

1501323 Robotics and Applications

Course Description:

Introduction to robotics technology and applications in automation system; Robot technology; Robot structure; Basic motion analysis and introduction to control and sensors; Robot programming; Artificial intelligence; Robot network design and control; Robotic and automation for industrial*; Automation simulation*; Automation verification*.

(*modified in the framework of an Erasmus + project: Asean Factori 4.0 Across South East Asian Nations: From Automation and Control Training to the Overall Roll-out of Industry 4.0 609854-EPP-1-2019-1-FR-EPPKA2-CBHE-JP)

Learning outcome:

1. Students can discuss the content of robotic and automation.
2. Students can analyze the behavior of robotic and automation system.
3. Students understand the function of industrial robotic and automation.

Lecturer:

Assoc. Prof. Punnarumol Temdee, Ph.D.
 Asst. Prof. Roungsan Chaisricharoen, Ph.D.
 Asst. Prof. Santichai Wicha, Ph.D.
 Lect. Chayapol Kamyod, Ph.D.

Credit: 3(3-0)

Lecture: 45 Hours (30 hours of modified content)

Assessments:

Attendance	10%
HW/CW	20%
Midterm	25%
Final	25%
Project	20%

Lecture (seminar):

Content	Hours
Robotic technology	3
Motion control	6
AI and programming for robotic and automation	6
Autonomous industrial process*	6
Automation and robotic components for industrial*	6
Robotic and automation for industrial*	6
Automation simulation*	6
Automation verification*	6

(*modified in the framework of an Erasmus + project: Asean Factori 4.0 Across South East Asian Nations: From Automation and Control Training to the Overall Roll-out of Industry 4.0 609854-EPP-1-2019-1-FR-EPPKA2-CBHE-JP)

1501323 Robotics and Applications



Program: Bachelor program in Computer Engineering

Credit: 3(3-0) Lecture: 45 Hours (30 hours of modified content)

1st Semester, Academic Year: 2024

Assoc. Prof. Punnarumol Temdee, Ph.D.

Asst. Prof. Roungsan Chaisricharoen, Ph.D.

Asst. Prof. Santichai Wicha, Ph.D.

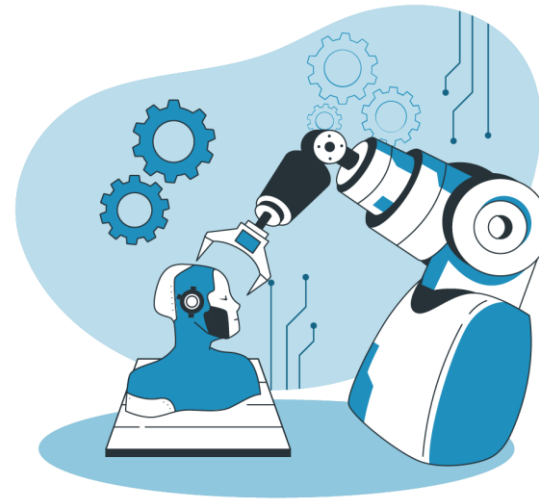
Lect. Chayapol Kamyod, Ph.D.



Co-funded by the
Erasmus+ Programme
of the European Union

This course has been modified in the framework of an Erasmus + project: Asean Factori 4.0 Across South East Asian Nations: From Automation and Control Training to the Overall Roll-out of Industry 4.0

609854-EPP-1-2019-1-FR-EPPKA2-CBHE-JP



Lecture 04: Autonomous Industrial Process

Objectives

- Develop a comprehensive understanding of the concepts and principles underlying Autonomous Industrial Processes, including levels of autonomy and the design of autonomous systems.
- Explore the relationship between automation and autonomy in the industrial context, and identify how automation can fuel and contribute to achieving higher levels of autonomy in industrial processes.
- Investigate the practical applications of Robotic Process Automation (RPA) across various industries, and understand where and why RPA is used to enhance efficiency and productivity.
- Examine the best practices and methodologies for implementing RPA solutions, with a focus on ensuring successful integration and maximizing the benefits of automation.
- Evaluate the benefits and limitations of RPA and autonomous industrial processes, and assess how these technologies can be leveraged to improve field operations, reduce human intervention, and enhance overall industrial efficiency.

Outlines

- Autonomous Industrial Process
- Achieving an Autonomous System
- Level of Autonomy
- The Autonomous Process Plant
- Design of Autonomous Industrial Process
- Refinement into Key Features
- Industrial Autonomy vs. Automation
- Automation Fuels Autonomy
- Field Operations Autonomy
- Pathways to Autonomy
- Onboard Industrial Autonomy
- Autonomy vs. Human Intervention
- What is Robotic Process Automation?
- Why Robotic Process Automation?
- Where do you use RPA?
- Example of RPA
- RPA in Different Fields
- Differences between Test Automation and RPA
- RPA Implementation Methodology
- Best Practices of RPA Implementation
- General Use of RPA
- Application of RPA
- Cases for RPA uses
- Robotic Process Automation tools
- Benefits of RPA
- Disadvantages of RPA
- Other Autonomous Industrial Process



Autonomous Industrial Process

- Autonomous industrial processes refer to the use of advanced technologies and automation systems to manage and control various aspects of industrial operations without human intervention or with minimal human oversight.
- These processes are often found in manufacturing, production, and other industrial settings, and they aim to improve efficiency, productivity, safety, and reliability.
- Autonomous industrial processes are continually evolving, driven by advancements in technology and the desire to improve efficiency, reduce costs, and enhance safety in industrial operations.



Autonomous Industrial Process (Cont.)

Here are some key components and concepts related to autonomous industrial processes:

1. **Automation and Control Systems:** Autonomous industrial processes rely on sophisticated automation and control systems that can monitor and adjust parameters in real-time. These systems can include programmable logic controllers (PLCs), industrial robots, sensors, actuators, and other hardware and software components.
2. **Data and Sensors:** Sensors are crucial for collecting data on various aspects of the industrial process. This can include temperature, pressure, humidity, flow rates, and more. The data collected is then used to make real-time decisions and adjustments.

Autonomous Industrial Process (Cont.)

3. **Internet Machine Learning and Artificial Intelligence:** Machine learning and AI technologies are often employed to make autonomous industrial processes more intelligent. These technologies can analyze data, predict failures, optimize processes, and adapt to changing conditions.
4. **Internet of Things (IoT):** IoT devices and connectivity play a significant role in autonomous industrial processes. They allow different components and devices to communicate and share data, enabling centralized control and monitoring.
5. **Autonomous Vehicles:** In industries such as logistics and warehousing, autonomous vehicles like self-driving forklifts and drones are used for material handling, reducing the need for human operators.

Autonomous Industrial Process (Cont.)

6. **Predictive Maintenance:** Predictive maintenance uses data analytics and AI to predict when equipment is likely to fail. This allows maintenance to be scheduled proactively, minimizing downtime and reducing costs.
7. **Process Optimization:** Autonomous systems can continuously optimize industrial processes to improve efficiency and reduce waste. This can involve adjusting parameters such as speed, temperature, and pressure to achieve desired outcomes.
8. **Safety:** Safety is a critical consideration in autonomous industrial processes. Redundant safety systems, emergency shutdown procedures, and fail-safe mechanisms are often in place to prevent accidents.
9. **Energy Efficiency:** Autonomous processes can be designed to optimize energy consumption, reducing costs and environmental impact.



Autonomous Industrial Process (Cont.)

10. **Remote Monitoring and Control:** In some cases, industrial processes can be monitored and controlled remotely, allowing for greater flexibility and reducing the need for on-site personnel.
11. **Regulatory Compliance:** Industries must adhere to regulations and standards. Autonomous systems can help ensure compliance by monitoring and recording data for auditing purposes.
12. **Human-Machine Collaboration:** While the aim is to reduce human intervention, there are still scenarios where human expertise is required. In these cases, humans and machines collaborate to make critical decisions.

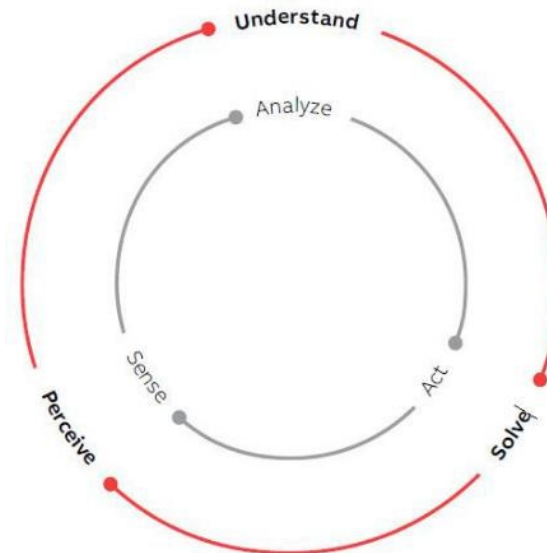


Achieving an Autonomous System

- An automation system typically performs precisely defined instructions within a limited scope of operation.
- For example, a motor is running too fast (sense), the controller decides to reduce the speed (analyze) and reduces the current to the motor (act).
- An autonomous system feedback loop adds another shell, applying the same principle but on a more complex level, dealing also with what is not known or foreseen.

Achieving an Autonomous System (Cont.)

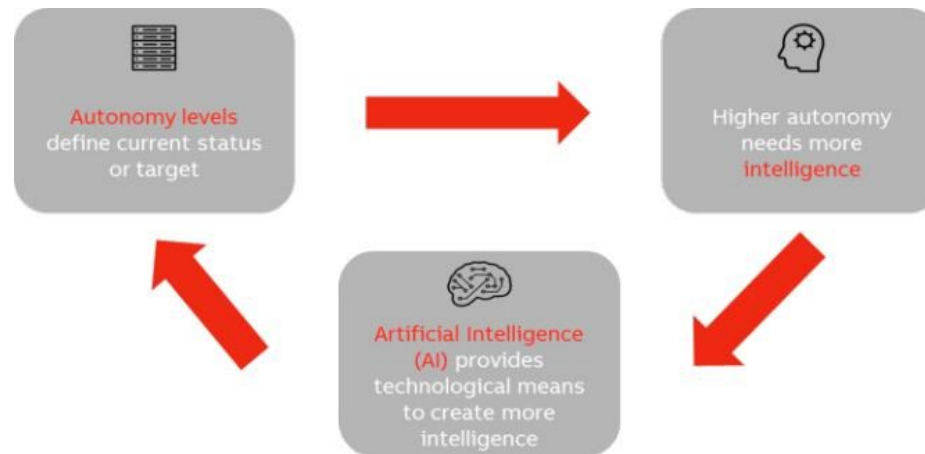
- A self-driving vehicle identifies an obstruction (perceive), recognizes that a potentially dangerous situation may arise (understand) and takes corrective action by modifying the speed and trajectory of the vehicle (solve).



Loops in classical control systems (grey) and autonomous systems (red)

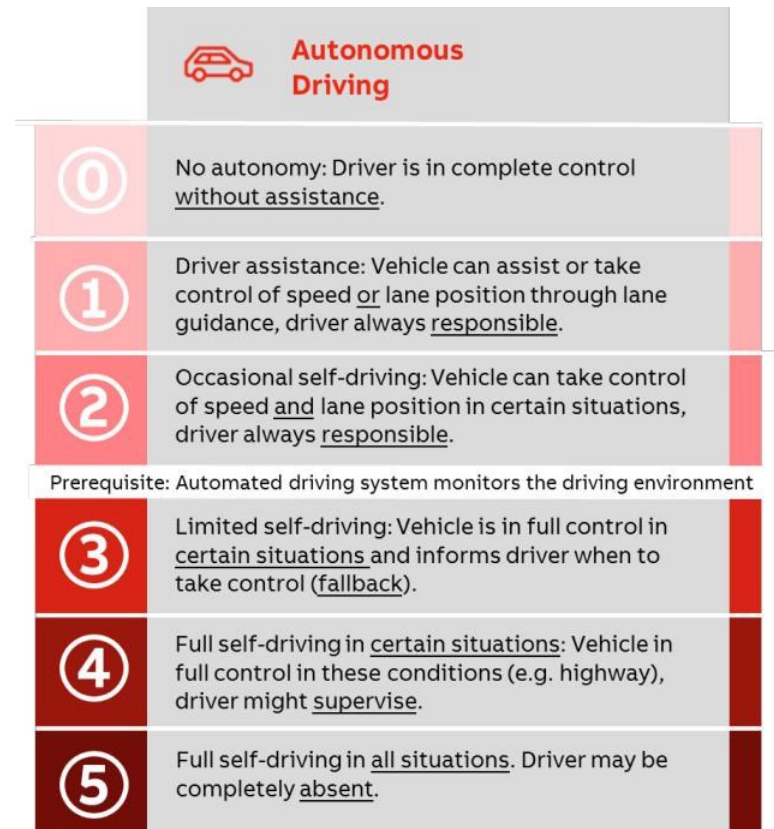
Achieving an Autonomous System (Cont.)

- There can be a tendency to confuse AI with autonomous systems.
- While AI indeed has a close relation to autonomous systems, it is a technological means through which a specific level of autonomy can be achieved.



Relation between autonomy and artificial intelligence

Level of Autonomy

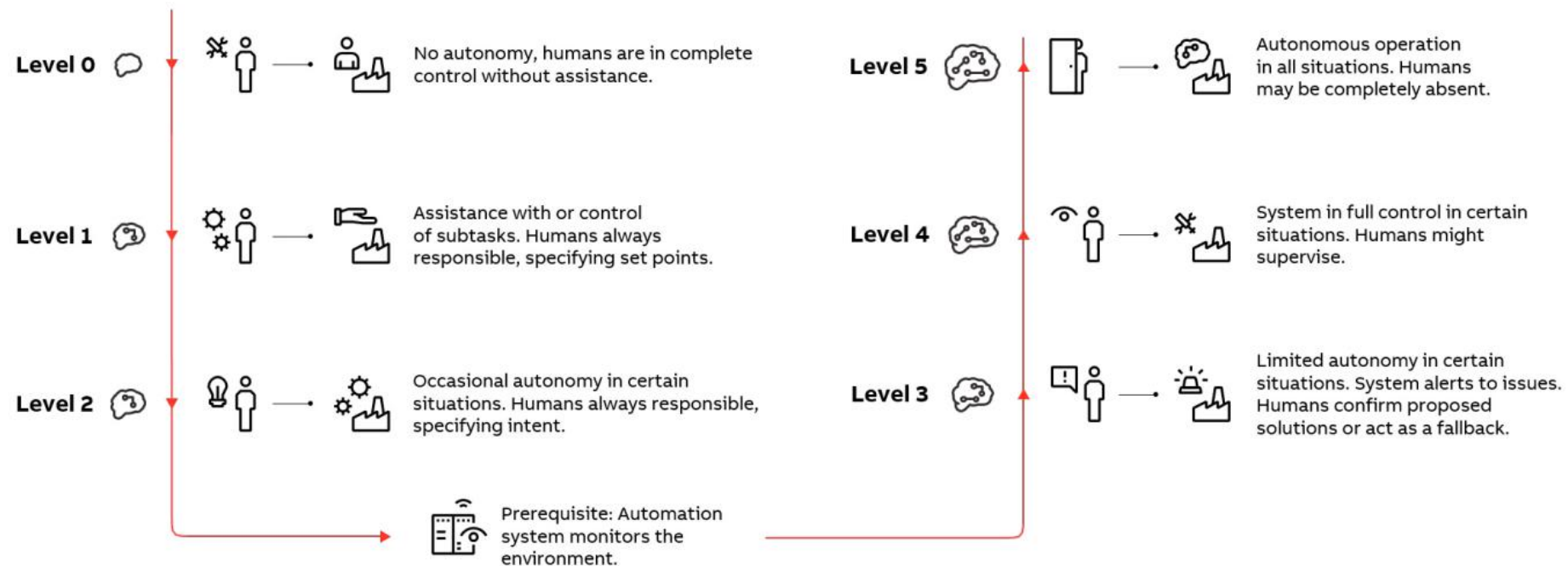


Summary of autonomy levels in automotive domain, based on (SAE, 2018)

Level of Autonomy (Cont.)

Autonomous systems

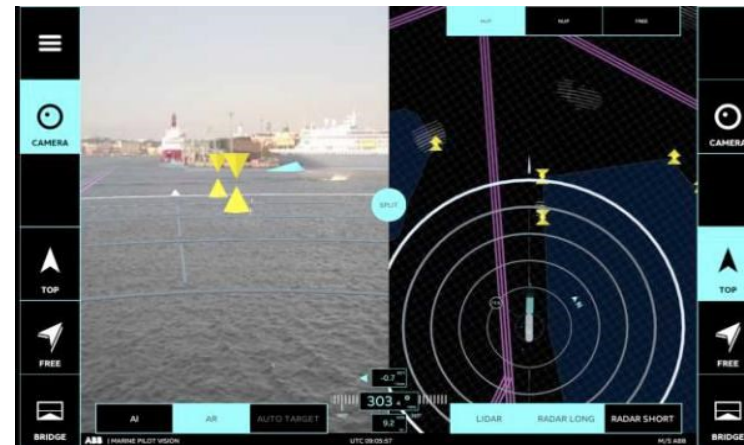
Top-level definition



Taxonomy of levels of autonomous systems

Level of Autonomy (Cont.)

- Examples include software that helps localize underground mine vehicles or provides situational awareness for ships by additional sensing such as lidar and radar.



- ABB Ability™ Marine Pilot Vision combining visual image as seen from the bridge as well as a 360-degree view of the ship based on additional sensing

The Autonomous Process Plant

While autonomy could be brought to any part and life-cycle phase of a plant, the primarily considered lifecycle phase in the current paper is operations and can be further split into three autonomous key features:

- **Field Operations / Maintenance:** Operations that are done in the process plant (in the field) itself.
- **Control Room Operations:** Operations that are today primarily done by operators, inside a control room using special operator interfaces (human machine interfaces) in order to understand the process and act according to the current needs of the plant.



The Autonomous Process Plant (Cont.)

The potential and cost to transform the above fields to an autonomous sub-system depends on the current condition of the plant in question.

- **Planning and Scheduling:** Planning and scheduling are tasks not performed by the operator but by dispatchers. They include the planning of the production order of different batches/products in a process plant, as well as the scheduling of the batches/products.

The Autonomous Process Plant (Cont.)

- From a software point of view, an autonomous plant requires a system that is capable of real-time supervision, that is able to react on unexpected events, to self-diagnose its components, and to report both its current state as well as a forecast to human supervision.
- All those requirements have significant impact on the automation architecture of an autonomous plant. The final result is still unclear and a highly disputed field of research.
- The following fields are commonly agreed as ongoing trends: Communication will often also involve 5G, time-sensitive networking (TSN), IP-based protocols, and etc.



Design of Autonomous Industrial Process

Designing and structuring an autonomous industrial process requires careful planning, integration of various technologies, and consideration of specific industrial needs:

- **Define Objectives and Requirements:** Clearly define the objectives of the autonomous industrial process.
- **Data Acquisition and Sensing:** Select and install sensors to collect data on critical variables such as temperature, pressure, flow rates, and product quality.
- **Data Transmission and Connectivity:** Establish a robust communication network to transmit data from sensors to a centralized control system.
- **Centralized Control System:** Develop or select a centralized control system that can process and analyze data in real-time.

Design of Autonomous Industrial Process (Cont.)

- **Machine Learning and AI Integration:** Integrate machine learning and AI algorithms to analyze and make decisions based on the collected data.
- **Automation Hardware:** Implement automation hardware such as industrial robots, autonomous vehicles, and robotic arms as needed for the process.
- **IoT and Edge Computing:** Use IoT devices and edge computing solutions to enable data processing and decision-making at the edge of the network.
- **Safety Systems:** Implement redundant safety systems and emergency shutdown procedures to prevent accidents and respond to failures.
- **Process Optimization and Adaptation:** Develop algorithms and logic for continuous process optimization, allowing the system to adjust parameters to achieve desired outcomes.

Design of Autonomous Industrial Process (Cont.)

- **User Interface and Monitoring:** Create a user-friendly interface for operators to monitor the process and intervene when necessary.
- **Predictive Maintenance:** Develop predictive maintenance algorithms that monitor the health of equipment and schedule maintenance proactively.
- **Energy Efficiency and Environmental Impact:** Integrate energy management systems to optimize energy consumption and reduce environmental impact.
- **Documentation and Training:** Maintain comprehensive documentation of the autonomous industrial process, including standard operating procedures, safety guidelines, and maintenance schedules.

Design of Autonomous Industrial Process (Cont.)

The design and structure of an autonomous industrial process should be a collaborative effort involving engineers, data scientists, automation experts, and domain-specific professionals to create a system that optimizes industrial operations while maintaining safety and compliance.


- **Testing and Validation:** Conduct thorough testing and validation of the autonomous industrial process to ensure it meets the defined objectives and requirements.
- **Scalability and Future-Proofing:** Design the system to be scalable, allowing for easy expansion or adaptation to changing needs.
- **Regulatory Compliance:** Ensure that the autonomous industrial process complies with relevant industry standards, regulations, and safety guidelines.

Refinement into Key Features

Comparison of autonomy in automotive and process automation domains

	Autonomous Car	Autonomous Plant/Factory
Lifecycle phase	Operation	Operations, possibly others too ¹
Autonomous key feature	Driving	Control room operations, field operation, process optimization, production scheduling
Human role(s)	Driver	Control room and field operator, process engineer, dispatcher, ...
# of tasks / set points	Low: speed, lane position	High: dozens to hundreds, including temperatures, flows, or pressures.
# of artifacts	Medium: ECUs, sensors, valves, ...	High: I/Os, controllers, valves, pumps, compressors, ...
Homogeneity	High: every driver can drive every car	Low: operators cannot (easily) switch between plants/factories

Refinement into Key Features (Cont.)

	 Autonomous Plant	Control Room Operation	Field Operation / Maintenance	Planning and Scheduling
0	No Autonomy: Humans carry out all necessary operations without assistance.	Humans carry out control of all assets without assistance. Low-level automation might be available (e.g., safety system).	All field operator tasks executed by humans.	Manual development of plans and the corresponding schedule.
1	Operations Assistance: Automation system provides decision support for necessary operations by remote / digital assistance. Humans always responsible.	Automation of control loops during steady-state. Manual startup and shutdown of the plant. Manual execution of transitions. Alarm based notification.	Automation system notifies humans about field activities. Some tasks are automated, e.g. operating valves.	ERP plan creation on human request. Human decides when and how to execute the plans and adapts plans.
2	Automation system is in control in certain situations on request (humans pull support, e.g. for plant startup). Humans always responsible.	Automation system assisted plant startup, transition, steady-state, and shutdown. Manual fault correction supported by decision support system.	System guided field operation tasks. Humans get instructions what to do and when by decision support system.	Adaptations of plans to current situations by operator request.
3	Automation system is in control in certain situations. Plant actively alerts to issues and proposes solutions. Humans confirm.	Automated plant shutdown, startup and transition, on human request. Automatic correction of known deviations. Decision support for unexpected/unknown faults.	Most tasks required for standard operations are automated, like shutdown, startup and transition phases. Number of humans in the field heavily reduced.	Continuous feedback and re-planning in case of production deviations. Manual schedule release.
4	Autonomous operations in certain situations: automation system has full control in these situations, humans supervise actions.	Autonomous control in certain situations with automatic fault and deviation correction and avoidance.	Almost human free field operation. Only human field operation in exceptional situations.	Continuous autonomous planning and scheduling without user interaction. Detection of production deviations and re-planning. Manual schedule release.
5	Fully autonomous operation in all situations. Humans may be completely absent.	Full autonomous control, fault correction and avoidance in all situations. No human supervision required.	Full autonomous field operation, no manual actions in the field necessary. No humans remain in the plant.	Autonomous development and execution of plans and schedules. Autonomous re-planning in case of production deviations. No human interaction.

Comparison of autonomy in automotive and process automation domains

Industrial Autonomy vs. Automation

- Industry and business sectors have been working toward autonomous systems for many years.
- Rapid advancement in software tools, robotics, discrete production, and autonomous vehicles in mining and offshore/subsea activities are just a few examples that prove sustainable business value and ROI.
- Automation performs a series of highly organized pre-programmed tasks but requires human intervention and control.



Industrial Autonomy vs. Automation (Cont.)

- As with continuous steady-state operations, the sequence of tasks could take minutes, hours, or even longer to complete.
- They are solely responsible for ensuring that the operation is completed safely, on time, and profitably.
- An operator, for example, might be in charge of safely starting up a unit or performing rudimentary switches or grade changes.
- Autonomy goes beyond automation to include layers of machine cognition and smart sensing to predict and adapt to contingencies, eliminating operator intervention.
- Autonomous systems control all aspects of the task from startup to shutdown, ensuring safety along the way.



Automation Fuels Autonomy

- **Industrial Automation** has dramatically benefited process industries and will continue to be beneficial even with industrial autonomy.
- Despite the slow pace of change, adoption promises to pick up rapidly as the advancement of enabling technologies, such as industrial software platforms, artificial intelligence (AI), machine learning, deep learning, cloud and edge computing, wireless sensors, and enhanced communications and networking become more widely available.

Field Operations Autonomy

- Field operations autonomy at Levels 0 through 4 implies moving from low-level autonomy to mid-level autonomy, where humans carry out operations.
- The system recognizes and guides operators on what to do and how to complete the task.
- To gain more autonomy, manual tasks must be fully automated, and human intervention only required where necessary.
- Level 5 field operation autonomy means that systems operate self-sufficiently, allowing field personnel to focus on other value-add tasks that optimize the field's long-term performance and profitability.

Pathways to Autonomy

- Process industries are now closer to autonomy than they have ever been.
- Many operating companies are trying to achieve fully uncrewed remote operations, as is the case for very complicated, remote, or dangerous facilities, such as mining operations, hydraulic fracturing jobs, and offshore gas production platforms.

Pathways to Autonomy (Cont.)

Reasons to prioritize autonomy include:

- Reducing operating costs while increasing production
- Improving sustainability by reducing harmful greenhouse gas emissions
- Making it safer and more attractive for people to operate onsite facilities
- Removing people from unsafe situations
- Enhancing management of dispersed assets with remote or scarce resources

Pathways to Autonomy (Cont.)

Most of the processes that manufacturers use to achieve remote operations apply to autonomy:

- Converting manually operated machinery to fully automated machinery
- Using procedural standardization to automate manual tasks like a machine or plant startup and shutdown
- Leveraging redundant and adaptable controls, communications, and key equipment
- Implementing smart sensors to enhance monitoring of both equipment and processes

Pathways to Autonomy (Cont.)

Most of the processes that manufacturers use to achieve remote operations apply to autonomy (Cont.):

- Conducting pilotless surveillance, remote monitoring, and inspection
- Performing regular robotic maintenance
- Using analytics, machine learning, and other AI techniques to enable predictive maintenance
- Using AI and other analytics to monitor processes for imminent abnormal situations continuously and initiate appropriate action
- Applying stringent modeling approaches to develop digital twins

Onboard Industrial Autonomy

- Companies can onboard industrial autonomy by building greenfield autonomous plants.
- Another option is to use autonomous technologies, like the KICS platform, in existing facilities to enable cross-functional data integration and supervisory process control.
- Autonomous technologies may take the shape of an AI system learning to regulate process variables by opening and closing valves or combining procedural automation and AI to execute more sophisticated tasks.
- Both higher-level functions and production processes can be made autonomous.
- Safety, margin optimization, performance, supply-chain management, compliance, and other manufacturing operations and tasks could benefit from autonomy beyond its typical controls and efficiency focus.



Autonomy vs. Human Intervention

- As autonomous capabilities rapidly expand, there are many unanswered questions regarding the role of the human.
- Making routine jobs autonomous frees workers to perform other tasks requiring more hands-on human involvement.
- It frees up workers to work on other projects that can drive greater efficiency and optimization in the plant/field.
- Humans are vital to making critical, complex decisions that require a rational thought process.

What is Robotic Process Automation?

- **Robotics Process Automation (RPA)** allows organizations to automate tasks just like a human being was doing them across applications and systems.
- RPA can be used to automate workflow, infrastructure, back office processes which are labor intensive. These software bots can interact with an in-house application, website, user portal, etc.
- People sometimes refer to Robotic Process Automation (RPA) as robotic software.



What is Robotic Process Automation? (Cont.)

- The RPA is software with **virtual robots** that humans can use to interact with various other software types and elements in the digital system infrastructures.
- Humans use the RPA robots to handle repetitive digital tasks.
- In particular, the ones that involve many rules and constraints are the tasks that become the specializations of RPA robots.
- RPA is an **easy-to-use software** (or tool) that human users can utilize to shorten significant time to do several digital tasks.



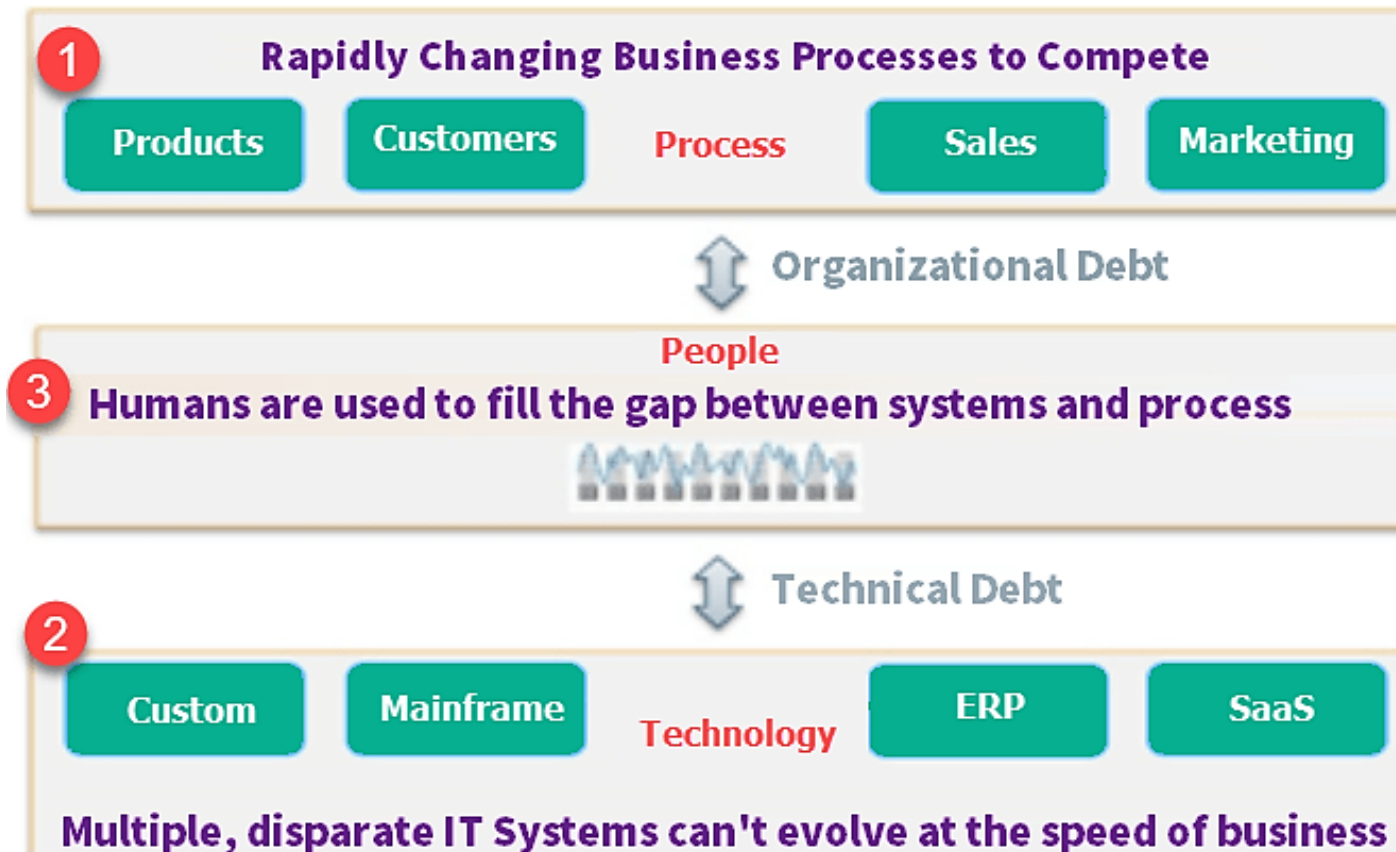
What is Robotic Process Automation? (Cont.)

Example of RPA

- Humans usually have to input the data in individual columns or rows one-by-one.
- The RPA helps in generating auto-fill data.
- The humans (users) will see various recommendations every time they type a specific alphabet or character in the field.



Why Robotic Process Automation?

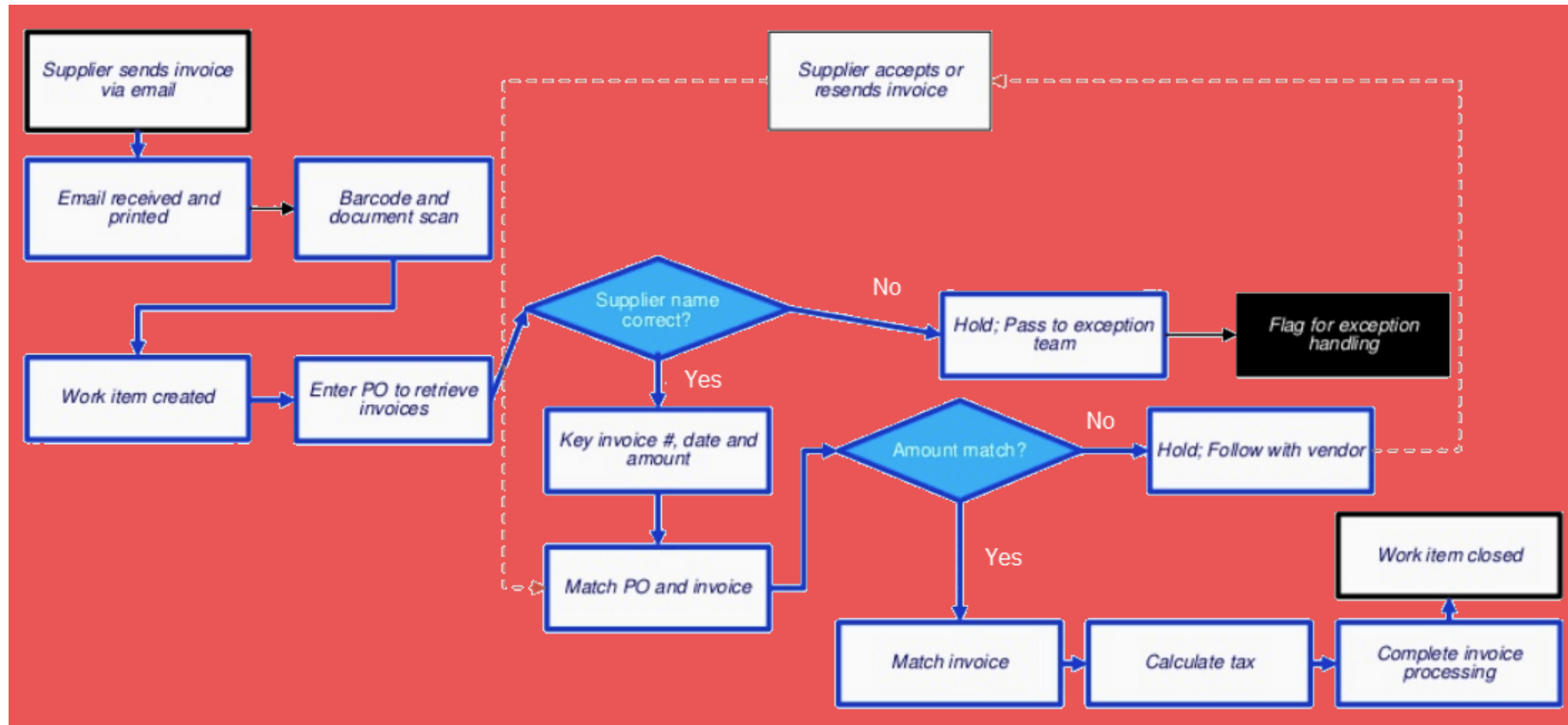




Where do you use RPA?

- **Automating the data correction** is one of the benefits of RPA.
- Sales orders and invoice processing are the two areas where the RPA process can shine.
- RPA also increases data consistency levels across the company department systems.

Example of RPA



Example of RPA

Description	Can be Automated via RPA?
Open invoice email from the supplier and print it for records	Yes
Barcode Scanning	Manual
Create work item in a legacy software system	Yes
Enter PO to retrieve Invoices	Yes
Check supplier name is correct or not?	Yes
Key Invoice, Data and Amount	Yes
Check if Amount is matches or not?	Yes
If amount match is yes the Matched Invoice, Calculate Tax	Yes
Complete Invoice Processing	Yes
Work Item Closed	Yes

RPA in Different Fields

The following industries are the industries that adopt the RPA process the most compared to the others:

- **Banking, Finance, and Insurance:** There are massive amounts of transactional processes in this industry that RPA can automate, from internal-only documents to deposits, withdrawals, and payouts.
- **Healthcare:** The benefits of RPA in healthcare industries include shorter waiting times, more detailed medical records, and reduced loads of administrative works.



RPA in Different Fields (Cont.)

- **Manufacturing:** The RPA in manufacturing industries does not only facilitate administrative works. It also works hand-in-hand with the existing software bots to streamline operational processes.
- **Transport and Logistics:** The order management, the distribution cycles of each order that customers make, and the supply chain linkages are the three things that become the benefits of RPA in the transport and logistics industries.
- **Utility companies:** The three products of utility companies are electricity, gas, and water. There are always monetary transactions in utility companies since many people need these three products. The job of RPA involves automating these transactions.

