily.

The production of industrial robots on the one hand, and the planning, integration, and operation of robot workcells on the other hand, are largely independent engineering tasks. In order to be produced in sufficiently large quantities, a robot design should meet the requirements for the widest set of potential applications. As this is difficult to achieve in practice, various classes of robot designs regarding payload capacity, number of robot axes, and workspace volume have emerged for application categories such as assembly, palletizing, painting, welding, machining, and general handling tasks.

Generally, a robot workcell consists of one or more robots with controllers and robot peripherals: grippers

or tools, safety devices, sensors, and material transfer components for moving and presenting parts. Typically, the cost of a complete robot workcell is four times the cost of the robots alone.

A robot workcell is usually the result of customized planning, integration, programming, and configuration, requiring significant engineering expertise. Standardized engineering methods, tools, and best-practice examples have become available to reduce costs and provide more predictable performance [42.2].

Today's industrial robots are mainly the result of the requirements of capital-intensive large-volume manufacturing, mainly defined by the automotive, electronics, and electrical goods industries. Future industrial robots will not be a mere extrapolation of today's designs with respect to features and performance data, but will rather follow new design principles addressing a wider range of application areas and industries. At the same time, new technologies, particularly from the information technol-

ogy (IT) world, will have an increasing impact on the design, performance, and cost of future industrial robots.

International and national standards now help to quantify robot performance and define safety precautions, geometry, and media interfaces. Most robots operate behind secure barriers to keep people at a safe distance [42.3]. Recently, improved safety standards have allowed direct human–robot collaboration, permitting robots and human factory workers to share the same workspace [42.4].

We will first present a historical introduction to industrial robotics with a selection of contemporary application examples, then the basic principles that are used in industrial robotics and a review of programming methods will be presented. We will also discuss tools (end-effectors) and system integration requirements. The chapter will be closed with the presentation of selected, unsolved problems that currently inhibit the wider application of industrial robots.

42.1 A Short History of Industrial Robots

The invention of the industrial robot dates back to 1954 when George Devol filed a patent on a *programmed article transfer* (Fig. 42.1). After teaming up with Joseph Engelberger, the first robot company, Unimation, was founded and put the first robot into service at a General Motors plant in 1961 for extracting parts

from a die-casting machine. Most of the hydraulically actuated *Unimates* were sold through the following years for workpiece handling and for spot-welding of car bodies [42.5]. Both applications were successful, which means that the robots worked reliably and ensured uniform quality. Soon, many other companies
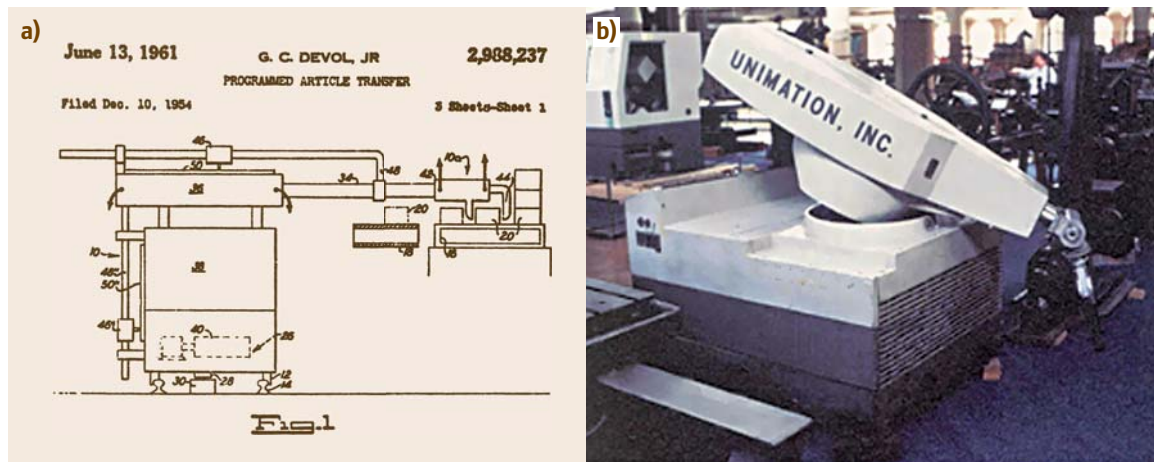

Fig. 42.1a,b The invention of the industrial robot **(a)** This patent was the start of a joint effort of G. Devol and J. Engelberger to form the first robot company, Unimation, a fusion of the words universal and automation. The company was acquired by Westinghouse in the late 1980s. **(b)** The first Unimation performed a rather simple handling task in 1961 at a General Motors plant. Other car manufacturers followed. The photo shows the first robot installed at Ford from their Museum in Dearborn
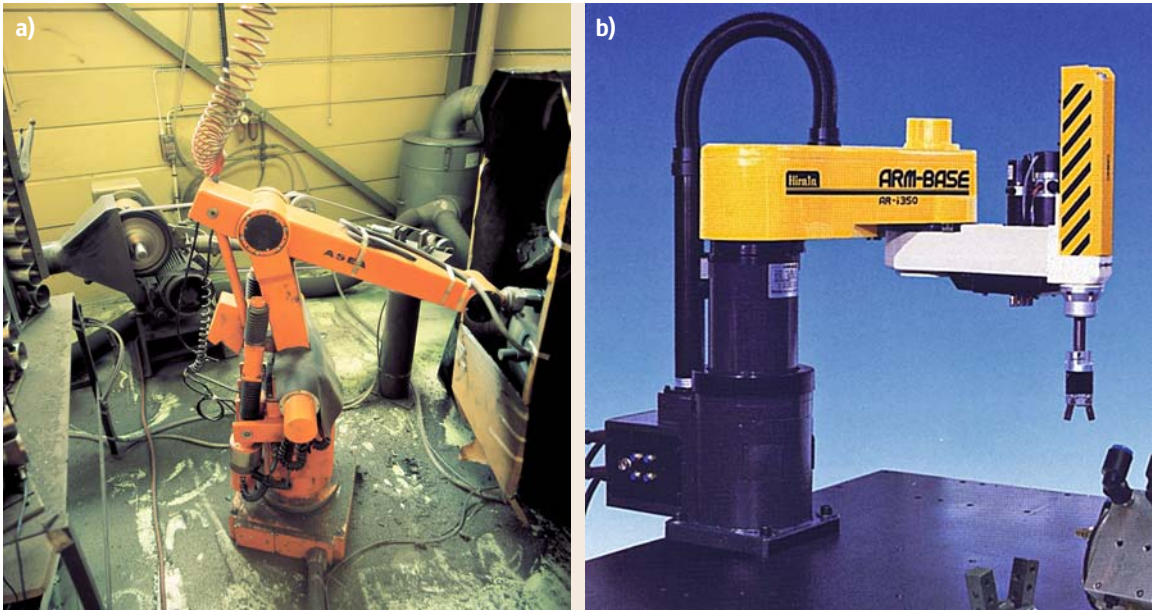
Fig. 42.2a,b The all electric **(a)** IRB-6 and **(b)** a SCARA-type kinematic **(a)** First introduced in 1973, the IRB-6 has been a breakthrough development as it was the first serially-produced robot product, which combined all-electric-drives technology and a microcomputer for programming and motion control. The robot proved very robust. Life-times of more than 20 years in harsh productions were reported (by courtesy ABB Automation, Friedberg) **(b)** The *selective compliance assembly robot arm* (SCARA) is particularly suited for assembly tasks as it combines rigidity in the vertical axis and compliance in the horizontal axis. In 1978, the first Hirata AR-300 was put together. Depicted is the successor design, the AR-i350. The SCARA design combines three or four rotational and one translational axis (by courtesy HIRATA Robotics, Mainz)
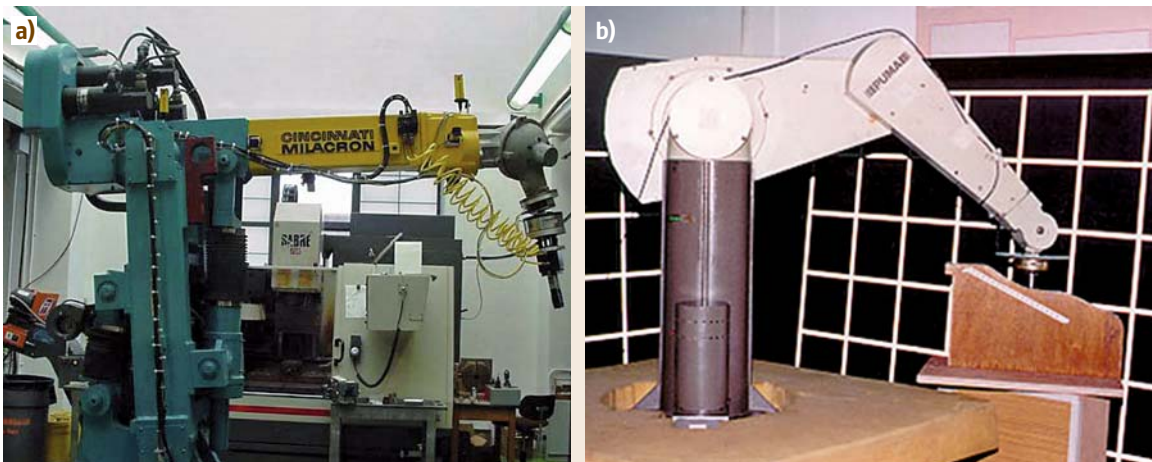

