Task Oriented Programming and Service Algorithms for Smart Robotic Cells

Stefano Trapani

Academic Supervisor: Prof. Marina Indri
Company Supervisor: Ing. Rosario Cassano
Converting standard production lines into smart factories, without changing the initial layout of the line

Two approaches:
1. Automatic offline programming methodology
2. Advanced functionalities implemented in standard industrial manipulators
Motivation and goals

- Converting standard production lines into smart factories, without changing the initial layout of the line

- Two approaches:
  1. Automatic offline programming methodology
  2. Advanced functionalities implemented in standard industrial manipulators

### Classic approach
- Sharp skilled programmers with a very high knowledge of the process
- Possible problems are solved by the experience of the programmer (e.g., how to satisfy cycle time constraints, avoid collisions, etc.)
- Time consuming and hence suitable for cells used to perform the same process for long time (low reconfigurability)

### Automatic Task-Oriented approach
- More intuitive automatic programming approach
- Soft skilled programmers must specify the features of the robotic cell and a structured set of tasks defining the process (CAD softwares)
- Fast process allowing high level of cell reconfigurability
Motivation and goals

- Converting standard production lines into *smart factories*, without changing the initial *layout* of the line

- **Two approaches**:  
  1. Automatic *offline programming* methodology  
  2. Advanced *functionalities* implemented in *standard industrial manipulator*

  - Increase the *smartness* of the robotic cell  
  - Increase the number of *applications*
Motivation and goals

- Converting standard production lines into smart factories, without changing the initial layout of the line
- Two approaches:
  1. Automatic offline programming methodology
  2. Advanced functionalities implemented in standard industrial manipulator

PART 1
Development of a task-based robot programming approach, that automatizes the programming of a generic robotic cell, providing as a result the work-flow defining the required process

PART 2
Development of a set of service algorithms based on the information already available in standard industrial robots
- Definition of a generic task using a **minimal set of basic actions**
- Analysis of a wide range of **industrial applications** provided by the main robot constructors in order to find a **common set of features**

### Tasks

1. **Arc welding**
   - **Cosmetic sealing**
   - **Polishing and deburring**
   - **Laser welding/cutting**
   - **Plasma cutting and water jet**
   - **Spot welding**

2. **Machine tending**
   - **Handling**
   - **Processing machining**
   - **Press brake bending**
   - **Interpress**
   - **Foundry**

3. **Assembly**
   - **Packaging**
   - **Painting**

---

- Requiring a path
- Involving only the motion of the work-piece
- Needing further elaborations to obtain a work-path

---

S. Trapani
After a proper pre-elaboration the applications belonging to the third class may fall into one of the first two classes.

Two main types of tasks can be defined: **Standard tasks** and **Special tasks**.

**Special tasks**

- Assembly
- Packaging
- Painting

**Standard tasks**

1. Arc welding
   - Cosmetic sealing
   - Polishing and deburring
   - Laser welding/cutting
   - Plasma cutting and water jet cutting
   - Spot welding

2. Machine tending
   - Handling
   - Processing machining
   - Press brake bending
   - Interpress
   - Foundry

---

S. Trapani
After a proper pre-elaboration the applications belonging to the third class may fall into one of the first two classes.

Two main types of tasks can be defined: **Standard tasks** and **Special tasks**.

Only standard tasks are managed by the **task model** in which:
- the real machines performing a process along a predefined path using a specific tool are denoted as **workers**.
- the real machines equipped with a proper gripper carrying the work-piece from a starting point to a final point are denoted as **positioners**.

**Standard tasks**

Processes carried out on a specific path or devoted to carry the work-piece.
Task-Oriented Programming - Approach

A three-step approach

CAD Design

- Physical definition of the robotic cell (robots, objects, workpieces)
- Definition of the tasks (type, paths, required tools)
- Generation of the description file (e.g., a xml file) used as input of the Task-Oriented Programming module

Task-Oriented Programming

- Automatic definition of the sequence of tasks and related path planning of the required process

Deploy into the robotic line

- Translation of the output of the Task-Oriented Programming into a program specific for the adopted robot controller

S. Trapani
Matlab script providing an xml file containing the information necessary to run the automatic task-oriented programming tool.