Differences between Test Automation and RPA

Parameter	Test Automation	RPA
Goal	Reduce Test execution time through automation	Reduce headcount through automation
Task	Automate repetitive Test Cases	Automate repetitive Business processes
Coding	Coding knowledge required to create Test Scripts	Wizard-driven, and coding knowledge not required
Tech Approach	Supports limited software environment. Example: Selenium can support only web applications.	Supports a wide array of software environments
Example	Test cases are automated	Data entry, forms, loan processing, is automated
Application	Test Automation can be run on QA, Production, Performance, UAT environments	RPA is usually run only on production environments
Implementation	It can automate a product.	It can automate a product as well as a service.
Users	Limited to technical users.	Can be used across the board by all stakeholders.

RPA Implementation Methodology

Industrial Robotic Process Automation (RPA) is a technology that automates rule-based, repetitive tasks within an industrial settings:

- Planning/Analysis
- Development/Bot Development
- Testing
- Support & Maintenance

RPA Implementation Methodology (Cont.)

Planning/Analysis:

- **Identify Business Objectives:** Begin by clearly defining the specific objectives and goals of implementing RPA in the industrial environment.
- **Process Assessment:** Conduct a detailed analysis of the existing industrial processes. Identify repetitive, rule-based, and high-volume tasks suitable for automation.
- **ROI Analysis:** Calculate the potential return on investment (ROI) for each identified process.
- **Technology Selection:** Select the appropriate RPA platform and tools that fit the industrial environment's requirements. Ensure compatibility with existing systems and technologies.
- **Infrastructure Readiness:** Assess the existing IT infrastructure and make necessary upgrades or changes to support the RPA implementation.
- **Compliance and Security:** Address data security and compliance concerns, ensuring that RPA implementation adheres to industry standards and regulations.

RPA Implementation Methodology (Cont.)

Development/Bot Development:

- **Design Automation Workflows:** Design the automation workflows and logic for the selected processes. Define how the bots will interact with various applications, systems, and data sources.
- **Bot Development:** Develop and configure the RPA bots using the chosen RPA platform. This includes creating scripts or workflows that mimic the actions of a human user.
- **Integration:** Integrate RPA bots with other enterprise systems and applications, allowing them to exchange data and perform end-to-end processes seamlessly.
- **Quality Assurance:** Conduct code reviews and quality assurance testing to ensure the RPA bots work as intended and meet the defined requirements.
- **Scalability:** Ensure that the RPA solution can scale to accommodate changes in process volume and complexity.



RPA Implementation Methodology (Cont.)

Testing:

- **Unit Testing:** Test individual RPA bots to ensure they perform their tasks accurately and efficiently.
- **Integration Testing:** Test the end-to-end automation process to ensure that all components and systems work together as expected.
- **User Acceptance Testing (UAT):** Involve end-users and subject matter experts in UAT to validate that the RPA solution meets their needs and expectations.
- **Performance Testing:** Assess the performance of the RPA bots under various conditions to identify and address any bottlenecks or issues.
- **Security Testing:** Perform security testing to identify and mitigate vulnerabilities in the RPA solution.

RPA Implementation Methodology (Cont.)

Support & Maintenance:

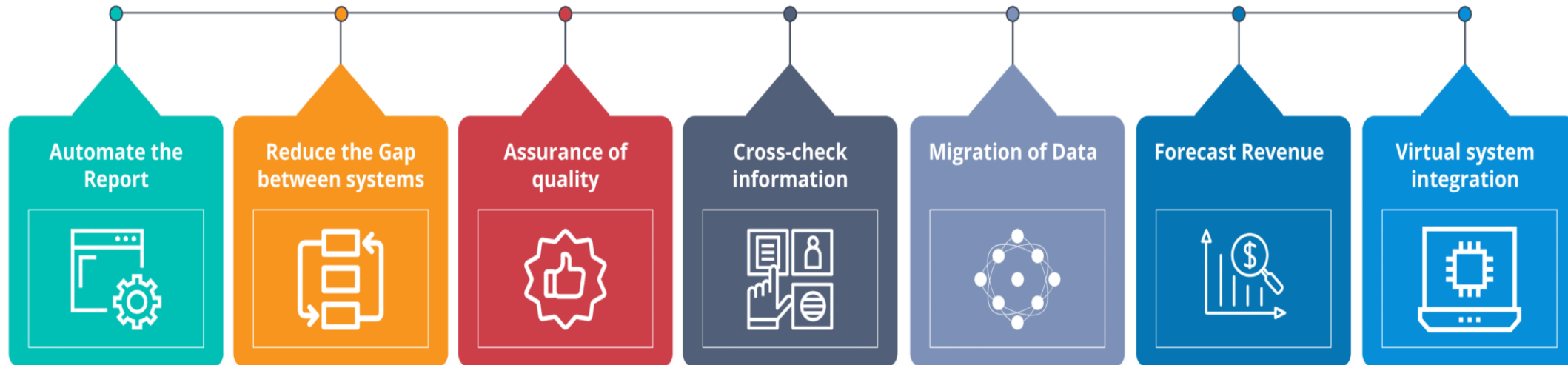
- **Monitoring:** Implement a monitoring system to continuously track the performance of RPA bots and processes. This includes error handling and alerting mechanisms.
- **Issue Resolution:** Establish a process for identifying and resolving issues, whether they are related to RPA bot failures, data discrepancies, or system integration problems.
- **Training:** Provide training to the RPA support and maintenance team to ensure they can manage, update, and troubleshoot the bots effectively.
- **Version Control:** Implement version control for RPA scripts and workflows to keep track of changes and maintain a history of bot development.
- **Continuous Improvement:** Regularly assess the RPA implementation to identify opportunities for improvement, including adding new automation tasks and optimizing existing ones.

Best Practices of RPA Implementation

- One should consider **business impact** before opting for RPA process
- Define and focus on the desired **ROI**
- Focus on **targeting larger groups** and automating large, impactful processes
- Combine attended and unattended RPA
- **Poor design**, change management can wreak **havoc**
- Don't forget the impact on people

General Use of RPA

General Use of RPA



Application of RPA



Cases for RPA uses

- Financial transactions are the most frequent areas that require the RPA process. Such things include processing sales orders, refunds, payrolls for employees, and invoicing.
- The RPA can even help in comparing prices (ranges) between similar products or services. Furthermore, the RPA use cases in financial transactions also include automating the financial report generation processes.
- The RPA application can also automate customer service responses. As a result, companies that apply the RPA tools will shorten their times to generate leads for marketing purposes.

Robotic Process Automation tools

Selection of RPA Tool should be based on following 4 parameters:

- **Data:** Easy of reading and writing business data into multiple systems
- **Type of Tasks mainly performed:** Ease of configuring rules-based or knowledge-based processes.
- **Interoperability:** Tools should work across multiple applications
- **AI:** Built-in AI support to mimic human users
- Popular Robotic Automation Tools: **Blue prism, Automation AnyWhere, UiPath etc.**

Robotic Process Automation tools (Cont.)

Popular RPA Tools

1. **Eggplant** – The end-to-end automation process is one of Eggplant’s specializations that lets it interact with multiple systems in a device.
2. **JAMS by HelpSystems** – This enterprise IT solution centralizes the scheduling process in a company across all platforms and applications.
3. **Power Automate by Microsoft** with the MS security technology-protected tool automation feature.
4. **UiPath** – The open-source RPA tool can automate any desktop or web apps.
5. **Blue Prism** does not require advanced programming skills to generate flowcharts.

Robotic Process Automation tools (Cont.)

Popular RPA Tools (Cont.)

6. **OpenConnect** with the secure and encrypted data system that addresses almost all types of operational and competitive challenges.
7. **WorkFusion**, a SaaS crowd computing platform that lets the users deploy as many RPA robots as they need to automate software tasks for a group.
8. **Pega**, an RPA tool for medium and large businesses to provide cloud-based solutions.
9. **Nice System** with its virtual assistant to automate mundane employee tasks.
10. **Kofax** integrates well with the Kapow Katalyst platform.



Robotic Process Automation tools (Cont.)

Tools Comparison

- **UiPath, Automation Anywhere, and Blue Prism** are three examples of RPA use cases that belong to the Top 3 category.
- **Blue Prism** is the most convenient to use since people can access it through mobile applications. The execution speed is also an impressive one. It makes companies like IBM and Deloitte trust Blue Prism.
- **UiPath** is another user-friendly RPA tool that does not require any programming knowledge. So, it is suitable for critical sectors that need emergency responses to anything like the healthcare and finance sectors.
- **Automation Anywhere** has Microsoft technology as the base that employees worldwide know, understand, and use. It can deploy robots on large scales, too.



Benefits of RPA

- Large numbers of the process can easily have automated.
- **Cost are reduced** significantly as the RPA takes care of repetitive task and **saves precious time** and resources.
- Programming skills are not needed to configure a software robot. Thus, any non-technical staff can set up a bot or even record their steps to automate the process.
- Robotic process automation support and allows all regular compliance process, with error-free auditing.
- The robotic software can rapidly model and **deploy the automation process**.
- The defects are tracked for each test case story and the sprint.
- There is no human business which means there is no need for time for the requirement of training.
- Software robots do not get tired. It increases which helps **to increase the scalability**

Disadvantages of RPA

- **Limited adaptability to physical tasks:** RPA is primarily designed for automating digital and rule-based tasks. It cannot handle physical tasks, which are prevalent in manufacturing and certain industrial settings. This limitation can restrict its applicability in industries that rely heavily on physical processes.
- **High upfront costs:** Implementing RPA in an industrial environment can be expensive, especially when it involves integrating with existing machinery or equipment.
- **Initial setup and implementation costs:** RPA implementation can be expensive due to software costs, training, and infrastructure setup, making it a barrier for some organizations.
- **Lack of adaptability:** RPA bots may struggle to adapt to changes in processes, requiring reprogramming when significant changes occur, which can be time-consuming.
- **Resistance from employees:** Employees may resist RPA implementation due to concerns about job security and the potential for job displacement, making change management essential.

Other Autonomous Industrial Process

Programmable Logic Controllers (PLCs), Industrial PCs, and Distributed Control Systems (DCS) are all essential components in the context of autonomous industrial processes.

These systems play different but interconnected roles in ensuring the smooth and efficient operation of various industrial processes:

- PLCs (Programmable Logic Controllers)
- Industrial PCs
- DCS (Distributed Control Systems)



Other Autonomous Industrial Process (Cont.)

PLCs (Programmable Logic Controllers):

- PLCs are widely used in industrial automation to control and monitor a variety of processes and equipment. They are designed for real-time control and are well-suited for tasks that require high-speed and precision.
- In autonomous industrial processes, PLCs often serve as the "brains" of the system, executing control algorithms and logic that enable machines and devices to operate autonomously.
- PLCs can process sensor data, make decisions based on predefined logic, and issue commands to actuators and other devices, allowing for autonomous control of equipment and processes.



Other Autonomous Industrial Process (Cont.)

Imagine an automated conveyor system in a manufacturing facility. In this scenario, PLCs can be used to control the movement of products along the conveyor line.

- Multiple PLCs are distributed along the conveyor line, each responsible for a specific section.
- PLCs are connected to sensors that detect the presence of products and control motors to move them.
- The PLCs communicate with each other to ensure smooth and synchronized movement of products.



Other Autonomous Industrial Process (Cont.)

Industrial PCs:

- Industrial PCs are more versatile and powerful than PLCs. They can run complex software applications, including advanced control algorithms and data analytics.
- In autonomous industrial processes, industrial PCs can serve as the central computing platform for overseeing and coordinating various aspects of the operation.
- They can handle data processing, machine learning, and communication with other systems, making them critical for managing and optimizing autonomous operations.



Other Autonomous Industrial Process (Cont.)

In a food processing plant, quality control and data analysis are critical. Industrial PCs can be used for these purposes:

- Industrial PCs are placed at various points on the production line to capture images of food products.
- These PCs use image recognition software to assess product quality and identify defects.
- Data from the inspections are collected, stored, and analyzed in real-time by the industrial PCs.
- Visualization software on the PCs provides operators with real-time data on product quality.



Other Autonomous Industrial Process (Cont.)

DCS (Distributed Control Systems):

- DCS is a specialized control system used in large-scale industrial processes. It comprises multiple PLCs and controllers distributed throughout a facility, often connected to a central supervisory control system.
- DCS is essential in autonomous industrial processes as it provides centralized monitoring and control of a wide range of subsystems and processes.
- It enables integration, coordination, and optimization of various control loops, helping to maintain efficient and autonomous operation of the entire industrial facility.

Other Autonomous Industrial Process (Cont.)

A petrochemical refinery is a complex industrial process requiring extensive control and monitoring. DCS is well-suited for this environment:

- Distributed Control Systems are installed throughout the refinery, controlling various processes such as distillation, chemical reactions, and safety systems.
- DCS provides a centralized control room where operators can monitor and manage the entire facility.
- Redundant controllers ensure high availability and fault tolerance.
- Historical data logging allows for process analysis and compliance with regulatory requirements.

Summary

- Autonomous industrial processes aim to achieve high levels of automation, reducing the need for human intervention. This involves designing and refining systems to operate independently, enhancing industrial efficiency.
- RPA is a technology that uses software robots to automate repetitive tasks, resulting in increased productivity and reduced errors. It finds applications across various fields and industries.
- Understanding and defining levels of autonomy is crucial for implementing autonomous systems. These levels range from fully manual to fully autonomous, with varying degrees of human involvement.
- Successful RPA implementation requires following specific methodologies and best practices to ensure seamless integration, efficiency gains, and risk mitigation.
- Industrial autonomy goes beyond basic automation by striving for self-sufficiency in processes. Automation fuels autonomy by reducing the need for human intervention and decision-making.



Review Questions

1. What are the key differences between automation and autonomy in industrial processes, and how does automation contribute to achieving higher levels of autonomy?
2. Can you explain the concept of levels of autonomy in autonomous industrial processes and provide examples of how different levels impact operations and human involvement?
3. What are the main advantages and disadvantages of Robotic Process Automation (RPA), and in which fields or industries is RPA most commonly applied?
4. Describe the best practices and methodologies for implementing RPA solutions, and explain how these practices ensure a successful and efficient integration of RPA in business processes.
5. How does the design and refinement of autonomous systems play a crucial role in achieving an autonomous industrial process, and what are the key features that need to be considered in the design of such systems?

References

- Refresh Science (2021). Robotic Process Automation (RPA) – PowerPoint Presentation. Retrieved 31 October 2023, from <https://refreshscience.com/robotic-process-automation-rpa-ppt/>
- SAE (2018). Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, Standard J3016_201806, Rev 2018-06-15, SAE International. VDI (2017). Automation engineering of modular systems in the process industry. Part 1: General Concept and Interfaces, VDI-Standard: VDI/VDE/Namur 2658, Beuth Verlag.
- Müller, M., Müller, T., Ashtari Talkhestani, B., Marks, P., Jazdi, N., & Weyrich, M. (2021). Industrial autonomous systems: a survey on definitions, characteristics and abilities. *at-Automatisierungstechnik*, 69(1), 3-13.
- Lopez, C. P., Santórum, M., & Aguilar, J. (2019). Autonomous cycles of collaborative processes for integration based on industry 4.0. In *Information Technology and Systems: Proceedings of ICITS 2019* (pp. 177-186). Springer International Publishing.
- Gamer, T., Hoernicke, M., Kloepper, B., Bauer, R., & Isaksson, A. J. (2019). The autonomous industrial plant-future of process engineering, operations and maintenance. *IFAC-PapersOnLine*, 52(1), 454-460.



Lecture 05: Automation and Robotic Components for Industrial

Objectives

- Gain a comprehensive understanding of various robotic technologies and their applications in industrial, automotive, aerial, marine, and space domains.
- Investigate the diverse software architectures used in robotics, including their role in enabling autonomous behavior and coordination.
- Examine training algorithms and techniques applied in autonomous systems, considering their impact on system performance and learning.
- Explore the historical developments and influential figures in the field of robotics to appreciate the evolution of these technologies.
- Assess the current and future impacts of autonomous systems on industries, society, and technological advancements.

Outlines



- Types of Robots in Industrial
- Current Uses of Robots
- Robot Software Architectures
- Forms of Software Architectures
- A Newer Form
- Autonomous Vehicles
- Mission Planning
- Some Details
- Path Planning
- Following a Path
- Sensor Interpretation
- Performing Sensor Interpretation
- Obstacle Avoidance
- Failure Handling/Recovery
- Autonomous Ground Vehicles
- Road-Based AVs
- More Road-Based AVs
- RALPH
- Another Approach: ALVINN
- Training
- Over Training
- Training Algorithm
- More on ALVINN
- ALVINN Hybrid Architecture
- Stanley
- Images of Stanley
- Stanley Software
- Processing Pipeline
- Sensors
- DARPA Path Planning
- Higher Level Planning
- The DARPA Grand Challenge Race
- Autonomous Aircraft
- Autonomous Submarines
- Autonomous Space Probes
- Mars Rovers
- Rodney Brooks/MIT
- Requirements for Robot Construction
- The Approach
- Brooks Round 1: Small Mobile Robots
- Brooks Round II: Cardea
- Cardea Detecting Doors
- Cardea's Behavior
- Brooks Round III: Cog
- Rationale Behind Cog
- Cog's Capabilities
- Brooks Round IV: Lazlo
- Brooks Round V: Meso
- Brooks Round VI: Theory of Body



Types of Robots in Industrial

- Mobile robots – robots that move freely in their environment
 - We can subdivide these into indoor robots, outdoor robots, terrain robots, etc based on the environment(s) are programmed to handle.
- Robotic arms – stationary robots that have manipulators, usually used in construction (e.g., car manufacturing plants)
 - These are usually not considered AI because they do not perform planning and often have little to no sensory input.
- Autonomous vehicles – like mobile robots, are a combination of vehicle and computer controller
 - Autonomous cars, autonomous plane drones, autonomous helicopters, autonomous submarines, autonomous space probes
 - There are different classes of autonomous vehicles based on the level of autonomy, some are only semi-autonomous.



Types of Robots in Industrial (Cont.)

- Soft robots – robots that use soft computing approaches (e.g., fuzzy logic, neural networks)
- Mimicking robots – robots that learn by mimicking
 - For instance, robots that learn facial gestures or those that learn to touch or walk or play with children.
- Softbots – software agents that have some degrees of freedom (the ability to move) or in some cases, software agents that can communicate over networks
- Nanobots – theoretical at this point, but like mobile robots, they will wander in an environment to investigate or make changes
 - The environment will be microscopic worlds, e.g., the human body, inside of machines.



Current Uses of Robots

- There are over 3.5 million robots in use in society of which, about 1 million are industrial robots.
 - 50% in Asia, 32% in Europe, 16% in North America
- Factory robot uses:
 - Mechanical production, e.g., welding, painting
 - Packaging – often used in the production of packaged food, drinks, medication
 - Electronics – placing chips on circuit boards
 - Automated guided vehicles – robots that move along tracks, for instance as found in a hospital or production facility
- Other robot uses:
 - Bomb disabling
 - Exploration (volcanoes, underwater, other planets)
 - Cleaning – at home, lawn mowing, cleaning pipes in the field, etc
 - Fruit harvesting

Robot Software Architectures

- Traditionally, the robot is modeled with centralized control
 - A central processor running a central process is responsible for planning
 - Other processors are usually available to control motions and interpret sensor values
 - passing the interpreted results back to the central processor
- In such a case, we must implement a central reasoning mechanism with a pre-specified representation
 - Identify a reasonable process for planning and a reasonable representation for representing the plan in progress and the environment



Forms of Software Architectures

- **Human controlled** – of no interest to us in AI
- **Synchronous** – central control of all aspects of the robot
- **Asynchronous** – central control for planning and decision making, distributed control for sensing and moving parts
- **Insect-based** – with multiple processors, each processor contributes as if they constitute a colony of insects contributing to some common goal
- **Reactive** – no pre-planning, just reaction (usually synchronized), also known as **behavioral control**:
 - A compromise is to use a 3-layered architecture, the bottom layer is reactive, the middle layer keeps track of reactions to make sure that the main plan is still be achieved, and the top level is for planning that is used when reactive planning is not needed.



A Newer Form

- **Subsumption** – The robot is controlled by simple processes rather than a centralized reasoning system:
 - Each process might run on a different processor
 - The various processes compete to control the robot
 - Processes are largely organized into layers of relative complexity (although no layer is particularly complex)
 - Layers typical lack variables or explicit representations and are often realized by simple finite state automata and minimal connectivity to other layers
- This is a reactive-based architecture with minimal planning.
- This is also known as a **behavioral-based architecture**.



Autonomous Vehicles

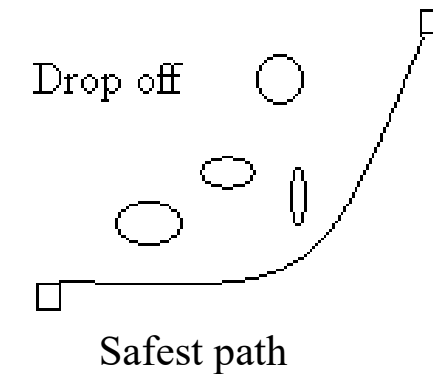
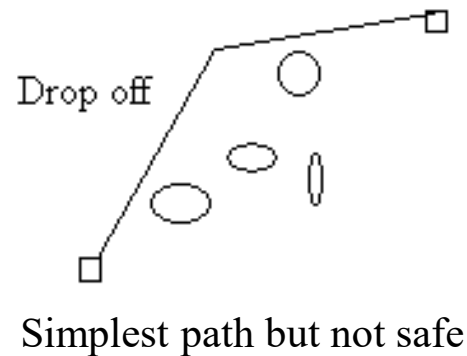
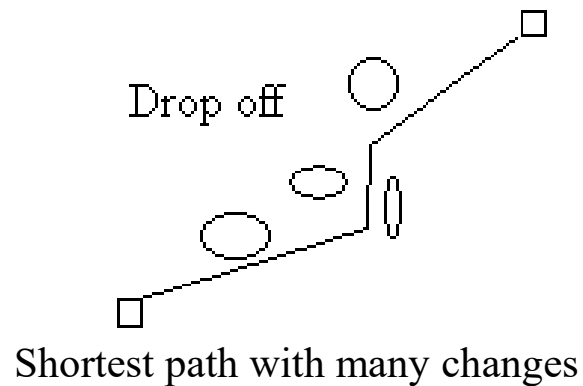
- Since industrial robots largely do not require much or any AI, we are mostly interested in autonomous vehicles whether they are based on actual vehicles, or just mobile machines.
- What does an autonomous vehicle need?
 - they usually have high-level goals provided to them
 - from the goal(s), they must plan how to accomplish the goal(s)
 - **mission planning** – how to accomplish the goal(s)
 - **path planning** – how to reach a given location
 - **sensor interpretation** – determining the environment given sensor input
 - **obstacle avoidance and terrain sensing**
 - **failure handlings/recovery from failure**

Mission Planning

- As the name implies, this is largely a planning process
 - How to accomplish given goals?
 - This may be through rule-based planning, plan decomposition, or plans may be provided by human controllers.
 - In many cases, the mission goal is simple: go from point A to point B so that no planning is required.
 - For a mobile robot (not an autonomous vehicle), the goals may be more diverse:
 - reconnaissance and monitoring
 - search (e.g., find enemy locations, find buried land mines, find trapped or injured people)
 - go from point A to point B but stealthily
 - monitor internal states to ensure mission is carried out

Some Details

- The robot must balance the desire for the safest path, the shortest distance path, and the path that has fewer changes of orientation.
 - Variations of the A* algorithm (best-first search) might be used.
 - Heuristics might be used to evaluate safety versus simplicity versus distance.



Path Planning

- How does the vehicle/robot get from point A to point B?
 - Are there obstacles to avoid? Can obstacles move in the environment?
 - Is the terrain going to present a problem?
 - Are there other factors such as dealing with water current (autonomous sub), air current (autonomous aircraft), blocked trails (indoor or outdoor robot)?
- Path planning is largely geometric and includes:
 - Straight lines
 - Following curves
 - Tracing walls
- Additional issues are:
 - How much of the path can be viewed ahead?
 - Is the robot going to generate the entire path at once, or generate portions of it until it gets to the next point in the path, or just generate on the fly?
 - If the robot gets stuck, can it backtrack?



Following a Path

- Once a path is generated, the robot must follow that path, but the technique will differ based on the type of robot.
 - For an indoor robot, path planning is often one of following the floor.
 - Using a camera, find the lines that make up the intersection of floor and wall, and use these as boundaries to move down
 - For an autonomous car, path planning is similar but follows the road instead of a floor.
 - Using a camera, find the sides of the road and select a path down the middle
 - For an all-terrain vehicle, GPS must be used although this may not be 100% accurate.

Sensor Interpretation

- Sensors are primarily used to
 - ensure the vehicle/robot is following an appropriate path (e.g., corridor, road)
 - and to seek out obstacles to avoid
- It used to be very common to equip robots with sonar or radar but not cameras because
 - cameras were costly
 - vision algorithms required too much computational power and were too slow to react in real time
- Outdoor vehicles/robots commonly use cameras and lasers (if they can be afforded).
- Additionally, a robot might use GPS,
 - so the robot needs to interpret input from multiple sensors

Performing Sensor Interpretation

- There are many forms:
 - Simple neural network recognition
 - more common if we have a single source of input, e.g., camera, so that the NN can respond with “safe” or “obstacle”
 - Fuzzy logic controller
 - can incorporate input from several sensors
 - Bayesian network and hidden Markov models
 - for single or multiple sensors
 - Blackboard/KB approach
 - post sensor input to a blackboard, let various agents work on the input to draw conclusions about the environment
- Since sensor interpretation needs to be real-time, we need to make sure that the approach is not overly elaborate.



Obstacle Avoidance

- What happens when an obstacle is detected by sensors? It depends on the type of robot and the situation
 - in a mobile robot, it can stop, re-plan, and resume
 - in an autonomous ground vehicle, it may slow down and change directions to avoid the obstacle (e.g., steer right or left) while making sure it does not drive off the road – notice that it does not have to re-plan because it was in motion and the avoidance allowed it to go past the obstacle
 - or it might stop, back up, re-plan and resume
 - an underwater vehicle or an air-based vehicle may change depth/altitude
- While obstacle avoidance is a low-level process, it may impact higher level processes (e.g., goals) so replanning may take place at higher levels.



Failure Handling/Recovery

- If the vehicle is not 100% autonomous, it may wait for new instructions.
- If the vehicle is on its own, it must first determine if the obstacle is going to cause the goal-level planning to fail
 - if so, replanning must take place at that level taking into account the new knowledge of an obstacle
 - if not, simple rules might be used to get it around the obstacle so that it can resume
- If a failure is more severe than an obstacle (e.g., power outage, sensor failure, uninterpretable situation)
 - then the ultimate failsafe is to stop the robot and have it send out a signal for help
 - if the robot is a terrain vehicle, it may “pull over”
 - a submarine may surface and broadcast a message “help me”
 - what about an autonomous aircraft?

Autonomous Ground Vehicles

- The most common form of AV is a ground vehicle
 - We can break these down into four categories
 - Road-based autonomous automobiles
 - automatic cars programmed to drive on road ways with marked lanes and possible must contend with other cars
 - All-terrain autonomous automobiles
 - automatic cars/jeeps/SUVs programmed to drive off road and must contend with different terrains with obstacles like rocks, hills, etc
 - All-terrain robots
 - like the all-terrain automobiles but these can be smaller and so more maneuverable – these may include robots that use tank treads instead of wheels
 - Crawlers
 - like all-terrain robots except that they use multiple legs instead of wheels/treads to maneuver



Road-Based AVs

- We currently do not have any truly autonomous road-based AVs but many research vehicles have been tested
 - **NavLab5 (CMU) – performed “no hands across America”**
 - the vehicle traveled from Pittsburgh to San Diego with human drivers only using brakes and accelerator, the car did all of the steering using RALPH
 - **ARGO (Italy) drove 2000 km in 6 days**
 - using stereoscopic vision to perform lane-following and obstacle avoidance, human drivers could take over as needed, either complete override or to change behavior of the system (e.g., take over steering, take over speed)

More Road-Based AVs

- Both **NavLab** and **ARGO** would drive on normal roads with traffic
- The **CMU Houston-Metro Automated bus** was designed to be completely autonomous
 - Only drive in specially reserved lanes for the bus so that it did not have to contend with other traffic
 - Two buses tested on a 12 km stretch of Interstate 15 near San Diego, a stretch of highway designated for automated transit
 - As with NavLab, the Houston-Metro buses use RALPH (see the next slide)
- CityMobile – European sponsored approach for vehicles that not only navigate through city streets autonomously,
 - but perform deliveries of people and goods
 - for such robots, the “mission” is more complex than just go from point a to point b, these AVs have higher level planning



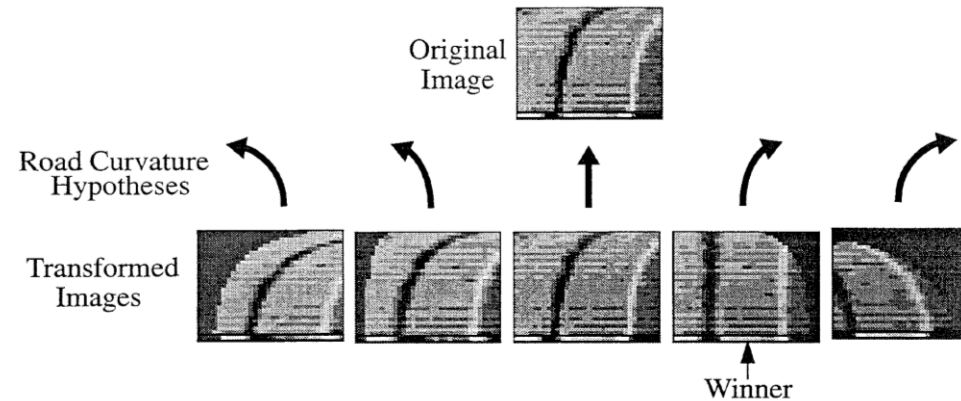
RALPH

● Rapidly Adapting Lateral Position Handler

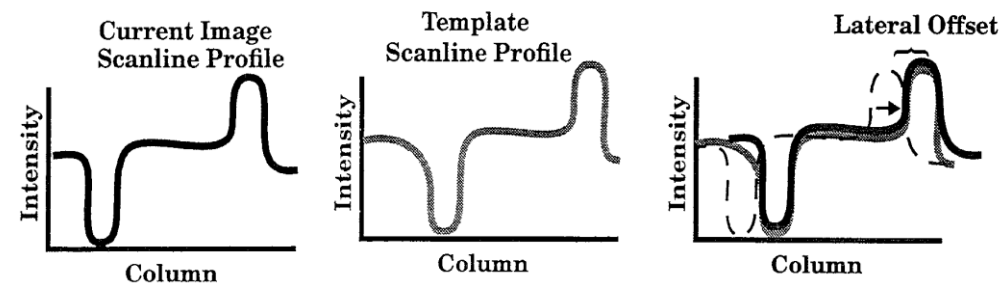
- Steering is decomposed into three steps:
 - Sampling the image (the painted lines of a road, the edges/berms/curbs)
 - Determining the road curvature
 - Determining the lateral offset of the vehicle relative to the lane center
- The output of the latter two steps are used to generate steering control
- Image is sampled via camera and A/D convertor board
 - the scene is depicted in grey-scale along with enhancement routines
 - a trapezoidal region is identified as the road, and the rest of the image is omitted (as unimportant)
- RALPH uses a “hypothesize and test” routine to map the trapezoidal region to possible curvature in the road to update its map (see the next slide).



RALPH (Cont.)

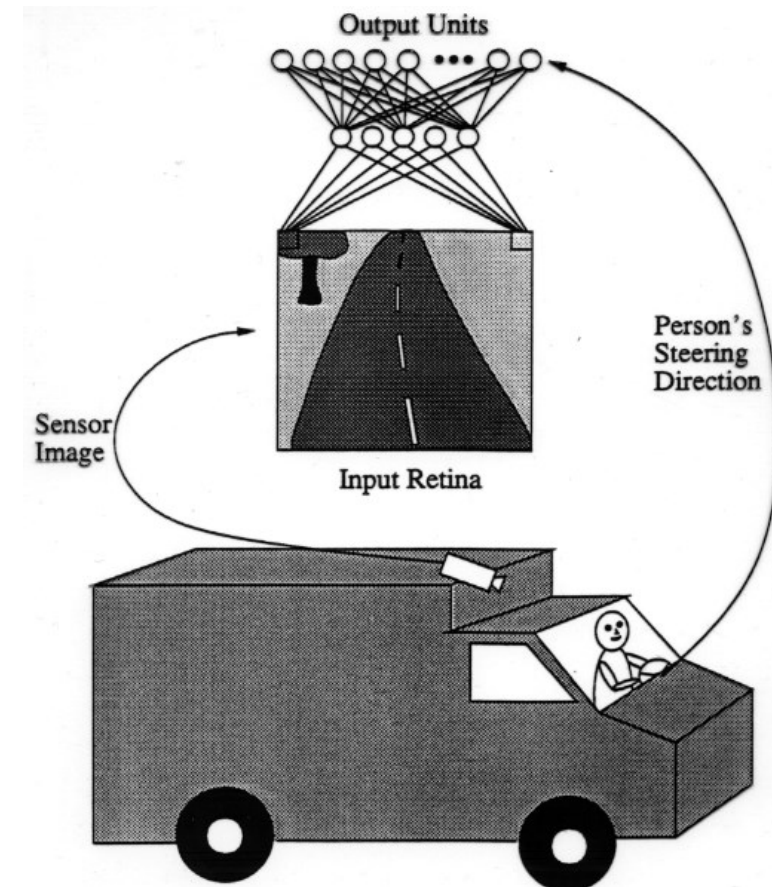


- The curvature is processed using a variety of different techniques and summed into a “scan line”
 - RALPH uses 32 different templates of “scan lines” to match the closest one which then determines the lateral offset (steering motion).



Another Approach: ALVINN

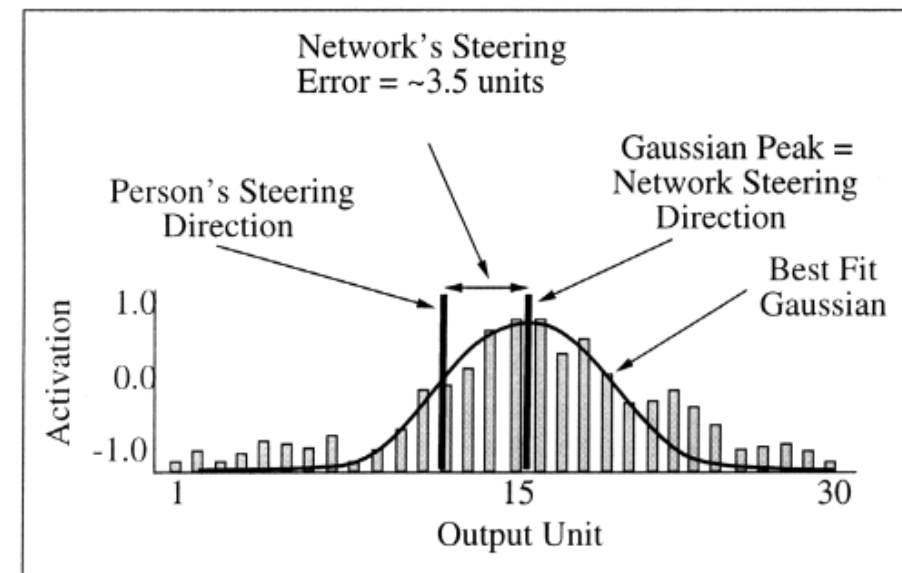
- A different approach is taken in **ALVINN** which uses a trained neural network for vehicular control:
 - The neural network learns steering actions based on camera input:
 - The neural network is trained by human response.
 - The input is the visual signal and the feedback into the backprop algorithm is what the human did to the steering wheel.



Training

- Training feedback combines the actual steering as performed by the human with a Gaussian curve to denote “typical” steering
 - Computed error for backprop is actual steering – Gaussian curve value.

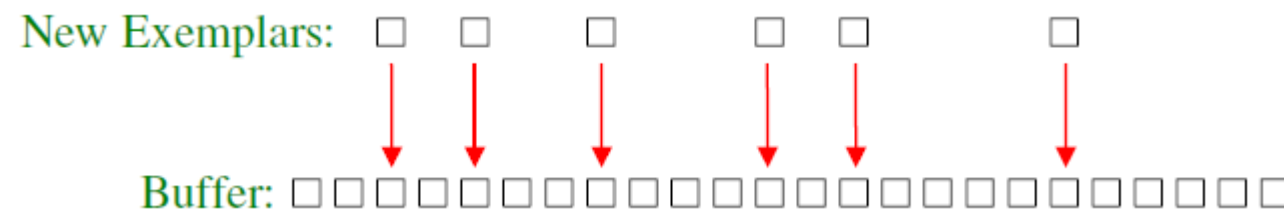
- Additionally, if the human drives well, the system doesn’t learn to make steering corrections
 - Video images are randomly shifted slightly to provide the NN with the ability to learn that keeping a perfectly straight line is not always desired.