Fig. 42.3a,b Cincinnati Milacron T3 and the Unimation PUMA 560 **(a)** In 1974, Cincinnati Milacron introduced the first microcomputer controlled robot. The first T3 (*The Tomorrow Tool*) models used hydraulic drives, later they were replaced by electric motors as shown in the photo. The CM robotics division was acquired by ABB in the late 1970s **(b)** This 6 axis PUMA (programmable universal machine for assembly) came close to the dexterity of a human arm. After its launch in 1979 by Unimation it became one of the most popular arms and was used as a reference in robotics research for many years
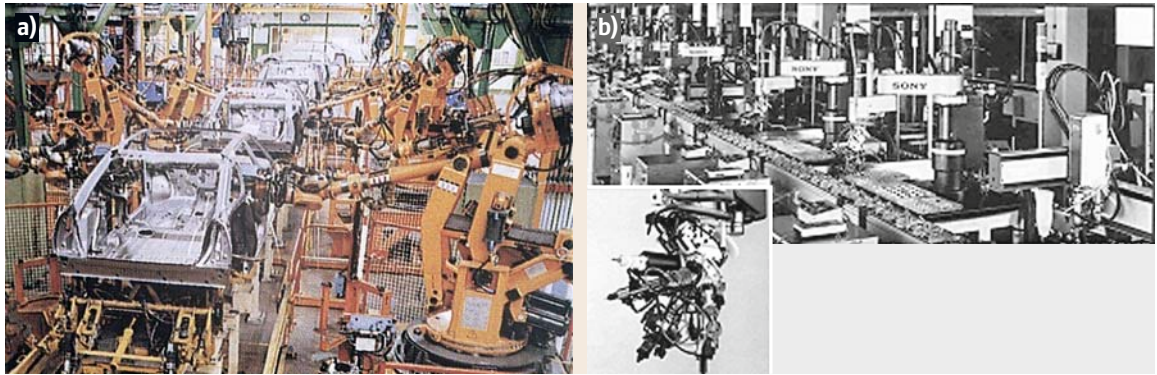
Fig. 42.4a,b Spot-welding line and videocassette recorder (VCR) assembly as pioneering applications for industrial robot applications. **(a)** Spot welding quickly became a primary application for robots as these jobs were particularly exhausting and hazardous for workers. A typical car body welding line from 1985 is displayed. The car model shown is a French Citroën CX **(b)** An automated VCR assembly line (ca. 1989) with SCARAs carrying a turret with multi-gripper tools. Typically five parts are added by one robot before the VCR is moved to the next station of the fully automated assembly line
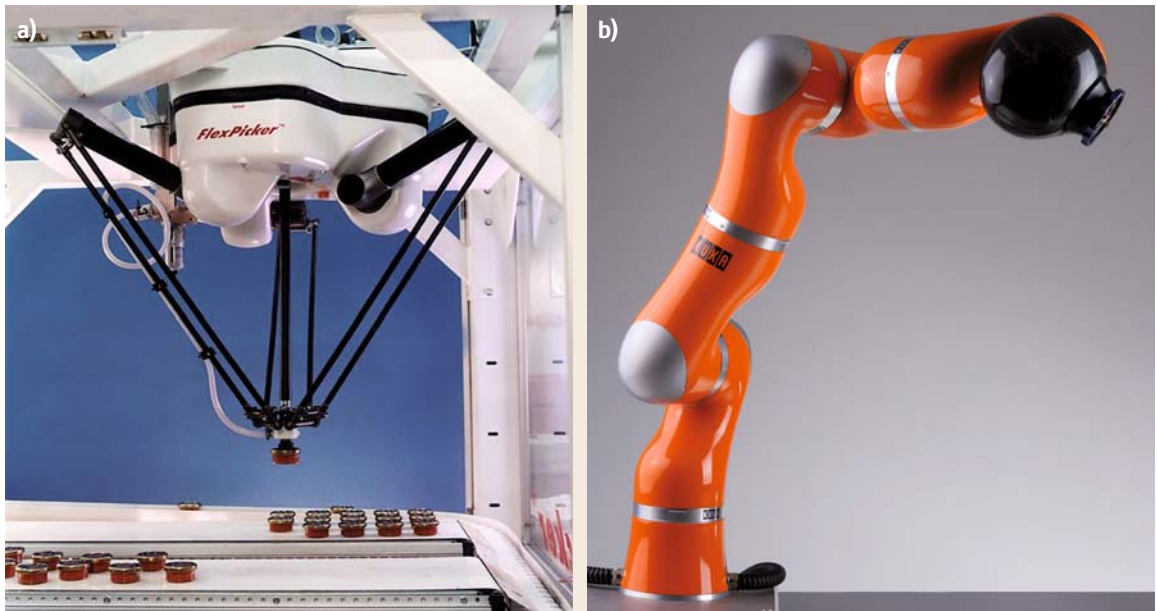

Fig. 42.5a,b The ABB FlexPicker parallel robot for high speed picking and the KUKA light weight robot arm **(a)** *Parallel kinematic machines* (PKM) represent an interesting approach to achieve high stiffness at low inertia, thus allowing accurate, high speed motions. Initially suggested by Clavel this 4 axis robot (called Delta) is used for high speed pick-and-place tasks. The robot reaches accelerations of up to 10 g **(b)** The KUKA light-weight robot is the result of a long research and development process by DLR, towards an arm design with a weight-to-payload ratio of 1:1. The 7 axis arm which is suited for human-robot cooperation imitates the dexterity of a human arm

started to develop and manufacture industrial robots. An innovation-driven industry was born. However, it took many years until this industry became profitable.

The breakthrough Stanford Arm was designed as a research prototype in 1969 by *Victor Scheinman* (see Chap. 3), at that time a mechanical engineering stu-
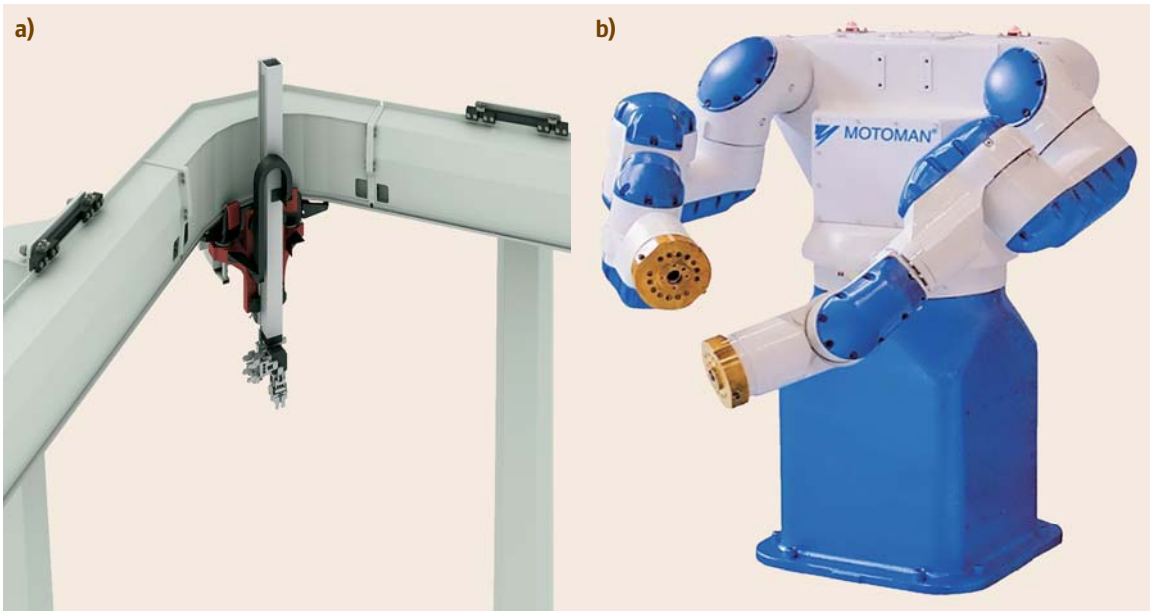
Fig. 42.6a,b Cartesian robot mobility and two-armed robot dexterity **(a)** The roboLoop of Güdel is a curved-track gantry and transfer system. One or more robot arms circulate as carriers in a closed transfer system. The system can be installed suspended, in gantry configuration, or as a floor-standing system. A signal bus allows the control and coordination of multiple robo-carriers (by courtesy Güdel, Langenthal) **(b)** Motoman's DA-20 dual-arm robot provides high-speed motion with two six-axis arms that provide enhanced, human-like flexibility of movement. The robot also provides jig-less operation with one robot arm holding a part while the other performs operations on the held part (by courtesy Motoman, Kalmar)

dent working in the Stanford Artificial Intelligence Laboratory (SAIL) [42.6]. The six-degree-of-freedom (6-DOF) all-electric manipulator was controlled by a standard computer, a Digital Equipment PDP-6. The non-anthropomorphic kinematic configuration with one prismatic and five rotational joints was configured such that the equations for solving the robot kinematics were simple enough to speed up computations. Drives consisted of direct-current (DC) electric motors, harmonic drive and spur gear reducers, potentiometers and tachometers for position and velocity feedback. Subsequent robot designs were strongly influenced by Scheinman's concepts (Figs. 42.2 and 42.3).

In 1973, the company ASEA (now ABB) introduced the first microcomputer-controlled all-electric industrial robot, the IRB-6, which allowed continuous path motion, a precondition for arc-welding or machining (Fig. 42.2). The design proved to be very robust and robot lifetimes of more than 20 years were reported [42.7]. In the 1970s intense diffusion of robots into car manufacturing set in mostly for (spot-)welding and handling applications (Fig. 42.4).

In 1978, the *selective compliance assembly robot arm* (SCARA) was invented by *Hiroshi Makino* of Yamanashi University, Japan [42.8]. The ground-breaking four-axis low-cost design was perfectly suited for small parts assembly as the kinematic configuration allows fast and compliant arm motions (Fig. 42.2). Flexible assembly systems based on the SCARA robot in conjunction with compatible product designs (design for assembly, DFA) have contributed significantly to creating a worldwide boom in high-volume electronics production and consumer products [42.9].

Requirements regarding a robot's speed and weight have led to novel kinematic and transmission designs. From early on, the reduction of the mass and inertia of robot structures was a primary research target, where the human arm with a weight-to-load ratio of 1:1 was considered the ultimate benchmark. In 2006, this goal was reached in the form of the company KUKA lightweight robot, a compact 7-DOF robot arm with advanced force-control capabilities [42.10]. Another approach towards lightweight and stiff structures has been pursued since the 1980s by developing parallel kinematic machines
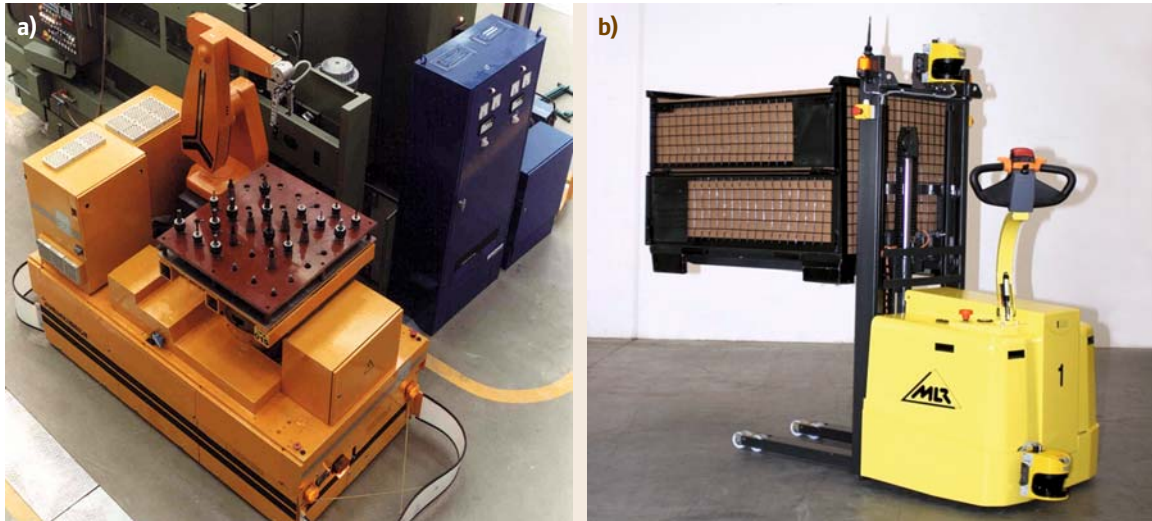
Fig. 42.7a,b A mobile robot arm for machine (un)loading and an autonomous fork lift **(a)** The MORO (mobile robot, 1984), developed at Frauenhofer IPA, Stuttgart, was one of the first prototypes to combine a robot arm on a wire-bound mobile platform which follows a wire buried in the floor. Mobile arms were particularly used in highly automated clean room manufacturing facilities to automatically load and unload process equipment **(b)** This automated fork-lift by MLR combines manual operation with fully autonomous navigation and (un-)loading of boxes and pallets. A laser scanner using retro-reflecting tapes or other landmarks (walls, pillars) avoids constraints imposed by simple buried wire guidance technology (by courtesy MLR System, Ludwigsburg)

which connect the machine's basis with its end-effector by three to six parallel struts [42.11]. These so-called parallel robots (see Chaps. 3 and 12) are particularly suited to achieve the highest speeds (e.g., for picking), precision (e.g., for machining), or handling high work loads (Fig. 42.5). However, workspace volumes tend to be smaller than those of serial or open kinematic chain robots which are comparable in size.