Automatic definition of the sequence of tasks and related path planning of the required process.

Simulator of the Comau control system provided by Comau.
Task-Oriented Programming

Automatic definition of the sequence of tasks and related path planning of the required process.
Automatic definition of the sequence of tasks and related path planning of the required process.

**Task-Oriented Programming**

**Pre Processing**
- **Process Model**
  - **Process Optimization**
    - **Path Planning**

**CORE**

1) **Mapping**
   - FLG Generator
   - SCG Generator

2) **Merging**
   - T1
   - ST1, ST2
   - PATH 1, PATH 2, PATH 3

3) **Searching Algorithm**
   - AND_split, TASK1, AND_joint
   - TASK2
   - TASK3
   - TASK4
   - TASK5
   - TASK6
   - TASK7
   - OR_split
   - TASK8, TASK9

4) **Optimization**
   - Local
   - Global

S. Trapani
The task model-based approach allows to take into account both physical constraints and functional ones between machineries and tasks.

- **Assumption:** task as a sequence of **four basic phases** (or by a subset of them): i) **picking** the workpiece, ii) **positioning** the work-piece within a proper sub-set of the working-area, iii) **working**, iv) **placing** the work-piece.

- **Physical constraints:** the adoption of some machineries can be limited because of their location or their physical characteristics; such constraints can be modeled by a proper graph called **Spatial Constraints Graph** (SCG).

- **Functional constraints:** the set of relations between the required tasks and the available machineries (e.g., task \( \text{task}_1 \) can be performed by machine \( \text{machine}_1 \) ), can be modeled by a proper graph called **Functional Link Graph** (FLG).

---

**Diagram:**

1) **Mapping**

- FLG Generator
- SCG Generator

2) **Merging**

3) **Searching Algorithm**

4) **Optimization**

- Local
- Global
A set of *logical entities* and a *three-level task descriptor* (Tasks, Sub-Tasks and Paths) have been developed to build the FLG and the SCG within a *mapping* process.

<table>
<thead>
<tr>
<th>Entities</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Buffers</strong></td>
<td>Real objects used to store the work-piece</td>
</tr>
<tr>
<td><strong>Positioners</strong></td>
<td>Real objects able to grip the work-piece</td>
</tr>
<tr>
<td><strong>Workers</strong></td>
<td>Real objects able to perform a specific process</td>
</tr>
<tr>
<td><strong>Objects</strong></td>
<td>Real objects that need to be processed</td>
</tr>
<tr>
<td><strong>Virtual</strong></td>
<td>Synchronization and physical connections</td>
</tr>
</tbody>
</table>

The **PATH** element corresponds to the *minimum action* which can be executed by a machinery. A complex task can be obtained by a proper *sequence* of PATHs.
The entities can be connected according to two criteria: 1) the type of the task and 2) the level of possible interaction between entities.

The two models are obtained by mapping the real objects of the robotic cell into the corresponding logical entities, and then applying a set of rules that allow to create the connections between the entities.

**FLG rule**
A Worker and a PATH element can be linked only if the functional constraints are satisfied.

**SCG rules**
- Two entities can be linked only if the spatial constraints are satisfied.
- Worker can be linked only to Positioner.
- Buffer can be linked only to Positioner.
- A pair of entities can be linked only if the spatial constraints are satisfied.
- Positioners can be linked with other Positioners only if explicitly required.
Remarks on FLG

- The **Functional Link Graph** (FLG) defines the possible relations between each **PATH** element and the available **workers**

- Different scenarios are taken into account
  1. **Several workers** can perform the **same task**
  2. Some tasks need to be synchronized
  3. The work-piece is physically modified during the process
Remarks on FLG

- The **Functional Link Graph** (FLG) defines the possible relations between each **PATH** element and the available **workers**

- Different scenarios are taken into account
  1. Several workers can perform the same task
  2. Some **tasks** need to be **synchronized**
  3. The work-piece is physically modified during the process
Remarks on FLG

- The **Functional Link Graph** (FLG) defines the possible relations between each **PATH** element and the available **workers**

