1

ROBOTS

Unit Structure

1.0 OBJECTIVE

Here we will discuss about Robots and its implementation!!! That is Robotics. We will see where and when it born. And what are its characteristics.

1.1 INTRODUCTION

Robotics is the intersection of science, engineering and technology that produces machines, called robots, that substitute for (or replicate) human actions. Pop culture has always been fascinated with robots. R2-D2. Optimus Prime.

1.2 WHAT IS A ROBOT?

A robot is the product of the robotics field, where programmable machines are built that can assist humans or mimic human actions. Robots were originally built to handle monotonous tasks (like building cars on an assembly line), but have since expanded well beyond their initial uses to perform tasks like fighting fires, cleaning homes and assisting with incredibly intricate surgeries. Each robot has a differing level of autonomy, ranging from human-controlled bots that carry out tasks that a

human has full control over to fully-autonomous bots that perform tasks without any external influences.

1.3 CHARACTERISTICS OF ROBOT

As technology progresses, so too does the scope of what is considered robotics. In 2005, 90% of all robots could be found assembling cars in automotive factories. These robots consist mainly of mechanical arms tasked with welding or screwing on certain parts of a car. Today, we're seeing an evolved and expanded definition of robotics that includes the development, creation and use of bots that explore Earth's harshest conditions, robots that assist law-enforcement and even robots that assist in almost every facet of healthcare.

While the overall world of robotics is expanding, a robot has some consistent characteristics:

1. Robots all consist of some sort of mechanical construction. The mechanical aspect of a robot helps it complete tasks in the environment for which it's designed. For example, the Mars 2020 Rover's wheels are individually motorized and made of titanium tubing that help it firmly grip the harsh terrain of the red planet.

2. Robots need electrical components that control and power the machinery. Essentially, an electric current (a battery, for example) is needed to power a large majority of robots.

3. Robots contain at least some level of computer programming. Without a set of code telling it what to do, a robot would just be another piece of simple machinery. Inserting a program into a robot gives it the ability to know when and how to carry out a task.

1.4 HISTORY OF ROBOTS

The notion of robots or robot-like automates can be traced back to medieval times. Although people of that era didn't have a term to describe what we would eventually call a robot they were nevertheless imagining mechanisms that could perform human-like tasks. In medieval times, automatons, human-like figures run by hidden mechanisms, were used to impress peasant worshippers in church into believing in a higher power. The automatons, like the clock jack pictured here, created the illusion of self-motion (moving without assistance). The clock jack was a mechanical figure that could strike time on a bell with its axe. This technology was virtually unheard of in the 13th century. So imagine how aweinspiring an automaton was to someone just like you!

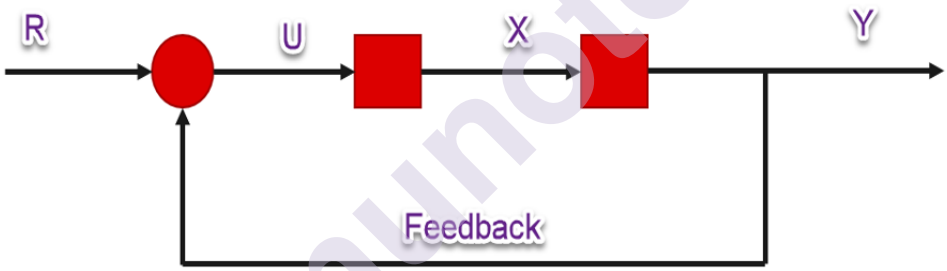
In the 18th century, miniature automatons became popular as toys for the very rich. They were made to look and move like humans or small animals. The pretty musician in the picture was built around 1890. She can turn her head from side to side while playing the instrument with her hands and keeping time with her foot.

In literature, humankind's vivid imagination has often reflected our fascination with the idea of creating artificial life. In 1818, Mary Shelly wrote Frankenstein, a story about the construction of a human-like creature. For Shelly, a robot looked like man but had the ability to function like a machine. It was built of human components, which could be held together by nuts and bolts. Notice there are even clips to hold the top of the head together! Shelly considered that a robot had to be bigger than a regular person and had to have super human strength.

In 1921, Karel Capek, a Czech playwright, came up with an intelligent, artificially created person, which he called "robot". The word "robot" is Czech for worker, and was gradually incorporated into the English language without being translated. As you can see, even a hundred years after Shelly's Frankenstein, Capek's idea of a robot is still one in which the creation resembles the human form. You can see in the picture that the robot looks much more rigid and machine-like than the woman standing next to it.

While the concept of a robot has been around for a very long time, it wasn't until the 1940's that the modern day robot was born, with the arrival of computers.

1.5 CONTROL THEORY



The above image is the control loop diagram of a Robot. In this diagram:

R is the reference signal that we want the robot to be.

Y is the output of the robot.

U is the control signal i.e difference between output & reference signal.

X is the representative of what the system is currently doing.

Suppose we want to design an autonomous car that will run at a constant speed of 60 km/hr. So, in this case, the reference speed (**R**) will be 60 km/hr. Now, due to some unknown factors (like friction, controller sensitivity), our controller runs at 50 km/hr which will be our (**Y**).

Hence, the control signal (**U**) will be a difference between two speeds i.e. 10 km/hr which is also called the error signal. Now, the present state of the system (i.e the current speed) is the **X** variable.

1.5.1 Core topics in Control Theory

- Stability* Analyze the stability of a closed-loop system → Eigenvalue analysis or Lyapunov function method
- Controllability* Analyze which dimensions (DoFs) of the system can actually in principle be controlled
- Transfer function Analyze the closed-loop transfer function, i.e., "how frequencies are transmitted through the system". (→ Laplace transformation)
- Controller design Find a controller with desired stability and/or transfer function properties
- Optimal control* Define a cost function on the system behavior. Optimize a controller to minimize costs

1.6 TYPES OF ROBOTS

Mechanical bots come in all shapes and sizes to efficiently carry out the task for which they are designed. All robots vary in design, functionality and degree of autonomy. From the 0.2 millimeter-long "RoboBee" to the 200 meter-long robotic shipping vessel "Vindskip," robots are emerging to carry out tasks that humans simply can't. Generally, there are five types of robots:

1) Pre-Programmed Robots

Pre-programmed robots operate in a controlled environment where they do simple, monotonous tasks. An example of a pre-programmed robot would be a mechanical arm on an automotive assembly line. The arm serves one function — to weld a door on, to insert a certain part into the engine, etc. — and its job is to perform that task longer, faster and more efficiently than a human.

2) Humanoid Robots

Humanoid robots are robots that look like and/or mimic human behavior. These robots usually perform human-like activities (like running, jumping and carrying objects), and are sometimes designed to look like us, even having human faces and expressions. Two of the most prominent examples of humanoid robots are Hanson Robotics' Sophia (in the video above) and Boston Dynamics' Atlas.

3) Autonomous Robots

Autonomous robots operate independently of human operators. These robots are usually designed to carry out tasks in open environments that do not require human supervision. They are quite unique because they use sensors to perceive the world around them, and then employ decision-making structures (usually a computer) to take the optimal next step based on their data and mission. An

example of an autonomous robot would be the Roomba vacuum cleaner, which uses sensors to roam freely throughout a home.

4) Teleoperated Robots

Teleoperated robots are semi-autonomous bots that use a wireless network to enable human control from a safe distance. These robots usually work in extreme geographical conditions, weather, circumstances, etc. Examples of teleoperated robots are the human-controlled submarines used to fix underwater pipe leaks during the BP oil spill or drones used to detect landmines on a battlefield.

5) Augmenting Robots

Augmenting robots either enhance current human capabilities or replace the capabilities a human may have lost. The field of robotics for human augmentation is a field where science fiction could become reality very soon, with bots that have the ability to redefine the definition of humanity by making humans faster and stronger. Some examples of current augmenting robots are robotic prosthetic limbs or exoskeletons used to lift hefty weights.

1.7 CYBERNETICS

1.7.1 What does the word "cybernetics" mean?

"Cybernetics" comes from a Greek word meaning "the art of steering". Cybernetics is about having a goal and taking action to achieve that goal. Knowing whether you have reached your goal (or at least are getting closer to it) requires "feedback", a concept that was made rigorous by cybernetics. From the Greek, "cybernetics" evolved into Latin as "governor". Draw your own conclusions.

1.7.2 When did cybernetics begin?

Cybernetics as a process operating in nature has been around for a long time; actually, for as long as nature has been around.

Cybernetics as a concept in society has been around at least since Plato used it to refer to government.

In modern times, the term became widespread because Norbert Wiener wrote a book called "Cybernetics" in 1948. His sub-title was "control and communication in the animal and machine". This was important because it connects control (actions taken in hope of achieving goals) with communication (connection and information flow between the actor and the environment). So, Wiener is pointing out that effective action requires communication. Later, Gordon Pask offered conversation as the core interaction of systems that have goals.

Wiener's sub-title also states that both animals (biological systems) and machines (non-biological or "artificial" systems) can operate according to

cybernetic principles. This was an explicit recognition that both living and non-living systems can have purpose. A scary idea in 1948.

1.7.3 What's the connection between “cybernetics” and “cyberspace”?

William Gibson, who popularized the term “cyberspace”, said this in a 1982 interview: “ 'Cyber' is from the Greek word for navigator. Norbert Wiener coined 'cybernetics' around 1948 to denote the study of 'teleological mechanisms' [systems that embody goals].“ —NY Times Sunday Magazine 2007

1.8 GREY WALTER TORTOISE

In 1949 neurophysiologist Grey Walter built two robot “tortoises” to show that complexity could arise out a very simple nervous system. “Elmer” and “Elsie” each had a light sensor, a touch sensor, a propulsion motor, a steering motor, and two electronic valve-based “neurons.” He found that even with this modest equipment they were capable of phototaxis, finding their way to a recharging station when their batteries ran low. In a subsequent experiment he watched as a robot moved in front of a mirror and responded to its own reflection. “It began flickering,” he wrote. “Twittering, and jigging like a clumsy Narcissus.” He argued that if this behavior were observed in an animal it “might be accepted as evidence of some degree of self-awareness.”

He found that other simple robots were capable of Pavlovian conditioning. When a robot had been taught to seek its “food” near a stool in the middle of the floor, Walter took to blowing a police whistle and kicking the robot before it found the target. “After it had been whistled at and kicked about a dozen times, it learned that a whistle meant trouble. We then removed the specific stimulus — the stool. The whistle was blown, and it avoided the place as if there were a stool there.”

He advanced to a two-note whistle: One pitch was sounded before the robot touched an object, to associate it with avoidance. The other was sounded before it found its food, to associate it with appetite. “The effect of giving both notes was almost always disastrous; it went right off into the darkness on the right-hand side of the room and hovered round there for five minutes in a sort of sulk. It became irresponsive to stimulation and ran round in circles.”

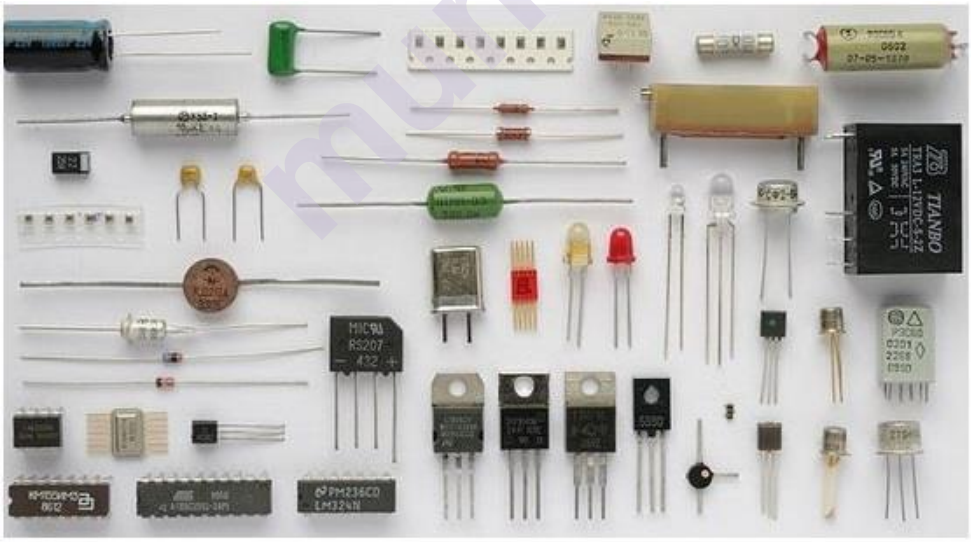
“As you would expect, there are only three ways of alleviating this condition. One of them is rest; in this case that was sufficient, it was left alone to play around in the dark until the effect of all the trauma had died down and it found its way home in the end. Another method is shock, to turn the circuits right off and start again with a clean bill. The most satisfactory method for my purpose is surgery, to dissect out the circuit.”

1.9 ANALOG ELECTRONIC CIRCUIT

Electronic Components

Similar to a brick that constructs a wall, a component is the basic brick of a circuit. A **Component** is a basic element that contributes for the development of an idea into a **circuit** for execution.

Each component has a few basic properties and the component behaves accordingly. It depends on the motto of the developer to use them for the construction of the intended circuit. The following image shows a few examples of electronic components that are used in different electronic circuits.



Just to gather an idea, let us look at the types of Components. They can either be **Active Components** or **Passive Components**.

Active Components

- Active Components are those which conduct upon providing some external energy.

- Active Components produce energy in the form of voltage or current.
- **Examples** − Diodes, Transistors, Transformers, etc.

Passive Components

- Passive components are those which start their operation once they are connected. No external energy is needed for their operation.
- Passive components store and maintain energy in the form of voltage or current.
- **Examples** − Resistors, Capacitors, Inductors, etc.

We also have another classification as **Linear** and **Non-Linear** elements.

Linear Components

- Linear elements or components are the ones that have linear relationship between current and voltage.
- The parameters of linear elements are not changed with respect to current and voltage.
- **Examples** − Diodes, Transistors, Transformers, etc.

Non-linear Components

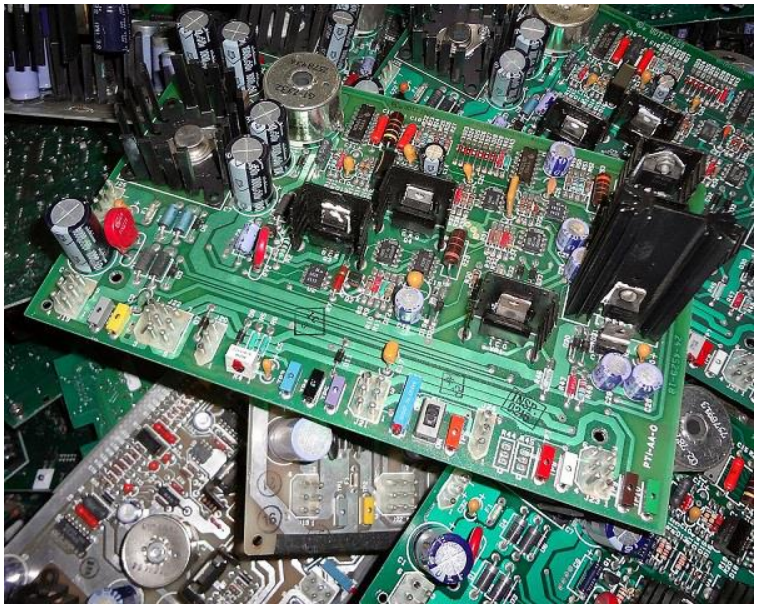
- Non-linear elements or components are the ones that have a non-linear relationship between current and voltage.
- The parameters of non-linear elements are changed with respect to current and voltage.
- **Examples** − Resistors, Capacitors, Inductors, etc.

These are the components intended for various purposes, which altogether can perform a preferred task for which they are built. Such a combination of different components is known as a **Circuit**.

Electronic Circuits

A certain number of components when connected on a purpose in a specific fashion makes a **circuit**. A circuit is a network of different components. There are different types of circuits.

The following image shows different types of electronic circuits. It shows Printed Circuit Boards which are a group of electronic circuits connected on a board.



Electronic circuits can be grouped under different categories depending upon their operation, connection, structure, etc. Let's discuss more about the types of Electronic Circuits.

Active Circuit

- A circuit that is build using Active components is called as **Active Circuit**.
- It usually contains a power source from which the circuit extracts more power and delivers it to the load.
- Additional Power is added to the output and hence output power is always greater than the input power applied.
- The power gain will always be greater than unity.

Passive Circuit

- A circuit that is build using Passive components is called as **Passive Circuit**.
- Even if it contains a power source, the circuit does not extract any power.
- Additional Power is not added to the output and hence output power is always less than the input power applied.
- The power gain will always be less than unity.

Electronic circuits can also be classified as **Analog, Digital**, or **Mixed**.

Analog Circuit

- An analog circuit can be one which has linear components in it. Hence it is a linear circuit.
- An analog circuit has analog signal inputs which are continuous range of voltages.

Digital Circuit

- A digital circuit can be one which has non-linear components in it. Hence it is a non-linear circuit.
- It can process digital signals only.
- A digital circuit has digital signal inputs which are discrete values.

Mixed Signal Circuit

- A mixed signal circuit can be one which has both linear and non-linear components in it. Hence it is called as a mixed signal circuit.
- These circuits consist of analog circuitry along with microprocessors to process the input.

Depending upon the type of connection, circuits can be classified as either **Series Circuit** or **Parallel Circuit**. A Series Circuit is one which is connected in series and a **parallel circuit** is one which has its components connected in parallel.

Now that we have a basic idea about electronic components, let us move on and discuss their purpose which will help us build better circuits for different applications. Whatever might be the purpose of an electronic circuit toprocess,tosend,toreceive,toanalyzetoprocess,tosend,toreceive,toan alyze, the process is carried out in the form of signals. In the next chapter, we will discuss the signals and the type of signals present in electronic circuits.

1.10 REACTIVE THEORY

A reactive robotic system tightly couples perception to action without the use of intervening abstract representations or time history. Purely reactive systems are at one extreme of the robotic systems spectrum. Reactive robotic systems have the following characteristics:

Behaviors serve as the basic building blocks for robotic actions. A behavior in these systems typically consists of a simple sensorimotor pair, with the sensory activity providing the necessary information to satisfy the applicability of a particular low-level motor reflex response.

Use of explicit abstract representational knowledge is avoided in the generation of a response. Purely reactive systems react directly to the world as it is sensed, avoiding the need for intervening abstract representational knowledge.

This obviates the need for an ``accurate'' (in the sense of true-to-life) world model. This could be of particular value in

- dynamic
- hazardous
- complex

worlds - wherever an accurate model would be difficult (time-consuming) or impossible to calculate and/or rapid response is required. (Constructing abstract world models is a time-consuming and error-prone process and

thus reduces the potential correctness of a robot's action in all but the most predictable worlds.[RA])

Animal models of behavior often serve as a basis for these systems. [RA]

Biology provides an existence proof that many of the tasks we would like our robots to undertake are indeed doable.

the biological sciences, such as neuroscience, ethology, and psychology, have elucidated various mechanisms and models that may be useful in ``operationalizing'' robots.

These systems are inherently modular from a software design perspective. This enables a reactive robotic system designer to expand his robot's competency by adding new behaviors without redesigning or discarding the old.

This accretion of capabilities over time and resultant reusability is very useful for constructing increasingly more complex robotic systems.

1.11 BRAITENBERG'S VEHICLE

When looking at mechanisms with cognitive functionality (and artificial intelligence in general) it is useful to begin with the simplest cases.

Braitenberg `vehicles' are very simple mobile machines that use basic sensory-motor connections to produce seemingly cognitive behaviors.

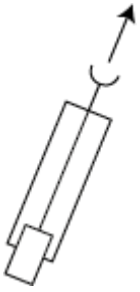
Braitenberg's approach

`We will talk about machines with very simple internal structure ... when we look at these machines or vehicles as if they were animals in a natural environment ... we will be tempted, then, to use psychological language in describing their behavior. And yet we know very well that there is nothing in these vehicles that we have not put in ourselves.'

This and all other quotes and figures are from the first five chapters of (Braitenberg, 1984)

Vehicle 1: approach

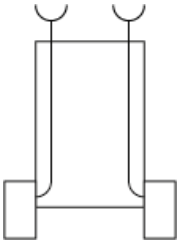
Vehicle 1: the simplest vehicle. The speed of the motor (rectangular box at the tail end) is controlled by a sensor (half circle on a stalk, at the front end). Motion is always forward, in the direction of the arrow, except for perturbations.



Vehicle 2a: fear

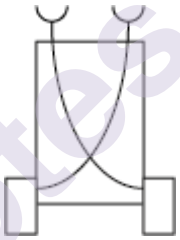
This vehicle spend more time in the places with less stimulation, and speeds up when exposed to more stimulation. If the stimulation is directly ahead, the vehicle may hit the source.

Otherwise, it will tend to turn away from the stimulation.

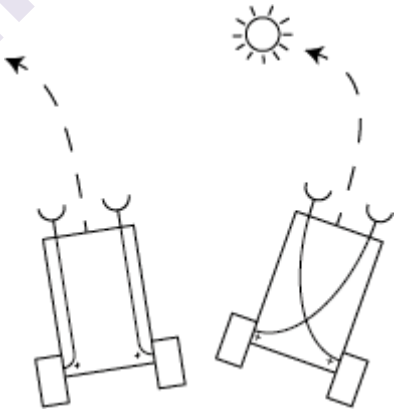


Vehicle 2b: aggression

If the sensor-motor connections are crossed, the behaviour changes. If the stimulation is directly ahead, the vehicle moves directly towards it as before. But, if the stimulation is to one side, the vehicle will tend to veer towards it with increasing speed.



Fear and aggression



Anthropomorphic interpretation

Braitenberg illustrates the potential for over-blown interpretation.

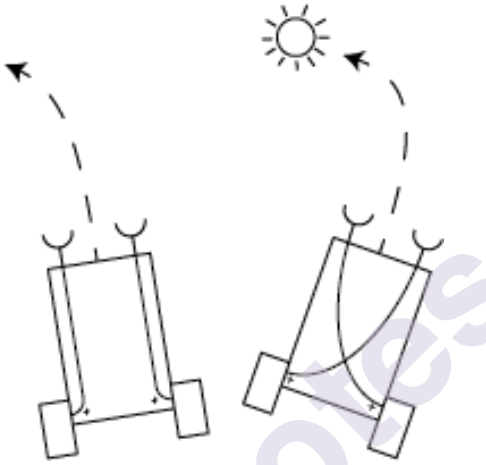
`Let Vehicles 2a and 2b move around in their world for a while and watch them. Their characters are quite opposite. Both DISLIKE sources. But 2a becomes restless in their vicinity and tends to avoid them, escaping until it safely reaches a place where the influence of the source is scarcely felt. Vehicle 2a is a COWARD, you would say.

Not so Vehicle 2b. It, too, is excited by the presence of sources, but resolutely turns toward them and hits them with high velocity, as if it wanted to destroy them. Vehicle 2b is AGGRESSIVE, obviously.'

Inhibition

`The violence of Vehicle 2b, no less than the cowardice of the companion 2a, are traits that call for improvement. ... What comes to mind is to introduce some inhibition in the connections between the sensors and the motors, switching the sign of the influence from positive to negative. This will make the motor slow down when the corresponding sensor is activiated.'

Vehicles 2a and 2b



Vehicle 3a: love

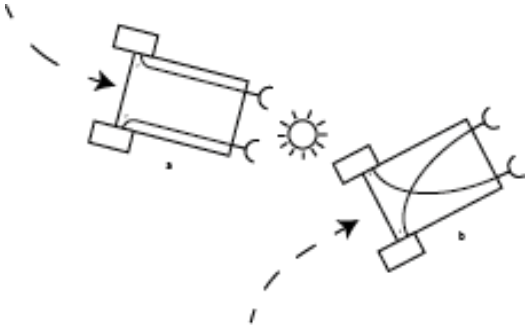
Making the connections of vehicles 2a and 2b *inhibitory* produces vehicles 3a and 3b.

These show completely different behaviour.

Approaching the stimulation, Vehicle 3a will orient towards it and come to rest facing it.

Vehicle 3b on the other hand will come to rest facing away from the stimulation.

Love v. exploration

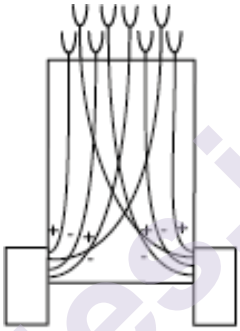


Optimistic interpretation

`You will have no difficulty giving names to this sort of behavior. These vehicles LIKE the source, you will say, but in different ways. Vehicle 3a LOVES it in a permanent way, staying close by in quiet admiration from the time it spots the source to all future time. Vehicle 3b, on the other hand, is an EXPLORER. It likes the nearby source all right, but keeps an eye open for other, perhaps stronger sources, which it will sail to, given a chance, in order to find a more permanent and gratifying appeasement.'

Vehicle 3c: a system of values

`... not just one pair of sensors but four pairs, turned to different qualities fo the environment, say light, temoperature, oxygen, concentration, and amount of organic matter.'

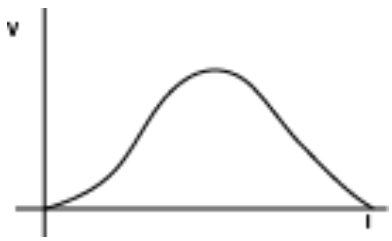


Expected behavior

Given appropriate connections, `this is a vehicle with really interesting behavior. It dislikes high temperature, turns away from hot places, and at the same time seems to dislike light bulbs with even greater passion, since it turns towards them and destroys them... You cannot help admitting that Vehicle 3c has a system of VALUES, and, come to think of it, KNOWLEDGE'

Vehicle 4: special tastes

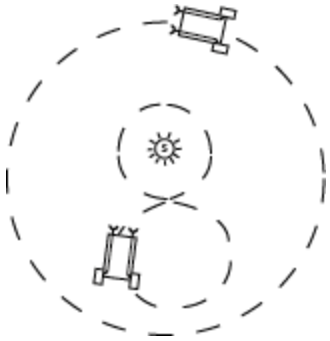
`Let us consider the following improvement. The activation of a certain sensor will make the corresponding motor run faster but only up to a point, where the speed of the motor reaches a maximum. Beyond this point, the speed will decrease again.'



`A 4a vehicle might navigate towards a source (as Vehicle 2b would) and then turn away when the stimulus becomes too strong, circle back and then turn away again over and over again'

Effects of non-linear, non-monotonic connections

`You will have a hard time imagining the variety of behavior displayed by vehicles of brand 4a.'



1.12 SUMMARY

- Robotics is the intersection of science, engineering and technology that produces machines, called robots, that substitute for (or replicate) human actions.
- A robot is the product of the robotics field, where programmable machines are built that can assist humans or mimic human actions.
- Robots all consist of some sort of mechanical construction. The mechanical aspect of a robot helps it complete tasks in the environment for which it's designed.
- Robots contain at least some level of computer programming. Without a set of code telling it what to do, a robot would just be another piece of simple machinery.
- Mechanical bots come in all shapes and sizes to efficiently carry out the task for which they are designed. All robots vary in design, functionality and degree of autonomy
- Pre-programmed robots operate in a controlled environment where they do simple, monotonous tasks.
- Humanoid robots are robots that look like and/or mimic human behavior. These robots usually perform human-like activities (like running, jumping and carrying objects), and are sometimes designed to look like us, even having human faces and expressions.
- Autonomous robots operate independently of human operators.
- Teleoperated robots are semi-autonomous bots that use a wireless network to enable human control from a safe distance.
- Augmenting robots either enhance current human capabilities or replace the capabilities a human may have lost.
- "Cybernetics" comes from a Greek word meaning "the art of steering". Cybernetics is about having a goal and taking action to achieve that goal.

1.13 QUESTIONS

1. WHAT IS A ROBOT?
2. What are Characteristics of Robot?
3. Write a short note on History of Robots
4. What is control theory?
5. What are types of Robot?
6. What does the word “cybernetics” mean?
7. When did cybernetics begin?
8. What's the connection between “cybernetics” and “cyberspace”?
9. Write a short note on Grey Walter Tortoise.
10. What is Reactive Theory?
11. Write a short note on Braitenberg’s Vehicle .

1.14 REFERENCE FOR FURTHER READING

- https://www.nasa.gov/audience/forstudents/k-4/stories/nasa-knows/what_is_robotics_k4.html
- https://www.geeksforgeeks.org/robotics-introduction/
- https://ewh.ieee.org/sb/bangalore/nhce/STORY.pdf
- https://dataunbox.com/control-theory-
- https://ipvs.informatik.uni-stuttgart.de/mlr/marc/teaching/13-Robotics/08-controlTheory.pdf
- https://pangaro.com/definition-cybernetics.html
- http://cs.oswego.edu/~odendahl/coursework/csc338/notes/paradigms/reactive.html

❄❄❄❄❄❄❄

2

ARTIFICIAL INTELLIGENCE

Unit Structure

2.0 OBJECTIVE

Here we will discuss about AI. And how it implemented in Robotics. Then different components of Robotics

2.1 INTRODUCTION

Artificial intelligence leverages computers and machines to mimic the problem-solving and decision-making capabilities of the human mind.

2.2 WHAT IS ARTIFICIAL INTELLIGENCE?

While a number of definitions of artificial intelligence (AI) have surfaced over the last few decades, John McCarthy offers the following definition in this 2004 paper (PDF, 106 KB) (link resides outside IBM), " It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable."

However, decades before this definition, the birth of the artificial intelligence conversation was denoted by Alan Turing's seminal work, "Computing Machinery and Intelligence" (PDF, 89.8 KB) (link resides outside of IBM), which was published in 1950. In this paper, Turing, often referred to as the "father of computer science", asks the following

question, "Can machines think?" From there, he offers a test, now famously known as the "Turing Test", where a human interrogator would try to distinguish between a computer and human text response. While this test has undergone much scrutiny since its publish, it remains an important part of the history of AI as well as an ongoing concept within philosophy as it utilizes ideas around linguistics.

Stuart Russell and Peter Norvig then proceeded to publish, Artificial Intelligence: A Modern Approach (link resides outside IBM), becoming one of the leading textbooks in the study of AI. In it, they delve into four potential goals or definitions of AI, which differentiates computer systems on the basis of rationality and thinking vs. acting:

Human approach:

- Systems that think like humans
- Systems that act like humans

Ideal approach:

- Systems that think rationally
- Systems that act rationally

Alan Turing's definition would have fallen under the category of "systems that act like humans."

At its simplest form, artificial intelligence is a field, which combines computer science and robust datasets, to enable problem-solving. It also encompasses sub-fields of machine learning and deep learning, which are frequently mentioned in conjunction with artificial intelligence. These disciplines are comprised of AI algorithms which seek to create expert systems which make predictions or classifications based on input data.

Today, a lot of hype still surrounds AI development, which is expected of any new emerging technology in the market. As noted in Gartner's hype cycle (link resides outside IBM), product innovations like, self-driving cars and personal assistants, follow "a typical progression of innovation, from overenthusiasm through a period of disillusionment to an eventual understanding of the innovation's relevance and role in a market or domain." As Lex Fridman notes here (01:08:15) (link resides outside IBM) in his MIT lecture in 2019, we are at the peak of inflated expectations, approaching the trough of disillusionment.

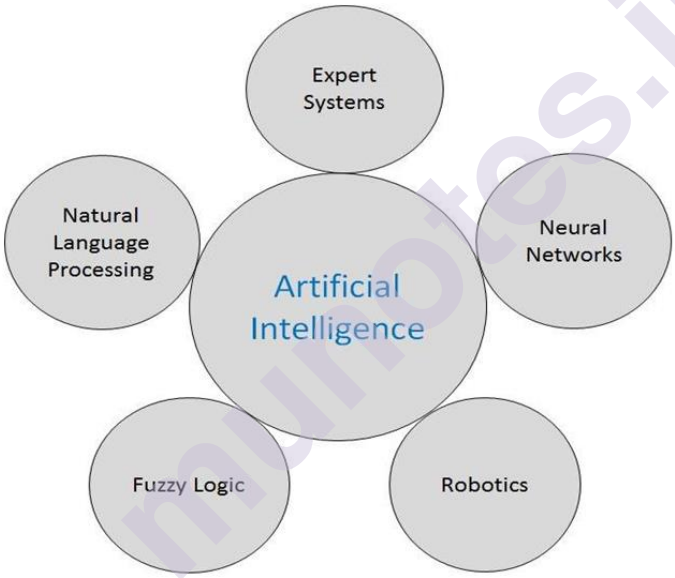
2.3 TYPES OF ARTIFICIAL INTELLIGENCE - WEAK AI VS. STRONG AI

Weak AI—also called Narrow AI or Artificial Narrow Intelligence (ANI)—is AI trained and focused to perform specific tasks. Weak AI drives most of the AI that surrounds us today. 'Narrow' might be a more accurate descriptor for this type of AI as it is anything but weak; it enables some very robust applications, such as Apple's Siri, Amazon's Alexa, IBM Watson, and autonomous vehicles.