Still, Cartesian robots are ideally suited for covering large workspaces. Apart from the traditional design using three orthogonal translational axes, the company Güdel introduced a curved-track gantry in 1998. The concept allows one or more robot arms to track curves and to circulate in a closed transfer system. Thus, the workspace of robot can be immensely increased at high speed and precision. This may be particularly valuable in logistics or machine tending [42.12].

Two-handed dexterous manipulation can be critical for complex assembly tasks, simultaneous handling and processing of workpieces, or the handling of large objects. The first commercial robot for synchronized, two-handed manipulation was introduced by MOTOMAN in 2005 [42.13]. As a dual-arm robot imi-

tates the reach and dexterity of human arms, it can be placed on a site that previously accommodated human workers. Thus, capital expenditure can be reduced. It features 13 axes of motion: six axes per arm, plus a single axis for the base rotation (Fig. 42.6).

In parallel to industrial robots *automated guided vehicles* (AGVs) have emerged. These mobile robots are used for moving workpieces or loading equipment from point to point. Within the concept of automated *flexible manufacturing systems* (FMS) AGVs have become an important part of their routing flexibility. Initially AGVs relied on prepared floors such as embedded wires or magnets for motion guidance. Meanwhile, freely navigating AGVs are used in large-scale manufacturing and logistics. Usually, their navigation is based on laser scanners that provide an accurate two-dimensional map of the actual environment for self-localization and obstacle avoidance [42.14]. Early on, combinations of AGVs and robot arms were realized to automatically load and unload machine tools (Fig. 42.7). Only in some selected environments such as (un-)loading process equipment in the semiconductor industry these mobile arms were economically advantageous.

By 2007, the evolution of industrial robots was marked by the following main trends [42.15]:

- The average robot unit price fell to about one-third of its equivalent price in 1990, which meant that automation is becoming more affordable. At the same time, robot performance parameters such as speed, load capacity, and mean time between failures (MTBF) have dramatically improved. These improvements provide a faster return on investment, particularly for small, short-run batch production.
- Off-the-shelf components from personal computer (PC) technologies, consumer software, and the IT industry have contributed to improved performance-to-cost ratios. Today, most manufacturers integrate PC-based processors in their controllers as well as software for programming, communication, simulation, and maintenance from high-volume IT-markets.

- Multiple robots can be programmed and synchronized in real time by one controller, which allows robots to cooperate precisely on a single workpiece.
- Increasingly, vision systems for object identification, localization, and quality control have become an integral part of the robot controller.
- Robots are networked by fieldbuses or ethernet for control, configuration, and maintenance.
- New financing arrangements allow end-users to rent a robot or even have a robot workcell operated by a specialized company or even the robot supplier in order to reduce risks or to save on investment capital.
- Training and education have become important services to end-users to increase acceptance of robot technology. Specific multimedia material and courses aim at educating industrial engineers and workforce to effectively plan, program, operate, and maintain industrial robot workcells.

42.2 Typical Applications and Robot Configurations

42.2.1 Welding

Welding ranks among the most important joining processes in manufacturing. Manual welding requires highly skilled workers, as small imperfections in the weld can lead to severe consequences. Why is a robot suited to perform this critical job? Modern welding robots have the following characteristics:

- Computer control allows the programming of task sequences, robot motions, external actuators, sensors, and communication with external devices.
- Free definition and parameterization of robot positions/orientations, reference frames, and trajectories.
- High repeatability and positioning accuracy of trajectories. Typically repeatability is some ± 0.1 mm and positioning accuracy is of the order of ± 1.0 mm.
- High speeds of the end-effector of up to 8 m/s.
- Typically, articulated robots have six DOF so that commanded orientations and positions in their workspace can be reached. Workspace extensions by mounting the robot on a linear axis (seventh DOF) are common, especially for welding of large structures.
- Typical payloads of 6–100 kg.
- Advanced programmable logic controller (PLC) capabilities such as fast input/output control and synchronizing actions within the robot workcell.

- Interfacing to high-level factory control through fieldbuses or ethernet.

Metal inert/active gas (MIG/MAG) welding is the predominant application of industrial robotics today. The automatic arc-welding process is based on a consumable wire electrode and a shielding gas that are fed through a welding gun (Fig. 42.8). Electric current sources, torches, and peripheral devices for automatic cleaning and maintaining the torch (anti-splatter, wire-cutting, tool changer etc.) are offered by specialized companies. Often sensors are used to track welding gaps and measure weld seams either before or synchronously to the welding process, thus adapting the robot's trajectory in the presence of workpiece variation and distortion. Also, cooperating robots have been introduced where one robot fixes and moves the workpiece in synchronization with another robot carrying a welding tool so that the weld can be performed with the pool of molten metal horizontal.

Another interesting robot task is multilayer welding. In regular intervals the robot measures the profile of the weld gap and adaptively generates subsequent tool paths to apply successive weld seams until the final required geometry is reached. In the example shown in Fig. 42.8c up to 70 seams can be produced by the robot to weld thick metal plates.
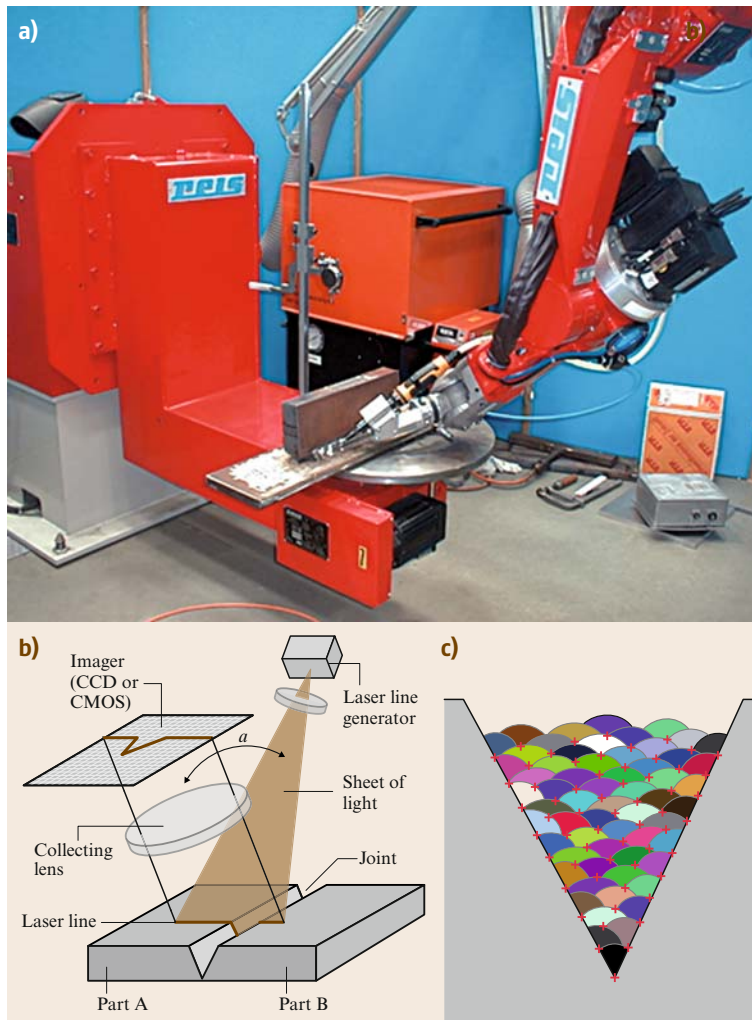
Fig. 42.8a–c Robot welding with sensor guidance. (**a**) The welding robot workcell consists of a 6 DOF robot, a turning table so that the seam is accessible and welded in a flat position, the weld source with a fume extractor and the welding torch. Safety fences and light shields are not shown. The welding torch is attached to the robot flange. A service station (not shown) in the robot workcell is approached regularly to clean the gas nozzle. (**b**) A typical weld seam sensor is based on the so-called laser triangulation principle. The sensor is attached to the robot and a laser projects a sharp stripe on the seam. The stripe is detected by an imager, typically a 2-D CCD camera. From the extracted contour the position of the seam relative to the sensor is calculated. (**c**) Multiple seams can be generated based on this information

42.2.2 Car Body Assembly

Early on, car body assembly became the predominant robot application as the benefit for robot automation was apparent. Handling and positioning the metal sheets, spot welding, and transport of the body frames was either hazardous, physically demanding to the worker, or difficult to realize on fixed automation lines given the desired variety of car body configurations to be assembled on the one production line (Fig. 42.9). In the stamping section metal sheets are cut into plates (or *blanks*) ready for pressing into body panels. In subsequent steps robots load the panels onto a tray that fixes the panels for other robots to be spot-welded. After inspection the finished bodies are transferred by conveyor to the paint shop.

Individual assembly units and components such as motors, transmissions, axles, doors or fenders are pre-mounted in separate areas. The car body is delivered to the body assembly area *just in sequence*, i. e. at the right time and place on the assembly line. The climax of the assembly process(wedding) – when the engine, drive and chassis first meet takes place before the last parts are monted on the car.