- Different scenarios are taken into account
  1. Several workers can perform the same task
  2. Some tasks need to be synchronized
  3. The **work-piece is physically modified** during the process

  - Three possible cases: 1) joining, 2) splitting, 3) none

<table>
<thead>
<tr>
<th>action_type</th>
<th>effect on the model</th>
</tr>
</thead>
<tbody>
<tr>
<td>joining</td>
<td>Decreasing of the number of object entities</td>
</tr>
<tr>
<td>splitting (rejection)</td>
<td>Number of object entity unchanged</td>
</tr>
<tr>
<td>splitting (division)</td>
<td>Increasing of the number of object entities</td>
</tr>
<tr>
<td>none</td>
<td>Number of object entity unchanged</td>
</tr>
</tbody>
</table>
The **High Level Model** is thus obtained joining the SCG and the \( n \) FLGs (one for each object)
All the possible work-flows carrying out the process are represented using a specific model called **Work Flow Model** (WFL).

The WFM is a recursive model defined by **five basic blocks**. The **TASK** block can be used to combine basic blocks to create a more complex one, thanks to its recursive nature. It is also used to define the basic actions.

<table>
<thead>
<tr>
<th>WFM block</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND_Split</td>
<td>Open a parallel execution</td>
</tr>
<tr>
<td>OR_Split</td>
<td>Open a mutual exclusion execution</td>
</tr>
<tr>
<td>AND_Join</td>
<td>Close a parallel execution</td>
</tr>
<tr>
<td>OR_Join</td>
<td>Close a mutual exclusion execution</td>
</tr>
<tr>
<td>TASK</td>
<td>Recursive basic blocks or a basic task</td>
</tr>
</tbody>
</table>
All the possible work-flows carrying out the process are represented using a specific model called **Work Flow Model** (WFL)

The WFM is a recursive model defined by **five basic blocks**. The TASK block can be used to combine basic blocks to create a more complex one, thanks to its recursive nature. It is also used to define the basic actions

<table>
<thead>
<tr>
<th>Basic Tasks</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task_Pick</td>
<td>Picking action</td>
</tr>
<tr>
<td>Task_Place</td>
<td>Placing action</td>
</tr>
<tr>
<td>Task_Exec</td>
<td>Working carried out by a Worker</td>
</tr>
<tr>
<td>Task_FlyPass</td>
<td>Passage of the work-piece</td>
</tr>
</tbody>
</table>

![Diagram of basic tasks](image)
All the possible work-flows carrying out the process are represented using a specific model called **Work Flow Model** (WFL)

The WFM is a recursive model defined by **five basic blocks**. The TASK block can be used to combine basic blocks to create a more complex one, thanks to its recursive nature. It is also used to define the basic actions

<table>
<thead>
<tr>
<th>Complex block</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TASK_parallel</strong></td>
<td>one AND Split block, one AND Join block and n TASK Blocks</td>
</tr>
<tr>
<td><strong>TASK_mutex</strong></td>
<td>one OR Split block, one OR Join block and n TASK blocks</td>
</tr>
<tr>
<td><strong>TASK_st_exec</strong></td>
<td>is defined by one TASK block followed by n OR Join blocks</td>
</tr>
</tbody>
</table>
The WFM can be created automatically by applying a specific searching algorithm to the HLM. The algorithm is divided into two flows, the blue one that at each iteration selects the next tasks to be performed, and the red one that computes all the corresponding work-flows. While scrolling the HLM a set of translation rules are applied in order to build the WFM.
Task-Oriented Programming – Searching Algorithm

3) Searching Algorithm

S. Trapani
The algorithm is based on a DFS search algorithm, applied on a not oriented connected graph (the SCG corresponds to \( g(V, E) \)) for \( k \) times, where \( k \) is the number of PATHs required by the process. The complexity can be approximated as \( O(k \cdot n \cdot m) \) where \( n = |V| \) and \( m = |E| \).
Local Optimization: the optimization process is not aware to be possibly included in a wider production line

- **Two levels** of optimization must be managed:
  - at path planning level (low level)
  - at work-flow level (high level)

- **Low Level Optimization**: the path planning process is carried out for each task of the WFM; such process can include optimality constraints

- **High Level Optimization**: the work-flow at minimal cost can be selected using the output of the Low Level Optimization process to weight the WFM nodes. An AO* algorithm can be used.