Over Training

- As we discussed when covering NNs, performing too many epochs of the training set may cause the NN to over train on that set:
 - The problem is that the NN may forget how to steer with older images as training continues.
 - The solution generated is to keep a buffer for older images along with the new images:
 - the buffer stores 200 images
 - 15 old images are discarded for new ones by replacing images with the lowest error and/or replacing images with the closest steering direction to the current images



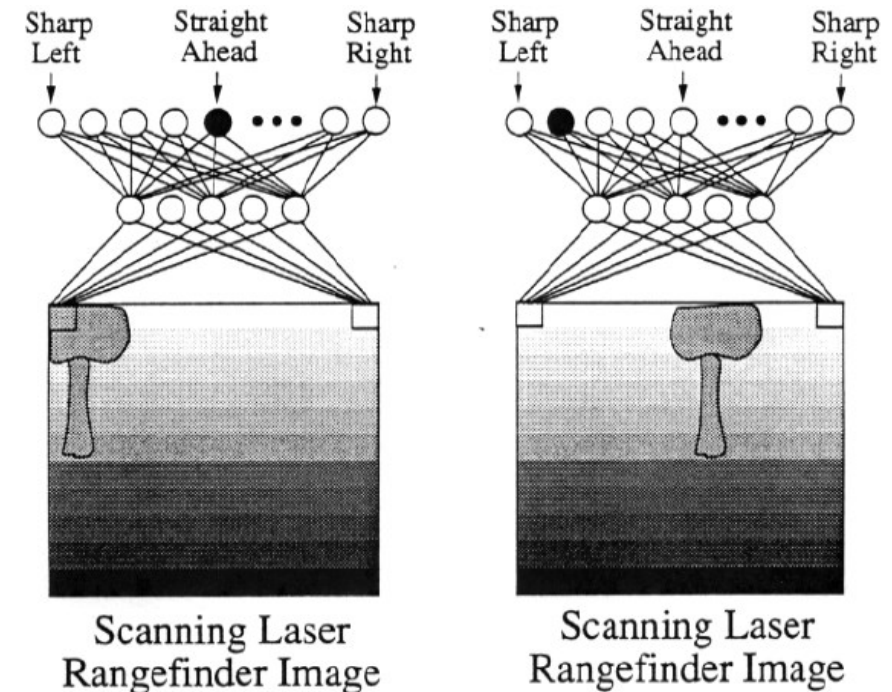


Training Algorithm

- Take current camera image + 14 shifted/rotated variants each with computed steering direction
 - Replace 15 old images in the buffer with these 15 new ones
 - Perform one epoch of backprop
 - Repeat until predicted steering reliably matches human steering
- The entire training only takes a few minutes although during that time, the training should encounter all possible steering situations
 - Two problems with the training approach are that
 - ALVINN is capable of driving only on the type of road it was trained on (e.g., black pavement instead of grey)
 - ALVINN is only capable of following the given road, it does not learn paths or routes, so it does not for instance turn onto another road way

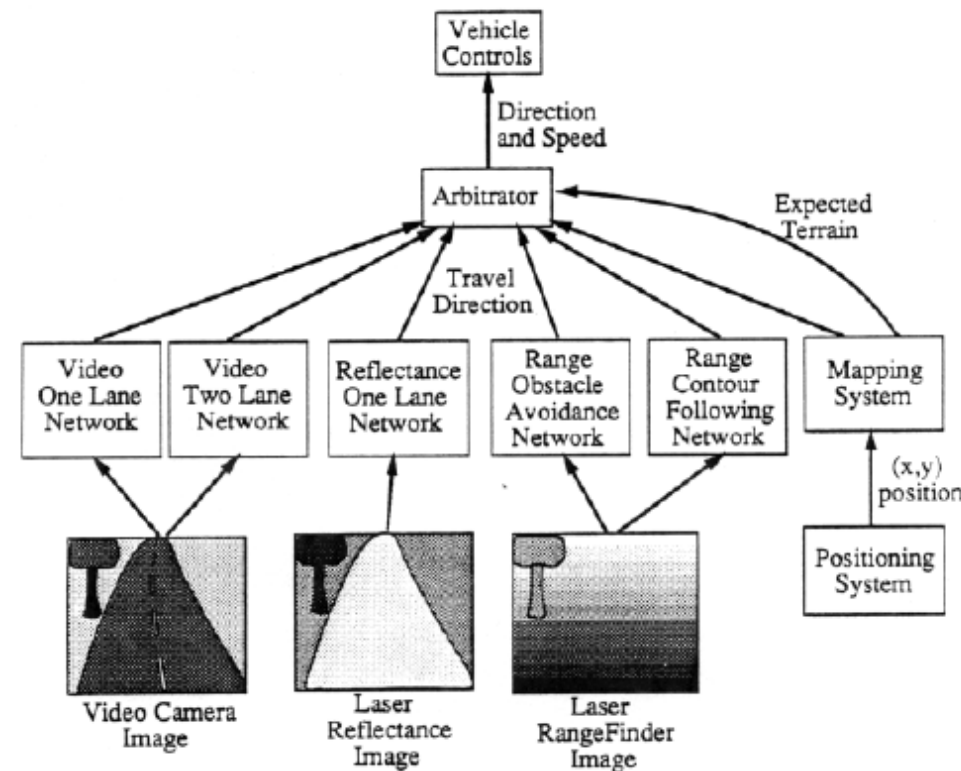
More on ALVINN

- To further enhance ALVINN, obstacle detection and avoidance can be implemented:
 - use a laser rangefinder to detect obstacles in the roadway.
 - train the system on what to do when confronted by an obstacle (steer to avoid, stop)
 - ALVINN can also drive at night using a laser reflectance image.





ALVINN Hybrid Architecture



By combining the steering NN, the obstacle avoidance NN, a path planner, and a higher level arbiter, ALVINN can be a fully autonomous ground vehicle.

Stanley

- We wrap up our examination of autonomous ground vehicles with Stanley, the 2005 winner of the DARPA Grand Challenge road race:
 - Based on a VW Touareg 4 wheel vehicle
 - DC motor to perform steering control
 - Linear actuator for gear shifting (drive, reverse, park)
 - Custom electronic actuator for throttle and brake control
 - Wheel speed, steering angle sensed automatically
 - Other sensors are:
 - five SICK laser range finders (mounted on the roof at different tilt angles) which can cover up to 25 m
 - a color camera for long distance perception
 - two RADAR sensors for forward sensing up to 200 m

Images of Stanley

- The top-mounted sensors (lasers)
- Computer control mounted in the back on shock absorbers
- Actuators to control shifting



Stanley's lasers can find obstacles in a cone region in front of the vehicle up to 25 m.

Stanley Software

- There is no centralized control, instead there are modules to handle each subsystem (approximately 30 of them operating in parallel).
 - Sensor data are time stamped and passed on to relevant modules.
 - The state of the system is maintained by local processes, and that state is communicated to other modules as needed.
 - Environment state is broken into multiple maps:
 - laser map
 - vision map
 - radar map
 - The health of individual modules (software and hardware) are monitored so that modules can make decisions based in part on the reliability of information coming from each module.

Processing Pipeline

- Sensor data time stamped, stored in a database of course coordinates, and forwarded
- Perception layer maps sensor data into vehicle orientation, coordinates and velocities
 - This layer creates a 2-D environment map from laser, camera and radar input.
 - Road finding module allows vehicle to be centered laterally.
 - Surface assessment module determines what speed is safe for travel (based on the roughness of the road, obstacles sited, and on whether the camera image is interpretable).
- The control layer regulates the actuators of the vehicle, this layer includes
 - Path planning to determine steering and velocity needed.
 - Mission planning which amounts to a finite state automata that dictates whether the vehicle should continue, stop, accept user input, etc.
- Higher levels include user interfaces and communication

Sensors

- Lasers are used for terrain labeling
 - Obstacle detection
 - Lane detection and orientation (levelness)
 - these decisions are based on pre-trained hidden Markov models
- Lasers can detect obstacles at a maximum range of 22 m which is sufficient for Stanley to avoid obstacles if traveling no more than 25 mph.
 - The color camera is used to longer range obstacle detection by taking the laser mapped image of a clear path and projecting it onto the camera image to see if that corridor remains clear
 - obstacle detection in the camera image is largely based on looking for variation in pixel intensity/color using a Gaussian distribution of likely changes
 - If the camera fails to find a drivable corridor, speed is reduced to 25 mph so that the lasers can continue to find an appropriate path.

DARPA Path Planning

- Prior to the race, DARPA supplied all teams with a RDDF file of the path
 - This eliminated the need for global path planning from Stanley
 - What Stanley had to do was
 - Local obstacle avoidance
 - Road boundary identification to stay within the roadway
 - Maintain a global map (aided by GPS) to determine where in the race it currently was
 - Note that since there is some degree of error in GPS readings, Stanley had to update its position on the map by matching the given RDDF file to its observation of turns in the road
 - Perform path smoothing to make turns easier to handle and match predicted road curvature to the actual road

Higher Level Planning

- Unlike ordinary AVs, this did not really affect Stanley
 - Stanley's only goal was to complete the race course in minimal time
 - Path planning was largely omitted
 - Obstacle avoidance, lane centering and trajectory computations were built into lower levels of the processing pipeline
 - Updating the map of its location was important
 - Stanley would drop out of automatic control into human control if needed (no such situation arose) or it would stop if commanded by DARPA
 - This could arise because Stanley was being approached or was approaching another vehicle, pausing the vehicle would allow the vehicles to all operate with plenty of separation – Stanley was paused twice during the road race



The DARPA Grand Challenge Race

- The race was approximately 130 miles in desert terrain that included wide, level spans and narrow, slanted and rocky areas.
- 2 hours before the race, teams were provided the race map, 2935 GPS coordinates, and associated speed limits for the different regions of the race:
 - Stanley was paused twice, to give more space to the CMU entry in front of it
 - After the second pause, DARPA paused the CMU entry to allow Stanley to go past it
- Stanley completed the race in just under 7 hours averaging 19.1 mpg having reached a top speed of 38 mpg:
 - 195 teams registered, 23 raced and only 5 finished



Autonomous Aircraft

- Today, most AAVs are drone aircraft that are remote controlled
 - The AV must perform some of the tasks such as course alteration caused because of air current or updraft, etc, but largely the responsibility lies on a human operator
 - There are also autonomous helicopters
- Another form of flying autonomous vehicle are smart missiles
 - These are laser guided but the missile itself must
 - make midcourse corrections
 - identify a target based on shape and home in on it
- Because of the complexity of flying and the need for precise, real-time control, true AAVs are uncommon and research lags behind other forms



Autonomous Submarines

- Unlike the AAVs, AUVs (U = underwater) are more common
 - Unlike the ground vehicles, AUVs have added complexity
 - 3-D environment
 - water current
 - lack of GPS underwater
 - AUVs can be programmed to reach greater depths than human-carrying submarines
 - AUVs can carry out such tasks as surveillance and mine detection, or they may be exploration vessels
 - One easy aspect of an AUV is failure handling, if the AUV fails, all it has to do is surface and send out a call for help
 - if the AUV holds oxygen on board, its natural state is to float on top of the water, so the AUV will not sink unless it is punctured or trapped underneath something

Autonomous Space Probes

- Most of our space probes are not very autonomous
 - They are too expensive to risk making mistakes in decision making
 - orbital paths are computed on Earth
- However, due to the distance and time lag for signals to reach the space probes, the probes must have some degree of autonomy
 - They must monitor their own health.
 - They must control their own rockets (firing at the proper time for the proper amount of time) and sensors (e.g., aiming the camera at the right angle).
- Probes have reached as far as beyond Neptune (Voyager II), Saturn (Cassini) and Jupiter (Galileo).

Mars Rovers

- Related to the ground-based AVs, Spirit and Opportunity are two small ground all-terrain AVs on Mars
 - The most remarkable thing about these rovers is their durability.
 - Their lifetime was estimated at 3 months but are still functioning 5 ½ years on.
 - Mission planning is entirely dictated by humans but path planning and obstacle avoidance is left almost entirely to the rovers themselves.
 - New software can be uploaded allowing us to reprogram the rovers over time.
 - The rovers can also monitor their own health (predominantly battery power and solar cells).

Rodney Brooks/MIT

- Brooks is the originator of the subsumption architecture (which itself led to the behavioral architecture).
- Brooks argues that robots can evolve intelligence without a central representation or any pre-specified representations.
- He argues as follows:
 - Incrementally build the capabilities of an intelligent system
 - During each stage of incremental development, the system interacts in the real world to learn
 - No explicit representations of the world, no explicit models of the world, these will be learned over time and with proper interaction
 - Start with the most basic of functions – the ability to move about in the world amid obstacles while not becoming damaged, even if people are deliberately trying to confuse them or get in their way



Requirements for Robot Construction

- Brooks states that for a robot to succeed, its construction needs to follow a certain methodology:
 - The robot must cope appropriately and timely with changes in its environment.
 - The robot should be robust with respect to its environment (minor changes should not result in catastrophic failure, graceful degradation is required).
 - The robot should maintain multiple goals and change which goals it is pursuing based in part on the environment – i.e., it should adapt.
 - The robot should do something in the world, have a purpose.



The Approach

- Each level is a fixed-topology network of finite state machines
 - Each finite state machine is limited to a few states, simple memory, access to limited computation power (typically vector computations), and access to 1 or 2 timers.
 - Each finite state machine runs asynchronously.
 - Each finite state machine can send and receive simple messages to other machines (including as small as 1-bit messages).
 - Each finite state machine is data driven (reactive) based on messages received
 - connections between finite state machines is hard-coded (whether by direct network, or by pre-stated address)
 - A finite state machine will act when given a message, or when a timer elapses.
- There is no global data, no global decision making, no dynamic establishment of communication.



Brooks Round 1: Small Mobile Robots

- Lower level – object avoidance
- There are finite state machines at this level for
 - sonar – emit sonar each second and if input is converted to polar coordinates, passing this map to collide and feelforce
 - collide – determine if anything is directly ahead of the robot and if so, send halt message to the forward finite state machine
 - feelforce – computes a simple repulsive value for any object detected by sonar and passes the computed repulsive force values to the runaway finite state machine
 - runaway – determines if any given repulsive force exceeds a threshold and if so, sends a signal to the turn finite state machine to turn the robot away from the given force
 - forward – drives the robot forward unless given a halt message

Middle Level

- This layer allows (or impels) the robot to wander around the environment:
 - wander – generates a random heading every 10 seconds to wander
 - avoid – combines the wander heading with the repulsive forces to suppress low level behavior of turn, forward and halt
 - in this way, the middle level, with a goal to wander, has some control over the lower level of obstacle avoidance but if turn or forward is currently being used, wander is ignored for the moment ensuring the robot's safety
 - control is in the form of inhibiting communication from below so that, if the robot is currently trying to wander somewhere, it ignores signals to turn around



Top Level

- This layer allows the robot to explore:
 - it looks for a distant place as a goal and can suppress the wander layer as the goal is more important
 - whenlook – the finite state machine that notices if the robot is moving or not, and if not, it starts up the freespace machine to find a place to move to while inhibiting the output of the wander machine from the lower level
 - pathplan – creates a path from the whenlook machine and also injects a direction into the avoid state machine to ensure that the given direction is not avoided (turned away from) by obstacles
 - integrate – this rectifies any problems with avoid by ensuring that if obstacles are found, the path only avoids them but continues along the path planned to reach the destination as discovered by whenlook

Analyzing This Robot

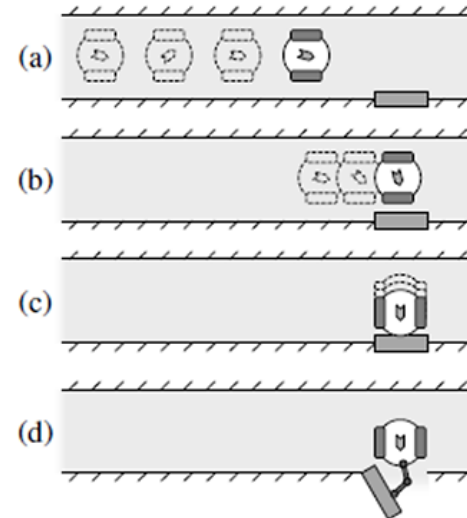
- This robot successfully maneuvers in the real world
 - with obstacles and even people trying to confuse or trick the robot
- Its goals are rudimentary – go somewhere or wander, and so it is unclear how successful this approach would be for a mobile robot with higher level goals and the need for priorities
 - The approach however is simplified making it easy to implement
 - no central representations
 - no awkward implementation (hundreds or thousands of rules)
 - no need for centralized communication or scheduling as with a blackboard architecture
 - no training/neural networks



Brooks Round II: Cardea

- A robot built out of a Segue
 - Contains a robotic arm to manipulate the environment (pushes doors open) and a camera for vision
 - Arm contains sensors to know if it is touching something
 - Robot contains “whiskers” along its base to see if it is about to hit anything
 - Robot uses a camera to track the floor
 - Looks for changes in pixel color/intensity to denote floor/wall boundary
 - Robot goal is to wander around and enter/open doors to “investigate” while not hitting people or objects

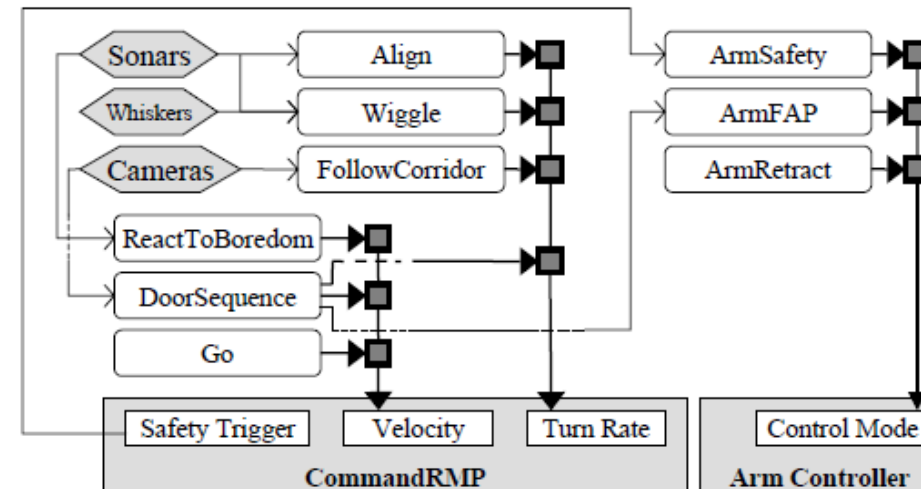
Cardea Detecting Doors



Phases of corridor navigation. In (a), the robot is proceeding down a corridor. The robot keeps away from the corridor walls using visual information from the lower fixed camera, subsumed by sonar and whiskers if necessary. In this phase, the robot moves quite rapidly (approximately 1 m/s). The active camera scans the walls, looking for a door. When it detects a potential door, the robot slows down (b). As the robot continues to navigate as normal, the active camera fixates the potential door. If it is not in the end sufficiently “door-like” after this examination, the robot speeds off again. Otherwise it halts parallel to the door, turns to face it, and then approaches it (c). Alignment with the door is done using sonar and whisker information. Once alignment is complete, the robot arm is extended to push the door open (d).



Cardea's Behavior



Like Brooks' smaller robots, Cardea has a simplistic set of behaviors based on current goal and sensor inputs.

- Align to a door way or corridor
- Change orientation or follow corridor
- Manipulate arm

Based on sonar, camera and whisker input and whether Cardea is currently interacting with a human that is interested or bored.

Brooks Round III: Cog

- The Cog robot is merely an upper torso and face shaped like a human.
 - Cog has
 - two arms with 12 joints each for 6 degrees of freedom per arm
 - two eyes (cameras), each of which can rotate independently of the other vertically and horizontally
 - vestibular system of 3 gyroscopes to coordinate motor control by indicating orientation
 - auditory system made up of two omni-directional microphones and an A/D board
 - tactile system on the robot arms with resistive force sensors to indicate touch
 - sensors in the various joints to determine current locations of all components

Rationale Behind Cog

- Brooks argues the following (much of these conclusions are based on psychological research).
 - Humans have a minimal internal representation when accomplishing normal tasks.
 - Humans have decentralized control.
 - Humans are not general purpose.
 - Humans learn reasoning, motor control and sensory interpretation through experience, gradually increasing their capabilities.
 - Humans have a reliance on social interaction.
 - Human intelligence is embodied, that is, we should not try to separate intelligence from a physical body with sensory input.

Cog's Capabilities

- Cog is capable of performing several human-like operations:
 - Eye movement for tracking and fixation
 - Head and neck orientation for tracking, target detection
 - Human face and eye detection to allow the eyes to find a human face and eyes and to track the motion of the face – identifies oval shapes and looks for changes in shading
 - Imitation of head nods and shakes
 - Motion detection and feature detection through skin color filtering and color saliency
 - Arm motion/withdrawal – it can use its arm to contact an object and withdraw from that object, and arm motions for playing with a slinky, using a crank, saw or swinging like a pendulum
 - Playing the drums to a beat by using its arms, vision and hearing

Brooks Round IV: Lazlo

- The robot is limited to just a human face:
 - The main intention of Lazlo is to learn from human facial gestures emotional states
 - They will add to Lazlo a face designed to have the same expressivity of a human face
 - Eyebrows and eyes
 - Mouth, lips, cheeks
 - Neck
 - They intend Lazlo to have the same basic expressions of emotional states at the level of a 5 or 6 year old child – such as the ability to smile or a frown or shake its head based on perceived emotional state.



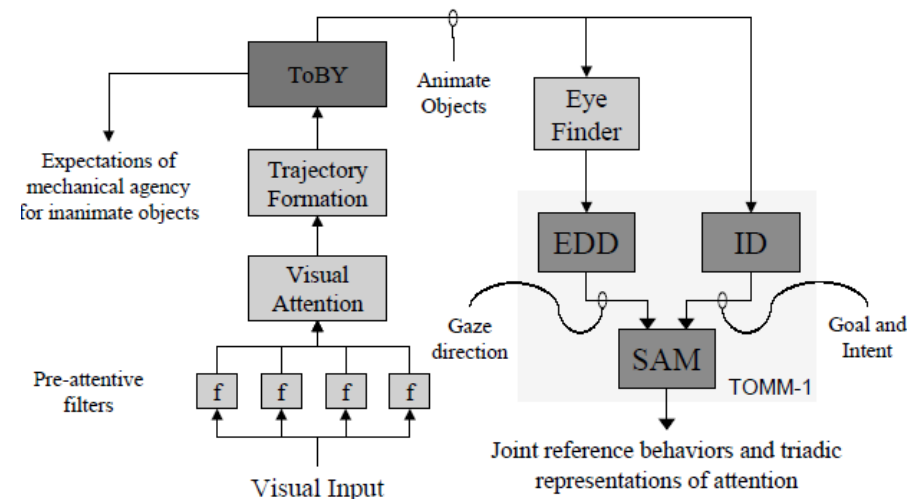
Brooks Round V: Meso

- Another on-going project is to study the biochemical subsystem of humans to mimic the energy metabolism of a human:
 - In this way, a robot might be able to better control its manipulators
 - This approach will (or is planned to) include
 - Rhythmic behaviors of motion (e.g. turning a crank)
 - Mimic human endurance (e.g., provide less energy when “tired”)
 - Determine states such as overexertion to lessen the amount of force applied
 - Better judge muscle movement to be more realistic (humanistic) in motion



Brooks Round VI: Theory of Body

- Model beliefs, goals, percepts of others.
- If a robot can have such belief states modeled, it might be able to respond more humanly in situations.
 - Theory of mind has been studied extensively in psychology, Brooks' group is concentrating on "theory of body"
 - At its simplest level, they are looking at distinguishing animate from inanimate objects
 - Animate stimuli to be tracked include eye direction detector, intentionality detector, shared attention
- They already have a start on this with Cog's ability to track eye and head movement.



Summary

- Androids? Long way away
 - Mimicking human walking is extremely challenging
 - Brooks' work has demonstrated the ability for a robot to learn to mimic certain human operations (eye movement, head movement, facial expressions)
- Human-level responses?
 - Steering, acceleration and braking control are adequate when terrain is not too difficult and when there is little to no traffic around
- Human-level reasoning?
 - Path planning and obstacle avoidance are acceptable
 - Mission planning is questionable
 - Failure handling and recovery are primitive in the current robots do not have the capability.
 - Robotic research explores many of the areas that AI investigates.

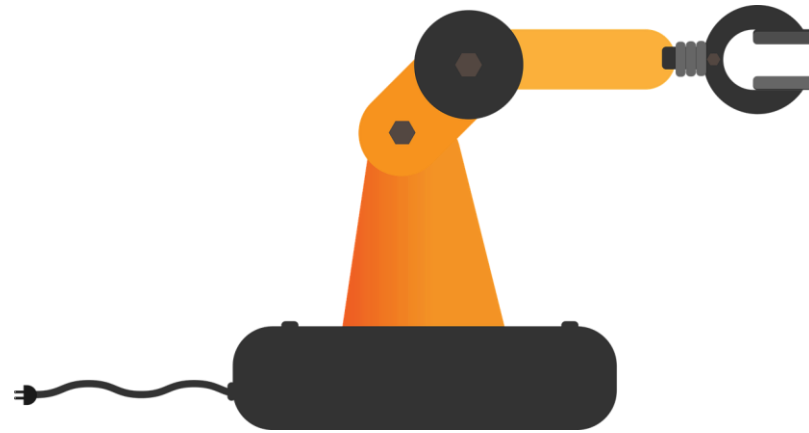
Review Questions

1. What approach should be taken for robotic research?
2. Should human and AVs be on the same roads at the same times?
3. How reliable can an AV be?
4. How reliable can AVs be in combat situations?
5. How will AVs certainly be useful in areas that are too dangerous or costly for humans to reach/explore?
6. What are the common types of robots used in industrial settings, and how do they contribute to manufacturing processes?
7. Explain the concept of robot software architectures. What are the main forms of software architectures, and how do they influence the performance of robots?
8. How does path planning contribute to the successful navigation of these vehicles?
9. How does ALVINN contribute to training algorithms, sensor interpretation, and obstacle avoidance in these vehicles?
10. What are the key capabilities and contributions of each round to the field of robotics (Small Mobile Robots, Cardea, Cog, Lazlo, Meso, Theory of Body)?



References

- Caitlin Barber (2015). Robots: An Introduction A robot can be defined as a computer controlled machine with some degrees of freedom – that is, the ability to move about in its.
- Rubio, F., Valero, F., & Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2), 1729881419839596.
- Kabir, H., Tham, M. L., & Chang, Y. C. (2023). Internet of robotic things for mobile robots: concepts, technologies, challenges, applications, and future directions. *Digital Communications and Networks*.
- Sanchez-Ibanez, J. R., Perez-del-Pulgar, C. J., & García-Cerezo, A. (2021). Path planning for autonomous mobile robots: A review. *Sensors*, 21(23), 7898.
- Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.
- Rubio, F., Valero, F., & Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2), 1729881419839596.



Lecture 06: Robotic and Automation for Industrial

Objectives

- Promote the widespread adoption of robotics and automation to improve efficiency, reduce costs, and enhance safety across various industries.
- Maximize the utilization of robots to handle dangerous and repetitive tasks, thereby freeing up human workers for more creative and complex roles.
- Foster innovation in robotics and automation technology to address specific needs in areas like industrial manufacturing, entertainment, space exploration, research, and more.
- Optimize human-robot collaboration and interface design for seamless integration of robots into various fields, ensuring safety and efficiency.
- Drive research and development efforts to advance robotic capabilities in terms of mobility, manipulation, programming, sensors, and perception, with a focus on achieving better results in challenging environments and applications.

Outlines

- What is Robotic and Automation for Industrial?
- Why Use Robots?
- What are the 7 broad areas often Using Robots?
- Dangerous Environments
- Industrial Robots
- Entertainment & Leisure Robots
- Space Robots
- Research Robots
- Underwater Robots
- Medical Robots
- Future Medical Robots
- Similarities and Differences
- Robotics and First Aid Basics
- Other Safety Precautions
- Safety at Competitions
- Major Fields of Industrial Robotics
- Human-Robotic Interface
- Mobility or Locomotion
- Manipulation
- Programming
- Sensors and Perception
- Forestry Automation
- Mining Teleoperation
- Robotic Welding
- Benefits of Robotic Welding
- Machine Loading
- Sequential Machine Loading
- Robots for Forging and Die-Casting
- Spray Painting
- Fabrication
- Robotic Assembly
- Engine Assembly
- Electrical or Electronic Machine Assembly
- Unusual Applications
- Forestry Teleoperation
- Mining Autonomous Machines
- Forestry Automation
- Future Forestry Technology Concepts
- Potential problems and Human Factors



What is Robotic and Automation for Industrial?

- Robotic and automation technologies for industrial applications refer to the use of robots, machines, and computer-controlled systems to perform tasks in various industrial settings.
- These technologies are designed to increase productivity, efficiency, safety, and consistency in manufacturing and other industrial processes.
- There are key aspects of robotic and automation in the industrial context:
 - Robotics
 - Automation
 - Types of Industrial Automation
 - Manufacturing
 - Logistics
 - Pharmaceuticals
 - Agriculture
 - Energy
 - Medicals

What is Robotic and Automation for Industrial? (Cont.)

- **Robotics:** Industrial robots are mechanical devices equipped with sensors, actuators, and computer control systems. They can perform a wide range of tasks, including welding, material handling, assembly, painting, and inspection. These robots can be programmed to work autonomously or in collaboration with human operators.
- **Automation:** Automation involves the use of machines and control systems to execute tasks with minimal human intervention. This can include automated conveyor systems, industrial control systems, and robotic process automation (RPA) for tasks like data entry and routine office work.

What is Robotic and Automation for Industrial? (Cont.)

Types of Industrial Automation

- **Fixed Automation:** Designed for high-volume production of a specific product or part. It is not easily reconfigurable for new products.
- **Flexible Automation:** More adaptable and can be reprogrammed or reconfigured for different tasks or products.
- **Programmable Automation:** Combines the features of fixed and flexible automation. It can be reprogrammed for different product variations within a product line.
- **CNC (Computer Numerical Control):** Used in machining processes, CNC machines are controlled by computer programs to precisely shape and cut materials.

What is Robotic and Automation for Industrial? (Cont.)

Benefits of Robotic and Automation in Industrial Settings

- **Increased productivity:** Machines and robots can work around the clock, reducing downtime and increasing output.
- **Improved quality:** Automation can produce products with consistent quality and precision.
- **Enhanced safety:** Robots can perform dangerous tasks, reducing the risk to human workers.
- **Cost savings:** Automation can reduce labor costs and improve resource utilization.
- **Reduced cycle time:** Automation can lead to faster production and quicker response to market demands.

What is Robotic and Automation for Industrial? (Cont.)

Applications of Robotic and Automation in Industrial Settings

- **Manufacturing:** Automation and robotics are commonly used in automotive, electronics, aerospace, and food industries for tasks such as welding, assembly, and material handling.
- **Logistics and warehousing:** Automated guided vehicles (AGVs) and robotic systems are used for materials handling and order fulfillment.
- **Pharmaceuticals:** Automation is used in drug manufacturing and quality control.
- **Agriculture:** Robots are used for planting, harvesting, and crop inspection.
- **Energy:** Automation is used in power generation and distribution.

What is Robotic and Automation for Industrial? (Cont.)

Challenges of Robotic and Automation in Industrial Settings

- Implementing robotic and automation systems may require significant capital investment and expertise.
- Maintenance and reprogramming can also be complex.
- Additionally, concerns about job displacement and cybersecurity are important considerations.
- Robotic and automation technologies continue to advance, offering new opportunities for industries to improve efficiency and competitiveness while addressing the challenges associated with their implementation.

Why Use Robots?

- Most robots are designed to be a helping hand. They help people with tasks that would be difficult, dirty, dangerous, or dull for a human
- Can carry very heavy loads
- Do not get bored doing the same job over and over again, 24 hours a day.
- Have been proven to increase productivity.



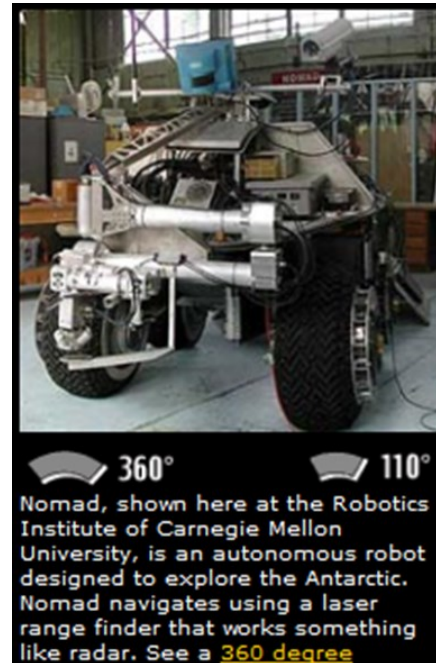
What are the 7 broad areas often Using Robots?

1. Dangerous environment
2. Industrial
3. Entertainment and Leisure
4. Space
5. Research
6. Underwater
7. Medical

Dangerous Environments



Robovolc System (Robovolc)



Autonomous Robot(Troop323bsa)

- Carrying out bomb disposal
- Collecting data from volcanoes
- Exploring
- Military
- Rescue

Dangerous Environments (Cont.)

- Predator flown via remote control by airmen on the ground flies up to 25,000 feet.
- Used to conduct reconnaissance and attack operations; takes real-time photos of troop movements on ground.



Air Force Predator (Troop323bsa)



Designed for Firefighter Rescue (Troop323bsa)



Soldier Rescue(Troop323bsa)

Industrial Robots



Robotic Arm (PNG Egg)

- A typical industrial robot is a versatile robotic system designed for various tasks in manufacturing and industrial settings.
- A typical industrial robot is a robot arm with several independent joints and you will see them welding, painting and handling heavy materials.
- Pick and place robots can move products from a conveyor belt to package them at very quick speeds.

Industrial Robots (Cont.)



Automated Guided Vehicle (AGV)(Daifuku)

An AGV is a mobile robot that follows markers or wires in the floor or uses vision or lasers.

- An example of a mobile robot that is in common use today is the automated guided vehicle (AGV).
- AGVs are autonomous or semi-autonomous vehicles that are used for the transportation of materials, goods, and products within a controlled environment, such as a manufacturing facility, warehouse, or distribution center.

Entertainment & Leisure Robots



C3PO and R2D2 (Salika)

- Entertainment and leisure robots are robots designed and developed to provide entertainment, companionship, and leisure activities to humans.
- These robots can serve various purposes, from providing companionship to performing tasks related to entertainment and recreation.
- There are some examples of entertainment and leisure robots: robot companion, robotic pets, entertainment robots, gaming robots, theme park robots, robotic toys, robotic assistants, and telepresence robots.



Entertainment & Leisure Robots (Cont.)

- **Robot Companions:** Companion robots are designed to interact with humans and provide emotional support and companionship.
- **Robotic Pets:** Robotic pets are designed to mimic the behavior of real animals, such as cats, dogs, or even dinosaurs.
- **Entertainment Robots:** These robots are created for the sole purpose of entertaining people.
- **Gaming Robots:** Some robots are designed specifically for playing games.
- **Theme Park Robots:** In theme parks and entertainment venues, robots are used to provide unique experiences.



Entertainment & Leisure Robots (Cont.)

- **Robotic Toys:** Robotic toys are designed for children and can include programmable robots, interactive plush toys, and educational robots that teach coding and STEM concepts.
- **Robotic Assistants:** Some robots are designed to assist with leisure activities, such as serving drinks, preparing food, or even giving massages.
- **Telepresence Robots:** Telepresence robots allow people to virtually attend events, explore new places, or interact with others remotely.

Space Robots



The International Space Stations's Two-Armed Robot
(Commons.wikimedia.org)

- Space robots, also known as **space exploration robots** or **robotic spacecraft**, are specialized robots designed for various tasks in space exploration and research.
- NASA is constantly developing and producing robots which can perform maintenance in space – especially on its International Space Station.

Space Robots (Cont.)



Robonaut 2 (Tharadhol)

- Humanoid robot joined crew of International Space Station.
- R2 is able to use the same tools station crew members use.
- In the future, the greatest benefits of humanoid robots in space may be as assistants or stand-in for astronauts during spacewalks or for tasks too difficult or dangerous for humans.

Research Robots

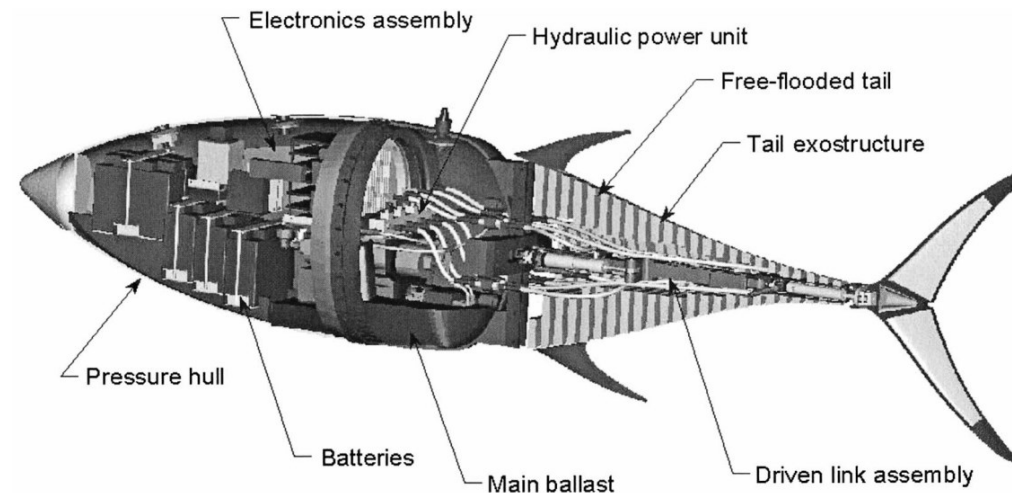


Asimo(Dga)

- One important area of robotics research is to enable the robot to cope with its environment
- Honda is the company that is spending a great deal of money developing research robots, such as the Asimo show on left.
- ASIMO moves like we do and could be useful to help the elderly or people in wheelchairs. It can answer the door, pick up the phone or get a cup of tea.

Underwater Robots

- Underwater robots are often remote-controlled vehicles with thrusters for maneuvering and robot arms for grabbing.
- They are particularly useful in the oil industry for welding and valve maintenance on oilrigs.