Strong AI is made up of Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI). Artificial general intelligence (AGI), or general AI, is a theoretical form of AI where a machine would have an intelligence equaled to humans; it would have a self-aware consciousness that has the ability to solve problems, learn, and plan for the future. Artificial Super Intelligence (ASI)—also known as superintelligence—would surpass the intelligence and ability of the human brain. While strong AI is still entirely theoretical with no practical examples in use today, that doesn't mean AI researchers aren't also exploring its development. In the meantime, the best examples of ASI might be from science fiction, such as HAL, the superhuman, rogue computer assistant in 2001: A Space Odyssey.

2.4 ARTIFICIAL INTELLIGENCE RESEARCH AREAS

The working domain of artificial intelligence is huge in width and breadth. Therefore before proceeding further considers the prospering and common research areas in the domain of artificial intelligence are:-



- **Expert System -** In artificial intelligence, an expert system are used for solving complex problems by reasoning about knowledge, represented primarily by if-then rules rather than by conventional procedural code. In general, an expert system is a computer system that uses the decision-making capability of a human expert.
- **Neural Networks -** Neural networks are system of interconnected ?neurons? which exchange messages between each other. In machine learning artificial neural networks (ANNs) belongs to a family of model inspired by biological neural networks (the nervous system of animals, present inside a brain) and are used for approximate functions or estimate a large number of inputs which are generally unknown.

- **Robotics -** Robotics is a branch of Artificial Intelligence (AI), it is mainly composed of electrical engineering, mechanical engineering and computer science engineering for construction, designing and application of robots. Robotics is science of building or designing an application of robots. The aim of robotics is to design an efficient robot.

- **Fuzzy logic -** Fuzzy logic was introduced in 1965 as a proposal of fuzzy set theory. It is applied to various fields, from artificial intelligence to control theory. Fuzzy logic is a form of many-valued logic in which truth table values of variable may be real number between 0 and 1.

- **Natural Language Processing -** Natural language processing (NLP) is a method of communicating with an intelligent system by using a natural language such as English. The input and output of NLP system is speech and written text.

2.5 VISION BASED NAVIGATION

The architecture of a vision system is strongly related to the application it is meant to solve. Some systems are "stand-alone" machines designed to solve specific problems (e.g. measurement/identification), while others are integrated into a more complex framework that can include e.g. mechanical actuators, sensors etc. Nevertheless, all vision systems operate are characterized by these fundamental operations:

Image acquisition. The first and most important task of a vision system is to acquire an image, usually by means of light-sensitive sensor. This image can be a traditional 2-D image, or a 3-D points set, or an image sequence. A number of parameters can be configured in this phase, such as image triggering, camera exposure time, lens aperture, lighting geometry, and so on.

Feature extraction. In this phase, specific characteristics can be extrapolated from the image: lines, edges, angles, regions of interest (ROIs), as well as more complex features, such as motion tracking, shapes and textures.

Detection/segmentation. at this point of the process, the system must decide which information previously collected will be passed on up the chain for further elaboration.

High-level processing. The input at this point usually consists of a narrow set of data. The purpose of this last step can be to:

Classify objects or object's feature in a particular class

Verify that the input has the specifications required by the model or class

Measure/estimate/calculate specifics parameters as position or dimensions of object or object's features

2.5.1 Vision System

Machine Vision is the discipline that encompasses imaging technologies and methods to perform automatic inspection and analysis in various applications, such as verification, measurement, process control. A very common approach in machine vision is to provide turnkey vision solutions, i.e. complete systems that can be rapidly and easily configured for use in the field. A vision system is usually made up of every component needed to perform the intended task, such as optics, lighting, cameras and software. When designing and building a vision system, it is important to find the right balance between performance and cost to achieve the best result for the desired application.

Usually vision systems are designed to work in on-line applications, where they have an immediate impact on the manufacturing process (real-time systems). A classic example of this on-line concept is the possibility to instantly reject a product deemed non-compliant: the way this decision is made, as well as the object features being evaluated, defines different classes of vision systems.

2.5.2 Types of Vision System

Several types of vision systems are available on the market, each being characterized by a different level of flexibility, performance and cost. Vision systems can usually be divided into three classes: PC based, compact and smart camera based.

PC based. The classic machine vision system consists of an industrial computer that manages and communicates with all the peripheral devices, such as cameras and lighting, quickly analyzing the information via software. This solution provides high computing power and flexibility, but size and cost can be significant. PC based systems are recommended for very complex applications, where multiple inspection tasks must be carried out at a fast rate with high-performance hardware.

Compact. A “lighter” version of a PC based system is called a Compact vision system. Although it may require some tradeoff between performance and cost, it is often enough for less demanding applications. Compact vision systems usually include a graphics card that acquires and transfers the information to a separate peripheral (e.g. an industrial tablet or an external monitor). Sometimes, compact vision systems not only manage the first level input - lightning, camera and trigger inputs - but also have embedded first level inputs.

Smart Cameras based. The simplest and most affordable vision systems are based on smart or intelligent cameras, normally used in combination with standard optics (typically a fixed focal length lens) and lighting. Although typically recommended for simpler applications, they are very easy to set up and provide similar functionalities to classic vision systems in a very compact form factor.

2.5.3 Application

Vision systems can do many different things: measurement, identification, sorting, code reading, character recognition, robot guidance etc. They can easily interact with other machinery through different communication standards. Here below are some of the main application categories for a vision system:

Measurement. One of the most important uses of vision technology is to measure, at various degrees of accuracy, the critical dimensions of an object within pre-determined tolerances.

Optics, lighting and cameras must be coupled to effective software tools, since only robust subpixeling algorithms will allow to reach the accuracy often required in measurement applications (e.g. even down to 1 um).

Defect detection. Here various types of product defects have to be detected for cosmetic and/or safety reasons. Examples of cosmetic flaws are stains, spots, color clumps, scratches, tone variations, etc. while other surface and/or structural defects, such as cracks, dents, but also print errors etc. can have more severe consequences.

Verification. The third major aim of a vision system is checking that a product has been correctly manufactured, in a more general sense that goes beyond what previously described; i.e. checking the presence/absence of pills in a blister pack, the correct placement of a seal or the integrity of a printed label.

Expert Systems - There are various applications which integrate machine, special information and software to impart advising and reasoning. These systems provide explanation and advice to the users.

Gaming - AI plays major role in strategic games such as poker, chess, tic-tac-toe, etc. Using artificial intelligence the machine can think of large number of possible moves based on general knowledge.

Natural Language Processing - Using natural language processing it is possible to interact with a computer that can understand natural language spoken by humans.

Vision systems - These systems interpret, understand, and comprehend a visual input on the computer.

Intelligent Robots - Robots are designed for performing the tasks given by a human. They have sensors embedded to detect physical data from the outside environment such as heat, light, sound, pressure, etc. They have multiple sensors, efficient processors and large memory, to exhibit intelligence. In addition, they are capable to learn from their mistakes and they can easily adapt to the new environment.

2.6 TYPES OF ROBOT CONTROL

Four types of robot control

1. Point-to-point (PTP) control robot

2. Continuous-path (CP) control robot

3. Controlled-path robot

4. Stop-to-Stop

1. Point to Point Control Robot (PTP):

The PTP robot is capable of moving from one point to another point. The locations are recorded in the control memory.

PTP robots do not control the path to get from one point to the next point. Common applications include:

- Component insertion
- Spot welding
- hole drilling
- Machine loading and unloading
- Assembly operations

2. Continuous-Path Control Robot (CP):

The CP robot is capable of performing movements along the controlled path. With CP from one control, the robot can stop at any specified point along the controlled path.All the points along the path must be stored explicitly in the robot's control memory. Applications Straight-line motion is the simplest example for this type of robot. Some continuous-path controlled robots also have the capability to follow a smooth curve path that has been defined by the programmer.

In such cases the programmer manually moves the robot arm through the desired path and the controller unit stores a large number of individual point locations along the path in memory (teach-in).

Typical applications include:

a. spray painting
b. finishing
c. gluing

Arc welding operations

3. Controlled-Path Robot:

In controlled-path robots, the control equipment can generate paths of different geometry such as straight lines, circles, and interpolated curves with a high degree of accuracy.

Good accuracy can be obtained at any point along the specified path.

Only the start and finish points and the path definition function must be stored in the robot's control memory.

It is important to mention that all controlled-path robots have a servo capability to correct their path.

4. Stop-to-Stop:

It is open loop system

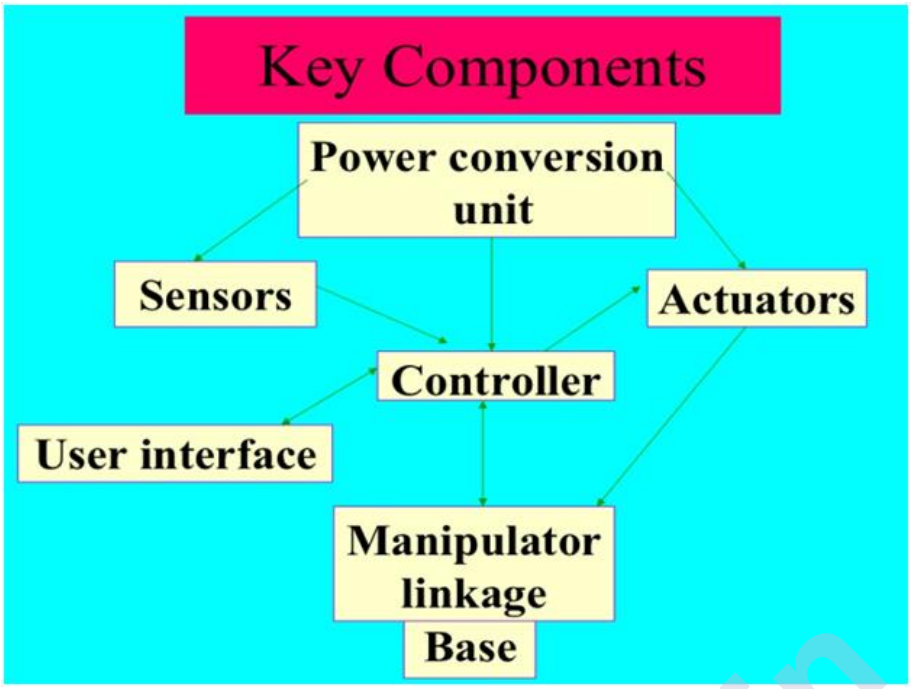
Position and velocity unknown to controller

On/off commands stored as valve states

End travel set by mechanical

2.7 ROBOT COMPONENTS

Consider the robot structure showing different components of robots are:



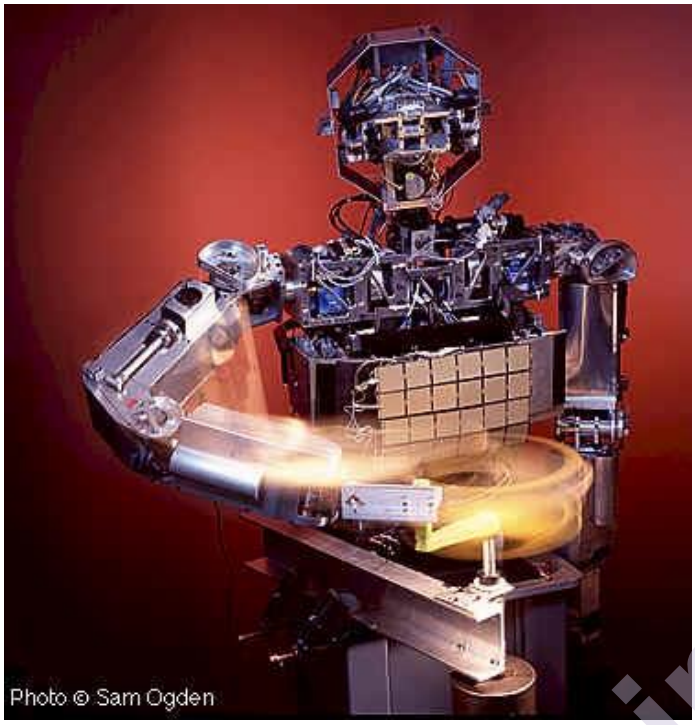
- **Power Supply -** The working power to the robot is provided by batteries, hydraulic, solar power, or pneumatic power sources.
- **Actuators -** Actuators are the energy conversion device used inside a robot. The major function of actuators is to convert energy into movement.
- **Electric motors (DC/AC)-** Motors are electromechanical component used for converting electrical energy into its equivalent mechanical energy. In robots motors are used for providing rotational movement.
- **Sensors -** Sensors provide real time information on the task environment. Robots are equipped with tactile sensor it imitates the mechanical properties of touch receptors of human fingerprints and a vision sensor is used for computing the depth in the environment.
- **Controller -** Controller is a part of robot that coordinates all motion of the mechanical system. It also receives an input from immediate environment through various sensors. The heart of robot's controller is a microprocessor linked with the input/output and monitoring device. The command issued by the controller activates the motion control mechanism, consisting of various controller, actuators and amplifier.

2.8 EFFECTORS

An effector is any device that affects the environment. Robots control their effectors, which are also known as end effectors. Effectors include legs, wheels, arms, fingers, wings and fins. Controllers cause the effectors to produce desired effects on the environment. An actuator is the actual mechanism that enables the effector to execute an action. Actuators

typically include electric motors, hydraulic or pneumatic cylinders, etc. The terms effector and actuator are often used interchangeably to mean "whatever makes the robot take an action." This is not really proper use. Actuators and effectos are not the same thing. And we'll try to be more precise in the class. Most simple actuators control a single degree of freedom, i.e., a single motion (e.g., up-down, left-right, in-out, etc.). A motor shaft controls one rotational degree of freedom, for example. A sliding part on a plotter controls one translational degree of freedom. How many degrees of freedom (DOF) a robot has is going to be very important in determining how it can affect its world, and therefore how well, if at all, it can accomplish its task. Just as we said many times before that sensors must be matched to the robot's task, similarly, effectors must be well matched to the robot's task also.

In general, a free body in space as 6 DOF: three for translation (x,y,z), and three for orientation/rotation (roll, pitch, and yaw). We'll go back to DOF in a bit. You need to know, for a given effector (and actuator/s), how many DOF are available to the robot, as well as how many total DOF any given robot has. If there is an actuator for every DOF, then all of the DOF are controllable. Usually not all DOF are controllable, which makes robot control harder. A car has 3 DOF: position (x,y) and orientation (theta). But only 2 DOF are controllable: driving: through the gas pedal and the forward-reverse gear; steering: through the steering wheel. Since there are more DOF than are controllable, there are motions that cannot be done, like moving sideways (that's why parallel parking is hard). We need to make a distinction between what an actuator does (e.g., pushing the gas pedal) and what the robot does as a result (moving forward). A car can get to any 2D position but it may have to follow a very complicated trajectory. Parallel parking requires a discontinuous trajectory w.r.t. velocity, i.e., the car has to stop and go. When the number of controllable DOF is equal to the total number of DOF on a robot, it is holonomic(for more information about holonomic). If the number of controllable DOF is smaller than total DOF, the robot is non-holonomic. If the number of controllable DOF is larger than the total DOF, the robot is redundant. A human arm has 7 DOF (3 in the shoulder, 1 in the elbow, 3 in the wrist), all of which can be controlled. A free object in 3D space (e.g., the hand, the finger tip) can have at most 6 DOF! So there are redundant ways of putting the hand at a particular position in 3D space. This is the core of why manipulations is very hard!



Two basic ways of using effectors:

- to move the robot around =>locomotion
- to move other object around =>manipulation

These divide robotics into two mostly separate categories:

- mobile robotics
- manipulator robotics

Mobility end effectors are discussed in more detail in the mobility section of this web site.

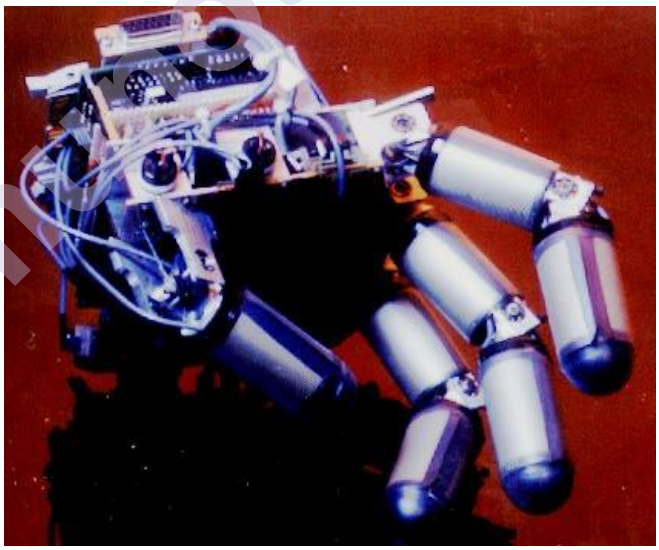
In contrast to locomotion, where the body of the robot is moved to get to a particular position and orientation, a manipulator moves itself typically to get the *end effector* (e.g., the hand, the finger, the fingertip) to the desired 3D position and orientation. So imagine having to touch a specific point in 3D space with the tip of your index finger; that's what a typical manipulator has to do. Of course, largely manipulators need to grasp and move objects, but those tasks are extensions of the basic reaching above. The challenge is to get there efficiently and safely. Because the end effector is attached to the whole arm, we have to worry about the whole arm; the arm must move so that it does not try to violate its own *joint limits* and it must not hit itself or the rest of the robot, or any other obstacles in the environment. Thus, doing autonomous manipulation is very challenging. Manipulation was first used in tele-operation, where human operators would move artificial arms to handle hazardous materials. It turned out that it was quite difficult for human operators to learn how to tele-operate complicated arms (such as duplicates of human arms, with 7 DOF). One alternative today is to put the human arm into an exo-skeleton (see lecture 1), in order to make the control more direct.

Using joy-sticks, for example, is much harder for high DOF. Why is this so hard? Because even as we saw with locomotion, there is typically no direct and obvious link between what the effector needs to do in physical space and what the actuator does to move it. In general, the correspondence between actuator motion and the resulting effector motion is called *kinematics*. In order to control a manipulator, we have to know its kinematics (what is attached to what, how many joints there are, how many DOF for each joint, etc.). We can formalize all of this mathematically, and get an equation which will tell us how to convert from, say, angles in each of the joints, to the Cartesian positions of the end effector/point. This conversion from one to the other is called computing the manipulator kinematics and *inverse kinematics*.

The process of converting the Cartesian (x,y,z) position into a set of joint angles for the arm (thetas) is called inverse kinematics. Kinematics are the rules of what is attached to what, the body structure. Inverse kinematics is computationally intense. And the problem is even harder if the manipulator (the arm) is redundant.

Manipulation involves

- trajectory planning (over time)
- inverse kinematics
- inverse dynamics
- dealing with redundancy



Manipulators are effectors. Joints connect parts of manipulators. The most common joint types are:

- rotary (rotation around a fixed axis)
- prismatic (linear movement)

These joints provide the DOF for an effector, so they are planned carefully.

Robot manipulators can have one or more of each of those joints. Now recall that any free body has 6 DOF; that means in order to get the robot's end effector to an arbitrary position and orientation, the robot requires a minimum of 6 joints. As it turns out, the human arm (not counting the hand!) has 7 DOF. That's sufficient for reaching any point with the hand, and it is also redundant, meaning that there are multiple ways in which any point can be reached. This is good news and bad news; the fact that there are multiple solutions means that there is a larger space to search through to find the best solution. Now consider end effectors. They can be simple pointers (i.e., a stick), simple 2D grippers, screwdrivers for attaching tools (like welding guns, sprayer, etc.), or can be as complex as the human hand, with variable numbers of fingers and joints in the fingers. Problems like reaching and grasping in manipulation constitute entire subareas of robotics and AI. Issues include: finding grasp-points (COG, friction, etc.); force/strength of grasp; compliance (e.g., in sliding, maintaining contact with a surface); dynamic tasks (e.g., juggling, catching). Other types of manipulation, such as carefully controlling force, as in grasping fragile objects and maintaining contact with a surface (so-called *compliant motion*), are also being actively researched. Finally, dynamic manipulation tasks, such as juggling, throwing, catching, etc., are already being demonstrated on robot arms.

Having talked about navigation and manipulation, think about what types of sensors (external and proprioceptive) would be useful for these general robotic tasks. *Proprioceptive* sensors sense the robot's actuators (e.g., shaft encoders, joint angle sensors, etc.); they sense the robot's own movements. You can think of them as perceiving internal state instead of external state. External sensors are helpful but not necessary or as commonly used.

2.8.1 What is Robot End Effector

A Robot End effector or a Robotic Gripper is a mechanical part attached to the end of the robot arm hardware that is intended for direct interaction of environment and adjacent. The purpose of this mechanical part is subject to the robot's application in the world. In the case of a serial manipulator, the robotic gripper usually lies in the hardware's last link. The end effector is also called Gripper, and it is analogous to the hand of a human body.

This is different from the wheels or legs of mobile robots in that the latter is used to facilitate the mobility of the robots only. But robot end effector is application-specific in nature and contains varied designs to accommodate varying purposes for manipulating an object.

Robot End Effector Design

End Effector position

The end effector is generally designed to be attached at the end of the robot manipulator. Hence the term 'End of Arm Tooling' is also used in appropriate cases. This facilitates direct contact of the end effector with the environment. Therefore, manipulation of an object takes place through the gripper in accordance with the application of the robot. These are often custom-tailored to fit special processes requirements, other than the ones that are generally used.

Types of End Effectors in robots

Robot End-effector is classified into four general types based on physical effect usage to achieve a stable grasp amongst the gripper and the object to be grabbed.

a) Impactive Gripper: These are jaws or claws that exhibit physical grasping directly impacting the object.

b) Ingressive Gripper: These are sharp-point surfaces like pins, needles or hackles which exhibit physical penetration inside the surface of the object. Applications can be seen in textile, carbon and glass fibre handling.

c) Astrictive Gripper: These grippers apply attractive forces to the surface of the object using a vacuum, magneto- or electroadhesion.

d) Contigutive Gripper: These grippers require direct contact to exhibit adhesion, such as glue, surface tension or freezing.

Robot End Effector gripper | Different types of end effectors

Some grippers are categorized by their principle of operation. A few of them have been briefly discussed below:

Bernoulli Gripper

It applies Bernoulli's principle by exploiting the airflow between the gripper and the object to be grasped. This generates a lifting force that brings the gripper and the object closer without letting them come into direct contact with each other. Hence, Bernoulli Gripper is a contcatless gripper. The applications of Bernoulli gripper can be seen in photovoltaic cell handling, semiconductor manufacturing industries, and in the textile industry.

Electrostatic Gripper

It exploits the characteristics of electrostatic charge by utilizing a charge-difference between the gripper and the object. The gripper itself usually activates this charge-difference.

Van der Waals Gripper

Van der Waals gripper exploits the low electrostatic force amongst the gripper and the elemental objects.

Capillary Gripper

Capillary grippers utilize a liquid meniscus's surface tension between the gripper and the object towards its orientation and grasping.

Cryogenic Gripper

Cryogenic grippers create ice by freezing a small amount of liquid, which is then used to supply the obligatory force to lift and grip the object. The application of cryogenic gripper can be seen in food handling and in textile grasping.

Ultrasonic Gripper

Ultrasonic gripper are complex in nature than the above categories, which exploit pressure standing waves to levitate up a part and enclose it at a certain level. Lifting can be seen both at the micro-level and the macro level. Micro-level levitation is evident in screw and gasket-handling. In contrast, macro-scale lifting can be seen in a photovoltic cell or Si substrate handling and in laser source.

Intrusive Gripper

This a specific category of jaw grippers that utilizes friction force for grasping objects. One example of an intrusive gripper is that of a needle gripper. These are termed intrusive grippers because they exploit both frictions and enclosing characteristic as that of standard mechanical grippers.

The most typical type of mechanical gripper is multi-fingered grippers containing two, three or even five fingers.

The end effectors can be widely used in the tooling industry for applications like spot-welding in an assembly, spray-painting to conform uniformity in the application of paint, and other situations where safeguarding human interests become essential. Surgical robots, too, have end effectors that are customized as per the requirements of the procedure.

How do End Effectors work?

A robot end effector is basically the business end of the robot. If it is not there, a robot is mostly of no use because it is devoid of the equipment that performs the main function in order to serve a particular purpose. For example, an articulated robotic arm is usually programmed to a reach particular location within its workspace. Still, without an end effector's availability, it cannot perform the operation that it is assigned to, thereby making its existence redundant.

Although this main equipment for tooling is custom designed for every purpose, the basic underlying working principle remains more or less the same. Hence it is of utmost importance that we understand how a robot end effector works. This will not only aid us in the design process of the equipment but also help us in choosing the right end effector for our purpose. The robotic gripper is physically mounted on the wrist of the manipulator, followed by the attachment of the power connections. The power connections can be hydraulic, pneumatic or electric in nature.

The main force component is generated at the base link, which produces motion. This motion is then transferred link by link until the robotic manipulator's extreme periphery, where the gripper is attached. This case is valid if a single actuator powers the base's robotic manipulator. But it can also lead to malfunctioning and structural failure even with the slightest deviation from the force limits of the individual linkages.

Advancement in engineering with the usage of indigenous actuators at every joint has produced more flexibility towards using a robotic manipulator. Every link can generate more power individually, and the robot end effector can exploit the complete power from the actuator attached at the wrist. This enables the end effector to lift heavier objects and better grasp unstructured environments.

Robot Drive Systems and End Effectors

The robot drive systems are responsible for supplying power for the whole robot's operation. The robot's speed, load-carrying capability, and efficiency are all determined by the drive mechanism. In order for the manipulator's body, arm, gesture, and wrist to execute the expected move, the movements of the individual joints must be properly controlled. The drive device that powers the robot does this job.

The most commonly used robot drive systems in industrial applications are discussed below.

Electric Drive

Electric drive systems can move robots at high speeds or with high electricity. This type of robot can be operated using either DC servo motors or DC stepping motors. It can be used with both rotational and linear joints. Small robots and precise systems can benefit from the electric drive system. Most notably, it has improved precision and consistency. This device does have one disadvantage: it is significantly more expensive. The Maker 110 robot is an example of this kind of drive mechanism.

Hydraulic Drive

Hydraulic drive systems are designed especially for massive robots. It is capable of deliver more power or speed than electric drive systems. Both linear and rotational joint might benefit from this drive mechanism. Rotary vane actuators generate rotary motions, while the hydraulic pistons produce linear motion. The most important downside of this drive is hydraulic oil leakage. 'Unimate 2000 series robots' is an example of a hydraulic drive system robot.

Pneumatic Drive | Pneumatic End Effector

Pneumatic drive systems are particularly well-suited to small robots with < 5o freedom. It has the potential to have high precision and speed. The rotary actuators can act on this drive mechanism to achieve rotary movements. The piston may also be used to provide translational gestures for sliding joints. As compared to hydraulic drive, this mechanism is less expensive. This method's downside is that it would not be suitable for quicker operations.

Electric and hydraulic drive systems are the most widely used two types of drive systems. A detailed discussion can be found here.

End Effector Force

Gripper Mechanism

Various forces are working throughout the body of a robotic manipulator. The dominating force in this list is the friction force because that is what determines how hard or soft the grip should be to prevent any possible damage to the object.

The grip of the end effector should be strong and flexible enough to withstand the weight of the object and also handle the motion and acceleration produced by the continuous movement of the object. Hence it is imperative to calculate the amount of force required by the gripper to grasp an object.

The formula to find the force required by the robot end effector for the necessary grip on an object, the following formula is used:

$$F = \frac{ma}{\mu n}$$

where F= force required to grip the object,

m= mass of the object,

a= acceleration of the object,

μ= coefficient of friction,

n= number of fingers in the gripper

The above equation is more of a generalized form and thus deemed incomplete in a variety of situations. To make it fit for a more realistic environment, another term is introduced that can be seen in the modified equation below. This will take care of the fluctuations in the force of gravitation that occur concerning the direction of the movement. For example, the upward movement of the object against gravity requires more force in the gripper than the downward motion of the object towards gravity.

$$F = \frac{m(a+g)}{\mu n}$$

Here, g is acceleration because of gravity, and a is the acceleration because of object movement.

A task-related grip criterion can be used to pick grasps that are most suitable for fulfilling basic task specifications for certain visually interactive manipulation tasks, such as writing and screwdriver handling. Several task-oriented grasp consistency criteria have been proposed to aid in the evaluation of a strong grasp that meets the specifications of the task.

Is the Robotic End Effector multi functional?

Robotic gripper can perform more than one task through design and manufacturing. For example, household robots are aimed at helping old and disabled people or anybody with restricted mobility, for that matter. Hence they must be capable of mapping the environment, move to desired locations and also grab the necessary objects.

On the other hand, industrial robotic manipulators used in the automation industry can have end effectors capable of grasping and picking up objects. That also can be used as a piece of tooling equipment. The ‘end of arm tooling’ is highly justified in these cases because the robot end effector serves the purpose literally depicted by the name.

In the case of a surgical robot, the robot end effector is custom designed and manufactured to pick up the surgical equipments from desired locations, manipulate them at the region being operated and also perform the actual procedure using those instruments.

Hence, a single robotic gripper can be designed and manufactured for multiple tasks and operations through thorough research and careful study of the precise movements to be generated.

2.12 SUMMARY

- Artificial intelligence leverages computers and machines to mimic the problem-solving and decision-making capabilities of the human mind.
- At its simplest form, artificial intelligence is a field, which combines computer science and robust datasets, to enable problem-solving.
- Weak AI—also called Narrow AI or Artificial Narrow Intelligence (ANI)—is AI trained and focused to perform specific tasks.
- Strong AI is made up of Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI).
- Machine Vision is the discipline that encompasses imaging technologies and methods to perform automatic inspection and analysis in various applications, such as verification, measurement, process control.
- The PTP robot is capable of moving from one point to another point. The locations are recorded in the control memory.
- The CP robot is capable of performing movements along the controlled path. With CP from one control, the robot can stop at any specified point along the controlled path.
- In controlled-path robots, the control equipment can generate paths of different geometry such as straight lines, circles, and interpolated curves with a high degree of accuracy.
- An effector is any device that affects the environment. Robots control their effectors, which are also known as end effectors. Effectors include legs, wheels, arms, fingers, wings and fins.

2.13 QUESTIONS

1. Define AI.
2. What are the types of AI?
3. Write a short note on Vision Based Navigation
4. What are Types of Vision System?
5. What are application of Vision Sysytem?
6. What are Types of Robot Control?
7. Write a short note on Robot Components.
8. Write a short note on Effectors.

2.14 REFERENCE FOR FURTHER READING

- https://www.ibm.com/cloud/learn/what-is-artificial-intelligence
- https://www.opto-e.com/basics/types-of-vision-systems
- https://www.brainkart.com/article/Four-types-of-robot-control_5131/#:~:text=%20%20%201%20Point-to-
  http://electronicsteacher.com/robotics/robotics-technology/effectors.php
- https://lambdageeks.com/robot-end-effector/

❄❄❄❄❄❄❄

3

ACTUATOR

Unit Structure

3.0 OBJECTIVE

From this chapter we will get knowledge of Actuators and its type. Why it is fundamentals of robotics and importance.

3.1 INTRODUCTION

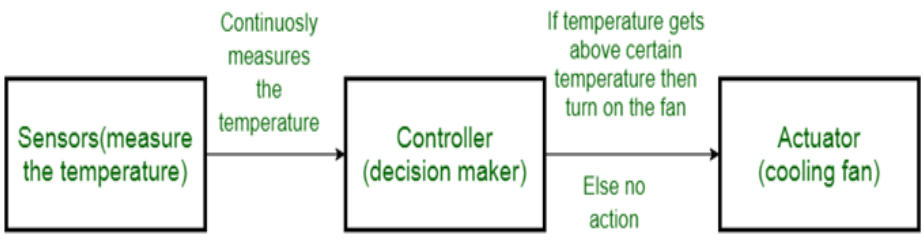
In IoT device is made up of a Physical object (“thing”) + Controller (“brain”) + Sensors + Actuators + Networks (Internet). An actuator is a machine component or system that moves or controls the mechanism or the system. Sensors in the device sense the environment, then control signals are generated for the actuators according to the actions needed to perform.