Today's industrial robots, particularly in the work-load category of $100–300$ kg, are to a large extent the results of the requirements stemming from this application:

- Required repeatability of at least ± 0.5 mm under typical loads of some 100 kg for the spot-welding
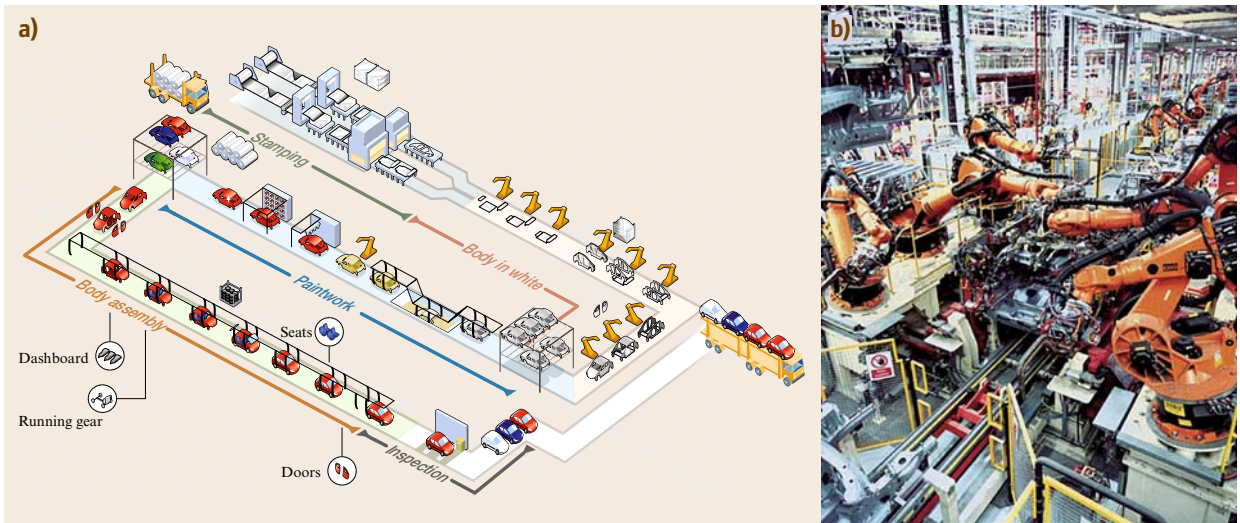
Fig. 42.9a,b Car body assembly. **(a)** A car body assembly usually follows the illustrated steps: Stamping of the metal sheet into plates, fixing and alignment of the plates on trays, spot welding, painting the car body, and final assembly of the car body (doors, dashboard, windscreens, power-train seats, and tires). Car factories can host well over 1000 robots working two to three shifts per day. (Courtesy PSA Peugeot Citroën, Paris and Art Movies, Paris) **(b)** The Mercedes A class assembly in Rastatt Germany is highly automated. The picture shows spot welding robots along the, *body in white* transfer line. Trays carrying car bodies pass through the *robot garden*

tool and cable package leads to stiff and heavy arm structures. A typical robot weight-to-payload ratio is of the order of 1500 kg:150 kg.

- Three-shift continuous operation requires highest reliability of both robot and equipment. Typical mean time between failure (MTBF) is reported to be around than 50 000 h [42.1].
- The line capacity depends on the robot's speed to place spot welds. Thus, the *point-to-point* (PTP) movement time between welding positions should be kept as short as possible. This is mainly achieved by powerful drives so that the mean power consumption of the robot drives is typically 5 kW.
- Most of the trajectories, positions, and orientations are generated using *offline programming* (OLP) systems. Accurate simulations of the robot motion depend on robot models, which incorporate both robot kinematic properties and controller characteristics. The *realistic robot simulation* (RSS) interface is the accepted standard format for robot models of offline programming systems [42.16].

42.2.3 Painting

Hazardous working conditions for human operators motivated Trallfa, a Norwegian company, to develop simple

spray-painting robots in 1969, particularly for spraying bumpers and other plastic parts in the automotive industry. Initially pneumatically driven for antiexplosion reasons, today's robot designs are fully electric with greatly improved spray guns. They also have hooks and grippers to open hoods and doors during the painting task. Hollow wrists that house gas and paint hoses allow fast and agile motions. Spray guns for robots have evolved dramatically for delivering uniform quality using as little paint and solvent as possible and also for switching between different paint colors. Originally spray-painting robots replicated movements copied from human workers. Most of the programming for robot painting today is done offline as state-of-the-art programming systems offer integrated process simulations to optimize paint deposition, thickness, and coverage (Fig. 42.10).

42.2.4 Material Transfer Automation

Generally, industrial practice in robot workcell planning aims at finding a compromise between reducing the variation of the workpiece position and the cost of sensor systems to compensate for residual variation. Nearly all parts arrive at robot workcells in a repeatable manner, either being stored in special magazines, or by being
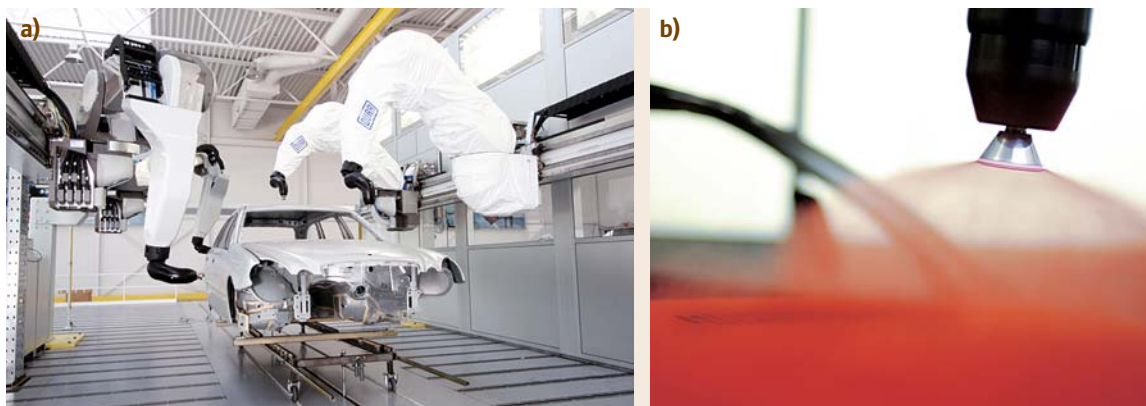
Fig. 42.10a,b Painting robots. (**a**) A multi-robot workcell for car body painting (**b**) High-speed rotating atomizer and charge. Painting robots. (**a**) are used for surface coating of car bodies and other parts. All current paint materials such as solvent-based paints, water-borne paints and powder can be applied. In car production, multiple robots work in parallel for optimal throughput and accessibility of the car body. Most of the programming which includes the synchronisation of the robots is performed offline. Paint guns are critical to the process quality. Figure (**b**) shows a Dürr EcoBell2 paint gun which atomizes the paint material at the edge of the rotating bell disk by centrifugal forces (by courtesy Dürr, Stuttgart)
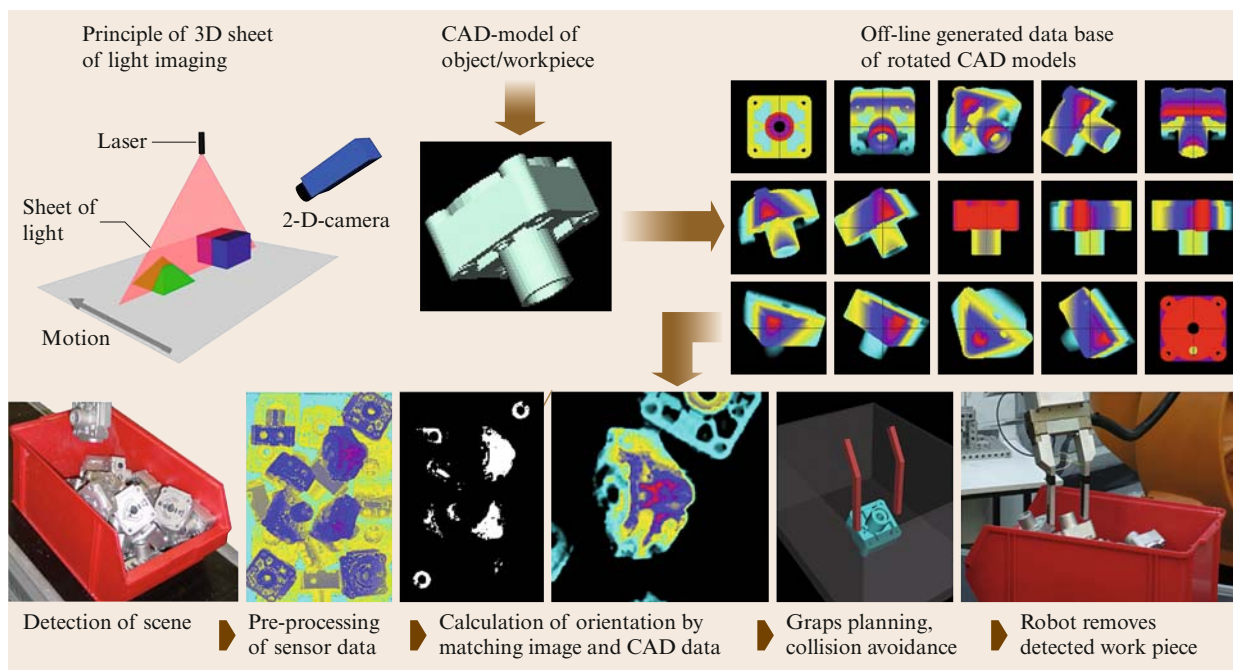

Fig. 42.11 Robot bin picking. A scene containing the objects is acquired by a 3-D sensor e.g. based on the laser triangulation principle. Beforehand this approach turns the CAD object virtually in discrete spatial angles in an off-line calculation. Feature histograms for each view are generated and stored in a database. A best match between actual feature histograms and the simulated sets of histograms determines the location of the object. A grasp has to be selected and a collision-free trajectory is generated. A typical cycle time of a location process is between 1 and 2 s

transported by vibrating devices that allow the parts to settle into a predictable orientation.

If randomly oriented in a carrier or a box, parts have to be properly located so that the robot can grasp them appropriately. This challenge in industrial robotics has been referred to as *bin-picking* and has been investigated by numerous researchers since the mid-1980s [42.17]. Even though an abundance of approaches has been presented a cost-effective standard solution has not been established yet [42.18].

Barely 10% of industrial robot installations in 2006 were sensor-equipped. However, it is expected that many future robots will have standard embedded force-torque and vision sensors. This is a precondition for advanced vision for object identification and localization in manufacturing.

