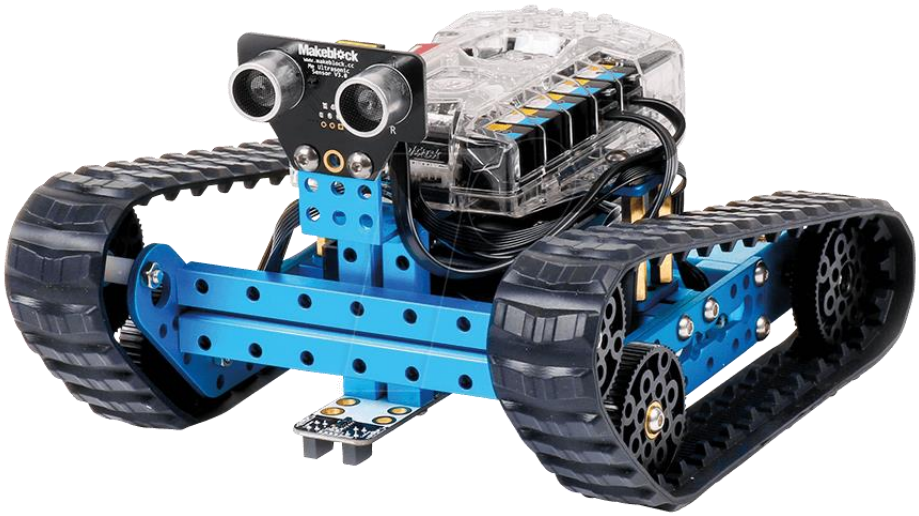
MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
STATE UNIVERSITY OF TELECOMMUNICATIONS

СТОРЧАК К. П., МИКОЛАЙЧУК В. Р., ТУШИЧ А. М.

ROBOTICS

EDUCATIONAL TEXTBOOK



Kyiv 2019

УДК 007.52: 004 (076.5)

ББК 3816.2

Гриф надано
Державним університетом
телекомунікацій
(протокол № 2 від 10.09.2018 р)

Рецензенти: доц., д.т.н. Бондарчук А. П.
проф., д.т.н. Кухах Н. І.

Навчальний посібник призначений для самостійної роботи студентів вищих навчальних закладів під час поглибленого вивчення дисципліни «Робототехніка».

Сторчак К. П., Миколайчук В. Р., Тушич А. М. Robotics. Навч. Посібник підготовано для студентів вищих навчальних закладів — Київ:ДУКІТ, 2019.-96 с.

Викладені основні відомості про роботів і робототехнічні системи різних видів. Представлено принципи будови маніпуляційних роботів і їх фізичні і технічні характеристики. Розглянуто інформаційні системи в робототехніці і мехатроніці, основи комп'ютерного зору, а також наведено принципи програмування роботів.

CONTENTS

Introduction

Many definitions have been suggested for what we call a robot. The word may conjure up various levels of technological sophistication, ranging from a simple material handling device to a humanoid. The image of robots varies widely with researchers, engineers, and robot manufacturers.

Robots in antiquity and through the Middle Ages were used primarily for entertainment. However, the 20th century featured a boom in the development of industrial robots. Through the rest of the century, robots changed the structure of society and allowed for safer conditions for labor. In addition, the implementation of advanced robotics in the military and NASA has changed the landscape of national defense and space exploration.

Robotics technology influences every aspect of work and home. Robotics has the potential to positively transform lives and work practices, raise efficiency and safety levels and provide enhanced levels of service. Even more, robotics is set to become the driving technology underpinning a whole new generation of autonomous devices and cognitive artefacts that, through their learning capabilities, interact seamlessly with the world around them, and hence, provide the missing link between the digital and physical world. Robotics is already the key driver of competitiveness and flexibility in large scale manufacturing industries.

Chapter1 Robots and their history

Introduction

It is widely accepted that today's robots used in industries originated in the invention of a programmed material handling device by George C. Devol. In 1954, Devol filed a U.S. patent for a new machine for part transfer, and he claimed the basic concept of teachin/playback to control the device. This scheme is now extensively used in most of today's industrial robots.

1.1 Era of Industrial Robots

Devol's industrial robots have their origins in two preceding technologies: numerical control for machine tools, and remote manipulation. Numerical control is a scheme to generate control actions based on stored data. Stored data may include coordinate data of points to which the machine is to be moved, clock signals to start and stop operations, and logical statements for branching control sequences.

The whole sequence of operations and its variations are prescribed and stored in a form of memory, so that different tasks can be performed without requiring major hardware changes. Modern manufacturing systems must produce a variety of products in small batches, rather than a large number of the same products for an extended period of time, and frequent changes of product models and production schedules require flexibility in the manufacturing system. The transfer line approach, which is most effective for mass production, is not appropriate when such flexibility is needed (Figure 1-1). When a major product change is required, a special-purpose production line becomes useless and often ends up being abandoned, despite the large capital investment it originally involved. Flexible automation has been a central issue in manufacturing innovation for a few decades, and numerical control has played a central role in increasing system flexibility. Contemporary industrial robots are programmable machines that can perform different operations by simply modifying stored data, a feature that has evolved from the application of numerical control.

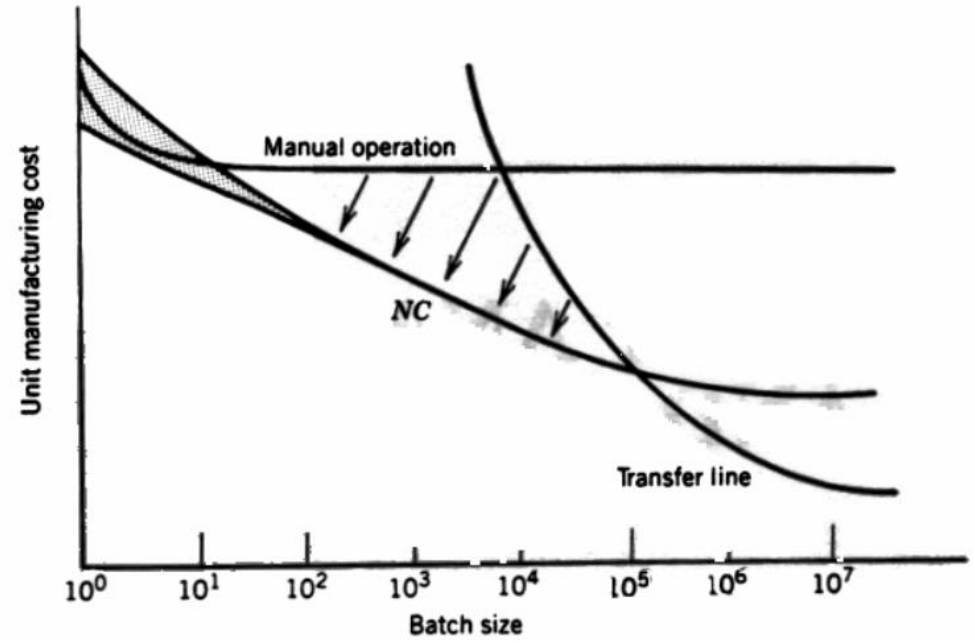
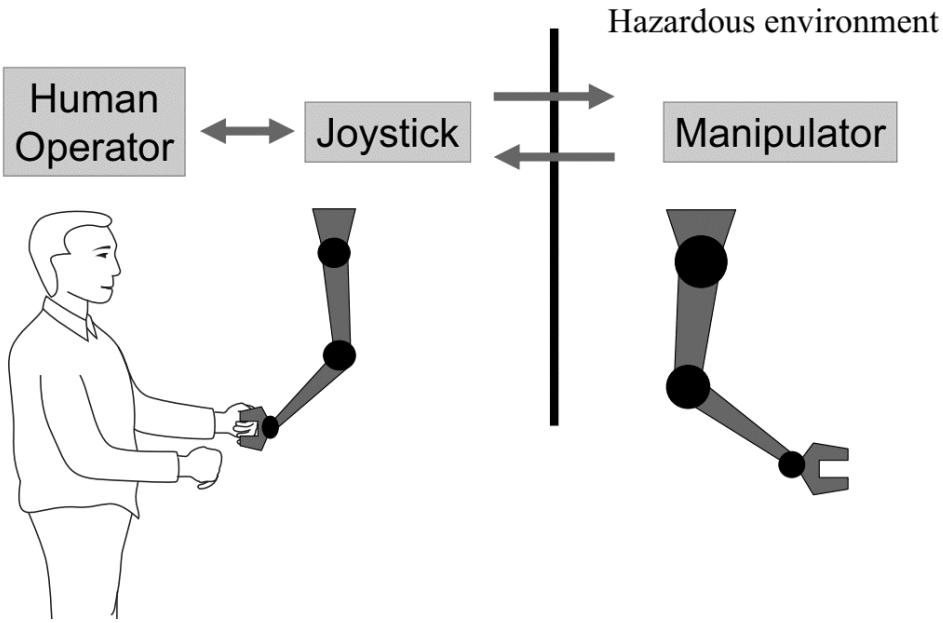


Figure 1-1 General trend of manufacturing cost vs. batch size

Another origin of today's industrial robots can be found in remote manipulators. A remote manipulator is a device that performs a task at a distance. It can be used in environments that human workers cannot easily or safely access, e.g. for handling radio-active materials, or in some deep sea and space applications. The first master-slave manipulator system was developed by 1948. The concept involves an electrically powered mechanical arm installed at the operation site, and a control joystick of geometry similar to that of the mechanical arm (Figure 1-2). The joystick has position transducers at individual joints that measure the motion of the human operator as he moves the tip of the joystick. Thus the operator's motion is transformed into electrical signals, which are transmitted to the mechanical arm and cause the same motion as the one that the human operator performed. The joystick that the operator handles is called the master manipulator, while the mechanical arm is called the slave manipulator, since its motion is ideally the replica of the operator's commanded motion. A master-slave manipulator has typically six degrees of freedom to allow the gripper to locate an object at an arbitrary position and orientation. Most joints are revolute, and the whole mechanical construction is similar to that of the human arm. This analogy with the human arm results from the need of replicating

human motions. Further, this structure allows dexterous motions in a wide range of workspaces, which is desirable for operations in modern manufacturing systems.

Contemporary industrial robots retain some similarity in geometry with both the human arm and remote manipulators. Further, their basic concepts have evolved from those of numerical control and remote manipulation. Thus a widely accepted definition of today's industrial robot is that of a numerically controlled manipulator, where the human operator and the master manipulator in the figure are replaced by a



numerical controller.

Figure 1-Assembly lines using spot welding robots

1.2 Creation of Robotics

The merge of numerical control and remote manipulation created a new field of engineering, and with it a number of scientific issues in design and control which are substantially different from those of the original technologies have emerged.

Robots are required to have much higher *mobility* and *dexterity* than traditional machine tools. They must be able to work in a large reachable range, access crowded places, handle a variety of workpieces, and perform flexible tasks. The high mobility and dexterity requirements result in the unique mechanical structure of robots, which parallels the human arm structure. This structure, however, significantly departs from traditional machine design. A robot mechanical structure is basically composed of cantilevered beams, forming a sequence of arm links connected by hinged joints. Such a structure has inherently poor mechanical stiffness and accuracy, hence is not appropriate for the heavy-duty, high-precision applications required of machine tools. Further, it also implies a serial sequence of servoed joints, whose errors accumulate along the linkage. In order to exploit the high mobility and dexterity uniquely featured by the serial linkage, these difficulties must be overcome by advanced design and control techniques.

The serial linkage geometry of manipulator arms is described by complex nonlinear equations. Effective analytical tools are necessary to understand the geometric and kinematic behavior of the manipulator, globally referred to as the manipulator kinematics. This represents an important and unique area of robotics research, since research in kinematics and design has traditionally focused upon single-input mechanisms with single actuators moving at constant speeds, while robots are multi-input spatial mechanisms which require more sophisticated analytical tools.

The dynamic behavior of robot manipulators is also complex, since the dynamics of multi-input spatial linkages are highly coupled and nonlinear. The motion of each

joint is significantly affected by the motions of all the other joints. The inertial load imposed at each joint varies widely depending on the configuration of the manipulator arm. Coriolis and centrifugal effects are prominent when the manipulator arm moves at high speeds. The kinematic and dynamic complexities create unique control problems that are not adequately handled by standard linear control techniques, and thus make effective control system design a critical issue in robotics.

Figure 1-4 Adept Direct-Drive robot

Finally, robots are required to interact much more heavily with peripheral devices than traditional numerically-controlled machine tools. Machine tools are essentially self-contained systems that handle workpieces in well-defined locations. By contrast, the environment in which robots are used is often poorly structured, and effective means must be developed to identify the locations of the workpieces as well as to communicate to peripheral devices and other machines in a coordinated fashion. Robots are also critically different from master-slave manipulators, in that they are autonomous systems.

Figure 1-5 Dexterous fingers

Master-slave manipulators are essentially manually controlled systems, where the human operator takes the decisions and applies control actions. The operator interprets a given task, finds an appropriate strategy to accomplish the task, and plans the procedure of operations. He/she devises an effective way of achieving the goal on the basis of his/her experience and knowledge about the task. His/her decisions are then transferred to the slave manipulator through the joystick. The resultant motion of the slave manipulator is monitored by the operator, and necessary adjustments or modifications of control actions are provided when the resultant motion is not adequate, or when unexpected events occur during the operation.

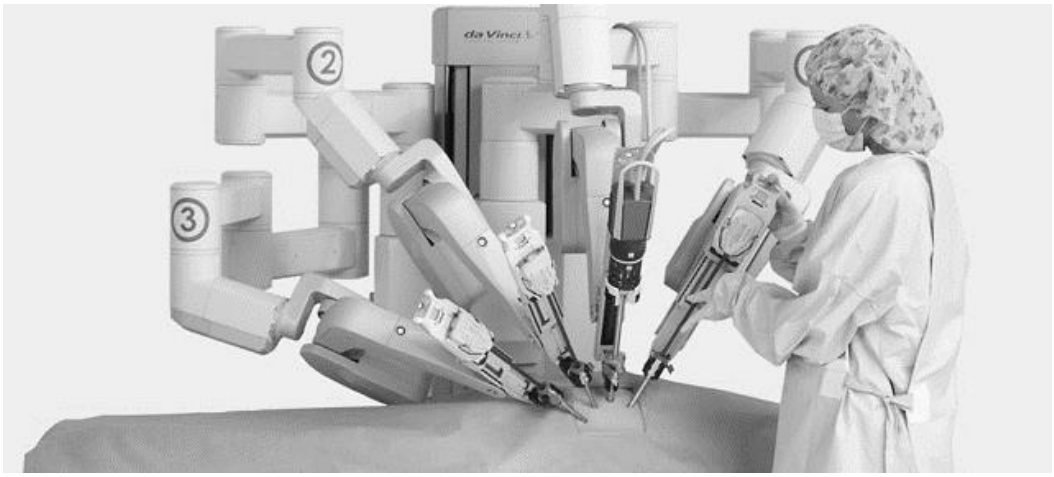


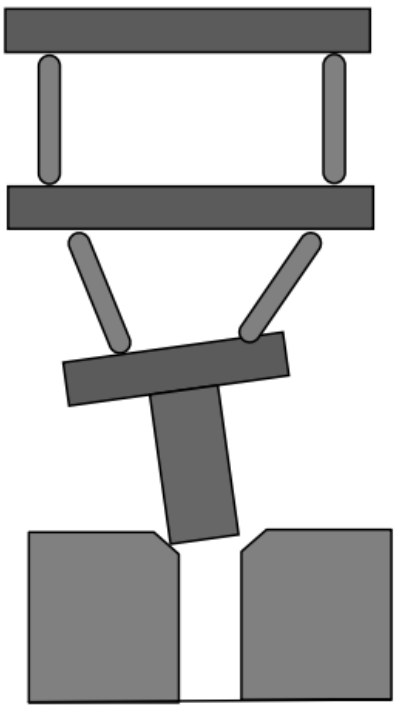
Figure 1-6 Medical robots for minimally invasive surgery

The human operator is, therefore, an essential part of the control loop. When the operator is eliminated from the control system, all the planning and control commands must be generated by the machine itself. The detailed procedure of operations must be set up in advance, and each step of motion command must be generated and coded in an appropriate form so that the robot can interpret it and execute it accurately. Effective means to store the commands and manage the data file are also needed . Thus, programming and command generation are critical issues in robotics. In addition, the robot must be able to fully monitor its own motion. In order to adapt to disturbances and unpredictable changes in the work environment, the robot needs a variety of sensors, so as to obtain information both about the environment (using external sensors, such as cameras or touch sensors) and about itself (using internal sensors, such as joint encoders or joint torque sensors). Effective sensor-based strategies that incorporate this information require advanced control algorithms. But they also imply a detailed understanding of the task.

1.3. Manipulation and Dexterity

Contemporary industrial needs drive the applications of robots to ever more advanced tasks. Robots are required to perform highly skilled jobs with minimum human assistance or intervention. To extend the applications and abilities of robots, it becomes important to develop a sound understanding of the tasks themselves.

In order to devise appropriate arm mechanisms and t develop effective control algorithms, we need to precisely understand how a given task should be accomplished and what sort of motions the robot should be able to achieve. To perform an assembly operation, for example, we need to know how to guide the assembly part to the desired location, mate it with another part, and secure it in an appropriate way. In a grinding operation, the robot must properly position the



grinding wheel while accommodating the contact force. We need to analyze the grinding process itself in order to generate appropriate force and motion commands.

A detailed understanding of the underlying principles and "know-how" involved in the task must be developed in order to use industrial robots effectively, while there is no such need for making control strategies explicit when the assembly and grinding operations are performed by a human worker. Human beings perform sophisticated manipulation tasks without being aware of the control principles involved. We have trained ourselves to be capable of skilled jobs, but in general we do not know what the acquired skills are exactly about. A sound and explicit understanding of manipulation operations, however, is essential for the long-term progress of robotics. This scientific aspect of manipulation has never been studied systematically before, and represents an emerging and important part of robotics research.

Figure 1-7 Remote-center compliance hand

1.4 Locomotion and Navigation

Robotics has found a number of important application areas in broad fields beyond manufacturing automation. These range from space and under-water exploration, hazardous waste disposal, and environment monitoring to robotic surgery, rehabilitation, home robotics, and entertainment. Many of these applications entail some locomotive functionality so that the robot can freely move around in an unstructured environment. Most industrial robots sit on a manufacturing floor and perform tasks in a structured environment. In contrast, those robots for non-manufacturing applications must be able to move around on their own. See Figure 1-8.

Locomotion and navigation are increasingly important, as robots find challenging applications in the field. This opened up new research and development areas in robotics. Novel mechanisms are needed to allow robots to move through crowded areas, rough terrain, narrow channels, and even staircases. Various types of legged robots have been studied, since, unlike standard wheels, legs can negotiate with

uneven floors and rough terrain. Among others, biped robots have been studied most extensively, resulting in the development of humanoids, as shown in Figure 1-9. Combining leg mechanisms with wheels has accomplished superior performance in both flexibility and efficiency. The Mars Rover prototype shown below has a rocker-buggy mechanism combined with advanced wheel drives in order to adapt itself to diverse terrain conditions. See Figure 1-10.

Figure 1-8 Automatically guided vehicle for meal delivery in hospitals

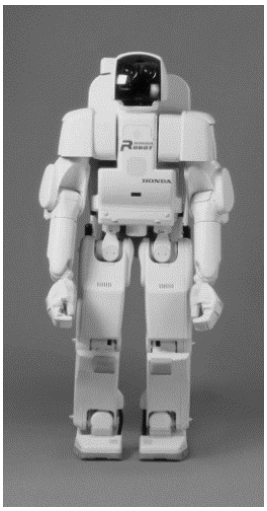


Figure 1-9 Honda's P3 humanoid robot

Navigation is another critical functionality needed for mobile robots, in particular, for unstructured environment. Those robots are equipped with range sensors and vision system, and are capable of interpreting the data to locate themselves. Often the robot has a map of the environment, and uses it for estimating the location. Furthermore, based on the real-time data obtained in the field, the robot is capable of updating and augmenting the map, which is incomplete and uncertain in unstructured environment. As depicted in Figure 1-10, location estimation and map building are simultaneously executed in the advanced navigation system. Such Simultaneous

Location and MApping (SLAM) is exactly what we human do in our daily life, and is an important functionality of intelligent robots.

The goal of robotics is thus two-fold: to extend our understanding about manipulation, locomotion, and other robotic behaviors and to develop engineering methodologies to actually perform desired tasks. The goal of this book is to provide entry-level readers and experienced engineers with fundamentals of understanding robotic tasks and intelligent behaviors as well as with enabling technologies needed for building and controlling robotic systems.

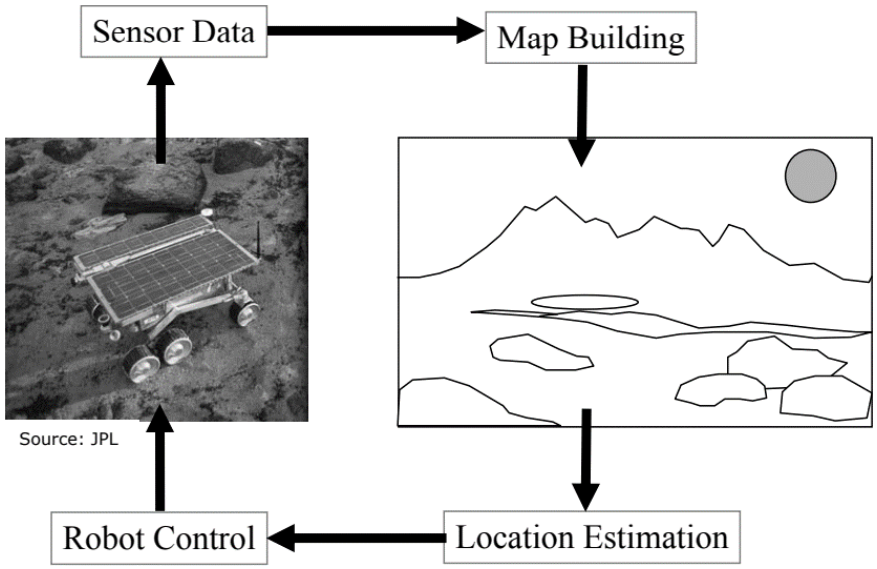


Figure 1-10 JPL's planetary exploration robot: an early version of the Mars Rover

There are many types of robots; they are used in many different environments and for many different uses, although being very diverse in application and form they all share three basic similarities when it comes to their construction:

1. Robots all have some kind of mechanical construction, a frame, form or shape designed to achieve a particular task. For example, a robot designed to travel across heavy dirt or mud, might use caterpillar tracks. The mechanical aspect is mostly the creator's solution to completing the assigned task and dealing with the physics of the environment around it. Form follows function.