Global Optimization: the optimization process takes into account the whole production line in order to achieve a global optimal result. A high level manager is required
Integration with OTE methodology

Integration of the proposed methodology in a general optimization framework based on the usage of Key Performance Indicators (KPI) called OTE/OEE, in collaboration with the Università Politecnica delle Marche (UNIVPM)

Fundamental configurations for OTE series, parallel, assembly, expansion

we construct from workflow

Tree structure having self-similar nodes

Interpretation of the cells (work unit, job)

\[ OEE = A_{\text{eff}} \times P_{\text{eff}} \times Q_{\text{eff}} \]
Integration of the proposed methodology in a general optimization framework based on the usage of Key Performance Indicators (KPI) called OTE/OEE, in collaboration with the Università Politecnica delle Marche (UNIVPM)

Fundamental configurations for OTE
- series, parallel, assembly, expansion

Interpretation of the cells (work unit, job)
\[ OEE = A_{eff} \times P_{eff} \times Q_{eff} \]

Original Interpretation
\[ OEE: A_{eff} \times P_{eff} \times Q_{eff} \rightarrow [0,1] \]

- **A_{eff}:** Availability
  - breakdowns, setup, adjustment, maintenance

- **P_{eff}:** Performance
  - reduced speed, idling, stoppages

- **Q_{eff}:** Quality
  - defects, rework, and yield
Integration with OTE methodology

Integration of the proposed methodology in a general optimization framework based on the usage of Key Performance Indicators (KPI) called OTE/OEE, in collaboration with the Università Politecnica delle Marche (UNIVPM)

**Fundamental configurations for OTE**
- series, parallel, assembly, expansion

**Interpretation of the cells (work unit, job)**
\[ OEE = A_{eff} \times P_{eff} \times Q_{eff} \]

**Adopted Interpretation**
\[ OEE: A_{eff} \times P_{eff} \times Q_{eff} \rightarrow [0,1] \]

- **A_{eff}: Availability**
  - faults or stoppings ratio, from probability distributions (MTBF, programmed maintenance, etc.)

- **P_{eff}: Performance**
  - related to the cycle time required to execute the given task

- **Q_{eff}: Quality**
  - to take into account the energy consumption

4) Optimization
- Local
- Global

S. Trapani
29
Integration with OTE methodology

Automatic Conversion Algorithm

OTE systems’ tree

WFM

Local

Global

4) Optimization

S. Trapani
ORE Methodology

### Mapping Table

<table>
<thead>
<tr>
<th>Cell objects</th>
<th>HML entity</th>
<th>HML symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ball</td>
<td>Object1</td>
<td>OBJ1</td>
</tr>
<tr>
<td>box 1</td>
<td>Buffer1 (input)</td>
<td>B1</td>
</tr>
<tr>
<td>box 2</td>
<td>Buffer1 (output)</td>
<td>B2</td>
</tr>
<tr>
<td>Racer 7 - 1A</td>
<td>Positioner1</td>
<td>P1</td>
</tr>
<tr>
<td>N34 110 - 2.2</td>
<td>Positioner2</td>
<td>P2</td>
</tr>
</tbody>
</table>

**Mapping between real object and HLM entities**

### Results on Task-Oriented programming

- **OREL** Simulator
- **Aeff, Peff, Qeff**
- **Mapping Table**
- **Motion condition**
- **OEE**

### Recursive OTE Methodology

**Different motion conditions**
The improvements are communicated at each iteration
On each iteration the appropriate suggested OTE is implemented
Once by favouring solutions for $P_{\text{eff}}$ and $A_{\text{eff}}$
The other by favouring $Q_{\text{eff}}$
The choice depends on the overall policy in the process
Both the policies lead to improvements of the process
Results on Task-Oriented programming

- Software implementation of the Task-Oriented Programming approach (without the optimization step)
- Building of the WFM for some simple but realistic robotic cells (e.g., Pick&Place and welding applications).
- Simulation tests using the Comau robot simulator ORL