One method for determining random object locations using the object's computer-aided design (CAD) data starts with point clouds of a scene acquired by a three-dimensional (3-D) sensor (Fig. 42.11). A selection of 3-D sensors is described in Chap. 22. The CAD model has been virtually turned into discrete spatial angles in an offline calculation. Feature histograms for each view are generated and stored in a database with
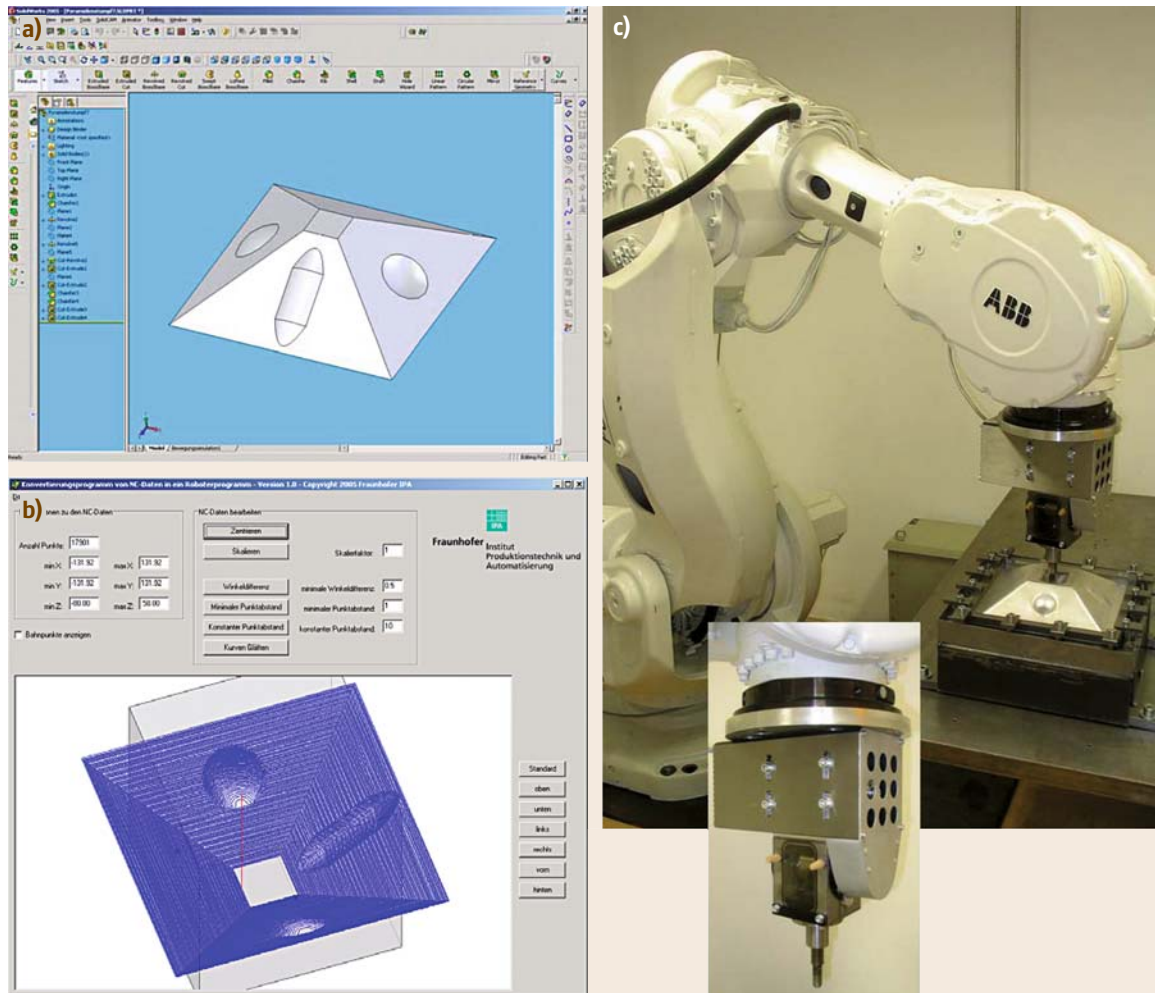

Fig. 42.12a–c Trajectory generation for incremental machining processes. In this example, the forming process of metal sheets is based on an oscillating stamp (amplitude 1 mm, 50 Hz frequency) which locally plastifies the metal in incremental steps. From the CAD model (**a**), the robot's trajectories are calculated offline on the basis of specific material models (**b**). Each line represents a part of the tool trajectory. (**c**) Shows a demonstrator robot workcell with the tool in detail

reduced end-effector speed (of some 50 mm/s). Thus, the sensor scales the applied force/torque information into a compliant robot motion. Obviously the physical human–robot cooperation has to obey safety precautions as the robot's and worker's workspaces overlap.

The ISO 10218-1:2006 standard specifies requirements and guidelines for the inherent safe design, protective measures, and information for use of industrial robots. It describes basic hazards associated with robots, and provides requirements to eliminate or adequately reduce the risks associated with these hazards. A novel element of this revised standard is the regulation of so-called collaborative operation, where the robot works in direct cooperation with a human within a defined workspace. Basically the collaborative operation depends on several criteria, which have to be met by the robot workcell [42.21]:

1. The hand-guiding equipment shall be located in the area of the end-effector.
2. The robot moves with safe reduced speed (less than 250 mm/s) and safe monitored position. This position monitoring shall be according to at least category 3 of ISO 13849, unless a risk assessment is performed and determines that a higher category is required [42.22, 23].
3. The robot must sense and keep a safe distance from the human. The distance relates to the attended

speed. The distance and speed monitoring shall be according to at least category 3 of ISO 13849, unless a risk assessment is performed and determines that another category is required.

If a robot's total power consumption can safely be limited to 80 W as well as its static impact forces to 150 N no additional sensor-based safety precautions are needed. Again, these conditions have to be secured by a risk assessment or security systems which comply with at least category 3 of ISO 13849 [42.21]. Currently, novel control algorithms, kinematic and actuator designs are being investigated to realize so-called intrinsically safe robots [42.24].

The described workcell complies with the ISO standard in such a way that the presence of the human is detected by activating both safety switches of the handle (first condition). The robot's built-in safe controller safely measures the end-effector's position and limits its velocity [42.25] (second condition). Meanwhile most of the robot manufactures provide safe category 3 controllers. If the worker's presence in the workspace is not known (third condition, not applicable here) a safe sensor system has to detect safely the workers location according to category 3. Three-dimensional sensors meeting category 3 first appeared on the market in 2006, thus opening up a wide field of potential collaborative operations [42.26].
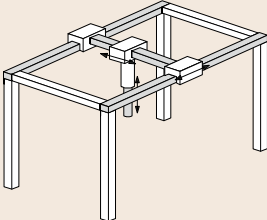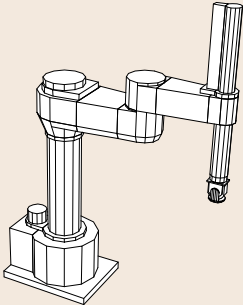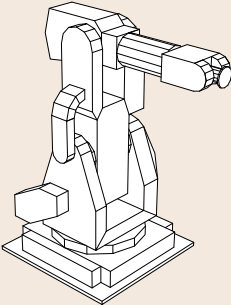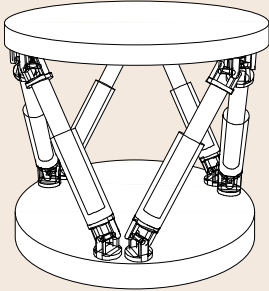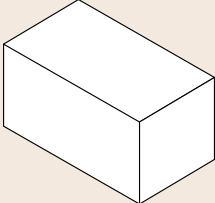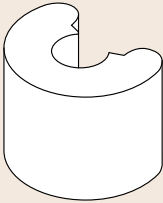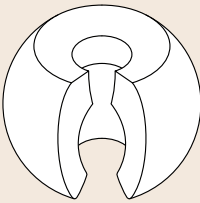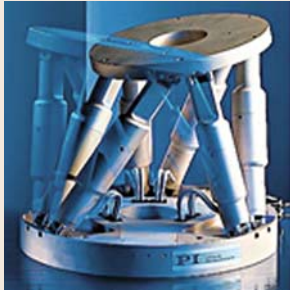
42.3 Kinematics and Mechanisms

The choice of mechanism, its kinematic properties, the computation methods used to determine joint motions, and the intended application of a robot manipulator are all closely related. The diagrams in Table 42.1 show several common types of robot mechanism.

With advances in the state of the art in kinematic algorithms and computer hardware processing capabilities, computation is much less of a constraint on mechanism choice than it was for early robot designers. The choice of mechanical structure of the robot depends mostly on fundamental mechanical requirements such as payload and workspace size. Considering a given level of cost, there is usually a tradeoff between workspace size and stiffness. To enable the robot to reach inside or around obstacles it is clearly advantageous to use an articulated mechanical design.

Considering also the stiffness and accuracy (in a practical sense considering what is reasonable to build), the picture is more complex. Each of the first three types in Table 42.1 we refer to as *serial kinematic machines* (SKMs), while the last is a *parallel kinematic machine* (PKM). To obtain maximum stiffness, again for a certain minimum level of cost, the end-effector is better supported from different directions, and here the PKM has significant advantages. On the other hand, if high stiffness (but not low weight and high dexterity) is the main concern, a typical computerized numerical control (CNC) machine (e.g., for milling) is identical in principle to the gantry mechanism. There are also modular systems with servo-controlled actuators that can be used to build both robots with purpose-designed mechanisms.

Table 42.1 Mechanical types of motions with different kinematic: Gantry is what a Cartesian coordinate robot is typically called, the SCARA is a *selective compliant articulated robot* (or *selective compliance assembly robot arm*, since the main application is planar assembly), the *Articulated* (all joints are rotary and placed in series after each other) robot is often referred to as an arm robot, and the *Parallel* is shown with prismatic joints, but can also have revolute joints such as the Delta robot (Fig. 42.5). Combinations are common

Gantry	SCARA	Serial articulated	Parallel
Form of workspace			
Product example			

42.4 Task Descriptions – Teaching and Programming

We cannot instruct a robot in the same way that one would instruct a human worker how to carry out a task. With skilled workers knowing the applications, devices, processes, and the general requirements on the product to be manufactured, we would only need to summarize *what* needs to be done.

In practice, since it is difficult to encode much of the required background knowledge, we aim for programming principles that resemble instructing a (totally) unskilled worker, telling precisely *how* every aspect of the task is to be performed. That requires a much more explicit way of instructing the robot, but one which is

still human-friendly. More specifically, as a wish-list that we will come back to at the end of the chapter, we would like to teach robots by

- manually guiding the robot to the positions of interest, or even along the desired paths or trajectories if human accuracy is enough
- having simple ways to make use of CAD data whenever available
- using different complementary modalities (paths of communication between the human and the robot), such as speech and gestures
- choreographing the task movements, for instance loops and conditions, without requiring extensive programming competencies
- means of describing acceptable variation, e.g., as expected or normal deviations from the nominal path
- specification of how external sensing should be used for new types of motions or for handling unknown variations

Initially, mainly during the 1970s and 1980s, there were some painting robots that could be programmed by manual guidance. This was possible due to the following abilities

- Applications such as painting permit the use of a lightweight arm, including the end-effector, possibly balanced with respect to gravity if needed.
- The accuracy requirements were (compared to today) modest so it was possible to use back-drivable drive trains and actuators, and the definition of the motions could then be done manually by the operator moving the end-effector along the path. The recorded poses, including the timing/speed information, then define the programmed motion.
- Since no inverse kinematics was needed during programming or real-time operation, it was not a problem from a computing point of view to use arm kinematics without singularities in the workspace.
- The requirements of optimality of painting motions were also modest, compared to recent years when environmental conditions (for nature and workers) calls for minimal use of paint.