Robotuna used for Exploration (Anderson & Chhabra, 2002)

Medical Robots

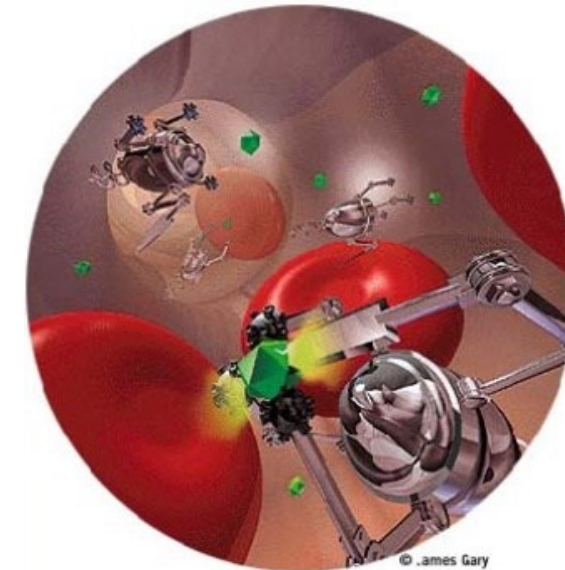
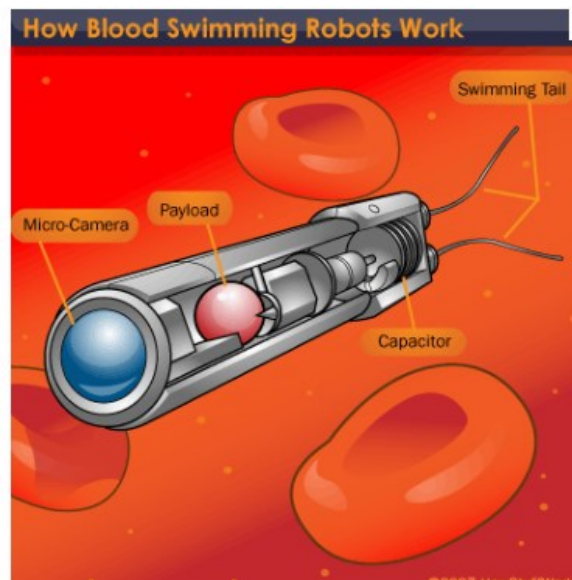
- Medical robots are robotic systems designed and used in various healthcare applications to assist healthcare professionals, improve patient care, and enhance the efficiency and precision of medical procedures.
- These robots can range from simple, stationary devices to highly complex, autonomous machines.



Revo-I Robot (Almujalhem & Rha, 2020)

Future Medical Robots

Scientists believe that tiny robots (called “nanorobots”) will be developed which will be used in patients’ bloodstreams to cure illness.



Blood Swimming Robots Work (Parmar et al., 2010) Bacteria-Hunter Nanobot (Parmar et al., 2010)

Nanomedicina (CN EAD)

Similarities and Differences

Remote-Control Devices	Telerobots	Autonomous robots
Physical link between controller and object being controlled	No physical connection to the remotely operated system (i.e. WiFi). Requires remote sensory feedback	Makes decisions based on programming and sensory feedback. Controlled by an internal computer.
Human operator is controlling the device without physically touching it	Human operator is controlling the device without physically touching it	Work for an extended period without human intervention
Short Distance	Any Distance	Any Distance
	Requires remote camera or interactive component to sense what is happening on the remote end of the system	Gains information and about the surrounding environment and adapt to changes.

Robotics and First Aid Basics

Prevention of Injuries:

- Dress appropriately and wear safety protection, such as goggles and ear plugs. Do not wear loose fitting clothing, hanging jewelry, long hair, or anything else that could get caught in equipment.
- Work in well-ventilated areas
- Do not drink or eat in the work area.
- Have a fire extinguisher nearby
- Work under proper supervision as required.



Robotics and First Aid Basics (Cont.)

First Aid:

- **Minor cuts & scrapes:** flush clean water for ≥ 5 minutes or until foreign matter is out. Apply antibiotic ointment (if no allergies), cover with dry sterile bandages.
- **Chemical burns:** quickly brush off gloved hand as much of chemical as possible. Flush area tap water.
- **1st degree or minor burns:** hold under cold water or apply cool wet compresses until pain eases. Cover loosely sterile gauze and bandages.
- **Foreign object in eye:** do not rub; blink eyes for tears flush out. If that doesn't work, flush clean running water or from bottle.



Other Safety Precautions

- Most serious injuries are when a person gets too close to the machinery. Stay outside the 3 foot operating radius when robotics are in use. Some machines have sensors to detect human presence and automatically stop operating.
- Before beginning to build, fix, or work on moving parts, make sure the energy sources are all disconnected.

Safety at Competitions

- Always wear eye protection, such as safety glasses.
- Wear ear protection since the noise is extreme at these events.
- Dress appropriately and apply basic first aid techniques if injury occurs.

Major Fields of Industrial Robotics

1. Human-robotic interface
2. Mobility or Locomotion
3. Manipulation
4. Programming
5. Sensors and Perception

Human-Robotic Interface

- How does the robot and operator communicate with each other?
- The Interface is HOW the human operator controls the robot.

Examples

- a) Controller for a Xbox or Wii game
- b) Computer keyboard used to program a robot.

Mobility or Locomotion

- How does the robot move?
- Some only need to move arms or grippers
- Others need to be completely mobile and move from place to place

Examples

- a) A robotic arm rotates and stops at a specific position to paint car parts
- b) An operator directs a Sedway personal transporter to move from one location to another.

Manipulation

- How does the robot physically handle objects?

Examples

- a) Mechanical claw picks up & transports objects.
- b) Robotic arm w/ mechanical grippers load candy into boxes.
- c) Robotic hand welds a seam on a car and paints the car.



Programming

- How the operator commands the robot to do what needs to be done.
- Software is written in the computer's language for what the robot understands.
- Some advanced program allows for the robot to learn and adapt to changes in the environment.

Sensors and Perception

- Robots rely on sensors to get information about their surroundings to determine where it is and what it should do next.

Examples

- a) Ultrasonic sensors determines the distance of objects by emitting sound pulses (too high humans to hear), and then measuring the time delay to detect the sound pulse echo. Used in submarine navigation since it works in the dark.



Sensors and Perception (Cont.)

Examples

- b) Light sensors can be used for simple navigation by allowing a robot to follow a line, such as AGVs. Other robots navigate using infrared light (the same invisible light used in your TV remote control).
- c) Touch sensors help otherwise blind robots with navigation: feelers, contact switches, bump sensors, all let a robot know when it has made contact with walls or objects. i.e. Robotic vacuums
- d) Radio signal sensors let robots communicate with each other at a distance. Electromagnetic sensors are used by robotic lawnmowers to stay within the bounds of the yard.

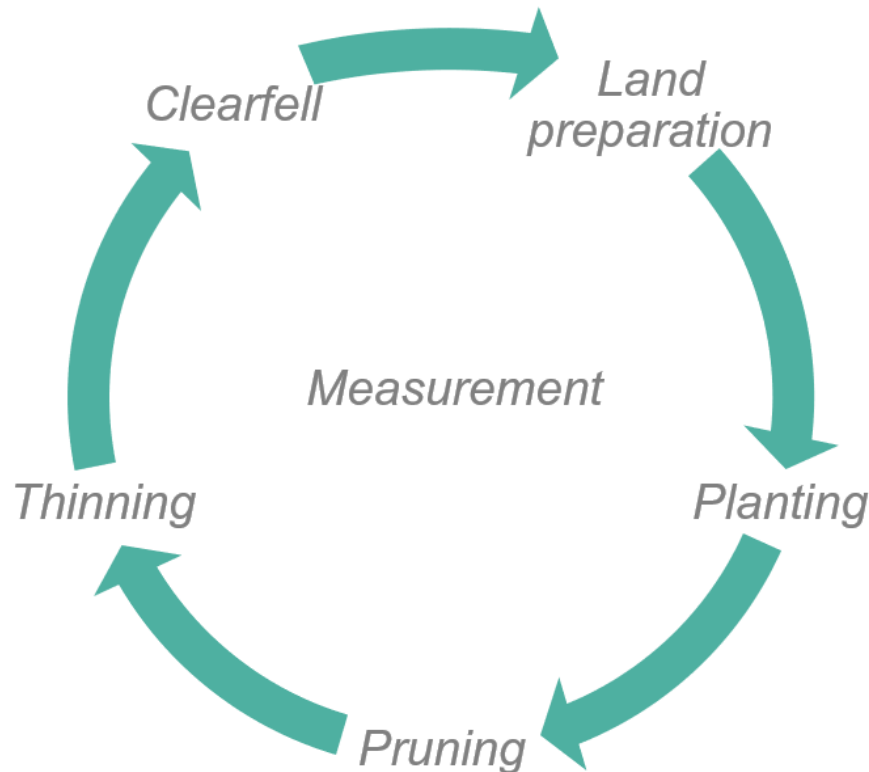
Forestry Automation



The Plusjack Walking Harvester Full-Scale Prototype (Ismoilov, 2016)

- Forestry automation involves the use of various technologies, machinery, and automated systems to streamline and improve various tasks and processes in the forestry industry.
- Automation in forestry aims to increase efficiency, reduce labor costs, enhance safety, and promote sustainable forest management.

Forestry Automation (Cont.)



Forestry Cycle (Nzif.Org.Nz)

- The forestry cycle, also known as the **forest management cycle** or the **forestry process**, is a systematic approach to managing forests sustainably while balancing ecological, economic, and social objectives.
- The forestry cycle typically involves a series of stages or activities that are repeated over time to ensure the long-term health and productivity of forest ecosystems.

Forestry Automation (Cont.)

- Motor-manual
- Operators exposed – hit by trees and rolling logs
 - Cycle time of minutes
- Mechanised
 - Operators in machine
 - cabs
 - Intense – cycle time of
 - seconds
 - Overuse injuries



Motor-Manual (Nzif.Org.Nz)



Mechanised (Nzif.Org.Nz)

Forestry Automation (Cont.)

- “Faster and faster work – operator becomes the bottle neck” *Ola Ringdahl, PhD May 2011, Umeå, Sweden*
- Automation introduced to reduce workload and improve productivity.
- Boom tip control is a part of forestry automation.



Automation(Nzif.Org.Nz)

Mining Teleoperation

- Remote operator
- Radio link
- Out of sight of machine



Remote Operator(Nzif.Org.Nz)



Out of Sight of Machine(Nzif.Org.Nz)

Mining Teleoperation (Cont.)

- Roy Hill Remote Operations
 - Centre – Perth
- 400 operators
- 15 mines, 31 pits, 4 ports,
 - 1600km railway
- 1300 km from Pilbara mines



400 Operators (Nzif.Org.Nz)

Robotic Welding

- Robotic Welding are a type of industrial robot specifically designed for performing welding tasks.
- They are widely used in various industries, including automotive, aerospace, construction, and manufacturing, to improve efficiency, consistency, and precision in welding operations.
- These robots offer several advantages over manual welding, such as increased productivity, reduced labor costs, and improved safety.



Robotic Welding (Standard Bots, 2023)



Benefits of Robotic Welding

- Robotic welding systems offer numerous advantages, including improved precision, productivity, safety, and cost savings, making them a valuable asset for companies looking to optimize their welding processes and remain competitive in their respective industries.
- Arc-on-time concept:
 - Manual welding (arc-on-time percentage is 20-30%)
 - Robotic welding (production level of 4 welders)
- “Weave Patterns” for weld paths:
 - Weave patterns Zigzag path
 - A robot can be programmed for weave patterns

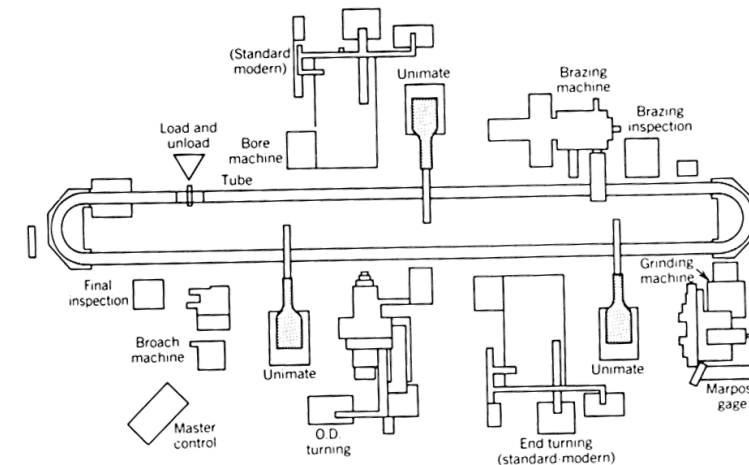
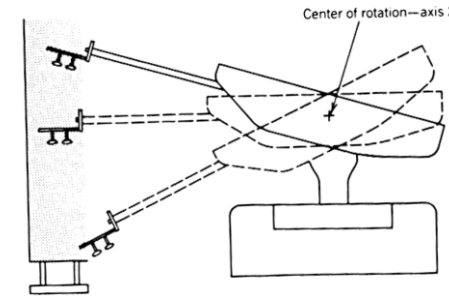
Machine Loading

Why robots to be used for machine loading?

- Safety and relief from handling heavy equipment
- Risk of amputations while feeding punch press by hand
- To eliminate production slowdowns
- To achieve high operating speed
- Small clearances make manual feeding a tricky job
- Reduction of scrap is a side benefit

Machine Loading (Cont.)

- Positioning problem may occur in machine loading.
- A robot has to pick and place on different elevation levels where robot is an axis-limit, polar configuration robot.
- Remedies
 - Work envelop
 - Suction cup pickup devices
 - Racks design
 - End effector with greater flexibility



Machine Loading (Cont.)

Problem:

The robot machine loading / unloading system diagrammed in figure has an eight seconds cycle time and a daily two shift production level of over 28,000 workpieces. Do the four milling machines perform sequential operations upon each workpieces in series, or do all four milling machines perform the complete machining cycle in parallel with each other – that is, is each part processed completely by only one machine, not all four?

Machine Loading (Cont.)

Series operation:

Production rate / day

$$= 1 \text{ part} / 8 \text{ sec} \times 60 \text{ sec/min} \times 60 \text{ min/hr} \times 16 \text{ hr/day}$$

$$= 7200 \text{ parts per day (two shifts)}$$

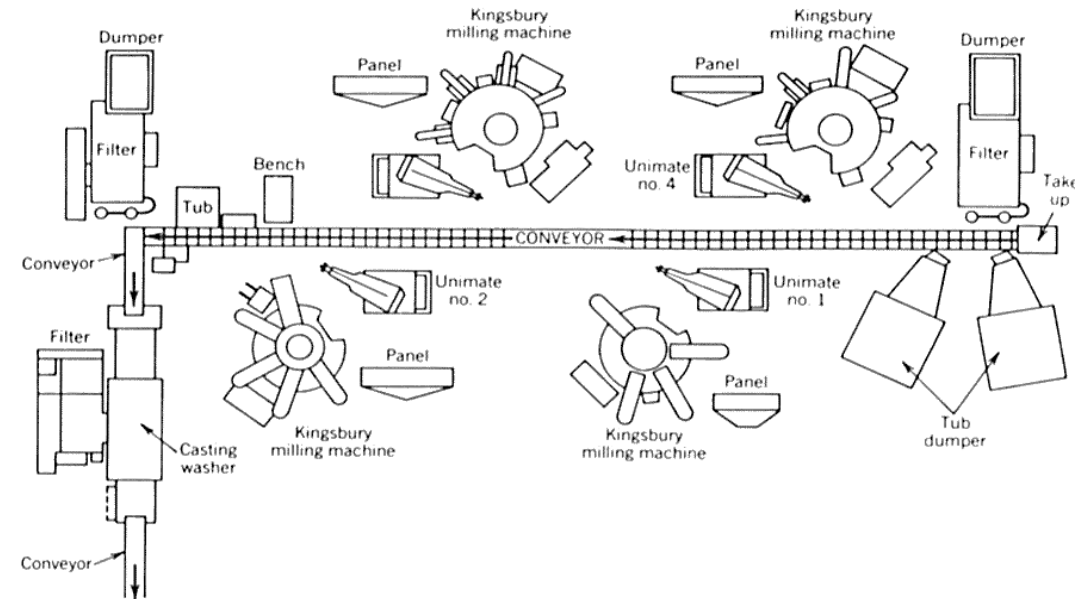
Parallel Operation:

Production rate / day

$$= 1 \text{ part} / 8 \text{ sec} \times 60 \text{ sec/min} \times 60 \text{ min/hr} \times 16 \text{ hr/day} \times 4 \text{ machines}$$

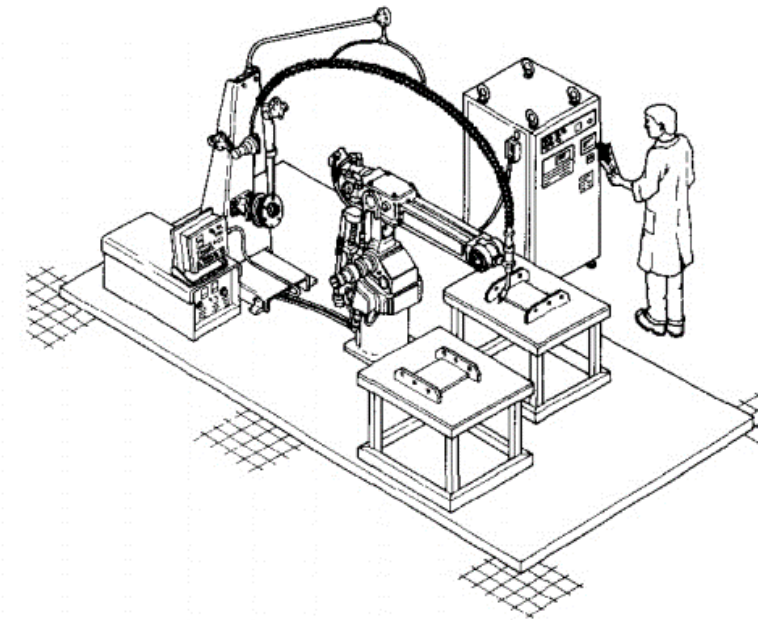
$$= 28,800 \text{ parts per day (two shifts)}$$

=> system runs on parallel basis



Sequential Machine Loading

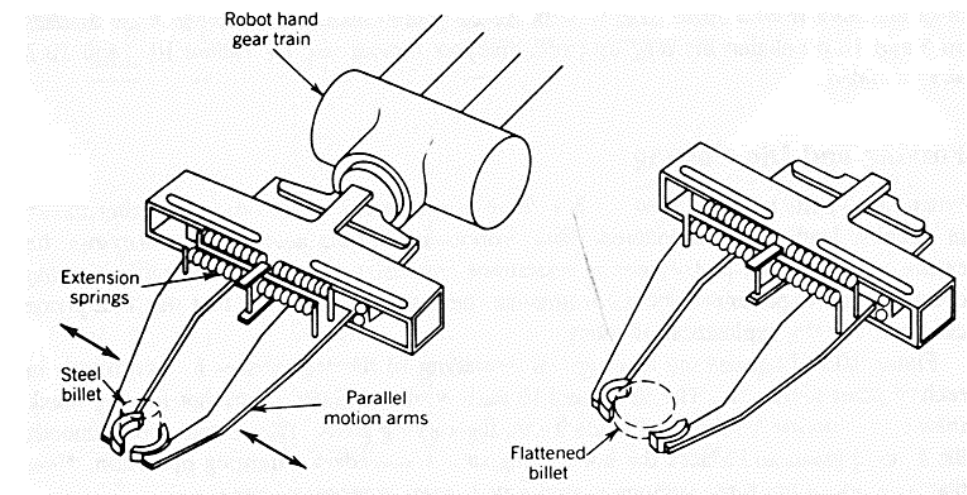
- Single robot can load or unload several machines
- One robot can load and unload three machines with help of two conveyors.
- 60% increase in production level is observed.
- Double handed grippers are of little value
 - We intend to work on single work piece sequentially
 - When one job is complete on one machine, the workpiece would be unloaded and loaded to the 2nd machine and so on.
 - There is no room for semi-finished parts in the workcell.



Robot Arc Welding Cell (Brain Kart)

Robots for Forging and Die-Casting

- Why robots for forging and die-casting?
 - Piece parts are hot
 - Environmental heat and noise
 - Risk of amputation while feeding
- Normally, a robot for forging is programmed to dip its hand into a cooling bath at appropriate intervals.
- Gripper must be capable to handle shape changes during forging.



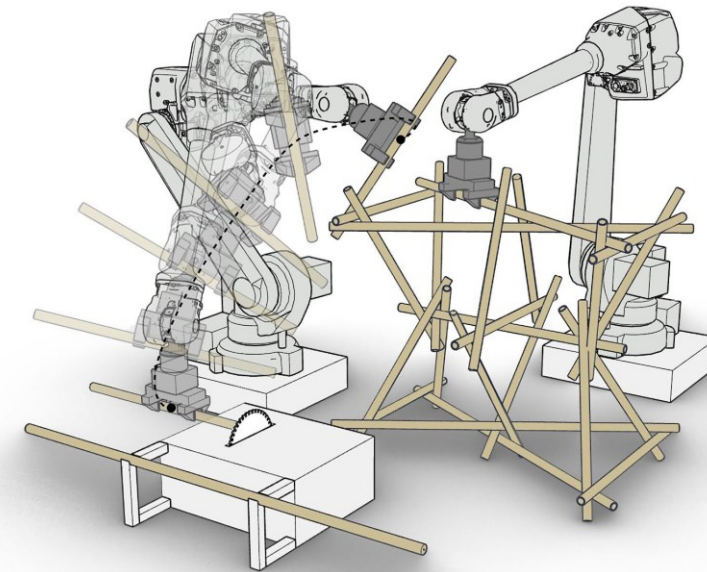


Spray Painting

- Environment in spray painting booths is very difficult to maintain according to safety and health standards.
- A level of consistency is difficult to achieve when human workers are employed.
- Line tracking capability for continuous feed conveyors.
- A robot can be taught (only once), by a skilled spray painter.
- Failures can be expected, so a manual touchup area is sometimes required.
- Equipment utilization is very important factor to consider.

Fabrication

- Welding
- Drilling
 - For small workpieces (PCB Drilling, fixtured parts)
 - For large workpieces (aircraft parts)
 - Drill must be oriented perpendicular to the surface.
 - A human achieves this by visual determination (source of error).
 - A robot with deterministic approach can replace a human worker.



Robotic Fabrication with the COMPAS
 Framework online workshop (Rhino Ceros)

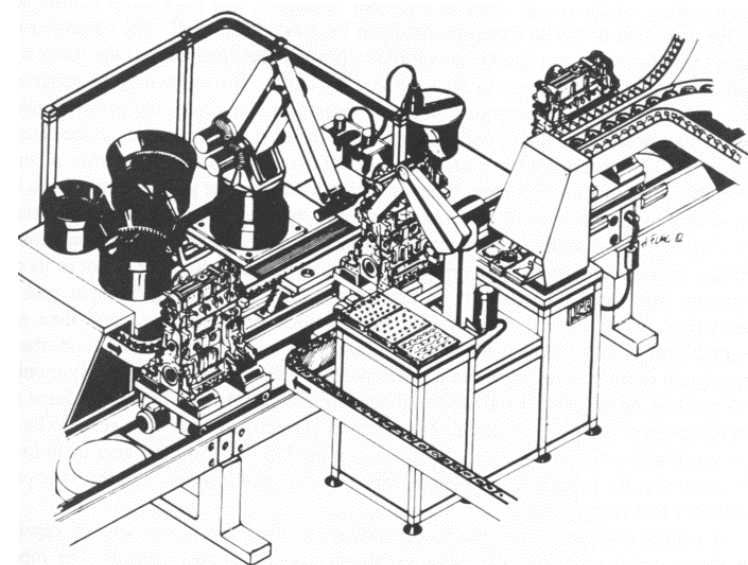


Robotic Assembly

- 3D criteria is useless for robotic assembly applications.
- Assembly requires precision, repeatability, variety of motions, sophisticated gripper devices, and sometimes compound gripper mechanisms.
- Basis of decision for assembly robot:
 - Save labor costs
 - Repetitive job is a boring job
 - Fast and efficient work requirement
 - Accuracy
 - Eliminate omissions or substitutions
 - Eliminate intentional omissions

Engine Assembly

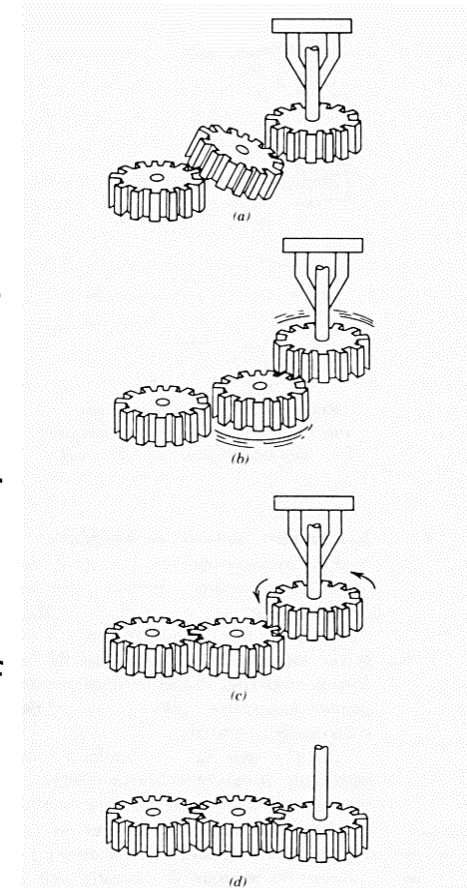
- Engine blocks travel on a pallet transfer mechanism.
- Robots facilitated by auxiliary conveyors for subassemblies and vibratory bowls for screws and bolts.
- Different grippers are provided in a magazine, mainly to do a variety of jobs.



Robotic engine assembly cell (Maci, 2002)

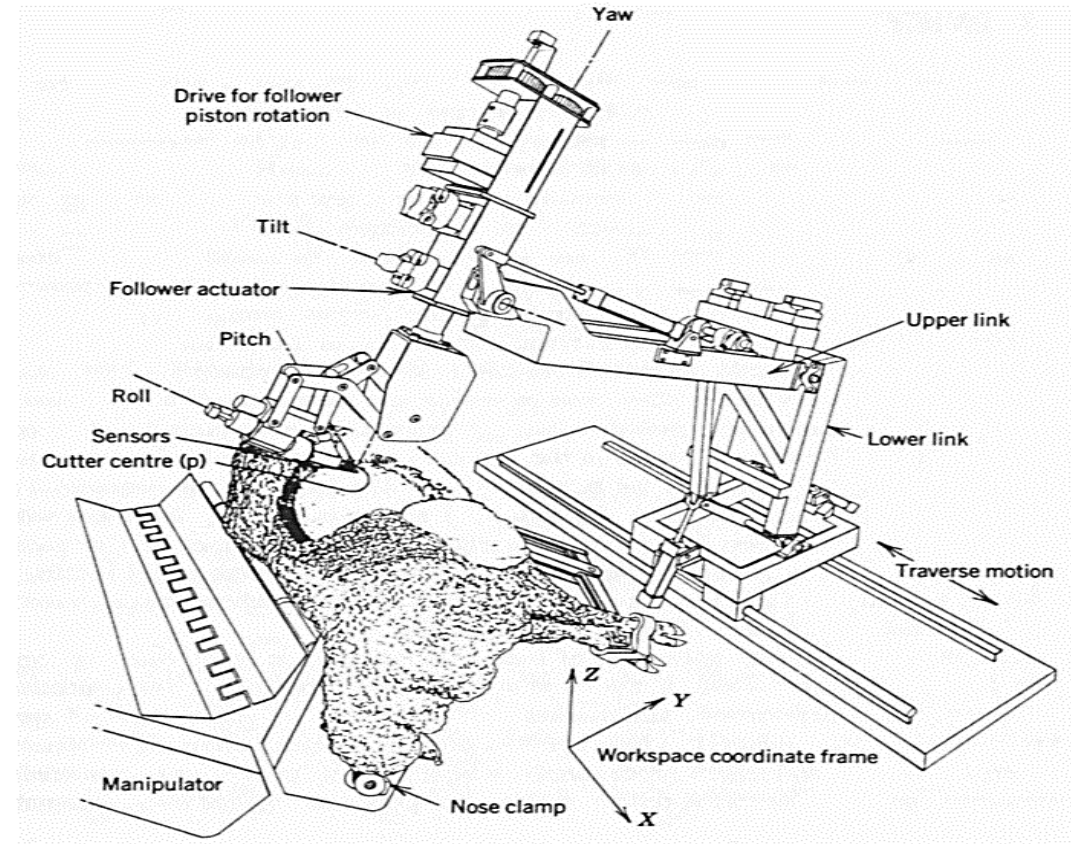
Electrical or Electronic Machine Assembly

- Indexed, palletized carrier type transfer mechanisms
- Tactile sensing
- Visual feedback loops
 - Since assembly operations require more “feel” for the objects being handled
 - Defects can also be detected
- Even more sophisticated / flexible robots are required for flexible manufacturing systems
- General Assembly Robots:
 - It can sense and assemble virtually any thing of appropriate size.
 - When combined with judgment, these robots can be used for humanoid robots.



Unusual Applications

- Sheep shearing robots
- Robots in Construction
- Robots for Hazardous Material cleaning
- Sojourner Rover
- Nano-robots



Sheep Shearing Robots (Issuu)

Forestry Teleoperation

Limited field of view for the operator

- Orientation issues – am I on a slope?
- Camera viewpoint so operator can understand where machine is poor depth of field perception, which makes it difficult to see hollows and rises in the terrain



Forest Growers Research (Mpi.Govt.Nz)

Mining Autonomous Machines

- Autonomous haul trucks:
 - 10 to 15 % decrease in fuel consumption
 - 15 % decrease in tyre wear
- Autonomous long-distance haul trains
- Automated drilling and tunnel-boring systems



Hans-Holger Brauckmann - Sandvik DD530 (Flickr)

Forestry Automation



Remote Controlled (Stuff, 2012)

- Log making and bucking
- Stem location and grappling



Sustainable Timber Harvesting & Cable (GMT Logging)

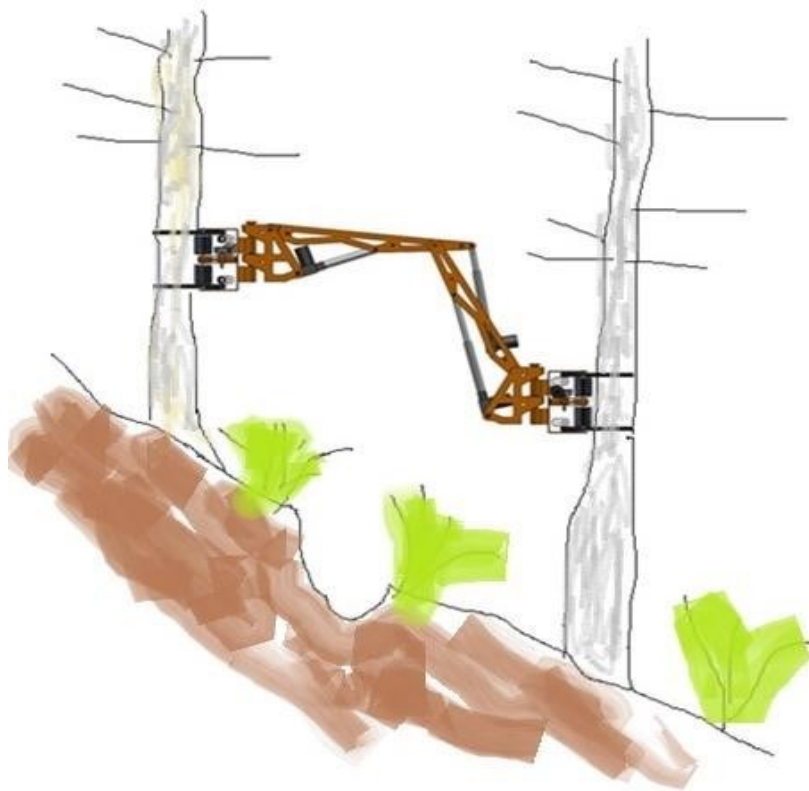
Future Forestry Technology Concepts

- The field of forestry is constantly evolving, driven by the need to sustainably manage forests and address environmental challenges such as deforestation, climate change, and biodiversity loss.
- Future forestry technology concepts are likely to incorporate innovative solutions to these issues.



Tree Harvester on Behance (Behance)

Future Forestry Technology Concepts (Cont.)



- Forestry machines with no cab refer to specialized equipment used in forestry operations that do not have an enclosed operator's cab.
- These machines are designed for specific tasks in the forestry industry and are operated remotely or with a separate cab-equipped vehicle.
- They are typically used in applications where there is less need for operator comfort or safety, and the primary focus is on efficient and specialized forest-related tasks.

Forestry Machines with No Cab (Frank Lloyd Wright)

Potential problems and Human Factors

- Forestry is an outdoor occupation
- Sedentary office task
- Disconnect from reality
- Loss of situational awareness
- No travel
- Interaction with work mates



Former Drone Operators (The Intercept, 2015)

Potential problems and Human Factors (Cont.)

Robotics and automation benefits on the follows:

- Reduce injury
- Appear to be inevitable
- Introduce new problems
- Open employment to wider range of people
- Reduce operator jobs
- Create new technical jobs
- Enable harvest of currently uneconomic forests
- Automation is a difficult technical problem



Summary

- Robotics and automation in industry are necessary for improving productivity, reducing costs, ensuring product quality, and enhancing workplace safety.
- The adoption of robots is primarily driven by their ability to automate repetitive and hazardous tasks, leading to increased efficiency and safety, while enabling continuous operations.
- Robots have diverse applications across seven broad areas, including hazardous environments, industrial manufacturing, entertainment, space exploration, research, underwater operations, and healthcare.
- Within industrial robotics, the research focus encompasses human-robot interaction, mobility, manipulation, programming, and sensor technology, aiming to optimize robot performance in a variety of applications.
- In academic discourse, the use of automation in sectors such as forestry, mining, and manufacturing is explored to improve worker safety, minimize environmental impact, and innovate for sustainable practices and future technology concepts.



Review Questions

1. How do robotics and automation contribute to improving efficiency and safety in industrial settings, and what are the key factors in their successful implementation?
2. What are the primary reasons for utilizing robots in various industries, and how do they impact productivity, precision, and worker roles?
3. Can you elaborate on the seven broad areas in which robots are commonly employed and provide examples of specific applications in each domain?
4. In the field of industrial robotics, what are the key research areas, and how do advancements in human-robot interfaces, mobility, manipulation, programming, and sensor technology impact industrial processes?
5. How does automation in fields like forestry, mining, and manufacturing address safety concerns, environmental impact, and sustainability, and what are some potential future technology concepts in these sectors?

References

- Badge, M. (2016). Robotics. Marilyn Farrand, Troop 148 - Charlotte, NC
- TEMPUS IV Project: 158644 – JPCR (2012). Robot Application in Manufacturing. Development of Regional Interdisciplinary Mechatronic Studies – DRIMS.
- Parker, R., Clinton, P., Bayne, K., & Hooper, B. (2017). Forestry automation and robotics. Scion, the New Zealand Forest Research Institute Limited.
- Anderson, J. M., & Chhabra, N. K. (2002). Maneuvering and stability performance of a robotic tuna. Integrative and comparative biology, 42(1), 118-126.
- Almujaalhem, A., & Rha, K. H. (2020). Surgical robotic systems: What we have now? A urological perspective. BJUI compass, 1(5), 152-159.
- Ringdahl, O. (2011). Automation in forestry: development of unmanned forwarders (Doctoral dissertation, Institutionen för datavetenskap, Umeå Universitet).
- Ismoilov, A. (2016). Suspended forestry machines for sustainable forestry (Doctoral dissertation, KTH Royal Institute of Technology).
- Parmar, D. R., Soni, J. P., Patel, A. D., & Sen, D. (2010). Nanorobotics in advances in pharmaceutical sciences. Int J Drug Dev and Res, 2, 247-56.
- Maci, C. A. (2002). Contribuții la construcția și utilizarea digitizoarelor 3D în proiectarea asistată de calculator și programarea roboților industriali (Doctoral dissertation, Universitatea "Politehnica" din Timișoara, Facultatea de Mecanică).



Lecture 07: Automation Simulation

Objectives

- Gain insight into the fundamental components of automation simulation and their applications across industries.
- Assess the features and capabilities of Gazebo as a simulation tool.
- Investigate how simulation contributes to the integration of robotics and automation in industrial processes.
- Develop proficiency in using Gazebo for simulating a variety of robotic and automation scenarios.
- Improve skills in creating realistic and efficient simulation models using Gazebo.

Outlines

- Automation Simulation
- Components of Automation Simulation
- Applications of Automation Simulation
- Simulation Models
- Simulation Software
- Dynamic Simulation
- Robotics and Automation in Industry
- Simulating Application
- What is Gazebo?
- Gazebo Distribution
- Gazebo Architecture
- SDF vs URDF
- Gazebo Features
- Gazebo Scheme
- Gazebo Modeling
- Gazebo Simulation
- Gazebo Control

Automation Simulation

- Automation simulation refers to the process of creating a virtual model or representation of an automated system to simulate its behavior and performance in a computerized environment.
- This simulation can be applied to various types of automated systems, including manufacturing processes, industrial equipment, robotics, and even complex business processes.
- Automation simulation often involves the use of specialized software that allows users to model the physical components and processes of the automated system.



Automation Simulation (Cont.)

The main purposes of automation simulation include:

- **Design and Validation:** Simulating an automated system allows engineers and designers to test and validate the design before physical implementation.
- **Optimization:** Simulation enables the optimization of automated processes by experimenting with different parameters, configurations, and scenarios. This can lead to improved efficiency, reduced costs, and enhanced overall performance.
- **Training:** Automation simulation can be used for training purposes, allowing operators and technicians to familiarize themselves with the automated system in a risk-free virtual environment.

Automation Simulation (Cont.)

- **Troubleshooting:** Simulations provide a platform for troubleshooting and identifying potential problems that may arise during the operation of an automated system. This allows for preemptive measures to be taken to prevent issues in real-world scenarios.
- **Cost Reduction:** By identifying and addressing issues early in the design phase, automation simulation can contribute to cost reduction. It minimizes the need for expensive modifications or corrections after the physical system is already in place.
- **Prototyping:** Simulation serves as a virtual prototype, enabling engineers and stakeholders to visualize the system's behavior and make informed decisions about its design and functionality.