2. Robots have electrical components which power and control the machinery. For example, the robot with caterpillar tracks would need some kind of power to move the tracker treads. That power comes in the form of electricity, which will have to travel through a wire and originate from a battery, a basic electrical circuit. Even petrol powered machines that get their power mainly from petrol still require an electric current to start the combustion process which is why most petrol powered machines like cars, have batteries. The electrical aspect of robots is used for movement (through motors), sensing (where electrical signals are used to measure things like heat, sound, position, and energy status) and operation (robots need some level of electrical energy supplied to their motors and sensors in order to activate and perform basic operations)

3. All robots contain some level of computer programming code. A program is how a robot decides when or how to do something. In the caterpillar track example, a robot that needs to move across a muddy road may have the correct mechanical construction and receive the correct amount of power from its battery, but would not go anywhere without a program telling it to move. Programs are the core essence of a robot, it could have excellent mechanical and electrical construction, but if its program is poorly constructed its performance will be very poor (or it may not perform at all). There are three different types of robotic programs: remote control, artificial intelligence and hybrid. A robot with remote control programing has a preexisting set of commands that it will only perform if and when it receives a signal from a control source, typically a human being with a remote control. It is perhaps

more appropriate to view devices controlled primarily by human commands as falling in the discipline of automation rather than robotics. Robots that use artificial intelligence interact with their environment on their own without a control source, and can determine reactions to objects and problems they encounter using their preexisting programming. Hybrid is a form of programming that incorporates both AI and RC functions.

As more and more robots are designed for specific tasks this method of classification becomes more relevant. For example, many robots are designed for assembly work, which may not be readily adaptable for other applications. They are termed as "assembly robots". For seam welding, some suppliers provide complete welding systems with the robot i.e. the welding equipment along with other material handling facilities like turntables etc. as an integrated unit. Such an integrated robotic system is called a "welding robot" even though its discrete manipulator unit could be adapted to a variety of tasks. Some robots are specifically designed for heavy load manipulation, and are labelled as "heavy duty robots".

Current and potential applications include:

- Military robots.
- Industrial robots. Robots are increasingly used in manufacturing (since the 1960s). According to the Robotic Industries Association US data, in 2016 automotive industry was the main customer of industrial robots with 52% of total sales.[21] In the auto industry, they can amount for more than half of the "labor". There are even "lights off" factories such as an IBM keyboard manufacturing factory in Texas that was fully automated as early as 2003.
- Cobots (collaborative robots).
- Construction robots. Construction robots can be separated into three types: traditional robots, robotic arm, and robotic exoskeleton.
- Agricultural robots (AgRobots). The use of robots in agriculture is closely linked to the concept of AI-assisted precision agriculture and drone usage. 1996-1998 research also proved that robots can perform a herding task.

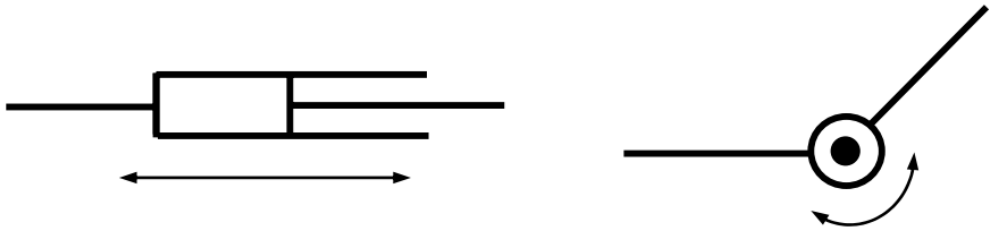
- Medical robots of various types (such as da Vinci Surgical System and Hospi).
- Kitchen automation. Commercial examples of kitchen automation are Flippy (burgers), Zume Pizza (pizza), Cafe X (coffee), Makr Shakr (coctails), Frobot (frozen yogurts) and Sally (salads). Home examples are Rotimatic (flatbreadsbaking) and Boris (dishwasher loading).
- Robot combat for sport – hobby or sport event where two or more robots fight in an arena to disable each other. This has developed from a hobby in the 1990s to several TV series worldwide.
- Cleanup of contaminated areas, such as toxic waste or nuclear facilities.
- Domestic robots.
- Nanorobots.
- Swarm robotics.
- Autonomous drones.

Chapter 2 Configuration of manipulators of industrial robots

In this chapter, we will first overview various types of mechanisms used for generating robotic motion, and introduce some taxonomy of mechanical structures before going into a more detailed analysis in the subsequent chapters.

2.1 Joint Primitives and Serial Linkages

A robot mechanism is a multi-body system with the multiple bodies connected together. We begin by treating each body as rigid, ignoring elasticity and any deformations caused by large load conditions. Each rigid body involved in a robot mechanism is called a link, and a combination of links is referred to as a linkage. In describing a linkage it is fundamental to represent how a pair of links is connected to each other. There are two types of primitive connections between a pair of links, as shown in Figure 3.1.1. The first is a prismatic joint where the pair of links makes a translational displacement along a fixed axis. In other words, one link slides on the other along a straight line. Therefore, it is also called a sliding joint. The second type of primitive joint is a revolute joint where a pair of links rotates about a fixed axis. This type of joint is often referred to as a hinge, articulated, or rotational joint[1].



(a) Prismatic joint (b) Revolute joint

Figure 3.1 Primitive joint types: (a) a prismatic joint and (b) a revolute joint

Combining these two types of primitive joints, we can create many useful mechanisms for robot manipulation and locomotion. These two types of primitive joints are simple to build and are well grounded in engineering design. Most of the robots that have been built are combinations of only these two types. Let us look at some examples.

[1]It is interesting to note that all biological creatures are made of revolute type joints; there are no sliding joints involved in their extremities.

2.2 Robot mechanisms analogous to coordinate systems

One of the fundamental functional requirements for a robotic system is to locate its end effecter, e.g. a hand, a leg, or any other part of the body performing a task, in three-dimensional space. If the kinematic structure of such a robot mechanism is analogous to a coordinate system, it may suffice this positioning requirement. Figures 2.1.2~ 4 show three types of robot arm structures corresponding tothe Cartesian coordinate system, the cylindrical coordinate system, and the spherical coordinate system respectively. TheCartesian coordinate robot shown in Figure 3.1.2 has three prismatic joints, corresponding to three axes denoted x,y,and z. The cylindrical robot consists of one revolute joint and two prismatic joints, with r, and z representing thecoordinates of the end-effecter. Likewise, the spherical robot has two revolute joints denoted and one prismatic jointdenoted r.

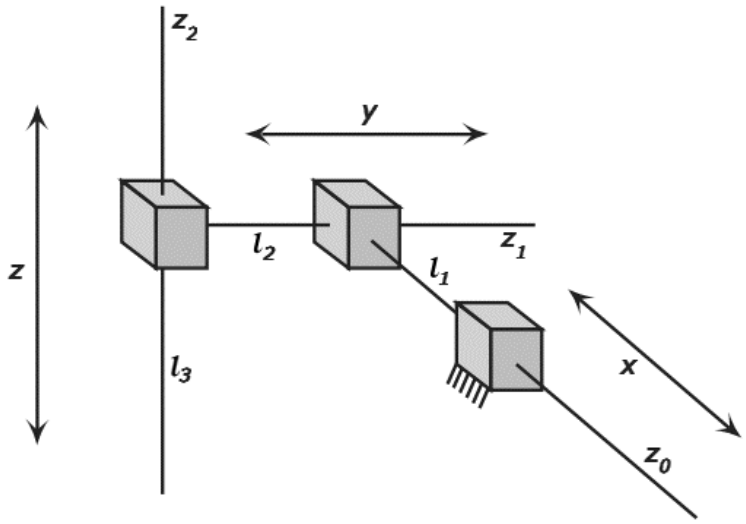


Figure 3.2 Cartesian coordinate robot

There are many other ways of locating an end-effecter in three-dimensional space. Figure 3.1.5 ~ 6 show other kinematicstructures that allow the robot to locate its end-effecter in three-dimensional space. Although these mechanisms have no analogy with common coordinate systems, they are capable of locating the end-effecter in space, and have salient features desirablefor specific tasks. The first one is a so-called SCALAR robot consisting of two revolute joints and one prismatic joint.This robot structure is particularly desirable for assembly automation in

manufacturing systems, having a wide workspace inthe horizontal direction and an independent vertical axis appropriate for insertion of parts.

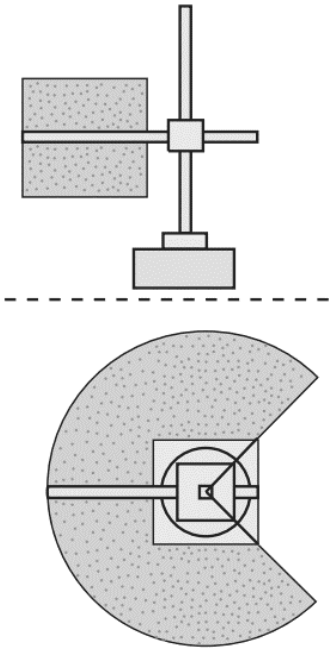


Figure 2.1.3 Cylindrical coordinate robot

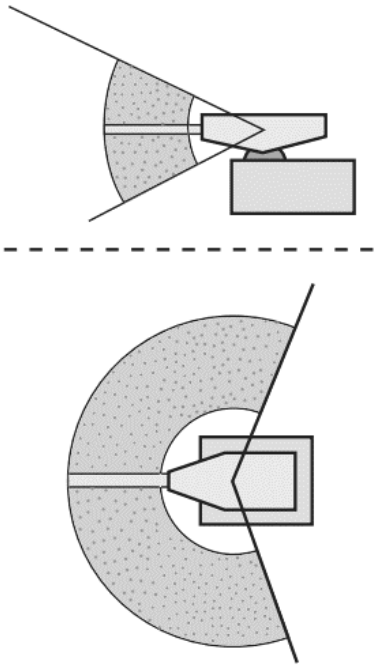


Figure 2.1.4 Spherical coordinate robot.

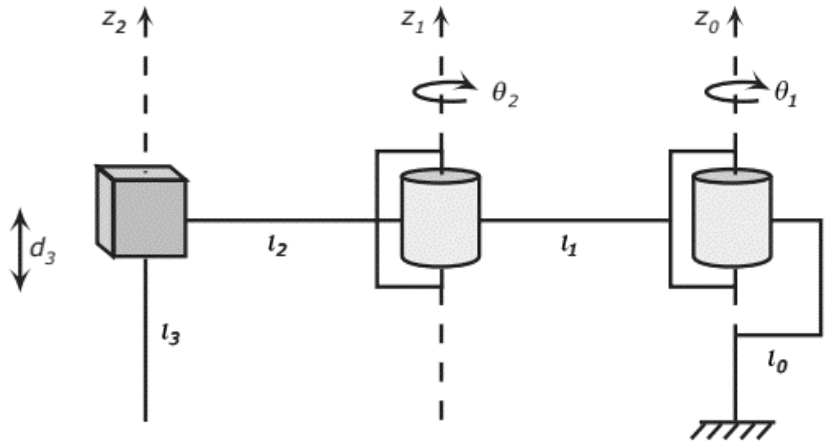


Figure 3.1.5 SCALAR type robot.

The second type, called an articulated robot or an elbow robot, consists of all three revolute joints, like a human arm. This type of robot has a great degree of flexibility and versatility, being the most standard structure of robot manipulators. The third kinematic structure, also consisting of three revolute joints, has a unique mass balancing structure. The counter balance at the elbow eliminates gravity load for all three joints, thus reducing toque requirements for the actuators. This structure has been used for the direct-drive robots having no gear reducer.

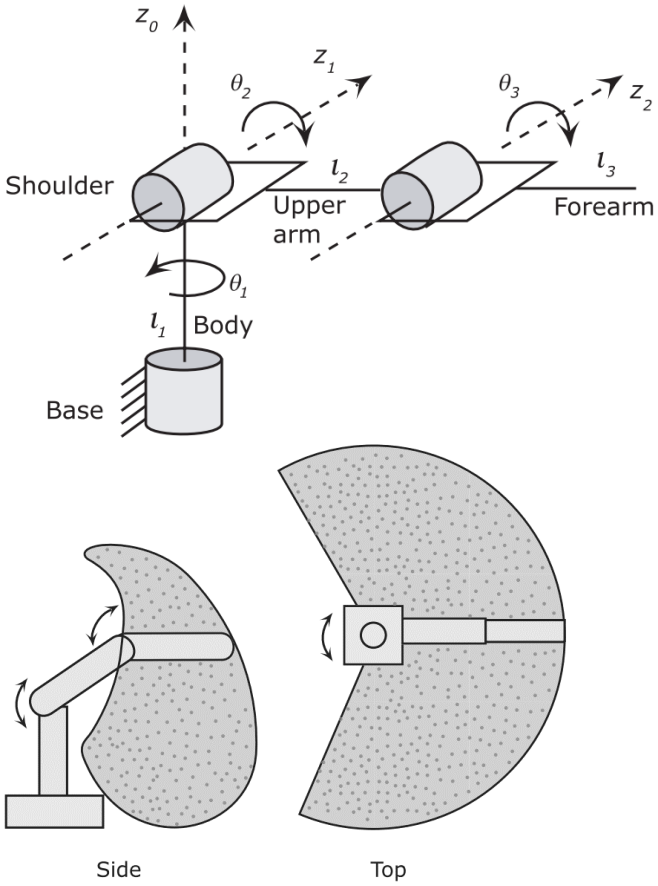


Figure 3.1.6 Articulated robot

Note that all the above robot structures are made of serial connections of primitive joints. This class of kinematic structures, termed a serial linkage, constitutes the fundamental makeup of robot mechanisms. They have no kinematic constraint in each joint motion, i.e. each joint displacement is a generalized coordinate. This facilitates the analysis and control of the robot mechanism. There are, however, different classes of mechanisms used for robot structures. Although more complex, they do provide some useful properties. We will look at these other mechanisms in the subsequent sections.

2.3 Parallel Linkages

Primitive joints can be arranged in parallel as well as in series. Figure 3.2.1 illustrates such a parallel link mechanism. It is a five-bar-linkage consisting of five links, including the base link, connected by five joints. This can be viewed as two serial linkage arms connected at a particular point, point A in the figure. It is important to note that there is a closed kinematic chain formed by the five links and, thereby, the two serial link arms must conform to a certain geometric constraint. It is clear from the figure that the end-effecter position is determined if two of the five joint angles are given. For example, if angles 1? and 3? of joints 1 and 3 are determined, then all the link positions are determined, as is the end-effecter's. Driving joints 1 and 3 with two actuators, we can move the end-effecter within the vertical plane. It should be noted that, if more than two joints were actively driven by independent actuators, a conflict among three actuators would occur due to the closed-loop kinematic chain. Three of the five joints should be passive joints, which are free to rotate. Only two joints should be active joints, driven by independent actuators.

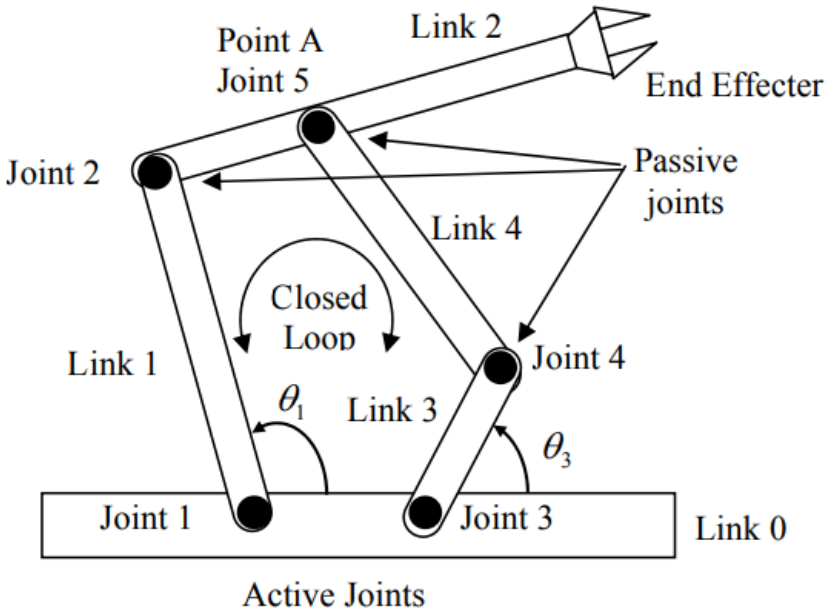


Figure 3.2.1 Five-bar-link parallel link robot

This type of parallel linkage, having a closed-loop kinematic chain, has significant features. First, placing both actuators at the base link makes the robot arm lighter, compared to the serial link arm with the second motor fixed to the tip of link 1. Second, a larger end-effecter load can be born with the two serial linkage arms sharing the load. Figure 3.2.2 shows a heavy duty robot having a parallel link mechanism.

Figure 3.2.2 Heavy-duty robot with parallel link mechanism

Figure 3.2.3 shows the Stewart mechanism, which consists of a moving platform, a fixed base, and six powered cylinders connecting the moving platform to

the base frame. The position and orientation of the moving platform are determined by the six independent actuators. The load acting on the moving platform is born by the six "arms". Therefore, the load capacity is generally large, and dynamic response is fast for this type of robot mechanisms. Note, however, that this mechanism has spherical joints, a different type of joints than the primitive joints we considered initially.

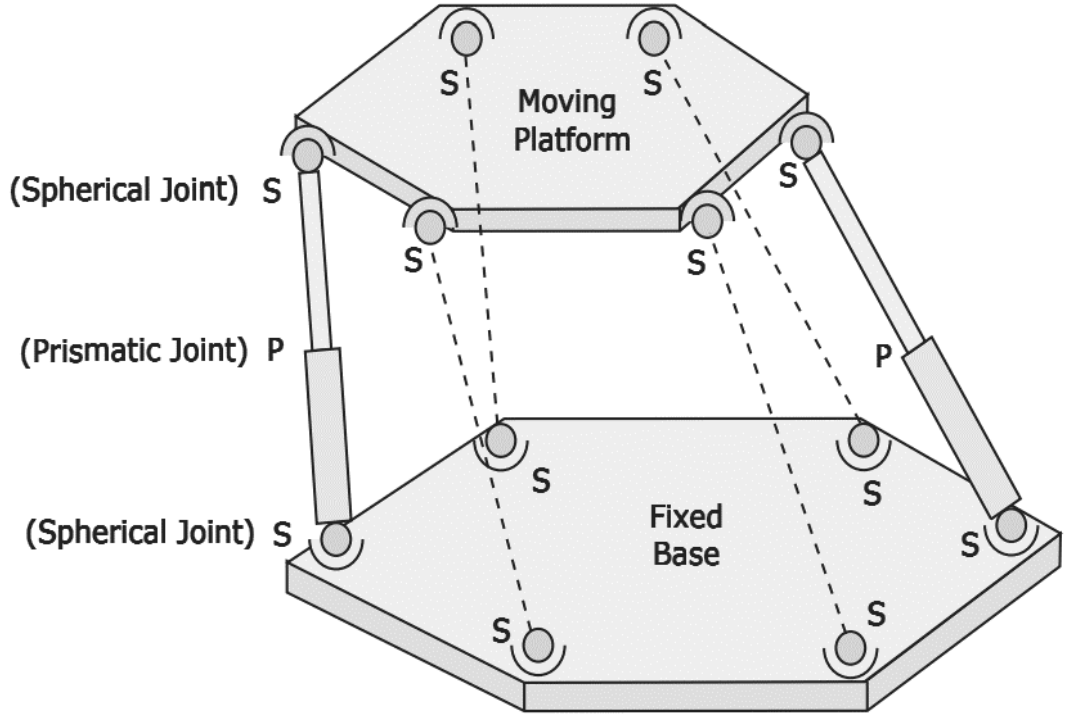


Figure 3.2.3 Stewart mechanism parallel-link robot

Chapter 3 Planar Kinematics

Kinematics is *Geometry of Motion*. It is one of the most fundamental disciplines in robotics, providing tools for describing the structure and behavior of robot mechanisms. In this chapter, we will discuss how the motion of a robot mechanism is described, how it responds to actuator movements, and how the individual actuators should be coordinated to obtain desired motion at the robot end-effecter. These are questions central to the design and control of robot mechanisms.

To begin with, we will restrict ourselves to a class of robot mechanisms that work within a plane, i.e. *Planar Kinematics*. Planar kinematics is much more tractable mathematically, compared to general three-dimensional kinematics. Nonetheless, most of the robot mechanisms of practical importance can be treated as planar mechanisms, or can be reduced to planar problems. General three-dimensional kinematics, on the other hand, needs special mathematical tools, which will be discussed in later chapters.

3.1 Planar Kinematics of Serial Link Mechanisms

Example 3.1 Consider the three degree-of-freedom planar robot arm shown in Figure 3.1.1. The arm consists of one fixed link and three movable links that move within the plane. All the links are connected by revolute joints whose joint axes are all perpendicular to the plane of the links. There is no closed-loop kinematic chain; hence, it is a serial link mechanism.