A servo motor is an example of an actuator. They are linear or rotatory actuators, can move to a given specified angular or linear position. We can use servo motors for IoT applications and make the motor rotate to 90 degrees, 180 degrees, etc., as per our need.

3.2 DEFINATION

Actuators, also known as drives, are mechanisms for getting robots to move. Most actuators are powered by pneumatics (air pressure), hydraulics (fluid pressure), or motors (electric current). Most actuation uses electromagnetic motors and gears but there have been frequent uses of other forms of actuation including NiTinOL"muscle-wires" and inexpensive Radio Control servos. To get a motor under computer control, different motor types and actuator types are used. Some of the motor types are Synchronous, Stepper, AC servo, Brushless DC servo, and Brushed DC servo. Radio Control servos for model airplanes, cars and other vehicles are light, rugged, cheap and fairly easy to interface. Some of the units can provide very high torque speed. A Radio Control servo can be controlled from a parallel port. With one of the PC's internal timers cranked up, it is possible to control eight servos from a common parallel port with nothing but a simple interrupt service routine and a cable. In fact, power can be pulled from the disk drive power connector and the PC can run all servos directly with no additional hardware. The only down side is that the PC wastes some processing power servicing the interrupt handler.

The following diagram shows what actuators do, the controller directs the actuator based on the sensor data to do the work.



3.3 TYPES OF ACTUATORS :

1. Hydraulic Actuators –

A hydraulic actuator uses hydraulic power to perform a mechanical operation. They are actuated by a cylinder or fluid motor. The mechanical motion is converted to rotary, linear, or oscillatory motion, according to the need of the IoT device. Ex- construction equipment uses hydraulic actuators because hydraulic actuators can generate a large amount of force.

Advantages :

- Hydraulic actuators can produce a large magnitude of force and high speed.
- Used in welding, clamping, etc.
- Used for lowering or raising the vehicles in car transport carriers.

Disadvantages :

- Hydraulic fluid leaks can cause efficiency loss and issues of cleaning.
- It is expensive.
- It requires noise reduction equipment, heat exchangers, and high maintenance systems.

2. Pneumatic Actuators –

A pneumatic actuator uses energy formed by vacuum or compressed air at high pressure to convert into either linear or rotary motion. Example- Used in robotics, use sensors that work like human fingers by using compressed air.

Advantages :

- They are a low-cost option and are used at extreme temperatures where using air is a safer option than chemicals.
- They need low maintenance, are durable, and have a long operational life.
- It is very quick in starting and stopping the motion.

Disadvantages :

- Loss of pressure can make it less efficient.
- The air compressor should be running continuously.
- Air can be polluted, and it needs maintenance.

3. Electrical Actuators –

An electric actuator uses electrical energy, is usually actuated by a motor that converts electrical energy into mechanical torque. An example of an electric actuator is a solenoid based electric bell.

Advantages :

- It has many applications in various industries as it can automate industrial valves.
- It produces less noise and is safe to use since there are no fluid leakages.
- It can be re-programmed and it provides the highest control precision positioning.

Disadvantages :

- It is expensive.
- It depends a lot on environmental conditions.

Other actuators are –

- **Thermal/Magnetic Actuators –**

  These are actuated by thermal or mechanical energy. Shape Memory Alloys (SMAs) or Magnetic Shape-Memory Alloys (MSMAs) are used by these actuators. An example of a thermal/magnetic actuator can be a piezo motor using SMA.

- **Mechanical Actuators –**

  A mechanical actuator executes movement by converting rotary motion into linear motion. It involves pulleys, chains, gears, rails, and other devices to operate. Example – A crankshaft.

  - Soft Actuators
  - Shape Memory Polymers
  - Light Activated Polymers
  - With the expanding world of IoT, sensors and actuators will find more usage in commercial and domestic applications along with the pre-existing use in industry.

3.4 DESIGN CONCEPT

The design is aimed at storing and releasing energy with high efficiency while providing controllability and keeping the weight and complexity small. The actuator provides a controllable joint torque by means of the variable stiffness accomplished with a flexible cantilever beam. A roller is mounted on a pulley-driven shuttle used to vary the point of application of

the force on the beam. This changes beam stiffness and consequently controls the joint moment. Due to beam elasticity, the mechanism can store and return potential energy.

A small electric motor and a servo drive with regenerative electronics are used to shift the contact roller. A supercapacitor is used to deliver or recover the electric energy that is converted by the DC motor/generator from the elastic potential energy of the beam. Supercapacitors are better suited as power-storage elements than batteries due to their high power density and ability to accept large charging currents . The control system for this actuator, and its energy storage and release capabilities, are described in Reference.

Figure 1 shows how the proposed VSA could be used in transfemoral prosthesis. A pair of twin swivelling rollers are used to deflect the cantilever beam, and roller position can be shifted by the motor-driven pulley system. As detailed below, a controllable knee moment is obtained by modulating roller position. With proper control, the design offers the opportunity to minimize energy consumption while maintaining normal gait patterns.

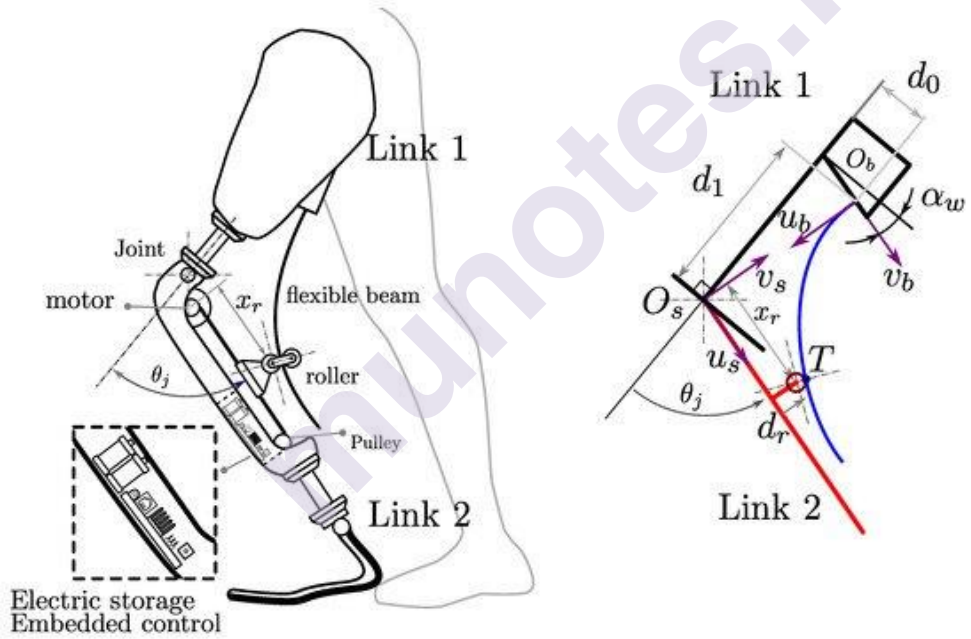


Figure 1. (**a**) General schematic of the proposed variable-stiffness actuator (VSA) design in a prosthetic leg application. (**b**) Geometry used for mechanism analysis.

Since rollers are not preloaded against the beam, we assume that contact occurs between the beam and only one of the rollers at a time. The corresponding reaction force can be decomposed into components perpendicular and parallel to the length of Link 2 (represented as the leg shank in **Figure 1**). The normal component creates a moment in the joint. The parallel component is a thrust force controlled with the motor-driven belt. When the beam releases energy, negative work associated with the thrust force is regenerated by the servo amplifiers and stored in batteries or

supercapacitors. Conversely, power may be drawn from the electric storage to drive the belt, deflect the beam, and store elastic energy. Such bidirectional exchange of electric and elastic energy is expected to significantly contribute to overall system efficiency.

3.5 GEOMETRY AND BEAM DEFLECTION STATICS

A desired knee-moment profile is obtained by suitably varying the point of application of force on the beam, which is, in turn, determined by knee angle and roller position. In order to study the use of the beam as a compliant mechanism, a cantilever beam, subject to large deflections, is examined. As shown in Figure 2, a co-ordinate frame (ub,vb) was established to analyze deflections. Contact between beam and rollers occurs at single point T. Therefore, normal force F must be perpendicular to both the beam and the roller in contact. Two component forces, nP and P, are defined along the ub and vb axes, respectively. The beam is expected to undergo deflections exceeding the assumptions of elementary bending theory. That is, a given point on the beam is displaced in a plane, not merely in the vb direction. Euler–Bernoulli beam theory with geometric nonlinearity is suitable to analyze such cases whenever beam thickness is small relative to its width and length, so that shear deformations may be neglected.

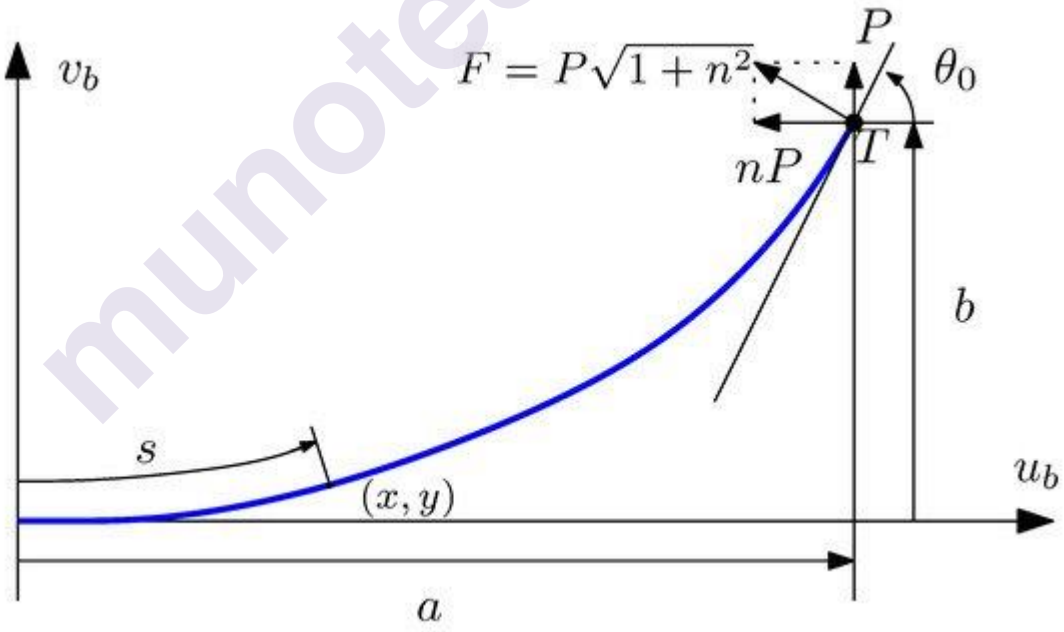


Figure 2. Geometry of beam deflection. Active length *L* is the arc length between fixed end and point *T*.

Modeling large deflections in beams was extensively studied with different levels of complexity and accuracy . However, the case under study introduces difficulty in that neither the active length of the beam nor the magnitude and orientation of the applied force are known a priori. Rather, these quantities become unknowns that appear in constraining equations associated with tangency with the roller. Therefore, only roller

position xr and the angle between the links θj are known. Note that the cantilever beam is subjected to single-point force. For this study, this point is considered the end forces used in Reference [**31**], and it defines the active length of the beam. The following sections describe the proposed solution method that returns the contact-force components, the active beam length, and the end deflections and other variables of interest, such as the joint moment and elastic energy stored in the beam.

3.6 HOW TO SELECT THE RIGHT ACTUATOR

Understanding the different types of actuators is a crucial step in making the best selection for your equipment. Since each kind has its unique purpose and energy requirements, we'll go over factors that will help you arrive at the best decision.

- **Power Source Availability**

  The first thing you have to consider is the compatibility of your power source. If you own an industrial site with an electrical source, perhaps the best choice—and the option with the most selections—would be electric actuators. If there are no electrical sources in the area, or you want a piece of fully functional equipment without electricity, you can opt for pneumatic or hydraulic types.

- **Required Movement**

  Another important factor when choosing an actuator is the range of movement that you need for your equipment. Is it linear, rotary, or an integration of both? Custom-made actuators can combine or chronologically create these motions to help you concretize the final equipment.

- **Precision**

  Some actuators are more precise than others. For example, air brakes are created through pneumatic actuators because air pressure is known to be efficient in the start and stop movements. Other actuators have a larger margin of movement variations, such as those operated through hydraulics.

  Any industry that requires a high level of precision for safety and success of operation should consider actuator types that have specific movements.

- **Safety and Environmental Concerns**

  Safety is another factor to consider when choosing an actuator for your equipment. Electrical or thermal actuators should be used with caution in areas with extreme temperatures or conducting hazards. For example, operating electrical actuators close to a water body without sealing or other safety measures may create an occupational hazard.

If your company is also committed to a reduced carbon footprint, you'll need to note each actuators' environmental impact. Typically, electrical actuators have little to no carbon footprint.

Official Guidelines

There are also specific guidelines to follow for industrial actuators in certain areas. For example, locations with a high presence of combustible gases should adhere to the requirements imposed by the National Electrical Manufacturers Association (NEMA).

3.7 MAINTAINING YOUR ACTUATOR

All equipment requires maintenance. Maintaining your actuators will help prevent major shutdowns, hazards, or loss of productivity. Here are some general tips to keep your actuators in top shape.

Regular inspection: Performing routine visual equipment checks will identify early signs of actuator issues. A mechanic with a keen eye is necessary to inspect for wear and tear.

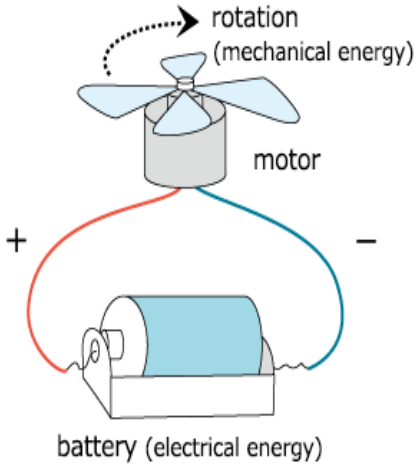
Replenish and replace: Hydraulic actuators sometimes need cylinder fluid replenishment. Always double-check for leaks and signs of low hydraulic fluid levels. Replace loose or damaged nuts, bolts, coils, or screws in your actuator parts as well.

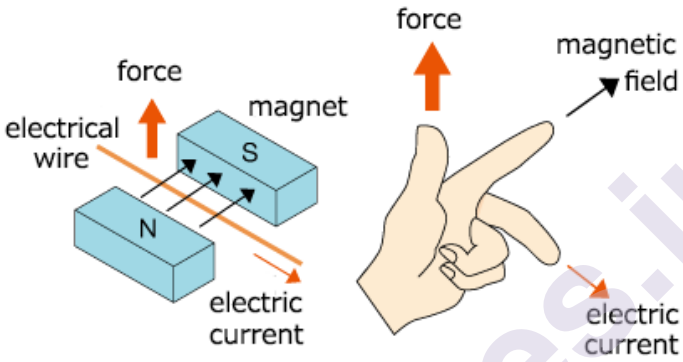
Measure performance data: In some cases, actuators won't show external signs of a problem, but you can trace issues through performance. Automated graphs and output computation may be necessary if you want to catch deeper issues.

3.8 MOTOR

A dictionary describes, "a motor is a machine that converts electrical energy to mechanical energy." In another words, the electrical energy is a "battery" and the mechanical energy is the "rotation." To explain a motor physically, the well-known "Fleming's left hand rule" is a good approach. When electric current flows through an electrical wire placed between two magnets facing with each other, it generates force. Electric current, magnetic field and motion respectively applies perpendicular directions each other just as when you open the middle finger (electric current), the forefinger (magnetic field) and the thumb (force) of your left hand respectively to mutually orthogonal axes.

Then, why does electric current which flow through the electrical wire generate force? This is because when the electric current flows through the electrical wire, it generates magnetic field around. The magnetic field attracts or repels magnetic field from magnets, which generate force to move the electric wire. The electrical energy here is "electric current," and the mechanical energy is "force."

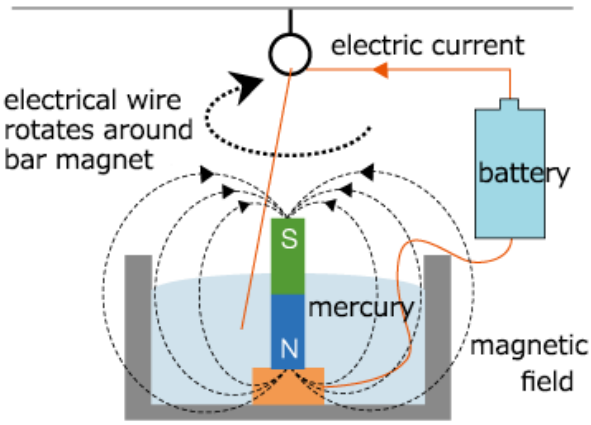




Fleming's left hand rule

3.8.1 The beginning of motors

In 1831, a British physicist, Michael Faraday, discovered the law of electromagnetic induction that electric current flows when you move magnets in air core coil. The law of electromagnetic induction proved that the electrical energy and the mechanical energy are mutually convertible. It is said that this is the catalyst of invention of motors. In those days, Great Britain was in the period of the first Industrial Revolution and steam power was the driving force of the revolution. No one could recognize the importance of motors which worked with electricity in those days without power network.



First motor that Faraday invented

3.8.2 Toward practical motors

Since the discovery of electromagnetic induction by Faraday, people had invented a number of motors. In 1834, Thomas Davenport invented a practical DC motor. After that, a Yugoslavian electrical engineer, later became an American, Nikola Tesla, came up with an idea to drive motors with alternating current. In 1882, the idea of the principle of rotating magnetic field suddenly hit his head when he was walking in the park. In 1887, he completed a practical two-phase AC motor (induction motor) using rotating magnetic field. Since then AC technologies such as transformer, three-phase-three-wire system have developed as well as power network. The more available the electricity became, the wider the usage of motors expanded.

Thanks to Tesla's breakthrough, now we are able to enjoy our lives with electricity and motors. By the way, Tesla once worked for the company run by the great inventor Edison, he came into collision with Edison and left the company within one year. Tesla left words cynically twisting Edison's words as saying, "Genius is 1 percent inspiration and 99 percent vain effort".

3.9 VARIOUS KINDS AND FEATURES OF MOTORS

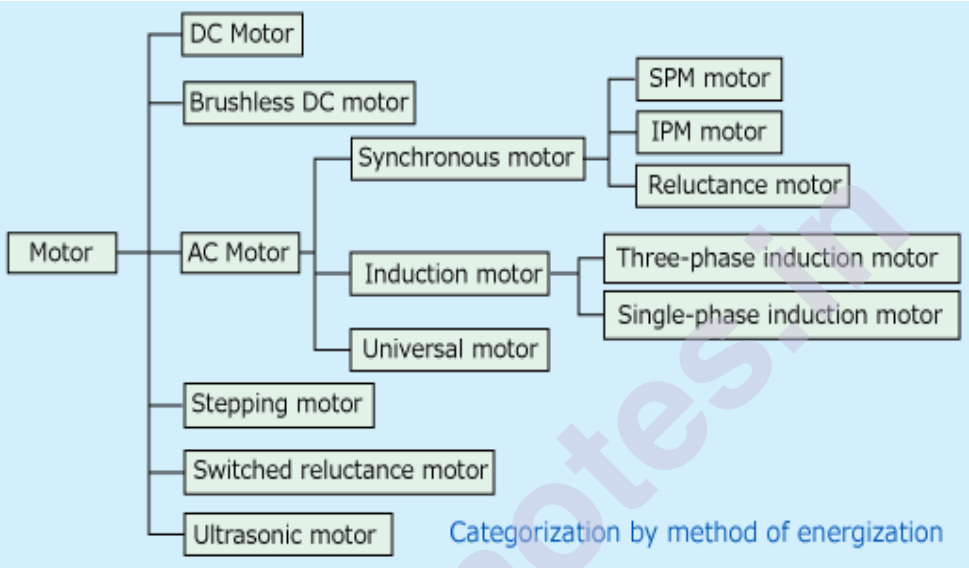
180 years after the birth of motors, its performance and usability improved significantly thanks to the progress in designing and manufacturing technologies and material technology as well as electronics. There are various ways to call motors depending on the categorization of functions and structures such as a servo motor for its precise work toward commands, a linear motor for its linear movement, a vibrating motor for its vibration to notify incoming call on mobile phone and a geared motor for combined speed reducer. Motors also have a few names although their structures are the same. Starting with a motor for coal mine, now that Yaskawa Electric's motors expand the usage to wide-ranging fields such as industrial machinery, robot and electric vehicle (EV). For example, the list below shows a few names used in motors for EV. People named motors to define the differences from others, resulted in leaving with many names for motors. It is such a complicated background but also a "proof of motor's diversification."

application		EV motor
categorization	the difference of energization	AC motor
	the difference between synchronous and asynchronous	synchronous motor
	the difference of rotor structure	IPM motor

EV : Electric Vehicle
AC : Alternative Current
IPM : Interior Permanent Magnet

3.10 CATEGORIZATION OF MOTORS

DC motors flow direct current (DC) through it, besides AC motors flow alternating current. A brushless DC electric motor is a DC motor which replaced its brush and commutator with semiconductor switching element. A universal motor is able to rotate the motor in high speed with AC 100V electricity for households while holding the same brush and commutator for DC motors. Other than these, there are a stepping motor that moves with square-wave current flow and a switched reluctance motor. An ultrasonic motor is a special motor that works by vibrating piezoelectric ceramic with applying high frequency voltage.



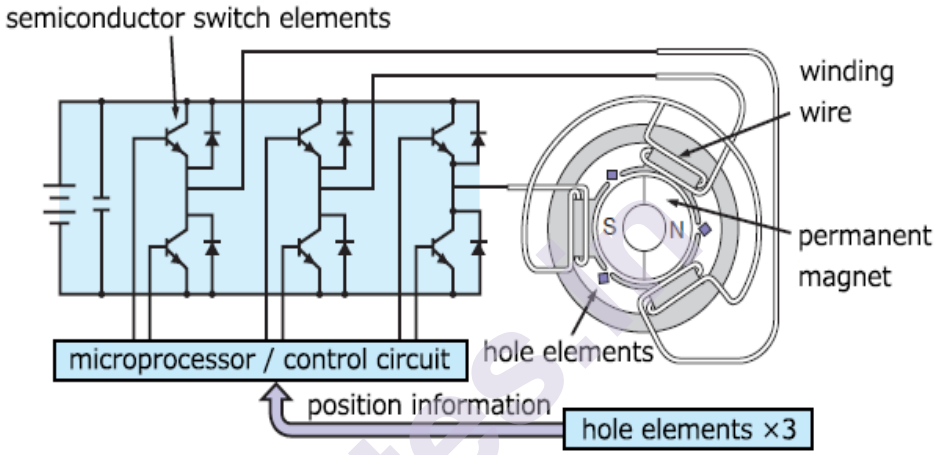
1) DC motors

The motor which many Japanese pupils used in their science experiments when they were in primary schools was DC motors. It is the most popular motor used in models, consumer electronics and vibration motors in mobile phone. To explain roughly about the structure of motors, there are rotor and stator in it. Rotor is a part connected to shaft, and stator is a fixed part which comprises the exterior.

The stator in DC motors holds permanent magnets and brushes that supply electric current to the rotor, and the rotor holds windings and a commutator. Once the brushes supply DC current to the commutator, electric current starts to flow through the windings connected to the commutator and generates torque. Here, the windings and the commutator have a mechanism to flow electric current in the way that torque keeps the same level. The greatest feature of DC motor is its usability that works with dry cell. You can change the direction of rotation by just changing the connection of motor wires. This is why DC motors are widely accepted.

2) Brushless DC motors

You can describe a brushless DC motor as "a motor without brushes while it has a feature similar to a DC motor." It holds windings in the stator and permanent magnets in the rotor as its structure. It doesn't have brushes and a commutator which DC motors used to have, instead it holds a semiconductor switch element outside the motor. It works to flow DC current all the time through two out of three phases of windings, phase U, V and W. It switches the current flow according to a permanent magnets' position detected by such as a hall element sensor and keeps generating the same level of torque.



3) Synchronous motors

On the other hand, a synchronous motor drives in sine wave using information detected by angle sensor attached to the edge of rotor. The synchronous motor is named after the mechanism that the rotation of the magnetic field generated by the three-phase windings synchronized to the rotation of the rotor. The structure of synchronous motors is basically the same as that of brushless DC motors. Therefore, people often take synchronous motors for brushless DC motors and vice versa.

One of the features of both synchronous motors and brushless DC motors is that they are able to prevent brush wear and electrical noise. They are also capable of downsizing, high output and high efficiency by using strong rare earth magnets. Thanks to these features, there is a wide range of usage such as information devices, home electronics, in-vehicle motors and servo motors. It is said that DC motors accounts for 70%, and the combined number of brushless DC motors and synchronous motors accounts for 20% of total number of small-sized motor produced.

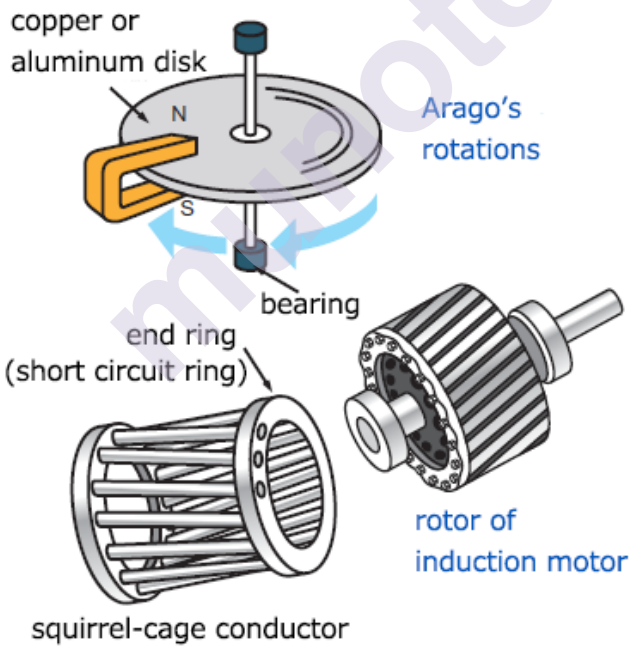
4) Induction motors

The rotation principle of induction motors is based on the "Arago's rotations" discovered by a French physicist Arago. It is a phenomenon that when you put an aluminum disc between a U-shaped magnet and move the magnet toward the rotating direction, then the aluminum disc starts to rotate in the same direction with a little time lag. When the magnetic field

from the U-shaped magnet changes on the aluminum disc, a spiral electric current flows through the aluminum disc (the law of the electromagnetic induction), and the action of the current and the magnetic field of the U-shaped magnet generates the electromagnetic force. Induction motors are the invention applied the Arago's rotations.

The stator of induction motors holds three-phase windings in its structure. And the rotor holds cage-shaped aluminum part (case-shaped conductor). When you drive the three-phased windings in sine wave, it generates a magnetic field which rotates at the frequency. Then, as in the principle of Arago's rotations, electric current flows through the squirrel-cage conductor which receives the changes of the magnetic field and the rotor starts to rotate with a little time lag.

Induction motors are less efficient compare to brushless DC motors and synchronous motors which use permanent magnets, however, they have other features such as they are applicable to the three-phase AC 200V commercial power source, possible to rotate without a hall element sensor or an angle sensor, hard to break, possible to operate efficiently with AC Drive and capable of large output with large-sized motor. Therefore, there are many use cases of induction motors in industrial area and vehicles. Similar to biodiversity, we have variety of motors which have a wide range of nature depending on the difference of the structures and distribution of materials.



3.11 DEGREE OF FREEDOM LOCOMOTION

Degrees of freedom (DoF) refer to the number of basic ways a rigid object can move through 3D space. There are six total degrees of freedom. Three correspond to rotational movement around the x, y, and z axes, commonly

termed pitch, yaw, and roll. The other three correspond to translational movement along those axes, which can be thought of as moving forward or backward, moving left or right, and moving up or down.

VR headsets and input devices are generally 3DoF or 6DoF.

3DoF means we can track rotational motion but not translational. For the headset, that means we can track whether the user has turned their head left or right, tilted it up or down, or pivoted left and right.

6DoF means we can additionally track translational motion. That means we can track whether the user has moved forward, backward, laterally, or vertically.

3.11.1 DoF vs 6DoF

In VR, DoF is a common term that stands for Degree of Freedom. The number before it stands for how many different axes are being tracked. It is significant when deciding which type of VR headset to buy depending on it's DoF tracking.

For instance, when purchasing a VR headset for the purpose of watching 360 videos, then a 3DoF VR headset is enough. This is because you cannot change the distance to the objects in the room. In 3DoF headsets, you can only move in 3D environments with joysticks but not your physical motion.

On the contrary, a 6DoF headset is necessary if you want to use it in 3D environments. With 6DoF, your physical movement is also tracked by the gyroscope to allow virtual movement.

There are three main axes which are known as the translational and three secondary axes which are known as rotational.

When a VR device has 3DoF it will either be rotational or translational. But, not both. On the other hand, a 6Dof experience allows for both. Therefore, if you want to track both the head and physical body movements, a 6DoF is definitely the best option for you to buy.

Rotational axes

There are three types of movements in rotational axes. They can be tracked by the movement of the head. These movements are known as rolling, pitching and yawing.

Rolling (X-axis)

Rolling is where the head pivots side to side.Try this by looking straight-forward. Then slowly tilt your head, where your right ear approaches your right shoulder. Now repeat the same movement with your left ear approaching your left shoulder. This movement along the rotational X-axis is known as rolling.

Yawing (Y-axis)

Yawing is where the head swivels along a horizontal axis. Try this by looking straight-forward. Then slowly move your head to face your right,

where your chin approaches your right shoulder. Now repeat the same movement in the opposite direction to your left. This movement along the rotational Y-axis is known as yawing.

Pitching (Z-axis)

PITCHING

Pitching is where the head tilts along a vertical axis. Try this by looking straight-forward. Then slowly move your head along the vertical axis where your chin approaches your neck. Now repeat the same movement with your chin moving away from your neck. This movement along the rotational Z-axis is known as pitching.

Transitional axes

There are three types of movements in transitional axes. They can be tracked by the movement of the body. These movements are known as: elevating, strafing and surging.

Strafing (X-axis)

STRAFING

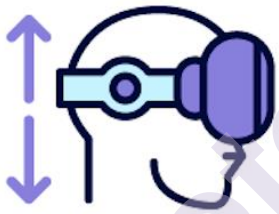
Strafing is when you move sideways in a straight-line. You can try this by moving in a straight-line to the left then moving to the right. This movement along the transitional x-axis is known as strafing.

Surging (Y-axis)

SURGING

Surging is when you move front and back in a straight line. This can be done by taking steps forward or backwards in a straight-line. This movement along the transitional y-axis is known as surging.

Elevating (Z-axis)



ELEVATING

Elevating is when you move your body vertically along the z-axis. This can be done by jumping up and down or crouching and then getting back up. This movement along the transitional z-axis is known as elevating.

3.12 SUMMARY

- In IoT device is made up of a Physical object (“thing”) + Controller (“brain”) + Sensors + Actuators + Networks (Internet).
- An actuator is a machine component or system that moves or controls the mechanism or the system.
- Actuators, also known as drives, are mechanisms for getting robots to move. Most actuators are powered by pneumatics (air pressure), hydraulics (fluid pressure), or motors (electric current).
- A hydraulic actuator uses hydraulic power to perform a mechanical operation.
- A pneumatic actuator uses energy formed by vacuum or compressed air at high pressure to convert into either linear or rotary motion.

- All equipment requires maintenance. Maintaining your actuators will help prevent major shutdowns, hazards, or loss of productivity. Here are some general tips to keep your actuators in top shape.
- A dictionary describes, "a motor is a machine that converts electrical energy to mechanical energy." In another words, the electrical energy is a "battery" and the mechanical energy is the "rotation."
- DC motors flow direct current (DC) through it, besides AC motors flow alternating current.
- You can describe a brushless DC motor as "a motor without brushes while it has a feature similar to a DC motor."
- The rotation principle of induction motors is based on the "Arago's rotations" discovered by a French physicist Arago. It is a phenomenon that when you put an aluminum disc between a U-shaped magnet and move the magnet toward the rotating direction, then the aluminum disc starts to rotate in the same direction with a little time lag.
- Degrees of freedom (DoF) refer to the number of basic ways a rigid object can move through 3D space.