Components of Automation Simulation

Virtual Models

- **Physical Components:** Simulation involves creating virtual models of the physical components that make up the automated system. This could include machinery, robotic arms, conveyor belts, sensors, and other relevant elements.
- **Control Systems:** The simulation also replicates the control systems that govern the behavior of the automated system. This includes programming logic, algorithms, and feedback loops.

Simulation Software

- **Dynamics and Kinematics:** The software used for automation simulation often incorporates dynamics and kinematics to model the movement and interactions between different components. This is crucial for accurately representing the physical behavior of the system.
- **Real-time Simulation:** Some simulations operate in real-time, allowing users to observe and analyze the system's behavior as it would occur in the actual environment.



Components of Automation Simulation (Cont.)

Input Parameters

- **Scenarios and Variables:** Users can input various scenarios and manipulate variables to observe how the system responds under different conditions. This might involve changing production rates, adjusting machine parameters, or introducing faults to assess the system's resilience.

Visualization and Analysis Tools

- **3D Visualization:** Many automation simulations provide 3D visualizations, allowing users to see the system in action from different perspectives. This helps in understanding spatial relationships and detecting potential collisions or inefficiencies.
- **Data Analysis:** Simulation tools often include data analysis features to assess performance metrics, such as cycle time, throughput, energy consumption, and more.

Applications of Automation Simulation

Manufacturing

- **Process Optimization:** Simulations help optimize manufacturing processes by testing different layouts, production schedules, and resource allocations.
- **Robotics Integration:** Before deploying robots on the factory floor, simulations can validate their programming and ensure efficient collaboration with human workers.

Logistics and Supply Chain

- **Warehouse Design:** Simulations assist in designing and optimizing warehouse layouts for maximum efficiency in material handling and storage.
- **Transportation Planning:** Automated vehicle simulations aid in planning transportation routes and schedules.

Applications of Automation Simulation (Cont.)

Aerospace and Automotive

- **Vehicle Dynamics:** Simulation is crucial for assessing the performance and safety of vehicles, considering factors like aerodynamics, structural integrity, and fuel efficiency.
- **Manufacturing Line Design:** For industries like automotive, simulation helps design and optimize assembly lines.

Training and Education

- **Operator Training:** Simulations provide a safe and controlled environment for training operators on the use of automated systems, reducing the risk of accidents.
- **Educational Tools:** Simulation is used in academic settings to teach students about automation principles and system design.

Business Process Automation

- **Workflow Optimization:** In a business context, simulation can be applied to optimize and streamline various processes, such as order fulfillment, customer service, and resource allocation.



Simulation Models

- **Physical Models:** These represent the physical components of the automated system, such as machinery, sensors, actuators, and production materials. The models include details about dimensions, specifications, and behavior.
- **Logical Models:** Represent the control and decision-making aspects of the system. This includes algorithms, logic controllers, and software components that dictate how the physical components interact.



Simulation Software

- **Discrete Event Simulation (DES):** Commonly used for modeling systems where events occur at distinct points in time. DES is beneficial for simulating manufacturing processes, supply chain logistics, and workflow scenarios.
- **Continuous Simulation:** Used when the system's behavior is best represented as a continuous process. This is common in simulations of physical phenomena like fluid dynamics or chemical reactions.
- **Virtual Prototyping Tools:** These tools allow for the integration of hardware prototypes with software simulations, providing a comprehensive representation of the entire system.



Dynamic Simulation

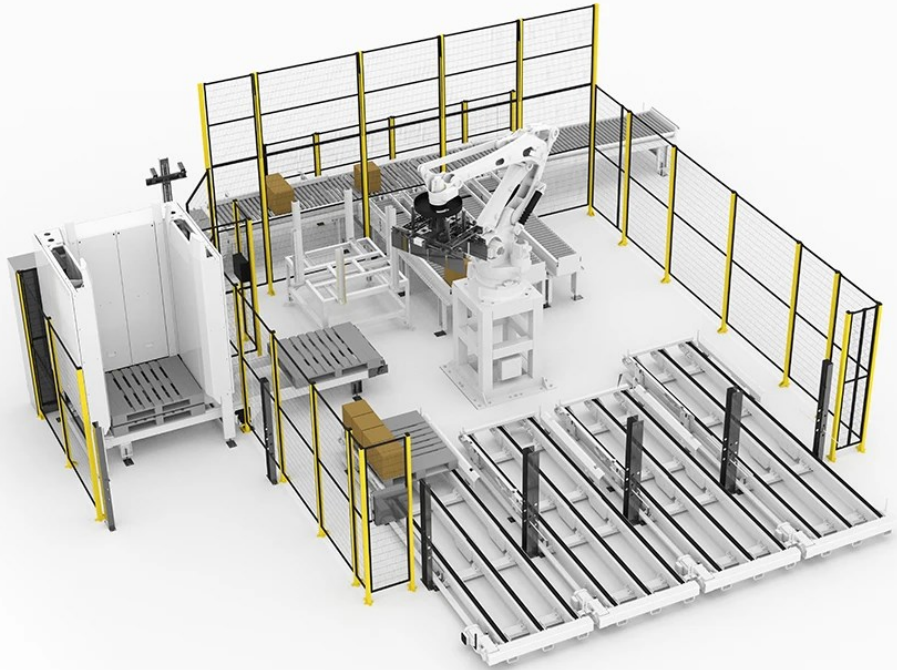
- **Time-based Simulation:** Automation simulations are often time-dependent, allowing users to observe and analyze how the system evolves over time. This is crucial for understanding dynamic behaviors and time-sensitive processes.
- **Human-in-the-Loop Simulation:** Some simulations allow for real-time interaction with human operators. This is valuable for training purposes and for testing how well automated systems integrate with human workflows.



Robotics and Automation in Industry

- **Robot Programming and Coordination:** Validating and optimizing the programming of robotic arms and their coordination in complex manufacturing or assembly processes.
- **Human-Robot Collaboration:** Testing how robots interact with human workers in shared workspaces to ensure safety and efficiency.
- **Factory Layout Optimization:** Simulations help design and optimize the layout of manufacturing facilities for maximum efficiency and minimal bottlenecks.
- **Production Planning:** Forecasting and testing different production schedules to optimize throughput and resource utilization.
- **Quality Control:** Simulating quality control processes to identify potential issues and optimize inspection strategies.

Simulating Application



Case Palletizer (Sanovo Technology Group)

Palletizing

- Palletizing involves the physical arrangement of products on pallets.
- Optimizes storage, transportation, and handling of goods.
- Key in logistics and manufacturing for efficient material handling.

Simulating Application (Cont.)



TT-1400RELP Robotic Case Erector, Loader and Palletizer (Tishma Technologies)

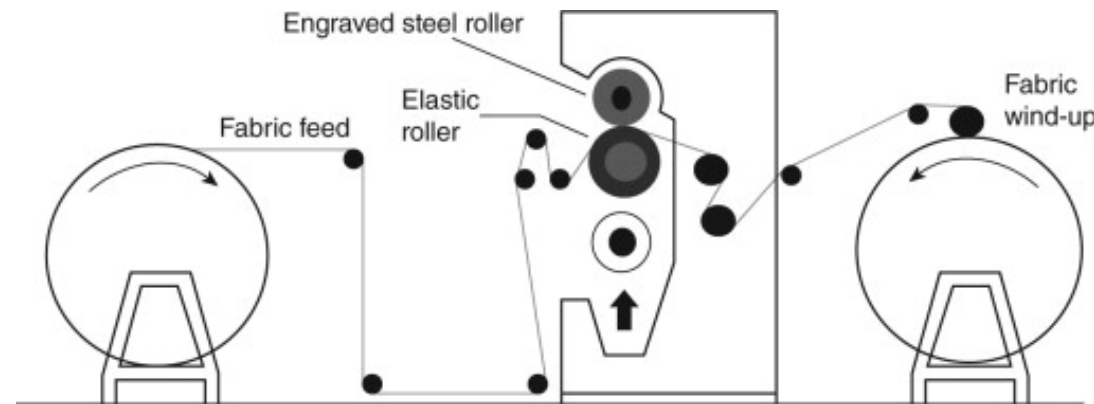
Palletizing Simulation

- Palletizing simulation streamlines the process of arranging and stacking products on pallets.
- Enables testing and refining palletizing algorithms in a virtual environment.
- Enhances efficiency, reduces errors, and optimizes palletizing operations.

Simulating Application (Cont.)

Mechanical Finishing – Machining

- Mechanical finishing in machining involves refining surfaces for precision and quality.
- Simulation aids in optimizing tool paths, speeds, and feeds for efficient finishing.
- Achieves high-quality products while minimizing production time.

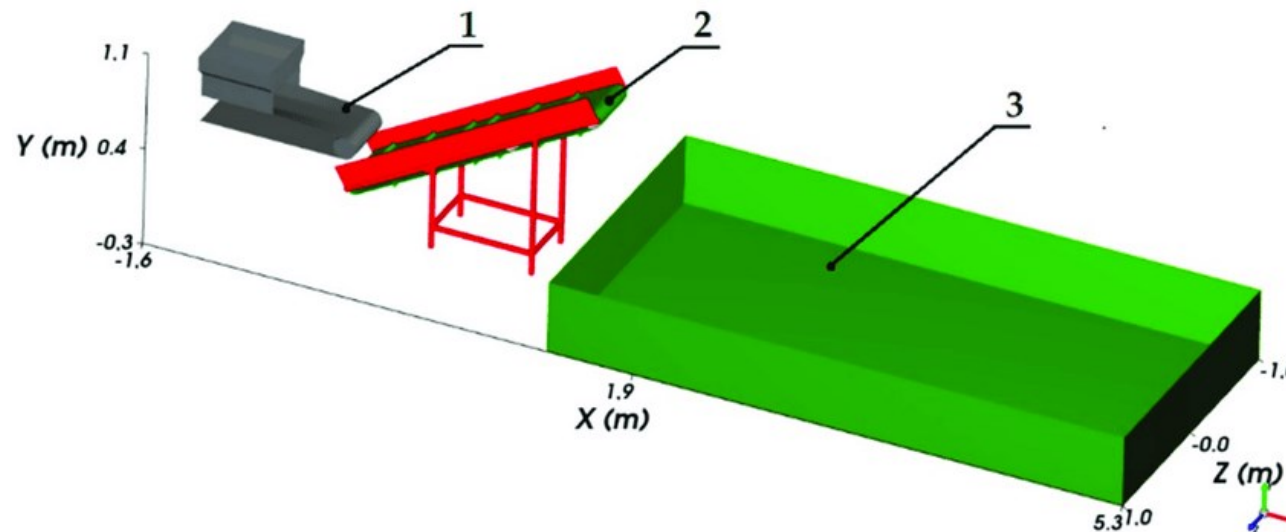


Mechanical Finishing Techniques for Technical Textiles (Kumar & Sundaresan, 2013)

Simulating Application (Cont.)

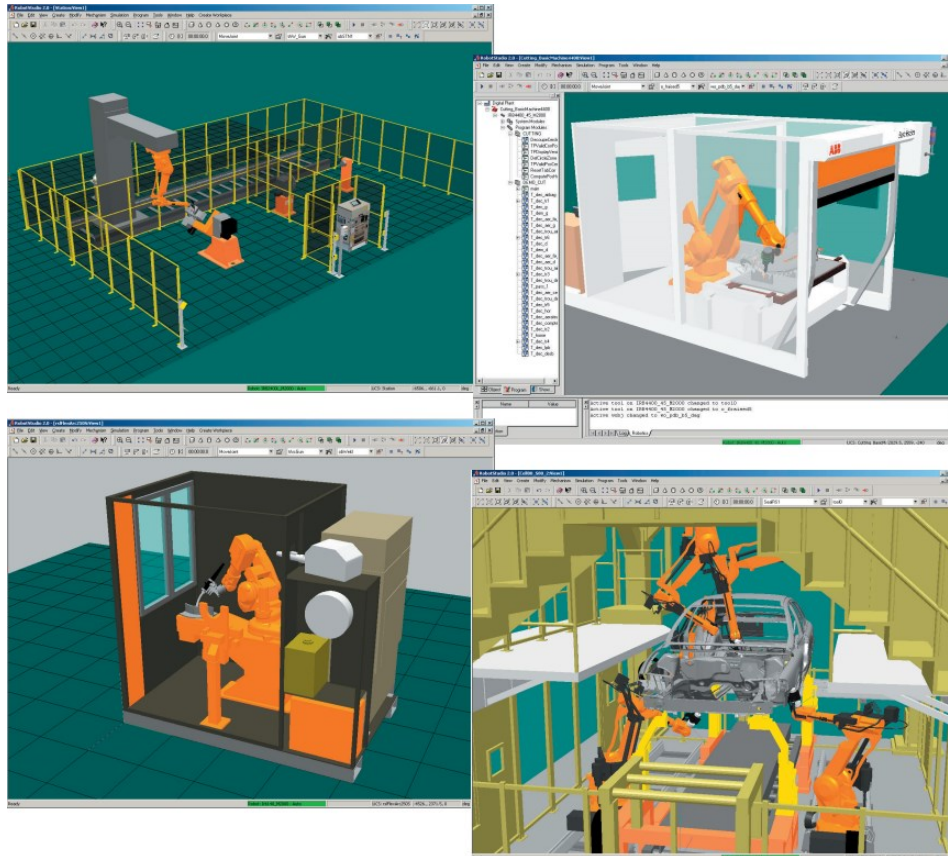
Wood Industry – Simulation

- Simulation in the wood industry optimizes processes such as cutting, shaping, and finishing.
- Enhances precision, reduces waste, and improves overall production efficiency.
- Customized simulations for various wood processing applications.



Simulator of Wood Chips Outlet (Gierz et al., 2020)

Simulating Application (Cont.)

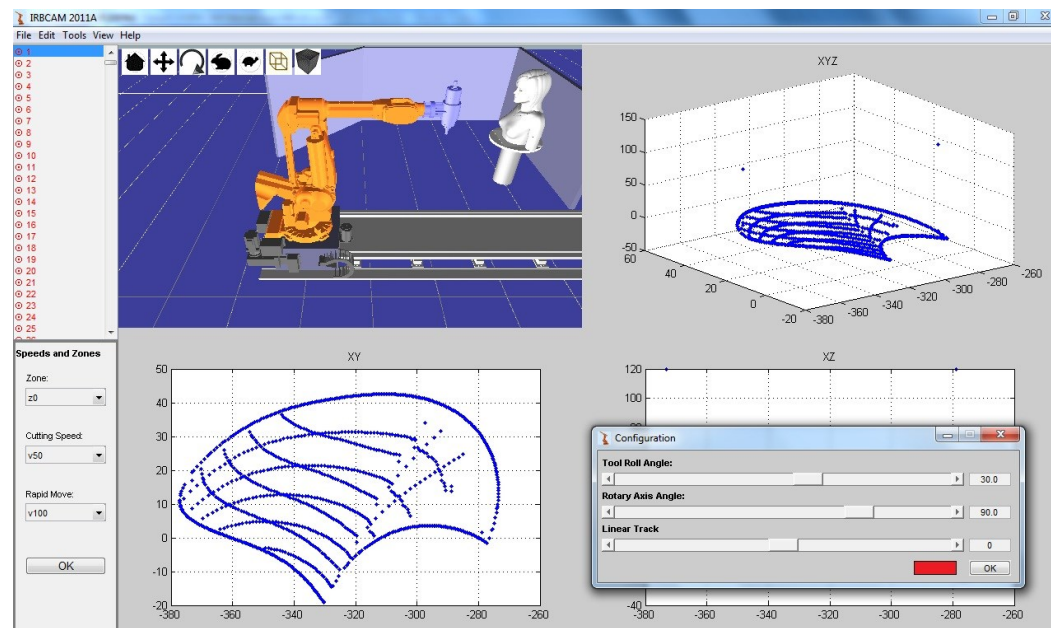


True Offline Programming (RobotStudio)

Mechanical Finishing – Machining

- Mechanical finishing in machining involves refining surfaces for precision and quality.
- Simulation aids in optimizing tool paths, speeds, and feeds for efficient finishing.
- Achieves high-quality products while minimizing production time.

Simulating Application (Cont.)

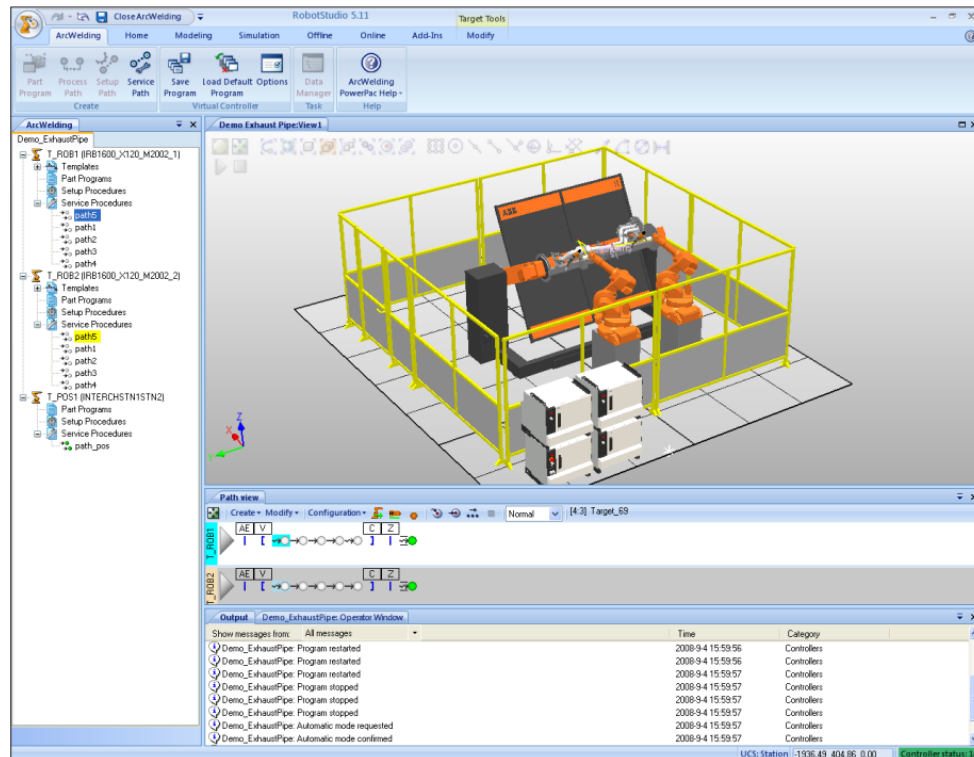


IRBCAM (Premium Solutions)

Mechanical Finishing IRBCAM

- IRBCAM in mechanical finishing optimizes CAM (Computer-Aided Manufacturing) processes.
- Simulation enhances toolpath planning, reducing errors and enhancing efficiency.
- Consistent high-quality mechanical finishing achieved.

Simulating Application (Cont.)



ArcWelding PowerPac for RobotStudio 5 (ABB)

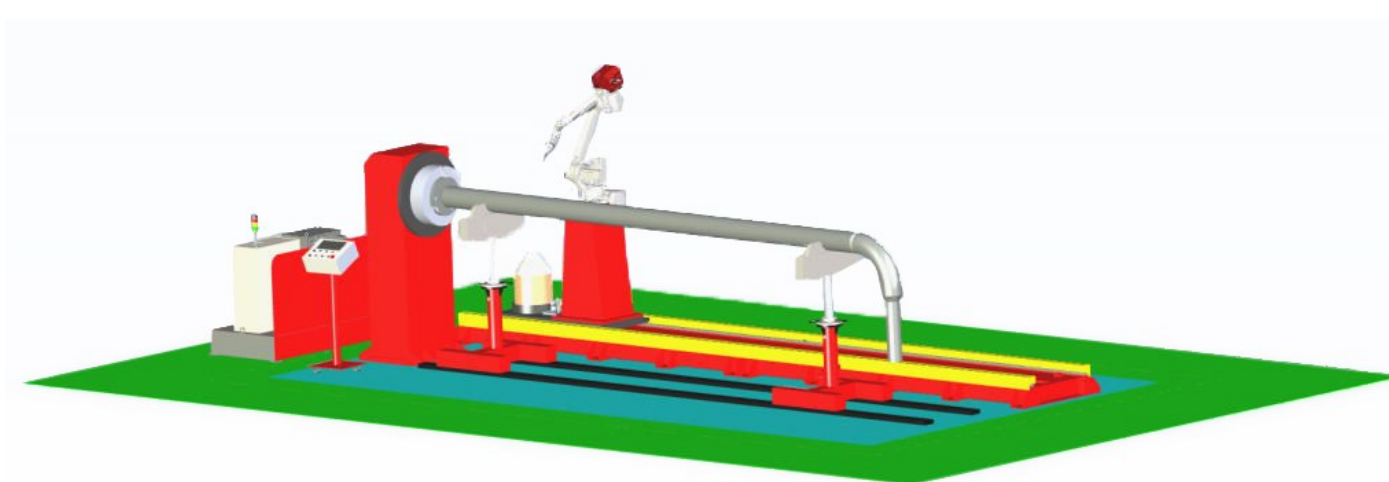
Arc Welding Power Pac – Simulation

- Simulation of Arc Welding Power Pac ensures precise and efficient welding operations.
- Virtual testing of welding parameters, improving weld quality and minimizing defects.
- Enhances safety and reduces material waste in welding processes.

Simulating Application (Cont.)

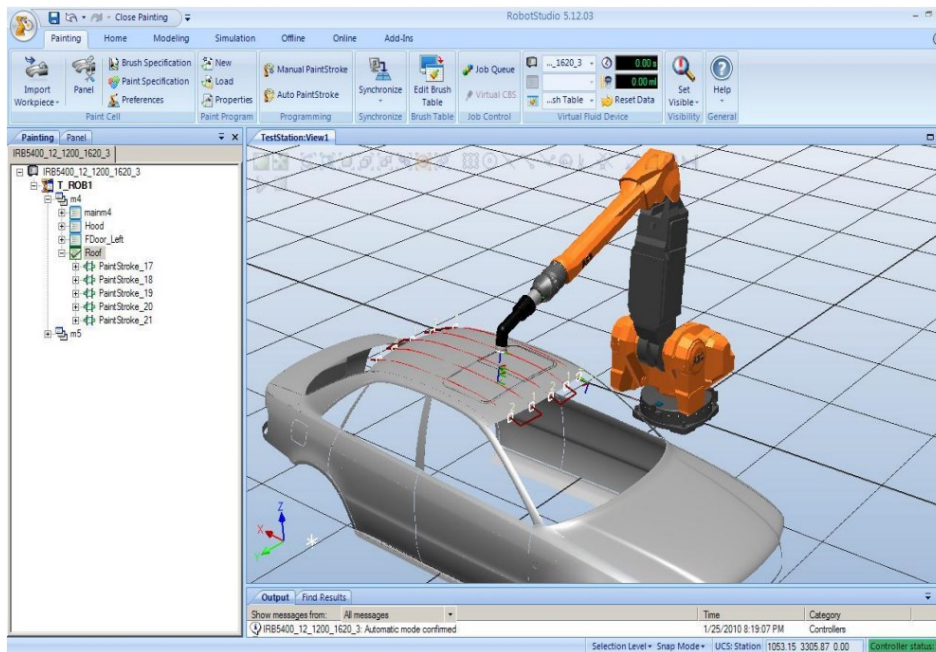
Arc Welding

- Arc welding involves the fusion of materials using an electric arc.
- Essential in various industries for joining metals.
- Precision and efficiency in welding operations.



Robotic Arc Welding Machine (Primo Automation)

Simulating Application (Cont.)

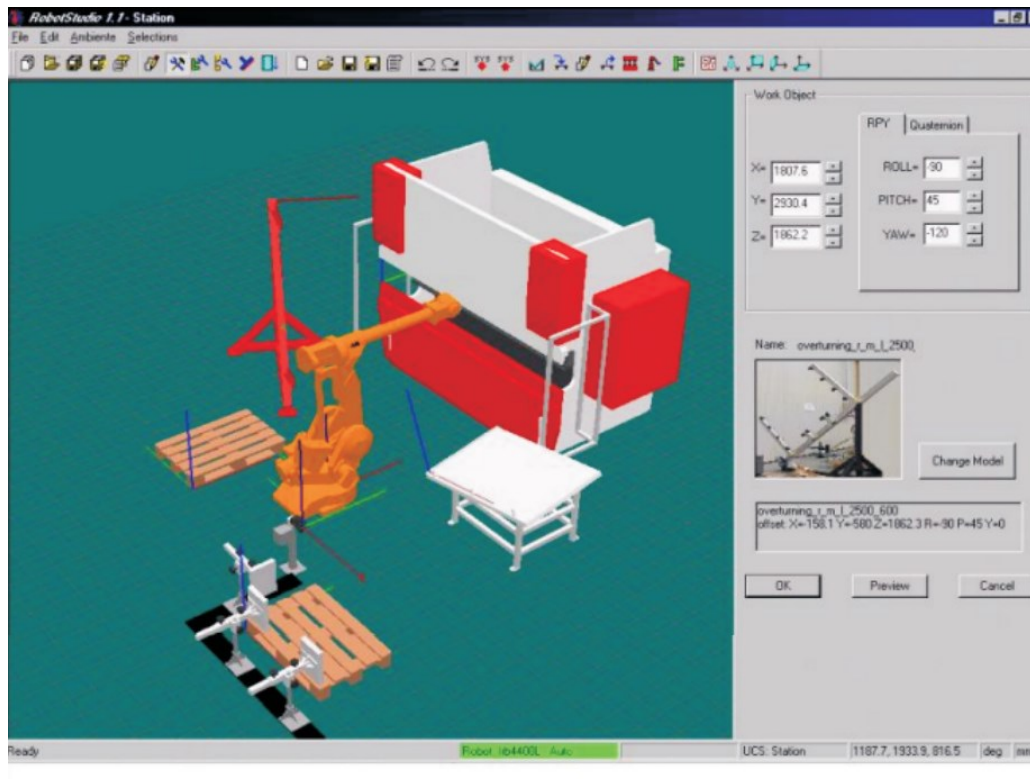


Painting PowerPac (ABB)

Painting (Paint Power Pac)

- Paint Power Pac simulation optimizes the painting process for efficiency and quality.
- Virtual testing of paint application parameters, minimizing errors and enhancing precision.
- Improves surface finish and reduces material waste.

Simulating Application (Cont.)



BendWizard's Operation (ABB)

Machine Tending (Bend Wizard)

- Machine Tending with Bend Wizard optimizes the handling and processing of materials.
- Simulation aids in programming robotic arms for precise material manipulation.
- Improved production efficiency and reduced downtime in machine tending operations.

Simulating Application (Cont.)

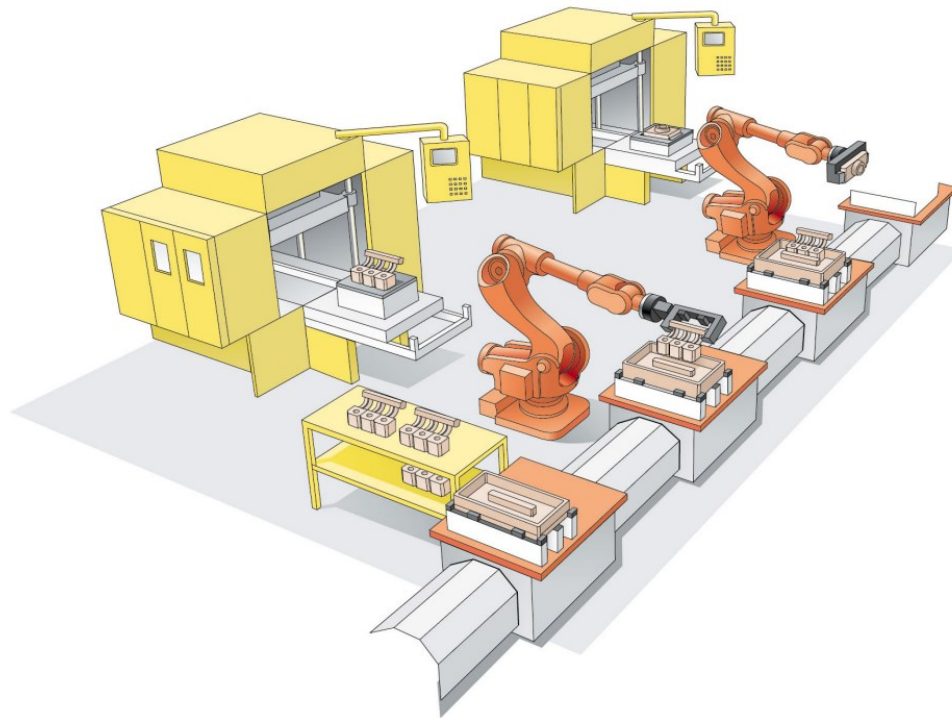


TruBend SL30 (WiCAM)

Machine Tending- Bend

- Machine tending in bending processes involves the automation of material handling.
- Simulation ensures precise bending sequences and minimizes errors.
- Increased productivity and consistency in bending operations.

Simulating Application (Cont.)

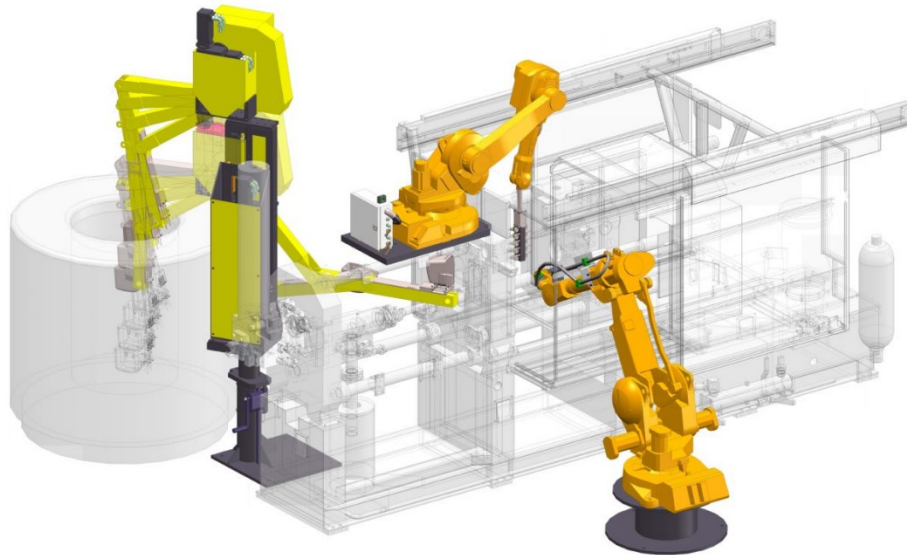


Foundry (ABB)

Machine Tending

- General machine tending involves the automation of tasks like loading and unloading machines.
- Simulation optimizes robotic movements and enhances overall process efficiency.
- Versatile applications across manufacturing industries.

Simulating Application (Cont.)

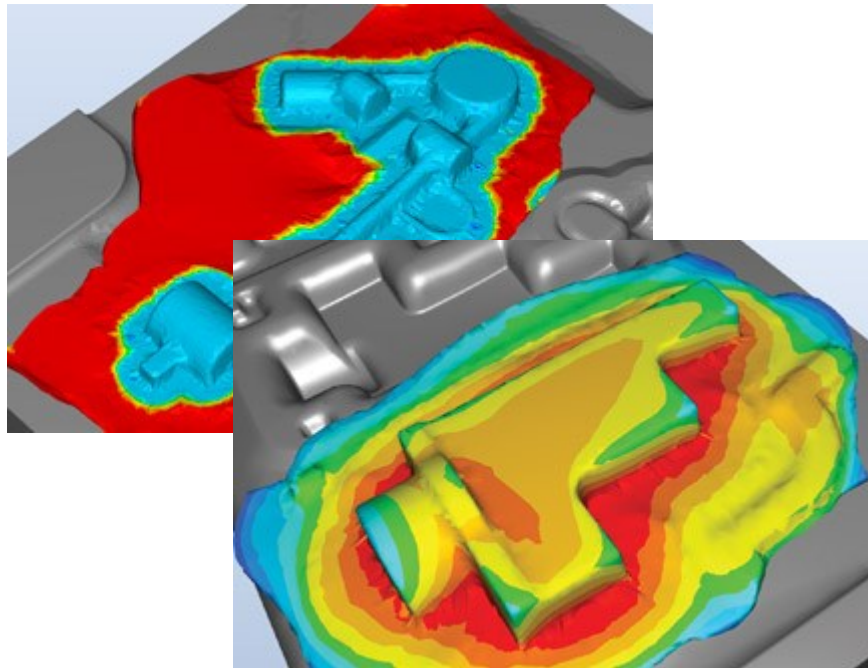


Cell Integration (Pomacautomation.com)

Injection Moulding

- Injection molding simulation optimizes the design and manufacturing of plastic components.
- Virtual testing of mold designs and injection parameters ensures product quality.
- Reduces trial and error, enhancing efficiency in plastic manufacturing.

Simulating Application (Cont.)



Forging Simulation (whitildesley.com)

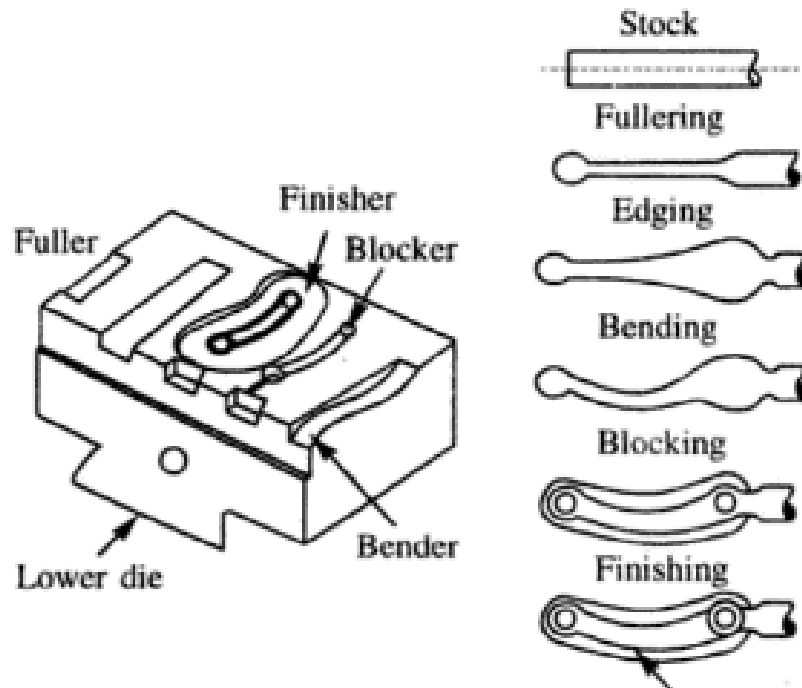
Forging

- Extreme heat, pollution and noise turn forges into one of the toughest workplaces imaginable and an ideal environment for robot-based automation.

Key Points:

- Importance: Shaping metal through compressive forces in forging.
- Simulation benefits: Efficient process design, optimized tooling, improved product quality.

Simulating Application (Cont.)



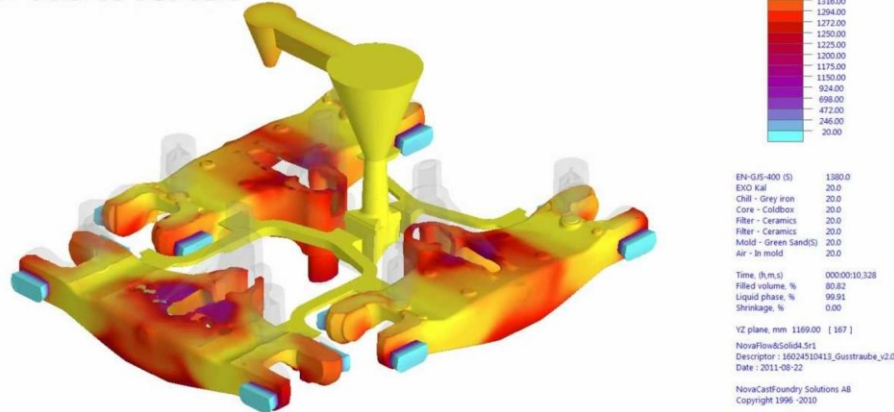
Forging (mechtechguru.com)

Forging

- Forging involves shaping metal through localized compressive forces.
- Simulation aids in designing efficient forging processes and optimizing tooling.
- Improved product strength, durability, and reduced material waste.

Simulating Application (Cont.)


NOVACAST



NovaCast - Iron Casting Simulation (Novacast Systems)

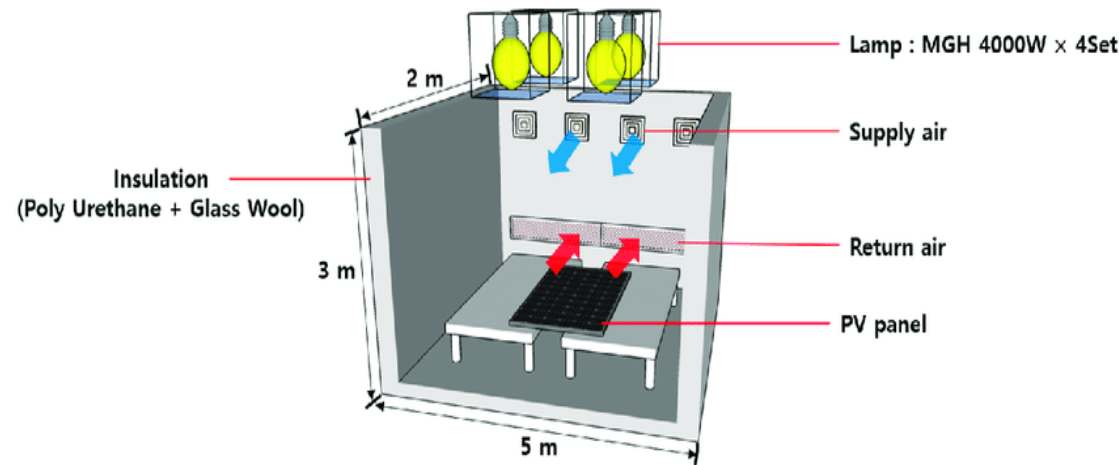
Casting

- Casting simulation enhances the design and manufacturing of metal components.
- Virtual testing of casting processes optimizes mold design and material flow.
- Precision in manufacturing, reduced defects, and improved production efficiency.