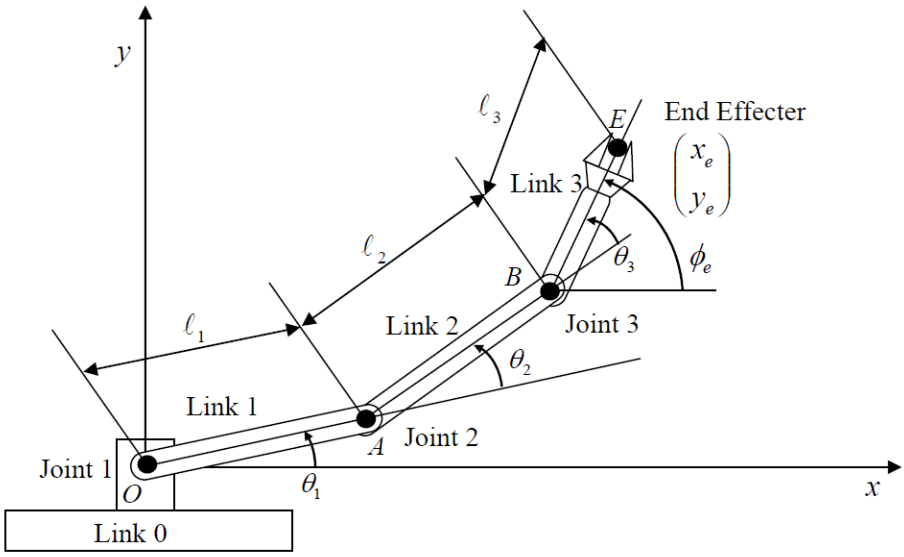


Figure 3.1 Three dof planar robot with three revolute joints

To describe this robot arm, a few geometric parameters are needed. First, the length of each link is defined to be the distance between adjacent joint axes. Let points *O*, *A*, and *B* be the locations of the three joint axes, respectively, and point E be a point fixed to the end-effecter. Then the link lengths are $l_1 = \overline{OA}$, $l_2 = \overline{AB}$, $l_3 = \overline{BE}$. Let us assume that Actuator 1 driving link 1 is fixed to the base link (link 0), generating angle θ_1, while Actuator 2 driving link 2 is fixed to the tip of Link 1, creating angle θ_2 between the two links, and Actuator 3 driving Link 3 is fixed to the tip of Link 2, creating angle θ_3, as shown in the figure. Since this robot arm performs tasks by moving its end-effecter at point E, we are concerned with the location of the end-effecter. To describe its location, we use a coordinate system, *O-xy*, fixed to the base link with the origin at the first joint, and describe the end-effecter position with coordinates x_e and y_e. We can relate the end-effecter coordinates to the joint angles determined by the three actuators by using the link lengths and joint angles defined above:

$$x_e = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \quad (3.1)$$

$$y_e = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \quad (3.2)$$

This three dof robot arm can locate its end-effecter at a desired orientation as well as at a desired position. The orientation of the end-effecter can be described as the angle the centerline of the end-effecter measured from the positive *x* coordinate axis. This end-effecter orientation ϕ_e is related to the actuator displacements as

$$\phi_e = \theta_1 + \theta_2 + \theta_3 \quad (3.3)$$

The above three equations describe the position and orientation of the robot end-effecter viewed from the fixed coordinate system in relation to the actuator displacements. In general, a set of algebraic equations relating the position and orientation of a robot end-effecter, or any significant part of the robot, to actuator or active joint displacements, is called ***Kinematic Equations,*** or more specifically, ***Forward Kinematic Equations*** in the robotics literature.

Exercise 3.1

Shown below in Figure 3.1.2 is a planar robot arm with two revolute joints and one prismatic joint. Using the geometric parameters and joint displacements, obtain the kinematic equations relating the end-effecter position and orientation to the joint displacements.

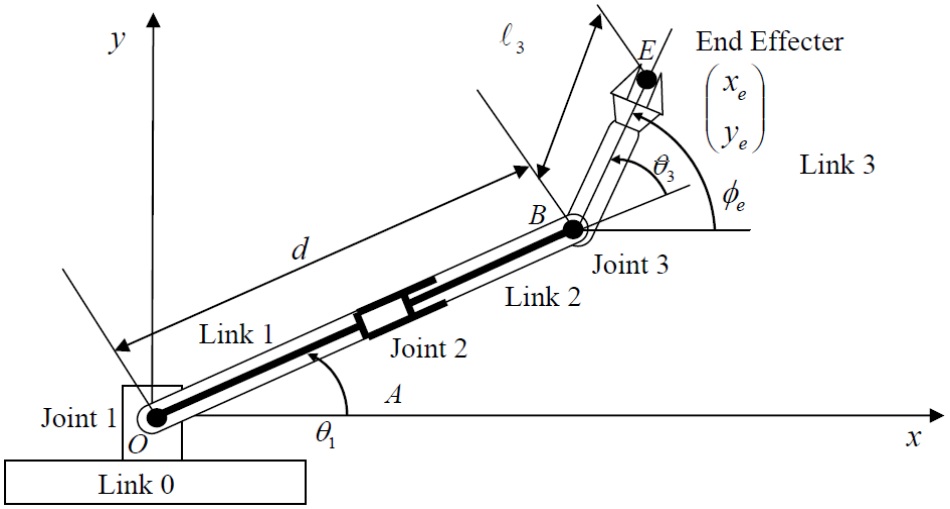


Figure 3.2 Three dof robot with two revolute joints and one prismatic joint

Now that the above Example and Exercise problems have illustrated kinematic equations, let us obtain a formal expression for kinematic equations. As mentioned in the previous chapter, two types of joints, prismatic and revolute joints, constitute robot mechanisms in most cases. The displacement of the *i-th* joint is described by distance d_i if it is a prismatic joint, and by angle θ_i for a revolute joint. For formal expression, let us use a generic notation: q_i. Namely, joint displacement q_i represents either distance d_i or angle θ_i depending on the type of joint.

$$q_i = \begin{cases} d_i & \text{Prismatic joint} \\ \theta_i & \text{Revolute joint} \end{cases} \qquad (3.4)$$

We collectively represent all the joint displacements involved in a robot mechanism with a column vector: $q = [q_1, q_2, ..., q_n]^T$, where *n* is the number of joints. Kinematic equations relate these joint displacements to the position and orientation of the end-effecter. Let us collectively denote the end-effecter position and orientation by vector ***p***. For planar mechanisms, the end-effecter location is described by three variables:

$$p = \begin{bmatrix} x_e \\ y_e \\ \phi_e \end{bmatrix} \qquad (3.5)$$

Using these notations, we represent kinematic equations as a vector function relating ***p*** to ***q***:

$$p = f(q), \;\; p \in \Re^{3\times 1}, \;\; q \in \Re^{n\times 1} \qquad (3.6)$$

For a serial link mechanism, all the joints are usually active joints driven by individual actuators. Except for some special cases, these actuators uniquely determine the end-effecter position and orientation as well as the configuration of the entire robot mechanism. If there is a link whose location is not fully determined by the actuator displacements, such a robot mechanism is said to be ***under-actuated***. Unless a robot mechanism is under-actuated, the collection of the joint displacements, i.e. the vector ***q,*** uniquely determines the entire robot configuration. For a serial link mechanism, these joints are independent, having no geometric constraint other than their stroke limits. Therefore, these joint displacements are ***generalized coordinates*** that locate the robot mechanism uniquely and completely. Formally, the number of generalized coordinates is called ***degrees of freedom.*** Vector ***q*** is called joint coordinates, when they form a complete and independent set of generalized coordinates.

3.2 Inverse Kinematics of Planar Mechanisms

The vector kinematic equation derived in the previous section provides the functional relationship between the joint displacements and the resultant end-effecter position and orientation. By substituting values of joint displacements into the right-hand side of the kinematic equation, one can immediately find the corresponding end-effecter position and orientation. The problem of finding the end-effecter position and orientation for a given set of joint displacements is referred to as the *direct kinematics problem.* This is simply to evaluate the right-hand side of the kinematic equation for known joint displacements. In this section, we discuss the problem of moving the end-effecter of a manipulator arm to a specified position and orientation. We need to find the joint displacements that lead the end-effecter to the specified position and

orientation. This is the inverse of the previous problem, and is thus referred to as the *inverse kinematics problem.* The kinematic equation must be solved for joint displacements, given the end-effecter can be achieved by moving each joint to the determined value.

In the direct kinematics problem, the end-effecter location is determined uniquely for any given set of joint displacements. On the other hand, the inverse kinematics is more complex in the sense that multiple solutions may exist for the same end-effecter location. Also, solutions may not always exist for a particular range of end-effecter locations and arm structures. Furthermore, since the kinematic equation is comprised of nonlinear simultaneous equations with many trigonometric functions, it is not always possible to derive a closed-form solution, which is the explicit inverse function of the kinematic equation. When the kinematic equation cannot be solved analytically, numerical methods are used in order to derive the desired joint displacements.

Example 3.2 Consider the three dof planar arm shown in Figure 3.1.1 again. To solve its inverse kinematics problem, the kinematic structure is redrawn in Figure 3.2.1. The problem is to find three joint angles $\theta_1, \theta_2, \theta_3$ that lead the end effecter to a desired position and orientation, *eeeyx*φ,,. We take a two-step approach. First, we find the position of the wrist, point B, from x_e, y_e, ϕ_e . Then we find θ_1, θ_2 from the wrist position. Angle θ_3 can be determined immediately from the wrist position.

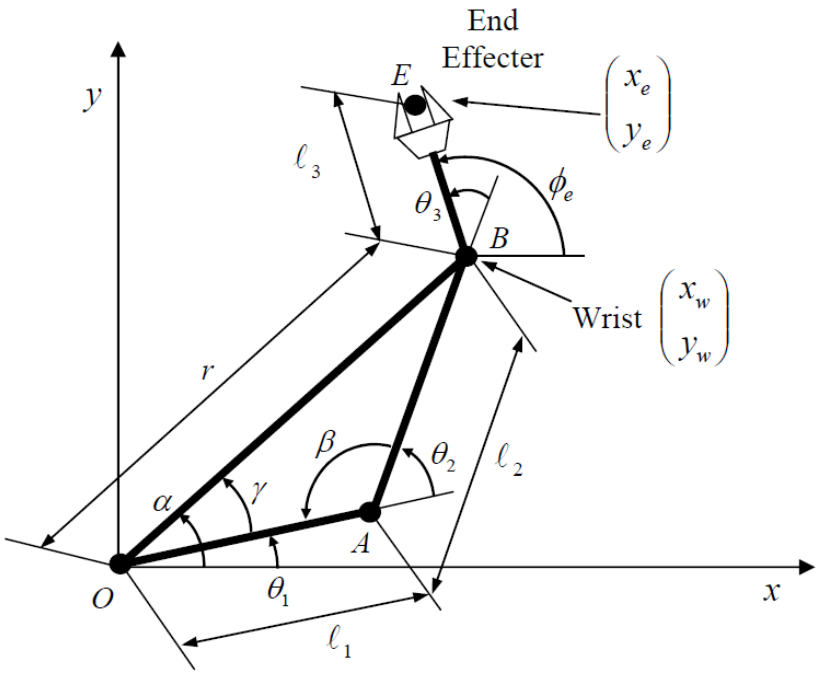


Figure 3.3 Skeleton structure of the robot arm of Example 3.1

Let x_w and y_w be the coordinates of the wrist. As shown in Figure 3.3, point B is at distance l_3 3 from the given end-effecter position E. Moving in the opposite direction to the end effecter orientation ϕ_e, the wrist coordinates are given by

$$\begin{aligned} x_w &= x_e - l_3 \cos\phi_e \\ y_w &= y_e - l_3 \sin\phi_e \end{aligned} \qquad (3.7)$$

Note that the right hand sides of the above equations are functions of x_e, y_e, ϕ_e, alone. From these wrist coordinates, we can determine the angle α shown in the figure.[2]

$$\alpha = \tan^{-1}\frac{y_w}{x_w} \qquad (3.8)$$

Next, let us consider the triangle *OAB* and define angles $\gamma, \beta,$ as shown in the figure. This triangle is formed by the wrist *B*, the elbow *A*, and the shoulder *O*. Applying the law of cosines to the elbow angle β yields

$$l_1^2 + l_2^2 - 2l_1 l_2 \cos\beta = r^2 \qquad (3.9)$$

[2]Unless noted specifically we assume that the arc tangent function takes an angle in a proper quadrant consistent with the signs of the two operands.

where $r^2 = x_w^2 + y_w^2$, the squared distance between O and B. Solving this for angle yields

$$\theta_2 = \pi - \beta = \pi - \cos^{-1} \frac{l_1^2 + l_2^2 - x_w^2 - y_w^2}{2 l_1 l_2} \qquad (3.10)$$

Similarly,

$r^2 + l_1^2 - 2 r l_1 \cos\gamma = l_2^2$ (3.2.5)

Solving this for γ yields

$$\theta_1 = \alpha - \gamma = \tan^{-1} \frac{y_w}{x_w} - \cos^{-1} \frac{x_w^2 + y_w^2 + l_1^2 - l_2^2}{2 l_1 \sqrt{x_w^2 + y_w^2}} \qquad (3.11)$$

From the above θ_1, θ_2 we can obtain

$$\theta_3 = \phi_e - \theta_1 - \theta_2 \qquad (3.12)$$

Eqs. (4), (6), and (7) provide a set of joint angles that locates the end-effecter at the desired position and orientation. It is interesting to note that there is another way of reaching the same end-effecter position and orientation, i.e. another solution to the inverse kinematics problem. Figure 3.3 shows two configurations of the arm leading to the same end-effecter location: the elbow down configuration and the elbow up configuration. The former corresponds to the solution obtained above. The latter, having the elbow position at point *A'*, is symmetric to the former configuration with respect to line *OB*, as shown in the figure. Therefore, the two solutions are related as

$$\begin{aligned} \theta_1' &= \theta_1 + 2\gamma \\ \theta_2' &= -\theta_2 \\ \theta_3' &= \phi_e - \theta_1' - \theta_2' = \theta_3 + 2\theta_2 - 2\gamma \end{aligned} \qquad (3.13)$$

Inverse kinematics problems often possess multiple solutions, like the above example, since they are nonlinear. Specifying end-effecter position and orientation does not uniquely determine the whole configuration of the system. This implies that vector ***p,*** the collective position and orientation of the end-effecter, cannot be used as generalized coordinates.

The existence of multiple solutions, however, provides the robot with an extra degree of flexibility. Consider a robot working in a crowded environment. If multiple configurations exist for the same end-effecter location, the robot can take a configuration having no interference with the environment. Due to physical

limitations, however, the solutions to the inverse kinematics problem do not necessarily provide feasible configurations. We must check whether each solution satisfies the constraint of movable range, i.e. stroke limit of each joint.

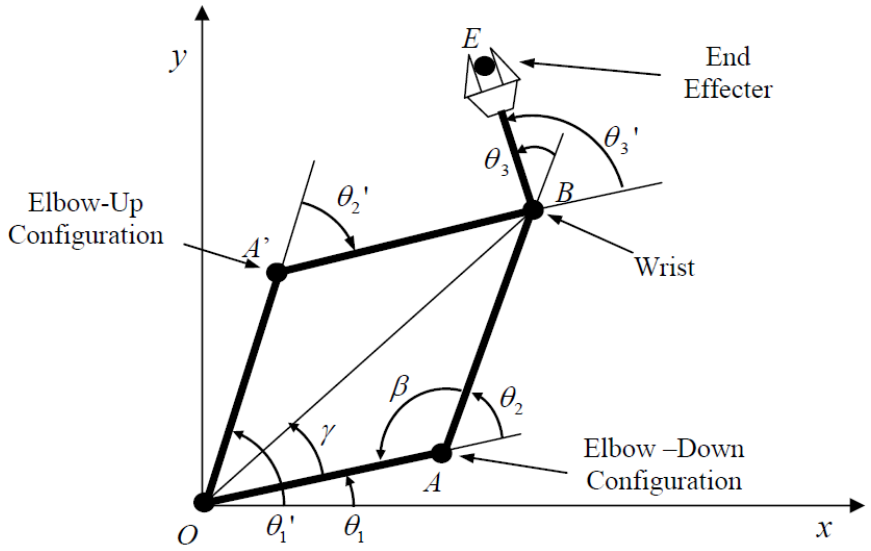


Figure 3.4Multiple solutions to the inverse kinematics problem of Example 3.2

3.3 Kinematics of Parallel Link Mechanisms

Example 3.3 Consider the five-bar-link planar robot arm shown in Figure 3.5.

$$\begin{aligned} x_e &= l_1 \cos\theta_1 + l_5 \cos\theta_2 \\ y_e &= l_1 \sin\theta_1 + l_5 \sin\theta_2 \end{aligned} \qquad (3.14)$$

Note that Joint 2 is a passive joint. Hence, angle θ_2 is a dependent variable. Using θ_2 , however, we can obtain the coordinates of point A:

$$\begin{aligned} x_A &= l_1 \cos\theta_1 + l_5 \cos\theta_2 \\ y_A &= l_1 \sin\theta_1 + l_5 \sin\theta_2 \end{aligned} \qquad (3.15)$$

Point A must be reached via the branch comprising Links 3 and 4. Therefore,

$$\begin{aligned} x_e &= l_3 \cos\theta_3 + l_4 \cos\theta_4 \\ y_e &= l_3 \sin\theta_3 + l_4 \sin\theta_4 \end{aligned} \qquad (3.16)$$

Equating these two sets of equations yields two constraint equations:

$$\begin{aligned} l_1 \cos\theta_1 + l_5 \cos\theta_2 &= l_3 \cos\theta_3 + l_4 \cos\theta_4 \\ l_1 \sin\theta_1 + l_5 \sin\theta_2 &= l_3 \sin\theta_3 + l_4 \sin\theta_4 \end{aligned} \qquad (3.17)$$

Note that there are four variables and two constraint equations. Therefore, two of the variables, such as θ_1, θ_3 , are independent. It should also be noted that multiple solutions exist for these constraint equations.

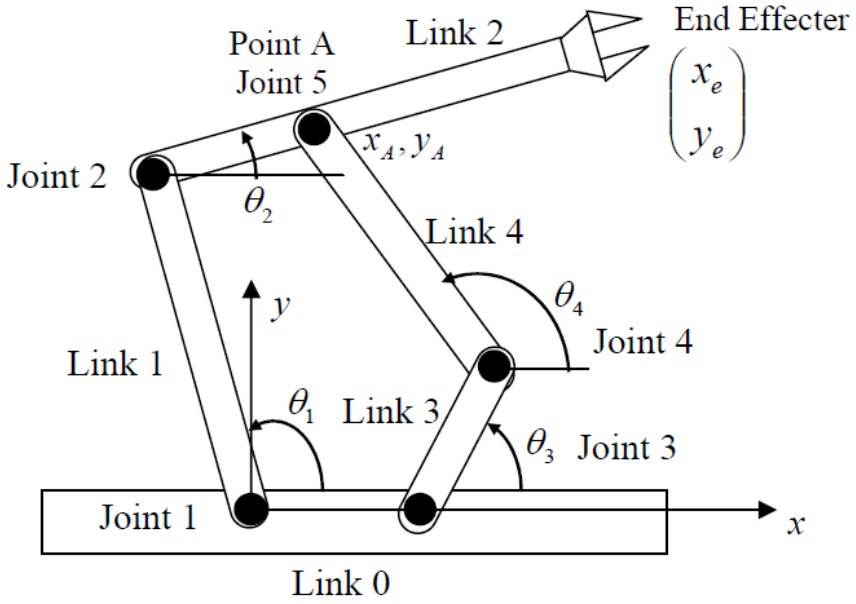


Figure 3.5 Five-bar-link mechanism

Although the forward kinematic equations are difficult to write out explicitly, the inverse kinematic equations can be obtained for this parallel link mechanism. The problem is to find θ_1, θ_3 that lead the endpoint to a desired position: x_e, y_e . We will take the following procedure:

Step 1 Given x_e, y_e, find θ_1, θ_2 by solving the two-link inverse kinematics problem.

Step 2 Given θ_1, θ_2 ,obtain x_A, y_A. This is a forward kinematics problem.

Step 3 Given x_A, y_A, find θ_3, θ_4 by solving another two-link inverse kinematics problem.

Example 3.4 Obtain the joint angles of the dog's legs, given the body position and orientation.

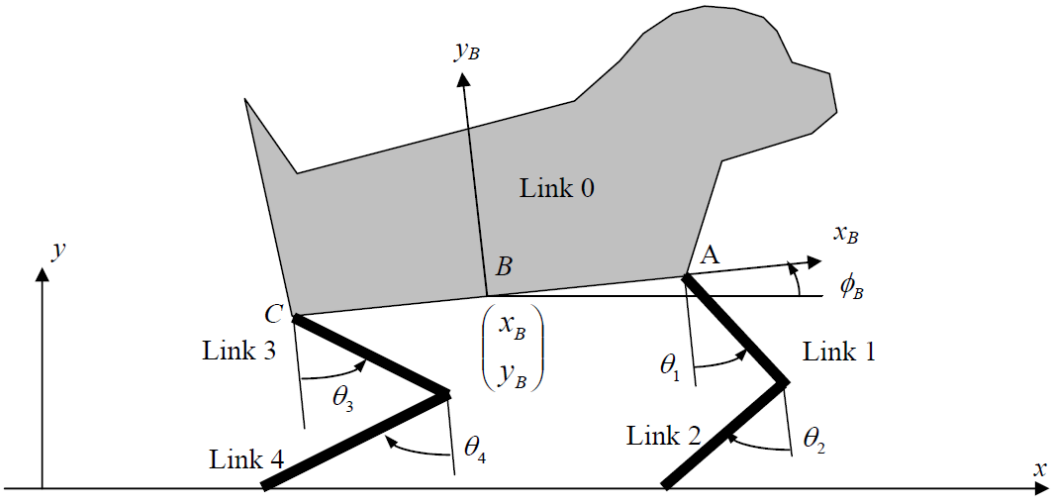


Figure 3.6 A doggy robot with two legs on the ground

The inverse kinematics problem:

Step 1 Given x_B, y_B, ϕ_B , find x_A, y_A and x_C, y_C

Step 2 Given x_A, y_A, find θ_1, θ_2

Step 3 Given x_C, y_C , find θ_3, θ_4

3.4 Redundant mechanisms

A manipulator arm must have at least six degrees of freedom in order to locate its end-effecter at an arbitrary point with an arbitrary orientation in space. Manipulator arms with less than 6 degrees of freedom are not able to perform such arbitrary positioning. On the other hand, if a manipulator arm has more than 6 degrees of freedom, there exist an infinite number of solutions to the kinematic equation. Consider for example the human arm, which has seven degrees of freedom, excluding the joints at the fingers. Even if the hand is fixed on a table, one can change the elbow position continuously without changing the hand location. This implies that there exist an infinite set of joint displacements that lead the hand to the same location. Manipulator arms with more than six degrees of freedom are referred to as *redundant manipulators.*

Chapter 4 Dynamics of manipulation robots

Dynamics is the branch of classical mechanics that deals with forces and/or torques required to cause the motion of a system of bodies.

The manipulator is represented as open-loop kinematical and dynamic circuit from several links, connected by rotary or prismatic joints. The generalized forces $\mathbf{f}_i$ and torques $\boldsymbol{\tau}_i$, that provide the required force f and torque t in the gripper must be considered.

Primary dynamical problem is: given a set of actuated joint torque and/or force functions ($\mathbf{f}_i$ or $\boldsymbol{\tau}_i$) it is necessary to calculate the resulting motion of the gripper in terms of generalized coordinates, speeds and accelerations as functions of time.