3.13 QUESTIONS

1. Write a short note on Actuators
2. What are the Types of Actuators?
3. How to Select the Right Actuator
4. Write a short note on Motors.
5. What are Various kinds and features of motors?
6. Explain all categories of Motors.
7. Write a short note on Degree of freedom Locomotion

3.14 REFERENCE FOR FURTHER READING

- http://electronicsteacher.com/robotics/robotics-technology/actuators.php
- https://www.avsystem.com/blog/iot-sensors-iot-actuators/
- https://www.mdpi.com/2076-0825/8/1/12/htm
- https://www.creativemotioncontrol.com/types-of-actuators/
- https://www.yaskawa-global.com/product/mc/about-motor
- https://www.smartvrlab.nl/3dof-vs-6dof-in-vr/

❄❄❄❄❄❄❄

4

MANIPULATORS

Unit Structure

4.0 OBJECTIVE

In this chapter we will discuss and study about robots arm, and its type. And we will study about sensor and its type. From this chapter we will get knowledge about mentioned topics

4.1 INTRODUCTION

An industrial robot is comprised of a robot manipulator, power supply, and controllers. Robotic manipulators can be divided into two sections, each with a different function: Robot Arm and Body

The arm and body of a robot are used to move and position parts or tools within a work envelope. They are formed from three joints connected by large links.

- **Robot Wrist**

  The wrist is used to orient the parts or tools at the work location. It consists of two or three compact joints.

  Robot manipulators are created from a sequence of link and joint combinations. The links are the rigid members connecting the joints, or axes. The axes are the movable components of the robotic manipulator that cause relative motion between adjoining links. The mechanical joints used to construct the robotic arm manipulator consist of five principal types. Two of the joints are linear, in which the relative motion between adjacent links is non-rotational, and three are rotary types, in which the relative motion involves rotation between links.

  The arm-and-body section of robotic manipulators is based on one of four configurations. Each of these anatomies provides a different work envelope and is suited for different applications.

- **Gantry Robots**

  These robots have linear joints and are mounted overhead. They are also called Cartesian and rectilinear robots.

- **Cylindrical Robots**

  Named for the shape of its work envelope, cylindrical anatomy robots are fashioned from linear joints that connect to a rotary base joint.

- **Polar Robots**

  The base joint of a polar robot allows for twisting and the joints are a combination of rotary and linear types. The work space created by this configuration is spherical.

- **Jointed-Arm Robots**

  This is the most popular industrial robotic configuration. The arm connects with a twisting joint, and the links within it are connected with rotary joints. It is also called an articulated robot.

4.2 END EFFECTORS

The end effector means the last link (or end) of the robot. At this endpoint, the tools are attached. In a wider sense, an end effector can be seen as the part of a robot that interacts with the work environment. End effectors may consist of a gripper or a tool.

EOAT – also known as end effectors – are crucial for unleashing the automation potential of industrial robots. One could even say that there

will be no automation advantages without at least one EOAT. End of Arm Tooling (EOAT) could be gripers for clamping the objects or tools for manufacturing operations: arc welding, spot welding, painting, polishing, grinding, machining, etc.

Robot grippers are the physical interface between a robot arm and the work piece. This end-of-arm tooling (EOAT) is one of the most important parts of the robot. One of the many benefits of material handling robots is the reduction of part damage. A gripper comes in direct contact with your product, so it's important to choose the right type of gripper for your operation.

A gripper is the mechanical or electrical End Of Arm Tooling (EOAT) device that enables the manipulation of an object. Or, in other words, it is a machine's controllable "hand" that grasps and releases parts that are being moved by the automation. There are many different types and sizes of grippers designed to pick up a wide variety of parts and materials.

Robot end effector is integrated with the wrist assembly

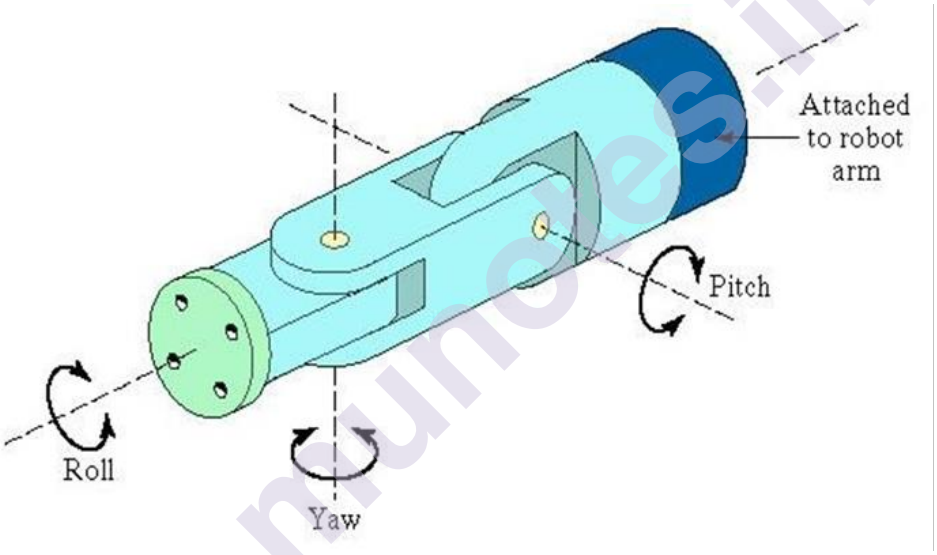


Fig . Wrist assembly

Basic connected with the wrist assembly:
Wrist assembly is attached to end-of-arm
End effector is attached to wrist assembly
Function of wrist assembly is to orient end-effector
Body-and-arm determines global position of end effector
Two or three degrees of freedom are normally used:

- Roll
- Pitch
- Yaw

With end of arm tooling we can consider the following:

The most essential part of robot is for determining its functionality is end effector, or end-of-arm-tooling (EOT). There are very different application for EOT. Most common are end effectors including welding devices (MIG, TIG welding, spot welding), spry guns also grinding and deburring devices (pneumatic disk or belt grinders); grippers (devices that can grip an object, usually electromechanical or pneumatic); picking objects by vacuum; EOT devices for machining, etc.

EOT are frequently highly complex. They can carry out different activities at the same time and may utilize various sensors.

End-effector systems are rapidly in developing

4.3 BASIC CLASSIFICATION OF END EFFECTORS

End effectors are the last part of a robot arm. They are directly connected to the wrist and have two basic functions: to grasp the objects or to do the work with the objects. For grasping are used different grippers. For doing the work (welding, painting, grinding, polishing, etc.) are used different tools. This have a general meaning as End of Arm Tooling, which is used sometimes also as a general meaning of end effectors.

1. Grippers

Mechanical grippers

Mechanical grippers are grippers that use mechanical fingers to manipulate objects (see Fig.a). Mechanical grippers have a distinctive design that is reminiscent of a crab's pincers. Mechanical grippers usually come with adjustable force and stroke features, enabling them to perform tasks with human-like precision and dexterity. The number of robot fingers vary depending on the model; however, most mechanical grippers have two to four fingers. The fingers are per default replaceable, allowing you to make the most of your investment.

Grippers grasp and manipulate objects during the work cycle. Typically the objects grasped are workpieces that need to be loaded or unloaded. The loading unloading tasks depend from industrial task. It could be realised based on one equipment or for replacing the object from one station to another. Typical loading-unloading operations are machine tool tending, execution of bending, pressing, etc operations, using for pick and place or palletizing activities. Grippers may be standardized or custom-designed to suit the physical specifications of the workpieces they have to grasp. Basic classification of grippers are given in a Table 1.

Different applications of mechanical grippers are given in Fig.a and b.

Table 1 End effectors – grippers basic classification

Type	**Description**
Mechanical gripper	Two or more fingers that can be actuated by robot controller to open and close on a workpiece
Vacuum gripper	Suction cups are used to hold flat objects
Magnetised devices	Making use of the principles of magnetism, these are used for holding ferrous workpieces
Adhesive devices	Deploying adhesive substances these hold flexible materials, such as fabric
Simple mechanical devices	For example, hooks and scoops
Dual grippers	Mechanical gripper with two gripping devices in one end effector for machine loading and unloading. Reduces cycle time per part by gripping two workpieces at the same time
Interchangeable fingers	Mechanical gripper whereby, to accommodate different workpiece sizes, different fingers may be attached
Sensory feedback fingers	Mechanical gripper with sensory feedback capabilities in the fingers to aid locating the workpiece; and to determine correct grip force to apply (for fragile workpieces)
Multiple fingered grippers	Mechanical gripper with the general anatomy of the human hand. Standard grippers
Standard grippers	Mechanical grippers that are commercially available, thus reducing the need to custom-design a gripper for each separate robot application

Fig a *Different application areas of mechanical grippers*

Fig b. *Different realization possibilities of mechanical grippers*

End effector for machine tending is give in the Figure

There is also a significant problem to choose the right end effector. First it is necessary to take care of the needed functionality. For loading unloading operations there is needed to grasp the object. There is also

needed to consider the measures of servicing objects, also the weight. Very important is to understand about the material of the objects, their surface quality conditions and other specific issues. For sheet metal products producing operations vacuum grippers are widely used.

Some parameters of grippers for palletizing operations are given in the Figure




Technical Parameters

Gripper type	Clamp (Single zone)	Clamp (Double zone)	Claw	Vacuum
Gripper weight (kg)	45	80	60	75
Max weight / lift (kg)	40	60	50	40
Product type	Case	Case	Bag	Case, pallet

There is necessary to consider that the end-effectors are working also together with the industrial robots. And the robot parameters and their functionality must fit to the parameters of end effectors.

1. Vacuum grippers

Vacuum grippers provide gripping through suction cups and are mostly used for handling workpieces with uneven surfaces or irregular shapes. Traditional vacuum grippers utilize external air supply systems which require high maintenance costs. Newer models run on electricity instead, eliminating the heavy costs in addition to improving the work environment due to producing less noise and dust. One example

VG10 vacuum gripper falls into the latter category.

Fig *Pneumatic gripper*

2. Adhesive grippers

These types of grippers grasp objects by sticking to them instead of holding them (see Fig). Adhesive grippers are typically used to pick up light-weight objects, such as fabrics. However, the reliability of an adhesive gripper will diminish with each successful operation due the adhesive substance losing its quality.

Grippers can grasp work pieces, centre and orientated them. It should include sensors to indicate, if a part is present or not. On the whole a gripper should be as lightweight as possible, for the maximum payload of a machine includes the weight of a work piece. Housing and fork should be held as close to the axis of grip as possible, so as to avoid high moments on the gripper.

Fig *Gripper with the changeable fingers*

The above shown three finger gripper is an adequate end effector for performing task of taking parts from input buffer and put them into the servicing machine and then take them out and keep them to output buffer.

The robotic grippers could be classified also according to the working principles: vacuum grippers, pneumatic grippers, hydraulic grippers and servo-electric grippers. Manufacturers choose grippers based on which handling application is required and the type of material in use.

3. Vacuum Grippers

The vacuum gripper has been the standard EOAT in manufacturing because of its high level of flexibility. This type of robot gripper uses a rubber or polyurethane suction cup to pick up items. Some vacuum grippers use a closed-cell foam rubber layer, rather than suction cups, to complete the application.

4. Pneumatic Grippers

The pneumatic gripper is popular due to its compact size and light weight. It can easily be incorporated into tight spaces, which can be helpful in the manufacturing industry. Pneumatic robot grippers can either be opened or closed, earning them the nickname “bang bang” actuators, because of the noise created when the metal-on-metal gripper operates.

5. Hydraulic Grippers

The hydraulic gripper provides the most strength and is often used for applications that require significant amounts of force. These robotic grippers generate their strength from pumps that can provide up to 2000psi. Although they are strong, hydraulic grippers are messier than other grippers due to the oil used in the pumps. They also may need more maintenance due the gripper being damaged because of the force used during the application.

6. Servo-Electric Grippers

The servo-electric gripper appears more and more in industrial settings, due to the fact that it is easy to control. Electronic motors control the movement of the gripper jaws. These grippers are highly flexible and allow for different material tolerances when handling parts. Servo-electric grippers are also cost effective because they are clean and have no air lines.

4.3.1 End of Arm Tooling

Additionally to material handling industrial robots are being used in three types of applications: processing operations; assembly and inspection. In processing operations the robot performs some processing actions, such as grinding, milling, machining etc. on the workpiece. The end effector is

equipped with the specialised tool required for the process, and the tool is moved relative to the surface of the workpiece. Table 2 outlines examples of processing operations that deploy robots. Welding (arc welding or spot welding is one of the most popular application area of industrial robots. The application areas of EOAT is given in the Table 2

Table 2. Industrial robot applications

Process	**Description**
Spot Welding	Metal joining process in which two sheet metal parts are fused together at localised points of contact by the deployment of two electrodes that squeeze the metal together and apply an electric current. The electrodes constitute the spot welding gun, which is the end effector tool of the welding robot
Arc Welding	Metal joining process that utilises a continuous rather than contact welding point process, in the same way as above. Again the end effector is the electrodes used to achieve the welding arc. The robot must use continuous path control, and a jointed arm robot consisting of six joints is frequently used.
Spray Coating	Spray coating directs a spray gun at the object to be coated. Paint or some other fluid flows through the nozzle of the spray gun, which is the end effector, and is dispersed and applied over the surface of the object. Again the robot must use continuous path control, and is typically programmed using manual lead-through. Jointed arm robots seem to be the most common anatomy for this application
Other applications	Other applications include: drilling, routing, and other machining processes; grinding, wire brushing, and similar operations; waterjet cutting; and laser cutting.

The robot end effector may also use tools (in Fig.11 for example grinding tools). Tools are used to perform processing operations (drilling, milling, painting, polishing, etc) on the workpiece. Typically the robot uses the tool relative to a stationary or slowly-moving object; in this way the process is carried out. For example, spot welding, arc welding, and spray painting – these all use a tool for processing the operation. Other examples where a tool is held by the robotic manipulator, and used against the workpiece include: rotating spindle for drilling, routing, grinding, and

similar operations. The end effector could also be used as a heating torch or as a water jet cutting tool for cutting operations. Also the robot tool could be used for painting operations, etc. For each instance, the robot controls both the position of the workpiece, and the position of the tool relative to the workpiece. In these cases the robot must be able to transmit control signals to the tool for starting, stooping, and otherwise regulating the tools actions. In the following figures some end-effectors and their use is illustrated.

Figure 11. *Grinding tool and robot-grinding application*

4.4 TELEOPERATION

Teleoperation (or **remote operation**) indicates operation of a system or machine at a distance. It is similar in meaning to the phrase "remote control" but is usually encountered in research, academia and technology. It is most commonly associated with robotics and mobile robots but can be applied to a whole range of circumstances in which a device or machine is operated by a person from a distance.

The term *teleoperation* is in use in research and technical communities as a standard term for referring to operation at a distance. This is as opposed to *telepresence* which is a less standard term and might refer to a whole range of existence or interaction that include a remote connotation.

4.4. 1 Teleoperation of Autonomous Vehicles

Teleoperation of Autonomous Vehicles, is the ability to remotely drive or assist a self-driving car.

Most leading companies in the industry believe that to bridge the gap between current self-driving capabilities and the requirements needed for widespread adoption of autonomous vehicles, there is a need to have Teleoperation capabilities for assisting self-driving cars, in situations of 'edge cases' – where the autonomous vehicle software stack has low confidence level in its ability to perform the correct action, or when the vehicle needs to operate outside of its standard operating parameters. Without remote assistance, in such situations the self-driving car would transition to a Minimum Risk Maneuverer (MRM) which is usually to stop.

Many AV companies plan on using teleoperations as part of their rollout for self driving cars. Examples of companies that have stated they will deploy, or currently deploying teleoperations solutions include Voyage.auto,[5] Denso, Waymo, GM Cruise, Aptiv, Zoox.

Teleoperation of Autonomous Vehicles includes privately owned self driving car use cases, such as self parking assistants, shared mobility use cases, e.g. in robotaxis and autonomous shuttles and industrial use cases, for example autonomous forklifts.

There are two main modes for Teleoperation of Autonomous Vehicles: Remote Driving, also called "Direct Driving" – where the remote operator performs the dynamic drive task, i.e. drives the car remotely, controlling the car's steering, acceleration and braking systems. Remote Assistance, also called "High Level Commands" – remote operators supervise the vehicle, and provide instructions, approve or correct the vehicle path, without actually performing the dynamic driving task. Some companies deploy a combination of both concepts, depending on the use case. Examples of companies that provide solutions in the field of Teleoperations are DriveU.auto, Scotti.ai, Phantom.Auto, Pylot, Ottopia, and Designated Driver.

4.5 SENSORS

A sensor is a window for a robot to the environment. Sensors allow robots to understand and measure the geometric and physical properties of objects in their surrounding environment, such as position, orientation, velocity, acceleration, distance, size, force, moment, temperature, luminance, weight, etc.

Sensors are generally classified into two groups: internal sensors and external sensors. Internal sensors such as its position sensor, velocity sensor, acceleration sensors, motor torque sensor, etc obtain the information about the robot itself, while external sensors such as cameras, range sensors (IR sensor, laser range finder, and ultrasonic sensor) contact and proximity sensors (photodiode, IR detector, RFID, touch, etc.) and force sensors gather the information in the surrounding environment.

Sensors are defined by various properties that describe their capabilities:

- Sensitivity (change of output and change of input)
- Linearity (constancy of output and input)
- Response Time (time required for a change in input to force a change in the output)
- Measurement/Dynamic range (difference between minimum and maximum)
- Accuracy (difference between measured & actual)
- Repeatability (difference between repeated measures)
- Resolution (smallest observable increment)
- Bandwidth (result of high resolution or cycle time)

4.5.1 Types of Robot Sensors

There are different sensors to choose from and we will identify the characteristics of few sensors, and also understand why and where they are used.

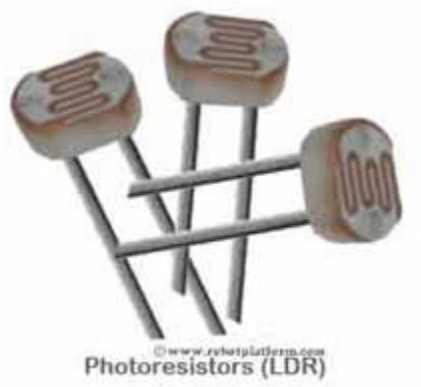
1. Light sensors

A Light sensor is used to detect light and create a voltage difference. The two main light sensors generally used in robots are Photoresistor and Photovoltaic cells. Other kinds of light sensors like Phototubes, Phototransistors, CCD's etc. are rarely used.

Photoresistor is a type of resistor whose resistance varies with change in light intensity; more light leads to less resistance and less light leads to more resistance. These inexpensive sensors can be easily implemented in most light dependant robots.

Photovoltaic cells convert solar radiation into electrical energy. This is especially helpful if you are planning to build a solar robot. Although

photovoltaic cell is considered as an energy source, an intelligent implementation combined with transistors and capacitors can convert this into a sensor.



2. Sound Sensor

As the name suggests, this sensor (generally a microphone) detects sound and returns a voltage proportional to the sound level. Sound SensorA simple robot can be designed to navigate based on the sound it receives. Imagine a robot which turns right for one clap and turns left for two claps. Complex robots can use the same microphone for speech and voice recognition.

Implementing sound sensors is not as easy as light sensors because Sound sensors generate a very small voltage difference which should be amplified to generate measurable voltage change.

3. Temperature Sensor

What if your robot has to work in a desert and transmit ambient temperature? Simple solution is to use a temperature sensor. Tiny temperature sensor ICs provide voltage difference for a change in temperature. Few generally used temperature sensor IC's are LM34, LM35, TMP35, TMP36, and TMP37.

4. Contact Sensor

Contact sensors are those which require physical contact against other objects to trigger. A push button switch, limit switch or tactile bumper switch are all examples of contact sensors. These sensors are mostly used for obstacle avoidance robots. When these switches hit an obstacle, it triggers the robot to do a task, which can be reversing, turning, switching on a LED, Stopping etc. There are also capacitive contact sensors which react only to human touch (Not sure if they react to animals touch). Touch screen Smart phones available these days use capacitive touch sensors (Not to be confused with older stylus based models). Contact Sensors can be easily implemented, but the drawback is that they require physical contact. In other words, your robot will not turn until it hits an object. A better alternative is to use a proximity sensor.

Limit Switch (Contact Sensor)

5. Proximity Sensor

This is a type of sensor which can detect the presence of a nearby object within a given distance, without any physical contact. The working principle of a Proximity sensor is simple. A transmitter transmits an electromagnetic radiation or creates an electrostatic field and a receiver receives and analyzes the return signal for interruptions. There are different types of Proximity sensors and we will discuss only a few of them which are generally used in robots.

i. Infrared (IR) Transceivers: An IR LED transmits a beam of IR light and if it finds an obstacle, the light is simply reflected back which is captured by an IR receiver. Few IR transceivers can also be used for distance measurement.

ii. Ultrasonic Sensor: These sensors generate high frequency sound waves; the received echo suggests an object interruption. Ultrasonic Sensors can also be used for distance measurement.

iii. Photoresistor: Photoresistor is a light sensor; but, it can still be used as a proximity sensor. When an object comes in close proximity to the sensor, the amount of light changes which in turn changes the resistance of the Photoresistor. This change can be detected and processed.

There are many different kinds of proximity sensors and only a few of them are generally preferred for robots. For example, Capacitive Proximity sensors are available which detects change in capacitance around it. Inductive proximity sensor detects objects and distance through the use of induced magnetic field.

6. Distance Sensor

Most proximity sensors can also be used as distance sensors, or commonly known as Range Sensors; IR transceivers and Ultrasonic Sensors are best suited for distance measurement

i. Ultrasonic Distance Sensors: The sensor emits an ultrasonic pulse and is captured by a receiver. Since the speed of sound is almost constant in air, which is 344m/s, the time between send and receive is calculated to give the distance between your robot and the obstacle. Ultrasonic distance sensors are especially useful for underwater robots.

ii. Infrared Distance sensor: IR circuits are designed on triangulation principle for distance measurement. A transmitter sends a pulse of IR signals which is detected by the receiver if there is an obstacle and based on the angle the signal is received, distance is calculated. SHARP has a family of IR transceivers which are very useful for distance measurement. A simple transmit and receive using a couple of transmitters and receivers will still do the job of distance measurement, but if you require precision, then prefer the triangulation method

iii. Laser range Sensor: Laser light is transmitted and the reflected light is captured and analyzed. Distance is measured by calculating the speed of light and time taken for the light to reflect back to the receiver. These sensors are very useful for longer distances.

iv. Encoders: These sensors (not actually sensors, but a combination of different components) convert angular position of a shaft or wheel into an analog or digital code. The most popular encoder is an optical encoder which includes a rotational disk, light source and a light detector (generally an IR transmitter and IR receiver). The rotational disk has transparent and opaque pattern (or just black and white pattern) painted or printed over it. When the disk rotates along with the wheel the emitted light is interrupted generating a signal output. The number of times the interruption happens and the diameter of the wheel can together give the distance travelled by the robot.

v. Stereo Camera: Two cameras placed adjacent to each other can provide depth information using its stereo vision. Processing the data received from a camera is difficult for a robot with minimal processing power and memory. If opted for, they make a valuable addition to your robot.

There are other stretch and bend sensors which are also capable of measuring distance. But, their range is so limited that they are almost useless for mobile robots.

7. Pressure Sensors

As the name suggests, pressure sensor measures pressure. Tactile pressure sensors are useful in robotics as they are sensitive to touch, force and pressure. If you design a robot hand and need to measure the amount of grip and pressure required to hold an object, then this is what you would want to use.

8. Tilt Sensors

Tilt sensors measure tilt of an object. In a typical analog tilt sensor, a small amount of mercury is suspended in a glass bulb. When mercury flows towards one end, it closes a switch which suggests a tilt.

9. Navigation / Positioning Sensors

The name says it all. Positioning sensors are used to approximate the position of a robot, some for indoor positioning and few others for outdoor positioning.

i. GPS (Global Positioning System): The most commonly used positioning sensor is a GPS. Satellites orbiting our earth transmit signals and a receiver on a robot acquires these signals and processes it. The processed information can be used to determine the approximate position and velocity of a robot. These GPS systems are extremely helpful for outdoor robots, but fail indoors. They are also bit expensive at the moment and if their prices fall, very soon you would see most robots with a GPS module attached.

ii. Digital Magnetic Compass: Similar to a handheld magnetic compass, Digital Magnetic compass provides directional measurements using the earth's magnetic field which guides your robot in the right direction to reach its destination. These sensors are cheap compared to GPS modules, but a compass works best along with a GPS module if you require both positional feedback and navigation. Philips KMZ51 is sensitive enough to detect earth's magnetic field.

iii. Localization: Localization refers to the task of automatically determining the location of a robot in complex environment. Localization is based on external elements called landmarks which can be either artificially placed landmarks, or natural landmark. In the first approach, artificial landmarks or beacons are placed around the robot, and a robot's sensor captures these signals to determine its exact location. Natural landmarks can be doors, windows, walls, etc. which are sensed by a robots sensor / vision system (Camera). Localization can be achieved using beacons which generate Wi-Fi, Bluetooth, Ultrasound, Infrared, Radio transmissions, Visible Light, or any similar signal.

10. Acceleration Sensor

An accelerometer is a device which measures acceleration and tilt. There are two kinds of forces which can affect an accelerometer: Static force and Dynamic Force

- Static Force: Static force is the frictional force between any two objects. For example earth's gravitational force is static which pulls an object towards it. Measuring this gravitational force can tell you how much your robot is tilting. This measurement is exceptionally useful in a balancing robot, or to tell you if your robot is driving uphill or on a flat surface.

- Dynamic force: Dynamic force is the amount of acceleration required to move an object. Measuring this dynamic force using an accelerometer tells you the velocity/speed at which your robot is moving. We can also measure vibration of a robot using an accelerometer, if in any case you need to.

Accelerometer comes in different flavors. Always select the one which is most appropriate for your robot. Some of the factors which you need to consider before selecting an accelerometer are:

i. Output Type: Analog or Digital

ii. Number of Axis: 1,2 or 3

iii. Accelerometer Swing: ±1.5g, ±2g, ±4g, ±8g, ±16g

iv. Sensitivity: Higher or Lower (Higher the better)

v. Bandwidth

11. Voltage Sensors

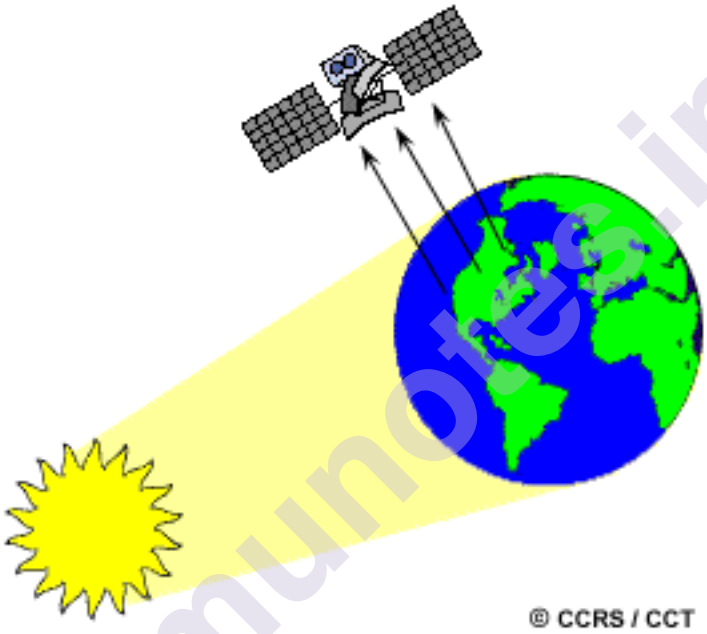
Voltage sensors typically convert lower voltages to higher voltages, or vice versa. One example is a general Operational-Amplifier (Op-Amp) which accepts a low voltage, amplifies it, and generates a higher voltage output. Few voltage sensors are used to find the potential difference between two ends (Voltage Comparator). Even a simple LED can act as a voltage sensor which can detect a voltage difference and light up. (not considering current requirements here)

12. Current Sensors

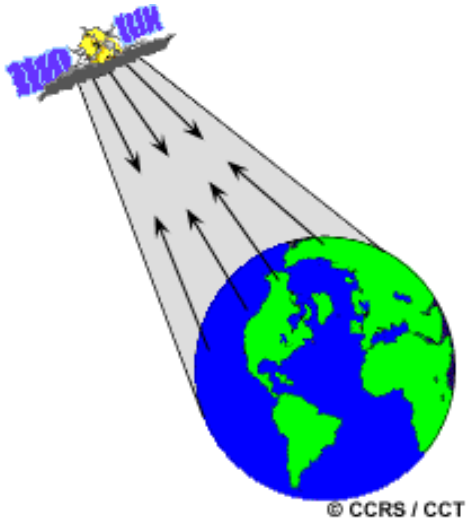
Current sensors are electronic circuits which monitor the current flow in a circuit and output either a proportional voltage or a current. Most current sensors output an analog voltage between 0V to 5V which can be processed further using a microcontroller.

4.6 PASSIVE VS. ACTIVE SENSING

So far, throughout this chapter, we have made various references to the sun as a source of energy or radiation. The sun provides a very convenient source of energy for remote sensing. The sun's energy is either reflected, as it is for visible wavelengths, or absorbed and then re-emitted, as it is for thermal infrared wavelengths. Remote sensing systems which measure energy that is naturally available are called passive sensors. Passive sensors can only be used to detect energy when the naturally occurring energy is available. For all reflected energy, this can only take place during the time when the sun is illuminating the Earth. There is no reflected energy available from the sun at night. Energy that is naturally emitted (such as thermal infrared) can be detected day or night, as long as the amount of energy is large enough to be recorded.



Active sensors, on the other hand, provide their own energy source for illumination. The sensor emits radiation which is directed toward the target to be investigated. The radiation reflected from that target is detected and measured by the sensor. Advantages for active sensors include the ability to obtain measurements anytime, regardless of the time of day or season. Active sensors can be used for examining wavelengths that are not sufficiently provided by the sun, such as microwaves, or to better control the way a target is illuminated. However, active systems require the generation of a fairly large amount of energy to adequately illuminate targets. Some examples of active sensors are a laser fluorosensor and a synthetic aperture radar (SAR).



4.7 SUMMARY

- An industrial robot is comprised of a robot manipulator, power supply, and controllers. Robotic manipulators can be divided into two sections, each with a different function: Robot Arm and Body
- Robot manipulators are created from a sequence of link and joint combinations. The links are the rigid members connecting the joints, or axes.
- The end effector means the last link (or end) of the robot. At this endpoint, the tools are attached. In a wider sense, an end effector can be seen as the part of a robot that interacts with the work environment. End effectors may consist of a gripper or a tool.
- A gripper is the mechanical or electrical End Of Arm Tooling (EOAT) device that enables the manipulation of an object. Or, in other words, it is a machine's controllable "hand" that grasps and releases parts that are being moved by the automation. There are many different types and sizes of grippers designed to pick up a wide variety of parts and materials.
- Two or three degrees of freedom are normally used:
  - Roll
  - Pitch
  - Yaw
- End effectors are the last part of a robot arm. They are directly connected to the wrist and have two basic functions: to grasp the objects or to do the work with the objects.
- Mechanical grippers are grippers that use mechanical fingers to manipulate objects

- Vacuum grippers provide gripping through suction cups and are mostly used for handling workpieces with uneven surfaces or irregular shapes.
- The pneumatic gripper is popular due to its compact size and light weight. It can easily be incorporated into tight spaces, which can be helpful in the manufacturing industry.
- The hydraulic gripper provides the most strength and is often used for applications that require significant amounts of force.
- **Teleoperation** (or **remote operation**) indicates operation of a system or machine at a distance.