Simulating Application (Cont.)

Solar

- Robotics in the solar industry involves tasks like panel assembly and maintenance.
- Simulation ensures efficient robotic movements and system integration.
- Enhanced productivity and reliability in solar-related processes.



Solar Simulator (Kim & Nam, 2019)

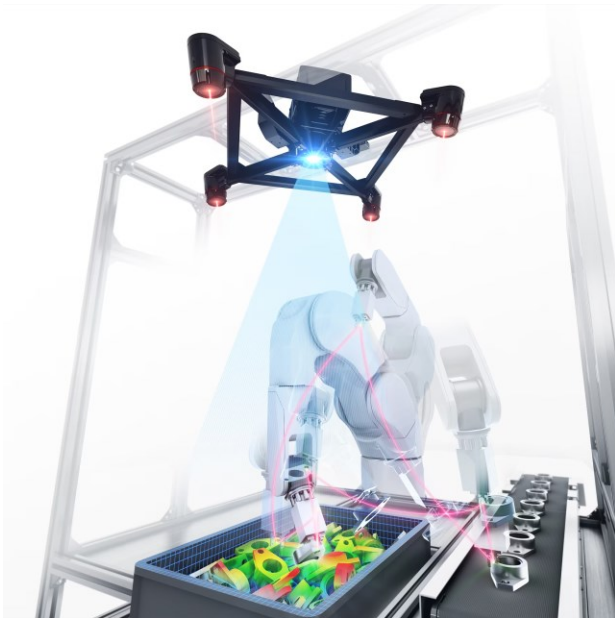
Simulating Application (Cont.)

What is TrueView?

Provide a robot with information about its environment:

- 3D location and orientation of objects
- Position and dimensions of object features
- Type and quality of objects and their features
- VGR means robots:
- Are “adaptive” and work better in realistic manuf. environments
- Can make intelligent decisions and prevent costly mistakes
- Machine Vision – Pickmaster
- Specializes in high-speed random placed objects moving conveyors.
- High-speed conveyor picking
- Inspection
- Identification

Simulating Application (Cont.)



3D Vision-Guided Robotics (keyence.com)

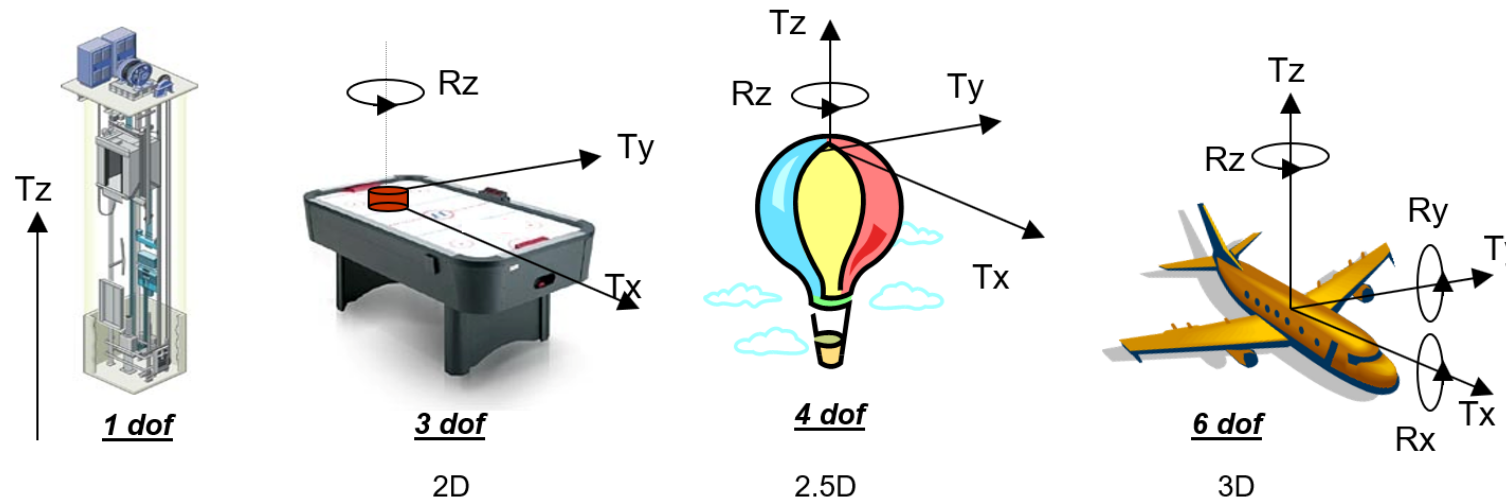
Vision Guided Robotics (VGR) - TrueView

- Robots “see” variation in part styles and location
- Eliminates costly precision fixturing, mechanical part crowding and dunnage
- Automates operations that previously required human interaction
- Increases “Up-Time” and eliminate robot crashes by seeing the part on racks
- Enhances quality via basic inspection and/or part identification

Simulating Application (Cont.)

Robotic Vision

- Robotic vision involves the integration of cameras and sensors for perception.
- Simulation aids in testing and refining vision algorithms for accurate object recognition.
- Crucial for tasks requiring visual feedback, such as pick and place.



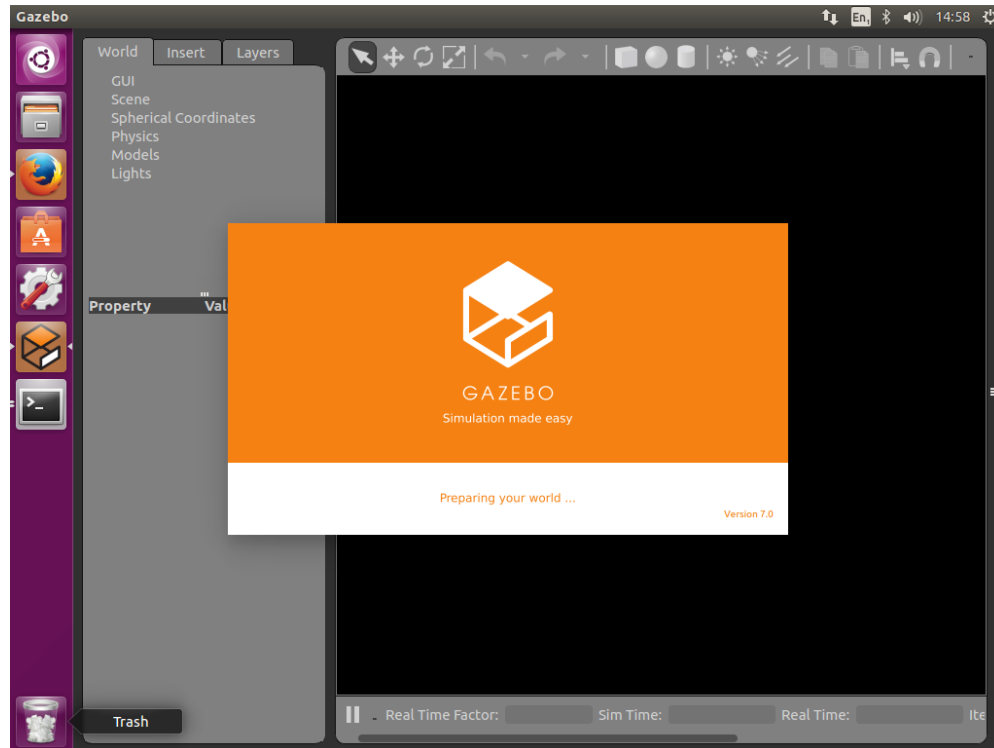
Integration of Vision for Robotic Perception



What is Gazebo?

- **Robot simulation** is an essential tool in every roboticist's toolbox. A well-designed simulator makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios.
- **Gazebo** offers the ability to accurately and efficiently **simulate** populations of robots in complex indoor and outdoor environments. At your fingertips is a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces.
- Gazebo is under active development at the **OSRF** (Open Source Robotics Foundation).

Gazebo Distribution



Gazebo (Zero.J, 2016)

- To achieve ROS integration with stand-alone Gazebo, a set of ROS packages (**gazebo_ros_pkgs**) provides wrappers around the stand-alone Gazebo.
- In our case, for the **UBUNTU 14.04.1 (Trusty LTS)** Virtual Machine we use the stable version of **ROS INDIGO (IGLOO, 2014)** and the **Gazebo 2.x version**.

Gazebo Architecture

Gazebo consists of **two processes**:

- **Server: runs the physics loop and generates sensor data**
 - *Executable*: gzserver
 - *Libraries*: Physics, Sensors, Rendering, Transport
- **Client: provides user interaction and visualization of a simulation.**
 - *Executable*: gzclient
 - *Libraries*: Transport, Rendering, GUI
- With Gazebo it is possible to create and run:
 - **Environment**: .world file
 - **Models (i.e. Robot)**: .sdf, .urdf, .xacro files

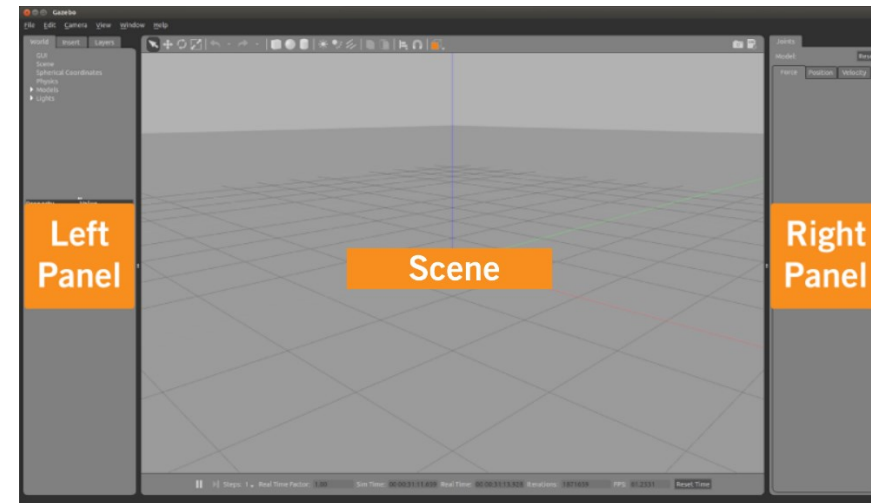
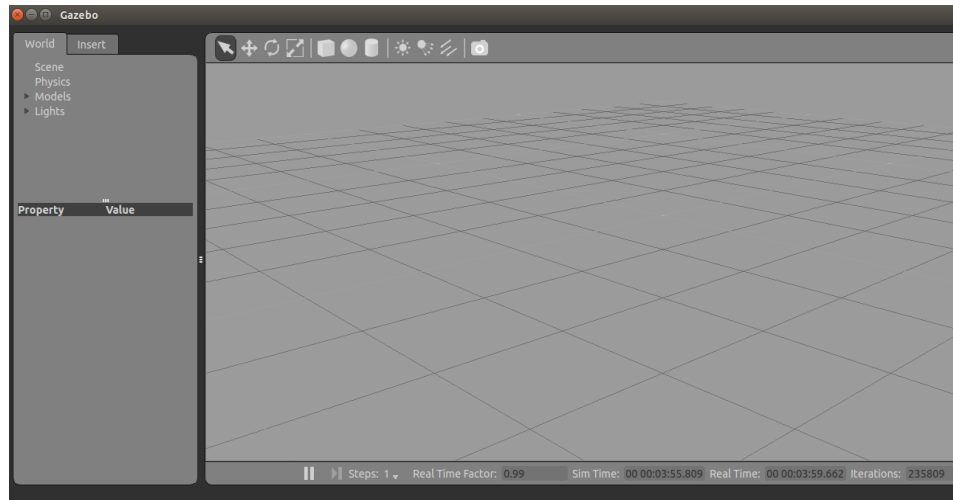


SDF vs URDF

- **SDF** is an XML file that contains a **complete description** for everything from the world level down to the robot level, including:
 - **Scene:** Ambient lighting, sky properties, shadows.
 - **Physics:** Gravity, time step, physics engine.
 - **Models:** Collection of links, collision objects, joints, and sensors.
 - **Lights:** Point, spot, and directional light sources.
 - **Plugins:** World, model, sensor, and system plugins.
- **URDF** can only specify the kinematic and dynamic properties of a single robot in isolation:
 - URDF can not specify the pose of the robot itself within a world
 - It cannot specify objects that are not robots, such as lights, heightmaps, etc.
 - Lacks friction and other properties
- **NB:** The **world** description file contains all the elements in a simulation, including robots, lights, sensors, and static objects:
 - This file is formatted using SDF and has a **.world** extension.
 - The Gazebo server (gzserver) reads this file to generate and populate a world.

Gazebo Features

- To use **gazebo** simulator as **stand-alone** we call it by terminal simply with:
 > *gazebo*
- A **GUI** appears with an empty world



Gazebo in Ubuntu (Peng et al., 2023)



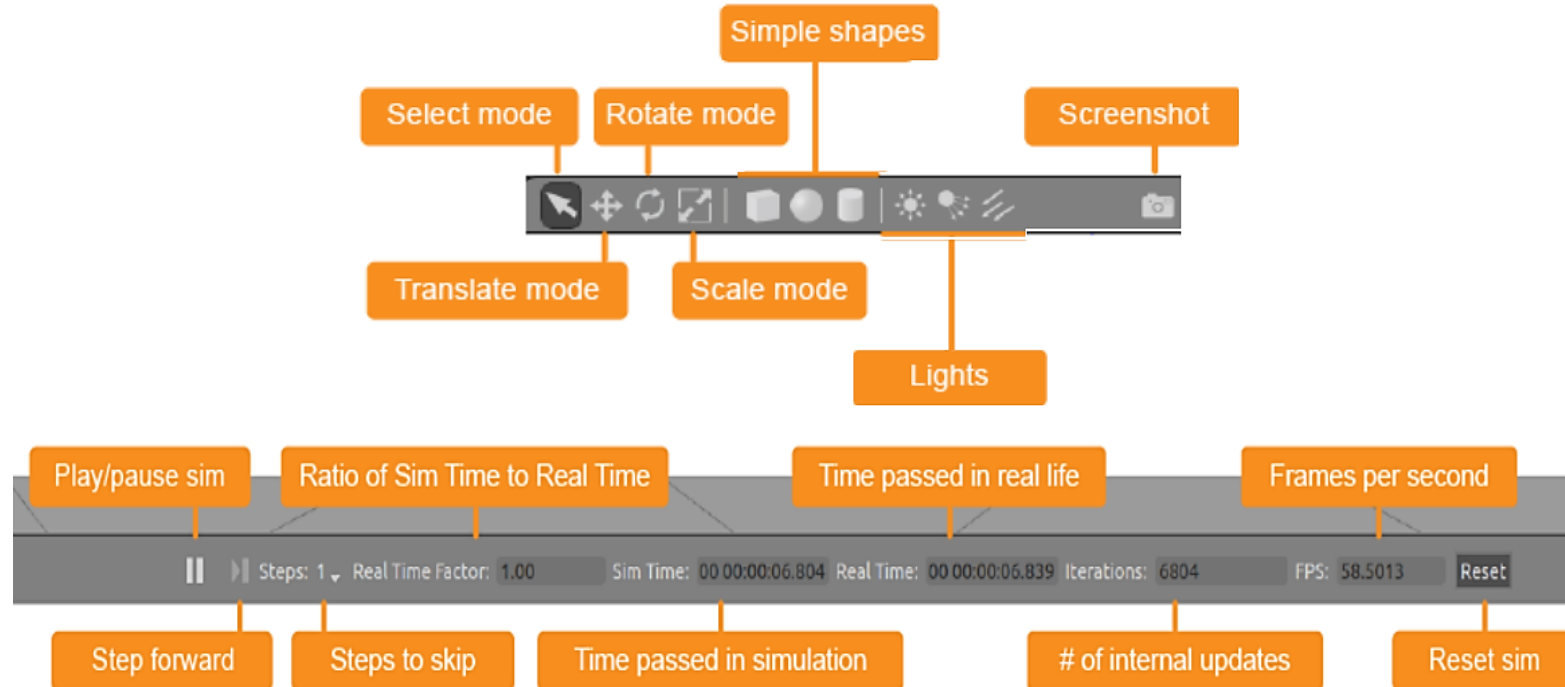
Gazebo Features (Cont.)

The GUI is composed by three panels:

- **Scene:** it is the main panel where the world and models are loaded and positioned
- **Left Panel:** it appears by default when you launch Gazebo. It is composed of two subtabs:
 - **WORLD:** The World tab displays the models that are currently in the scene, and allows you to view and modify model parameters, like their pose. You can also change the camera view angle by expanding the "GUI" option and tweaking the camera pose.
 - **INSERT:** The Insert tab is where you add new objects (models) to the simulation. To see the model list, you may need to click the arrow to expand the folder. Click (and release) on the model you want to insert, and click again in the Scene to add it.
- **Right Panel (hidden by default)**
 - The right panel is hidden by default. Click and drag the bar to open it. The right panel can be used to interact with the mobile parts of a selected model (the joints). If there are no models selected in the Scene, the panel does not display any information.

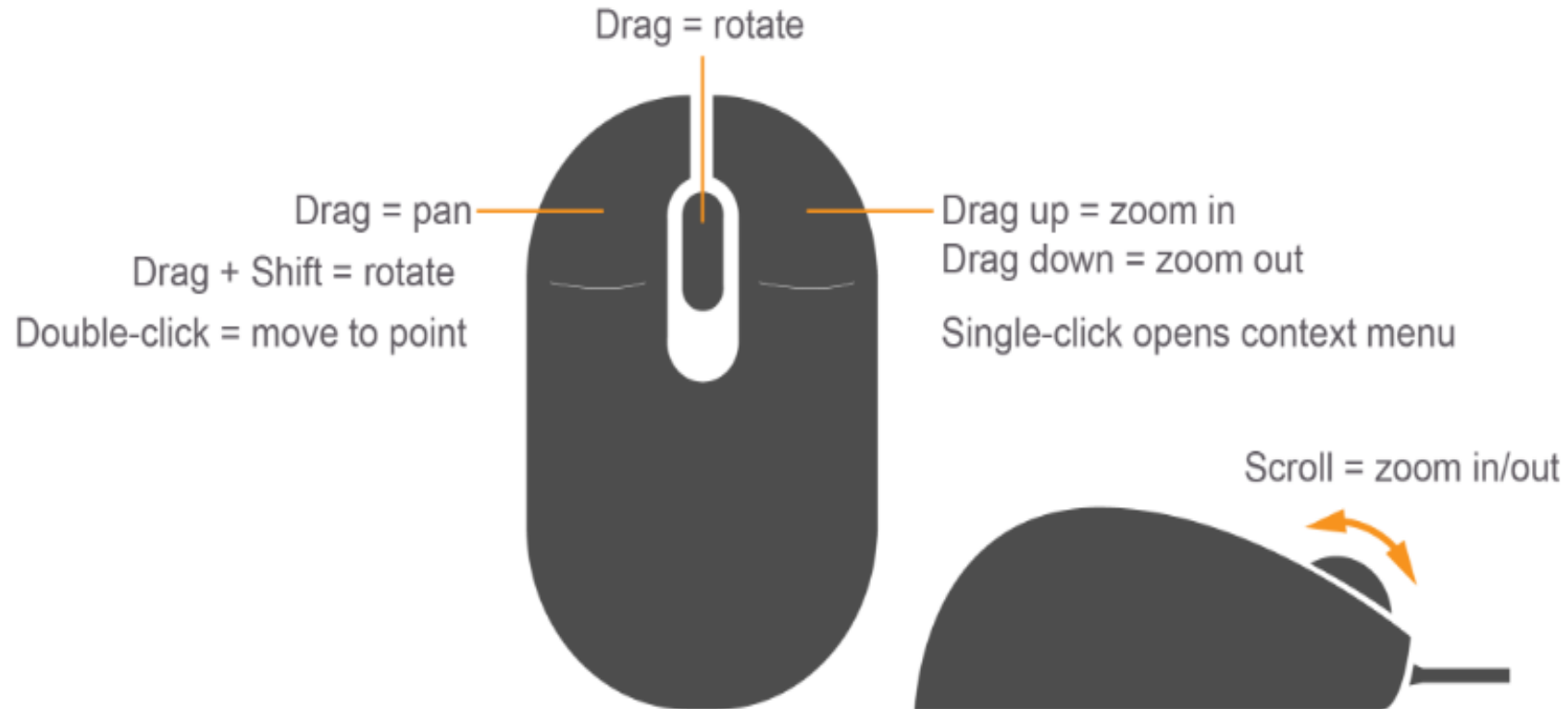


Gazebo Features (Cont.)



Gazebo GUI (<http://kaiyuryozin.blog45.fc2.com>)

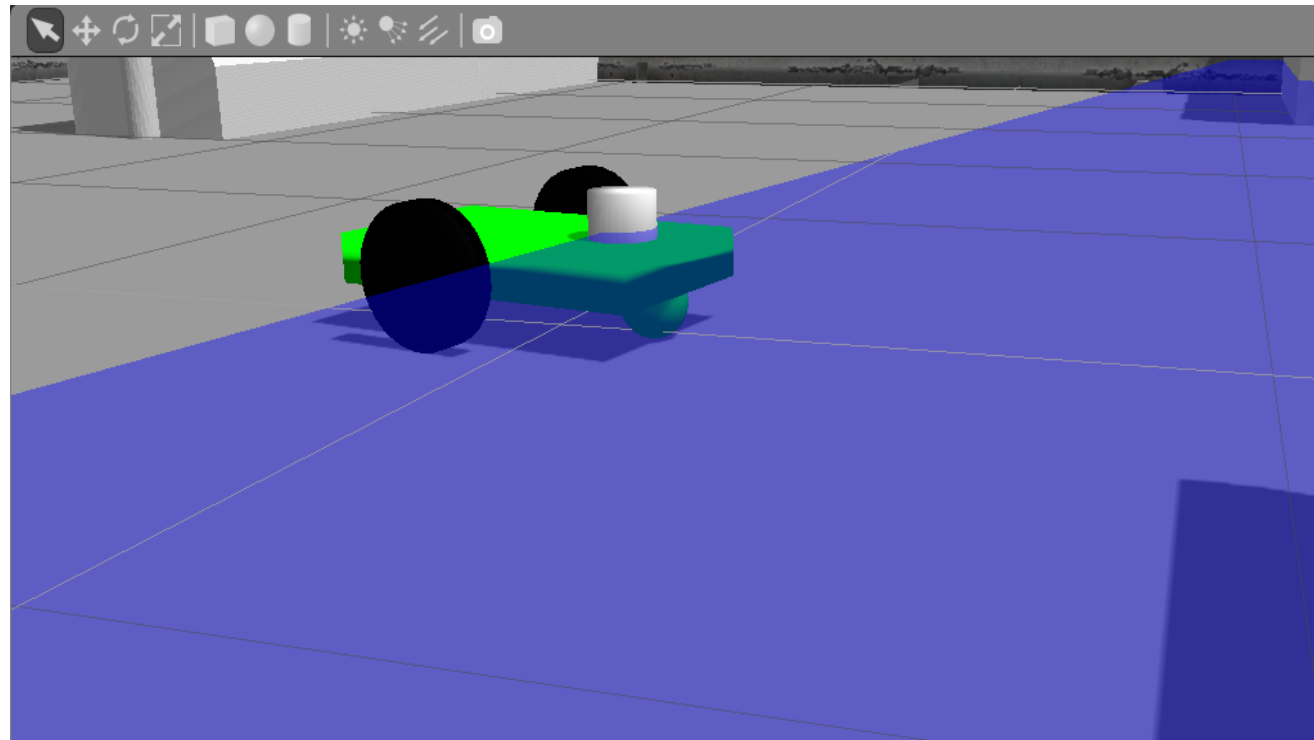
Gazebo Features (Cont.)



Gazebo Mouse Control (PNG Egg)

Gazebo Features (Cont.)

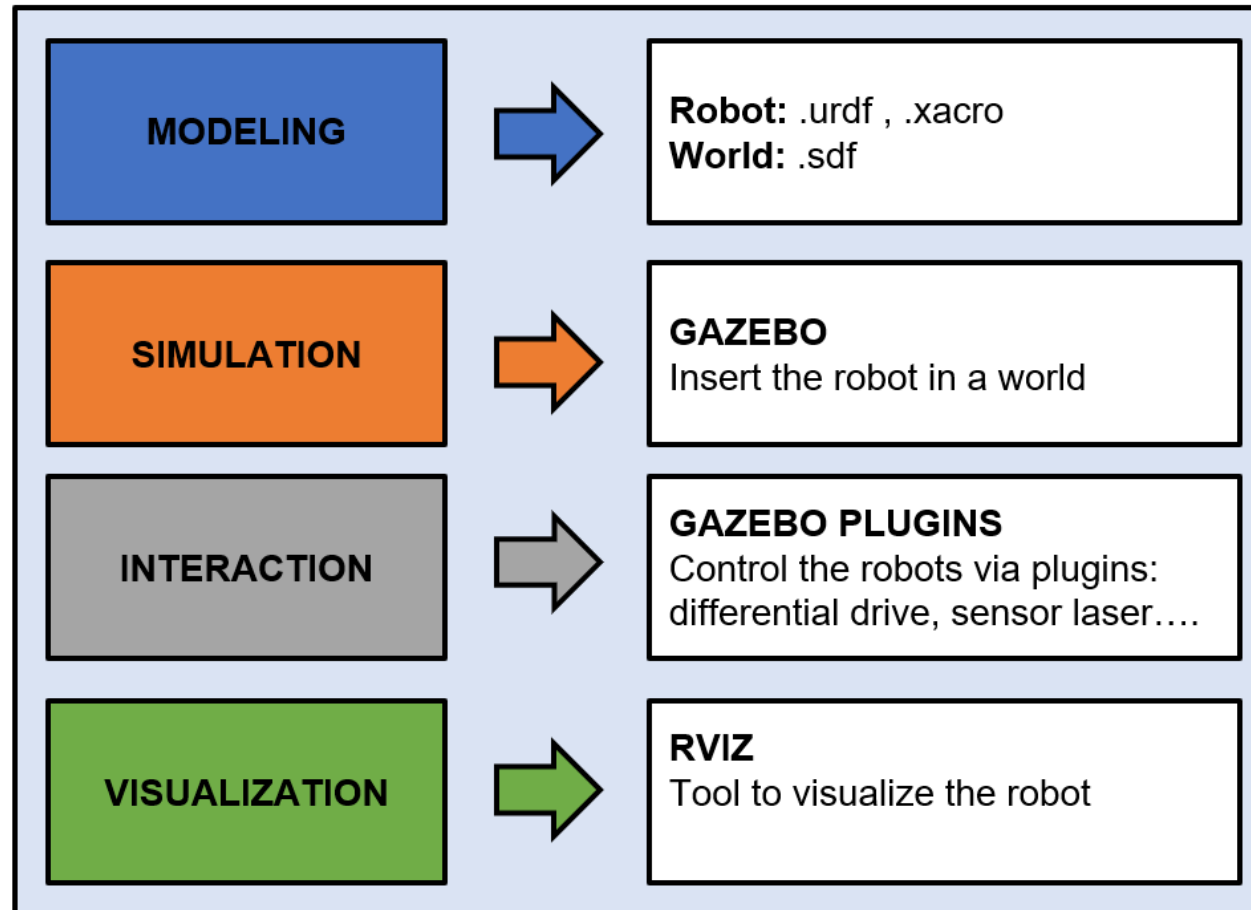
- We want to create a simple differential drive robot with a frontal laser scanner.



Vehicle Robot Simulation (Bonetti & De Cecco, 2019).



Gazebo Scheme





Gazebo Scheme (Cont.)

- Go to the current workspace with the following command: `> cd /home/student/catkin_ws` or `> cd ~/catkin_ws`
- Load the **default and devel workspaces** with: `> source /opt/ros/indigo/setup.bash`
`> source devel/setup.bash`
- Now go to the `src` directory
`> cd /home/student/catkin_ws/src`
- We create a new package inside the `src` workspace directory called `my_robot`:
`> catkin_create_pkg --rostdistro indigo my_robot roscpp rospy gazebo_msgs geometry_msgs sensor_msgs std_msgs urdf`
- To build your workspace (including compiling all of the executables in all of its packages) we have to return to the root directory of our workspace: `> cd /home/student/catkin_ws` or `> cd ~/catkin_ws`
- Build and create executable for all packages: `> catkin_make`
- Whenever you build a **new package, update your environment**: `> source devel/setup.bash`
- If the package is not visible, try to restart the shell or run the command: `> rospack find my_robot`



Gazebo Scheme (Cont.)

- Now go to the package directory:
> `cd /home/student/catkin_ws/src/my_robot`
- And we create the following directories to put our files for gazebo modeling and simulation:
> `mkdir gazebo_simulation`
> `cd gazebo_simulation`
> `mkdir world urdf`
- In the directory **world** there will be the files `.world` related to the environment
- In the directory **urdf** there will be the files `.xacro` related to the models (i.e. robot)

NB: A few more terminologies to familiarize are **xacro** and **macro**, basically **xacro** is a file format encoded in **xml**. **xacro** files come with extra features called macros (akin functions) that helps in reducing the amount of written text in writing robot description. Robot model description for Gazebo simulation is described in **URDF** model format and **xacro** files simplify the process of writing elaborate robot description.

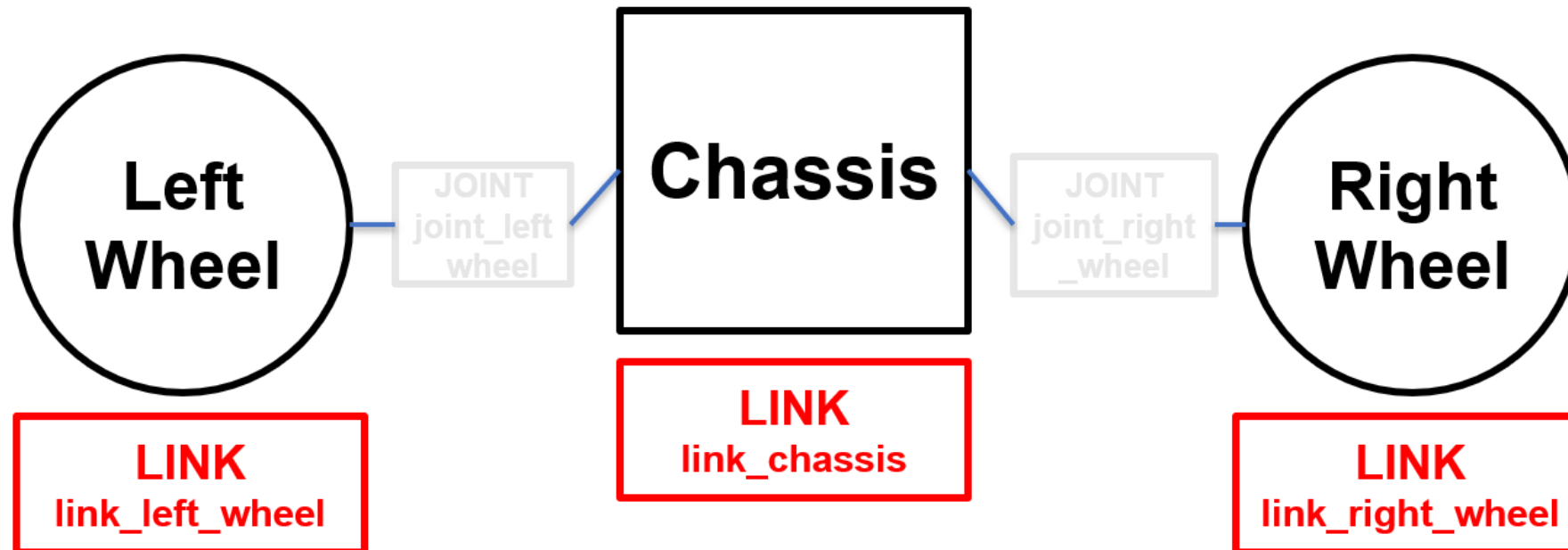
Gazebo Scheme (Cont.)

- Go to the **urdf** directory and create a file: > *gedit my_robot.xacro*
- Copy and paste the xml code:

```
<?xml version="1.0" encoding="UTF-8" ?>
<robot name= "my_robot" xmlns:xacro="https://www.ros.org/wiki/xacro" > </robot>
```

- And **now we introduce inside the robot tags (<robot > ... </robot>)**, the link, joint and other tags in order to represent our robot
- In this xacro file we will define the following:
 - **chassis + caster wheel** : link type element
 - **wheels** : link type elements (left + right)
 - **joints** : joint type elements (left + right)
- All of these elements have some common properties like inertial, collision and visual. The **inertial** and **collision** properties enable physics simulation and the **visual** property controls the appearance of the robot
- The **joints** help in defining the relative motion between **links** such as the motion of wheel with respect to the chassis. The **wheels** are links of cylindrical geometry, they are oriented (using rpy property) such that rolling is possible. The placement is controlled via the xyz property. For joints, we have specified the values for damping and friction.

Gazebo Modeling



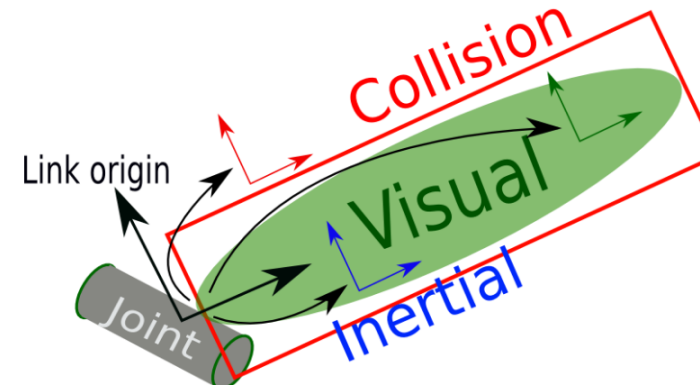
A Caster Wheel is the Chassis

Gazebo Modeling (Cont.)

```

<link name="link_chassis">
  <!-- pose and inertial -->
  <pose>0 0 0.1 0 0 0</pose>
  <inertial>
    <mass value="5"/>
    <origin rpy="0 0 0" xyz="0 0 0.1"/>
    <inertia ixx="0.0395416666667" ixy="0" ixz="0" iyy="0.106208333333" iyz="0" izz="0.106208333333"/>
  </inertial>
  <collision name="collision_chassis">
    <geometry>
      <box size="0.5 0.3 0.07"/>
    </geometry>
  </collision>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <box size="0.5 0.3 0.07"/>
    </geometry>
    <material name="blue"/>
  </visual>

```



Manipulator Motion Control (Peng et al., 2023)

Gazebo Modeling (Cont.)

```
<!-- caster front -->
  <collision name="caster_front_collision">
    <origin rpy=" 0 0 0" xyz="0.35 0 -0.05"/>
    <geometry>
      <sphere radius="0.05"/>
    </geometry>
    <surface>
      <friction>
        <ode>
          <mu>0</mu>
          <mu2>0</mu2>
          <slip1>1.0</slip1>
          <slip2>1.0</slip2>
        </ode>
      </friction>
    </surface>
  </collision>
  <visual name="caster_front_visual">
    <origin rpy=" 0 0 0" xyz="0.2 0 -0.05"/>
    <geometry>
      <sphere radius="0.05"/>
    </geometry>
  </visual>
</link>
```

Gazebo Modeling (Cont.)

```
<!-- Create wheel right -->
<link name="link_right_wheel">
  <inertial>
    <mass value="0.2"/>
    <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
    <inertia ixx="0.00052666666" ixy="0" ixz="0" iyy="0.00052666666" iyz="0" izz="0.001"/>
  </inertial>
  <collision name="link_right_wheel_collision">
    <origin rpy="0 1.5707 1.5707" xyz="0 0 0" />
    <geometry>
      <cylinder length="0.04" radius="0.1"/>
    </geometry>
  </collision>
  <visual name="link_right_wheel_visual">
    <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
    <geometry>
      <cylinder length="0.04" radius="0.1"/>
    </geometry>
  </visual>
</link>
```



Gazebo Modeling (Cont.)

```
<!-- Joint for right wheel -->  
<joint name="joint_right_wheel" type="continuous">  
  <origin rpy="0 0 0" xyz="-0.05 0.15 0"/>  
  <child link="link_right_wheel" />  
  <parent link="link_chassis"/>  
  <axis rpy="0 0 0" xyz="0 1 0"/>  
  <limit effort="10000" velocity="1000"/>  
  <joint_properties damping="1.0" friction="1.0" />  
</joint>
```


Gazebo Modeling (Cont.)

```

<!-- Left Wheel link -->
<link name="link_left_wheel">
  <inertial>
    <mass value="0.2"/>
    <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
    <inertia ixx="0.00052666666" ixy="0" ixz="0" iyy="0.00052666666" iyz="0" izz="0.001"/>
  </inertial>
  <collision name="link_left_wheel_collision">
    <origin rpy="0 1.5707 1.5707" xyz="0 0 0" />
    <geometry>
      <cylinder length="0.04" radius="0.1"/>
    </geometry>
  </collision>
  <visual name="link_left_wheel_visual">
    <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
    <geometry>
      <cylinder length="0.04" radius="0.1"/>
    </geometry>
  </visual>
</link>

```

Gazebo Modeling (Cont.)

```
<!-- Joint for left wheel -->  
<joint name="joint_left_wheel" type="continuous">  
  <origin rpy="0 0 0" xyz="-0.05 -0.15 0"/>  
  <child link="link_left_wheel" />  
  <parent link="link_chassis"/>  
  <axis rpy="0 0 0" xyz="0 1 0"/>  
  <limit effort="10000" velocity="1000"/>  
  <joint_properties damping="1.0" friction="1.0" />  
</joint>
```

Gazebo Modeling (Cont.)

```
<material name="black">
  <color rgba="0.0 0.0 0.0 1.0"/>
</material>
<material name="blue">
  <color rgba="0.203125 0.23828125 0.28515625 1.0"/>
</material>
<material name="green">
  <color rgba="0.0 0.8 0.0 1.0"/>
</material>
<material name="grey">
  <color rgba="0.2 0.2 0.2 1.0"/>
</material>
<material name="orange">
  <color rgba="1.0 0.423529411765 0.0392156862745 1.0"/>
</material>
<material name="brown">
  <color rgba="0.870588235294 0.811764705882 0.764705882353 1.0"/>
</material>
<material name="red">
  <color rgba="0.80078125 0.12890625 0.1328125 1.0"/>
</material>
<material name="white">
  <color rgba="1.0 1.0 1.0 1.0"/>
</material>
```

Gazebo Modeling (Cont.)

```
<gazebo reference="link_chassis">  
  <material>Gazebo/Orange</material>  
</gazebo>  
<gazebo reference="link_left_wheel">  
  <material>Gazebo/Blue</material>  
</gazebo>  
<gazebo reference="link_right_wheel">  
  <material>Gazebo/Blue</material>  
</gazebo>
```

Gazebo Modeling (Cont.)