Inverse dynamical problem is: given a trajectory of the gripper in terms of generalized coordinates, speeds and accelerations as functions of time it is necessary to find a set of actuated joint torque and/or force functions ($\mathbf{f}_i$ or $\boldsymbol{\tau}_i$) which will produce this motion.

There are three methods of solution:

– method of connected graphs,

– Newton – Euler method,

– Lagrange – Euler method.

Beside these, there are also recurrent approaches: recurrent method of Newton-Euler and recurrent method of Lagrange. They provide smaller time of calculations.

We'll consider Lagrange–Euler method and Newton-Euler method.

4.1 Lagrange – Euler method of inverse dynamics solution

Denavit–Hartenberg notation will be used. This method is based on the following equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right) = \tau_i, \quad (2)$$

where L is Lagrange function ($L = T - V$); T is kinetic energy of system; V is the potential energy; q_i are generalized coordinates (θ_i); τ_i are generalized forces and torques.

The motion of every link of manipulator can be described in 6-dimensional coordinate system $x_i = [p_x, p_y, p_z, \theta, \phi, \psi]$, where $p = [p_x, p_y, p_z]^T$ is the vector of coordinate origin of this link, and θ, φ, ψ are Eulerian angles. They connect n^{th} coordinate system of considered link with the base coordinate system (x_0, y_0, z_0). Then generalized coordinates are

$q_m = q_m(x_1, x_2 \ldots x_6, t)$, or $x_i = x_i(q_1, q_2 \ldots q_N, t)$,

where *t* is time.

It is necessary to note that unambiguous solution of inverse dynamics is possible only for the manipulator with 6 DOF. It means that $N = 6$.

Let's consider the body in the free falling. Its kinetic energy is equal to $\frac{1}{2}mV^2$, and the potential energy is *mgh*. Here *V* is the derivative from the coordinate *h*, that is $\frac{1}{2}mV^2 = \frac{1}{2}m\dot{h}^2$. Now let's compose the Lagrange equation:

$$\left(\frac{\partial L}{\partial \dot{q}_i}\right) = \frac{\partial}{\partial \dot{h}}\left(\frac{1}{2}m\dot{h}^2 - mgh\right) = 2\frac{1}{2}m\dot{h};$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) = \frac{d}{dt}\left(m\dot{h}\right) = m\ddot{h};$$

$$\left(\frac{\partial L}{\partial q_i}\right) = \frac{\partial}{\partial h}\left(\frac{1}{2}m\dot{h}^2 - mgh\right) = mg.$$

We know that $\ddot{h} = a$ - acceleration and *mg* is the gravity force. Since generalized torques and forces equal zero (no forces or torques act on the body), then the result will be:

$ma = mg$.

It is the classical formulation of second Newton's law.

Speeds and accelerations

The coordinates x_i in the link coordinate system can be converted into generalized Cartesian coordinates $x_i(j)$

$$x_{i(j)} = x_{i(j)}(q_1, q_2 \ldots q_N, t), \quad i, j = 1, 2 \ldots N$$

Then the speed is determined as following:

$$\dot{x}_{i(j)} = \sum_{m=1}^{N} \frac{\partial x_{i(j)}}{\partial q_m} \dot{q}_m + \frac{\partial x_{i(j)}}{\partial t}$$

and the acceleration

$$\ddot{x}_{i(j)} = \sum_{m=1}^{N} \left[\frac{\partial x_{i(j)}}{\partial q_m} \right] \ddot{q}_m + \sum_{m=1}^{N} \sum_{n=1}^{N} \left[\frac{\partial x^2_{i(j)}}{\partial q_m \partial q_n} \right] \dot{q}_m \dot{q}_n + 2 \sum_{m=1}^{N} \left[\frac{\partial x^2_{i(j)}}{\partial q_m \partial t} \right] \dot{q}_m + \frac{\partial x^2_{i(j)}}{\partial t^2}$$

Example Find components of speed and of acceleration for coordinate origin of gripper that works in cylindrical coordinate system.

Solution The coordinate system of gripper can be connected with Cartesian system as following:

$$x = r\cos\theta, \;\; y = r\sin\theta, \;\; z = z, \; r = (x^2 + y^2)^{1/2}, \; \theta = arctg(y / x)$$

Components of speed are:

$$\dot{x} = \dot{r}\cos\theta - r\dot{\theta}\sin\theta$$

$$\dot{y} = \dot{r}\sin\theta + r\dot{\theta}\cos\theta$$

Components of acceleration are:

$$\ddot{x} = \ddot{r}\cos\theta - \ddot{r}\dot{\theta}\sin\theta - \ddot{r}\dot{\theta}\sin\theta - r\ddot{\theta}\sin\theta - r\dot{\theta}\cos\theta =$$

$$= \cos\theta(\ddot{r} - r\dot{\theta}^{\,2}) - \sin\theta(\ddot{r}\,2\dot{r}\dot{\theta} + r\ddot{\theta})$$

$$\ddot{y} = (\ddot{r} - r\dot{\theta}^2\sin\theta - = (r\ddot{\theta} + 2\dot{r}\dot{\theta})\cos\theta$$

Distribution of robot mass and inertia tensor

When there is a rotation, then it is necessary to know the distribution of robot mass relatively to all axes of rotation. Then inertia tensor is used

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix},$$

where for robot with discrete distribution of mass:

$$I_{xx} = \sum_{i=1}^{N} m_i (z_i^2 + y_i^2), \;\; I_{yy} = \sum_{i=1}^{N} m_i (x_i^2 + z_i^2), I_{zz} = \sum_{i=1}^{N} m_i (x_i^2 + y_i^2); \quad (3)$$

$$I_{xy} = -\sum m_i x_i y_i, \; I_{xz} = -\sum m_i x_i z_i, \; I_{yz} = -\sum m_i y_i z_i$$

for robot with continuous distribution

$$I_{xx} = \iiint \rho(z^2 + y^2)dxdydz,\ I_{yy} = \iiint \rho(x^2 + z^2)dxdydz,\ I_{zz} = \iiint \rho(x^2 + y^2)dxdydz;$$

$$I_{xy} = -\iiint \rho\, xydxdydz, I_{yz} = -\iiint \rho yzdxdydz, I_{xz} = -\iiint \rho xzdxdydz.$$

where ρ is the density of link.

Example Find the inertia tensor for represented link of manipulator (fig. 1).

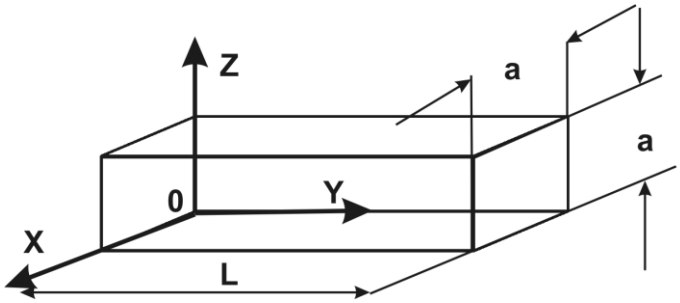


Figure 4.1

Solution

$$I_{xx} = \int_0^a \int_0^L \int_0^a (y^2 + z^2)\rho dxdydz = \int_0^a \int_0^L \rho a(y^2 + z^2)dydz = \int_0^a \left(\frac{1}{3}L^3 + z_2 L\right) a\rho dz$$

$$I_{xx} = \rho\left(\frac{a^2L^3}{3} + \frac{a^4L}{3}\right) = \frac{m}{3}\left(L^2 + a^2\right)$$

Speed of links

Let's the vector r_0^i determines the position of point P relatively the base coordinate system (x_0, y_0, z_0). The vector $\mathbf{r}_i$ will define the position of the same point, but in i^{th} coordinate system (fig. 2). Therefore, the speed of point P in the base coordinate system can be determined as

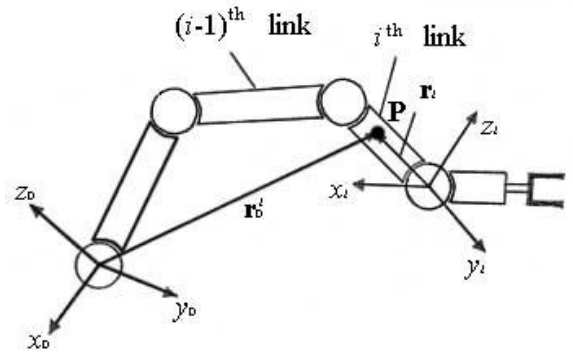


Fig. 4.2 Geometry of two-dimensional manipulator

$$v_i = \frac{d}{dt}\left(r_0^i\right).$$

Now it is time to remember that vector r_0^i can be rewritten using Denavit-Hartenberg notation in the following way:

$r_0^i = A_0^i r_i \quad or \quad r_{i-1}^i = A_{i-1}^i r_i.$

Thus we will obtain

$$v_0^i = v_i = \frac{d}{dt}\left(A_0^i r_i\right) = \sum_{j=1}^{i}\left(\frac{\partial A_0^i}{\partial q_i}\frac{dq_i}{dt}\right) r_i\,,$$

where q_i are independent variables θ_i. We can rewrite the last expression

$$v_0^i = \sum_{j=1}^{i}\left(\frac{\partial A_0^i}{\partial q_i}\dot{q}_i\right) r_i, \quad i = 1,\ 2\ ,\ ...,\ n.$$

Now we can show that

$$\frac{\partial A_{i-1}^i}{\partial q_i} = Q_i A_{i-1}^i,$$

where $\mathbf{Q}_i$ is the constant matrix. For example, for $\mathbf{A}_{i-1}^i$ we will have

$$\frac{\partial A_{i-1}^i}{\partial \theta_i} = \begin{bmatrix} -\sin\theta_i & -\cos\alpha_i\cos\theta_i & \sin\alpha_i\ \cos\theta_i & \text{-}a_i\sin\theta_i \\ \cos\theta_i & -\cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta_i & -\cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i\cos\theta_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the rotary joints the matrix $\mathbf{Q}_i$ will be $\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ and for prismatic joints the matrix $\mathbf{Q}_i$ will be $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

Let's define the value D_{ij} as $\frac{\partial A_0^i}{\partial q_j} = \begin{cases} A_0^{j-1} Q_j A_{j-1}^i, & j \le i, \\ 0, & j > i. \end{cases}$. Then it is easy to write the derivatives of higher orders

$$\frac{\partial D_{ij}}{\partial q_k} = D_{ijk} = \begin{cases} A_0^{j-1} Q_j A_{j-1}^{k-1} Q_k A_{k-1}^{i}, & i \geq k \geq j, \\ A_0^{k-1} Q_k A_{k-1}^{j-1} Q_j A_{j-1}^{i}, & i \geq j \geq k, \\ 0, & i < j \ \ or \ \ i > k. \end{cases}$$

4.2 Kinetic energy of manipulator

The kinetic energy *dT* of body with the mass *dm* is equal to

$$dT = \sum_{i=1}^{n} dT_i,$$

where dT_i is the kinetic energy of i^{th} link of manipulator. It can be found from the following expression:

$$dT_i = \frac{1}{2} tr\left(v_i v_i^T\right) dm_i.$$

where 'tr' designation means the trace operation, that is, the sum of the diagonal elements of matrix.

Let's consider the product in brackets:

$$v_i v_i^T = \begin{bmatrix} v_{1i} \\ v_{2i} \\ v_{3i} \\ 0 \end{bmatrix} \begin{bmatrix} v_{1i} & v_{2i} & v_{3i} & 0 \end{bmatrix} = \begin{bmatrix} v_{1i}^2 & v_{1i}v_{2i} & v_{1i}v_{3i} & 0 \\ v_{2i}v_{1i} & v_{2i}^2 & v_{2i}v_{3i} & 0 \\ v_{3i}v_{1i} & v_{3i}v_{2i} & v_{3i}^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Then from the expressions (6) and (7) we can write down:

$$dT_i = \frac{1}{2} tr\left[\sum_{p=1}^{i} D_{ip} \dot{q}_p r_i \left(\sum_{r=1}^{i} D_{ir} \dot{q}_r r_i \right)^T \right] dm_i$$

or

$$dT_i = \frac{1}{2} tr\left[\sum_{p=1}^{i} \sum_{r=1}^{i} D_{ip} \left(r_i dm_i r_i^T \right) D_{ir}^T \dot{q}_p \dot{q}_r \right].$$

Now we will integrate both parts of this expression:

$$T_i = \int dT_i = \frac{1}{2} tr\left[\sum_{p=1}^{i} \sum_{r=1}^{i} D_{ip} \left(\int r_i r_i^T dm_i \right) D_{ir}^T \dot{q}_p \dot{q}_r \right].$$

The integral component in brackets is the matrix of inertia of i^{th} link relatively the origin of joint coordinate system *i*:

$$J_i = \int r_i r_i^T dm_i = \begin{bmatrix} \int x_i^2 dm_i & \int x_i y_i dm_i & \int x_i z_i dm_i & \int x_i dm_i \\ \int x_i y_i dm_i & \int y_i^2 dm_i & \int y_i z_i dm_i & \int y_i dm_i \\ \int z_i x_i dm_i & \int y_i z_i dm_i & \int z_i^2 dm_i & \int z_i dm_i \\ \int x_i dm_i & \int y_i dm_i & \int z_i dm_i & \int dm_i \end{bmatrix}.$$

Let's express $\mathbf{J}_i$ in terms of inertia tensor $\mathbf{I}$ in the following way:

$$J_i = \begin{bmatrix} \left(-I_{xx} + I_{yy} + I_{zz}\right)/2 & I_{xy} & I_{xz} & m_i \bar{x}_i \\ I_{xy} & \left(I_{xx} - I_{yy} + I_{zz}\right)/2 & I_{yz} & m_i \bar{y}_i \\ I_{xz} & I_{yz} & \left(I_{xx} + I_{yy} - I_{zz}\right)/2 & m_i \bar{z}_i \\ m_i \bar{x}_i & m_i \bar{y}_i & m_i \bar{z}_i & m_i \end{bmatrix},$$

where $\bar{x}_i$, $\bar{y}_i$, $\bar{z}_i$ are coordinates of center of mass for i^{th} link in i^{th} coordinate system. Thus, the total kinetic energy of manipulator equals

$$T = \sum_{i=1}^{n} T_i = \frac{1}{2} \sum_{i=1}^{n} \sum_{p=1}^{i} \sum_{r=1}^{i} \left[tr\left(D_{ip} J_i D_{ir}^T\right) \dot{q}_p \dot{q}_r \right].$$

4.3 Potential energy of manipulator

The total potential energy V of manipulator that depends on the weight of robot can also be determined as the sum of all potential energies of separate links:

$$V = \sum_{i=1}^{n} V_i = \sum_{i=1}^{n} \left[-m_i g \left(A_0^i \bar{r}_i \right) \right],$$

where $\bar{r}_i$ is the position vector of center of mass of i^{th} link defined in i^{th} coordinate system, g is the gravitational acceleration.

Lagrange function

Lagrange function is defined as the difference between kinetic and potential energies:

$L = T - V$

or

$$L = \frac{1}{2} \sum_{i=1}^{n} \sum_{p=1}^{i} \sum_{r=1}^{i} \left[tr\left(D_{ip} J_i D_{ir}^T\right) \dot{q}_p \dot{q}_r \right] + \sum_{i=1}^{n} m_i g \left(A_0^i \bar{r}_i \right).$$

Let's differentiate Lagrange function in order to substitute it into Lagrange function:

$$\frac{\partial L}{\partial \dot{q}_k} = \frac{1}{2}\sum_{i=1}^{n}\sum_{p=1}^{i}\sum_{r=1}^{i}\left[tr\left(D_{ip}J_i D_{ir}^T\right)\left(\dot{q}_p\delta_{jk} + \dot{q}_j\delta_{kp}\right)\right] =$$

$$= \sum_{i=k}^{n}\sum_{p=1}^{i} tr\left[D_{ip}J_i D_{ik}^T\right]\dot{q}_p,$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) = \sum_{i=k}^{n}\sum_{p=1}^{i} tr\left[D_{ip}J_i D_{ik}^T\right]\ddot{q}_p +$$

$$+\sum_{i=k}^{n}\sum_{p=1}^{i}\sum_{m=1}^{i} tr\left[D_{ipm}J_i D_{ik}^T\right]\dot{q}_p\dot{q}_m + \sum_{i=k}^{n}\sum_{p=1}^{i}\sum_{m=1}^{i} tr\left[D_{ikm}J_i D_{ip}^T\right]\dot{q}_p\dot{q}_m.$$

In much the same way we can write

$$\frac{\partial L}{\partial q_k} = \frac{1}{2}\sum_{i=k}^{n}\sum_{j=1}^{i}\sum_{p=1}^{i} tr\left(D_{ijk}J_i D_{ip}^T\right)\dot{q}_j\dot{q}_p + \frac{1}{2}\sum_{i=k}^{n}\sum_{j=1}^{i}\sum_{p=1}^{i} tr\left(D_{ipk}J_i J_i D_{ij}^T\right)\dot{q}_j\dot{q}_p +$$

$$+\sum_{i=p}^{n} m_i g D_{ik}\bar{r}_i.$$

Lagrange equation

Lagrange equation in vector form is the following:

$$\tau = \mathbf{D}(\theta)\ddot{\theta} + \mathbf{H}(\theta,\ \dot{\theta}) + \mathbf{G}(\theta), \qquad (10)$$

where **D** is a matrix of mass $n \times n$, **H** is a vector ($n \times 1$) of centrifugal and Coriolis components, **G** is a vector ($n \times 1$) of gravitational components.

Then Lagrange equation can be represented as following:

$$\tau_i = \sum_{k=1}^{n} M_{ik}\ddot{q}_k + \sum_{k=1}^{n}\sum_{m=1}^{n} M_{ikm}\dot{q}_k\dot{q}_m + M_i,$$

where M_i, M_{ik}, M_{ikm} are generalized matrices of mass

$$M_i = \sum_{j=i}^{n}\left(-m_j g D_{ji}\bar{r}_j\right),$$

$$M_{ik} = \sum_{j=\max(i,k)}^{n} tr\left(D_{jk}J_j D_{ji}^T\right),$$

$$M_{ikm} = \sum_{j=\max(i,k,m)}^{n} tr\left(D_{jkm}J_j D_{ji}^T\right).$$

Lagrange equation can be expanded for this very case (fig. 4.1):

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix}=\begin{bmatrix}M_{11} & M_{12}\\ M_{21} & M_{22}\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix}+\begin{bmatrix}\sum_{k=1}^{n}\sum_{m=1}^{n}M_{1km}\dot{\theta}_k\dot{\theta}_m\\ \sum_{k=1}^{n}\sum_{m=1}^{n}M_{2km}\dot{\theta}_k\dot{\theta}_m\end{bmatrix}+\begin{bmatrix}M_1\\ M_2\end{bmatrix}.$$

Let's write this equation in expanded form:

$$M_{11}=tr\left(D_{11}J_1D_{11}^T\right)+tr\left(D_{21}J_2D_{21}^T\right),$$

$$M_{12}=M_{21}=tr\left(D_{22}J_2D_{21}^T\right),$$

$$M_{22}=tr\left(D_{22}J_2D_{22}^T\right),$$

$$M_1=m_1gD_{11}\bar{r}_1+m_1gD_{21}\bar{r}_2,$$

$$M_2=m_2gD_{22}\bar{r}_2.$$

Chapter 5 Information systems in robotics

Introduction

Along with matter and energy, ***information*** is the primary concept of our world and therefore cannot be defined in the strict sense. We can only list its main properties, for example:

1) information brings information about the surrounding world, which at the point in question was not before receiving it;

2) information is not material, but it manifests itself in the form of material carriers of discrete characters or primary signals;

3) the signs and primary signals carry information only for the recipient who is able to recognize it.

Information based on the unambiguous connection of signs or signals with real-world objects is called semantic or semantic. The information contained in the nature (order and relationship) of the following signs of the communicating, is called syntactic. Also in the general science of signs (semiotics), besides the listed ones, the sigmatic and pragmatic aspects of information are distinguished. In the first case we study the question of the choice of signs to denote real world objects, the second - on the value of information to achieve the objectives.

5.1 General information from the theory of information

Most of the information we use is communicated through one or another "language" that obeys certain statistical patterns. In the simplest case, when messages are written using ***n*** symbols x_i, the relative frequencies of occurrence of which are mutually independent and are completely determined by the prior probability $P(x_i)$, it is true

$$\sum_{i-1}^{n} P(x_i) = 1$$

The main properties of the information include:

1. The amount of information regarding an event x_i delivered by an event y_j is defined as the logarithm of the ratio of the a posteriori probability $P(x_i, y_j)$ to the a priori $p(x_i)$:

$$I(x_i, y_i) = \log \frac{P(x_i / y_j)}{p(x_i)}.$$

2. The measure of the amount of information is a symmetric function with respect to x_i and y_j:

$$I(x_i, y_i) = I(y_j, x_i).$$

The information delivered by the event y_j relative to the event x_i is equal to the information delivered by the event x_i relative to the event y_j.

3. With a fixed probability, mutual information $I(y_j, x_i)$ reaches a maximum when $P(x_i / y_j) = 1$ when y_j it reliably uniquely identifies x_i:

$$I(x_i) = -\log p(x_i).$$

4. Mutual information satisfies the additivity conditions:

$I(x_i, y_j, z_k) = \log \frac{P(x_i / y_j z_k)}{P(x_i)}$, i.e. information I is obtained with respect to x_i

with the compatibility of the observation of two events x_i and y_j.

The process of converting a message into a combination of characters according to a code is called ***encoding***; the process of recovering a message from a combination of characters is called ***decoding***. Code is a universal way of displaying information during its storage, transmission and processing.

The final sequence of characters a_j is called a word in a given alphabet. Each word in the code is called a code word (code combination). There are uniform, non-uniform, direct, inverse, additional codes.

Uniform codes, these are codes in which all combinations have the same length. For a uniform code, the number of possible combinations is m^n. An example of such a code is the five-digit Bodo code containing five binary elements

$(m=2, n=5)$. The number of possible code combinations is equal to $2^5 = 32$, which is enough to encode all the letters of the alphabet. The use of uniform codes does not require the transfer of separation characters between code combinations.

Non-uniform codes are characterized by the fact that their code combinations differ from each other not only in the mutual arrangement of symbols, but also in their number. This leads to the fact that different combinations have different duration. A typical example of non-uniform codes is Morse code, in which the characters 0 and 1 are used only in two combinations - as single (1 and 0) or as triple (111 and 000). The signal corresponding to one unit is called a point, three units - a dash. The symbol 0 is used as a sign separating the point from the dash, the point from the point and the dash from the dash. Aggregate 000 is used as a separator between code combinations.