4.8 QUESTIONS

1. Define robot.
2. Write a short note on End effectors
3. What are the Basic Classification of End Effectors.
4. Write a short note on **Vacuum grippers**
5. Write a short note on **Adhesive grippers**
6. Write a short note on Vacuum Grippers
7. Write a short note on Pneumatic Grippers
8. What do you mean by Teleoperation ?
9. What is Sensors and it types?
10. Differntiate between Active Sensors and Passive Sensors.

4.9 REFERENCE FOR FURTHER READING

- https://www.robots.com/faq/what-is-a-robot-manipulator
- https://www.tthk.ee/inlearcs/7-robot-end-of-arm-tooling/
- http://www.robotplatform.com/knowledge/sensors/types_of_robot_sensors.html
- https://roboticsbiz.com/sensors-in-robotics-7-common-sensors-used-in-robots/
- https://www.nrcan.gc.ca/maps-tools-publications/satellite-imagery-air-photos/remote- sensing-tutorials/introduction/passive-vs-active-sensing/14639
- https://www.smartvrlab.nl/3dof-vs-6dof-in-vr/

❄❄❄❄❄❄❄

5

SONAR, LASERS AND CAMERAS

Unit Structure

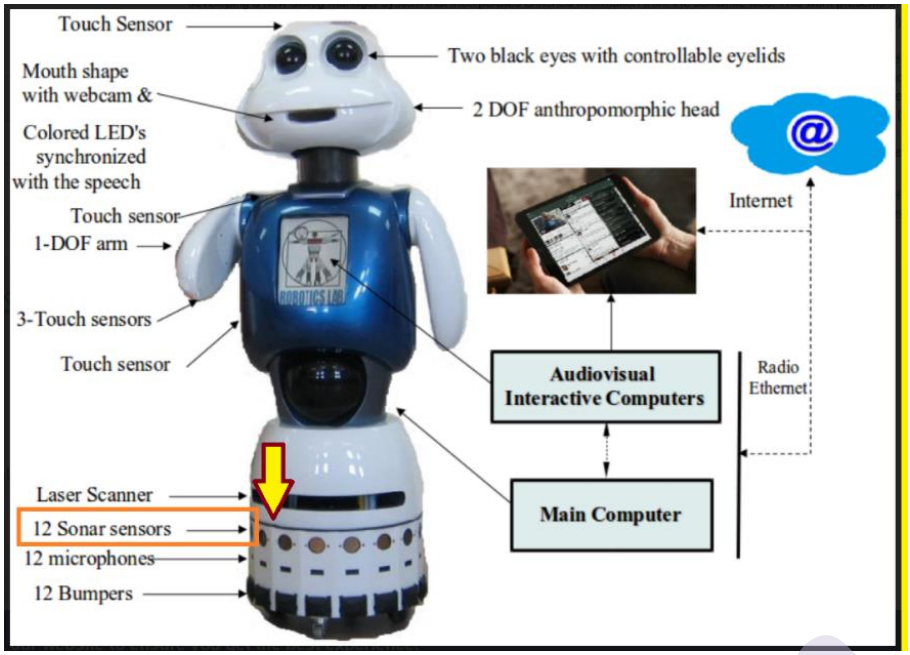
5.0 OBJECTIVE

In this chapter, we will learn about ultrasound, laser, and vision sensors, some of the most commonly used complex sensors in robotics.

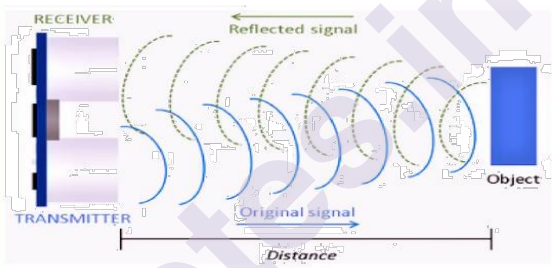
Ultrasonic and Sonar sensing, Specular Reflection, Laser Sensing, Visual Sensing, Cameras, Edge Detection, Motion Vision, Stereo Vision, Biological Vision, Vision for Robots, Feedback or Closed Loop Control: Example of Feedback Control Robot, Types of feedback control, Feed forward or Open loop control.

5.1 ULTRASONIC OR SONAR SENSING

Ultrasound literally means “beyond sound,” from the Latin ultra for “beyond” (used here in the same way as in “ultraviolet” and “ultraconservative”) and refers to a range of frequencies of sound that are beyond human SONAR hearing. It is also called sonar, from so(und) na(vigation and) r(anging). Figure below shows a mobile robot equipped with sonar sensors. The process of finding your (or a robot’s) location based on sonar is called echolocation. Echolocation works just the way the name sounds (no pun intended): sound bounces off objects and forms echoes that are used to find one’s place in the environment. That’s the basic principle. But before we get into the details, let’s first consider some examples



Ultrasonic distance sensor HC-SR04 Sonar Sensor



The principle of echolocation comes from nature, where it is used by several species of animals. Bats are the most famous for using sophisticated echolocation; cave bats that dwell in nearly total darkness do not use vision (it would not do them much good), but rely entirely on ultrasound. They emit and detect different frequencies of ultrasound, which allows them to fly effectively in very crowded caves that have complicated structures and are packed with hundreds of other flying and hanging bats. They do all this very quickly and without any collisions. Besides flying around in the dark, bats also use echolocation to catch tiny insects and find mates. Dolphins are another species known for sophisticated echolocation. What used to be secret research is now standard in aquarium shows: blindfolded dolphins can find small fish and swim through hoops and mazes by using echolocation. As usual, biological sensors are vastly more sophisticated than current artificial ones (also called "synthetic," because they have been "synthesized," not because they are made out of synthetics); bat and dolphin sonars are much more complex than artificial/synthetic sonars used in robotics and other applications. Still, synthetic sonars are quite useful.

So how do they work? Artificial ultrasound sensors, or sonars, are based on the time-of-flight prin-ciple, meaning they measure the time it takes something (in this case sound) to travel ("fly"). Sonars are active sensors consisting of an emitter and a detector. The emitter produces a chirp or

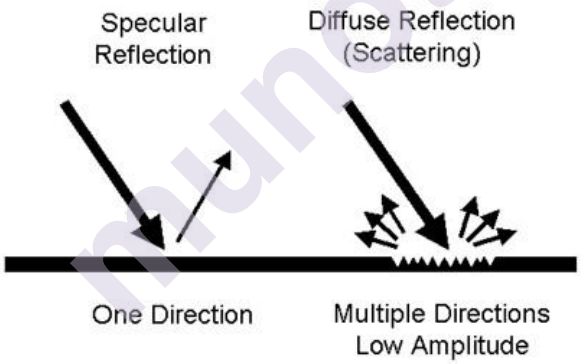
ping of ultrasound frequency. The sound travels away from the source and, if it encounters a barrier, bounces off it (i.e., reflects from it), and perhaps returns to the receiver (microphone). If there is no barrier, the sound does not return; the sound wave weakens (attenuates) with distance and eventually breaks down. If the sound does come back, the amount of time it takes for it to return can be used to calculate the distance between the emitter and the barrier that the sound encountered. Here is how it works: a timer is started when the chirp is emitted, and is stopped when the reflected sound returns. The resulting time is then multiplied by the speed of sound and divided by two. Why? Because the sound traveled to the barrier and back, and we are only trying to determine how far away the barrier is, its one-way distance. This computation is very simple, and relies only on knowing the speed of sound, which is a constant that varies only slightly due to ambient temperature. At room temperature, sound travels 1.12 feet per millisecond. Another way to put it is that sound takes 0.89 milliseconds to travel the distance of 1 foot. This is a useful constant to remember. The hardware for sonar sensing most commonly used in robotics is the Polaroid Ultrasound Sensor, initially designed for instant cameras. (Instant cameras were popular before digital cameras were invented, since they provided instant photos; otherwise people had to wait for film to be developed, which took at least a day, unless they a personal film development lab.) The physical sensor is a round transducer, approximately 1 inch in diameter, that TRANSDUCER emits the chirp/ping and receives the sound (echo) that comes back. A transducer is a device that transforms one form of energy into another. In the case of the Polaroid (or other ultrasound) transducers, mechanical energy is converted into sound as the membrane of the transducer flexes to produce a ping that sends out a sound wave that is inaudible to humans. You can actually hear most robot sonars clicking but what you hear is the movement of the emitter (the membrane), not the sound being sent out. The hardware (electronics) of ultrasound sensors involves relatively high power, because significant current is needed for emitting each ping. Importantly, the amount of current required is much larger than what computer processors use. This is just one of many practical examples showing why it is a good idea to separate the power electronics of a robot's sensing and actuation mechanisms from those of its controller processor. Otherwise, the robot's brain might have to literally slow down in order for the body to senseor move. The Polaroid ultrasound sensor emits sound that spreads from a 30-degree sound cone in all directions, and at about 32 feet attenuate to a point that they do not return to the receiver, giving the sensor a 32-foot range. The range of an ultrasound sensor is determined by the signal strength of the emitter, which is designed based on the intended use of the sensor. For robots (and for instant cameras, as it happens), the range of 32 feet is typically sufficient, especially for indoor environments.

5.2 SPECULAR REFLECTION:

Sonar sensing is based on the emitted sound wave reflecting from surfaces and returning to the receiver. But the sound wave does not necessarily bounce off the nearest surface and come right back, as we might hope it

would. Instead, the direction of reflection depends on several factors, including the properties of the surface (how smooth it is) and the incident angle of the sound beam and the surface (how sharp it is).

A major disadvantage of ultrasound sensing is its susceptibility to specular reflection. Specular reflection is the reflection from the outer surface of the object; this means the sound wave that travels from the emitter bounces off multiple surfaces in the environment before returning to the detector. This is likely if the surface it encounters is smooth and if the angle between the beam and the surface is small. The smaller the angle, the higher the probability that the sound will merely graze the surface and bounce off, thus not returning to the emitter but moving on to other surfaces (and potentially even more grazing bounces) before returning to the detector, if it returns at all. This bouncing sound generates a false faraway reading, one that is much longer than the straight-line distance between the robot (its sonar sensor) and the surface. The smoother the surface is, the more likely the sound is to bounce off. In contrast, rough surfaces produce more irregular reflections, which are more likely to return to the emitter. Think of it this way: as the sound hits a rough surface, it scatters, bouncing back at various angles relative to the various facets and grooves and features on the surface. At least some of the reflections are likely to go back to the emitter, and thus provide a rather accurate distance measure. In contrast, as the sound hits a uniformly smooth surface (a specular one), it may graze or bounce off it uniformly in a direction away from the detector.



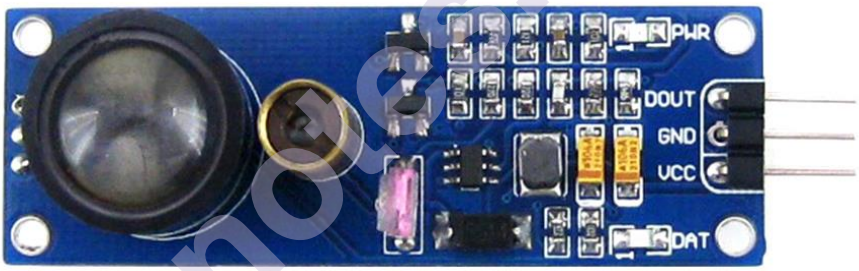
Specularity is a property of light as well as of sound, which adds to the challenges of machine vision; In the worst case of specular reflection, the sound bounces around the environment and does not return to the detector, thus fooling the sensor into detecting no object/barrier at all, or one that is far away instead of nearby. This could happen in a room full of mirrors, as at a carnival, or full of glass cases, as at a museum. A crude but effective way to combat specular reflection is to alter the environment in which the sonar-based robot has to navigate by making the surfaces less reflective.

In general, altering the environment to suit the robot is not a great idea and it is often not even possible (such as under the ocean or in space). How else can we get around the specular reflection problem? One solution is to use phased arrays of sensors to gain more accuracy. The basic idea is to use multiple sensors covering the same physical area, but activated out of

phase. Ultrasound sensors have been successfully used for very sophisticated robotics applications, including mapping complex outdoor terrain and indoor structures. Sonars remain a very popular, affordable ranging sensor choice in mobile robotics.

5.3 LASER SENSING

Lasers emit highly amplified and coherent radiation at one or more frequencies. The radiation may be in the visible spectrum or not, depending on the application. For example, when laser sensors are used as burglar detectors, they are typically not visible. Laser range sensors can be used through the time-of-flight principle, just like sonars. You can immediately guess that they are much faster, since the speed of light is quite a bit greater than the speed of sound. This actually causes a bit of a problem when lasers are used for measuring short distances: the light travels so fast that it comes back more quickly than it can be measured. The time intervals for short distances are on the order of nanoseconds and can't be measured with currently available electronics. As an alternative, phase-shift measurements, rather than time-of-flight, are used to compute the distance.



Lasers are different from sonars in many other ways stemming from the differences in the physical properties of sound and light. Lasers involve higher-power electronics, which means they are larger and more expensive. They are also much (much, much) more accurate. For example, a popular laser sensor, the SICK LMS200 scanning laser rangefinder, has a range of 8m and a 180-degree field of view. The range and bearing of objects can be determined to within 5mm and 0.5 degrees. The laser can also be used in a long-range mode (up to 80m), which results in a reduction in accuracy of only about 10cm. Another key distinction is that the emitted laser light is projected in a beam rather than a cone; the spot is small, about 3mm in diameter. Because lasers use light, they can take many more measurements than sonar can, thereby providing data with a higher resolution. Resolution refers to the process of separating or breaking something into its constituent parts.

In mobile robotics, simple sensors are usually most popular, and lasers, even planar ones, are not sufficiently affordable or portable for some applications.

5.4 VISUAL SENSING

Seeing requires a visual sensor, something akin to biological eyes. Cameras are the closest thing to natural eyes that we have available among synthetic sensors. Needless to say, any/all biological eyes are more complex than any cameras we have today. But to be fair, seeing is not done by eyes only, but largely by the brain. Eyes provide the information about the incoming light patterns, and the brain processes that information in complex ways to answer questions such as "Where are my car keys?" and make decisions such as "Stop right now; there they are, between the sofa cushions!" The research field that deals with vision in machines, including robots, is called, appropriately, machine vision. Robots have particular perceptual needs related to their tasks and environments, and so some parts of machine vision research are relevant and useful for robotics, while others have proven not to be, even if they are very useful for other applications. Therefore, machine vision and robotics are two separate fields of research with some overlap of interests, problems, and uses. raditionally, machine vision has concerned itself with answering the questions "What is that?", "Who is that?" and "Where is that?" To answer these, the approach has been to reconstruct what the world was like when the camera took its picture, in order to understand the picture, so that the question can be answered. After decades of research in machine vision, we know that visual reconstruction is an extremely difficult problem. Fortunately for robotics, it is not the problem that robots really need to solve. Instead, robots are typically concerned with acting so as to achieve their goals or, put more simply, with doing the right thing. Instead of answering the machine vision questions above, they need to answer action-related questions of the type: "Should I keep going or turn or stop?", "Should I grab or let go?", "Where do I turn?", and so on. It is usually not necessary to reconstruct the world in order to answer those questions.

5.5 CAMERAS

Cameras are biomimetic meaning they imitate biology, in that they work somewhat the way eyes do. Light, scattered from objects in the environment (which are collectively called the scene), goes through an opening (the iris, which is in the simplest case a pinhole, but usually is a lens) and hits the image plane. The image plane corresponds to the retina of the biological eye, which is attached to numerous light-sensitive (photosensitive) elements called rods and cones. These in turn are attached to nerves that perform early vision, the first stages of visual image processing, and then pass information on to other parts of the brain to perform high-level vision processing, everything else that is done with the visual input.

In machine vision, the computer needs to make sense out of the information on the image plane. If the camera is very simple, and uses a tiny pinhole, then some computation is required to determine the projection of the objects from the environment onto the image plane (note

that they will be inverted). If a lens is involved (as in vertebrate eyes and real cameras), then more light can get in, but at the price of being focused; only objects a particular range of distances from the lens will be in focus. This range of distances is called the camera's depth of field. The image plane is usually subdivided into equal parts called pixels, typically arranged in a rectangular grid. In a typical camera there are 512 × 512 pixels on the image plane. For comparison, there are 120×10^6 rods and 6×10^6 cones in the human eye. The projection of the scene on the image plane is called, not surprisingly, the image. The brightness of each pixel in the image is proportional to the amount of light that was reflected into the camera by the part of the object or surface that projects to that pixel, called the surface patch. Diffuse reflection consists of the light that penetrates into the object, is absorbed, and then comes back out.

Processing any time series information over time is pretty complicated. In the case of machine vision, each individual snapshot in time is called a frame, and getting frames out of a time series is not simple. In fact, it involves specialized hardware, called a frame grabber, a device that captures a single frame from a camera's analog video signal and stores it as a digital image. Now we are ready to proceed with the next step of visual processing, called image processing.

5.6 EDGE DETECTION

The typical first step (early vision) in image processing is to perform edge detection, to find all the edges in the image. In machine vision, an edge is defined as a curve in the image plane across which there is a significant change in the brightness. More intuitively, finding edges is about finding sharp changes in pixel brightness. Finding changes mathematically is done by taking derivatives. A simple approach to finding edges, then, is to differentiate the image and look for areas where the magnitude of the derivative is large, which indicates that the difference in the local brightness values is also large, likely due to an edge. Since it is impossible to distinguish "real" edges from those resulting from shadows simply by looking at pixel brightness/intensity in the image, some other method has to be used to do better. Unlike shadows, noise produces sudden and spurious intensity changes that do not have any meaningful structure. This is actually a good thing, since noise appears as peaks in the intensities, and those peaks can be taken away by a process called "smoothing."

To perform smoothing, we apply a mathematical procedure called convolution, which finds and eliminates the isolated peaks. Convolution applies a filter to the image; this is called convolving the image. This type of mathematical filter is really the same, in principle, as a physical filter, in that the idea is to filter out the unwanted things (in this case the spurious peaks that come from visual noise) and let through the good stuff (in this case the real edges). The process of finding real edges involves convolving the image with many filters with different orientations. Think back to the previous chapter, when we talked about polarized light filters. Those were physical filters, and here we are talking about mathematical

filters, but both have the same function: to separate out a particular part of the signal. In the case of the polarized filters, we were looking for a particular frequency of light; in the case of edge detection filters, we are looking for intensities with particular orientations.

Once we have edges, the next thing to do is try to find objects among all those edges. Segmentation is the process of dividing or organizing the image into parts that correspond to continuous objects.

Suppose that your robot has a bunch of line drawings of chairs in its memory. Whenever it sees an object in the environment, it performs edge detection, which produces something like a very bad line drawing, and compares the outcome with those stored drawings to see if any of them match what it saw in the environment, which would indicate it saw a chair. Those stored drawings are called models and the process is called modelbased vision. Models can be stored in a variety of forms; line drawings are just one form. Even though 2D line drawings are relatively simple and intuitive, using model matching for recognizing them is still a complex process.

Models can vary from simple 2D line drawings to weirdly processed, mathematically distorted images that combine all the various views of the object to be recognized in a mathematical way. For example, some very successful face recognition systems use only a few views of the person's face, and then do some interesting math to produce a model that can then recognize that person from many more points of view. Face recognition is a very popular problem in machine vision, and model-based approaches seem very well suited for it, since faces do have repeatable features, such as two eyes, a nose, and a mouth, with relatively constant ratios of distances in between those features (for most people, at least).

Face recognition is one of the important things that your brain performs very effectively, and is in fact fine-tuned through evolution to do very well. This is so because humans are such social animals, for whom it is very important to know who is who in order to establish and maintain social order. Imagine if you could not recognize faces, what would your life be like? There is a neurological disorder that produces that very deficit in some people; it is called prosopagnosia, from the Greek prosop for "face" and agnosia for "not knowing." It is rare and, so far, incurable.

5.7 MOTION VISION

The movement of the body and the camera on it makes vision processing more challenging and motion vision is a set of machine vision approaches that uses motion to facilitate visual processing. If the vision system is trying to recognize static objects, it can take advantage of its own motion. By looking at an image at two consecutive time steps, and moving the camera in between, continuous solid objects (at least those that obey physical laws we know about) will move as one, and their brightness properties will be unchanged. Therefore, if we subtract two consecutive images from one another, what we get is the "movement" between the

two, while the objects stay the same. If other objects also move in the environment, such as other robots and people, the vision problem becomes much harder.

5.8 STEREO VISION

in nature, creatures have two eyes, giving them binocular vision. The main advantage of having two eyes is the ability to see in stereo. Stereo vision, formally called binocular stereopsis, is the ability to use the combined points of view from the two eyes or camers to reconstruct three-dimensional solid objects and to perceive depth. The term stereo comes from the Greek stereos meaning "solid," and so it applies to any process of reconstructing the solid from multiple signals.

In stereo vision, we get two images, which we can subtract from one another, as long as we know how the two cameras or eyes are positioned relative to each other. The human brain "knows" how the eyes are positioned, and similarly we, as robot designers, have control over how the cameras on the robot are positioned as well and can reconstruct depth from the two images. So if you can afford two cameras, you can get depth perception and reconstruct solid objects.

This is the way 3D glasses, which make images in movies look solid, work. In normal movies, the images come from a single projector and both of your eyes see the same image. In 3D movies, however, there are two different images from two different projectors. That's why when you try to watch a 3D movie without the special glasses, it looks blurry. The two images do not come together, on the screen or in your brain. But when you put on the glasses, the two come together in your brain and look 3D. How? The special glasses let only one of the projected images into each of your eyes, and your brain fuses the images from the two eyes, as it does for everything you look at.

The ability to perceive in 3D using stereo is fundamental to realistic human/animal vision, and so it is involved in a variety of applications from video games to teleoperated surgery. If you lose the use of one of your eyes, you will lose the ability to see depth and 3D objects. To see how important depth perception is, try catching a ball with one eye closed. It's hard, but not impossible. That is because your brain compensates for the loss of depth for a little while. If you put a patch over one eye for several hours or longer, you will start to trip and fall and reach incorrectly toward objects.

What other properties of the image can we find and use to help in object detection?

Consider texture. Sandpaper looks quite a bit different from fur, which looks quite a bit different from feathers, which look quite a bit different from a smooth mirror, and so on, because all reflect the light in very different ways. Surface patches that have uniform texture have consistent and almost identical brightness in the image, so we can assume they come

from the same object. By extracting and combining patches with uniform and consistent texture, we can get a hint about what parts of the image may belong to the same object in the scene. Somewhat similarly, shading, contours, and object shape can also be used to help simplify vision. In fact, anything at all that can be reliably extracted from a visual image has been used to help deal with the object recognition problem. This is true not only for machines but also (and first) for biological vision systems.

5.9 BIOLOGICAL VISION

The brain does an excellent job of quickly extracting the information we need from the scene. We use model-based vision to recognize objects and people we know. Without it, we find it hard to recognize entirely unexpected objects or novel ones, or to orient ourselves, as in the typical example of waking up and not knowing where you are. Biological model-based vision is of course different from machine vision, and it is still poorly understood, but it works remarkably well, as you can tell when you effortlessly recognize a face in the crowd or find a lost object in a pile of other stuff. We use motion vision in a variety of ways in order to better understand the world around us, as well as to be able to move around while looking and not having it all result in a big blur. The latter is done through the vestibular ocular reflex (VOR, in which your eyes stay fixed even though your head is moving, in order to stabilize the image. We are very good at recognizing shadows, textures, contours, and various other shapes. In a famous experiment performed by a scientist named Johansson in the 1970s, a few dots of light were attached to people's clothes and the people were videotaped as they moved in the dark, so only the movement of the dots was visible. Any person watching the dots could immediately tell that they were attached to moving humans, even if only very few light dots were used. This tells us that our brains are wired to recognize human motion, even with very little information. Incidentally, this depends on seeing the dots/people from the side. If the view is from the top, we are not able to readily recognize the activity. This is because our brains are not wired to observe and recognize from a top-down view. Birds' brains probably are, so somebody should do that experiment.

5.10 VISION FOR ROBOTS

Robot vision has more stringent requirements than some other applications of machine vision, and only slightly less demanding requirements than biological vision. Robot vision needs to inform the robot about important things: if it's about to fall down the stairs, if there is a human around to help/track/avoid, if it has finished its job, and so on. Since vision processing can be a very complex problem, responding quickly to the demands of the real world based on vision information is very difficult. It is not only impractical to try to perform all the above steps of image processing before the robot gets run over by a truck or falls down the stairs, but fortunately it may be unnecessary. There are good ways of simplifying the problem. Here are some of them:

1. Use color; look for specifically and uniquely colored objects, and recognize them that way (such as stop signs, human skin, etc.).

2. Use the combination of color and movement; this is called color blob tracking and is quite popular in mobile robotics. By marking important ob jects (people, other robots, doors, etc.) with salient (meaning "noticeable," "attention-getting"), or at least recognizable colors, and using movement to track them, robots can effectively get their work done without having to actually recognize objects.

3. Use a small image plane; instead of a full 512 × 512 pixel array, we can reduce our view to much less, for example, just a line (as is used in linear CCD cameras).

4. Combine other, simpler and faster sensors with vision. For example, IR cameras isolate people by using body temperature, after which vision can be applied to try to recognize the person. Grippers allow us to touch and move objects to help the camera get a better view. Possibilities are endless.

5. Use knowledge about the environment; if the robot is driving on a road marked with white or yellow lines, it can look specifically for those lines in the appropriate places in the image. This greatly simplifies following a road, and is in fact how the first, and still some of the fastest, robot road and highway driving is done.

   Complex sensors imply complex processing, so they should be used selectively, for tasks where they are required or truly useful.

5.11 FEEDBACK OR CLOSED LOOP CONTROL

Feedback control is a means of getting a system (a robot) to achieve and maintain a desired state, usually called the set point, by continuously comparing its current state with its desired state. Feedback refers to the information that is sent back, literally "fed back," into the system's controller. The desired state of the system, also called the goal state, is where the system wants to be. Not surprisingly, the notion of goal state is quite fundamental to goal-driven systems, and so it is used both in control theory and in AI, two very different fields, In AI, goals are divided into two types: achievement and maintenance.

Achievement goals are states the system tries to reach, such as a particular location, perhaps the end of a maze. Once the system is there, it is done, and need not do any more work. AI has traditionally (but not exclusively) concerned itself with achievement goals. Maintenance goals, on the other hand, require ongoing active work on the part of the system. Keeping a biped robot balanced and walking, for example, is a maintenance goal. If the robot stops, it falls over and is no longer maintaining its goal of walking while balanced. Similarly, following a wall is a maintenance goal. If the robot stops, it is no longer maintaining its goal of following a wall.

The goal state of a system may be related to internal or external state, or a combination of both. For example, the robot's internal goal may be to keep its battery power level in some desired range of values, and to recharge the battery whenever it starts to get too low. In contrast, the robot's external goal state may be to get to a particular destination, such as the kitchen. Some goal states are combinations of both, such as the goal state that requires the robot to keep its arm extended and balance a pole. The arm state is internal (although externally observable), and the state of the pole is external. The goal state can be arbitrarily complex and consist of a variety of requirements and constraints. Anything the robot is capable of achieving and maintaining is fair game as a goal. Even what is not doable for a robot can be used as a goal, albeit an unreachable one. The robot may just keep trying and never get there.

5.12 EXAMPLE OF FEEDBACK CONTROL ROBOT

How would you write a controller for a wall-following robot using feedback control?

The first step is to consider the goal of the task. In wall-following, the goal state is a particular distance, or range of distances, from a wall. This is a maintenance goal, since wall-following involves keeping that distance over time. Given the goal, it is simple to work out the error. In the case of wall following, the error is the difference between the desired distance from the wall and the actual distance at any point in time. Whenever the robot is at the desired distance (or range of distances), it is in the goal state. Otherwise, it is not. We are now ready to write the controller, but before we do that, we should consider sensors, since they will have to provide the information from which both state and error will be computed.

What sensor(s) would you use for a wall-following robot and what information would they provide?

Would they provide magnitude and direction of the error, or just magnitude, or neither? For example, a contact bump sensor would provide the least information (it's a simple sensor, after all). Such a sensor would tell the robot only whether it has hit a wall; the robot could sense the wall only through contact, not at a distance. An infra red sensor would provide information about a possible wall, but not the exact distance to it. A sonar would provide distance, as would a laser. A stereo vision system could also provide distance and even allow for reconstructing more about the wall, but that would most definitely be overkill for the task. As you can see, the sensors determine what type of feedback can be available to the robot. Whatever sensor is used, assume that it provides the information to compute distance-to-wall. The robot's goal is to keep distance-to-wall at a particular value or, more realistically, in a particular range of values (let's say between 2 and 3 feet). Now we can write the robot's feedback controller in the form of standard if-then-else conditional statements used in programming, like this one:

If distance-to-wall is exactly as desired,

then keep going.

If distance-to-wall is larger than desired,

then turn by 45 degrees toward the wall,

else turn by 45 degrees away from the wall.

Deciding how often to compute the error, how large a turning angle to use, and how to define the range of distances all depend on the specific parameters of the robot system: the robot's speed of movement, the range of the sensor(s), and the rate with which new distance-to-wall is sensed and computed, called the sampling rate. The calibration of the control parameters is a necessary, very important, and time-consuming part of designing robot controllers.

5.13 TYPES OF FEEDBACK CONTROL

The three most used types feedback control are proportional control (P), proportional derivative control (PD), and proportional integral derivative control (PID). These are commonly referred to as P, P D, and PID control. Let's learn about each of them and use our wall-following robot as an example system.

5.13.1 Proportional Control

The basic idea of proportional control is to have the system respond in proportion to the error, using both the direction and the magnitude of the error. A proportional controller produces an output o proportional to its input i, and is formally written as: $o = K_p i$

K_p is a proportionality constant, usually a constant that makes things work, and is specific to a particular control system.

What would a proportional controller for our wall fallowing robot look like?

It would use distance-to-wall as a parameter to determine the angle and distance and/or speed with which the robot would turn. The larger the error, the larger the turning angle and speed and/or distance; the smaller the error, the smaller the turning angle and speed and/or distance. In control theory, the parameters that determine the magnitude of the system's response are called gains. Determining the right gains is typically very difficult, and requires trial and error, testing and calibrating the system repeatedly.

If the value of a particular gain is proportional to that of the error, it is called proportional gain. Damping refers to the process of systematically decreasing oscillations. A system is properly damped if it does not oscillate out of control, meaning its oscillations are either completely avoided (which is very rare) or, more practically, the oscillations gradually

decrease toward the desired state within a reasonable time period. Gains have to be adjusted in order to make a system properly damped. This is a tuning process that is specific to the particular control system (robot or otherwise). Actuator uncertainty makes it impossible for a robot (or a human, for that matter) to know the exact outcome of an action ahead of time, even for a simple action such as "Go forward three feet." While actuator uncertainty keeps us from predicting the exact outcome of actions, we can use probability to estimate and make a pretty good guess, assuming we know enough about the system to set up the probabilities correctly.

5.13.2 Derivative Control

When the system is close to the desired state, it needs to be controlled differently than when it is far from it. Otherwise, the momentum generated by the controller's response to the error, its own correction, carries the system beyond the desired state and causes oscillations. One solution to this problem is to correct the momentum as the system approaches the desired state.

This is called the derivative term, because velocity is the derivative (the rate of change) of position. Thus, a controller that has a derivative term is called a D controller. A derivative controller produces an output o proportional to the derivative of its input i:

$o = K_d \frac{di}{dt}$ K_d is a proportionality constant, as before, but this time with a different name, so don't assume you can use the same number in both equations.

5.13.3 Integral Control

Another level of improvement can be made to a control system by introducing the so-called integral term, or I. The idea is that the system keeps track of its own errors, in particular of the repeatable, fixed errors that are called steady state errors. The system integrates (sums up) these incremental errors over time, and once they reach some predetermined threshold (once the cumulative error gets large enough), the system does something to compensate/correct. An integral controller produces an output o proportional to the integral of its input i:

$$o = K_f \int i(t)dt$$ K_f is a proportionality constant.

How do we apply integral control to our wall-following robot?