- **Now we have our model!**
- In order to **run** it into **gazebo**, we have to use the **launch** tool.
- It is a ROS Tool used for launching multiple nodes that use XML **.launch files*
- If not yet running, launch automatically starts a roscore
- **In general**, browse to the folder and start a launch file with:
 > *roslaunch file_name.launch*
- And start a launch file from a package with:
 > *roslaunch package_name file_name.launch*
- The file structure of a .launch file is composed of several elements:
 - **launch**: Root element of the launch file
 - **node**: Each *<node>* tag specifies a node to be launched
 - **name**: Name of the node (free to choose)
 - **pkg**: Package containing the node
 - **type**: Type of the node, there must be a corresponding executable with the same name
 - **output**: Specifies where to output log messages (screen: console, log: log file)

Gazebo Modeling (Cont.)

- **In our case**, we want to:
 - Include the robot in Gazebo
 - Open Gazebo
 - (the environment till now is empty)
- To do the above we create our launch file. Go to **package** folder and create a **launch** directory:
 - > ***cd ~/catkin_ws/src/my_robot/***
 - > ***mkdir launch***
- Inside the launch directory, create the launch file calling:
 - > ***gedit gazebo.launch***

Gazebo Modeling (Cont.)

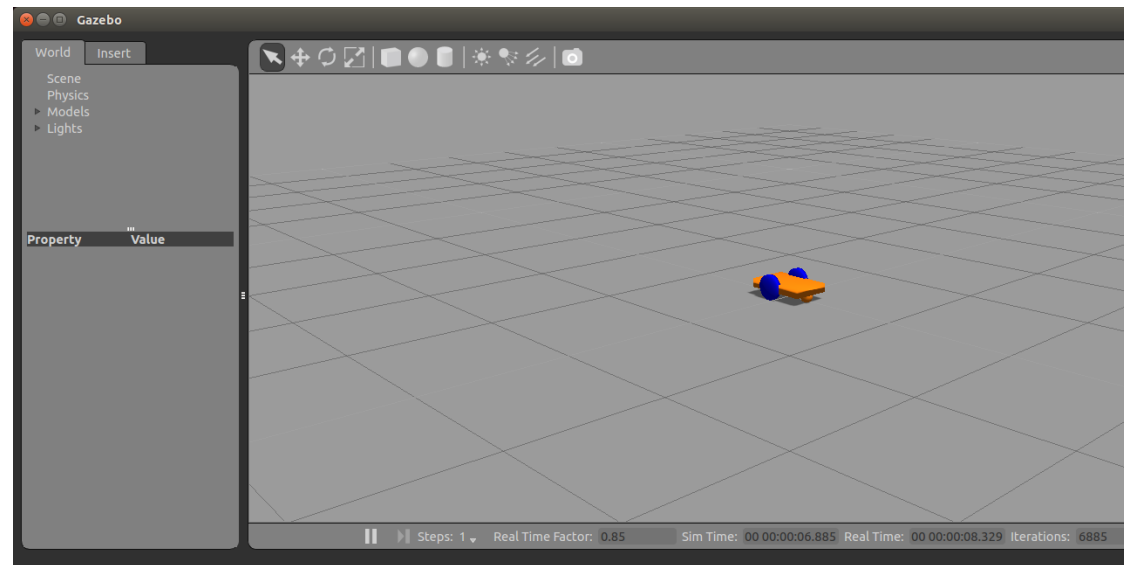
```
<?xml version="1.0" encoding="UTF-8"?>
<launch>
  <!-- OPEN GAZEBO AND LOAD THE EMPTY WORLD AS INITIALIZATION -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="use_sim_time" value="true"/>
    <arg name="debug" value="false"/>
  </include>
  <!-- ADD THE ROBOT DESCRIPTION FROM THE URDF (.xacro) FILE -->
  <param name="robot_description" command="$(find xacro)/xacro.py '$(find
    my_robot)/gazebo_simulation/urdf/my_robot.xacro'" />
  <!-- Name of robot and starting positions (x, y, z) -->
  <arg name="robot_name" default="my_robot"/>
  <arg name="x" default="0"/>
  <arg name="y" default="0"/>
  <arg name="z" default="0.5"/>
```

Gazebo Modeling (Cont.)

```
<!-- SPAWN THE MODEL OF THE ROBOT IN THE GAZEBO WORLD -->
<node name="spawn_$(arg robot_name)" pkg="gazebo_ros" type="spawn_model" output="screen"
  args="-unpause -urdf -param robot_description -model $(arg robot_name) -x $(arg x) -y $(arg y) -z
  $(arg z)" />
<!-- send fake joint values -->
<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
  <param name="use_gui" value="False"/>
</node>
<node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher">
</node>
<!-- RVIZ Visualizator -->
<!-- From configuration file -->
<!--<node pkg="rviz" type="rviz" name="rviz"></node> -->
</launch>
```


Gazebo Simulation

- After that, call in a terminal the launch file using the following command:
 > *roslaunch my_robot gazebo.launch*
- Gazebo as plugin of ROS (so not stand-alone) is opened and the robot will appear in the empty world.



Gazebo in Ubuntu (Peng et al., 2023)



Gazebo Control

Now we are ready to add **control** to our robot. We will add a new element called **plugin** to our **my_robot.xacro** file. We will add a differential drive **plugin** to our robot. So **reopen the file** and **add at the end of the file the new tag** that looks like follows:

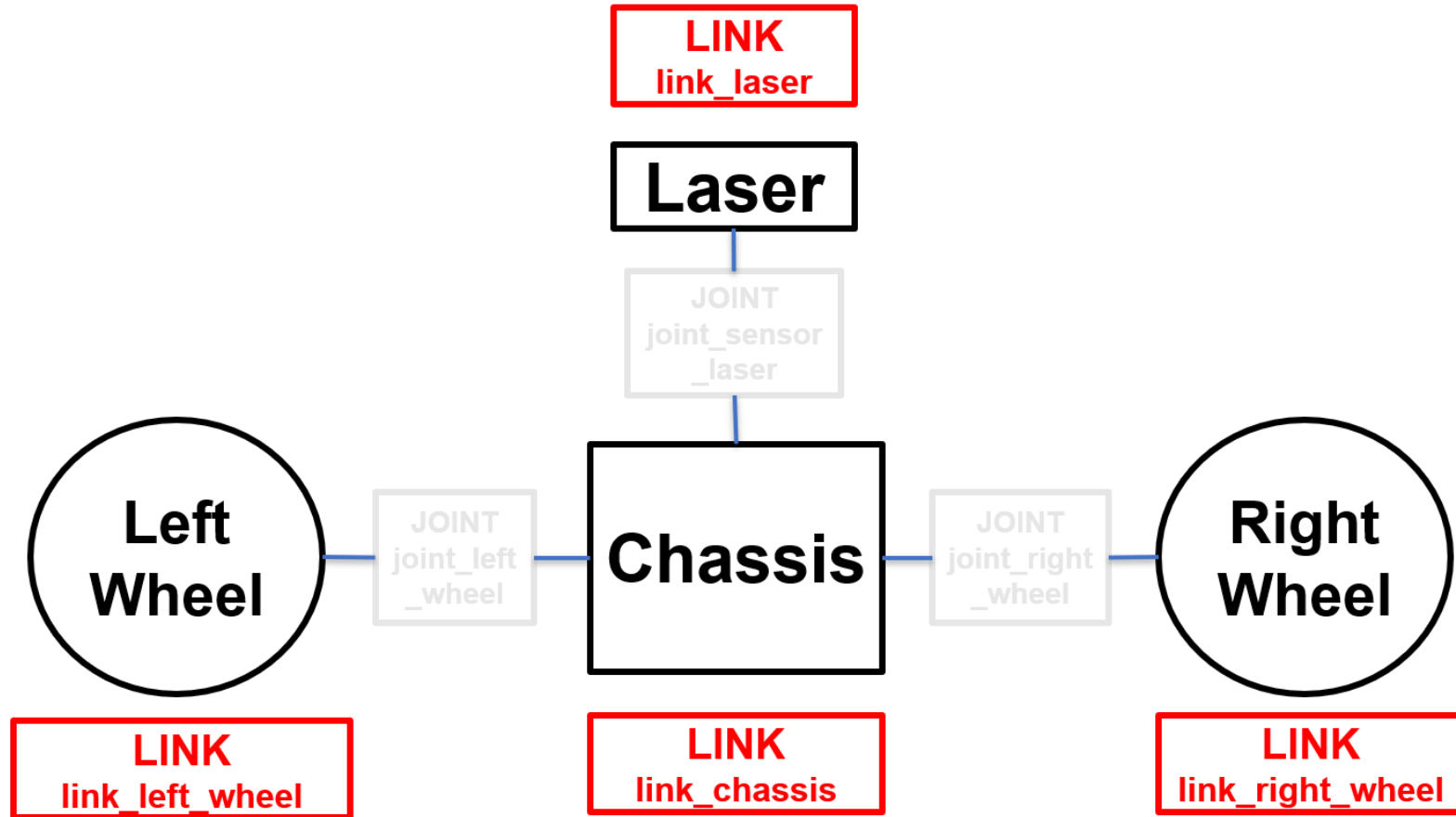
```
<gazebo>
  <plugin filename="libgazebo_ros_diff_drive.so" name="differential_drive_controller">
    <alwaysOn>true</alwaysOn>
    <updateRate>20</updateRate>
    <leftJoint>joint_left_wheel</leftJoint>
    <rightJoint>joint_right_wheel</rightJoint>
    <wheelSeparation>0.4</wheelSeparation>
    <wheelDiameter>0.2</wheelDiameter>
    <torque>0.1</torque>
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <robotBaseFrame>link_chassis</robotBaseFrame>
  </plugin>
</gazebo>
```



Gazebo Control (Cont.)

- To control the motion of the robot we can use the **keyboard_teleop** to publish motion commands using the keyboard.
- So re-launch the **gazebo.launch** file in a shell:
 > ***roslaunch my_robot gazebo.launch***
- In a second shell use the following command:
 > ***roslaunch teleop_twist_keyboard teleop_twist_keyboard.py***
- Now we can make the robot navigate with **keyboard** keys.

Gazebo Control (Cont.)





Gazebo Control (Cont.)

Now we are ready to add a sensor to our robot. We will add a sensor laser to our **my_robot.xacro** file. So we create a new link and joint element over the link chassis and, as the differential drive, we will add a **plugin** to our robot. So reopen the file and add at the end the new tags that looks like follows:

```

<link name="sensor_laser">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <mass value="1" />
    <inertia ixx= "0.00145833333"
              ixy = "0" ixz = "0"
              iyy= "0.00145833333"
              iyz = "0" izz= "0.00125" />
  </inertial>

```

Inertia Values Computation

$$I_{xx} = 1 \cdot (3 \cdot 0.05 \cdot 0.05 + 0.1 \cdot 0.1) / 12$$

$$I_{yy} = 1 \cdot (3 \cdot 0.05 \cdot 0.05 + 0.1 \cdot 0.1) / 12$$

$$I_{zz} = 1 \cdot (0.05 \cdot 0.05) / 2$$

Gazebo Control (Cont.)

```
<visual>
  <origin xyz="0 0 0" rpy="0 0 0" />
  <geometry>
    <cylinder radius="0.05" length="0.1"/>
  </geometry>
  <material name="white" />
</visual>
<collision>
  <origin xyz="0 0 0" rpy="0 0 0"/>
  <geometry>
    <cylinder radius="0.05" length="0.1"/>
  </geometry>
</collision>
</link>

<joint name="joint_sensor_laser" type="fixed">
  <origin xyz="0.15 0 0.05" rpy="0 0 0"/>
  <parent link="link_chassis"/>
  <child link="sensor_laser"/>
</joint>
```

Gazebo Control (Cont.)

```
<gazebo reference="sensor_laser">
  <sensor type="ray" name="head_hokuyo_sensor">
    <pose>0 0 0 0 0 0</pose>
    <visualize>true</visualize>
    <update_rate>20</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>720</samples>
          <resolution>1</resolution>
          <min_angle>-1.570796</min_angle>
          <max_angle>1.570796</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.10</min>
        <max>5.0</max>
        <resolution>0.01</resolution>
      </range>
      <noise>
        <type>gaussian</type>
        <mean>0.0</mean>
        <stddev>0.01</stddev>
      </noise>
    </ray>
  </sensor>
</gazebo>
```



Gazebo Control (Cont.)

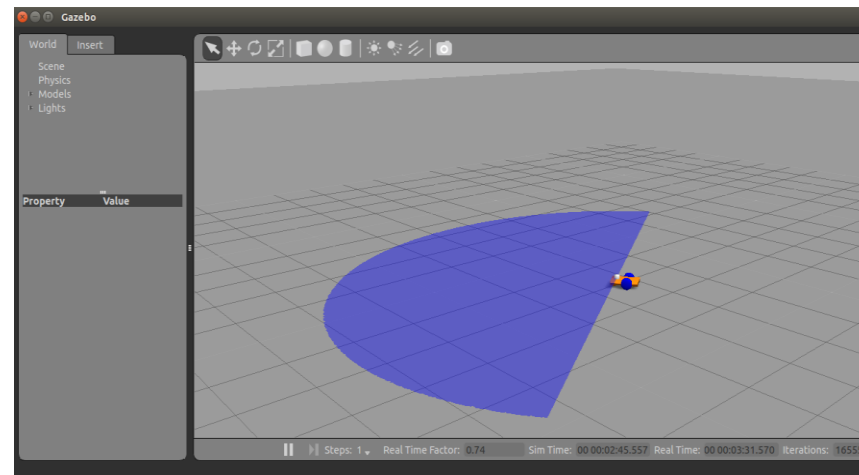
```
<plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
  <topicName>/my_robot/laser/scan</topicName>
  <frameName>sensor_laser</frameName>
</plugin>
</sensor>
</gazebo>
```

This code specifies many important parameters:

- **update_rate** : Controls how often (how fast) the laser data is captured
- **samples** : Defines how many readings are contained in one scan
- **resolution** : Defines the minimum angular distance between readings captured in a laser scan
- **range** : Defines the minimum sense distance and maximum sense distance. If a point is below minimum sense its reading becomes zero(0) and if a point is further than the maximum sense distance its reading becomes inf. The **range resolution** defines the minimum distance between 2 points such that two points can be resolved as two separate points.
- **noise** : This parameter lets us add gaussian noise to the range data captured by the sensor
- **topicName** : Defines the name which is used for publishing the laser data
- **frameName** : Defines the link to which the plugin has to be applied

Gazebo Control (Cont.)

- With this plugin incorporated in the urdf file we are now ready to simulate and visualize the laser scan in action.
- So re-launch the **gazebo.launch** file in a shell:
> roslaunch my_robot gazebo.launch
- In a second shell use the following command:
> rosrn teleop_twist_keyboard teleop_twist_keyboard.py



Gazebo in Ubuntu (Peng et al., 2023)

Summary

- Automation Simulation is a multidimensional field, involving the understanding of fundamental components and their applications across diverse industries.
- The simulation process relies on intricate models to accurately represent dynamic systems.
- Gazebo stands out as a versatile simulation software, offering features ranging from modeling to control.
- In the realm of Robotics and Automation in Industry, simulation plays a pivotal role in design and optimization.
- Gazebo's distribution, architecture, and modeling techniques, including distinctions like SDF vs URDF, contribute to its effectiveness.
- Proficiency in Gazebo extends to its features, schemes, and control mechanisms, making it a comprehensive tool for simulating various applications in the dynamic landscape of automation.



Review Questions

1. What are the fundamental components of Automation Simulation, and how do they contribute to the accuracy of simulations in diverse industries?
2. How does Gazebo, as a simulation software, differentiate itself in terms of features, modeling capabilities, and control mechanisms?
3. In what ways does simulation play a crucial role in the field of Robotics and Automation in Industry, particularly in the design and optimization processes?
4. Explain the distinctions between SDF and URDF file formats in the context of Gazebo modeling, and why are these distinctions important?
5. How does Dynamic Simulation add complexity to the modeling of dynamic systems, and why is it essential in the broader scope of automation simulation?

References

- Kumar, R. S., & Sundaresan, S. (2013). Mechanical finishing techniques for technical textiles. *Advances in the dyeing and finishing of technical textiles*, 135-153.
- Gierz, Ł., Warguła, Ł., Kukla, M., Koszela, K., & Zwiachel, T. S. (2020). Computer aided modeling of wood chips transport by means of a belt conveyor with use of discrete element method. *Applied Sciences*, 10(24), 9091.
- Bonetti, L. & De Cecco, M. (2019). *Robotic Perception and Action*.
- Kim, J., & Nam, Y. (2019). Study on the cooling effect of attached fins on PV using CFD simulation. *Energies*, 12(4), 758.
- Peng, G., Lam, T. L., Hu, C., Yao, Y., Liu, J., & Yang, F. (2023). Robot System Modeling. In *Introduction to Intelligent Robot System Design: Application Development with ROS* (pp. 139-189). Singapore: Springer Nature Singapore.



Lecture 08: Automation Verification

Objectives

- Gain a comprehensive understanding of the principles and processes involved in automation verification within the field of robotics.
- Define and recognize the specific requirements essential for successful automation verification in the realm of robotics.
- Investigate and comprehend the critical reasons and needs that drive the necessity for automation verification in the field of robotics.
- Evaluate and outline the advantages and positive outcomes associated with the implementation of automated verification processes in robotics.
- Grasp the concept of automated testing, including its advantages and limitations, and discern its role in the broader context of the test life cycle, tool acquisition, and decision-making processes.

Outlines

- Automation Verification in Robotics
- Requirements for Automation Verification in Robotics
- Necessity for Automation Verification in Robotics
- Benefits of Automated Verification in Robotics
- What is Automated Testing?
- Automated Testing Advantages
- Limitation of Automated Testing
- Automated Testing vs. Manual Testing
- Automated Test Life-Cycle Methodology (ATLM)
- Decision to Automate Test
- Test Tool Acquisition
- Test Process Analysis
- Test Planning
- Test Design
- Test Development
- Test Execution
- Automated Testing in CAR IMM Iasi
- TTCN-3
- Advantages of Using TTCN-3
- Black-Box Testing with TTCN-3
- Tux
- Other Automated Test Tools
- Automated Testing Examples
- TestEra Framework
- Steps of the TestEra Framework
- TestEra Spec
- TestEra Analysis
- TestEra Example
- Input Specification
- Correctness Criteria
- Counter-Examples
- Abstraction and Concretization Translations
- Concretization
- TestEra Case Studies
- Korat
- An Example: Binary Tree
- Finitization
- Non-isomorphic Instances for finBinaryTree
- Isomorphic Instances
- Generating Test Cases
- Isomorphism
- Generating Test Cases
- Using Contracts in Testing
- Korat Performance

Automation Verification in Robotics

- Automation verification in robotics refers to the process of confirming that a robotic system or automation solution meets its specified requirements and operates correctly.
- This involves assessing the performance, functionality, and safety of the robotic system through various testing and verification methods.
- The goal is to ensure that the system behaves as expected under different conditions and that it complies with the intended design and functionality.
- Automated testing is a broader concept that applies to various fields, including software development, where it is commonly used. In the context of robotics and automation, automated testing involves using automated tools and scripts to assess the performance and functionality of a system.

Automation Verification in Robotics (Cont.)

- In the context of robotics and automation, automated testing involves using automated tools and scripts to assess the performance and functionality of a system including unit testing, integration testing, and system testing, among others.
- The objective of automation verification is to improve efficiency, reliability, and repeatability of the testing process.
- Automation verification in robotics is a specialized form of automated testing tailored to the unique requirements of robotic systems.

Automation Verification in Robotics (Cont.)

There are some key differences between automation verification in **robotics** and **automated testing**:

Scope of Application:

- **Automation Verification in Robotics:** Primarily focused on verifying the performance, functionality, and safety of robotic systems and automation solutions.
- **Automated Testing:** A broader concept that can be applied to various domains, including software development, hardware testing, and, in this context, robotics.

Object of Testing:

- **Automation Verification in Robotics:** Focuses on the entire robotic system, including hardware, software, and the integration of components.
- **Automated Testing:** Can be applied to individual components (unit testing), the interaction between components (integration testing), or the entire system (system testing).

Automation Verification in Robotics (Cont.)

Purpose:

- **Automation Verification in Robotics:** Primarily concerned with ensuring that the robotic system meets its specified requirements and operates as intended.
- **Automated Testing:** Aims to identify defects, errors, and issues in the system to improve its overall quality and reliability.

Testing Methods:

- **Automation Verification in Robotics:** Involves a combination of physical testing, simulation, and other methods specific to robotics to verify the system's performance.
- **Automated Testing:** Encompasses a wide range of testing methods, including test automation tools, frameworks, and scripts that can be applied to different types of testing.



Requirements for Automation Verification in Robotics

The requirements for Automated Verification in Robotics are tailored to the specific characteristics of robotic systems:

- **Precision and Accuracy:** Ensuring that the robotic system performs tasks with the required precision and accuracy.
- **Sensory Input Validation:** Verifying the accuracy of sensory inputs, such as camera data or sensor readings, crucial for decision-making in robotic applications.
- **Movement and Navigation Testing:** Validating the robot's movement and navigation capabilities, including obstacle avoidance and path planning.
- **Real-time Performance:** Assessing the real-time performance of the robotic system, especially in applications where timing is critical.
- **Integration with External Systems:** Verifying seamless integration with external systems or devices that the robotic application may interact with.

Necessity for Automation Verification in Robotics

The necessity of Automated Verification in Robotics arises from the following considerations:

- **Complexity of Robotic Systems:** Robotic applications often involve intricate algorithms, sensor integrations, and complex control mechanisms, making manual testing impractical.
- **Consistent Performance:** Automated Verification ensures consistent and repeatable testing, crucial for maintaining the reliability of robotic systems over time.
- **Efficiency and Speed:** With the ability to run tests 24/7, Automated Verification significantly improves the efficiency and speed of the testing process compared to manual methods.
- **Risk Mitigation:** Automated testing helps identify potential issues and vulnerabilities in the early stages of development, reducing the risk of errors in real-world applications.

Benefits of Automated Verification in Robotics

Embracing Automated Verification in Robotics yields a multitude of benefits:

- **Reliability:** Automated tests provide consistent and reliable results, crucial for the dependability of robotic systems in diverse applications.
- **Time Efficiency:** The automated nature of testing significantly reduces the time required for verification, allowing for rapid iterations in the development process.
- **Cost Savings:** Automated Verification optimizes resource utilization, minimizing the need for extensive human testing efforts and associated costs.
- **Early Issue Detection:** Automated tests catch potential issues early in the development cycle, preventing the escalation of problems in real-world scenarios.
- **Enhanced Scalability:** As robotic applications scale in complexity, Automated Verification ensures scalability by facilitating the testing of diverse scenarios and functionalities.

What is Automated Testing?

- **Automated Testing:** The management and performance of test activities, to include the development and execution of test scripts having as objective the verification of test requirements, using an automated test tool.
- The automation of test activities reveals its greatest values in instances where test scripts are repeated or where test scripts subroutines are created and then invoked repeatedly by a number of test scripts.
- Given the continual changes and additions to requirements and software, automated tests serves as an important control mechanism to ensure accuracy and stability of the software through each build.

Automated Testing Advantages

- **Costs and Efficiency**
 - Detection of the errors that reached production phase (with regression tests)
 - Multiple users simulation
 - Reusable of the old scripts, creation of the new scripts is reduce
 - Automatic execution of performance tests in the beginning of the production, less costs for improving the performance
- **Time Economy**
 - Quick analysis in case of changing of environment parameters
 - Short duration of the testing cycles
 - Better estimation for test planning
 - A large number of tests can be executed over night
 - Quick generation of testing preconditions
- **Quality Increase**
 - Automatic compare of results
 - More consistent results due to repeating tests



Limitation of Automated Testing

- Most of the times an Automated Testing system cannot tell if something "looks good" on the screen or when a pictogram or a window is not displayed well
- There are a bunch of problems that can appear when trying to automate the testing process:
 - Unrealistic expectations (e.g. expectation that automated tests will find a lot of errors)
 - Poor testing experience
 - Maintenance of automated tests
- Automated testing will never replace definitely the manual testing
- Tests that should not be automated are:
 - Tests that are executed very rare
 - Tests that system is very unstable
 - Tests that can be verified easily manually but hardly automated
 - Tests that need physical interaction

Automated Testing vs. Manual Testing

- Pros of **Automated testing**
 - If a set of tests must be ran repeatedly, automation is a huge win.
 - It offers the possibility to run automation against code that frequently change to catch regressions.
 - It offers the possibility to add a large test matrix (e.g. different languages on different OS platforms).
 - Automated tests can be run the same time on different machines, whereas manual tests must be run sequentially.
 - It offers more time for the test engineer to invoke greater depth and breadth of testing, focus on problem analysis, and verify proper performance of software following modifications and fixes.
 - Combined with the opportunity to perform programming tasks, this flexibility promotes test engineer retention and improves his morale.
- Cons of **Automated testing**
 - Costs - Writing the test cases and writing or configuring the automate framework that is used costs more initially than running the test manually.
 - Some tests cannot be automated – manual tests are needed.

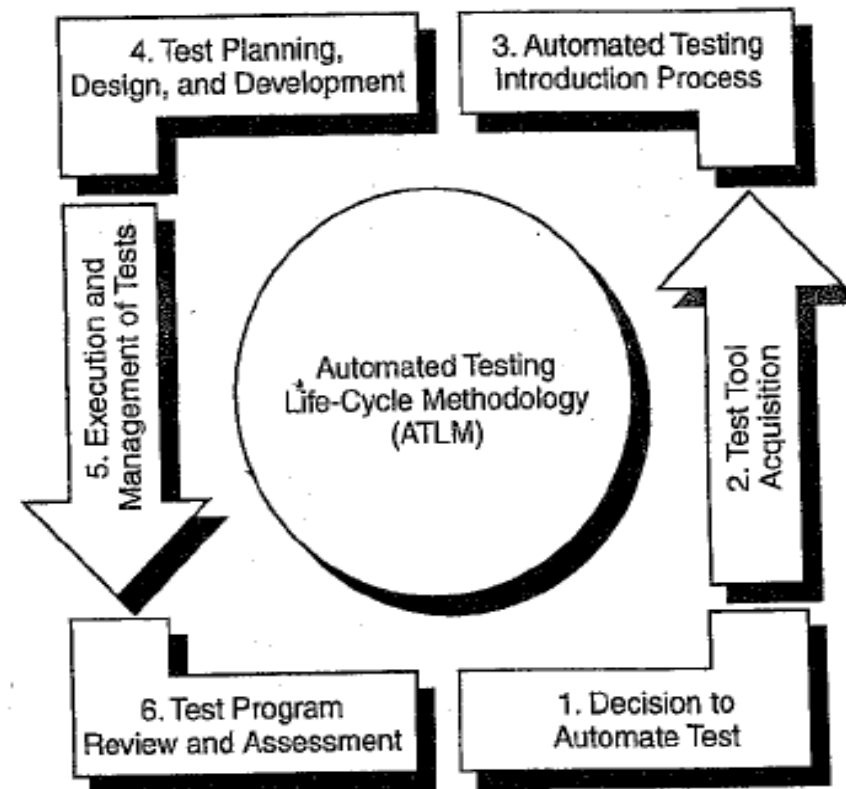


Automated Testing vs. Manual Testing (Cont.)

- Pros of **Manual testing**
 - If the test case runs once or twice most likely is a manual test. Less cost than automating it.
 - It allows the tester to perform more ad-hoc (random tests). Experience has proven that more bugs are found via ad-hoc (Experience Based Testing) than via automation testing. And more time the testers spends playing with the feature, the greater are the chances of finding real user problems.
- Cons of **Manual testing**
 - Running tests manually can be very time consuming
 - The manual tests are requiring more people and hardware
 - Each time there is a new build, the tester must rerun all required tests - which after a while would become very boring and tiresome.

Automated Test Life-Cycle Methodology (ATLM)

1. Decision to Automate Test
2. Test Tool Acquisition
3. Automated Testing Introduction Process
4. Test Planning, Design and Development
5. Execution and Management of Tests
6. Test Program Review and Assessment



Decision to Automate Test

Overcoming False Expectations

EXPECTATION

- Automatic test plan generation
- Test tool fits all
- Imminent test effort reduction
- Tool ease of use
- Universal application of test automation
- 100% test coverage

REALITY

- No tool can create automatically a comprehensive test plan
- No single tool can support all operating systems environments and programming languages
- Initial use of an automated test tool can actually increase the test effort
- Using an automated tool requires new skills, additional training is required
- Not all the tests required for a project can be automated (e.g. some test are physically impossible; time or cost limitation)
- It is impossible to perform an exhaustive testing of all the possible inputs (simple or combination) to a system

Decision to Automate Test (Cont.)

- | | |
|---|---|
| ▶ Production of a reliable system | ▶ Improved requirements definition |
| | ▶ Improved performance/load/stress testing |
| | ▶ Improved partnership with development team |
| | ▶ Improved system development life cycle |
| ▶ Improvement of the quality of the test effort | ▶ Improved build verification testing (smoke testing) |
| | ▶ Improved regression testing |
| | ▶ Improved multiplatform/software compatibility testing |
| | ▶ Improved execution of the repetitive tests |
| | ▶ Improved focus on advanced test issues |
| | ▶ Execution of tests that manual testing can't accomplish |
| | ▶ Ability to reproduce software defects |
| | ▶ After-hours testing |
| ▶ Reduction of test effort and minimization of schedule | |

Test Tool Acquisition

- Identify which of the various tool types suit the organization system environment, considering:
 - The group/department that will use the tool.
 - The budget allocated for the tool acquisition.
 - The most/least important functions of the tool etc.
- Choose the tool type according to the stage of the software testing life cycle
- Evaluate different tools from the selected tool category
- Hands-on tool estimation – request product demonstration (evaluation copy)
- Following the conclusion of the evaluation process, an evaluation report should be prepared

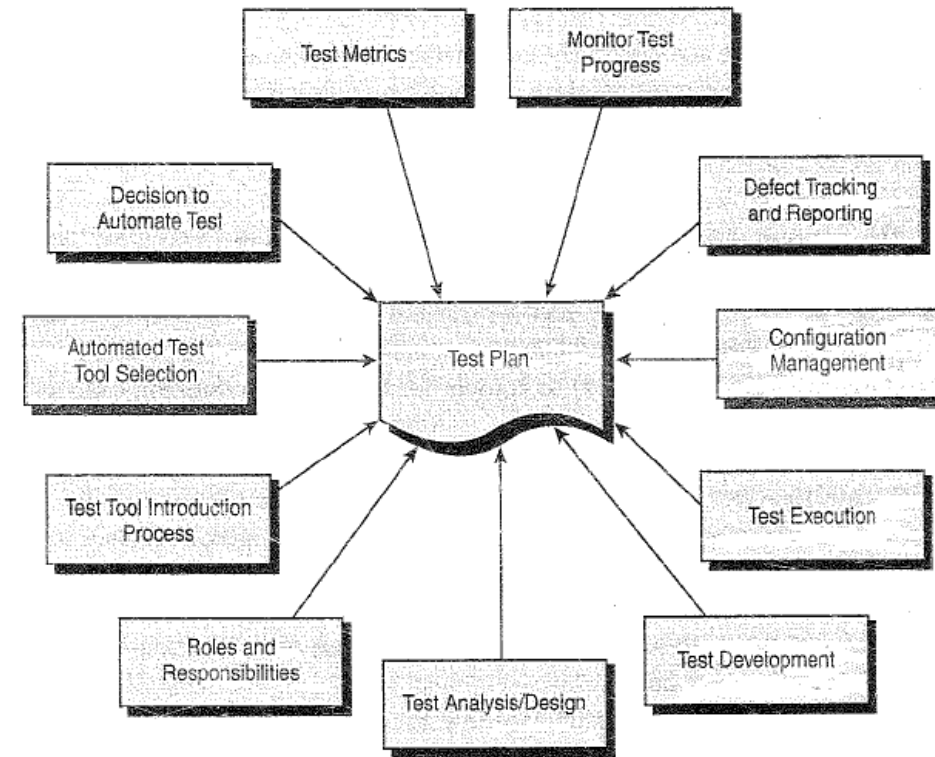
Business analysis phase	<ul style="list-style-type: none"> ▶ business modeling tools ▶ configuration management tools ▶ defect tracking tools
Requirements definition phase	<ul style="list-style-type: none"> ▶ requirements management tools ▶ requirements verifiers tools
Analysis and design phase	<ul style="list-style-type: none"> ▶ database design tools ▶ application design tools
Programming phase	<ul style="list-style-type: none"> ▶ syntax checkers/debuggers ▶ memory leak and run-time error detection tools ▶ source code or unit testing tools ▶ static and dynamic analyzers
Testing phase	<ul style="list-style-type: none"> ▶ test management tools ▶ network testing tools ▶ GUI testing tools (capture/playback) ▶ non-GUI test drivers ▶ load/performance testing tools ▶ environment testing tools

Test Process Analysis

Process review	<ul style="list-style-type: none">▶ Test process characteristics (goals, strategies, methodologies) have been defined and they are compatible with automated testing▶ Schedule and budget allows process implementation▶ The test team is involved from the beginning of SDLC
Test goals	<ul style="list-style-type: none">▶ Increase the probability that application under test will behave correctly under all circumstances▶ Increase the probability that application meets all the defined requirements▶ Execute a complete test of the application within a short time frame
Test objectives	<ul style="list-style-type: none">▶ Ensure that the system complies with defined client and server response times▶ Ensure that the most critical end-user paths through the system perform correctly▶ Incorporate the use of automated test tools whenever feasible▶ Perform test activities that support both defect prevention and defect detection▶ Incorporate the use of automated test design and development standards to create reusable and maintainable scripts
Test strategies	<ul style="list-style-type: none">▶ Defect prevention (early test involvement, use of process standards, inspection and walkthroughs)▶ Defect detection (use of automated test tools, unit/integration/system/acceptance test phase)

Test Planning

- The test planning element of the ATLM incorporates the review of all activities required in the test program
- It ensures that testing processes, methodologies, techniques, people, tools, schedule and equipment are organized and applied in an efficient way
- Key elements: planning associated with project milestone events, test program activities and test program-related documentation.
- The following must be accomplished:
 - the technical approach for these elements is developed;
 - personnel are assigned
 - performance timelines are specified in the test program schedule.
- Test planning is not a single event, but rather a process. It is the document that guides test execution through to a conclusion, and it needs to be updated frequently to reflect any changes.