Direct code - a binary number code that matches the image with the record of the number itself. The value of the sign bit for positive numbers is 0, and for negative numbers - 1. The sign bit is usually the extreme bit in the bit grid. In some cases, when writing code, the sign bit is separated by a comma. For example, in the case where one byte is allocated to the code entry, the direct code is 0.0001101 for the +1101 number, the direct code is 1.0001101 for the –1101 number.

The reverse code for a positive number is the same as the direct code. For a negative number, all digits of the number are replaced with the opposite ones (1 to 0, 0 to 1), and one is entered in the sign bit. For example, for the number +1101, the direct code is 0.0001101; the return code is 0.0001101. For the number –1101, the direct code is 1,0001101; the reverse is 1,1110010.

The additional code of a positive number is the same as the direct code. For a negative number, an additional code is formed by obtaining a reverse code and adding 1 to the lower order.

5.2 The concept of a signal. Classes and types of signals

Signals of the most diverse types are widely used in everyday life; therefore, the already intuitive concept of a signal has a rather specific content. Nevertheless,

this concept should be considered in more detail and give definitions characterizing the signal from different positions and covering all types of signals.

A signal is a display of ***a message***; the signal is a material carrier of information. Whatever the specific signal - sound, light or radio signal, book, record or movie - the whole point of creating this signal lays in the display of certain information. In the end, any information, and consequently, any signal, is addressed to the recipient and is of some value only if the recipient exists (or possibly exists). The sender and receiver are always separated by space or time; signals provide communication between them. Hence the addition (or, rather, an explanation) to the definition given above**:** ***a signal*** is a means of transferring information in space and time.

The above definitions cannot serve as a basis for the theory of the structure of signals, since they consider a signal from its service side and are not related to the structure of the signal. The infinite variety of signals, the equivalence of physically completely different representations of one message - all this requires a definition that answers the question: “What is a signal?”, I.e. a definition that considers a signal from the point of view of a person who is interested in a signal, not as an aid, but as an object of study.

Consideration of any situations in which signals are involved leads to the conclusion that, although the signal is always associated with a material object, most of the specific (physical, chemical, etc.) properties of this object are insignificant. In the end, it does not matter what sort of paper and ink composition the letter is written on; it differs from all other letters as a signal by the state of the distribution of color over the surface of the sheet.

When making a radio broadcast, a whole number of physically different objects are used to display the message: typewritten text of the broadcast - voice of the speaker - electromagnetic waves - current fluctuations in the electromagnet winding - loudspeaker sound - vibrations of the listener's ear drum - oscillatory processes in the listener's nerve. As links in this circuit, you can turn on sound recording and playback on a tape recorder, etc. The common thing that binds such a variety of objects is that

they all serve to form signals. In a certain sense, we can say that these objects themselves "serve as signals," but it is more essential that the same object (for example, an electromagnetic field) can carry different signals. Consequently, as signals, not the objects themselves are used, but their states. Signal formation is the change in the state of the object. This statement requires development, since, obviously, the reverse is not true: not every change in the state of an object is a signal. Impact on an object that changes its state, only then will lead to the formation of a signal when this impact is performed according to certain rules. The existence of such rules ensures consistency between the message and the signal. The existence of this correspondence, in turn, makes it possible to extract a message from the received signal. This possibility can be realized only if the rules for changing the state of the object (that is, the rules of signal formation) are known to the party that received the signal, or are partially known, at least to the extent that, relying on this partial information and signal analysis, fully define these rules. Now we can give a refined definition: ***a signal is a change in the state of a material object produced according to predetermined rules (that is, using a predetermined code).***

Since the signals are used to transfer information in space and time, only those objects can be used to form signals, the states of which are sufficiently stable with respect to a change in time or position in space. Quantitative requirements for stability are imposed in accordance with the specific conditions of use of the signal.

In terms of stability, all signals can be divided into two classes.

The first class includes signals, which are used as stable, stable states of physical systems. Examples of signals of this type can be: a book, a photographic image, the state of a tape recorder film, the state of a ferrite matrix of computer memory, a register state (trigger system) of a computer, the position of a railroad semaphore rod, the location of a triangulation tower, etc. Such signals are called ***static signals.***

In the ***second class***, the signals are combined, which use the dynamic states of the force fields. As mentioned in the previous paragraph, the signal occurs when the state of an object changes. In contrast to other material systems, the fields are

characterized by the fact that a change in their state cannot be localized in the (non-isolated) part of the field and leads to the propagation of a disturbance. When a disturbance propagates in a field, the configuration parameters and the structures of this disturbance possess a certain stability, which makes it possible to use such field states as signals. Examples of such signals are: sound signals (change in the state of the field of elastic forces in a gas, liquid, or solid), light and radio signals (changes in the state of the electromagnetic field). Let us call the signals of the second class ***dynamic signals.***

Due to the characteristic difference between dynamic and static signals, their practical use is also different. Dynamic signals are used primarily for transmission, and static - for storing information. However, these functions cannot be fully separated. Dynamic signals can be used to store information, as is the case, for example, in storage devices on ultrasonic delay lines of electronic digital computers. In a sense, it can be said that such static signals as newspapers and letters are more intended to be transmitted than to store information.

Despite the huge variety of signals, by the method of generating them and extracting information from them at the receiving end (ie, by encoding and decoding), all signals are divided into three large groups.

The first group includes signals that can be called communication signals, or ***direct signals***. Such signals include signals used in the telegraph, telephone, television, telecontrol, acoustic and light communications, written and printed letter signals, etc. The characteristic features of this group of signals are that, first, the sender and receiver of the signal are always present and the signal is intended to transmit information from the first to the second; secondly, the code is fully known to both parties; thirdly, in the part that does not affect the conditions for the existence of a signal, the code is conditional, i.e. is built by agreement of the advising parties and by agreement can be changed.

The second group is formed by the signals with which the measurements are made - ***the signals for measurements***. Measuring a certain value is comparing it with the corresponding standard, therefore, when measuring, there are always two signals:

the reference and the comparison with it. In some situations (for example, in radiolocation) the signal to be compared is a reference signal ("probing") changed during the propagation process; in other cases (for example, when measuring the length with a ruler), the compared signal exists independently of the reference signal. The peculiarity of the reference signal is that everything is known about it, and he, therefore, does not carry any information; for convenience of comparing a wide class of signals with a standard, the latter is usually periodic, although this is not necessary.

The third group can be attributed to the so-called natural signals. Signals always appear as states of physical objects. In a certain sense, one can say that any state of any physical object can be considered as a signal even if bringing this object into this state is not at all connected with the transfer of any information, but occurred due to natural causes. We can say that we have before us a "signal with a not completely known code".

The signal is always a function of time. Depending on what values the argument can take (time t) and signal levels, they are divided into 4 types.

1. ***Continuous, or analog, signals*** (Fig. 6.1) are defined for all points in time and can take all values from the specified range. Most often, the physical processes that generate the signals are continuous. This explains the second name of the signals of this type - analog, i.e. similar to generating processes (random signals of this type are called continuous random processes).

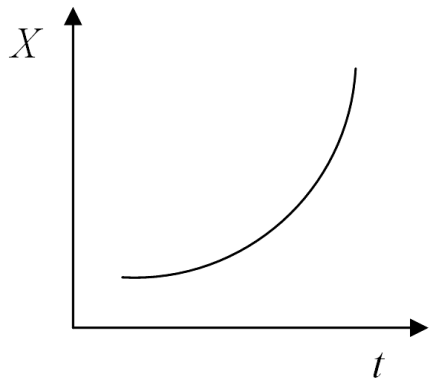


Fig. 1.1. Continuous, or analog, signal

2. ***Discrete, or discretely continuous,*** signals (Fig. 6.2) are determined only at certain points in time and can take any level values. The time interval Δt between adjacent samples is called the sampling step. Often such signals are called discrete in

time (random signals of this type are called processes with discrete time or continuous random sequences).

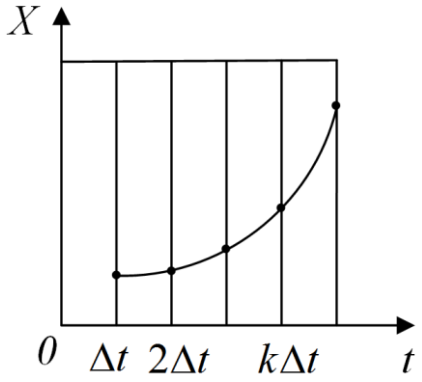


Fig. 1.2. ***Discrete, or discretely continuous,*** signal

3. ***Level discrete, or quantized, signals*** (Fig. 6.3) are defined for all points in time and take only the allowed values of the levels, separated from each other by the quantization step size $\Delta x = x_{k+1} + x_k$ (random signals of this type are called discrete random processes). The signal conversion operation in which it is discretized by level or by time or simultaneously by level and time is called quantization.

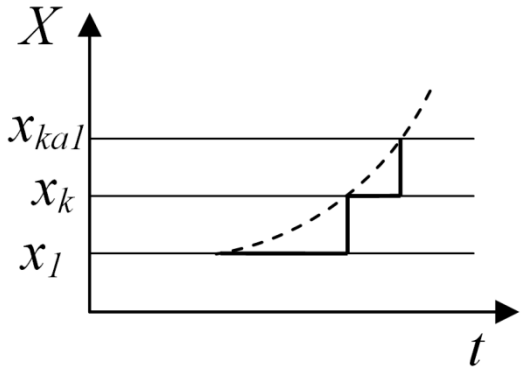


Fig. 1.3. Level discrete or quantized signals

Time quantization - the transformation of a signal into a sequence of successive pulses whose amplitude, duration or frequency depend on the amplitude of the input signal.

Level quantization - the signal transformation, which consists in surrounding the instantaneous value to some nearest pre-determined, fixed value, is called a level.

4. ***The discrete by level and by time signals*** (Fig. 1.4) are determined at individual allowed points in time and can take only the allowed values of the levels (random signals of this type are called discrete random sequences).

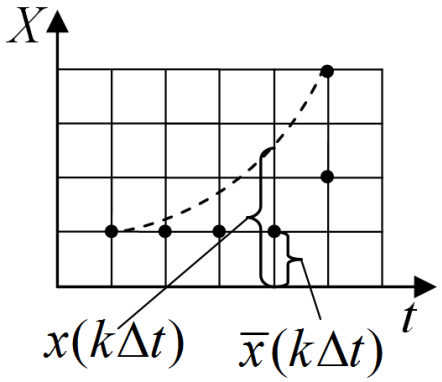


Рис. 1.4. The discrete by level and by time signals

Digital signal processing, DSP, - the conversion of signals presented in digital form. Any continuous (analog) signal can be subjected to time sampling and level quantization (digitization), i.e. presented in digital form. If the signal sampling frequency is higher than twice the highest frequency in the signal spectrum $F_{\max}$, i.e. $F_d > 2 \cdot F_{\max}$ then the obtained discrete signal *s(k)* is equivalent to the signal *s(t)* (see the Kotelnikov theorem).

With the help of mathematical algorithms, *s(k)* is converted into some other signal $s_1(k)$, having the required properties. The process of converting signals is called ***filtering***, and the device that performs filtering is called a ***filter***. Since the signal samples arrive at a constant speed F_d, the filter must have time to process the current sample before the next one arrives, i.e. process the signal in real time. For signal processing (filtering) in real time, use special computing devices - digital signal processors.

There are methods of signal processing in the time and frequency domain. The equivalence of the time-frequency transformations is uniquely determined by the Fourier transform.

Filters are:

- continuous, discrete, linear and nonlinear;
- electrical, mechanical, acoustic, etc.

The main tasks of filtering:

– ***linear filtering*** - selection of the signal in the frequency domain; synthesis of filters matched with signals; frequency division channels; Hilbert's digital converters and differentiators; channel correctors;

– ***spectral analysis*** - processing of speech, sound, seismic, hydroacoustic signals; pattern recognition;

– ***time-frequency analysis*** - image compression, hydro- and radar, various detection tasks;

– ***adaptive filtering*** - processing of speech, images, pattern recognition, noise suppression, adaptive antenna arrays;

– ***nonlinear processing*** - calculation of correlations, median filtration; synthesis of amplitude, phase, frequency detectors, speech processing, vector coding;

– ***multi-speed processing*** - interpolation (increase) and decimation (decrease) of the sampling rate in multi-speed telecommunication systems, audio systems.

A general model of information system

Any (artificial or natural) system of interacting objects can be considered as an information system. Any part of a set of interacting objects (in particular, one of the objects) can be studied in order to extract information about another part of this set (in particular, about another individual object), since the interaction ensures that the states correspond, reflection, content information. The objects that make up the information system can have a completely arbitrary nature.

From this, of course, it does not follow that information theory is intended to replace or embrace other sciences that study the specific interactions between objects of a particular class. But from this it follows that among the infinite set of properties that are inherent in any system of interacting objects, an integral property is the property of objects to reflect each other, to contain information about each other. In some phenomena, informational relations do not play a significant role or are disguised - then the science that studies these phenomena can achieve certain success without using information theory; in other cases, the information approach is inevitable.

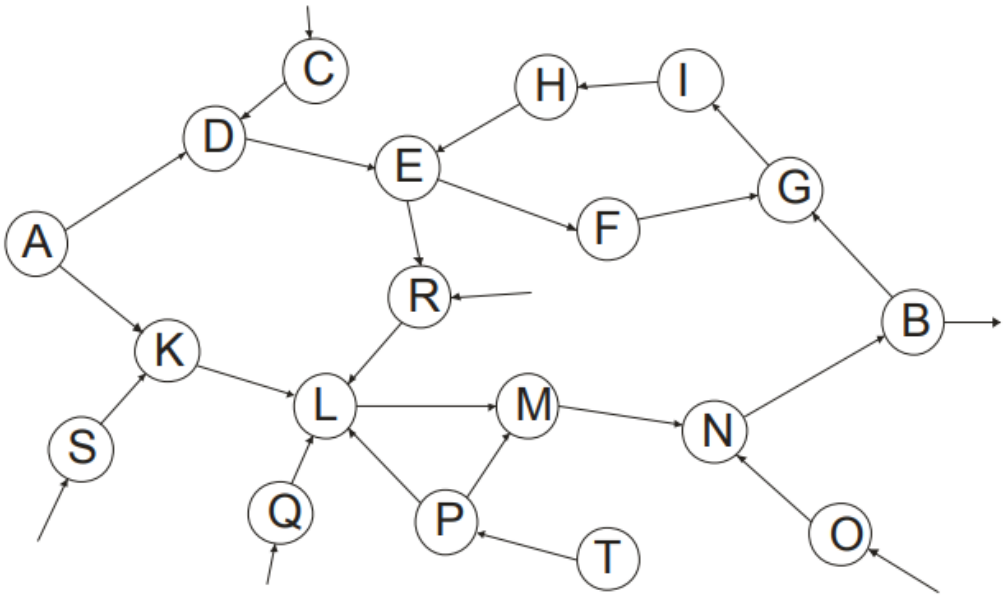


Fig. 6.5. Simplified object system layout

Suppose we have a system of objects of arbitrary nature, interconnected with each other. Of the many connections of a particular object with others, it is usually possible to single out only a few of the most significant ones, omitting the others from consideration. In this case, some complex system of objects can be simplified as shown in Fig. 6.5.

Essential connections between objects are depicted by arrows, the direction of which corresponds to the transition from cause to effect. Due to the presence of direct connections, object B, for example, contains information about objects O, G, N; connections through other objects provide content in the object of information about objects A, R, S, O, etc.

Typically, the recipient is interested in information about any one object, for example A, and object B is observed in order to extract this particular information. Information about the object of interest to the recipient is considered as useful, information about other objects appears to be unnecessary, useless and even harmful, since its presence may make it difficult to extract useful information. If the recipient does not have comprehensive information about the remaining objects, their influence should be considered as “interference” or “noise”. Thus, whenever the influence of an object that does not interest us violates the uniqueness of the correspondence between

the states of objects A and B, they say that interference occurs. Corresponding objects (for example, C, R, S, T, O, Q) are considered to be sources of interference.

If objects A and B do not interact directly, then the correspondence of their states is established due to the presence of chains of intermediate objects. There can sometimes be several such binding sequences of objects (D, E, F, G and K, L, M, N in Fig. 1.5); sometimes only part of the multiplet sequence (L, M and L, P, M). In these cases, they talk about multichannel, multipath or multipath systems.

Finally, in the structure of an information system there may exist closed sequences of objects that carry useful information (for example, E, F, G, I in Fig. 1.5). Such systems are commonly referred to as systems with feedback loops. Feedback loops can cover both several intermediate objects ("internal" loops) and the entire system, connecting the final and initial objects ("external" loops).

So, in any information system, objects of the following four types can be distinguished:

1. The initial object. The rest of the system is used to obtain information about this particular object. The initial object is often called the source of information.

2. The target object. Knowing the law of correspondence between the states of the initial and final objects and directly observing the latter, the recipient extracts information about the state of the former.

3. Intermediate, auxiliary objects. With the help of these objects the correspondence between the initial and final objects is established.

4. Objects, the interaction with which destroys the uniqueness of the correspondence of the states of the initial and final objects; sources of interference.

It should be noted that sometimes the separation of these types of objects can be carried out only conditionally. The simplest example is a real amplifier; essentially being an object of the third type, it is simultaneously a source of thermal noise. Another example is the communication line on tropospheric scattering. On the one hand, the presence of tropospheric inhomogeneities ensures the very existence of communication at a distance, on the other, the chaotic movements of the same inhomogeneities cause uncontrolled signal fading, making communication difficult.

However, for convenience of consideration, even such systems are artificially depicted as an equivalent combination of objects of these four types.

We emphasize once again that the same real-life information system may be qualitatively different for two observers with different information about this system. In order to extract information about object A, observing object B, it is necessary to know the law of correspondence of their states. If the observer does not know this law, the observation of the object cannot directly give him the necessary information, the whole system is in a qualitatively different state for him than for the observer who knows this law. To be convinced of the reality of such a situation, it is enough to imagine a distressed radioactive yacht, on which, after the radio operator's death, there were no people familiar with Morse code. Another example is a reader looking at a book written in an unfamiliar language.

The question of the classification of information systems according to their functional purpose or use cannot be considered sufficiently fully considered: there are many subtle differences between such systems. However, several types of information systems differ quite clearly: 1) communication systems, or information transfer systems, 2) information storage systems, 3) information processing (conversion) systems, 4) measurement systems, 5) observation or research systems.

5.3 Communication systems

Communication systems called information system, the main function of which is to transfer information in the space.

There are many types of communication systems, such as mail and broadcasting, telephone and telegraph, acoustic communication systems and signal cable of divers, etc. Technical communication systems deserve special consideration, in which dynamic signals are used to transfer information from one point to another. To excite a dynamic signal, a special transmitting system is created, and a receiving system is used to record the signal at the destination. The set of objects connecting the transmitting and receiving systems is called a communication line. For example, in telephone communication, a line is a pair of wires; in radio communication, a communication line is the space in which radio waves propagate.

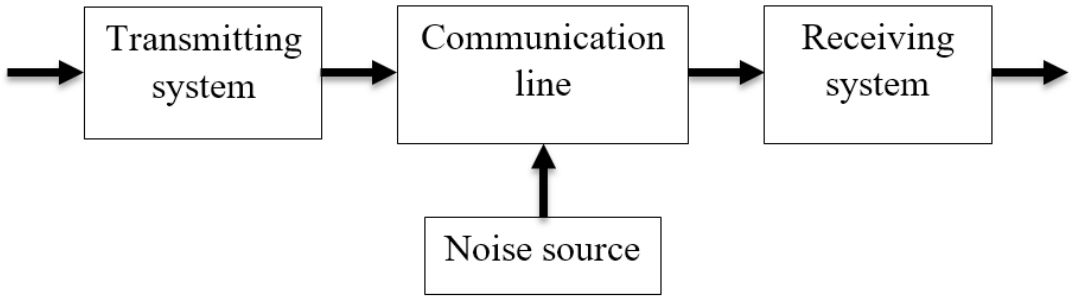


Fig. 6.6. Communication channel block diagram

If it is necessary to describe the potential capabilities of the communication system, it can be agreed not to take into account the specific features of the transmitter and receiver. In this case, the initial object in the communication system is considered the input device of the transmitting system, and the final object is the output signal of the receiving system.

Let's consider some types of technical communication systems that work with dynamic signals.

1. ***Single channel communication system***. A communication system designed for one-way transmission of information between two given points will be called a ***single-channel system*** or simply a ***communication channel***. The simplest block diagram of the communication channel is shown in Fig. 6.6 and contains, in addition to the transmitting and receiving systems and communication lines, source of noise, acting in general to all elements of the system. Noise arising (or created artificially) in a communication line is called external (in radio communications, for example, industrial interference, atmospheric and cosmic noise, noise from extraneous radio facilities, etc.). Internal noise is usually understood as noise arising in the transmitting and receiving systems (thermal noise of resistance, noise of electronic tubes, etc.). Under some conditions it is sometimes possible to neglect the effect of noise; Such a communication system is called a channel without noise. In other cases, the noise cannot be excluded from consideration. Usually, however, it is considered that internal noises can be either neglected compared to external ones, or that the system allows for "recalculation" of all sources into one equivalent with known characteristics, and this noise is carried to the communication line. It is advisable to distinguish between communication channels operating on continuous and discrete

signals. This refers to the discreteness of the channel in the sense that the sets of elementary symbols at the input and output of the communication line are discrete and finite. The problems that arise when considering discrete communication channels can be illustrated with the help of the diagram in fig. 1.7. It differs from the general scheme (see Fig. 1.6) by the details of both the transmitting and receiving systems: the transmitting system is represented by two encoding devices, and the receiving system - by two decoding devices.

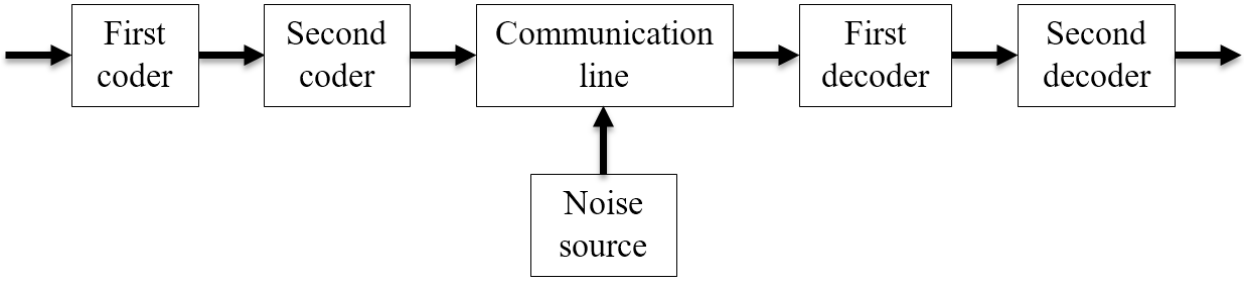


Fig. 1.7. The problems that arise when considering discrete communication channels

The input signal of the communication system can be printed text or a graphic image, sound wave, instrument readings, etc. The purpose of the first encoder is to present the input signal in some standard form, for example, as a sequence of binary symbols. The main problem of such a recoding is that the standard representation is the most economical, i.e. would require (on average) the smallest possible number of binary characters.