It is often referred to as an inevitable problem that there are singularities to be handled within the workspace. However, to simplify the kinematics and its inverse from a software point of view (e.g., during the 1980s considering the power of the microprocessors and algorithms available at that time), robots were actually designed to have simple (to compute) inverse kinematics. For instance, the wrist orientation was decoupled

from the translation by the arm by using wrist axes that intersected with the arm axes. The resulting singularities within the workspace could be managed by restrictions on wrist orientations, but an unfortunate implication was that robot arms were no longer back-drivable (close to the singularities) when designs were (due to engineering and repeatability requirements) adopted to standard industrial controllers. Then with the development of microprocessor-based industrial controllers and the definition of motions based on jogging (manual moves by for instance using a joystick) and CAD data, still lacking efficient and robust techniques for human-like instructions (speech, gestures, etc.), the means of robot programming became closer to computer programming (extended with motion primitives).

Robot programming languages and environments have traditionally been separated into online programming (using the actual robot in situ) and offline programming (using software tools without occupying the robot). With the increasing power of offline programming tools, their emerging ability to connect to the physical robot, and the increasing level of software functions that are embedded into the robot control system, online programming is now unusual, except to verify and manually adjust programs generated offline. Still, of course, it is economically important to minimize downtime for robot programming, and advanced sensor-based applications may be too hard to develop without access to the true dynamics of the physical workcell for fine-tuning. Nevertheless, robot languages and software tools must provide for both methods of programming.

Even though robot languages from different manufacturers look similar, there are many semantic differences that have to do with both the meaning when programs are running (the robot performing its operations) and the way the robot is instructed. The need to ensure that existing robot programs can operate with replacement robots and controllers, and also to make use of existing knowledge in robot programmers and incorporated into offline programming software, requires manufacturers to continue to support their original proprietary languages. Features such as backward execution (at least of motion statements) and interactive editing during interpretation by the robot (in combination with restrictions to make the programming simpler) also make robot languages different from conventional computer programming languages.

During the last decade and in current developments, the trend has been to focus on the tool (robot end-effector) and on the process knowledge needed for the manufacturing process, and to let the operator express

Table 42.2 Example: Simple pick-and-place operation with a typical industrial robot controller

ABB Rapid (a proprietary language) program: positions obtained from a charge-coupled device (CCD) camera; end-effector tool is pneumatic. MoveJ denotes joint-space motion and MoveL denotes linear/Cartesian motion. Basically there are four arguments for such motions: target position (here relative, by use of the offset function), the maximum motion speed (here using predefined v-constants with the number specifying the desired speed in mm/s along the path), the desired accuracy before continuing (fine referring to no corner blending at all), and the tool being used (including frames, inertia, etc., here in the tool0 data record).

PROC cam_pick()	
MoveJ app_point, v1500, z25, tool0;	// Preposition near the working table (using a 25 mm tolerance)
MoveJ Offs(camera11,x,y,-30), v1500, fine, tool0;	// Approach 30 mm above the object object, position x, y from webcam
temp1:=CRobT();	// Acquire current position
MoveL Offs(temp1,0,0,-38), v400, fine, tool0;	// Move to object: account for suction cup flexibility (8 mm)
Set DO08;	// Vacuum ON, pick object
WaitTime 1.5;	// Wait 1.5 s
MoveL Offs(temp1,0,0,10), v400, fine, tool0;	// Move up 10 mm
MoveJ app_point, v1500, z25, tool0;	// Move "fly-by" to box with 25 mm tolerance
MoveJ place_box, v1500, fine, tool0;	// Move to box position
Reset DO08;	// Vacuum OFF, release object
Set DO07;	// Blow ON
WaitTime 0.8;	// Wait 0.8 s
Reset DO07;	// Blow OFF
ENDPROC	

Table 42.3 Robot-related centricity, ranging from a high-level view of the work to be accomplished in terms of the product to manufacture, to a low-level robot motion view that, in practice, constraints what manufacturing operations can be performed

User application → / *Robot constraint* ↓

Product: Description of the final shape and assemblies of the workpieces, in terms of that product; the robot system plans the operations.

Process: Based on known sequences of specific manufacturing operations, each of these is specified in terms of their processes parameters.

Tool: The motion of the robot-held tool is specified in terms of programs or manual guidance; the user knows the process it accomplishes.

Arm: The robot arm and its end-plate for tool mounting is programmed how to move in Cartesian space; the user knows the tool.

Joint: For each specified location the joint angles are specified, so straight-line motions are difficult; the robot provides coordinated servo control.

the robot task in such terms. This development results in a need for an increased level of abstraction to simplify the programming, reflecting the fact that the so-called robot programmer knows the production process very well, but has quite limited programming skills. To understand why such a high level of abstraction has not come into widespread usage, we may compare this with the early days of industrial robotics when there was no kinematics software built into the controllers, and hence the robots were programmed via joint-space motions. (Kinematics here deals with the relation between robot motors/joints and the end-effector motions.)

Built-in inverse kinematics permits tool motion to be specified in Cartesian coordinates, which is clearly

a great simplification in many applications. That is, the robot user could focus more on the work to be carried out by the robot and less on the robot itself. However, robot properties such as joint limits cannot be neglected. Until the beginning of the 1990s, robots did not follow programmed trajectories very well at high speeds or accelerations, and full accuracy could only be achieved at low speed. Modern robot systems with high-performance model-based motion control perform their tasks with much greater accuracy at high speed due to model based control features, see Chap. 6 [42.27, 28].

There are increasing opportunities to raise the level of abstraction to simplify the use of robots even more by encoding more knowledge about the robot, tools, the process, and workcells into control systems.

Example 42.1:
A machine part consisting of plates and pipes of aluminium is to be manufactured, and there is robot equipment available for production of this (and other) types of workpieces.

- A *product*-centric system would generate configurations and robot programs, and instruct the operator for whatever manual assistance that might be needed (e.g., clamping and fixturing). The system would determine the welding data such as the type of welding and how many passes for each seam.
- A *process*-centric system would instead accept input in terms of the welding parameters to use, including the order of the welds. The system would select input signals to the process equipments (such as what output voltage the robot controller should set such that the welding will be done with a certain current), and robot programs specifying the motions of the weld pistol would be generated.
- A *tool*-centric way of programming would require the operator to set up the process manually by configuring the equipment in terms of their native settings, and appropriate tool data would be configured such that the robot controller can accomplish the programmed motions, which are specified by giving coordinates and motions data referring to the end-effector.
- An *arm*-centric system is similar to the tool-centric approach in that Cartesian and straight-line motions can be (and need to be) explicitly specified/programmed by the programmer, but extra work is needed since the robot does not support a general tool frame.
- A *joint*-centric style would be needed if one of the very early robot systems is being used, requiring

a straight line to be programmed by lots of joint-space poses close to each other.

Thus, the question is, are you programming robot joint servos to make the robot provide a service for you, are you programming/commanding an arm how to move a tool, is it the tool that is made programmable by means of the robot, is it manufacturing services that are ordered by specifying the desired process parameters, or is it an intelligent system that simply can produce your product?

As a final goal related to the product-centric view, so-called task-level programming is desirable. This has been a goal since the 1980s, and implicitly also since the very beginning of robotics. It would mean that the user simply tells the robot what should be done and the robot would know how to do it, but it would require extensive knowledge about the environment and so-called machine intelligence. The need for extensive modeling of the environment of the robot is well known. Sensing of the environment is costly, but in an industrial environment it should only be needed occasionally. Modeling has to encompass full component dynamics and the limitations of the manufacturing process. With these difficulties, task-level programming is not yet achievable in practice.

As indicated in the application examples, it is now common practice to generate robot programs from geometric data in CAD files. That is, the CAD application could be the environment used for specifying how the robot should perform the required operations on the specified parts. This is not quite task-level programming since human operators do the overall planning. CAD software packages are powerful 3-D tools and are now very common among manufacturing companies. Consequently, using those tools for robot programming is desirable since the operator may start the offline programming of the necessary manufacturing operations using the 3-D model of the product. There are two possible approaches:

1. Use the CAD interface to parameterize predefined robot programs, tailoring their behavior by using geometric information [42.16, 29]. This means defining motions, adding process parameters, deciding on what to do with data from devices, etc.
2. Generate the entire robot code from the information extracted from a CAD file, including the interface code to handle cell devices [42.30, 31].

The example presented in Fig. 42.14 was designed to run from the CAD package INVENTOR [42.32],

Fig. 42.14 Human-user interface for the CAD programming environment

from Autodesk. Basically it enables a user to program a workcell composed by a robot and a bending machine. The robot is used to feed and handle the sheets of metal to the bending machine, and to palletize the final product. The depicted programming interface is an illustrative example of the possibilities, while there are a number of major vendors of CAD or offline programming system providing similar features in current and upcoming products.

42.5 End-Effectors and System Integration

It is interesting to note that connecting different workcell devices with each other, and integrating them into a working system, is hardly mentioned in the robotics literature. Nevertheless, in actual nontrivial installations, this part typically represents half of the cost of the installation. The automation scenario includes all the problems of integrating computers and their peripherals, plus additional issues that have to do with the variety of (electrically and mechanically incompatible) devices and their interaction with the physical environment (including its inaccuracies, tolerances, and unmodeled physical effects such as backlash and friction). The number of variations is enormous so it is often not possible to create reusable solutions. In total this results in a need for extensive engineering to put a robot to work. This engineering is what we call system integration.

Carrying out system integration according to current practice is not a scientific problem as such (although how to improve the situation is), but the obstacles it comprises

form a barrier to applying advanced sensor-based control for improved flexibility, as is needed in short-series production. In particular, in future types of applications using external sensing and high-performance feedback control within the workcell, system integration will be an even bigger problem since it includes tuning of the feedback too.

For long production runs, the engineering cost of system integration is less of a problem since its cost per manufactured part is small. On the other hand, the trend towards shorter series of customized products, or products with many variants that are not kept in stock, calls for high flexibility and short changeover times. Flexibility in this context refers to variable product variants, batch sizes, and process parameters. In particular this is a problem in small and medium-sized productions, but the trend is similar for larger enterprises as flexibility requirements are continuously on the rise. One may think that simply by using well standards for *in-*

Table 42.4 Stages of system integration, typically carried out in the order listed

Physical	Selecting equipment based on dimensioning for mechanical size, load, and stress
	Mechanical interfacing (locations, adapter plates, etc.)
	Electrical power supply (voltages and currents for robots, effectors, feeders, etc.)
	Connections for analog signals (shielding, scaling, currents, binary levels, etc.)
Communication	Interconnections for single-bit digital I/O
	Byte-wise data communication, including latencies and bit rates
	Transfer of byte sequences
Configuration	Configuration of messages between interacting devices
	Establishment of services
	Tuning for performance and resource utilization
Application	Definition of application-lever functions/services
Task	Application programming, using the application-level services

put/output (I/O) and well-defined interfaces, integration should be just a matter of connecting things together and run the system. Let us examine why this is not the case.