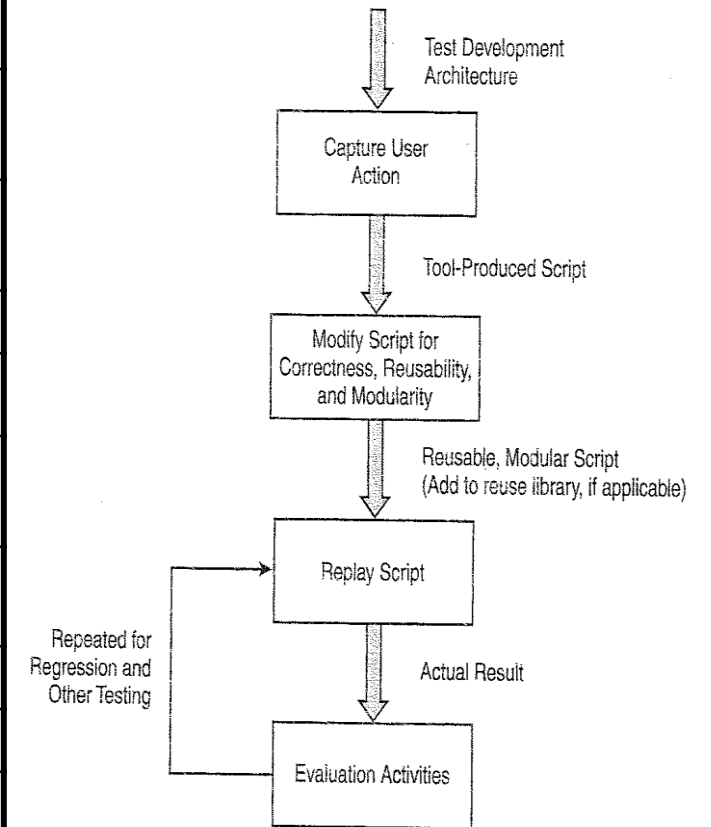
Test Design

- Test program model definition: identifies the test techniques that apply to the test program
- Test architecture definition: involves the selection of a test architecture model and population of model attributes
- Test procedure definition: groups logically the test procedures
- Automated/manual: test mapping (decision what to automate)

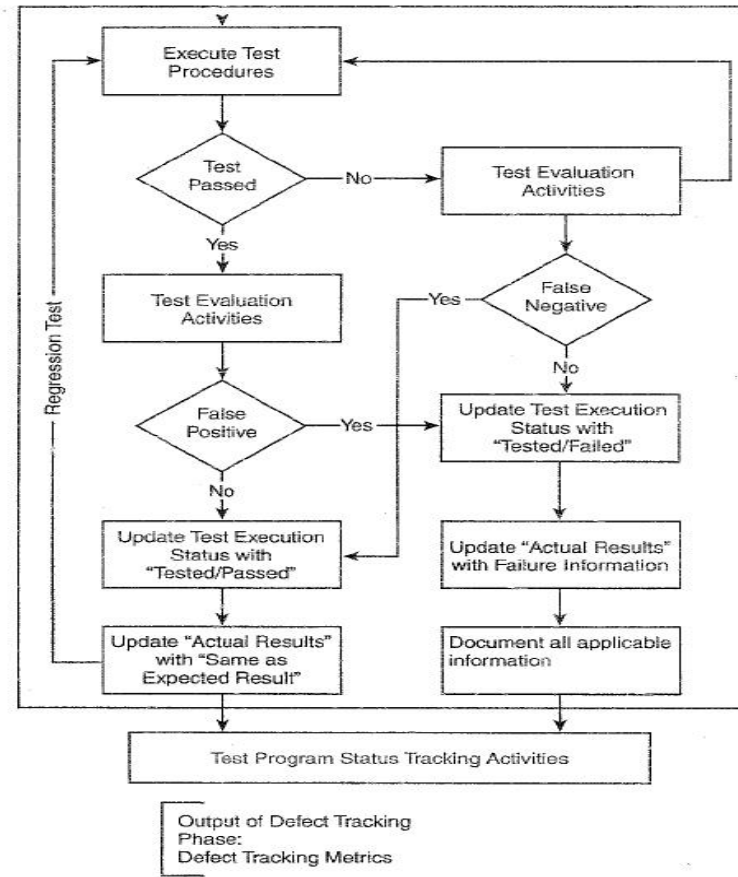
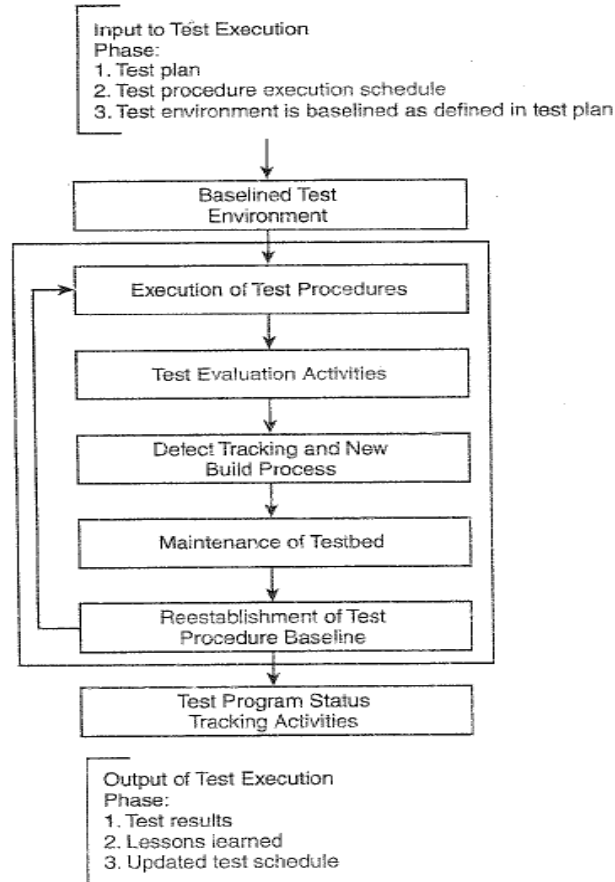
Black-Box Test Techniques	Automated Test Tools
▶ Random Testing	▶ GUI Test Tools
▶ Regression Testing	▶ GUI/Server Test Tools
▶ Stress/Performance Testing	▶ Load Test Tools
▶ Security Testing	▶ Security Test Tools
▶ Data Integrity Testing	▶ Data Analysis Tools
▶ Configuration Testing	▶ Multiplatform Test Tools
▶ Functional Testing	▶ Load/GUI/Server Test Tools
▶ User Acceptance Testing	▶ GUI Test Tools
▶ Usability Testing	▶ Usability Measurement Tools
▶ Alpha/Beta Testing	▶ Load/GUI/Server Test Tools
▶ Boundary Value Analysis	▶ Develop program code to perform tests
▶ Backup and Recoverability Testing	▶ Load/GUI/Server Test Tools

Test Development

Development Guideline Topics	Description
Design-to-Development Transition	Specify how design and setup activities will be translated into test development action
Reusable Test Procedures	Test procedures need to be reusable for highest test program return on investment
Data	Avoid hard-coding data values into scripts
Capture/Playback	Outlines on how to apply the use of capture/playback recording
Maintainable Test Procedures	A test procedure whose defects are easy to remove and can easily be adapted to meet new requirements
Test Script Documentation	Test script documentation is important for test procedure maintainability
Naming Standards	Defines the standard naming convention for test procedures
Modularity	Guidelines for creating modular test scripts
Global Files	Globally declared functions are available to any procedure
Constants	Use of constants in order to support maintainable procedure



Test Execution



Automated Testing in CAR IMM Iasi

- Requirements management : DOORS
- Test management tool: SiTempo
- Automated test tools: TUX, TTCN-3, Silk Test

TTCN-3

- **TTCN-3** is the **Testing and Test Control Notation** version **3**
- **Internationally standardized** testing language for formally defining test scenarios – design purely for testing
 - developed from 1999 – 2002 at the *European Telecommunications Standards Institute* (ETSI)
- ▶ A programming language that has been used for more than 20 years in standardization as well as industry.
- ▶ Area of Testing:
 - Conformance and Functionality Testing
 - Pre-integration Testing
 - Integration and Interoperability Testing
 - Regression Testing
 - Load/Stress Testing

```

testcase Hello_Bob () {
  p.send("How do you do?");

  alt {
    [] p.receive("Fine!") {
      setverdict(pass);
    }
    [else] {
      setverdict(inconc); // Bob asleep!
    }
  }
}

```

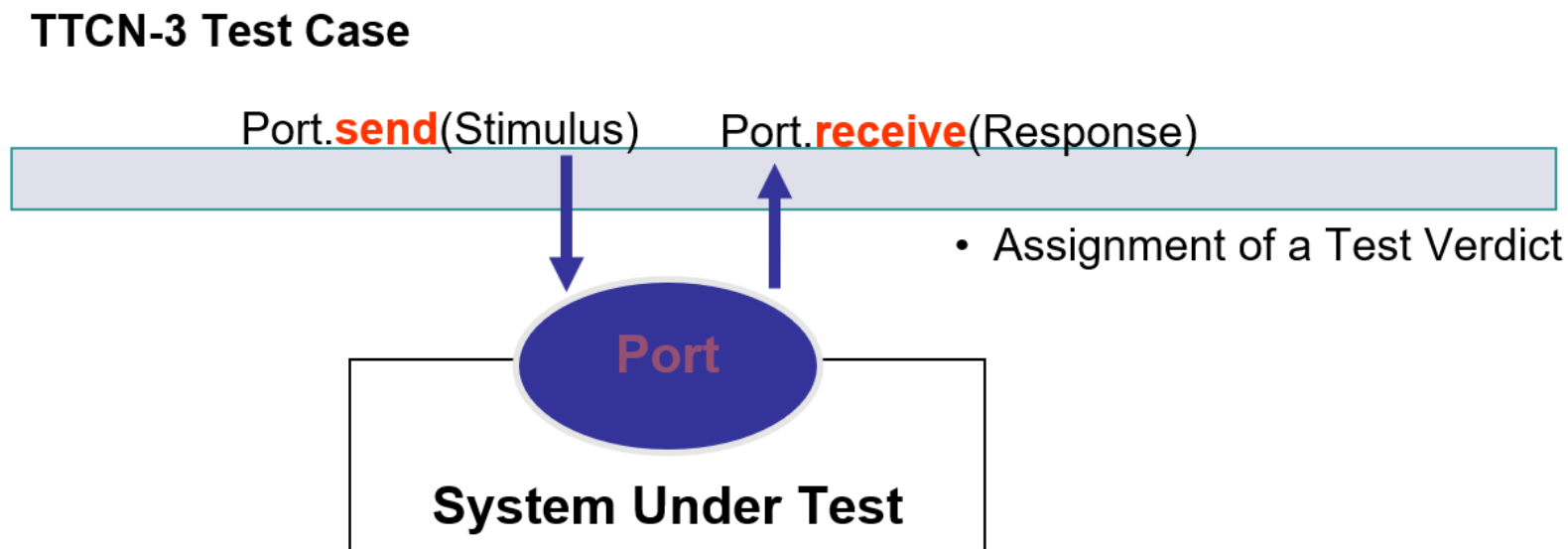
Advantages of Using TTCN-3

- Systematic product testing
- Large percentage of testing coverage
- Parallel testing and develop processes
- Automated testing so it can be run without user intervention
- Complex and multiple scenarios implementation
- Clear and exact bug reports
- Easy management of errors and bug fixing
- Reusable tests and consistent version validation
- Automated regression tests on all levels
- Control and exact status of product quality in all development stages



Black-Box Testing with TTCN-3

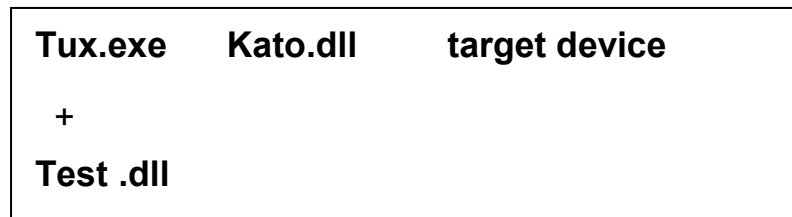
- TTCN-3 is applicable for all kinds of **black-box** testing for reactive and distributed systems.



Tux

The Tux test harness (Tux) is a 32-bit client/server test harness that executes test modules stored in dynamic-link library (.dll) files. Tux groups tests, maintains statistics, and provides a user interface (UI). With Tux, you can write clean, short, and platform-independent test cases without the overhead required to write a complete application. Tux was developed for real time tests, driver tests and is suitable for performance tests.

Most simple set-up for executing Tux tests on a target device:



Tux.exe - Test harness

Kato.dll - Logging engine

Properties of Test .dll:

- ▶ Test .dll contains test cases, test strategy, test sequence, in other words: test implementation and test intelligence
- ▶ Source code is written in C / C++
- ▶ Must export certain mandatory functions for Tux.exe

Other Automated Test Tools

Mercury Interactive - www.mercuryinteractive.com	Application performance management	<ul style="list-style-type: none"> ▶ <u>Topaz</u> – solution for performance management from the end-user perspective: real time alerts receiver, user problems correlation with system data, service levels management
	Test management	<ul style="list-style-type: none"> ▶ <u>TestDirector</u> – global solution for test process management, containing four modules: Requirements, Test Plan, Test Lab, Defects Manager
	Functional testing	<ul style="list-style-type: none"> ▶ <u>Astra FastTrack</u> – defect management tool for Web applications ▶ <u>Astra QuickTest</u> – tool that allows both the beginner and experienced testers to test the dynamic Web applications ▶ <u>WinRunner</u> – functional testing solution for GUI applications ▶ <u>QuickTest professional</u> - tool for automation process of functional and regression testing of the dynamic Web applications ▶ <u>XRunner</u> – automation functional testing of the X-Window applications
	Load testing	<ul style="list-style-type: none"> ▶ <u>LoadRunner</u> – load and performance testing management ▶ <u>Astra LoadTest</u> – quick solution for performance and scalability testing of the Web applications ▶ <u>LoadRunner TestCenter</u> – complete solution that allows multiple and concurrent testing management, from different locations. All the resources are centralized and accessible from a single control center

Other Automated Test Tools (Cont.)

Segue - www.seguate.com	Application performance management	<ul style="list-style-type: none"> ▶ SilkVision – solution for application behavior monitoring ▶ SilkTest – tool for application scalability and stress testing ▶ SilkPerformer – simulates the application performance and stress testing ▶ SilkPlan – process management package
Rational - www.rational.com	Application performance management Test management	<ul style="list-style-type: none"> ▶ Rational Suite TestStudio ▶ Rational Test RealTime ▶ Rational TestManager
Compuware Corporation www.compuware.com	Application performance management	<ul style="list-style-type: none"> ▶ Abend-AID ▶ File-AID ▶ PointForward ▶ QACenter ▶ XPEDITER
	Production Readiness Quality Assurance Development Integration and	<ul style="list-style-type: none"> ▶ Abend-AID ▶ NuMega ▶ QA Center ▶ File-AID ▶ Uniface ▶ Vantage

Automated Testing Examples

- TestEra is a specification-based functional (black-box) testing tool:
 - It requires the user to write input/output specifications.
- Korat is also a kind of functional (black-box) testing tool:
 - It requires the user to write a specification as a method in the class that is being testing.
- Both tools are used for unit testing:
 - They test of complex data structures.
- Both tools automatically generate test cases from specifications:
 - They exhaustively generate all non-isomorphic test cases within a given scope.

TestEra Framework

- TestEra is a framework for automated testing of Java programs.
- TestEra's main focus is unit testing of complex data structures.
 - Examples: Red-black trees, linked lists etc.
- TestEra has also been used in analyzing larger Java programs.
 - Examples: Alloy Analyzer, Intentional Naming System

TestEra Framework (Cont.)

- TestEra automatically generates all non-isomorphic test cases within a given input size:
 - The input size corresponds to the scope in Alloy specifications.
- TestEra evaluates the correctness criteria for the automatically generated test cases.
- TestEra uses Alloy and Alloy Analyzer to generate the test cases and to evaluate the correctness criteria.
- TestEra produces concrete Java inputs as counterexamples to violated correctness criteria.

TestEra Framework (Cont.)

TestEra framework requires the following:

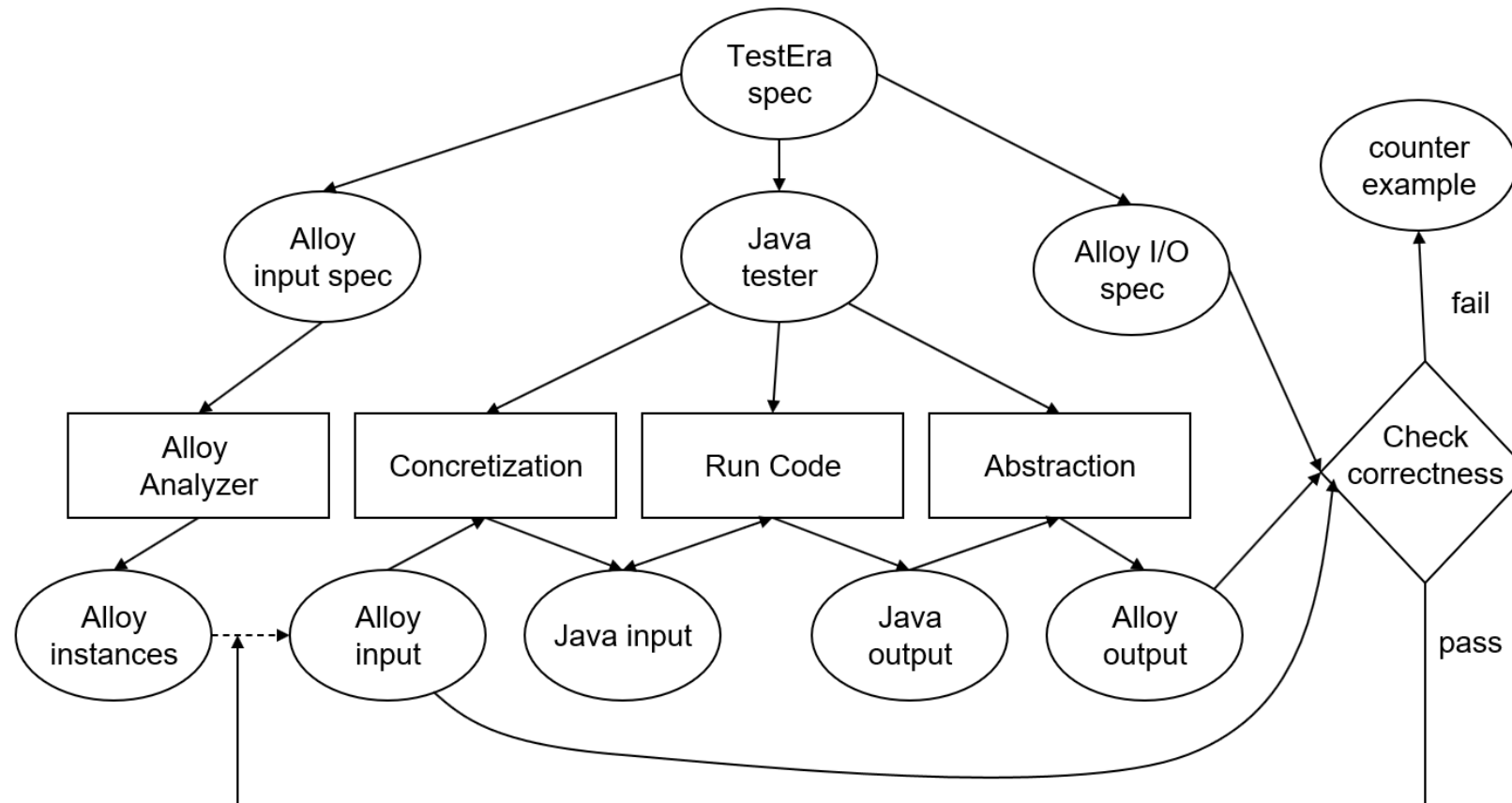
- A specification of inputs to a Java program written in Alloy precondition
- A correctness criterion written in Alloy class invariant and post-condition.
- An concretization function which maps instances of Alloy specifications to concrete Java objects.
- An abstraction function which maps the concrete Java objects to instances of Alloy specifications.



TestEra Framework (Cont.)

- TestEra generates all the non-isomorphic input instances using Alloy Analyzer:
 - Two instances are isomorphic if there is a one to one mapping between the atoms of the two instances which preserve all the relations.
- TestEra uses the concretization function to translate the instances of the Alloy specification to Java inputs:
 - These inputs form the test set .
- TestEra runs the program on the test set.
- TestEra maps the output produced by the program to Alloy using the abstraction function.
- TestEra uses the Alloy Analyzer to check the input and the output against the correctness criteria.

TestEra Framework (Cont.)



Steps of the TestEra Framework

- Identify a sequence of method calls to analyze
- Create an Alloy specification for the inputs of these methods
- Create an Alloy specification of the correctness criteria by relating the inputs to the outputs of these methods
- Define a concretization translation $a2j$ from an Alloy instance of the input specification to a Java input to these methods
- Define an abstraction translation $j2a$ from a Java output to an Alloy instance of the specification of the correctness criteria that relates inputs to outputs

TestEra Spec

- A TestEra specification is a combination of Alloy and Java code:
 - This specification is split to three files.
 - Alloy input specification
 - Java code for:
 - translating input instances from Alloy to Java
 - running the sequence of Java methods to test
 - translating the Java output back to Alloy
 - Alloy specification for the correctness criteria which relates the input values to the output values

TestEra Analysis

- TestEra uses Alloy Analyzer to generate all non-isomorphic instances of the Alloy input specification.
- Each instance is translated to Java input using concretization:
 - This forms the test case for the sequence of Java methods to be tested.
- The sequence of methods are run on this input.
- The produced output is translated using abstraction back to Alloy.
- The input and output instances are checked against the correctness criterion.
- If the check fails a counter-example is reported, otherwise next Alloy instance is used for testing.

TestEra Example

A recursive method for performing merge sort on acyclic singly linked lists Java:

```
class List {  
    int elem;  
    List next;  
    static List mergeSort(List l) { ... }  
}
```

Alloy:

```
module list  
import integer  
sig List {  
    elem: Integer,  
    next: lone List }
```

- Signature declaration introduces the List type with functions:
- elem: List → Integer
- next: List → List
- next is a partial function which is indicated by the keyword lone

Input Specification

```
module list
import integer
sig List {
  elem: Integer,
  next: lone List }

fun Acyclic(l: List) {
  all n: l.*next | lone n.~next // at most one parent
  no l.~next } // head has no parent

one sig Input in List {}

fact GenerateInputs {
  Acyclic(Input) }
```

- Sub-signature Input is a subset of list.
- It has exactly one atom which is indicated by the keyword one.

Correctness Criteria

```
fun Sorted(l: List) {  
  all n: l.*next | some n.next => n.elem <= n.next.elem }
```

```
fun Perm(l1: List, l2: List)  
  all e: Integer | #(e.~elem & l1.*next) =  
                  #(e.~elem & l2.*next) }
```

denotes cardinality of sets

```
fun MergeSortOK(i: List, o: List) {  
  Acyclic(o)  
  Sorted(o)  
  Perm(i, o) }
```

```
one sig Output in List {}
```

```
fact OutputOK {  
  MergeSortOk(Input, Output) }
```

Counter-Examples

- If an error is inserted in the method for merging where `(l1.elem <= l2.elem)` is changed to `(l1.elem >= l2.elem)`
- Then TestEra generates a counter example

Counterexample found:

Input List: 1 -> 1 -> 3 -> 2

Output List: 3 -> 2 -> 1 -> 1



Abstraction and Concretization Translations

- An abstraction function: j2a translate Java instance to Alloy instance.
- A concretization function: a2j translate Alloy instance to Java instance.
- These functions are written in Java by the user.

Concretization

- Concretization is implemented in two stages:
 1. Create a Java object for each atom in Alloy specification and store this correspondence in a map
 2. Establish the relationships among the Java objects created in the first step

TestEra Case Studies

- Red-Black Trees:
 - They tested the implementation of Red-Black trees in `java.util.TreeMap`
 - They introduced some bugs and showed that they can catch them with TestEra framework.
- Intentional Naming System:
 - A naming architecture for resource discovery and service location in dynamic networks
 - Found some bugs
- Alloy Analyzer:
 - Found some bugs in the Alloy Analyzer using TestEra framework

Korat

- Another automated testing tool:
 - Similar to TestEra but does not require extra Alloy specifications
 - Application domain is again unit testing of complex data structures
- It uses Java predicates to generate the test cases:
 - These are Java methods which return a boolean value
 - For example pre and post-conditions of methods
- Korat generates the test cases from pre and postconditions of methods.
- There is no need to write an extra specification if the class contract is written as Java predicates (like the JContractor approach).

Korat (Cont.)

- Korat uses the method precondition to automatically generate all nonisomorphic test cases up to a give small size:
 - Given a predicate and a bound on the size of its inputs Korat generates all nonisomorphic inputs for which the predicate returns true.
- Korat then executes the method on each test case and uses the method postcondition as a test oracle to check the correctness of output:
 - Korat exhaustively explores the bounded input space of the predicate but does so efficiently by monitoring the predicate's executions and pruning large portions of the search space.

An Example: Binary Tree

```
import java.util.*;
class BinaryTree {
    private Node root;
    private int size;
    static class Node {
        private Node left;
        private Node right;
    }
    public boolean repOk() {
        // this method checks the class invariant:
        // checks that empty tree has size zero
        // checks that the tree has no cycle
        // checks that the number of nodes in the tree is
        // equal to its size
    }
}
```

Finitization

- Korat uses a finitization description to specify the finite bounds on the inputs (scope)

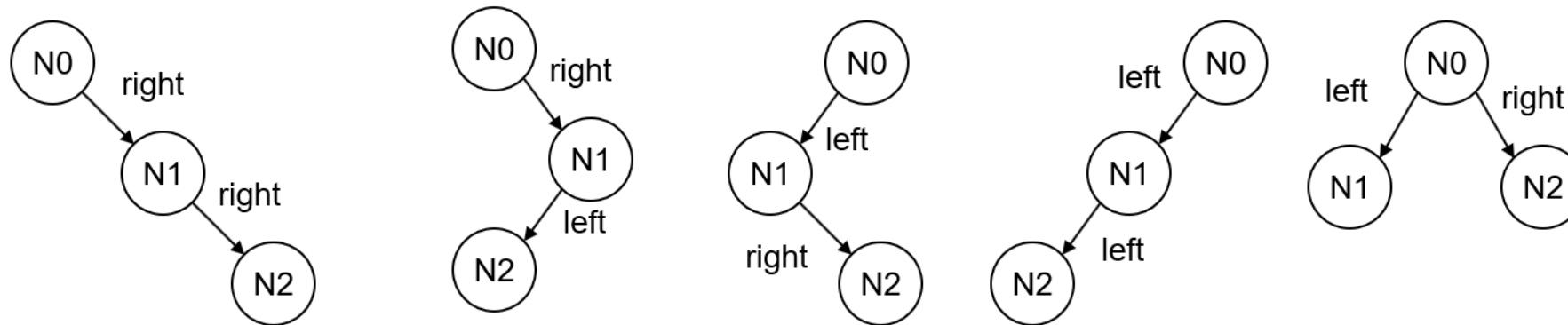
```
public static Finitization finBinaryTree(int NUM_node){
    Finitization f = new Finitization(BinaryTree.class);
    ObjSet nodes = f.createObjectSet("Node", NUM_node);
    nodes.add(null);
    f.set("root", nodes);
    f.set("size", NUM_node);
    f.set("Node.left", nodes);
    f.set("Node.right", nodes);
    return f;
}
```

Creates a set of objects of Type "Node" with NUM_node objects in the set

The value of size is set to NUM_node

- Korat automatically generates a finitization skeleton based on the type declarations in the Java code.
- Developers can restrict or extend this default finitization.

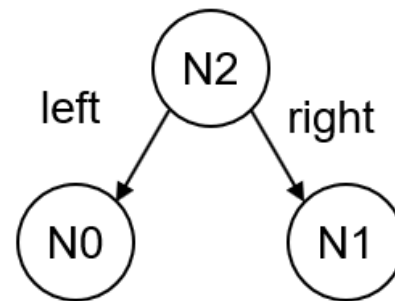
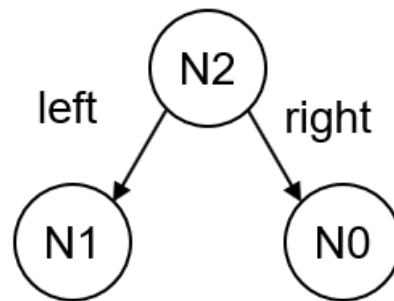
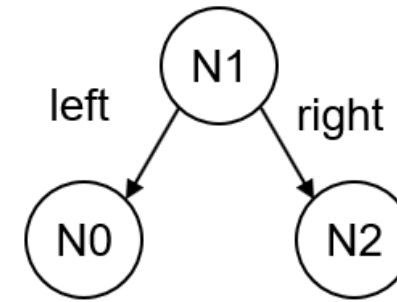
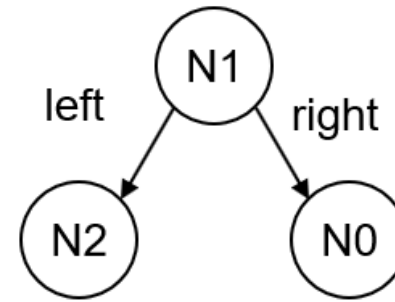
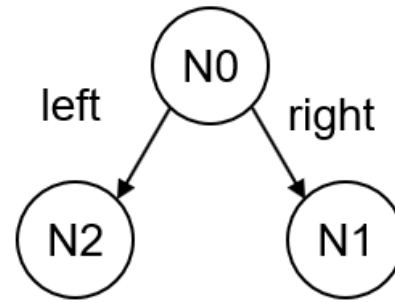
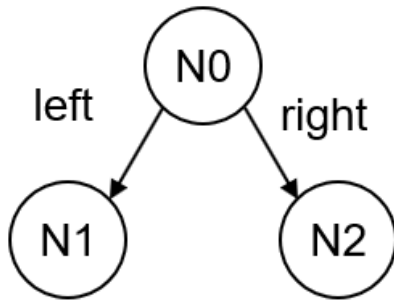
Non-isomorphic Instances for finBinaryTree



- Korat automatically generates non-isomorphic instances within a given bound
- Each of the above trees correspond to 6 isomorphic trees. Korat only generates one tree representing the 6 isomorphic trees.
- For `finBinaryTree(7)` Korat generates 429 non-isomorphic trees in less than a second



Isomorphic Instances





Generating Test Cases

- The crucial component of Korat is the test case generation algorithms.
- Consider the binary tree example with scope 3:
 - There are three fields: root, left, right.
 - Each of these fields can be assigned a node instance or null.
 - There is one root field and there is one left and one right field for each node instance.
 - Given n node instances, the state space (the set of all possible test cases) for the binary tree example is:
$$(n+1)^{2n+1}$$
 - Most of these structures are not valid binary trees:
 - They do not satisfy the class invariant
 - Most of these structures are isomorphic (they are equivalent if we ignore the object identities).

Generating Test Cases (Cont.)

- There are two techniques Korat uses to generate the test cases efficiently:
 1. Korat only generates non-isomorphic test cases.
 2. Korat prunes the state space by eliminating sets of test cases which do not satisfy the class invariant.



Isomorphism

- The isomorphism definition used in Korat is the following:
 - O_1, O_2, \dots, O_n are sets objects from n classes.
 - $O = O_1 \cup O_2 \cup \dots \cup O_n$
 - P : the set of consisting of null and all values of primitive types that the fields of objects in O can contain.
 - $r \in O$ is a special root object.
 - Given a test case C , O_C is the set of objects reachable from r in C
- Two test cases C and C' are *isomorphic* iff there is a permutation π on O , mapping objects from O_i to objects from O_i for all $1 \leq i \leq n$, such that:

$$\forall o, o' \in O_C . \forall f \in \text{fields}(o) . \forall p \in P.$$

$$o.f == o' \text{ in } C \text{ iff } \pi(o).f == \pi(o') \text{ in } C' \text{ and } o.f == p \text{ in } C \text{ iff } \pi(o).f == p \text{ in } C'$$

Isomorphism (Cont.)

- In Korat isomorphism is defined with respect to a root object.
 - for example `this`
- Two test cases are defined to be isomorphic if the parts of their object graphs reachable from the root object are isomorphic.
- The isomorphism definition partitions the state space (i.e. the input domain) to a set of isomorphism partitions:
 - Korat generates only one test case for each partition class.

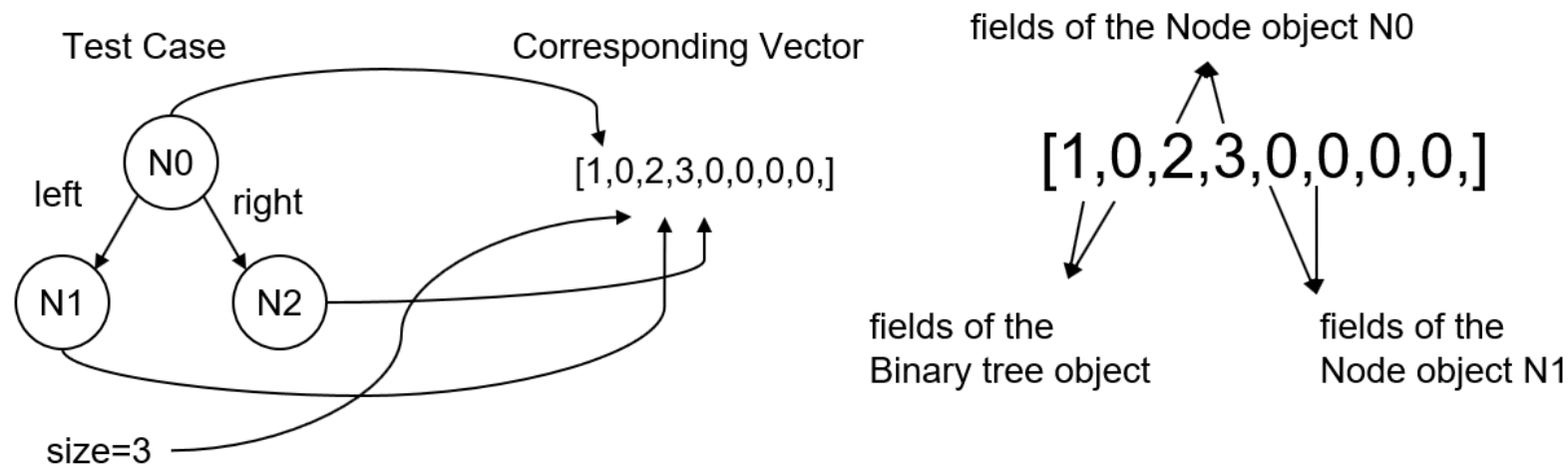
Generating Test Cases

- Korat only generates the test cases which satisfies the input predicate: class invariant and the precondition
- Korat explores the state space efficiently using backtracking
 - It does not generate all instances one by one and check the input predicate
 - It prunes its search of the state space based on the evaluation of the input predicate
 - If the method that checks the input predicate returns false without checking a field then there is no need to generate test cases which assign different values to that field
 - In order to exploit this, Korat keeps track of the fields that are accessed before the predicate returns false
 - For this to work well, predicate method should return false as soon as it detects a violation



Generating Test Cases (Cont.)

- Korat orders all the elements in every class domain and every field domain.
- Each test case is represented as a vector of indices into the corresponding field domains.
 - The class domain is ordered as $N0 < N1 < N2$.
 - The field domains for root, left and right are ordered as $null < N0 < N1 < N2$.
 - The size domain has one element which is 3.





Generating Test Cases (Cont.)

- Search starts with a candidate vector set to all zeros.
- For each candidate vectors, Korat sets fields in the objects according to the values in the vector.
- Korat then invokes repOk (i.e., class invariant) to check the validity of the current candidate.
 - During the execution of repOk, Korat monitors the fields that repOk accesses, it stores the indices of the fields that are accessed by the repOk (field ordering).
 - For example, for the binary tree example, if the repOk accesses root, N0.left and N0.right, then the field ordering is 0, 2, 3.
- Korat generates the next candidate vector by backtracking on the fields accessed by repOk.
 - First increments the field domain index for the field that is last in the field-ordering.
 - If the field index exceeds the domain size, then Korat resets that index to zero and increments the domain index of the previous field in the field ordering.

Generating Test Cases (Cont.)

- Korat achieves non-isomorphic test case generation using the ordering of field domains and the vector representation.
- While generating the test cases, Korat ensures that the indices of the objects that belong to the same class domain are listed in nondecreasing order in the generated candidate vectors.
- This means that during backtracking, Korat looks for fields:
 - that precede the field that is accessed last
 - that have an object from the same class domain as the field that is accessed last.
 - and makes sure that the object assigned to the field that is accessed last is higher in the ordering than those objects.



Using Contracts in Testing

- Korat checks the contracts written in JML on the generated instances.

```
//@ public invariant repOk(); //class invariant

/*@ requires has(n)                // precondition
   @ ensures !has(n)                // postcondition
   @*/
public void remove(Node n) {
    ...
}
```

- Korat uses JML tool-set to translate JML annotations into runtime Java assertions.



Using Contracts in Testing (Cont.)

- Given a finitization, Korat generates all non-isomorphic test cases within the given scope (defined by the finitization) that satisfy the class invariant and the pre-condition.
- The post-conditions and class invariants provide a test oracle for testing:
 - For each generated test case, execute the method that is being tested and check the class invariant and the post-condition.
- Korat uses JML tool-set to automatically generate test oracles from method post-conditions written as annotations in JML.

Korat Performance

- Checking a BinaryTree implementation with scope 8 takes 1/53 seconds, with scope 11 takes 56.21 seconds, with scope 12 takes 233.59 seconds.
- Test case generation with Korat is more efficient than the test case generation with Alloy Analyzer.

Summary

- Automation Verification in Robotics involves ensuring the accuracy and reliability of automated processes in robotic systems.
- Requirements for Automation Verification in Robotics include defining specific criteria and standards to validate the efficiency of automated systems.
- The Necessity for Automation Verification in Robotics arises from the need to enhance the dependability and safety of robotic operations through systematic verification.
- Benefits of Automated Verification in Robotics include improved efficiency, reduced errors, and increased confidence in the performance of robotic systems.
- Automated Testing is a process of using software tools to execute tests, with advantages such as speed and repeatability, limitations in handling complex scenarios, and differences from manual testing in terms of execution methodology.



Review Questions

1. What are the key principles behind automation verification in the context of robotics?
2. How does automation verification contribute to the reliability of robotic systems?
3. What specific criteria and standards are crucial for effective automation verification in robotics?
4. How do these requirements ensure the efficiency of automated systems in robotics?
5. What advantages are associated with implementing automated verification processes in the field of robotics?
6. How do these benefits contribute to the overall performance and dependability of robotic systems?
7. What distinguishes automated testing from manual testing in terms of execution methodology?
8. When is it more appropriate to use automated testing over manual testing, and vice versa?
9. What is TTCN-3, and how does it play a role in automated testing?
10. What advantages does the use of TTCN-3 offer in comparison to other automated testing approaches?

References

- Palani, N. (2021). Automated Software Testing with Cypress. CRC Press.
- Bultan, T. (2008). Testing, Automated Testing. 272: Software Engineering Fall 2008
- Ropota, A. (2016). Automated Testing. Continental.
- Khurshid, S., & Marinov, D. (2004). TestEra: Specification-based testing of Java programs using SAT. Automated Software Engineering, 11, 403-434.
- Boyapati, C., Khurshid, S., & Marinov, D. (2002). Korat: Automated testing based on Java predicates. ACM SIGSOFT Software Engineering Notes, 27(4), 123-133.
- Berner, S., Weber, R., & Keller, R. K. (2005, May). Observations and lessons learned from automated testing. In Proceedings of the 27th international conference on Software engineering (pp. 571-579).
- Marinov, D., & Khurshid, S. (2001, November). TestEra: A novel framework for automated testing of Java programs. In Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001) (pp. 22-31). IEEE.
- Marinov, D., & Khurshid, S. (2001, November). TestEra: A novel framework for automated testing of Java programs. In Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001) (pp. 22-31). IEEE.
- Boyapati, C., Khurshid, S., & Marinov, D. (2002). Korat: Automated testing based on Java predicates. ACM SIGSOFT Software Engineering Notes, 27(4), 123-133.