Due to the presence of interference in the communication line, the correspondence between the sent and received symbols ceases to be one-to-one. In the general case, attempts to determine by the received symbol which of the possible symbols was sent are inevitably associated with errors. However, it is possible to mitigate the effect of interference by suitable signal transcoding. In the simplest case, this possibility can be realized with the help of multiple repetition of the transmission and subsequent comparison of the received texts. Such a method, however, is applicable only with small probabilities of errors and, moreover, dramatically increases the required transmission time. There are ways to more efficient coding.

The purpose of the second encoder and the first decoder is to implement the chosen method of noise-resistant coding and decoding of the signal. The main problem with this is to minimize the probability of errors, although the interference randomly distorts the useful signal. Ideally, the output of the first encoder is the same as the input of the second decoder. So, the function of the second encoder is to represent the standard sequence of characters in the selected code, and the first decoder restores the signal again in a standard form according to the adopted sequence.

Finally, the function of second decoder is to restore input signal of the entire system;it is assumed that the standard representation of the signal was accurately determined by the first decoder.

2. ***Multichannel communication system.*** Quite often, it is necessary to transfer information from a group of close sources to a group of recipients concentrated in another location. A good example is the need to transfer data from various instruments installed on an artificial satellite to a group of ground-based recording devices. Another example is the telephone connection between two major cities. On the one hand, it is necessary to create a separate communication channel for each sender – receiver pair. On the other hand, economic considerations (for example, the high cost of building a wired communication line) or technical difficulties (for example, creating a separate communication line for each instrument on the satellite) prevent an increase in the number of separate communication lines for each pair of objects to be connected. The way out is to combine the channels by sending all the information through a single line of communication (if, of course, it allows for this). Such combined communication systems are called ***multichannel.***

In order for a signal to be addressed to a specific recipient and to be sent only to him, it is obviously necessary to provide the signals of various channels with some additional physical sign, a selection parameter by which filtering would be performed at the receiving end. Therefore, in the multi-channel communication system, additional devices appear to separate the signals belonging to different channels. For multichannel systems, a specific feature is that, as a result of the non-ideal separation,

the signals of adjacent channels somewhat distort the signal of this channel; This so-called crosstalk is usually the main type of interference in such systems.

3. ***Multi-path (multiray) communication channels***. In some cases, you have to use such communication channels, the signal in for some physical reasons is split into several components. At the same time, each of the signal components follows a separate path, undergoing specific transformation paths (changes, delay in time, and sometimes non-linear distortions). If it were possible at the receiving end to separate the signals that came in different ways, then we would have one of the communication systems discussed above. However, this possibility is often not available, and the receiver captures a signal that is the result of some total component impact, coming in different ways. Such communication systems are called multipath (multiray).

An example of a multipath communication system can serve as radio communication on short waves that propagate by the ground wave and by reflection from the ionosphere (sometimes undergoing multiple reflections from the ionosphere and the surface of the Earth). Another example is radio communication on scattering, in which the inhomogeneities of the troposphere or ionosphere can be considered as separate scattering centers emitting separate components of the signal.

4. ***Communication systems with random parameters***. The main difficulties in considering multipath channels arise not so much due to the need to take into account the ratios between the signal components, but rather due to the fact that it is necessary to take into account temporary changes in the conditions for passing components along different paths. These changes are usually random in nature, which causes a number of difficulties. To overcome these difficulties, a convenient means was to construct a model of a communication system with random parameters.

In some cases, consideration of multipath channels with random changes in the parameters of the signal components can be approached purely phenomenologically: the whole system is considered as one-channel, single-track, and all probabilistic properties are attributed to random changes in some imaginary parameters of such a channel. Such a model turns out to be useful not only when considering multipath

communication systems; there are systems that are directly displayed by such a model (for example, communication on meteor tracks), which makes its study even more important.

5. ***Complex communication systems.*** For some purposes it is necessary to construct integrated communication systems: duplicate channels to improve communication reliability; transmit, information on a sequence of channels with different properties; to construct networks with a complex interlacing of channels, etc. Such composite communication systems will be called complex. Complex communication systems have a number of specific features that must be considered when building and using.

Information Storage Systems

In the overwhelming majority of practical situations, the information available by the time t cannot or should not be used immediately; but usually there is confidence that this information will be needed in the future. To ensure the transfer of information in time and created a variety of information storage systems. Examples of such systems include a tape recorder and notebook, a library and storage devices of personal computers, an atlas of geographical maps, an art gallery, function tables, etc.

For technical information storage systems (which are of primary interest in the information-theoretic approach), the main characteristics are the information capacity (that is, the maximum amount of information the system is capable of storing) and the long-term storage of information without loss limits). When considering high-speed systems that include storage devices, an important parameter is the sampling time, i.e. the time interval between the moment of accessing the memory and the moment of obtaining the necessary information.

The variety of information storage systems is very large, even if you do not go into the technical differences between specific systems. Distinguish (by duration of storage) long-term and operational storage devices (memory); there are memories that can be accessed as many times as necessary, but there are memories that store information only until the first access to them. There are memory devices that can

only be accessed at fixed points in time (usually periodically repeated); There are chargers that can be accessed at arbitrary points in time.

With all the variety of information storage systems, several basic subsystems with different functional purposes can be distinguished in them. In addition to the information storage itself, there is an address system that provides searching for a number of features of the desired storage cell; input and output blocks provide the ability to request and issue data, as well as entering new information both by crowding out unnecessary ones and by filling in free cells.

The introduction of storage devices into complex information systems greatly expands the capabilities of the latter and, of course, significantly increases the difficulty of their research. Therefore systems with memory and systems without memory are usually considered separately.

Information converters

All kinds of operations occurring in information systems are not limited only to the transfer and storage of information. In many systems, the most significant is the processing of information, sometimes simple, sometimes very complex. For such processing are special devices - information converters. The transformation of information poses a number of complex problems, only a part of which is resolved by the theory of information to a sufficient degree fully.

Recoding is one of the relatively simple and at the same time most frequent transformations. We have already had the opportunity to discuss the functioning of the coding and decoding devices that make up the communication systems. Essentially, transcoding is an operation of transition from presenting some information in one code to presenting it in another code (code means the entire set of rules for generating a signal). This statement, being correct, is too general in nature and requires clarification. We will call transcoding such a conversion of one signal to another, in which the amount of information carried by the second signal is equal to the amount of information carried by the first. This, of course, does not mean that each element of the secondary signal carries the same load as the element of the

primary signal. Recoding can be done in such a way that one signal will contain more elements than the other.

By the principle of operation, all encoders can be divided into two classes. The first includes memoryless encoders, which transcode the signal elementwise, or instantly. A more difficult object to study is memory encoders. Their peculiarity is that each element of the output signal of a device is generally determined not by one element of the input signal, but by some set of such elements. Examples of such devices are devices that perform optimal coding for transmitting a signal over a noisy channel, devices for transmitting secret codes (excluding the simplest codes), as well as filters with finite bandwidth, etc.

Information converters, which have also been studied in relatively detail, include information storage devices. An integral part of the accumulation of information is its memorization, therefore, storage devices are sometimes considered as drives. However, accumulation can (or should) sometimes be made not by memorizing all input signals, but by memorizing the result of some processing of these signals. The accumulator is, for example, an adder, the output of which cannot unambiguously determine the terms. The accumulator is a device issuing its histogram by the input realization of a random process. The purpose with which drives are created is the accumulation of necessary information. If all the incoming information is needed, then the drive is simply a keeper of information, a storage device. If the input signals carry not only useful information, but also unnecessary, then the drive plays the role of a filter that selects and accumulates only what is necessary for further use; in this case, accumulation does not boil down to simple memorization.

Listing the typical systems of information transformation, mention should be made of the comparison devices, which establish the degree of similarity of the compared signals; solvers, i.e. systems that display the space of input signals to the solution space (control signals); quantizing devices that associate discrete mappings with continuous signals; filters that select signals based on certain characteristics, and a whole range of other information processing systems. Each of these systems is

associated with a number of information problems; first of all - the question of the optimality of such systems in the sense of minimal loss of useful information (or in some close to this sense).

Other types of information systems

The variety of information systems is very large, and, apparently, the issues of their classification will be discussed in the scientific literature. In addition to the three types of information systems discussed above, several more characteristic groups of systems can be distinguished, of which we briefly discuss measurement systems and research systems.

The main features of the measurement systems are associated with the features of the measuring signals. The measurement system includes: probes and reference signal generators; communication lines through which signals are sent and retracted from the measurement object; A system for comparing a signal from a measured object with a standard. The main problems in the design of measurement systems are: the selection of the most informative signals; reduction of noise (measurement errors), i.e. bringing to a possible minimum the impact of uncontrolled changes, not objects of interest to us; finding such data processing methods that would allow to extract the maximum available information.

Research systems are created to receive and decode signals whose code is not fully known. Such systems include devices for receiving and extracting information from natural signals, systems created for intercepting and deciphering the enemy's radiograms encoded with a secret "unshake-resistant" code, etc. One of the main points in the work of such systems is the promotion and testing of hypotheses regarding an unknown code. This procedure poses very difficult problems. After all, if a very large number of hypotheses can be put forward relative to an unknown code, then simply going through them becomes a hopeless matter. The problem arises of selecting "the most plausible" hypotheses, the solution of which is based on a comparative assessment of the plausibility of hypotheses. The introduction of this assessment is not a trivial matter. Naturally, the more information one assessment or

another takes into account, the more effective the research procedure becomes. The application of information theory to the study of such systems seems quite justified.

Finishing the discussion of the types of information systems, we note once again that in practice there are often systems that cannot be unambiguously attributed to any of the specified types. Such systems are a complex system that includes communication channels, information processing and storage systems, and other devices. An example is a universal digital computer. An even more complex complex is, for example, the Earth satellite launch management system. Such complex systems can be considered in parts, but a number of issues (especially some issues of optimality) must sometimes be solved taking into account the interaction of all parts.

We also point out a very poorly studied and very promising class of information systems, the characteristics of which change during the operation of the systems in such a way that the properties of the system as a whole improve (in a sense); these are so-called self-adjusting, or self-organizing, systems.

5.4. Information Systems Classification

Recall that information systems are called a set of interrelated means that carry out the transmission, storage and processing of information, they are also called information and computing systems. In the information system data comes from the source of information. These data are sent to storage or undergo some processing in the system and then transferred to the consumer.

Feedback can be established between the consumer and the information system itself. In this case, the information system is called closed. The feedback channel is necessary when it is necessary to take into account the reaction of the consumer to the information received.

The information system consists of the source of information, the hardware of the IC, the software of the IC, the consumer of information.

There are 3 classes of information systems used in mechatronics, according to the degree of their automation:

Automatic information systems **-** perform all information processing operations without human participation; various robots. An example of automated

information systems are some Internet search servers, such as Google, where the collection of information about sites is performed automatically by a search robot and the human factor does not affect the ranking of search results.

Manual information systems - are characterized by the absence of modern technical means of processing information and all operations are performed manualy.

Automated Information Systems (AIS) - the most popular class of IS. They assume that both the person and the hardware are involved in the processing of information, with the main role being played by the computer.

AIS is a set of software and hardware designed to automate activities related to the storage, transmission and processing of information.

AIS are, on the one hand, a kind of information systems (IS), on the other - automated systems (AS), as a result of which they are often called IS or AS.

In the AIS, the following information is responsible for storing information: at the physical level - built-in memory devices (RAM), external drives, disk arrays, at the software level - the OS file system, document storage systems, multimedia, etc.

Chapter 6Computer vision

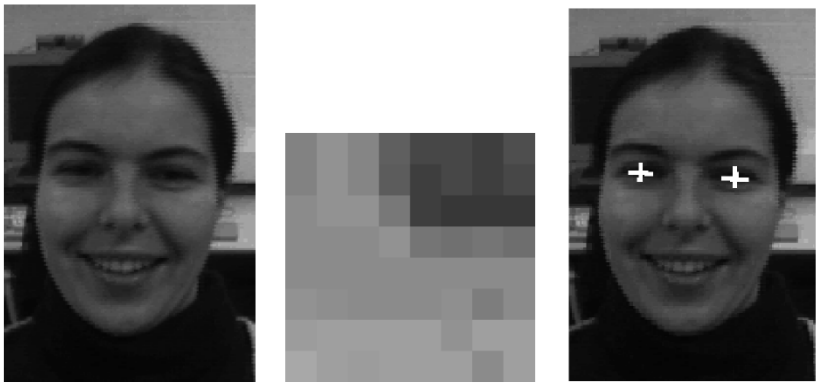
In order to make decisions about real objects, it is almost always necessary to construct some description or model of them from the image. Because of this, many experts will say that *the goal of computer vision is the construction of scene descriptions from images.* Critical issues raised in this chapter and studied in the remainder of the text include the following.

Sensing: How do sensors obtain images of the world? How do the images encode properties of the world, such as material, shape, illumination and spatial relationships?

Encoded Information: How do images yield information for understanding the 3D world, including the geometry, texture, motion, and identity of objects in it?

Representations: What representations should be used for stored descriptions of objects, their parts, properties and relationships?

Algorithms: What methods are there to process image information and construct descriptions of the world and its objects?



	0	1	2	3	4	5	6	7
0	130	146	133	95	71	71	62	78
1	130	146	133	92	62	71	62	71
2	139	146	146	120	62	55	55	55
3	139	139	139	146	117	112	117	110
4	139	139	139	139	139	139	139	139
5	146	142	139	139	139	143	125	139
6	156	159	159	159	159	146	159	159
7	168	159	156	159	159	159	139	159

Figure 6.1: (Top left) Image of a face, (top center) aubimage of 8x8 pixels from the right eye region, (top right) eye location detected by a computer program, and (bottom) intensity values from the 8x8 aubimage. Images courtesy of Vera Bakic.

6.1 Machines that see?

Scientists and science fiction writers have been fascinated by the possibility of building intelligent machines, and the capability of understanding the visual world ia a prerequisite that some would require of such a machine. Much of the human brain ia dedicated to vision. Humana solve many visual problems effortlessly, yet moat have little analytical understanding of visual cognition aa a process. Allan Turing, one of the fathers of both the modern digital computer and the field of artificial intelligence, believed that a digital computer would achieve intelligence and the ability to understand scenes. Such lofty goals have proved difficult to achieve and the richness of human imagination ia not yet matched by our engineering. However, there has been surprising progress along some lines of research. While building practical systems ia a primary theme of this text and artificial intelligence ia not, we will sometimes ponder the deeper questions, and, where we can, make some assessment of progress. Consider, for example, the following scenario, which could be realized within the next few years. A TV camera at your door provides images to your home computer which you have trained to recognize some faces of people important to you. When you call in to your home message center, your computer not only reports the phone messages, but it also reports probable visits from your sister Eleanor and Chad the paper boy.

6.2 Application problems

The applications of computers in image analysis are virtually limitless. Only a small sample of applications can be included here, but these will serve us well for both motivation and orientation to the field of study.

A preview of the digital image

A digital image might represent a cartoon, a page of text, a person's face, a map of Katmandu, or a product for purchase from a catalog. A digital image contains a fixed number of rows and columns of *pixels,* short for *picture elements.* Pixels are like little tiles holding quantized values - small numbers, often between 0 and 255, that represent the brightness at the points of the image. Depending on the coding scheme, 0 could be the darkest and 255 the brightest, or visa-versa. At the top left in

Figure 1.1 is a printed digital image of a face that is 257 rows high by 172 columns wide. At the top center is an 8 x 8 subimage extracted from the right eye of the left image. At the bottom of the figure are the 64 numbers representing the brightness of the pixels in that subimage. The numbers below 100 in the upper right of the subimage represent the lower reflection from the dark of the eye (iris), while the higher numbers represent the brighter white of the eye. A color image would have three numbers for each pixel, perhaps one value for red, one for blue, and one for green. Digital images are most commonly displayed on a monitor, which is basically a television screen with a digital image memory. A color image that has 500 rows and 500 columns is roughly equivalent to what you see at one instant of time on your TV. A pixel is displayed by energizing a small spot of luminescent material; displaying color requires energizing 3 neighboring spots of different materials. A high resolution computer display has roughly 1200 by 1000 pixels. The next chapter discusses digital images in more detail, while coding and interpretation of color in digital images is treated in Chapter 6.

Image Database Query

Huge digital memories, high bandwidth transmission and multimedia personal computers have facilitated the development of image databases. Good use of the many existing images requires good retrieval methods. Standard database techniques apply to images that have been augmented with text keys; however, *content-based* retrieval is needed and is a topic of much current research. Suppose that a newly formed company wants to design and protect a new logo and that an artist has created several candidates for the company to consider. A logo cannot be used if it is too similar to one of an existing company, so a database of existing logos must be searched. This operation is analagous to patent search and is done by humans, but could be greatly aided by machine vision methods. See Figure 1.2. There are many similar problems. Suppose an architect or an art historian wants to search for buildings with a particular kind of entryway. It would be desirable to just provide a picture, perhaps fetched from the database itself, and request the system to produce other similar pictures. In a later chapter, you will see how geometric, color, and

texture features can be used to aid in answering such an image database query. Suppose that an advertising agency wants to search for existing images of young children enjoying eating. This semantic requirement, which is simple for humans to understand, presents a very high level of difficulty for machine vision. Characterizing “children”, “enjoyment”, and “eating” would require complex use of color, texture, and geometric features. We note in passing that a computer algorithm has been devised that decides whether or not a color image contains a naked person. This could be useful for parents who want to screen images that their children retrieve from the web.

Figure6.2**:** Image query by example: query image(left) and two most similar images produced by an image database system (Courtesy of Takenobu Igarashi).

Inspecting crossbars for holes

In the late 1970’s an engineer in Milwaukee implemented a machine vision system that successfully counted the number of bolt holes in crossbars made for truck companies. The truck companies demanded that every crossbar be inspected before being shipped to them, because a missing bolt hole on a partly assembled truck was a very costly defect. Either the assembly line would have to be stopped while the needed hole was drilled, or worse, a worker might ignore placing a required bolt in order to keep the production line running. To create a digital image of the truck crossbar, lights were placed beneath the existing transfer line and a digital camera above it. When a crossbar came into the field of view, an image was taken. Dark pixels inside the shadow of the crossbar were represented as l’s indicating steel, and pixels in the bright holes were represented as 0’s, indicating that the hole was drilled.

The number of holes can be computed as the number of *external comers* minus the number of *internal corners* all divided by four. Figure 6.3shows three bright holes (’0’s) in a background of T’s. An *external comer* is just a 2 x 2 set of neighboring pixels containing exactly 3 ones while an *internal comer* is a 2 x 2 set of neighboring pixels containing exactly 3 zeroes. Example processing of an image with 7 rows and 33 columns is shown in the figure and a skeleton algorithm is also shown. Holecounting is only one example of many simple, but powerful operations possible with digital images. (As the exercises below show, the holecounting algorithm is correct only if the holes are “4-connected” and “simply connected” — that is, they have no background pixels inside them. These concepts are discussed further in Chapter 3 and in more detail in the text by Rosenfeld.)

Examining the inside of a human head.

Magnetic resonance imaging (MRI) devices can sense materials in the interior of 3D objects. Figure 1.4 shows a section through a human head: brightness is related to movement of material, so this is actually a picture of blood *flow*. One can “see” important blood vessels.

The whispy comet-like structures are associated with the eyes. MRI images are used by doctors to check for tumors or blood flow problems such as abnormal vessel constrictions or expansions. The image at the right in Figure 1.4 was made from a copy of the one on the left by making every pixel of value 208 or more bright (255) and those below 208 dark (0). Most pixels correctly show blood vessels versus background, but there are many incorrectly “colored” pixels of both types. Machine vision techniques are often used in medical image analysis, although usually to aid in data presentation and measurement rather than diagnosis itself. Wouldn’t it be great if we could “see” thoughts occuring in the brain! Well, it turns out that MRI can sense organic activity related to thought processes and this is a very exciting current area of research.

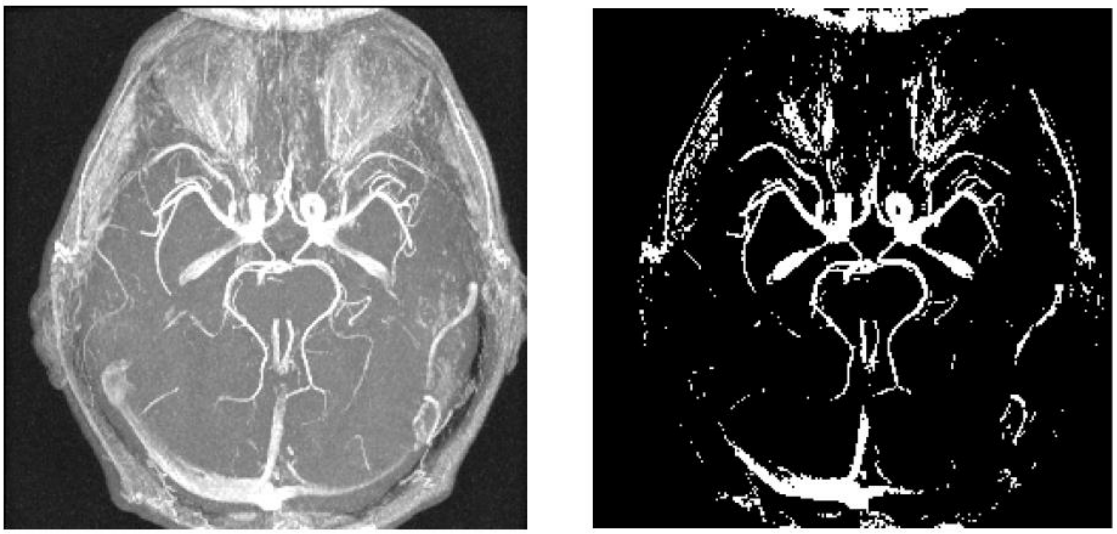
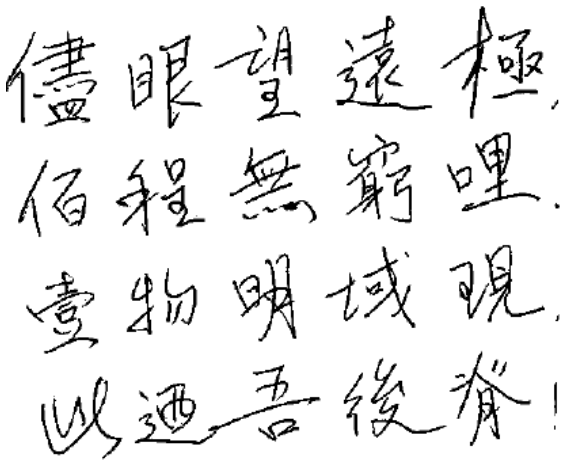


Figure 6.4: Magnetic resonance image (left) where brightness relates to blood flow and binary image (right) resulting from changing all pixels with value 208 or above to 255 and those below 208 to 0. Image courtesy of James Siebert, Michigan State Radiology.