Case Study:
Application - phase1: spot welding process in PATH1 and PATH2; phase2: arc welding process in PATH3; Starting point: Collection point#1; End point: Collection point#3

Robotic cell: i) four robots equipped with a tool compatible with the required tasks, ii) two robots with a gripper suitable to pick the adopted work-piece and iii) three collection points used to place the work-piece when necessary
Motivation and goals

- Converting standard production lines into **smart factories**, without changing the initial **layout** of the line

- **Two approaches:**
  1. Automatic **offline programming** methodology
  2. **Advanced functionalities** to be implemented in **standard industrial manipulator**

**PART 1**
Development of a task-based robot programming approach, that **automatizes** the programming of a generic robotic cell, providing as a result the work-flow defining the required process

**PART 2**
Development of a set of service algorithms based on the information already available in standard industrial robots
Improve the accuracy of the robot dynamic model

\[ M(q)\ddot{q} + C(q, \dot{q})\dot{q} + \tau_f(\dot{q}) + g(q) = \tau \]

\[ \tau_{res} = \tau - \hat{\tau} \]

Friction modelling and identification is a well known problem in robotics, especially at low velocity; its solution allows
- Improvement of the control performances
- Enhancement of the robustness of all the applications based on the comparison between the actual motor currents and the estimated ones

General framework for friction identification is proposed
- Applicable to different manipulators
- Handling the data acquisition and parameters identification phases

The friction model is based on a previous static model with further improvements to roughly approximate some dynamic features of friction

Extension of the framework for dynamic friction model
- Based on the LuGre friction model
- Improvements of the model for the industrial context
Friction Identification Framework – Data Acquisition

User program written using the COMAU programming language

Standard C5G controller for COMAU manipulators

Output file containing all the acquired data

---

Dynamic model: \[ M(q)\ddot{q} + C(q, \dot{q})\dot{q} + \tau_f(\dot{q}) + g(q) = \tau \]

Rules:

1. Only one joint per time must be moved
2. Movements must be performed in the wider possible position range in order to increase the part of the motion executed at constant velocity
3. The number of measurements at low velocity must be a good trade-off between acquisition time and model accuracy
**Friction Identification Framework – Data Acquisition**

User program written using the COMAU programming language

**PDL2**

Standard C5G controller for COMAU manipulators

**Output file containing all the acquired data**

**Dynamic model:**

\[
M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + \tau_f(\dot{q}) + g(q) = \tau
\]

**Rules:**
1. Only one joint per time must be moved
2. Movements must be performed in the wider possible position range in order to increase the part of the motion executed at constant velocity
3. The number of measurements at **low velocity** must be a good trade-off between acquisition time and model accuracy
Friction Identification Framework – Data Acquisition

**User program**
written using the COMAU programming language

**Standard C5G controller** for COMAU manipulators

**Output file** containing all the acquired data

**Dynamic model:**
\[ M(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_f(q) + g(q) = \tau \]

**Rules:**
1. Only one joint per time must be moved
2. Movements must be performed in the wider possible position range in order to increase the part of the motion executed at constant velocity
3. The number of measurements at **low velocity** must be a good trade-off between acquisition time and model accuracy

A subsequent **cleaning** phase allows to remove the gravity component

S. Trapani
Cleaning the data in order to eliminate gravity contribution and to select only data acquired at **constant velocity**

**Splitting** the data in order to standardize input format

Computation of **statistical information** which will be used to define feasible **bounds** of the friction torques

\[ \frac{1}{N} \sum_{i=1}^{N} v_{f,i} \theta \]
Friction Identification Framework - Identification

\[ \bar{\tau}_{f,j} = \frac{1}{N} \sum_{i=1}^{N} \tau_{f,j}^{(i)} \]

\[ \sigma_{\tau_{f,j}} = \sqrt{\frac{\sum_{i=1}^{N} (\tau_{f,j}^{(i)} - \bar{\tau}_{f,j})^2}{N}} \]

\[ b(v_j) = \bar{\tau}_{f,j} \pm k \cdot \max \left( \left[ \sigma_{\tau_{f,1}}, \sigma_{\tau_{f,2}}, \ldots, \sigma_{\tau_{f,m}} \right] \right) \]