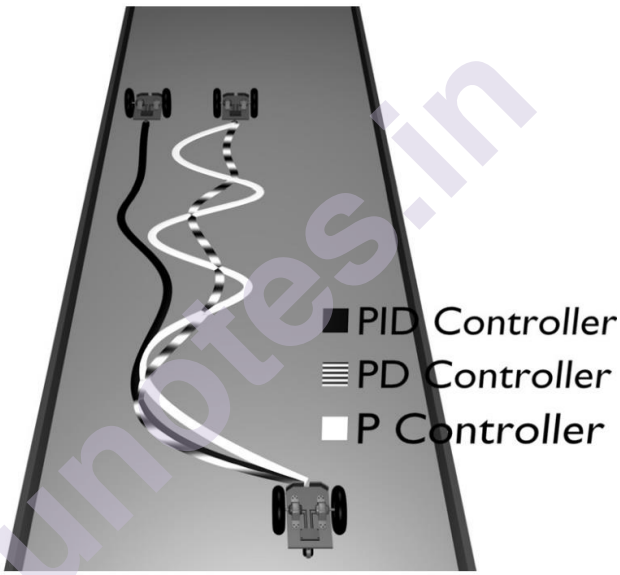
Consider a lawn-mowing robot which carefully covers the complete lawn by going from one side of the yard to the other, and moving over by a little bit every time to cover the next strip of grass in the yard. Now suppose that the robot has some consistent error in its turning mechanism, so whenever it tries to turn by a 90-degree angle to move over to the next strip on the grass, it actually turns by a smaller angle. This makes it fail to cover the yard completely, and the longer the robot runs, the worse its

coverage of the yard gets. But if it has a way of measuring its error, even only once it gets large enough (for example, by being able to detect that it is mowing already mowed grass areas a lot), it can apply integral control to recalibrate.

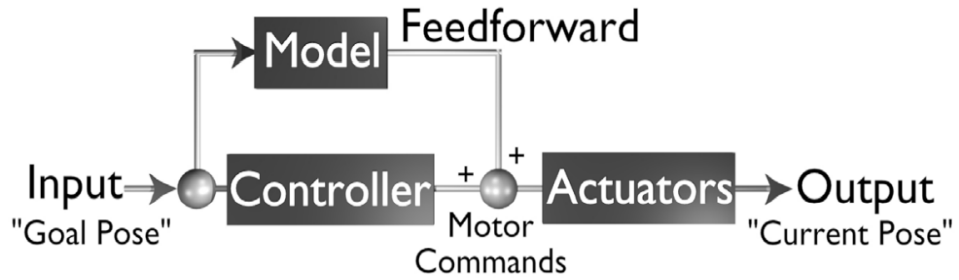
5.13.4 PD and PID Control

P D control is a combination, actually simply a sum, of proportional (P) and derivative (D) control terms: $o = K_p i + K_d \frac{di}{dt}$

P D control is extremely useful and applied in most industrial plants for process control. PID control is a combination (yes, again a sum) of proportional P, integral I, and derivative D control terms: $o = K_p i + K_f \int i(t)dt + K_d \frac{di}{dt}$



Before we move on, we should put control theory into perspective relative to robotics. As you have been reading and will read much more in the chapter to come, getting robots to do something useful requires many components. Feedback control plays a role at the low level, for example, for controlling the wheels or other continuously moving actuators. But for other aspects of robot control, in particular achieving higher-level goals (navigation, coordination, interaction, collaboration, human-robot interaction other approaches must come into play that are better suited for representing and handling those challenges. Those levels of robot control use techniques that come from the field of artificial intelligence, but that have come a long way from what AI used to be about in its early days.



5.14 FEED FORWARD OR OPEN LOOP CONTROL

Feedback control is also called closed loop control because it closes the loop between the input and the output, and provides the system with the error as feedback.

What is an alternative to closed loop control?

open loop control or feedforward control, does not use sensory feedback, and state is not fed back into the system. Thus the loop between the input and output is open, and not really a loop at all. In open loop control, the system executes the command that is given, based on what was predicted, instead of looking at the state of the system and updating it as it goes along, as in feedback control. In order to decide how to act in advance, the controller determines set points or sub-goals for itself ahead of time. This requires looking ahead (or looking forward) and predicting the state of the system, which is why the approach is called feedforward.

Open loop or feed-forward control systems can operate effectively if they are well calibrated and their environment is predictable and does not change in a way that affects their performance. Therefore they are well suited for repetitive, state-independent tasks.

5.15 SUMMARY

Sensor complexity is based on the amount of processing the data require. Sensors may also have complex mechanisms, but that is not what we are as concerned with in robotics. Ultrasound sensing is relatively high-power and is sensitive to specular reflections. Motion vision, stereo vision, model-based vision, active vision, and other strategies are employed to simplify the vision problem.

Feedback control aims to minimize system error, the difference between the current state and the desired state. Error is a complex concept with direction and magnitude, and cumulative properties.

❄❄❄❄❄❄❄

6

LANGUAGES FOR PROGRAMING ROBOT

Unit Structure

6.0 OBJECTIVE

Robot control can take place in hardware and in software, but the more complex the controller is, the more likely it is to be implemented in software. Robot brains typically involve computer programs of some type or another, running on the robot in real time. The brain should be physically on the robot, but that may be just prejudice based on how biological systems do it. In this chapter we will learn how to program a robot.

Topics : Algorithm, Architecture, The many ways to make a map, What is planning, Cost of planning, Reactive systems, Action selection, Subsumption architecture, How to sequence behavior through world, hybrid control, Behavior based control and Behavior Coordination, Behavior Arbitration, Distributed mapping, Navigation and Path planning.

6.1 ALGORITHM

Brains, robotic or natural, use their programs to solve problems that stand in the way of achieving their goals and getting their jobs done. The process of solving a problem using a finite (not endless) step-by-step procedure is called ALGORITHM an algorithm, and is named for the Iranian mathematician, Al-Khawarizmi (if that does not seem similar at all, it may be because we are not pronouncing his name correctly). The field of computer science devotes a great deal of research to developing and analyzing algorithms for all kinds of uses, from sorting numbers to creating, managing, and sustaining the Internet. Robotics also concerns itself with the development and analysis of algorithms (among many other

things, as we will learn) for uses that are relevant to robots, such as navigation, manipulation, learning, and many others. You can think of algorithms as the structure on which computer programs are based.

6.2 ARCHITECTURE

The term architecture is used here in the same way as in "computer architecture", where it means the set of principles for designing computers out of a collection of well-understood building blocks. Similarly, in robot architectures, there is a set of building blocks or tools at your disposal to make the job of robot control design easier. Robot architectures, like computer architectures, and of course "real" building architectures, where the term originally comes from, use specific styles, tools, constraints, and rules.

Any programming language worth its beans is called "Turing universal," which means that, in theory at least, it can be used to write any program. This concept was named after Alan Turing, a famous computer scientist from England who did a great deal of foundational work in the early days of computer science, around World War II. To be Turing universal, a programming language has to have the following capabilities: sequencing (a then b then c), conditional branching (if a then b else c), and iteration (for a=1 to 10 do something). Amazingly, with just those, any language can compute everything that is computable. Proving that, and explaining what it means for something to be computable, involves some very nice formal computer science theory.

Various programming languages have been used for robot control, ranging from general-purpose to specially designed ones. Some languages have been designed specifically to make the job of programming particular robot control architectures easier. Remember, architectures exist to provide guiding principles for good robot control design, so it is particularly convenient if the programming language can make following those principles easy and failing to follow them hard.

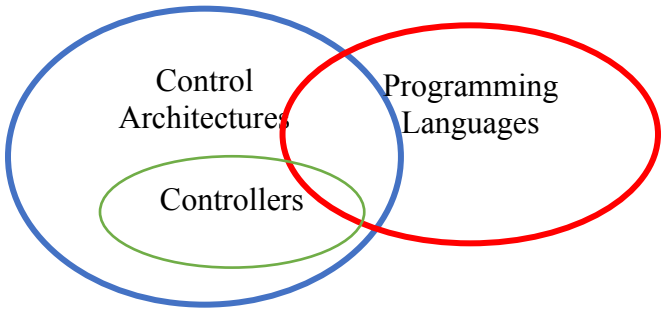


Figure: The relationship among control architectures, controllers, and programming languages.

Regardless of which language is used to program a robot, what matters is the control architecture used to implement the controller, because not all architectures are the same. On the contrary, as you will see, architectures impose strong rules and constraints on how robot programs are structured, and the resulting control software ends up looking very different. We have already agreed that there are numerous ways you can program a robot, and there are fewer, but still many, ways in which you can program a robot well. Conveniently, all those effective robot programs fall into one of the known types of control architectures. Better still, there are very few types of control (that we know of so far).

They are:

1. Deliberative control
2. Reactive control
3. Hybrid control
4. Behavior-based control.

Because robotics is a field that encompasses both engineering and science, there is a great deal of ongoing research that brings new results and discoveries. At robotics conferences, there are presentations, full of drawings of boxes and arrows, describing architectures that may be particularly well suited for some robot control problem. That's not about to go away, since there are so many things robots could do, and so much yet to be discovered. But even though people may keep discovering new architectures, all of them, and all the robot programs, fit into one of the categories listed above, even if the programmer may not realize it.

what are some important questions to consider that can help us decide which architecture to use. For any robot, task, and environment, many things need to be considered, including these:

- Is there a lot of sensor noise?
- Does the environment change or stay static?
- Can the robot sense all the information it needs? If not, how much can it sense?
- How quickly can the robot sense?
- How quickly can the robot act?
- Is there a lot of actuator noise?
- Does the robot need to remember the past in order to get the job done?
- Does the robot need to think into the future and predict in order to get the job done?
- Does the robot need to improve its behavior over time and be able to learn new things?

Control architectures differ fundamentally in the ways they treat the following important issues:

- Time: How fast do things happen? Do all components of the controller run at the same speed?
- Modularity: What are the components of the control system? What can talk to what?
- Representation: What does the robot know and keep in its brain?

6.3 THE MANY WAYS TO MAKE A MAP

Representation is the form in which information is stored or encoded in the robot. Representation is more than memory. In computer science and robotics, we think of memory as the storage device used to keep information. Just referring to memory does not say anything about what is stored and how it is encoded: is it in the form of numbers, names, probabilities, x,y locations, distances, colors? Representation is what encodes those important features of what is in the memory.

What is represented and how it is represented has a major impact on robot control. This is not surprising, as it is really the same as saying "What is in your brain influences what you can do." In this chapter we will learn about what representation is all about and why it has such an important role in robot control.

In principle, any internal state is a form of representation. In practice, what matters is the form and function of that representation, how it is stored and how it is used. "Internal state" usually refers to the "status" of the system itself, whereas "representation" refers to arbitrary information about the world that the robot stores.

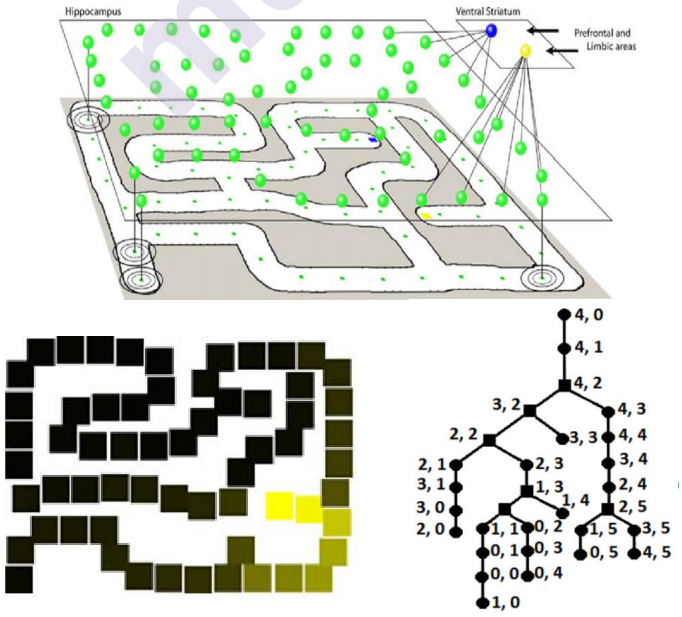


Figure: A few possible representation options for a maze-navigating robot.

WORLD MODEL Representation of the world is typically called a world model. A map is the most commonly used example of a world model. To give an illustration of how the representation of a particular world, its map, can vary in form, consider the problem of exploring a maze.

What can the robot store/remember to help it navigate a maze?

- The robot may remember the exact path it has taken to get to the end of the maze (e.g., "Go straight 3.4 cm, turn left 90 degrees, go straight 12 cm, turn right 90 degrees."). This remembered path is a type of map for getting through the maze. This is an odometric path.
- The robot may remember a sequence of moves it made at particular landmarks in the environment (e.g., "Left at the first junction, right at the second junction, straight at the third."). This is another way to store a path through the maze. This is a landmark-based path.
- The robot may remember what to do at each landmark in the maze (e.g., "At the green/red junction go left, at the red/blue junction go right', at the blue/orange junction go straight."). This is a landmark-based map; it is more than a path since it tells the robot what to do at each junction, no matter in what order it reaches the junction. A collection of landmarks connected with links is called a topological map because it describes the topology, the connections, among the landmarks. Topological maps are very useful.
- The robot may remember a map of the maze by "drawing it" using exact lengths of corridors and distances between walls it sees. This is a metric map of the maze, and it is also very useful.

The above are not nearly all the ways in which the robot can construct and store a model of the maze. However, the four types of models above already show you some important differences in how representations can be used. The first model, the odometric path, is very specific and detailed, and that it is useful only if the maze never changes, no junctions become blocked or opened, and if the robot is able to keep track of distances and turns very accurately. The second approach also depends on the map not changing, but it does not require the robot to be as specific about measurements, because it relies on finding landmarks (in this case, junctions). The third model is similar to the second, but connects the various paths into a landmark-based map, a network of stored landmarks and their connections to one another. Finally, the fourth approach is the most complicated because the robot has to measure much more about the environment and store much more information. On the other hand, it is also the most generally useful, since with it the robot can now use its map to think about other possible paths in case any junctions become blocked.

What Can the Robot Represent? Maps are just one of the many things a robot may want to represent or store or model in its "brain." For example, the robot may want to remember how long its batteries last, and to remind itself to recharge them before it is too late. This is a type of a self-model. Also, the robot may want to remember that traffic is heavy at particular times of day or in particular places in the environment, and to avoid

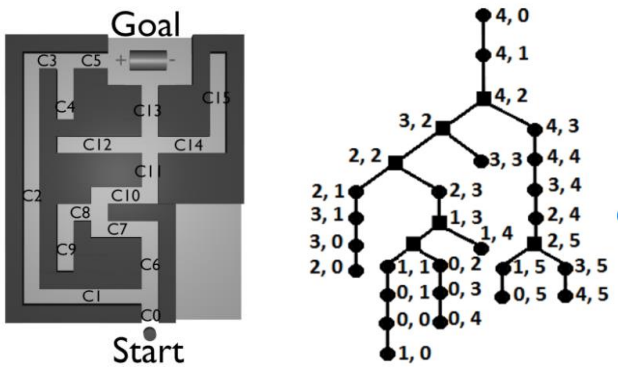
traveling at those times and in those places. Or it may store facts about other robots, such as which ones tend to be slow or fast, and so on. There are numerous aspects of the world that a robot can represent and model, and numerous ways in which it can do it. The robot can represent information about:

- Self: stored proprioception, self-limitations, goals, intentions, plans
- Environment: navigable spaces, structures
- Objects, people, other robots: detectable things in the world
- Actions: outcomes of specific actions in the environment
- Task: what needs to be done, where, in what order, how fast, etc.

Costs of Representing In addition to constructing and updating a representation, using a representation is also not free, in terms of computation and memory cost. Consider maps again: to find a path from a particular location in the map to the goal location, the robot must plan a path. This process involves finding all the free/navigable spaces in the map, then searching through those to find any path, or the best path, to the goal. Because of the processing requirements involved in the construction, maintenance, and use of representation, different architectures have very different properties based on how representation is handled, as we will see in the next few chapters. Some architectures do not facilitate the use of models (or do not allow them at all, such as reactive ones), others utilize multiple types of models (hybrid ones), still others impose constraints on the time and space allowed for the models being used (behavior-based ones). How much and what type of representation a robot should use depends on its task, its sensors, and its environment. As we will see when we learn more about particular robot control architectures, how representation is handled and how much time is involved in handling it turns out to be a crucial distinguishing feature for deciding what architecture to use for a given robot and task. So, as you would imagine, what is in the robot's head has a very big influence on what that robot can do.

6.4 WHAT IS PLANNING

Planning is the process of looking ahead at the outcomes of the possible actions, and searching for the sequence of actions that will reach the desired goal.



Search is an inherent part of planning. It involves looking through the available representation "in search of" the goal state. Sometimes searching the complete representation is necessary (which can be very slow, depending on the size of the representation), while at other times only a partial search is enough, to reach the first found solution. For example, if a robot has a map of a maze, and knows where it is and where it wants to end up (say at a recharging station at the end of the maze), it can then plan a path from its current position to the goal, such as those shown in figure above. The process of searching through the maze happens in the robot's head, in the representation of the maze (such as the one shown in figure above, not in the physical maze itself. The robot can search from the goal backwards, or from where it is forward. Or it can even search in both directions in parallel. That is the nice thing about using internal models or representations: you can do things that you can't do in the real world. Consider the maze in figure above, where the robot is marked with a black circle, and the goal with the recharging battery. Note that at each junction in the maze, the robot has to decide how to turn. The process of planning involves the robot trying different turns at each junction, until a path leads it to the battery. In this particular maze, there is more than one path to the goal from where the robot is, and by searching the entire maze (in its head, of course) the robot can find both of those paths, and then choose the one it likes better. Usually the shortest path is considered the best, since the robot uses the least time and battery power to reach it. But in some cases other criteria may be used, such as which path is the safest or the least crowded.

OPTIMIZATION: The process of improving a solution to a problem by finding a better one is called optimization.

OPTIMIZATION CRITERIA: As in the maze example above, various values or properties of the given problem can be optimized, such as the distance of a path; those values are called optimization criteria.

OPTIMIZING SEARCH Optimizing search looks for multiple solutions (paths, in the case of the maze), in some cases all possible paths.

6.5 COST OF PLANNING

Whenever a large state space is involved, planning is difficult. To deal with this fundamental problem, AI researchers have found various ways to speed things up. One popular approach is to use hierarchies of states, where first only a small number of "large," "coarse," or "abstract" states is considered; after that, more refined and detailed states are used in the parts of the state space where it really matters. The graph of the maze shown in figure above is an example of doing exactly that: it lumps all states within a corridor of the maze together and considers them as a single corridor state in the graph. There are several other clever methods besides this one that can speed up search and planning. These are all optimization methods for planning itself, and they always involve some type of compromise.

Deliberative, planner-based architectures involve three steps that need to be performed in sequence:

1. Sensing (S)

2. Planning (P)

3. Acting (A), executing the plan.

For this reason, deliberative architectures, which are also called SPA (sense- plan-act) architectures, shown in figure below, have serious drawbacks for robotics. Here is what they are:

Drawback 1: Time-Scale

Drawback 2: Space

Drawback 3: Information

Drawback 4: Use of Plans

Drawback 1: Time-Scale

If the planning process is slow compared with the robot's moving speed, it has to stop and wait for the plan to be finished, in order to be safe. So to make progress, it is best to plan as rarely as possible and move as much as possible in between. If planning is fast, then execution need not be open loop, since replanning can be done at each step; unfortunately, typically this is impossible for real-world problems and robots. Generating a plan for a real environment can be very slow.

Drawback 2: Space

It may take a great deal of space (i.e., memory storage) to represent and manipulate the robot's state space representation. The representation must contain all information needed for planning and optimization, such as distances, angles, snapshots of landmarks, views, and so on. Computer memory is comparatively cheap, so space is not as much of a problem as time, but all memory is finite, and some algorithms can run out of it. Generating a plan for a real environment can be very memory-intensive.

Drawback 3: Information

The representation used by the planner must be updated and checked as often as necessary to keep it sufficiently accurate for the task. Thus, the more information the better.

Generating a plan for a real environment requires updating the world model, which takes time.

Drawback 4: Use of Plans

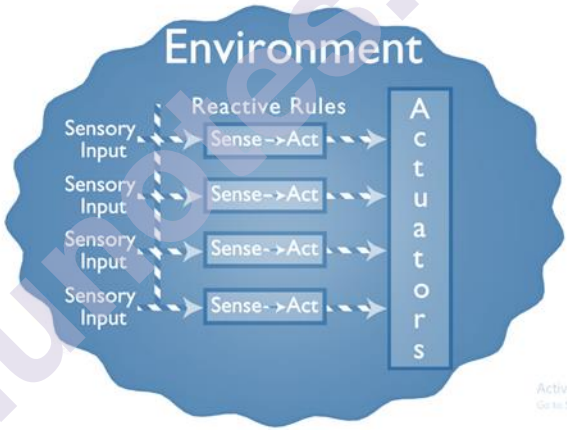
In addition to all of the above, any accurate plan is useful only if:

- The environment does not change during the execution of the plan in a way that affects the plan
- The robot knows what state of the world and of the plan it is in at all times
- The robot's effectors are accurate enough to execute each step of the plan in order to make the next step possible.

Executing a plan, even when one is available, is not a trivial process.

6.6 REACTIVE SYSTEMS

Reactive control is one of the most commonly used methods for robot control. It is based on a tight connection between the robot's sensors and effectors. Reactive systems use a direct mapping between sensors and effectors, and minimal, if any, state information. They consist of collections of rules that couple specific situations to specific actions, as shown in figure below:



Reactive systems consist of a set of situations (stimuli, also called conditions) and a set of actions (responses, also called actions or behaviors). The situations may be based on sensory inputs or on internal state. For example, a robot may turn to avoid an obstacle that is sensed, or because an internal clock indicated it was time to change direction and go to another area. These examples are very simple; reactive rules can be much more complex, involving arbitrary combinations of external inputs and internal state. The best way to keep a reactive system simple and straightforward is to have each unique situation (state) that can be detected by the robot's sensors trigger only one unique action of the robot. In such a design, the conditions are said to be mutually exclusive, meaning they exclude one another; only one can be true at a time.

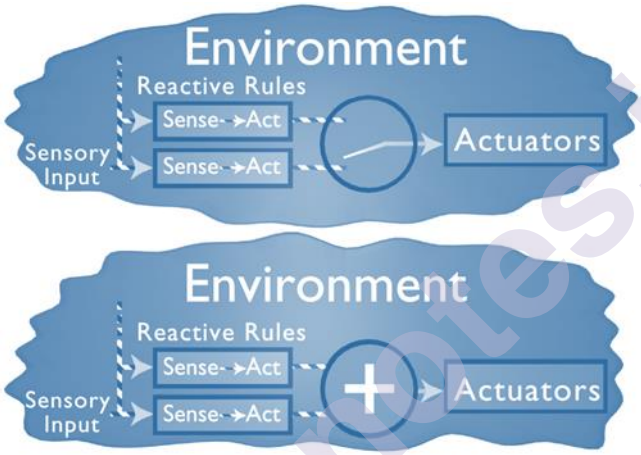
6.7 ACTION SELECTION

Action selection is the process of deciding among multiple possible actions or behaviors. It may select only one output action or may combine the actions to produce a result. These two approaches are called arbitration and fusion.

Command arbitration is the process of selecting one action or behavior from multiple candidates.

Command fusion is the process of combining multiple candidate actions or behaviors into a single output action/behavior for the robot.

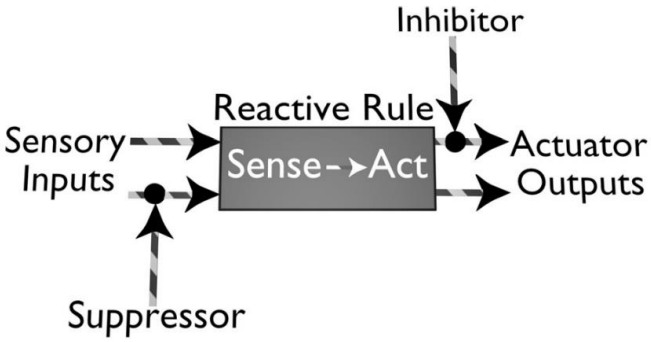
It turns out that action selection is a major problem in robotics, beyond reactive systems, and that there is a great deal of work (theory and practice) on different methods for command arbitration and fusion.



Reactive systems must be able to support parallelism, the ability to monitor and execute multiple rules at once. Practically, this means that the underlying programming language must have the ability to multitask, to execute several processes/rules/commands in parallel.

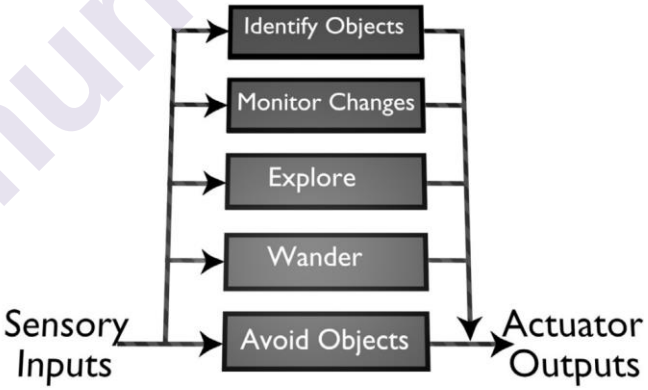
6.8 SUBSUMPTION ARCHITECTURE

The basic idea behind Subsumption Architecture is to build systems incrementally, from the simple parts to the more complex, all the while using the already existing components as much as possible in the new stuff being added. Here is how it works. Subsumption systems consist of a collection of modules or layers, each of which achieves a task. For example, they might be move-around, avoidobstacles, find-doors, visit-rooms, pick-up-soda-cans, and so on. All of the task-achieving layers work at the same time, instead of in sequence. This means the rules for each of them are ready to be executed at any time, whenever the right situation presents itself.



The name "Subsumption Architecture" comes from the idea that higher lay

How to sequence behavior through world ers can assume the existence of the lower ones and the goals they are achieving, so that the higher layers can use the lower ones to help them in achieving their own goals, either by using them while they are running or by inhibiting them selectively. In this way, higher layers "subsume" lower ones. The design of subsumption controllers is called bottom-up, because it progresses from the simpler to the more complex, as layers are added incrementally. This is good engineering practice, but its original inspiration came to the inventor of Subsumption Architecture from biology. Brooks was inspired by the evolutionary process, which introduces new abilities based on the existing ones. Genes operate using the process of mixing (crossover) and changing (mutation) of the existing genetic code, so complete creatures are not thrown out and new ones created from scratch; instead, the good stuff that works is saved and used as a foundation for adding more good stuff, and so complexity grows over time.



in Subsumption Architecture, we use strongly coupled connec- tions within layers, and loosely coupled connections between layers.

6.9 HYBRID CONTROL

Hybrid control involves the combination of reactive and deliberative control within a single robot control system. This combination means that fundamentally different controllers, time-scales (short for reactive, long for deliberative), and representations (none for reactive, explicit and elaborate world models for deliberative) must be made to work together effectively. And that, as we will see, is a tall order.

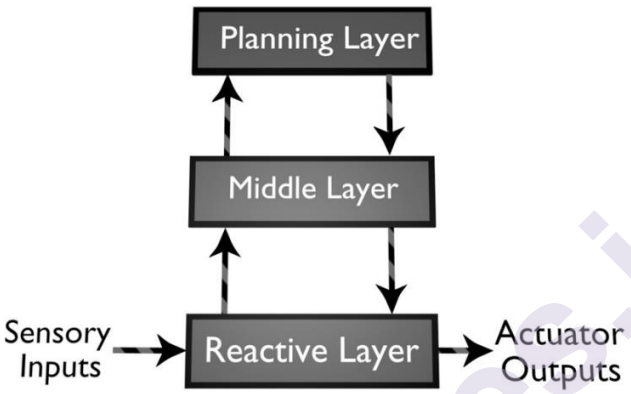
In order to achieve the best of both worlds, a hybrid system typically consists of three components, which we can call layers or modules (though they are not the same as, and should not be confused with, layers/modules used in reactive systems):

• A reactive layer

• A planner

• A layer that links the above two together.

As a result, hybrid architectures are often called three-layer architectures and hybrid systems, three-layer systems.



The middle layer has a hard job, because it has to:

- Compensate for the limitations of both the planner and the reactive system
- Reconcile their different time-scales
- Deal with their different representations
- Reconcile any contradictory commands they may send to the robot.

We already know how to do both of the component systems (reactive collision-free navigation and deliberative path planning), so how hard can it be to put the two together?

Here is how hard:

- What happens if the robot needs to deliver a medication to a patient as soon as possible, yet it does not have a plan for a short path to the patient's room? Should it wait for the plan to be computed, or should it move down the corridor (in which direction?) while it is still planning in its head?
- What happens if the robot is headed down the shortest path but suddenly a crew of doctors with a patient on a stretcher starts heading its way? Should it just stop and get out of the way in any direction, and wait as long as it may take, or should it start replanning an alternative path?

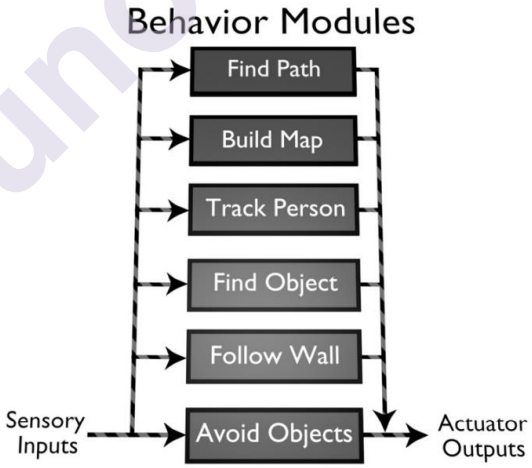
- What happens if the plan the robot computed from the map is blocked, because the map is out of date? • What happens if the patient was moved to another room without the robot knowing?
- What happens if the robot keeps having to go to the same room, and so has to replan that path or parts of it all the time?
- What if, what if, what if.

6.10 BEHAVIOR BASED CONTROL AND BEHAVIOR COORDINATION

Behavior-based control (BBC) grew out of reactive control, and was similarly inspired by biological systems.

The primary inspiration came from several main challenges:

- Reactive systems are too inflexible, incapable of representation, adaptation, or learning.
- Deliberative systems are too slow and cumbersome.
- Hybrid systems require complex means of interaction among the components.
- Biology seems to have evolved complexity from simple and consistent components.



Behavior-based control (BBC) involves the use of "behaviors" as modules for control. Thus, BBC controllers are implemented as collections of behaviors. The first property of behavior-based control to remember is that it's all about behaviors.

What is a behavior? There is a definite answer to that question, and in fact one of the strengths of BBC comes from different ways in which people have encoded and implemented behaviors, which are also sometimes called behavior-achieving modules. But don't assume that anything goes, and any piece of code can be a behavior.

There are some rules of thumb about behaviors and constraints on how to design them and what to avoid in implementing them:

- Behaviors achieve and/or maintain particular goals. A homing behavior achieves the goal of getting the robot to the home location. A wall-following behavior maintains the goal of following a wall.
- Behaviors are time-extended, not instantaneous. That means they take some time to achieve and/or maintain their goals. After all, it takes a while to go home or follow a wall.
- Behaviors can take inputs from sensors and also from other behaviors, and can send outputs to effectors and to other behaviors. This means we can create networks of behaviors that "talk to" each other.
- Behaviors are more complex than actions. While a reactive system may use simple actions like stop and turn-right, a BBC uses time-extended behaviors like the ones we saw above, as well as others like find-object, followtarget, get-recharged, hide-from-the-light, aggregate-with-your-team, find-mate, etc. behaviors can be designed at a variety of levels of detail or description. This is called their level of abstraction, because to abstract is to take details away and make things less specific. Behaviors can take different amounts of time and computation. In short, they are quite flexible, and that is one of the key advantages of BBC.

The power and flexibility of BBC comes not only from behaviors but also from the organization of those behaviors, from the way they are put together into a control system. Here are some principles for good BBC design:

- Behaviors are typically executed in parallel/concurrently, much as in reactive systems, in order to enable the controller to respond immediately when needed.
- Networks of behaviors are used to store state and to construct world models/representations. When assembled into distributed representations, behaviors can be used to store history and to look ahead into the future.
- Behaviors are designed so that they operate on compatible time-scales. This means it is not good BBC design to have some very fast behaviors and some very slow ones. Why not? Because that makes the system hybrid in terms of the time-scale, and that interfacing different time-scales is a challenging problem.