I looked as hard as I could see,
beyond 100 plus infinity
an object of bright intensity
– it was the back of me!

Figure 6.5: (Left) Chinese characters and (right) English equivalent. Is it possible that a machine could automatically translate one into the other? Chinese characters and poem courtesy of John Weng; poem in English by Stockman.

Processing scanned text pages

A common problem is to convert information from paper documents into digital form for information systems. For example, we might want to make an old book available on the Internet, or we might need to convert a blueprint of some object into a geometry file so that the part can be made by a numerically controlled machine

tool.

Figure 6.5 shows the same message in both Chinese and English. The Chinese characters were written on paper and scanned into an image of 482 rows and 405 columns. The postscript file encoding the graphics and printed in the figure has a size of 68,464 bytes. The English version is stored in a file of 115 bytes, each holding one ASCII character. There is an entire range of important applications in processing documents. Recognizing individual characters from the dots of the scanner or FAX files is one such application that is done fairly well today, provided that the characters conform to standard patterns. Providing a semantic interpretation of the information, possibly to be used for indexing in a large database, is a harder problem.

Accounting for snow cover using a satellite image

Much of the earth's surface is scanned regularly from satellites, which transmit their images to earth in digital form. These images can then be processed to extract a wealth of information. For example, inventory of the amount of snow in the watershed of a river may be critical for regulating a dam for flood control, water supply, or wildlife habitat. Estimates of snow mass can be made by accounting for the number of pixels in the image that appear as snow. A pixel from a satellite image might result from sensing a 10 meter by 10 meter spot of earth, but some satellites reportedly can see much smaller spots than that. Often, the satellite image must be compared to a map or other image to determine which pixels are in a particular area or watershed. This operation is usally manually-aided by a human user interacting with the image processing software and will be discussed more in Chapter 11 where image matching is covered. Figure 1.6 is a photograph taken on a space shuttle flight managed by the Johnson Space Center in Houston, Texas. It shows the town of Wenatchie, Washington, where the Wenatchie River flows into the Columbia River.

Computers are known for their ability to handle large amounts of data; certainly the earth scanning satellites produce a tremendous amount of data useful for many purposes. For example, counts and locations of snow pixels might be input to a computer program that simulates the hydrology for that region. (Temperature information for the region must be input to the program as well.) Another related

application is taking inventory of crops and predicting harvests. Yet another is taking inventory of buildings for tax purposes: this is usually done manually with pictures taken from airplanes.

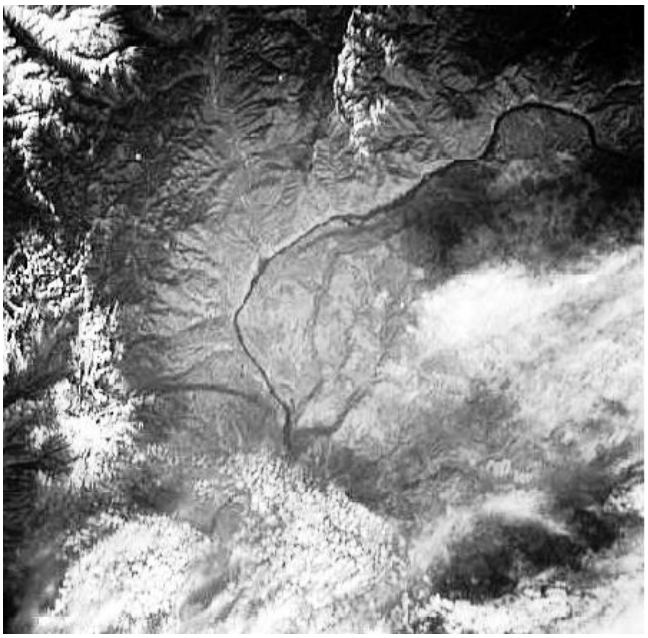


Figure **1.6:** Photo of the Wenatchie and Columbia Rivers in Washington State. Courtesy of Johnson Space Center; see http://earthrise.sdsc.edu.

Understanding a scene of parts

At many points of manufacturing processes, parts are transferred on conveyors or in boxes. Parts must be individually placed in machines, packed, inspected, etc. If the operation is dull or dangerous, a vision-guided robot might provide a solution. The underlying image of Figure 1.7 shows three workpieces in a robot's workspace. By recognizing edges and holes, the robot vision system is able to guess at both the identity of a part and its position in the workspace. Using a 3D model made by computer-aided-design (CAD) for each guessed part and its guessed position, the vision system then compares the sensed image data with a computer graphic generated from the model and its position in space. Bad matches are rejected while good matches cause the guess to be refined. The bright lines in Figure 1.7 show three such refined matches between the image and models of the objects it contains. Finally, the robot eye-brain can tell the robot arm how to pick up a part and where to put it. The problems and techniques of 3D vision are covered in the later chapters of this text.

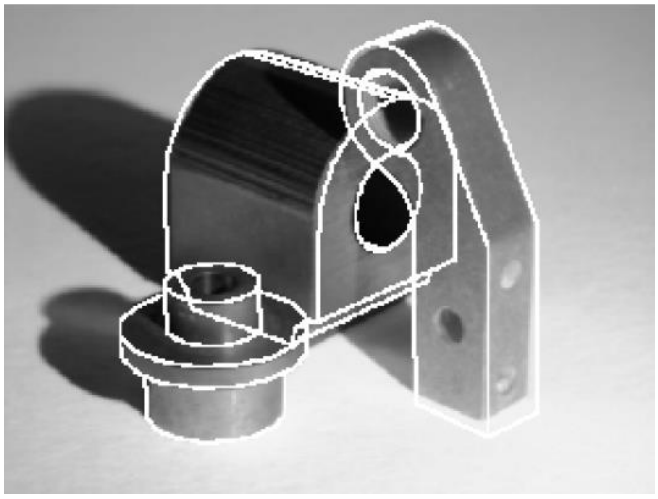


Figure 1.7: An inspection or assembly robot matches stored 3D models to a sensed 2D image (Courtesy of Mauro Costa).

6.3 Operations on Images

Operations can be grouped into different categories depending on their structure, level, or purpose. Some operations are for the purpose of improving the image solely for human consumption, while others are for extracting information for downstream automatic processing. Some operations create new output images, while others output non-image descriptions. A few important categories of image operations follow.

Changing pixels in small neighborhoods

Pixel values can be changed according to how they relate to a small number of neighboring pixels, for example, neighbors in adjacent rows or columns. Frequently, isolated l's or 0's in a binary image will be reversed in order to make them the same as their neighbors. The purpose of this operation could be to remove likely noise from the digitization process. Or, it could be just to simplify image content; for example, to ignore tiny islands in a lake or imperfections in a sheet of paper. Another common operation is to change *border pixels* to be *background pixels* as shown in Figure 1.8. The images of bacteria have fuzzy borders and often fuse together. By changing the black border pixels to white, the bacteria images, although smaller, have clearer borders and some formerly fusing pairs are separated. These operations are treated in Chapter 3.

Figure 6.8: (Left) Binary image of bacteria (in the original microscope image, the bacteria were blue due to their fluoresence); (right) cleaner image resulting from changing black pixels that had one or more white neighbors to white. Original image courtesy of Frank Dazzo.

Enhancing an entire image

Some operations treat the entire image in a uniform manner. The image might be too dark - say its maximum brightness value is 120 - so all brightness values can be scaled upby a factor of 2 to improve its displayed appearance. Noise or unnecessary detail can be removed by replacing the value of every input pixel with the average of all nine pixels in its immediate neighborhood. Alternatively, details can be enhanced by replacing each pixel value by the contrast between it and its neighbors. Figure **6.9** shows a simple contrast computation applied at all pixels of an input image. Note how the boundaries of most objects are well detected. The output image results from computations made only on the local 3x3 neighborhoods of the input image. Chapter 5 describes several of these kinds of operations. Perhaps an image is taken using a fish eye lens and we want to create an output image with less distortion: in this case, we have to "move" the pixel values to other locations in the image to move them closer to the image center. Such an operation is called *image warping***.**

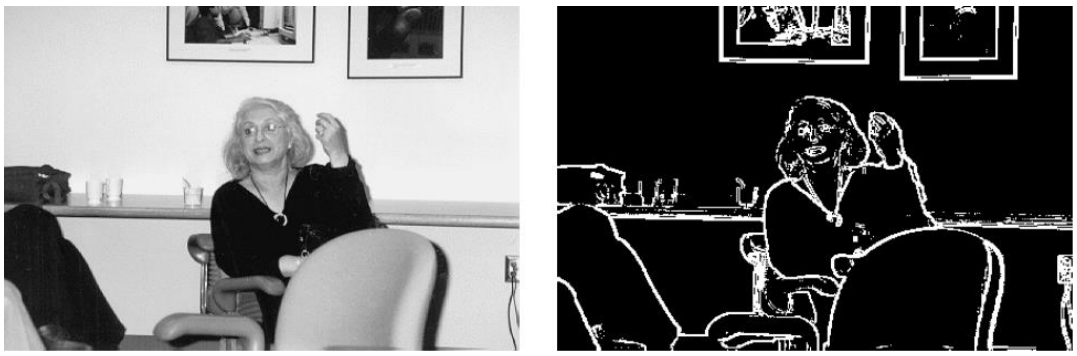


Figure6.9**:** Contrast in the left image is shown in the right image. The top 10% of the pixels in terms of contrast are made bright while the lower 90% are made dark. Contrast is computed from the 3x3 neighborhood of each pixel.

Combining multiple images

An image can be created by adding or subtracting two input images. Image subtraction is commonly used to detect change over time. Figure 6.10shows two images of a moving part and the difference image resulting from subtracting the corresponding pixel values of the second image from those of the first image. Image subtraction captures the boundary of the moving object, but not perfectly. (Since negative pixel values were not used, not all changes were saved in the output image.) In another application, urban development might be more easily seen by subtracting an aerial image of a city taken five years ago from a current image of the city. Image addition is also useful. Figure 6.11shows an image of Thomas Jefferson “added” to an image of the great arch opening onto the lands of the Louisiana Purchase; more work is needed in this case to *blend* the images better.

Computing features from an image

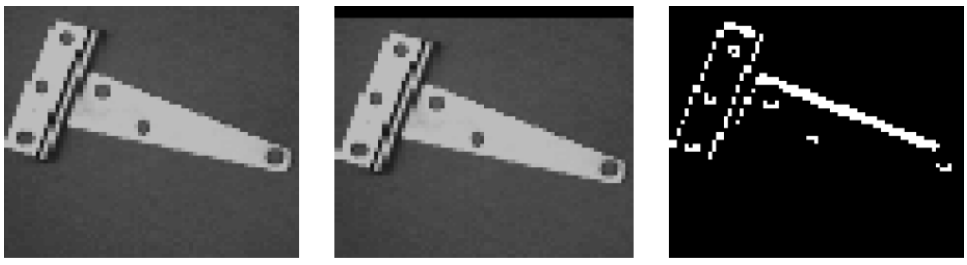


Figure 6.10**:** Images of a moving part (left and center) and a difference image (right) that captures the boundary of the part.

Figure 6.11**:** Image of the great archway at St. Louis (left); face of Jefferson (center); and, combination of the two (right).

We have already seen the example of counting holes. More generally, the regions of 0's corresponding to holes in the crossbar inspection problem could be images of objects, often called *blobs* - perhaps these are microbes in a water sample. Important features might be average object area, perimeter, direction, etc. We might want to output these important features separately for every detected object. Figure 6.12shows output from a well-known algorithm applied to the bacteria image of Figure 6.8giving features of separate regions identified in the image, including the region area and location. Regions with area of a few hundred pixels correspond to isolated bacteria while the large region is due to several touching bacteria.

Extracting non-iconic representations

Higher-level operations usually extract representations of the image that are non-iconic, that is, data structures that are not like an image. (Recall that extraction of such descriptions is often defined to be the goal of computer vision.) Figure 6.12shows a non-iconic description derived from the bacteria image. In addition to examples already mentioned, consider a report of the count of microbes of type A and B in a slide from a microscope or

Object	Area	Bounding Box	Centroid
1	247	[(20 , 26), (32 , 56)]	(26.1, 42.0)
2	6	[(25 , 22), (26 , 24)]	(25.5, 23.0)
3	116	[(35 , 72), (54 , 86)]	(44.1, 79.4)
4	4	[(37 , 69), (38 , 70)]	(37.5, 69.5)
5	15	[(46 , 86), (50 , 89)]	(47.6, 87.4)
6	586	[(49 , 122), (95 , 148)]	(71.7, 134.8)
7	300	[(54 , 91), (77 , 112)]	(65.6, 101.9)
8	592	[(57 , 138), (108 , 163)]	(83.6, 150.8)
9	562	[(57 , 158), (104 , 183)]	(81.2, 171.4)
10	5946	[(74 , 195), (221 , 313)]	(138.0, 256.5)
11	427	[(204 , 115), (229 , 151)]	(217.1, 132.3)
12	797	[(242 , 42), (286 , 97)]	(264.9, 71.8)
13	450	[(248 , 170), (278 , 204)]	(262.7, 188.1)
14	327	[(270 , 182), (291 , 216)]	(279.9, 200.3)
15	264	[(293 , 195), (311 , 221)]	(300.8, 206.7)
16	145	[(304 , 179), (316 , 193)]	(310.4, 186.4)

Figure 6.12**:** Components automatically identified in the bacteria image in Figure 1.8(right) Single bacteria have an area of a few hundred pixels: the large component consist of several touching bacteria and the tiny objects 2,4 and 5 are due to noise.

the volume of traffic flow between two intersections of a city computed from a video taken from a utility pole. In another important application, the (iconic) input might be a scanned magazine article and the output a hypertext structure containing sections of recognized ASCII text and sections of raw images for the figures. As a final example, in the application illustrated in Figure 6.7**,** the machine vision system would output a set of three detections, each encoding a part number, three parameters of part position and three parameters of the orientation of the part. This scene description could then be turned over to the motionplanning system, which would decide on how to manipulate the three parts.

Chapter 7 Robot programming languages and systems

In this chapter, we begin to consider the interface between the human user and an industrial robot. It is by means of this interface that a user takes advantage of all the underlying mechanics and control algorithms we have studied in previous chapters.

The sophistication of the user interface is becoming extremely important as manipulators and other programmable automation are applied to more and more demanding industrial applications. It turns out that the nature of the user interface is a very important concern. In fact, most of the challenge of the design and use of industrial robots focuses on this aspect of the problem

Robot manipulators differentiate themselves from fixed automation by being "flexible," which means programmable. Not only are the movements of manipulators programmable, but, through the use of sensors and communications with other factory automation, manipulators can adapt to variations as the task proceeds.

In considering the programming of manipulators, it is important to remember that they are typically only a minor part of an automated process. The term workceil is used to describe a local collection of equipment, which may include one or more manipulators, conveyor systems, parts feeders, and fixtures. At the next higher level, workcells might be interconnected in factorywide networks so that a central control computer can control the overall factory flow. Hence, the programming of manipulators is often considered within the broader problem of programming a variety of interconnected machines in an automated factory workcell.

Teach by showing

Early robots were all programmed by a method that we will call teach by showing, which involved moving the robot to a desired goal point and recording its position in a memory that the sequencer would read during playback. During the teach phase, the user would guide the robot either by hand or through interaction with a teach pendant. Teach pendants are handlield button boxes that allow control of each manipulator joint or of each Cartesian degree of freedom. Some such controllers

allow testing and branching, so that simple programs involving logic can be entered. Some teach pendants have alphanumeric displays and are approaching hand-held terminals in complexity. Figure 7.1 shows an operator using a teach pendant to program a large industrial robot.

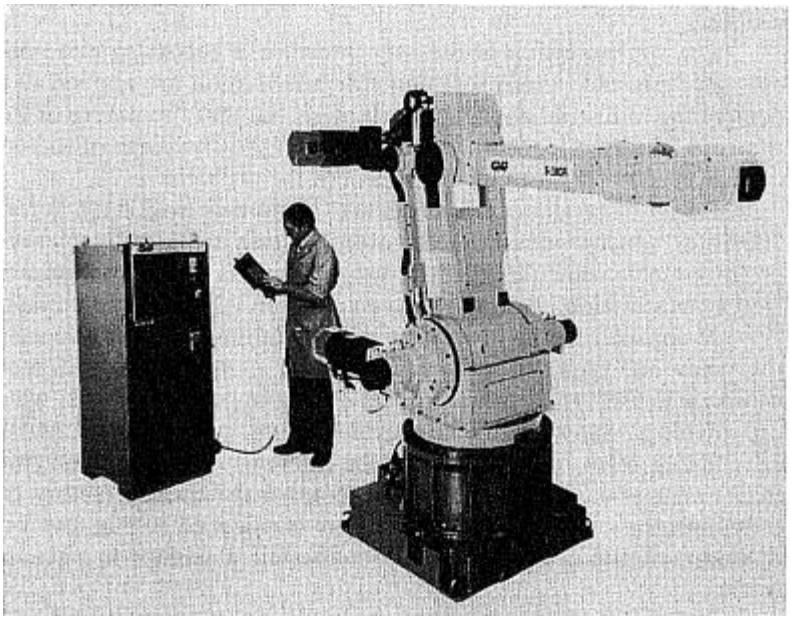


FIGURE 7.1: The GMF S380 is often used in automobile-body spot-welding applications. Here an operator uses a teach-pendant interface to program the manipulator. Photo courtesy of GIviFanuc Corp.

Explicit robot programming languages

Ever since the arrival of inexpensive and powerful computers, the trend has been increasingly toward programming robots via programs written in computer programming languages. Usually, these computer programming languages have special features that apply to the problems of programming manipulators and so are called robot programming languages (RPLs). Most of the systems that come equipped with a robot programming language have nonetheless retained a teach-pendant-style interface also.

Robot programming languages have likewise taken on many forms. We will split them into three categories:

1. **Specialized manipulation languages**. These robot programming languages have been built by developing a completely new language that, although

addressing robot-specific areas, might well be considered a general computer programming language. An example is the VAL language developed to control the industrial robots by Unimation, Inc. VAL was developed especially as a manipulator control language; as a general computer language, it was quite weak. For example, it did not support floating-point numbers or character strings, and subroutines could not pass arguments. A more recent version, V-Il, provided these features. The current incarnation of this language, V+, includes many new features. Another example of a specialized manipulation language is AL, developed at Stanford University. Although the AL language is now a relic of the past, it nonetheless provides good examples of some features still not found in most modern languages (force control, parallelism). Also, because it was built in an academic environment, there are references available to describe it. For these reasons, we continue to make reference to it.

2. **. Robot library for an existing computer language**. These robot programming languages have been developed by starting with a popular computer language (e.g., Pascal) and adding a library of robot-specific subroutines. The user then writes a Pascal program making use of frequent calls to the predefined subroutine package for robot-specific needs. An examples is AR-BASIC from American Cimfiex, essentially a subroutine library for a standard BASIC implementation. JARS, developed by NASA's Jet Propulsion Laboratory, is an example of such a robot programming language based on Pascal.

3. **Robot library for a new general-purpose language**. These robot programming languages have been developed by first creating a new general-purpose language as a programming base and then supplying a library of predefined robot-specific subroutines. Examples of such robot programming languages are RAPID developed by ABB Robotics, AML developed by IBM, and KAREL developed by GMF Robotics.

Studies of actual application programs for robotic workcells have shown that a large percentage of the language statements are not robot-specific. Instead, a great deal of robot programming has to do with initialization, logic testing and branching, communication, and so on. For this reason, a trend might develop to move away from

developing special languages for robot programming and move toward developing extensions to general languages, as in categories 2 and 3 above.

Task-level programming languages

The third level of robot programming methodology is embodied in task-level programniing languages. These languages allow the user to command desired subgoals of the task directly, rather than specify the details of every action the robot is to take. In such a system, the user is able to include instructions in the application program at a significantly higher level than in an explicit robot programming language. A task-level robot programming system must have the ability to perform many planning tasks automatically. For example, if an instruction to "grasp the bolt" is issued, the system must plan a path of the manipulator that avoids coffision with any surrounding obstacles, must automatically choose a good grasp location on the bolt, and must grasp it. In contrast, in an explicit robot programming language, all these choices must be made by the programmer. The border between explicit robot programming languages and task-level programming languages is quite distinct. Incremental advances are being made to explicit robot programming languages to help to ease programming, but these enhancements cannot be counted as components of a task-level programming system. True task-level programming of manipulators does not yet exist, but it has been an active topic of research and continues as a research topic today.

7.1A sample application

Figure 7.2 shows an automated workcell that completes a small subassembly in a hypothetical manufacturing process. The workcell consists of a conveyor under computer control that delivers a workpiece; a camera connected to a vision system, used to locate the workpiece on the conveyor; an industrial robot (a PUIVIA 560 is pictured) equipped with a force-sensing wrist; a small feeder located on the work surface that supplies another part to the manipulator; a computer-controlled press that can be loaded and unloaded by the robot; and a pallet upon which the robot places finished assemblies.