Example 42.2 Step 1 of 3:
Simple pick-and-insert (assembly) operation. As a first step let us consider only one robot performing pick-and-place from/to known locations, and assume it is a well-known object such that we can use one specific type of gripper (such as a gripper, in this case based on a vacuum cup). We then only need one digital output from the robot controller. Let us call this output GRIPPER (i.e., the name of the number of the output) and the values GRASP and RELEASE for the high and low signals depending on the sign of the hardware connection. An example of a robot program can be found in the previous section.

Of course the object-oriented software solution would be to have a gripper class with operations grasp and release. That would be appropriate for a robot simulator in pure software, but for integration of real systems such encapsulation of data (abstract data types) would introduce practical difficulties because:

- values are explicit and accomplished by external (in this case) hardware, and for testing and debugging we need to access and measure them,
- online operator interfaces permit direct manipulation of values, including reading of output values. The use of functions of object methods (so-called set:ers and get:ers) would only complicate the picture; maintaining consistency with the external devices (such as the definitions of the constants GRIPPER, GRASP, and RELEASE) would be no simpler.

Already in this toy example we can see that the user I/O configuration depends on both connected external devices (the gripper), and also on what I/O modules (typically units on some kind of field bus) that are in-

Robot
Force/torque sensor
Pneumatic tool-changer
Camera
3-Finger-robotic hand
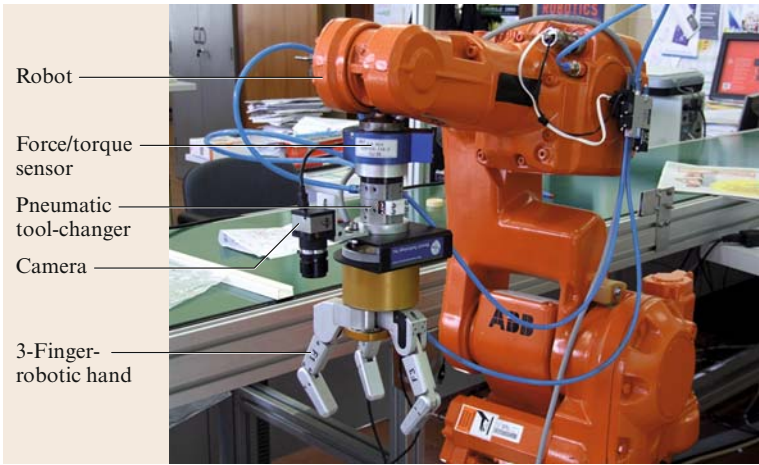
Fig. 42.15 Force controlled robot with multiple tools: force-torque sensor (JR3), robotic hand (Barrett), CCD USB camera (uEye) and tool-changer (ATI). This type of flexible gripper is not (yet) common in industry, where simple 1 DOF grippers with fingers tailored to the product are normally used
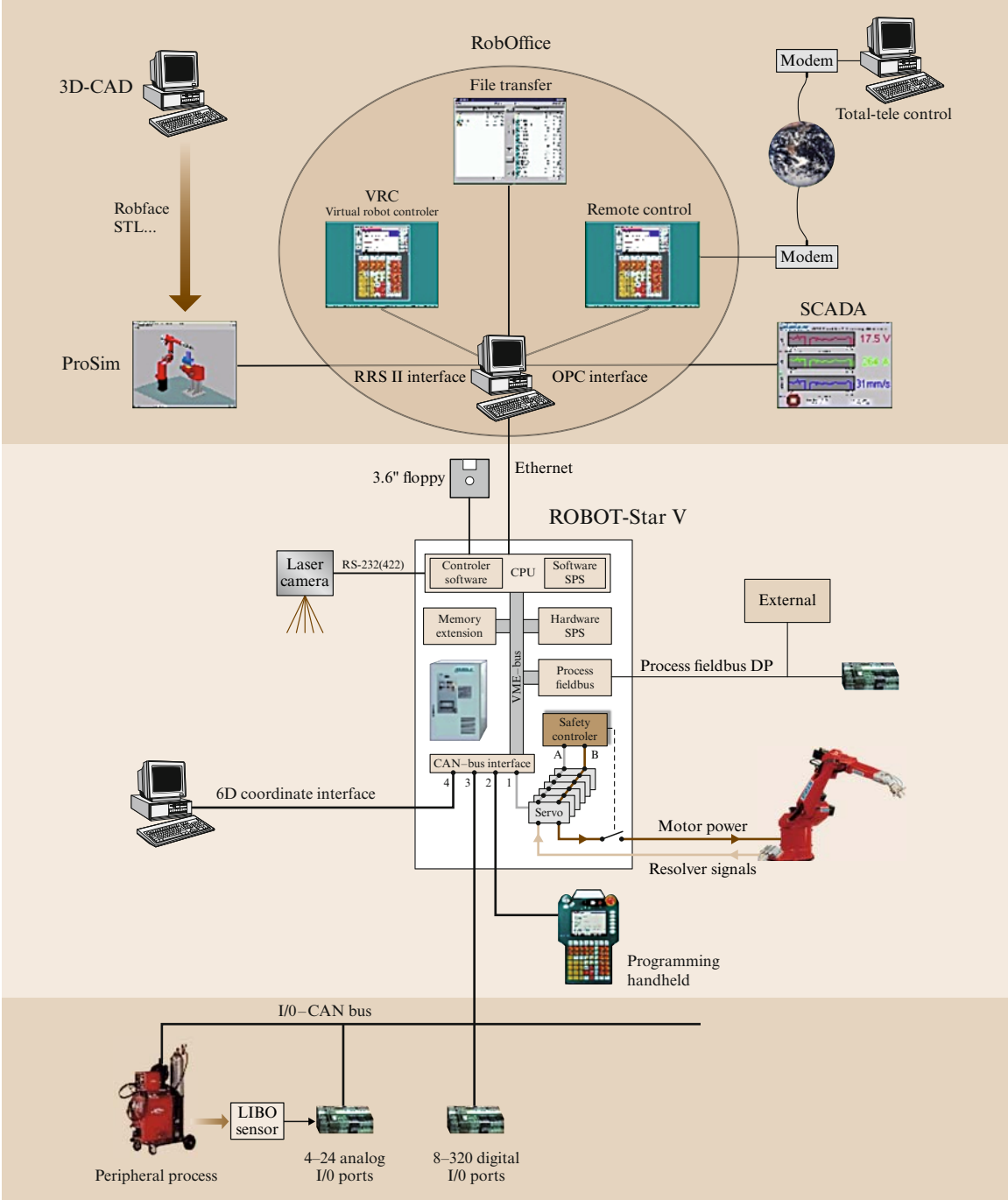
Fig. 42.16 Control modules and interfaces (Reis) on different levels, from low-level peripheral interfaces to high-level factory interfaces. The robot controller contains the middle part. The vertical integration in combination with heterogeneous hardware forms a major challenge (by courtesy Reis Robotics, Obernburg)

stalled in the robot controller (which may be subject to change when another task is given to the robot). Returning to our example, selecting modules and interfaces extends to the items of Table 42.4, where the application and task levels relate to robot programming and the research issue of how to simplify system integration, respectively.

Example 42.2 Step 2 of 3:

Simple pick-and-insert (assembly) operation. Assume we want to use a probe to detect the type and the location of the workpiece, that we need several different types of grippers depending on what object we should handle, and that the place operation needs to be a force-controlled insertion. Additionally, since we need different tools for different operations, we need a tool exchanger.

With these additional requirements we need more mechanical interfaces, the load/weight/performance considerations are more difficult (the weight of the tool exchanger and adaptor plate decreases the net payload for the grasped objects), and several more electrical interfaces need to be included.

Equipment examples are shown in Fig. 42.15, an example that extends the pick-and-insert case to include unknown positions and a gripper that can grab several types of pieces (including a wrist force sensor and force feedback in the robot fingers); details concerning robot programming and device configurations are omitted for brevity. In the presented example a three-finger robotic hand [42.33] was used to illustrate that these advanced and programmable tools are finding their way into industrial applications [42.34], for example, the hand used here has position and force-sensing capabilities. When grasping an object, the finger base and tip links move together. If a base link encounters an object it stops, while the tip link keeps moving until it makes contact with the object as well. If the tip links encounter an object first, the whole finger assembly stops moving.

The example depicted in Fig. 42.15 shows a setup where the robot can pick the workpiece from an unknown position; the working pieces are identified using a CCD camera that returns the number of pieces and their position. The robot is then commanded to pick one and update its information from the cell [42.30, 35]. This example uses two tasks: one to receive remote commands, and the other to implement the gripper and camera services.

Example 42.2 Step 3 of 3:

Simple pick-and-insert (assembly) operation. As a final requirement, to monitor the quality of the assembly operation, assume that we want to have slip detection embedded into the gripper and that we need logging of the force control signal. The production statistics should be provided on the overall plant level of the factory.

These additional requirements call for integration of low-level devices with high-level factory control system, and is hence called *vertical integration*, whereas integration within (for instance) the workcell level is called *horizontal integration*.

Lack of self-descriptive and self-contained data descriptions that are also useful at the real-time level further increases the integration effort since data interfaces/conversions typically have to be manually written. In some cases, as illustrated in Fig. 42.16, there are powerful software tools available for the integration of the robot user level and the engineering level [42.36, 37]. The fully (on all levels) integrated and nonproprietary system, sometimes referred to as a *digital factory*, still is a challenge, particularly for small and medium-sized productions [42.38, 39].

42.6 Conclusions and Long-Term Challenges

The widespread use of robots in standard, large-scale production such as the automotive industry, where robots (even with impressive performance, quality, and semiautomatic programming) perform repetitive tasks in very well-known environments, for some time resulted in the common opinion that *industrial robotics is a solved problem*. However, these applications comprise only a minor part of the industrial work needed in any wealthy society, especially considering the number of companies and the variety of applications. The use of robots in small and medium-sized manufacturing is still tiny.

Global prosperity and wealth requires resource-efficient and human-assistive robots. The challenges today are to recognize and overcome the barriers that are currently preventing robots from being more widely used.

Taking a closer look at the scientific and technological barriers, we find the following challenges:

- *Human-friendly task specification*, including intuitive ways of expressing permitted/normal/expected variations. That is, there are many upcoming and promising techniques for user-friendly human–robot interaction (such as speech, gestures, manual guidance, and so on), but the focus is still on specification of the nominal task (see Chaps. 58 and 59).