Behaviorbased systems have the following key properties:

1. The ability to react in real-time
2. The ability to use representations to generate efficient (not only reactive) behavior

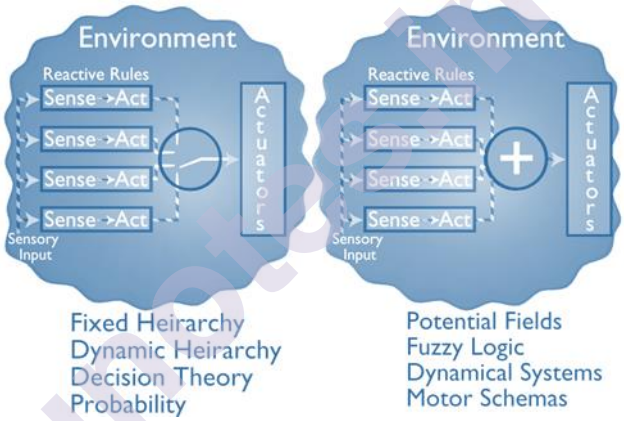
3. The ability to use a uniform structure and representation throughout the system (with no intermediate layer(s)).

Flocking, the behavior in which a group of robots moves together in a group. A robot that flocks with others does not necessarily need to have an internal flocking behavior. In fact, flocking can be implemented very elegantly in a completely distributed way.

6.11 BEHAVIOR ARBITRATION

Arbitration is the process of selecting one action or behavior from multiple possible candidates. Arbitration-based behavior coordination is also called competitive behaviour coordination, because multiple candidate behaviors compete but only one can win. Arbitration can be done with:

- A fixed priority hierarchy (behaviors have preassigned priorities)
- A dynamic hierarchy (behavior priorities change at run-time).



Dynamic arbitration usually involves computing some function to decide who wins. The function can be anything, including voting, fuzzy logic, probability, or spreading of activation, among many others.

6.12 NAVIGATION AND PATH PLANNING.

Navigation refers to the way a robot finds its way in the environment.

Getting from one place to another is remarkably challenging for a robot. In general, you will find that any robot controller spends a great deal of its code getting where it needs to be at any given time, compared with its other "high-level" goals. Getting any body part where it needs to be is hard, and the more complicated the robot's body, the harder the problem. The term "navigation" applies to the problem of moving the robot's whole body to various destinations. Although the problem of navigation has been studied in the domain of mobile robots (as well as flying and swimming robots), it applies to any robot that can move around. The body of the robot may be of any form; the locomotion mechanism takes care of moving the body appropriately, and the navigation mechanism tells it where to go.

Since a robot typically does not know exactly where it is, this makes it rather hard to know how to get to its next destination, especially since that destination may not be within its immediate sensory range. To better understand the problem, let's break it down into a few possible scenarios, in all of which the robot has to find some object.

- Suppose the robot has a map of its world which shows where the puck is. Suppose also that the robot knows where it is in its world, the map. What remains to be done to get to the puck is to plan a path between the robot's current location and the goal (the puck), then follow that plan. This is the path planning problem. Of course if anything goes wrong - if the map is incorrect or the world changes - the robot may have to update the map, search around, replan, and so on.

- Now suppose the robot has a map of its world which shows where the puck is, but the robot does not know where it is in the map. The first thing the robot must do is find itself in the map. This is the localization problem. Once the robot localizes within its map, i.e., knows where it is, it can plan the path, just as above.

- Now suppose the robot has a map of its world and knows where it is in its map, but does not know where the puck is in the map (or world, same thing). What good is the map if the location of the puck is not marked? Actually, it is a good thing indeed. Here is why. Since the robot does not know where the puck is, it will have to go around searching. Given that it has a map, it can use the map to plan out a good searching strategy, one that covers every bit of its map and is sure to find the puck, if one exists. This is the coverage problem.

- Now suppose the robot does not have a map of its world. In that case, it may want to build a map as it goes along; this is the mapping problem. Notice that not having a map does not mean the robot does not know where it is; you may not have a map of New York City, but if you are standing next to the Statue of Liberty, you know where you are: next to the Statue of Liberty. Basically, a robot's local sensors can tell it where it is if the location can be uniquely recognized, such as with a major landmark like the Statue of Liberty, or with a global positioning system (GPS). However, without a map you won't know how to get to the Empire State Building. To make the problem really fun, and also quite realistic for many real-world robotics domains, suppose the robot does not know where it is. Now the robot has two things it has to do: figure out where it is (localization) and find the puck (search and coverage).

- Now suppose that the robot, which has no map of its world and does not know where it is, chooses to build a map of its world as it goes along trying to localize and search for the puck. This is the simultaneous localization and mapping (SLAM) problem, also called concurrent mapping and localization (CML), but that term is

not as catchy or popular. This is a "chicken or egg" problem: to make a map, you have to know where you are, but to know where you are, you have to have a map. With SLAM, you have to do both at the same time.

To summarize, they are:

- Localization: finding out where you are
- Search: looking for the goal location
- Path planning: planning a path to the goal location
- Coverage: covering all of an area
- SLAM: localization and mapping at the same time.

One way to stay localized is through the use of odometry. Odometry comes from the Greek hodos meaning "journey" and metros meaning "measure," so it literally means measuring the journey. A more formal term for it is path integration. Robot odometry is usually based on some sensors of the movement of the robot's wheels, typically shaft encoders.

Search and Path Planning

Path planning involves finding a path from the robot's current location to the destination. This naturally involves the robot knowing its current location (i.e., being localized) and knowing the destination or goal location, both in a common frame of reference. It turns out that path planning is a pretty well understood problem, at least when all of the above information is available to the robot. Given a map of the environment and two special points on it (current and goal locations), the robot just has to "connect the dots." Typically, there are many possible paths between the start and the goal, and finding them involves searching the map. To make this searching computationally efficient, the map is usually turned into a graph, a set of nodes (points) and lines that connect them. Why bother? Because graphs are easy to search using algorithms that have been developed in computer science and artificial intelligence. A path planner may look for the optimal (the very best) path based on some criterion. A path may be optimal based on distance, meaning it is the shortest, it may be optimal based on danger, meaning it is the safest; or it may be optimal based on scenery, meaning it is the prettiest.

6.13 SUMMARY:

Roboticists try to avoid the undesirable but exploit the desirable emergent behavior. Different control architectures have different methods for exploiting or avoiding emergent behavior. Navigation involves a great many subproblems, which include odometry, localization, search and path planning, path optimization, and mapping. Each of these is a separate field of study.

❄❄❄❄❄❄❄

7

ARTIFICIAL INTELLIGENCE

Unit Structure

7.0 OBJECTIVE

Learn different searching techniques and find optimal solution.

7.1 INTRODUCTION

John McCarthy who has invented the word "Artificial Intelligence" in 1956 has defined AI as the science and engineering of making intelligence machine", especially intelligent computer programming.

Definition of AI: There are eight definition of AI. The definition of thinking is address to thought processes and reasoning. Definitions of acting address to behaviour process

Intelligence: Intelligence can be defined in many ways is a main characteristic of human being. it can be described as the capability to observe or infer information Using Intelligence human being is able to a take a decision, may be rational or irrational.

Artificial Intelligence (AI): Many human mental activities such as developing computer programs, working out mathematics, engaging in

rational reasoning, understanding language and interpreting it, even driving an automobile are said to be demand "intelligence". Several computers have been built that can perform such task as these.

AI is one of the newest fields in science and engineering and has a wide variety of application fields.AI applications range from the general fields like learning, perception and prediction to the specific field, such as writing stories. Proving mathematical, theorems, driving a bus on a crowded street, diagnoses and playing chess.

7.2. STATE SPACE SEARCH

7.2.1 Generate -and -Test

Generate -and -Test - Generate -and -Test is a search algorithm which uses a depth first search technique. It guarantees to find a solution in correct way. It produces complete solution and then testing is done.

Algorithm for generate -and -test

1. Generate possible solution which can either be a point in the problem space or path from the initial state
2. Test to see if this is possible solution is a real (actual) solution by comparing the state reached with the set of goal states.
3. If it a real solution then return the solution otherwise repeat from state 1.

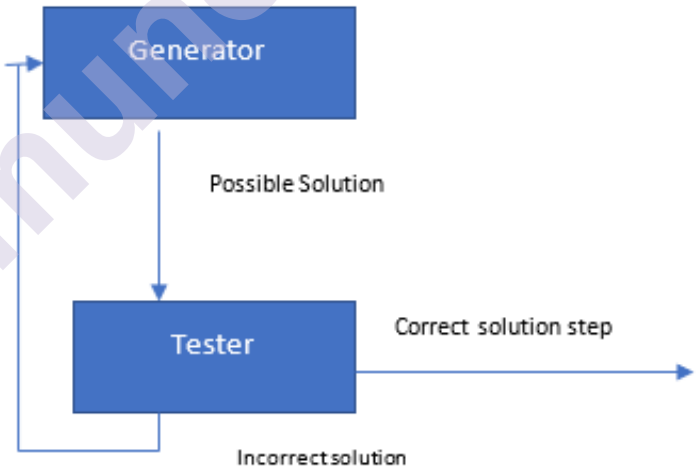


Fig 7.1.1. generate-And -Test

Is a set of all possible combination of a solution of a states given in the search tree .i.e to find a solution from a given initial state to the final state with successor state function. In general we may search graph rather than tree as then same state can be reached from multiple paths.

Construction of state space

State Space is implicitly defined as the Initial state, action & transition model together

The root node of a search tree is a initial node or start node of a tree. In this we need to check that the given state is a goal state.

If it is not a goal state then we need to explore the search tree and consider another node from current node. Considering the next node by applying successor function which generate new state from this we may get multiple states.

For each one of these we need to check goal test or repeat the expansion of each state

The state that we want to expand is depend upon the search strategies.

It is possible that some state can never lead to a goal state. Such a state we need not to expand. This decision is based on various conditions of the problem

Terminology used in search trees

- **Node in a tree :** It is a book keeping data structure to represent the structure configuration of a state in a search tree
- **State :** It reflects world configuration .It is mapping of sates and action to another new state
- **Transition model :**A description of what each action does. Eg. Function Result(S,a) returns a result from doing action state a in state S .It is also called successor function.
- **Fringe :** It is a collection of nodes that have been generated but not yet expanded
- **Leaf node** :Each node in fringe is leaf node (as it does not have further successor node)

Example :

a. The initial state

Fig 7.1.2

b) After Expanding Delhi

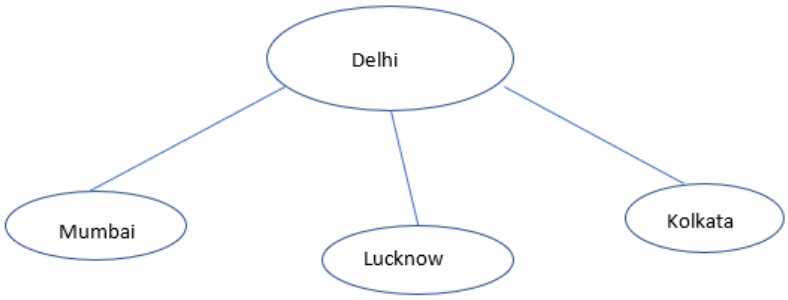


Fig 7.1.2

c) After expanding Mumbai

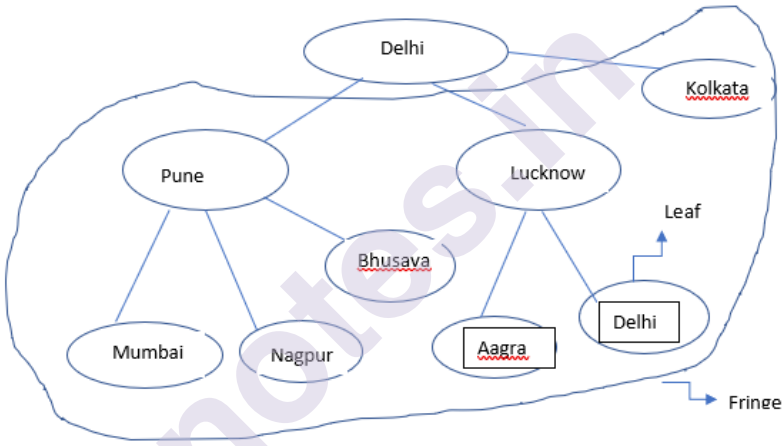


Fig 7.1.3 :state space search example

Search Tree

Consider the graph of state space graph shown in the figure.

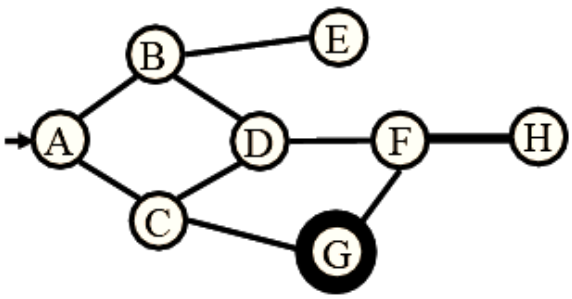


Fig 7.1.4: A State Space Graph

In the following graph shown all possible paths, by eliminating cycles from the paths, and we would get the complete search tree from a state space graph. In this figure A is a start node or root node and G is the goal

node. Every node may have a no child/children or more children. If a node X is a child of node Y, then node Y is said to be a parent of node X.

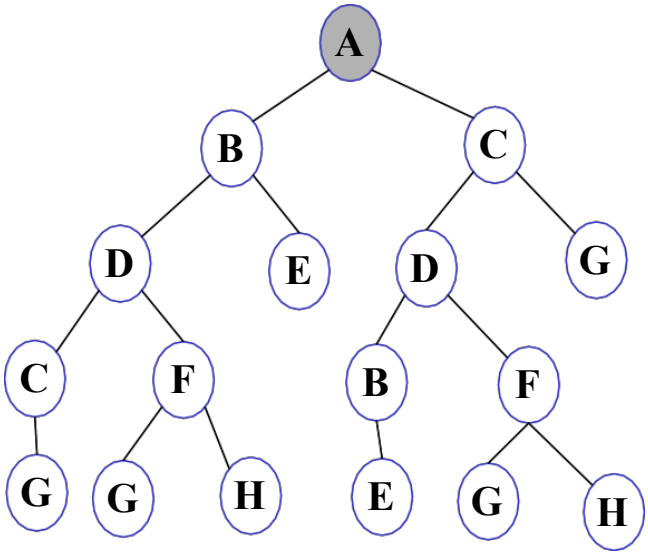


Fig 7.1.5 : Search graph converted into tree

Search Strategy:

It is a function that selects the next node to be expanded from current fringe. Strategy looks for best node for further expansion.

For finding best node each node needs to be examined. If fringe has many nodes then it would be computationally expensive.

The collection of un-expanded nodes (fringe) is implemented as queue, provided with the set of operations to be work with queue. These operations can be CREATE Queue, INSERT in Queue, DELELTE from queue and all necessary operations which can be used for general tree search algorithm.

Performance Evaluation

Completeness: Complete if n is finite i.e. algorithm provides an exact solution.

Optimality: Sometimes it happens that multiple solutions founds to a single problem. But algorithm try to give the best solution among the existed one.

Time Complexity: How much time to take an algorithm to find the solution.

Space Complexity: How much memory is required to perform the search algorithm. In a search tree where branching factor is b and maximum depth is m, then it requires storage (b^m) and can varies from algorithm to algorithm.

Major factors affect time and space complexity –

Time and space complexity are affected due to size of the State space search graph, which need to be stored as well as execute all the inputs to the algorithm. So, if state space graph is complex then more is the time and space required.

7.2.2 Simple search:

In simple search which works on the basis of combination of more than one strategy to solve the problem. Simple problems may be solved by one or two strategies but for complex problems solutions are not easy to solve. So we need to solve such problems by using combination of different techniques, So we can find the best optimal solution. **Simple search Algorithm is :**

```
SimpleSearch()
Open <- (startnode)
while open is not empty
    Open <-open(n)
If GoalTest(n) =True
    Then return n
Else
    pick some node n from
open list
```

There are a few searching algorithms or techniques are there call uninformed or blind search and informed search.

- **Uninformed search:**
- Breath first search
- Depth first search
- Depth limited search
- Iterative Deeping depth first search
- Bidirectional search
- Uniform cost search
- **Informed search**
- Heuristic search

7.2.3 Depth First Search

Procedure: In depth first search, the search tree is expanded depth wise i.e expand the deepest node in the current unexplored node set(fringe). As the leaf node is reached the search backtrack the to the previous node. It is shown in the given figure 7.2.3.1, where visited node are explored. Explored nodes with no children in the fringe are removed from memory.

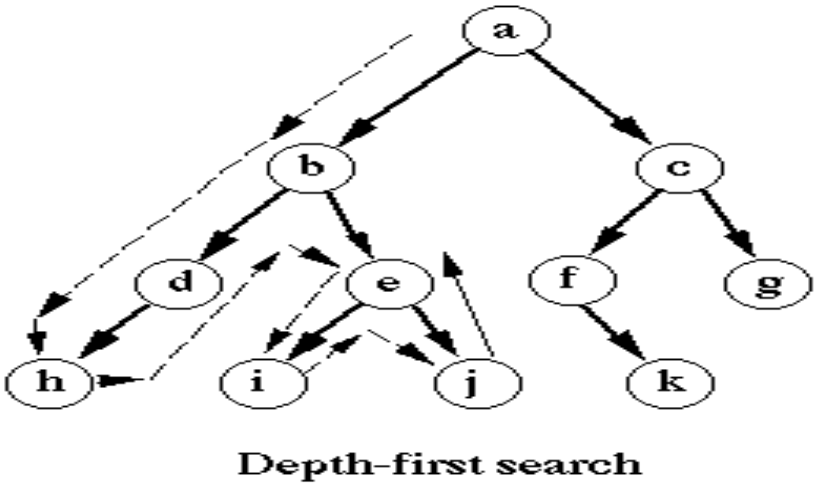


Fig 7.2.3.1 :DFS

Implementation: DFS can be implanted with last in first out (LIFO) data structure. The most recently generated node which is on the top in the fringe, is chosen first for expansion. As the node expanded, it is removed from the fringe and its successor are added. So when there are no more successor to add to the fringe the search "back tracks" to the next deepest node that is still explored can be implemented in two ways, recursive and non-recursive.

Non recursive implantation of DFS

Algorithm DFS
1. Put the root node on a stack
2. While (stack is not empty)
   a) pop a node from the stack
      i)If (node is a goal node) return success;
      ii) push all children of node onto the stack
3. return failure;

Recursive implantation of DFS

1. If node is goal node ,return success
2. For each child c of node
   i) If DFS (c) is successful
      Return success
3. Return failure

Performance Evaluation:

Complete:	• No: fails in infinite-depth spaces, spaces with loops. Modify to avoid repeated spaces along path • Yes: if m is finite spaces
Time complexity	• A depth first search may generate all of the O(bm) nodes in the search tree • Not good if **m** is much larger than **d** • But if the solutions are exists, this may be faster than breadth-first search
Space Complexity	• $O(b^m)$,m is maximum depth and b is the branching factor
Optimal	• No ,as there is no assurance the shallowest solution

7.2.4 Breadth First Search:

Procedure: In BFS a root node is expanded first, then all successor of the root nodes are expanded next fig 7.2.7.1 then their successors and so on shown in fig 7.2.3.2. It means all nodes are expanded from the tree root node and level by level to at the given depth in a search tree

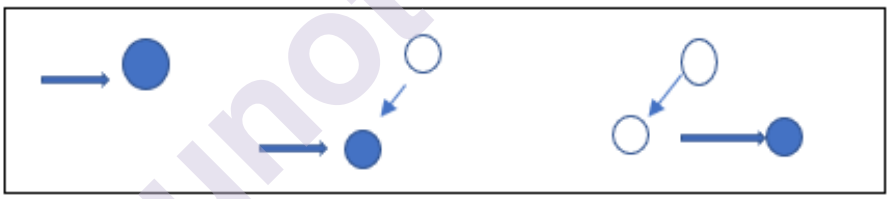
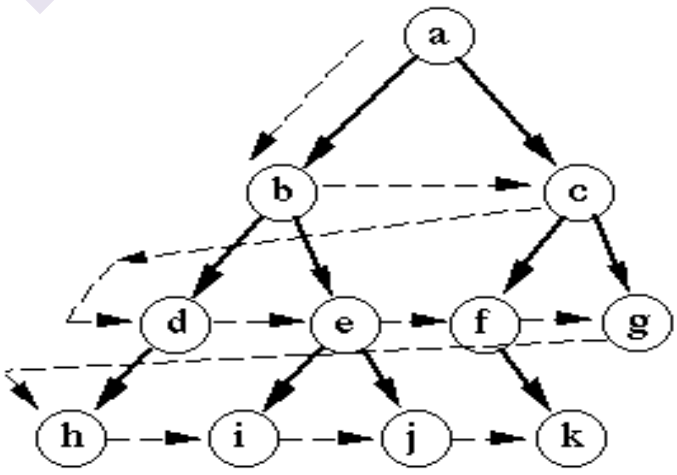


Fig 7.2.4.1 :Working of the BFS on binary search



Breadth-first search

Fig 7.2.4.2: BFS example

Implementation: BFS can be implemented using first in first out (FIFO)queue data structure. Fringe (unexpanded node) will be stored and processed As soon as node is visited it is added to queue. All newly generated nodes are added in the end of the queue which means that shallow nodes are expanded before deeper nodes.

Algorithm BFS

1. Put the root node on a queue
2. While (queue is not empty)
   b) Remove a node from the queue
      i)If (node is a goal node) return success;
      ii) put all children of node onto the queue;
3. return failure;

Performance Evaluation of BFS:

Complete Yes, if b (max branching factor) is finite

Time complexity $1 + b + b^2 + \ldots + b^d + b(b^d - 1) = O(b^{d+1})$ exponential in d

Space complexity
- $O(b^{d+1})$
- Keeps every node in memory
- This is the big problem; an agent that generates nodes at 10 MB/sec will produce 860 MB in 24 hours

Optimal
- Yes (if cost is 1 per step); It is optimal as it finds the shallowest solution
- otherwise not optimal in general

7.1.5 Difference between BFS and DFS

Breath first search	Depth first search
BFS traverses the tree level wise i.e. each node near to root will be visited first. The nodes are expanded from left to right.	BFS traverses the tree depth wise i.e. nodes in a particular branch are visited till leaf node and then continues branch by branch from left to right in the tree
BFS can implemented using queue which is first in first out (FIFO) list	DFS can implemented using stack which is last in first out (LIFO) list
BFS always provides the shallowest	DFS does not assurance to provides

path solution	the shallowest path solution
Backtracking is not required in BFS	Backtracking may require in DFS
BFS requires more memory compared to DFS	DFS requires less memory compared to BFS
BFS is optimal and complete if branching factor is finite	DFS is not optimal nor complete even in case branching factor is finite
BFS can never get stuck into infinite loop	DFS gets stuck into infinite loop, as search tree are dense
Applications of BFS • To find shortest path • In spanning tree • In connectivity	Applications of DFS • Useful in cycle detection • In connectivity testing • Finding a path between vertices and weight in the graph

7.3 HEURISTIC SEARCH

7.3.1 Heuristic Function:

A heuristic function is an evaluation function, to which the search state is given as input and it generates the real representation of the state as output.

It maps the problem state description to measure desirability, usually represented as number weights. The value of a heuristic function at a given node in the search process gives a good estimate of that node being on the desired path to solution.

Best first search is an example of the general Tree -Search or Graph-Search algorithm in which a node is selected for expansion based on the evaluation function *f(n)*. The evaluation function is built as a cost estimate, so the node with lowest evaluation is expanded first.

The choice of *f* determines the search strategy. In best first search algorithm includes as a component of *f* a heuristic function, denoted h*(n)*

h(n) = estimated cost of the cheapest path from the state at node n to goal state

- Heuristic function is main component of best first search. It is denoted by h(n)=g(n)
- h(n)=g(n) = shortest and cheapest path from initial state to goal state

- For example- If we want to travel from Delhi -Mumbai route then we can give information about distance between two cities using heuristic function
- A heuristic function *h(n)* consider a node as input but it depends on only on the state at that node
- h(n) =0 if n is goal state.

7.3.2 Best first search

In depth first search all branches are not getting expanded. And breath first search never reaches on dead end paths. If we combine these properties of both DFS and BFS it would be "succeeding" only a path at a time, but change paths whenever some path look more capable than the current on. This is what the best first search.

Best first search is a search algorithm which travers the search tree by expanding the best node selected according to the heuristic value of nodes f(n).

Efficient selection of the current best node, to explore node priority queue is used to implement it.

Search will start at root node.

The node to be explored next is chosen on the basis of an evaluation function f(n)

The state having lowest value for f(n) shows that goal is nearest from this state (i.e f(n) indicates distance from current state to goal state)

For example

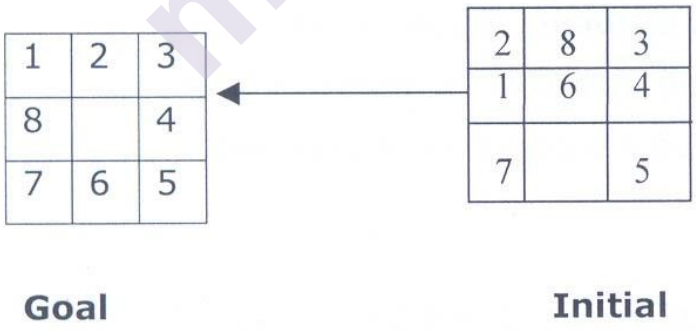


Fig 7.3.2.1. Heuristic function-8 puzzle

The first image shows the goal state , and the second image shows the initial state n. To find the goal state, **there is an appropriate sequence of moves are required, hence** *h(n) = 5* ,because the tiles 2, 8, 1, 6 and 7 are out of place.

Implementation: Best First Search can be implemented using priority queue where fringe will be sorted. The nodes in a fringe will be stored in priority queue with increasing value f(n) .i.e ascending order of f(n).The most importance will be given to node which has minimum f(n) value.

Best first search uses two ordered list to store the path. These are OPEN and CLOSED lists

OPEN list stores the nodes that have been generated but not visited yet.

ClOSED list stored the nodes that have been already visited. It check all nodes examine and should not repeat again, if visited node visited again but the different parent then it compare which existed value and the new value and the value which is minimum then it update the new parent.

Algorithm:

```
OPEN = [initial state]
CLOSED =0
     IF OPEN is empty then
{
     return failure
}
 n= heuristic best state
If   n== goal state then
{
     return path from start state to goal state
}
For each node is available
{
        if node not in OPEN and CLOSE list then
        {     add node to open list
              Set n as his parent node
        }
        Delete n from OPEN list
        Add n to CLOSED list
}
while (open list is not empty)
}
```


Performance Measure for Best First Search

Completeness	Not complete, if n is infinite
Time complexity	Worst case time complexity $O(b^m)$ where m is the maximum depth
Space complexity	$O(b^m)$ maintain a queue of all unexplored states.
Optimal	Not optimal, may not give optimal solution all the time

7.3.4 Local Search algorithm

The local search algorithm is a heuristic search algorithm. It helpful to find pure optimal solution to find the best state as per the essential objective function.

It operates using a single current node and generally move only to neighbours of that node. Generally, the path followed by the search are not reserved. Though local search algorithms are not planned.

They have two key advantages

1. They use very little memory - usually a constant amount and

2. They can often find practical solutions in large or infinite state space.

In which the aim is to find the best state according to an objective function. Many optimization problems do not fit the standard search model.

7.3.3 Hill Climbing search

Hill climbing search algorithm steepest ascent it is simply a loop that continually moves in the direction of increasing value that is uphill. Terminates when it reaches a peak there are no neighbours has a higher value. The algorithm does not maintain a search tree so the data structure for the current node need only record the state and the value of the objective function. Hill climbing does not look ahead beyond the immediate neighbours of the current state.

Hill climbing is sometimes call greedy local search because it grabs a good neighbour state without thinking ahead about where to go next. Although great grid is considered one of the seven deadly sins, it turns out that greedy algorithm often performed quite well. Hill climbing often makes rapid progress towards a solution because it is usually quite easy to improve a bad state. Unfortunately, hill climbing often gets stuck for local maxima,ridges,platue

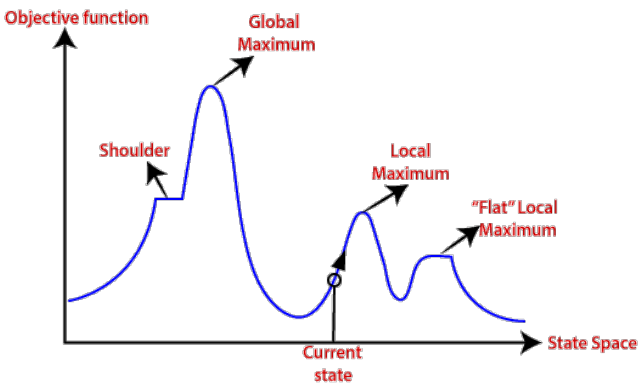


Fig 7.3.3.1 Hill climbing

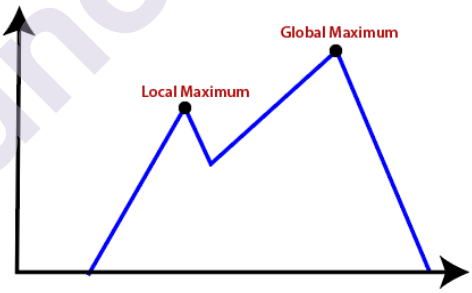
Simple Hill Climbing

Algorithm

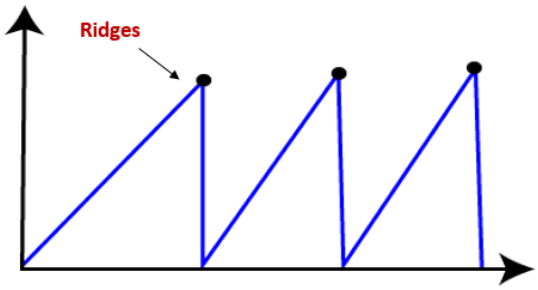
1. Evaluate the initial state.
2. Loop until a solution is found or there are no new operators left to be applied:
   - Select and apply a new operator
   - Evaluate the new state:
     goal → quit
     better than current state → new current state

Difficulties in hill climbing algorithm:

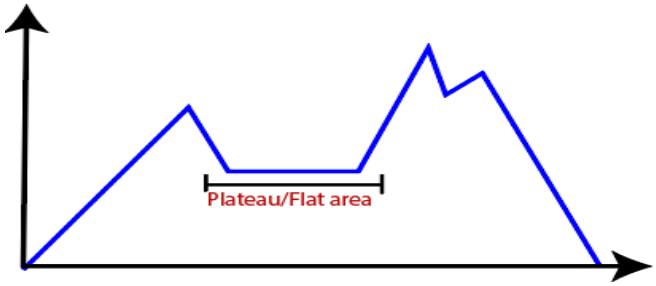
Local Maxima: local maximum is a peak that is higher than each of its neighbouring state but lower than the global maximum. Hill climbing algorithm that reach the vicinity of a local maximum will be drawn upward towards the peak but will then he stuck with nowhere else to go.



Ridges : Ridges is a sequence of local Maxima that is very difficult for greedy algorithm navigate. here all the values are same so it would be difficult to select the exact value that is the value which we give the optimal solution.



Plateau: batteries are flat area of the state space landscape. it can be a flat local maximum from which no uphill exit exists or a shoulder, from which progress is possible.



Example: 8 queen problems where each state has 8 Queens on the board one four column. The successor offer stayed are all possible states generated by moving a single Queen to another square in the same column (so each state has 8x7=56 successors). The heuristic cost function h is the number of pairs of queens that are attacking each other either OK perfect perfect solution or indirectly. The global minimum of this function is zero Which occurs at perfect solution. Figure (a) shows a state with is h= 17, the figure(b) also shows the value of all its successors it's the best random successors having h=12.

Hill climbing algorithms naturally choose randomly among the set of the best successor if there is more than one.