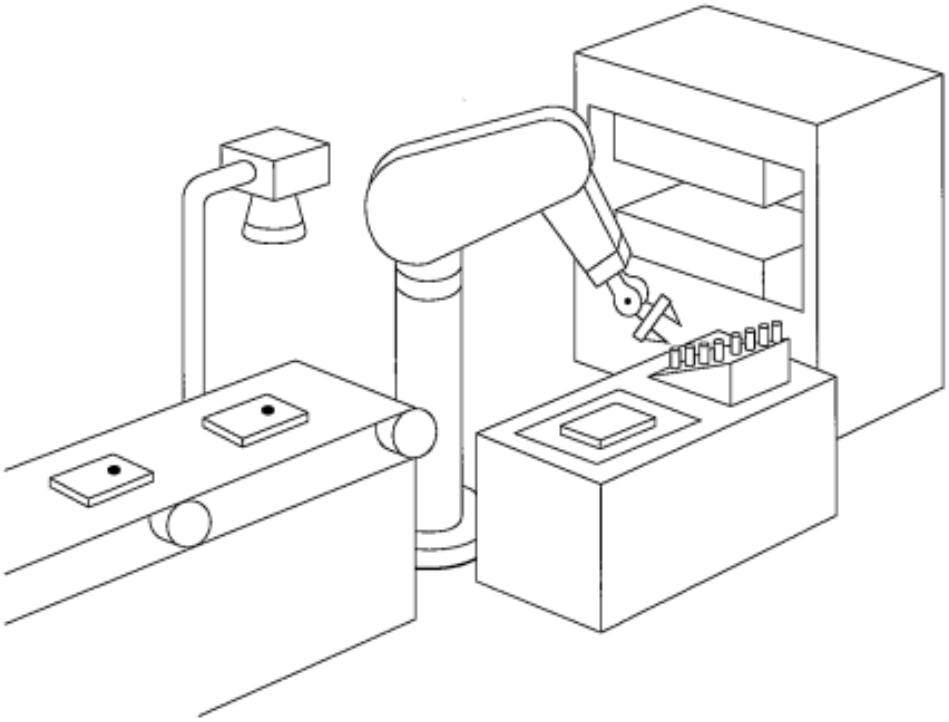


Figure 7.2: An automated workcell containing an industrial robot.

The entire process is controlled by the manipulator's controller in a sequence, as follows:

1. The conveyor is signaled to start; it is stopped when the vision system reports that a bracket has been detected on the conveyor.

2. The vision system judges the bracket's position and orientation on the conveyor and inspects the bracket for defects, such as the wrong number of holes.

3. Using the output of the vision system, the manipulator grasps the bracket with a specified force. The distance between the fingertips is checked to ensure that the bracket has been properly grasped. If it has not, the robot moves out of the way and the vision task is repeated.

4. The bracket is placed in the fixture on the work surface. At this point, the conveyor can be signaled to start again for the next bracket—that is, steps 1 and 2 can begin in parallel with the following steps.

5. A pin is picked from the feeder and inserted partway into a tapered hole in the bracket. Force control is used to perform this insertion and to perform simple checks on its completion. (If the pin feeder is empty, an operator is notified and the manipulator waits until commanded to resume by the operator.)

6. The bracket—pin assembly is grasped by the robot and placed in the press.

7. The press is commanded to actuate, and it presses the pin the rest of the way into the bracket. The press signals that it has completed, and the bracket is placed back into the fixture for a final inspection.

8. By force sensing, the assembly is checked for proper insertion of the pin. The manipulator senses the reaction force when it presses sideways on the pin and can do several checks to discover how far the pin protrudes from the bracket.

9. If the assembly is judged to be good, the robot places the finished part into the next available pallet location. If the pallet is ftll, the operator is signaled. If the assembly is bad, it is dropped into the trash bin.

10. Once Step 2 (started earlier in parallel) is complete, go to Step 3.

This is an example of a task that is possible for today's industrial robots. It should be clear that the definition of such a process through "teach by showing" techniques is probably not feasible. For example, in dealing with pallets, it is laborious to have to teach all the pallet compartment locations; it is much preferable to teach only the corner location and then compute the others from knowledge of the dimensions of the pallet. Further, specifying interprocess signaling and setting up parallelism by using a typical teach pendant or a menu-style interface is usually not possible at all. This kind of application necessitates a robot programminglanguage approach to process description. (See Exercise 7.5.) On the other hand, this application is too complex for any existing task-level languages to deal with directly. It is typical of the great many applications that must be addressed with an explicit robot programming approach. We will keep this sample application in mind as we discuss features of robot programming languages.

7.2Requirements of a robot programming language

World modeling

Manipulation programs must, by definition, involve moving objects in three-dimensional space, so it is clear that any robot programming language needs a means of describing such actions. The most common element of robot programming

languages is the existence of special geometric types. For example, types are introduced to represent joint-angle sets, Cartesian positions, orientations, and frames. Predefined operators that can manipulate these types often are available. The "standard frames" introduced in Chapter 3 might serve as a possible model of the world: All motions are described as tool frame relative to station frame, with goal frames being constructed from arbitrary expressions involving geometric types.

Given a robot programming environment that supports geometric types, the robot and other machines, parts, and fixtures can be modeled by defining named variables associated with each object of interest. Figure 7.3 shows part of our example workcell with frames attached in task-relevant locations. Each of these frames would be represented with a variable of type "frame" in the robot program.

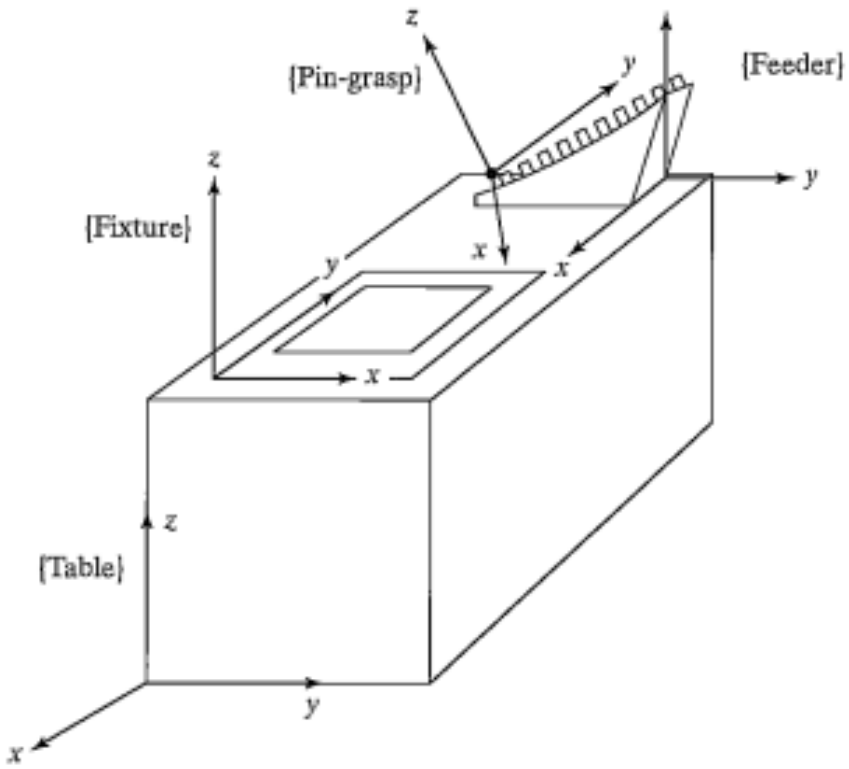


Figure 7.3: Often, a workcell is modeled simply, as a set of frames attached to relevant objects.

In many robot programmiing languages, this ability to define named variables of various geometric types and refer to them in the program forms the basis of the world model. Note that the physical shapes of the objects are not part of such a world model, and neither are surfaces, volumes, masses, or other properties. The extent to

which objects in the world are modeled is one of the basic design decisions made when designing a robot programming system. Most present-day systems support only the style just described.

Some world-modeling systems allow the notion of **aflixments** between named objects—that is, the system can be notified that two or more named objects have become "affixed"; from then on, if one object is explicitly moved with a language statement, any objects affixed to it are moved with it. Thus, in our application, once the pin has been inserted into the hole in the bracket, the system would be notified (via a language statement) that these two objects have become affixed. Subsequent motions of the bracket (that is, changes to the value of the frame variable "bracket") would cause the value stored for variable "pin" to be updated along with it.

Ideally, a world-modeling system would include much more information about the objects with which the manipulator has to deal and about the manipulator itself. For example, consider a system in which objects are described by CAD-style models that represent the spatial shape of an object by giving definitions of its edges, surfaces, or volume. With such data available to the system, it begins to become possible to implement many of the features of a task-level programming system

Motion specification

A very basic function of a robot programming language is to allow the description of desired motions of the robot. Through the use of motion statements in the language, the user interfaces to path planners and generators of the style described in Chapter 7. Motion statements allow the user to specify via points, the goal point, and whether to use joint-interpolated motion or Cartesian straight-line motion. Additionally, the user might have control over the speed or duration of a motion.

To ifiustrate various syntaxes for motion primitives, we will consider the following example manipulator motions: (1) move to position "goall," then (2) move in a straight line to position "goal2," then (3) move without stopping through "vial" and come to rest at "goal3." Assuming all of these path points had already been taught or described textually, this program segment would be written as follows:

```
In VAL II,
```

```
move goal1
moves goal2
move via1
move goal3
In AL (here controlling the manipulator "garm"),
move garm to goal1;
move garm to goal2 linearly;
move garm to goal3 via via1;
```

Most languages have similar syntax for simple motion statements like these. Differences in the basic motion primitives from one robot programming language to another become more apparent if we consider features such as the following:

1. the ability to do math on such structured types as frames, vectors, and rotation matrices;

2. the ability to describe geometric entities like frames in several different convenient representations—along with the ability to convert between representations;

3. the ability to give constraints on the duration or velocity of a particular move—for example, many systems allow the user to set the speed to a fraction of maximum, but fewer allow the user to specify a desired duration or a desired maximum joint velocity directly;

4. the ability to specify goals relative to various frames, including frames defined by the user and frames in motion (on a conveyor, for example).

Flow of execution

As in more conventional computer programming languages, a robot programming system allows the user to specify the flow of execution—that is, concepts such as testing and branching, looping, calls to subroutines, and even interrupts are generally found in robot programming languages.

More so than in many computer applications, parallel processing is generally important in automated workcell applications. First of all, very often two or more robots are used in a single workcell and work simultaneously to reduce the cycle time of the process. Even in single-robot applications, such as the one shown in Fig. 7.2, other workcell equipment must be controlled by the robot controller in a parallel

fashion. Hence, signal and wait primitives are often found in robot programming languages, and occasionally more sophisticated parallel-execution constructs are provided.

Another frequent occurrence is the need to monitor various processes with some kind of sensor. Then, either by interrupt or through polling, the robot system must be able to respond to certain events detected by the sensors. The ability to specify such event monitors easily is afforded by some robot programming languages

Programming environment

As with any computer languages, a good programming environment fosters programmer productivity. Manipulator programming is difficult and tends to be very interactive, with a lot of trial and error. If the user were forced to continually repeat the "edit-compile-run" cycle of compiled languages, productivity would be low. Therefore, most robot programming languages are now interpreted, so that individual language statements can be run one at a time during program development and debugging. Many of the language statements cause motion of a physical device, so the tiny amount of time required to interpret the language statements is insignificant. Typical programming support, such as text editors, debuggers, and a file system, are also required.

Sensor integration

An extremely important part of robot programming has to do with interaction with sensors. The system should have, at a minimum, the capability to query touch and force sensors and to use the response in if-then-else constructs. The ability to specify event monitors to watch for transitions on such sensors in a background mode is also very useful.

Integration with a vision system allows the vision system to send the manipulator system the coordinates of an object of interest. For example, in our sample application, a vision system locates the brackets on the conveyor belt and returns to the manipulator controller their position and orientation relative to the camera. The camera's frame is known relative to the station frame, so a desired goal frame for the manipulator can be computed from this information.

the manipulator can be computed from this information. Some sensors could be part of other equipment in the workcell—for example, some robot controllers can use input from a sensor attached to a conveyor belt so that the manipulator can track the belt's motion and acquire objects from the belt as it moves.

The interface to force-control capabilities, as discussed in Chapter 9, comes through special language statements that allow the user to specify force strategies. Such force-control strategies are by necessity an integrated part of the manipulator control system—the robot programming language simply serves as an interface to those capabilities. Programming robots that make use of active force control might require other special features, such as the ability to display force data collected during a constrained motion.

In systems that support active force control, the description of the desired force application could become part of the motion specification. The AL language describes active force contro in the motion primitives by specifying six components of stiffness (three translational and three rotational) and a bias force. In this way, the manipulator's apparent stiffness is programmable. To apply a force, usually the stiffness is set to zero in that direction and a bias force is specified—for example,

move garm to goal

with stif±ness=(80, 80, 0, 100, 100, 100)

with force=20*ounces along zhat;

7.3Problems peculiar to robot programming languages

Advances in recent years have helped, but programming robots is still difficult. Robot programming shares all the problems of conventional computer programming, plus some additional difficulties caused by effects of the physical world.

Internal world model versus external reality

A central feature of a robot programming system is the world model that is maintained internally in the computer. Even when this model is quite simple, there are ample difficulties in assuring that it matches the physical reality that it attempts to model. Discrepancies between internal model and external reality result in poor or failed grasping of objects, coffisions, and a host of more subtle problems.

This correspondence between internal model and the external world must be established for the program's initial state and must be maintained throughout its execution. During initial programming or debugging, it is generally up to the user to suffer the burden of ensuring that the state represented in the program corresponds to the physical state of the workcell. Unlike more conventional programming, where only internal variables need to be saved and restored to reestablish a former situation, in robot programming, physical objects must usually be repositioned.

Besides the uncertainty inherent in each object's position, the manipulator itself is limited to a certain degree of accuracy. Very often, steps in an assembly will require the manipulator to make motions requiring greater precision than it is capable of. A common example of this is inserting a pin into a hole where the clearance is an order of magnitude less than the positional accuracy of the manipulator. To further complicate matters, the manipulator's accuracy usually varies over its workspace.

In dealing with those objects whose locations are not known exactly, it is essential to somehow refine the positional information. This can sometimes be done with sensors (e.g., vision, touch) or by using appropriate force strategies for constrained motions

During debugging of manipulator programs, it is very useful to be able to modify the program and then back up and try a procedure again. Backing up entails restoring the manipulator and objects being manipulated to a former state. However, in working with physical objects, it is not always easy, or even possible, to undo an action. Some examples are the operations of painting, riveting, driJling, or welding, which cause a physical modffication of the objects being manipulated. It might therefore be necessary for the user to get a new copy of the object to replace the old, modified one. Further, it is likely that some of the operations just prior to the one being retried wifi also need to be repeated to establish the proper state required before the desired operation can be successfully retried.

Context sensitivity

Bottom-up programming is a standard approach to writing a large computer program in which one develops small, low-level pieces of a program and then puts

them together into larger pieces, eventually attaining a completed program. For this method to work, it is essential that the small pieces be relatively insensitive to the language statements that precede them and that there be no assumptions concerning the context in which these program pieces execute. For manipulator programming, this is often not the case; code that worked reliably when tested in isolation frequently fails when placed in the context of the larger program. These problems generally arise from dependencies on manipulator configuration and speed of motions.

Manipulator programs can be highly sensitive to initial conditions—for example, the initial manipulator position. In motion trajectories, the starting position will influence the trajectory that will be used for the motion. The initial manipulator position might also influence the velocity with which the arm wifi be moving during some critical part of the motion. For example, these statements are true for manipulators that follow the cubic-spline joint-space paths studied in Chapter 7. These effects can sometimes be dealt with by proper programming care, but often such problems do not arise until after the initial language statements have been debugged in isolation and are then joined with statements preceding them.

form an operation at one location is likely to need to be tuned (i.e., positions retaught and the like) to make it work at a different location. Changes in location within the workcell result in changes in the manipulator's configuration in reaching goal locations. Such attempts at relocating manipulator motions within the workcell test the accuracy of the manipulator kinematics and servo system, and problems frequently arise. Such relocation could cause a change in the manipulator's kinematic configuration—for example, from left shoulder to right shoulder, or from elbow up to elbow down. Moreover, these changes in configuration could cause large arm motions during what had previously been a short, simple motion.

ions during what had previously been a short, simple motion. The nature of the spatial shape of trajectories is likely to change as paths are located in different portions of the manipulator's workspace. This is particularly true of joint-space

trajectory methods, but use of Cartesian-path schemes can also lead to problems when singularities are nearby.

When testing a manipulator motion for the first time, it often is wise to have the manipulator move slowly. This allows the user a chance to stop the motion if it appears to be about to cause a coffision. It also allows the user to inspect the motion closely. After the motion has undergone some initial debugging at a slower speed it is then desirable to increase motion speeds. Doing so might cause some aspects of the motion to change. Limitations in most manipulator control systems cause greater servo errors, which are to be expected if the quicker trajectory is followed. Also, in force-control situations involving contact with the environment, speed changes can completely change the force strategies required for success.

The manipulator's configuration also affects the delicacy and accuracy of the forces that can be applied with it. This is a function of how well conditioned the Jacobian of the manipulator is at a certain configuration, something generally difficult to consider when developing robot programs.

Error recovery

Another direct consequence of working with the physical world is that objects might not be exactly where they should be and, hence, motions that deal with them could fail. Part of manipulator programming involves attempting to take this into account and making assembly operations as robust as possible, but, even so, errors are likely, and an important aspect of manipulator progranmiing is how to recover from these errors.

Almost any motion statement in the user's program can fail, sometimes for a variety of reasons. Some of the more common causes are objects shifting or dropping out of the hand, an object missing from where it should be, jamming during an insertion, and not being able to locate a hole.

The first problem that arises for error recovery is identifying that an error has indeed occurred. Because robots generally have quite limited sensing and reasoning capabilities, error detection is often difficult. In order to detect an error, a robot program must contain some type of explicit test. This test might involve checking the

manipulator's position to see that it lies in the proper range; for example, when doing an insertion, lack of change in position might indicate jamming, or too much change might indicate that the hole was missed entirely or the object has slipped out of the hand. If the manipulator system has some type of visual capabilities, then it might take a picture and check for the presence or absence of an object and, if the object is present, report its location. Other checks might involve force, such as weighing the load being carried to check that the object is still there and has not been dropped, or checking that a contact force remains within certain bounds during a motion.

Every motion statement in the program might fail, so these explicit checks can be quite cumbersome and can take up more space than the rest of the program. Attempting to deal with all possible errors is extremely difficult; usually, just the few statements that seem most likely to fail are checked. The process of predicting which portions of a robot application program are likely to fail is one that requires a certain amount of interaction and partial testing with the robot during the program-development stage.

Once an error has been detected, an attempt can be made to recover from it. This can be done totally by the manipulator under program control, or it might involve manual intervention by the user, or some combination of the two. In any event, the recovery attempt could in turn result in new errors. It is easy to see how code to recover from errors can become the major part of the manipulator program.

The use of parallelism in manipulator programs can further complicate recovery from errors. When several processes are running concurrently and one causes an error to occur, it could affect other processes. In many cases, it will be possible to back up the offending process, while allowing the others to continue. At other times, it Wi-Fi be necessary to reset several or all of the running processes.

References

1. Джастін Дж. Кром BASIC Face-off // PC Tech Journal. Вересень 1987.P. 136.
2. Carne, Nick (8 березня, 2019). "Дослідники роблять мільйон крихітних роботів". Журнал «Космос». 8 березня 2019.
3. Зунт, Домінік. "Хто ж насправді винайшов слово" робот "і що це означає?". Сайт Karel Čapek. 2017-02-05.
4. Даффі, Вінсент Г. (19 квітня 2016). Довідник з цифрового моделювання людини: дослідження для прикладної ергономіки та інженерії людського фактора. CRC Press. ISBN 9781420063523 - через Книги Google.
5. Waurzyniak, Patrick (2006). "Майстри виробництва: Джозеф Ф. Енгельбергер". Товариство інженерів-виробників. 137 (1). Архів з оригіналу 2011-11-09.
6. "Щільність роботів зростає глобально". Асоціація Robotic Industries. 8 лютого 2018. 3 грудня 2018 року.
7. Колодний, Лора (4 липня 2017 року). "Роботи приходять до бургерного суглоба поруч з вами". CNBC. 3 грудня 2018 року.
8. Corner, Стюарт (23 листопада 2017 р.). "Робот, керований AI, робить" досконалу "плоскодонній". iothub.com.au. 3 грудня 2018 року.
9. Eyre, Michael (12 вересня 2014 року). "Борис" робот може завантажувати посудомийну машину ". BBC News. 3 грудня 2018 року.
10. Одна база даних, розроблена Міністерством енергетики США, містить інформацію про майже 500 існуючих робототехнічних технологій і може бути знайдена на інформаційному інструменті D&D Knowledge Management.
11. Даулінг, Кевін. "Джерела енергії для малих роботів" (PDF). Університет Карнегі-Меллона. 11 травня 2012 року.
12. Двонаправлений паралельний пружний привід і перекриття виконавчих шарів Raphaël Furnémont1, Glenn Mathijssen1,2, Tom Verstraten1, Dirk

Lefeber1 і Bram Vanderborght1 Опубліковано 26 січня 2016 • © 2016 IOP Publishing Ltd

13. Пратт, Джеррі Е .; Крупп, Бенджамін Т. (2004). "Серія еластичних приводів для роботів з ногами". Технологія безпілотних наземних транспортних засобів VI. Технологія безпілотних наземних транспортних засобів Vi. 5422. С. 135
14. www.imagesco.com, Зображення SI Inc -. "Приводи повітряних м'язів, що йдуть далі, с. 6".
15. "Повітряні м'язи". Тіньовий робот. Архівовано з оригіналу на 2007-09-27.
16. Тонду, Бертран (2012). "Моделювання штучних м'язів Маккіббена: огляд". Журнал систем та структур інтелектуальних матеріалів. 23 (3): 225–253. doi: 10.1177 / 1045389X11435435.
17. Драгани, Рахель (8 листопада 2018). "Чи може робот зробити вас" суперпрацівником "?". Verizon Communications. 3 грудня 2018 року.
18. Поллок, Емілі (7 червня 2018). "Промисловість будівельної робототехніки встановлюється удвічі до 2023 року". engineering.com. 3 грудня 2008 року.
19. Гріфт, Тоні Е. (2004). "Сільськогосподарська робототехніка". Університет Іллінойсу в Урбані-Шампейн. Архівовано з оригіналу 2007-05-04. 3 грудня 2018 року.
20. Томас, Джим (1 листопада 2017 року). "Як корпоративні гіганти автоматизують ферму". Новий інтернаціоналіст. 3 грудня 2008 року.
21. "OUCL робот Sheepdog проекту". Кафедра комп'ютерних наук, Оксфордський університет. 3 липня 2001 року.