  The foreseen variations, and the unforeseen variations experienced during robot work, are more difficult to manage. When instructing a human he/she has an extensive and typically implicit knowledge about the work and the involved processes. To teach a robot, it is an issue both how to realize what the robot does not know, and how to convey the missing information efficiently.

- *Efficient mobile manipulation*. Successful implementations and systems are available for both mobility and for manipulation, but accomplished in different systems and using different types of (typically incompatible) platforms. A first step would be to accomplish mobile manipulation at all, including the *combination* of legged locomotion see Chap. 56 (for stairs and rough terrain), autonomous navigation (with adaptive but predictable understanding of constraints), dexterous manipulation, and robust force/torque interaction with environments (that have unknown stiffness). As a second step, all this needs to be done with decent performance using reasonably priced hardware, and with interfaces according to the previous item. Thus, we are far from useful mobile manipulation.

- *Low-cost components including low-cost actuation*. Actuation of high-performance robots represent about a third of the overall robot cost, and improved modularity often results in a higher total hardware cost (due to less opportunities for mechatronic optimization). On the other hand, cost-optimized (with respect to certain applications) systems result in more-specialized components and smaller volumes, with higher costs for short-series production of those components. Since future robotics and automation solutions might provide the needed cost efficiency for short-series customized components, we can interpret this as a boot-strapping problem, involving both technical and business aspects. The starting point is probably new core components that can fit into many types of systems and applications, calling for more mechatronics research and synergies with other products.

- *Composition of subsystems*. In most successful fields of engineering, the principle of superposition holds, meaning that problems can be divided into subproblems and that the solutions can then be superimposed (added/combined) onto each other such that the total solution comprises a solution to the overall problem. These principles are of key importance in physics and mathematics, and within engineering some examples are solid-state mechanics, thermal dynamics, civil engineering, and electronics. However, there is no such thing for software, and therefore not for mechatronics (which includes software) or robotics (programmable mechatronics) either. Thus, composition of unencapsulated subsystems is costly in terms of engineering effort.

  Even worse, the same applies to encapsulated software modules and subsystems. For efficiency, system interconnections should go directly to known (and hopefully standardized) interfaces, to avoid the indirections and extra load (weight, maintenance, etc.) of intermediate adapters (applying to both mechanics for end-effector mounting and to software). Interfaces can be agreed upon, but the development of new versions typically maintains backward compatibility (newer devices can be connected to old controller), while including the reuse of devices calls for mechanisms for forward compatibility (automatic upgrade based on meta information of new interfaces) to cover the case that a device is connected to a robot that is not equipped with all the legacy or vendor-specific code.

- *Embodiment of engineering and research results*. Use or deployment of new technical solutions today still starts from scratch, including analysis, understanding, implementation, testing, and so on. This is the same as for many other technical areas, but the exceptional wide variety of technologies involved with robotics and the need for flexibility and upgrading makes it especially important in this field. Embodiment into components is one approach, but knowledge can be applicable to engineering, deployment, and operation, so the representation and the principle of usage are two important issues. Improved methods are less useful if they are overly domain specific or if engineering is experienced to be significantly more complicated. Software is imperative, as well as platform and context dependent, while know-how is more declarative and symbolic. Thus, there is still a long way to go for efficient robotics engineering and reuse of know-how.

- *Open dependable systems*. Systems need to be open to permit extensions by third parties, since there is no way for system providers to foresee all upcom-

ing needs in a variety of new application areas. On the other hand, systems need to be closed such that the correctness of certain functions can be ensured. Extensive modularization in terms of hardware and supervisory software makes systems more expensive and less flexible (contrary to the needs of openness). Highly restrictive frameworks and means of programming will not be accepted for widespread use within short-time-to-market development. Most software modules do not come with formal specification, and there is less understanding of such needs. Thus, systems engineering is a key problem.

● *Sustainable manufacturing*. Manufacturing is about transformation of resources into products, and productivity (low cost and high performance) is a must.

For long-term sustainability, however, those resources in terms of materials and the like must be recycled. In most cases this can be achieved by crushing the product and sorting the materials, but in some cases disassembly and automatic sorting of specific parts are needed. There is therefore a need for robots in recycling and demanufacturing. Based on future solutions to the above items, this is then a robot application challenge.

An overall issue is how both industry and academia can combine their efforts such that sound business can be combined with scientific research so that future development overcomes the barriers that are formed by the above challenges.

References

42.1 The International Federation of Robotics (IFR): *World Robotics 2007. Statistics, Market Analysis, Forecasts, Case Studies and Profitability of Robot Investment* (IFR Statistical Department, Frankfurt 2007), http://www.ifrstat.org

42.2 M.P. Groover: *Automation, Production Systems, and Computer-Integrated Manufacturing*, 2nd edn. (Prentice Hall, Upper Saddle River 2000)

42.3 B.S. Dhillon, A.R.M. Fashandi, K.L. Liu: Robot systems reliability and safety: a review, J. Qual. Mainten. Eng. **8**(3), 170–212 (2002)

42.4 R.D. Schraft, S. Schmid, S. Thiemermann: Man–robot cooperation in a flexible assembly cell, Assemb. Autom. **22**(2), 136–138 (2002)

42.5 S.Y. Nof: *Handbook of Industrial Robotics* (Wiley, New York 1985)

42.6 V.D. Scheinman: Design of a Computer Controlled Manipulator. Ph.D. Thesis (Stanford University, Department of Computer Science 1969)

42.7 L. Westerlund: *The Extended Arm of Man. A History of the Industrial Robot* (Informations Förlaget, Stockholm 2000)

42.8 H. Makino: Assembly Robot, Patent 4341502 (1980)

42.9 G. Boothroyd, L. Altling: Design for assembly and disassembly, Int. Inst. Prod. Eng. Res. Annal. (CIRP Annals) **41**(2), 625–636 (1992)

42.10 G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, M. Schedl: DLR's torque-controlled light weight robot III – are we reaching the technological limits now?, Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Vol. 2 (Washington 2002) pp. 1710–1716

42.11 R. Clavel: Device for the movement and positioning of an element in space, Patent 4976582 (1989)

42.12 R. Bloss: Innovation at IMTS, Ind. Robot **30**(2), 159–161 (2003)

42.13 Y. Kusuda: The international robot exhibition 2005 in Tokyo, Ind. Robot **33**(5), 342–348 (2006)

42.14 Y.R. Siegwart, I.R. Nourbakhsh: *Introduction to Autonomous Mobile Robots* (MIT Press, Cambridge 2004)

42.15 R.D. Schraft, M. Hägele, A. Breckweg: *Man and robotwithout separating systems*. Managers Navigator. World of Automation and Metalworking: 8th Edition for Europe (VDMA-Verlag, Frankfurt am Main 2006) pp. 4–5

42.16 R. Bernhard, G. Schreck, C. Willnow: Development of virtual robot controllers and future trends, 6th IFAC Symp. Cost oriented Automation (Fraunhofer IPK, Berlin 2001)

42.17 K. Ikeuchi, B.K.P. Horn, S. Nagata: Picking up an object from a pile of objects, A.I. Memo 726, Artificial Intelligence Laboratory, Massachusetts Institute of Technology (1983)

42.18 J. Kirkegaard, T.B. Moeslund: Bin-picking based on harmonic shape contexts and graph-based matching, Proc. 18th Int. Conf. Pattern Recogn. (ICPR'06) (Hong Kong 2006) pp. 581–584

42.19 K. Modrich: 3D machine vision solution for bin picking applications, Proc. Int. Robot. Vision Show (Rosemont 2007)

42.20 T. Schaefer, R.D. Schraft: Incremental sheet metal forming by industrial robots, Rapid Prototyp. J. **11**(5), 278–286 (2005)

42.21 International Organization for Standardization (ISO): *Robots for Industrial Environments – Safety Requirements* (ISO, Geneva 2007), ISO 10218-1:2006/Cor 1:2007

42.22 International Organization for Standardization (ISO): *Safety of Machinery – Safety-Related Parts of Control Systems – Part 1: General Principles for Design* (ISO, Geneva 2006), ISO 13849-1:2006

42.23 International Organization for Standardization (ISO): *Safety of Machinery – Safety-Related Parts of Control Systems – Part 2: Validation* (ISO, Geneva 2003), ISO 13849-2:2003

42.24 A. Albu-Schäffer, C. Ott, G. Hirzinger: A unified passivity based control framework for position, torque and impedance control of flexible joint robots, Int. J. Robot. Res. **26**, 23–39 (2007)

42.25 A. Kochan: Robots and operators work hand in hand, Ind. Robot **33**(6), 422–424 (2006)

42.26 Three-dimensional control and monitoring – The first safe camera system SafetyEYE opens up new horizons for safety and security, Press release Pilz GmbH and Co. KG, Ostfildern, Germany, (2006) http://www.pilz.de

42.27 B. Siciliano, L. Villani: *Robot Force Control*, Ser. Eng. Comput. Sci. (Springer, Berlin, Heidelberg 2000)

42.28 J.J. Craig: *Introduction to Robotics: Mechanics and Control* (Prentice Hall, Upper Saddle River 2003)

42.29 U. Thomas, F.M. Wahl, J. Maass, J. Hesselbach: Towards a new concept of robot programming in high speed assembly applications, Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS 2005) (2005) pp. 3827–3833

42.30 J.N. Pires, A. Loureiro, G. Bolmsjö: *Welding Robots* (Springer, London 2006)

42.31 A. Blomdell, G. Bölmsjo, T. Brogardh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, T.A. Robertsson, J. Wang: Extending an industrial robot controller: implementation and applications of a fast open sensor interface, IEEE Robot. Autom. Mag. **12**(3), 85–94 (2005)

42.32 Autodesk: *Inventor Application Programming Interface. Manual* (Autodesk Inc, 2007)

42.33 W. Townsend: The BarrettHand grasper – programmable flexible part handling and assembly, Ind. Robot **27**(3), 181–188 (2000)

42.34 A. Wolf, R. Steinmann, H. Schunk: *Grippers in Motion* (Springer, New York, 2005)

42.35 J.N. Pires: *Industrial Robot Programming, Building Applications for the Factories of the Future* (Springer, New York 2007)

42.36 R. Zurawski: *Integration Technologies for Industrial Automated Systems* (CRC, Boca Raton 2006)

42.37 M. Hobday, A. Davies, A. Prencipe: Systems integration: a core capability of the modern corporation, Ind. Corp. Change **14**(6), 1109–1143 (2005)

42.38 G. Wöhlke, E. Schiller: Digital planning validation in automotive industry, Comput. Ind. **56**(4), 393–405 (2005)

42.39 E. Westkämper: Strategic development of factories under the influence of emergent technologies, CIRP Ann. Manuf. Technol. **56**(1), 419–422 (2007)