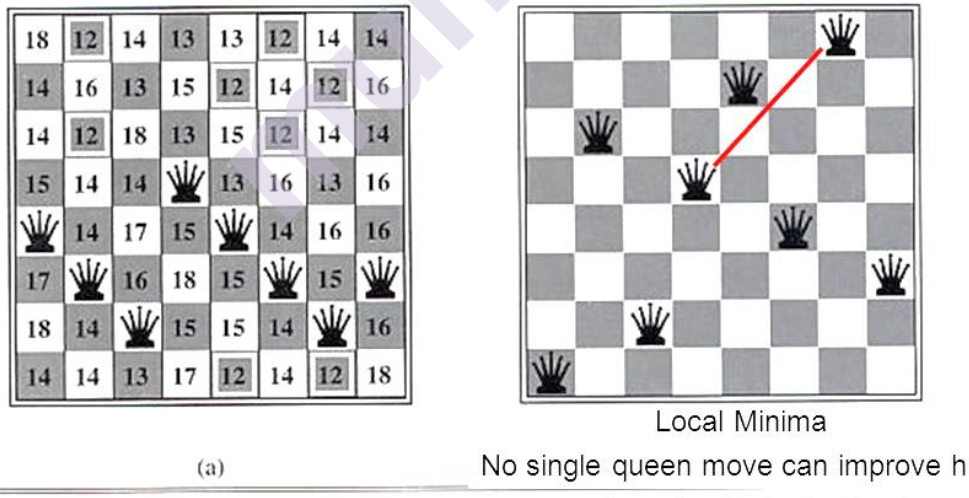


Figure 4.12 (a) An 8-queens state with heuristic cost estimate $h = 17$, showing the value of h for each possible successor obtained by moving a queen within its column. The best moves are marked. (b) A local minimum in the 8-queens state space; the state has $h = 1$ but every successor has a higher cost.

7.3.4 Local Maxima

Local maximum: .Local maxima is an example where the choice of a heuristic function determines where there are local maxima in the state space or not At a local maximum all neighbouring states have a values which is not better than the current state. Since hill-climbing uses a greedy approach, it will not move to the bad state and terminate itself.

The process will end even though a better solution may exist.To overcome local maximum problem Utilize backtracking technique. Maintain a list of visited states. If the search reaches an undesirable state, it can backtrack to the previous configuration and explore a new path.

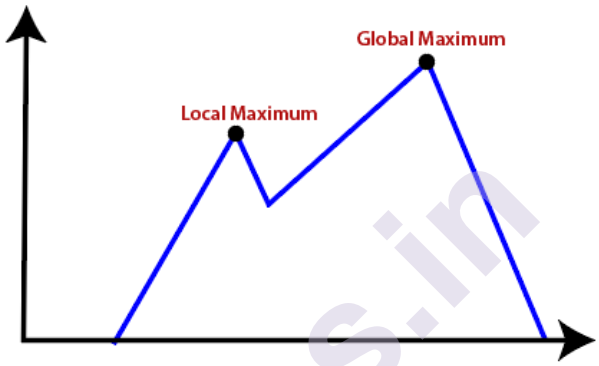


Fig 7.3.4 Local maxima

Simulated Annealing: Hill finding algorithm that never makes downhill most ordered states with lower value is guaranteed to be incomplete, because it can get stuck on a local maximum. in difference a purely random walk that is moving to a successor chosen consistently at random from the set of successes it's complete but extremely inefficient. So,it appears reasonable to try to combine hill climbing with a random walk in some way that is both efficiency and completeness. simulated annealing is such an algorithm. In metallurgy annealing is the process used to temper or harden a metals and glass by heating them to a high temperature and then gradually pulling them does allowing the material to reach low energy crystalline state.

- Simulated annealing search
- In simulated annealing initially the whole space is explored.

This is radiation of hill climbing

- Avoid the danger of being caught on a plateau or ridge make the producer list sensitive the starting point
- Simulated annealing is done to include a general survey off the scene to avoid climbing untruthful base hills.

There are two extra changes

1. rather than creating maxima, minimizations is done
2. The term objective function is used rather than heuristic

It becomes clear that this algorithm we have Valley descending rather than Hill climb. The property that the metal will jump through a higher energy level is given by P=exp 8-E/KT where K is Boltzmann's constant

Algorithm

1. Start with evaluating the initial state
2. apply each operator and loop until a goal state is found or till no new operators left to be applied as described below:-
3. I set T according to an annealing schedule you better the new state
4. if it is a goal state quit
5. ΔE = Val (current state)- Val (new state)
6. If $\Delta E<0$ then this is a current state.
7. Else find a new current state with probability $e^{\Delta E/T}$

7.3.5 Local beam search

Keeping just one state in memory might seem to be an extreme reaction so the problem of memory limitations. The local beam search algorithm keeps track of K state rather than just one stop it begins with a K random randomly generated state. At each step all the successors are all K states are generated. If anyone is a goal then algorithm halts. Otherwise select the K best successor from the complete list and repeats.

At first look K states

At first sight a local beam search with K states might seems to be nothing more than random K restart in parallel of in sequence.

By a local being search with K start state in random restart search every single search activity that's independent of other.

To implement local search threads are used. The K parallel search threads carry useful information

Anything in terms the states that generate the best successor say to the others come over here the grass is cleaner. The algorithm quickly terminates unfruitful branches exploration and moves its resources who the path seems most promising.

Limitations of local beam search

local beam search has limitations of lack of variation among the K states.

If the state concentrate on small area of state space then search becomes more expensive

7.3.6 Tabu Search:

Tabu search is an iterative search that starts from some initial feasible solution and attempts to determine the best solution in the manner of a hill-climbing algorithm.

That is search follows the diktat of the heuristic function as long as better choices are presented. But when there are no better choices, instead of terminating the local search as seen so far, it gives in to its explorative tendency to continue searching. Having got off an optimum, the algorithm should not return to it, because that is what the heuristic function would suggest. *Tabu* search modifies the termination criteria. The algorithm does not terminate on reaching a maximum, but continues searching beyond until some other criterion is met. One way to getting the most out of the search would be to keep track of the best solution found. This would be fairly straightforward while searching the solution space.

Tabu search is basically guided by the heuristic function. As a importance, even if it were to go beyond a local maximum, the heuristic function would tend to pull it back to the maxima. One way to drive the search away from the maxima is to keep a finite list in which the most recent nodes are stored. Such a list could be implemented as a circular queue of k elements, in which only the last k nodes are stored.

In a solution space search where the moves alter components of a solution, one could also keep track of which moves were used in the recent past. That is, the solution component that was disturbed recently cannot be changed. One way to implement this would be to maintain a memory vector with an entry for each component counting down the waiting period for changing the component.

In that sense, a local search does not necessarily evaluate all neighbourhood solutions. Generally, a subset of solutions is evaluated. If the optimal score is unknown, it must be told when to stop looking.

```
Algorithm Tabu Search (S0, var S, max_m, max_it)
 Set S = S0 and n_iter = 0
 Repeat
            m = 0
            best = 0
            it = it + 1
                  Repeat
                              m = m+1
                              Execute Check_the_neighboring_solution (S,Sm)

                              Execute Check_Tabu_list (S,Sn)
                                          if (f(Sm) > best then (best = f(Sm) and (m2= m))
                  until (m = max_m)

          Execute Add_to_Tabu_list (S,Sm2)

          S= Sm2

          If best > solution then solution = best
 Until n_iter = max_it
```

Tabu search Algorithm

7.4 FINDING OPTIMUM PATHS

7.4.1 Brute Force search

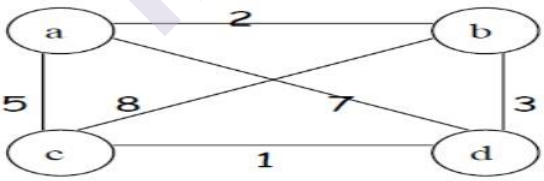
Brute force is a straightforward approach to problem solving, usually directly based on the problem's statement and definitions of the concepts involved. Though rarely a source of clever or efficient algorithms, the brute-force approach should not be overlooked as an important algorithm design strategy. Unlike some of the other strategies, brute force is applicable to a very wide variety of problems.

For some important problems (e.g., sorting, searching, string matching), the brute-force approach yields reasonable algorithms of at least some practical value with no limitation on instance size Even if too inefficient in general, a brute-force algorithm can still be useful for solving small-size instances of a problem.

A brute-force algorithm can serve an important theoretical or educational purpose. Sorting Problem Brute force approach to sorting Problem: Given a list of n orderable items (e.g., numbers, characters from some alphabet, character strings), rearrange them in nondecreasing order. Selection Sort

ALGORITHM

Selection Sort(A[0..n - 1])
//The algorithm sorts a given array by selection sort
//Input: An array A[0..n - 1] of orderable elements
//Output: Array A[0..n - 1] sorted in ascending order
for i 0 <-- 1 to n - 2 do
min <-- i
for j <-- i + 1 to n - 1 do
if A [j] < A[min]
Min <-- j



Tour	Length	
a → b → c → d → a	l = 2 + 8 + 1 + 7 = 18	
a → b → d → c → a	l = 2 + 3 + 1 + 5 = 11	optimal
a → c → b → d → a	l = 5 + 8 + 3 + 7 = 23	
a → c → d → b → a	l = 5 + 1 + 3 + 2 = 11	optimal
a → d → c → b → a	l = 7 + 3 + 8 + 5 = 23	

Swap A[I] and A[min]

7.4.2 Branch & Bound Algorithm

Branch and bound

The branch-and-bound (B&B) basis is a fundamental and widely-used procedure for producing exact solutions to NP-hard optimization problems. The technique, which was first proposed by Land and Doig, is often referred to as an algorithm; however, it is perhaps more appropriate to say that B&B encapsulates a family of algorithms that all share a common core solution procedure.

This procedure implicitly enumerates all possible solutions to the problem under consideration, by storing partial solutions called subproblems in a tree structure. Unexpanded nodes in the tree generate children by dividing the solution space into smaller regions that can be solved recursively (i.e., branching), and rules are used to prune off regions of the search space that are provably suboptimal (i.e., bounding). Once the entire tree has been explored, the best solution found in the search is returned. An early overview of the core B&B algorithm was provided.

```
1 Set L = {X} and initialize x̂
2 while L ≠ ∅:
3     Select a subproblem S from L to explore
4     if a solution x̂′ ∈ {x ∈ S | f(x) < f(x̂)} can be found: Set x̂ = x̂′
5     if S cannot be pruned:
6         Partition S into S_1, S_2, ..., S_r
7         Insert S_1, S_2, ..., S_r into L
8     Remove S from L
9 Return x̂
```

Branch and Bound Algorithm

Advantages

- In a branch and bound algorithm, we don't explore all the nodes in the tree. That's why the time complexity of the branch and bound algorithm is less when compared with other algorithms.
- If the problem is not large and if we can do the branching in a reasonable amount of time, it finds an optimal solution for a given problem.
- The branch and bound algorithm find a minimal path to reach the optimal solution for a given problem. It doesn't repeat nodes while exploring the tree.

Disadvantages

- The branch and bound algorithm are time-consuming. Depending on the size of the given problem, the number of nodes in the tree can be too large in the worst case.
- Parallelization is extremely difficult in the branch and bound algorithm.

7.4.4 Dijkstra's Algorithm

Dijkastra's algorithm is a graph search algorithm that solves the single-source optimal path problem for a graph with nonnegative edge path costs, producing an optimal shortest path tree.

This algorithm is often used in routing and as subroutine in other graph algorithms. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the optimal path to the destination vertex has been determined.

Traffic information systems use Dijkstra's algorithm in order to track the source and destinations from a given particular source and destination. The computation is based on Dijkstra's algorithm which is used to calculate the shortest path tree inside each area of the network.

Dijkstra's labelling method is a central procedure in shortest path algorithms. An out-tree is a tree originating from the source node to other nodes. The output of the labelling method is an out-tree from a source node s, to a set of nodes.

Three parts of information are required for each node i in the labelling method while constructing the shortest path tree

1. the distance label(i)

2. the parent-node/predecessor p(i),

3. the set of permanently labelled nodes

Where d(i) keeps an upper bound on the optimal shortest path distance from s to i; p(i) records the node that immediately precedes node i in the out-tree.

By iteratively adding a temporarily labelled node with the smallest distance label d(i) to the set of permanently labelled nodes, Dijkastra guarantees optimality. The algorithm can be terminated when the destination node is permanently labelled.

The major drawback of the algorithm is the fact that it does a blind search there by overwhelming a lot of time waste of necessary resources. Another disadvantage is that it cannot handle negative edges. This leads to acyclic graphs and most often cannot obtain the right shortest path.

Algorithm :

1. Set distance to start Node to zero.
2. Set all other distances to an infinite value.
3. We add the start Node to the non visited nodes set.
4. While the non visited nodes set is not empty.

5. Choose an evaluation node from the non visited nodes set, the evaluation node should be the one with the lowest distance from the source.

6. Calculate new distances to direct neighbours by keeping the lowest distance at each evaluation.

7. Add neighbours that are not yet visited to the non visited nodes set.

7.4.4 Refine search

Refine starts with the set of all node solutions represented by the root node. Each new node represent a subset of a solution. The each node can be seen as a partial solution, In which a part of the solution is specified. In each round, B&B selects and improved one node representing a partial solution, by specifying some more detail. This results in partitioning the set of solutions in that node. The candidate selected for refinement is the one that appears to have the lowest overall cost.

Assuming that there is a method for estimating the cost of a given (partial) solution, B&B refines the solution that has !he least estimated cost. The process of refinement continues until we have a complete solution at hand, and when no other candidate (partial) solution has a lower estimated cost. We need to ensure that the cost estimates are such that the algorithm guarantees an optimal solution. Ensuring that the estimated cost is a lower bound on the actual cost can accomplish this. That Is, a (complete) solution will never be cheaper than it is estimated to be. Then, candidates with estimates higher than that of some fully refined solution can be safely Ignored. It Is also desirable that the estimate be as high and as close to the actual cost as possible. because that will mean faster pruning of expensive candidates.

Consider Travelling sale person problem, initially we consider whole map because nothing is specifies. So solutions are permutation of all the cities',. We refined this solutions by considering the parameters we neede. Every time lower bounde estimated cost not feasible practically. So add all the known paths and so we can get increasing and better estimate cost or consider the smallest path. As we refine these partial paths so we get the better estimated cost and once we get the value above or more than the known cost of the complete solution then stop the refinement.

7.4.5 A * Algorithm

The most broadly known form of based first search he's called a start search. It evaluates nodes by combining g(n) look up the cost to reach the node and h(n), but cost to get from the node to the goal

$f(n)=g(n)+h(n)$

Since g(n) gives the path from the start node to node n, and h(n) is the estimated cost of the cheapest path from n to the goal.

$f(n)$ = estimated cost of the cheapest solution through n.

If we're trying to find the cheapest solution a practical thing to try first in the node with the lowest value of *g(n) + h(n)*. It turns out that this strategy is more than just reasonable: provided that the heuristic function h(n) satisfies certain conditions, A* search is both complete an opinion. the algorithm is identical to uniform cost search except that A* uses g + h instead of g.

As we observe the difference between A* and best first search is that the best first search for search only the heuristic estimated of h(n) is considered while A* counts for both distances traveled till a particular node and the estimation of distance need to travel more to reach to the goal node, always finds the cheapest solution.

Algorithm A*

1. The algorithm maintains who sets.

   Open list : open list keeps track of those nodes that need to be examined

   Closed list : closed list keeps track of nodes that have already been examined

2. Initially the open list contains just the initial number and the close list is the is empty each. Each node and maintained the following: g(n), h(n) ,f(n) as described above

3. Each node also kept record of pointer to its parent, so that later the best solution, if found can be retrieved. A* has a main loop that repeatedly gets the node, call it n with the lowest f(n) value from the OPEN list. If n is the goal node then we are done and the solution is given by backtracking from 'n'. Otherwise 'n' is removed from the open list and added to the CLOSED list. Next all the possible successor nodes of 'n' generate.

4. For each successor node 'n', if it is ready in the CLOSED list and the duplicate there has an equal or lower 'f' estimated and then we can safely remove the newly generated 'n' and move on. Similarly, if any is already in the open least and the copy over the word estimate we can discard the newly generated n and then move on.

5. If no better version of n exist on either the CLOSED or OPEN list we remove the lesser duplicates from the two lists and set n as parent of n. we also have to calculate the cost estimated for an as follows

   Set g(n) which is g(n) plus the cost of getting from n to n

   Set h(n) is the heuristic estimate of getting from n to the goal node

   Set f(n) is g(n) + h(n)

6. Lastly, add n open list and return to the beginning of the main loop.

Underestimated & overestimated heuristic function

The success of A* totally depends upon the design of heuristic function and how well it is able to evaluate each node by estimative estimating its distance from the goal node. Let us understand the effect of heuristic function on the execution of the algorithm and how the optimality gets affected by it

Underestimation: The heuristic function ***h*** never overestimates actual value from the current to goal that is the value generated by ***h*** is always lesser than the actual cost of the actual number of hopes required to reach through the whole state. In this case, A* within his guaranteed to find optimal path to a goal if one exists.

Overestimation: The value generated for each node is greater than the actual number of steps required to reach to the goal node.

7.4.6 Admissibility of A*

Admissibility of A* : A search algorithm is admissible for any graph it always terminates in an optimal path from start state to goal state if path exists. A heuristic is admissible if it is never overestimated the actual cost from the current state to the whole state. Otherwise, we can say that A* always terminate with the optimal path in case h(n) is an admissible heuristic function

A heuristic h(n)is admissible if for every node if $h(n) \leq h^*(n)$, where h*(n) is the true cost to reach the gold state from n. an admissible heuristic never overestimated the cost to reach the goal. Admissible heuristic is by nature optimistic 'because they think the cost of solving problem is less than it actually is.

An obvious example of an admissible heuristic is the straight line distance for stop straight line distance is admissible because the shortest path between any two points is a straight line so the straight line cannot overestimate the actual distance

Theorem : if h(n) is admissible , tree search using a study optimal

Proof : optimality of Easter admissible history

has been generated and is in the fridge. Let N be an unexpanded node in the fridge such that n is on the shortest path to an optimal goal g.

f (G2) = g(G2) since h(G2) = 0
g (G2) > g (G) since G2 is suboptimal
f (G) = g(G) since h(G)=0
h (n) ≤ h*(n) since h is admissible
g (n) + h(n) ≤ g(n) +h*(n)
f(n) ≤ g(G)
Hence f(G2) > f(n) , and A* will be never select G2 for expression.

Monotocity : If we put extra requirement on each h(n) which is consistency also called monotonous then we are sure it expects the optimal solution. A heuristic function h(n) is said to be consistent, if for every node n and every successor 'ns' of n generated by action 'a' the estimated cost of reaching the goal from an then the state cost of getting to 'ns.

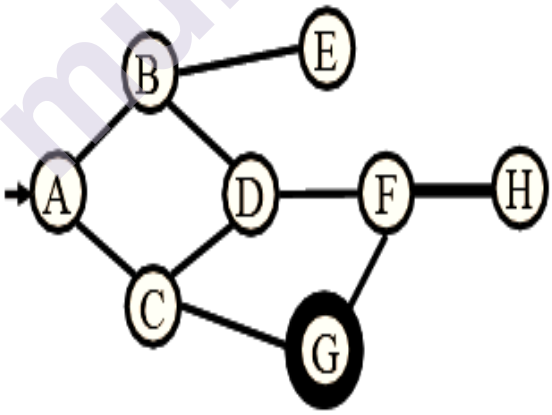
h(n)<= cost (n,a, ns)'+h (ns)

Properties of A*

1. **Completeness:** it is complete, as it will always find solution if one exist
2. **Optimality:** yes it is optimal
3. **Time complexity:** $O(b^m)$ as the number of nodes grows exponentially with solution cost
4. **space complexity**: $O(b^m)$ as it keeps all nodes in memory.

Solved Problem

1.Consider the tree and its search space in the shown in the figure, show how breadth first search works on this graph.



Step 1: Convert the given graph into search tree.

Initially fringe contains only one node corresponding to the source state A.

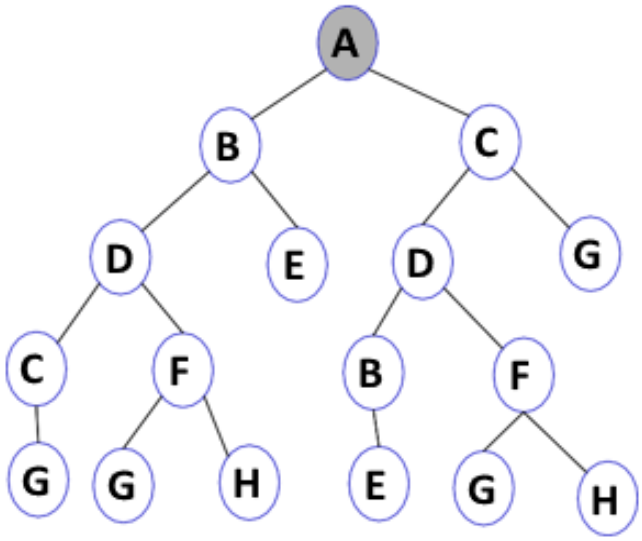


Fig:1 :search tree

FRINGE: A

Step 2: A is removed from fringe. The node is expanded, and its children B and C are generated. They are placed at the back of fringe.

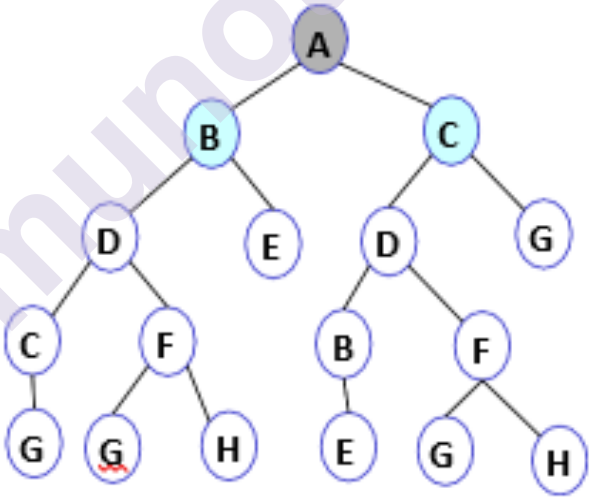


Fig: 2

FRINGE: B C

Step 3: Node B is removed from fringe and is expanded. Its children D, E are generated and put at the back of fringe.

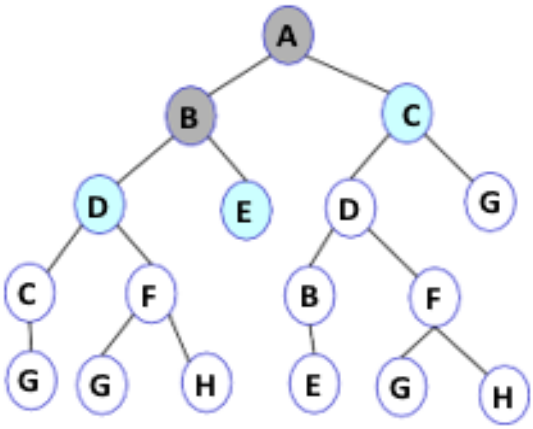


Fig: 3

FRINGE: C D E

Step 4: Node C is removed from fringe and is expanded. Its children D and G are added to the back of fringe.

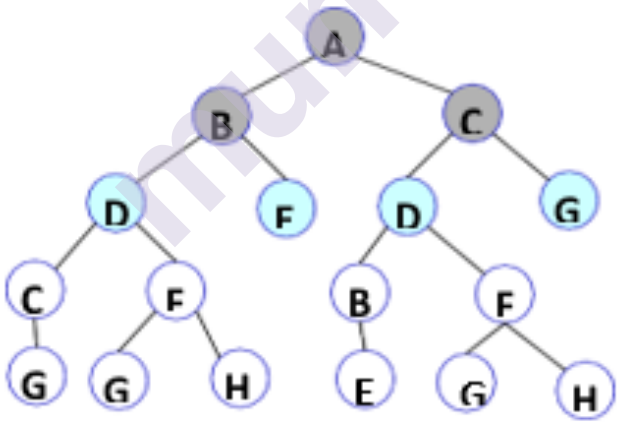


Fig: 4.

FRINGE: D E D G

Step 5: Removed node D from the fringe. Its children C and F are generated and added to the back of fringe.

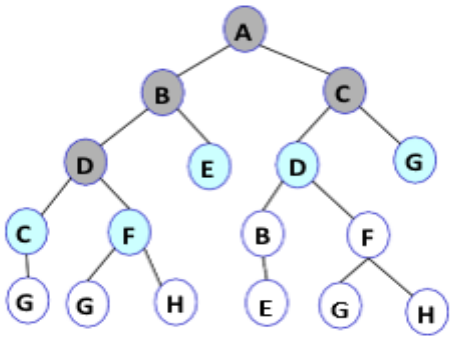


Fig: 5

FRINGE: E D G C F

Step 6: Node E is removed from fringe. It has no children.

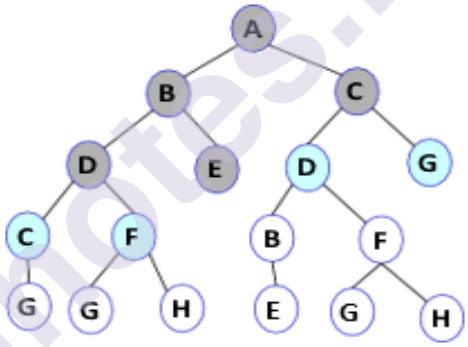
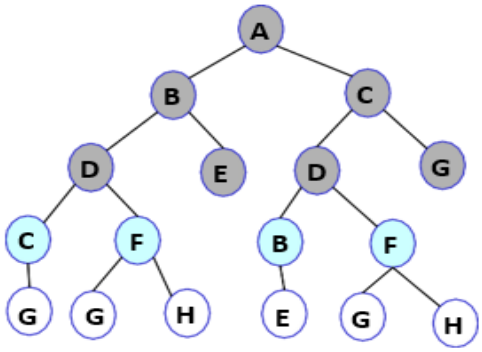


Fig.6

FRINGE: D G C F

Step 7: Node D is expanded; B and F put in queue.

are



Goal!

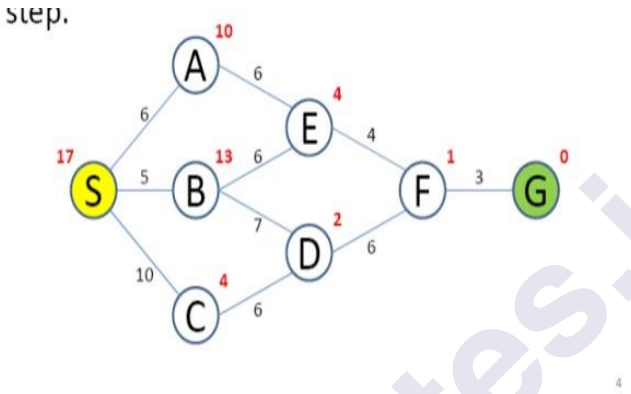
Fig:7

FRINGE: G C F B F

Step 8: G is selected for expansion. It is found to be a goal node. So the algorithm returns the path A C G by following the parent pointers of the node corresponding to G. The algorithm terminates.

2.Apply A* algorithm on the graph.Heuristic values are given alongwith the nodes.

S is the start node and G is the goal node.



Consider node S

G(n) =current node and distance of S g(n)=0, h(n)=17

. S= g(n)+h(n) = 0+17 =17

Step I: **Expnad node S**

Node A,B,C are connnected with the S

S-> A = g(n)+h(n) =6+10=16

S-> B= g(n)+h(n) =5+13 =18

S-> C= g(n)+h(n) =10+4 =14

Step -II

Expand node C

(S->C)->D = g(n)+h(n)=(10+6)+2 =18

Expand node A

(S->A)->E= g(n)+h(n)=(6+6)+4=16

Expand node B

(S->B)->E= g(n)+h(n)=(5+6)+4=15

(S->B)->D= g(n)+h(n)=(5+7)+2=14

Step III:

Expand node D

(S->C->D)->F= g(n)+h(n)=(10+6+6)+3=25 ……Path 1

Expand node E

(S->B ->D)->F= g(n)+h(n)=(5+7+6)+3=21 ……. Path 2

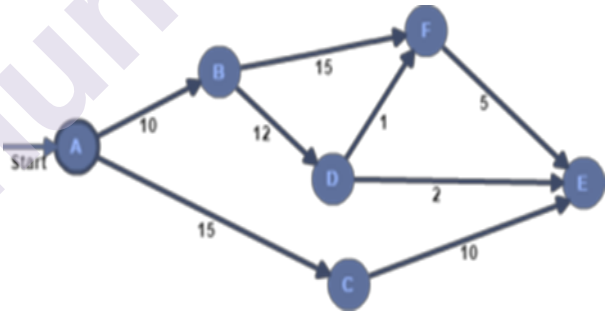
(S->A->E)->F= g(n)+h(n)=(6+6+4)+3=19 ……. Path 3

(S->B->E)->F = g(n)+h(n)=(5+6+4)+3=18 ……. Path 4

There are four paths available from source node to goal node S to G

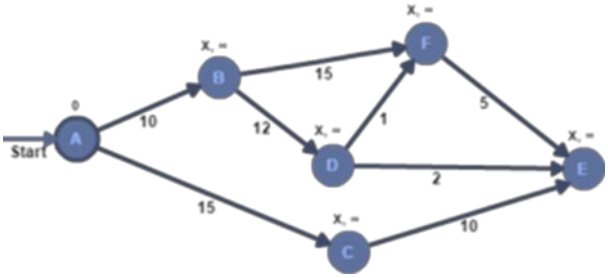
The best optimal path is S-B-E-F and value =18

3.Find the shortest path using Djkstra algorithm,source node is A and goal node is E



Step 1: Initialization

Before we start exploring all paths in the graph, we first need to initialize all nodes with an infinite (∞) distance and an unknown predecessor, except the source.



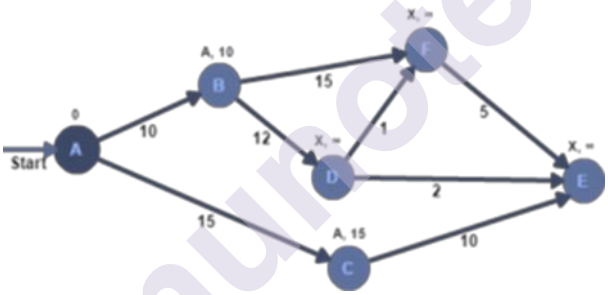
Step 2 : Evaluation

b. add the edge weight to the evaluation node distance, then compare it to the destination's distance. e.g. for node B, 0+10 is lower than ∞, so the new distance for node B is 10, and the new predecessor is A, the same applies to node C.

c. Node A is then moved from the unsettled nodes set to the settled nodes.

d. Nodes B and C are added to the unsettled nodes because they can be reached, but they need to be evaluated.

e. Now that we have two nodes in the unsettled set, we choose the one with the lowest distance (node B), then we reiterate until we settle all nodes in the graph:



Non visited Node	visited	Evalu-ation node	A	B	C	D	E	F
A	-	A	0	A-10	A-15	X-∞	X-∞	X-∞
B,C	A	B	0	A-10	A-15	B-22	X-∞	B-25
C,F,D	A,B	C	0	A-10	A-15	B-22	C-25	B-25
D,E,F	A,B,C	D	0	A-10	A-15	B-22	C-25	D-23
E,F	A,B,C,D	F	0	A-10	A-15	B-22	C-25	D-23
E	A,B,C,D,F	E	0	A-10	A-15	B-22	C-25	D-23

Finally, we can calculate the shortest paths from node A are as follows:

Node B : A –> B (total distance = 10)

Node C : A –> C (total distance = 15)

Node D : A –> B –> D (total distance = 22)

Node E : A –> B –> D –> E (total distance = 24)

Node F : A –> B –> D –> F (total distance = 23)

Review Questions

1. What is generate and test
2. Explain state space search.
3. Explain depth first search with example
4. Explain breath first search with example
5. Differentiate between BFS and DFS
6. Explain heuristic function with example
7. Explain Best first search
8. Explain hill climbing algorithm.
9. Explain local maxima in hill climbing in detail
10. Explain local beam search
11. Explain Tabu search
12. Explain brute force algorithm with example
13. Explain branch and bound algorithm with example
14. Explain Refine search
15. Explain Djkstra's algorithm with example
16. Explain A* algorithm. What is admissibility in A* algorithm.

References:

A First course in Artificial Intelligence, Deepak Khemani, Tata McGraw Hill Education (India) private limited (2013)

Artificial Intelligence: A Modern Approach, 3e, Stuart Jonathan Russell, Peter Norvig, Prentice Hall Publications (2010).

Artificial Intelligence Illuminated, Ben Coppin, Jones and Bartlett Publishers Inc (2004)

Artificial Intelligence A Systems Approach, M Tim Jones, Firewall media, New Delhi (2008)

Artificial Intelligence -Structures and Strategies for Complex Problem Solving., 4/e, George Lugar, Pearson Education (2002).

❄❄❄❄❄❄❄