Feasibility of friction values by means of a pair of bounds for each velocity \( v_j \)

S. Trapani
Friction Identification Framework - Identification

Definition of the set of velocities in which the **validity** of the model is guaranteed

\[
VR \in \{v: v_{lim} \leq v \leq v_{max+}, v > 0\} \cup \{v: v_{max-} \leq v \leq -v_{lim}, v < 0\}
\]

\[
v_{lim} = \frac{p}{100} v_{0+}
\]
Identification of the friction parameters

\[ \tau_f(v) = \tau_s \frac{\pi}{2} \arctan(v K_v) + \tau_{sc} \frac{\pi}{2} \arctan(v \delta) + \tau_v v + \tau_{nlv} v^2 \frac{\pi}{2} \arctan(v K_v) \]

Assigning a known value to \( \delta \) the identification problem turns into a Linear In Parameters (LIP) problem, which can be solved using Least Squares (LS) method

\[ \theta = (\Phi \Phi^T)^{-1} \Phi \ T_f \]

The optimization is based on the computation of the LS algorithm for each value of \( \delta \) between \( \delta_{min} \) and \( \delta_{max} \) with step \( \varepsilon \)
\[ \tau_f(v) = \tau_s \frac{\pi}{2} \arctan(v K_v) + \tau_{sc} \frac{\pi}{2} \arctan(v \delta) + \tau_v v + \tau_{nlv} v^2 \frac{\pi}{2} \arctan(v K_v) \]
Two different models are computed for acceleration and deceleration phases. A proper filtering action manages the values provided by the two models:

\[ \tau_{fa}(v) = f_{lim} \left( \tau_s \frac{\pi}{2} \arctan(v K_v) + \tau_{sc} \frac{\pi}{2} \arctan(v \delta) + \tau_v v + \tau_{nlv} v^2 \frac{\pi}{2} \arctan(v K_v) \right) \]

\[ \tau_{fb}(v) = f_{lim} \left( \tau_s \frac{\pi}{2} \arctan(v K_v) + \tau_{sc} \frac{\pi}{2} \arctan(v 1000\delta) + \tau_v v + \tau_{nlv} v^2 \frac{\pi}{2} \arctan(v K_v) \right) \]

A limiting function is defined using \( v_{lim} \):

The limitation is active only for velocities outside \( VR \). The typical torque peak at low velocity is cut during decelerations, in order to roughly reproduce the hysteretic behavior of friction.

\[ f_{lim}(v) = \begin{cases} 1, & |v| \geq v_{lim} \\ \frac{v}{|v|}, & |v| < v_{lim} \end{cases} \]
Experimental tests were performed using a standard COMAU Smart NS12. Comparison is carried out using two performance indices: the Root Mean Square Error (RMSE) and the Mean Value (MV).

The best results are obtained for the third joint in the first test, with a reduction of 39% for RMSE and 28% for MV.
The forces due to a wrong definition of the robot dynamic model parameters like the payload are computed

Inverse static equation

\[ F = \left( J^T(q) \right)^{-1} \tau_{res} \]

Residual torque

\[ \tau_{res} = \tau - \hat{\tau} \]

- Robot must be far from singularities
- Check of the matrix condition number
The forces due to a **wrong definition** of the robot dynamic model parameters like the **payload** are computed.

### Inverse static equation

\[ F = \left( J^T(q) \right)^{-1} \tau_{res} \]

\[ F = [f_x \ f_y \ f_z \ N_x \ N_y \ N_z] \]

### Residual torque

\[ \tau_{res} = \tau - \hat{\tau} \]

### Payload error

\[ \text{Payload}\_error = \frac{f_z}{g} \]

### Gravity acceleration

**Check of the matrix condition number**

**Robot must be far from singularities**

---

S. Trapani
The forces due to a **wrong definition** of the robot dynamic model parameters like the **payload** are computed.

**Inverse static equation**

\[ F = \left( J^T(q) \right)^{-1} \tau_{res} \]

\[ F = [f_x \quad f_y \quad f_z \quad N_x \quad N_y \quad N_z] \]

**Residual torque**

\[ \tau_{res} = \tau - \hat{\tau} \]

**Payload error**

\[ \text{Payload_error} = \frac{f_z}{g} \]

- **Ideal conditions**: \( f_z \) is constant
- **Real conditions**: \( f_z \) is varying with \( q, \dot{q}, \ddot{q} \)

Check of the matrix **condition number**

- Robot must be far from **singularities**

- Extracting **DC component** of \( f_z \) to separate the **payload error** from model errors

\[ P(i) = \frac{P(i-1) \cdot N + P(i)}{N + 1} \]
Good results are obtained for a COMAU NJ 130

- Real payload 130 kg declared payload 0 kg
- Mean value 129.6 kg
- Standard deviation 0.31 kg
- Relative mean error 0.31%
A rapid and robust collision detection is a fundamental issue for the safety of a robotic cell in any industrial environment, not only in the next future when a high presence of collaborative robots is expected, but also in current, standard production lines.

Goals and benefits:
- Preservation of the robot mechanical parts in case of impact
- Monitoring of the correct execution of the programmed task
  - Detection of failures whose effects are similar to those of a collision

Industrial requirements:
- Avoidance of false collision alarms
- Wide applicability and portability of the SW implementation
- Avoiding specific customizations
- Using only the sensors that usually equip an industrial manipulator
Approach based on the residual current

Detection based on a time varying threshold function

The threshold is given by the sum of two terms:
- An estimate of the absolute value of the model error in absence of collisions $\hat{m}_{err}(t)$
- The sensitivity of the virtual sensor $\text{Coll}_{bias}(t)$

Different approaches to compute the model error ($\hat{m}_{err}(t)$) are adopted when:
- The current is in the steady state
- The current is not in the steady state
Collision Detection - Approach

Behaviour of the estimate of the model error for the first joint of a COMAU NJ4 170
Collision Detection - FSM

- \( I_i \)
- \( I_{DM,i} \)

Moving
- Synchronous currents

Reversing
- \( I_i \) changes its trend

Reversing_DM
- \( I_{DM,i} \) changes its trend

Steady
- \( I_{DM,i} \) almost constant

Impulse
- Unexpected impulse of \( I_i \)

Safe state

Unsafe state
The best threshold is used by a proper identification of the collision sensitivity $\text{Coll}_{\text{ident}}(t)$.

The following adaptation law is applied for the $i$-th joint after the user request:

$$S(t) = \hat{\text{m}}_{\text{err}}(t) + \text{Coll}_{\text{bias}}(t)$$

$$\text{Coll}_{\text{bias},i}(t_{\text{adapt}}) = \overline{\text{Coll}}_{\text{Ident},i}^{(k)} + e^{(-t_{\text{adapt}}/\tau_a)} \left( \text{Coll}_{0,i} - \overline{\text{Coll}}_{\text{Ident},i}^{(k)} \right)$$
The best threshold is used by a proper identification of the collision sensitivity $\text{Coll}_{\text{ident}}(t)$.

The following adaptation law is applied for the $i$-th joint after the user request

$$S(t) = \hat{m}_{\text{err}}(t) + \text{Coll}_{\text{bias}}(t)$$

$$\text{Coll}_{\text{bias},i}(t_{\text{adapt}}) = \overline{\text{Coll}}_{\text{Ident},i}^{(k)} + e^{-t_{\text{adapt}}/\tau_a} \left( \text{Coll}_{0,i} - \overline{\text{Coll}}_{\text{Ident},i}^{(k)} \right)$$

$$\text{Coll}_{\text{Ident},i}(t) = |R_i(t)| - \hat{m}_{\text{err},i}(t)$$

$$\text{Coll}_{0,i} = \text{Coll}_{\text{bias},i}$$

$t_{\text{adapt}} = 0$
Collision Detection - Results

<table>
<thead>
<tr>
<th>Performed Tests</th>
<th>DT Adapt (s)</th>
<th>DT Basic (s)</th>
<th>Average Reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td>top → bottom EOS</td>
<td>0.026</td>
<td>0.106</td>
<td>75.5</td>
</tr>
<tr>
<td>top → bottom NR</td>
<td>0.024</td>
<td>0.186</td>
<td>87.1</td>
</tr>
<tr>
<td>left → right EOS</td>
<td>0.010</td>
<td>0.044</td>
<td>77.3</td>
</tr>
<tr>
<td>left → right NR</td>
<td>0.010</td>
<td>0.046</td>
<td>78.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performed Tests</th>
<th>Ax</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>top → bottom EOS</td>
<td></td>
</tr>
<tr>
<td>top → bottom NR</td>
<td></td>
</tr>
<tr>
<td>left → right EOS</td>
<td>▪</td>
</tr>
<tr>
<td>left → right NR</td>
<td>▪</td>
</tr>
</tbody>
</table>

Basic

Adaptive
Post-collision reaction and Manual Guidance

- Manage both collision reaction and manual guidance
- Sensor-less approach
- Distinguish accidental collisions from intended human-robot contacts
Manage both collision reaction and manual guidance

Sensor-less approach

Distinguish accidental collisions from intended human-robot contacts

- Stop the robot as fast as possible
- Reduction of the impact force

- Robot compliant to the applied forces
Post-collision reaction and Manual Guidance

- Manage both collision reaction and manual guidance
- Sensor-less approach
- Distinguish accidental collisions from intended human-robot contacts
Post-collision reaction and Manual Guidance

Waiting → Monitoring → Collision Reaction

2Hz Low-Pass filter

\[ I_{mg(j)}(t_k) = a_1 \cdot I_{mg(j)}(t_{k-1}) + a_2 \cdot I_{mg(j)}(t_{k-2}) + b_1 \cdot R(j)(t_k) \]

\[ \tau_{mg}(t) = K_1 \cdot I_{mg}(t) \]

\[ F_{mg}(t) = (J(q)^T)^{-1} \tau_{mg}(t) \]

\[ F_{mg(ax)}(t) > T h_{1H(ax)}(t) \]

\[ F_{mg(ax)}(t) < T h_{1L(ax)}(t) \]

10Hz High-Pass filter

\[ I_{cd(j)}(t) = c_1 \cdot I_{res(j)}(t_k) + c_2 \cdot I_{res(j)}(t_{k-1}) + c_3 \cdot I_{res(j)}(t_{k-2}) + c_4 \cdot R(j)(t_{k-3}) \]

\[ T h_{cd(j)}(t) = k_{cd_c(j)} + k_{cd_v(j)} \frac{|\dot{q}_j(t)|}{\dot{q}_{j,max}} + k_{cd_a(j)} \frac{\ddot{q}_j(t)}{\ddot{q}_{j,max}} \]
Post-collision reaction and Manual Guidance

\[
\begin{align*}
& \frac{1}{N_{mg}} \sum_{k=1}^{N_{mg}} \left( F_{mg,s(ax)}(t - k - 1) \right) < C_{flat(ax)} \\
& \text{Th}_2 L_{(ax)}(t) < F_{mg(ax)}(t) < \text{Th}_2 H_{(ax)}(t)
\end{align*}
\]

S. Trapani
Post-collision reaction and Manual Guidance

- Waiting
- Monitoring
- Manual Guidance
- Collision Reaction

\[ \Delta p_{mg} = K_{mg}^{-1} \cdot F_{mg} \]

\[ \Delta p_{cr}(t) = K_{cr}^{-1} \cdot F_{cr}(t) \]

First phase

- mg_exit
- mg_enter
- cr_enter
- cr_exit

after 1s
Post-collision reaction and Manual Guidance

\[ \Delta p_{mg} = K_{mg}^{-1} \cdot F_{mg} \]

Waiting \[ \rightarrow\] Monitoring \[ \rightarrow\] Manual Guidance

\( mg\_exit \)

\( mg\_enter \)

Collision Reaction

\( cr\_enter \)

\( cr\_exit \)

Second phase

\[ N_{dec} = a_s P_{cr} + b_s \]

S1: \[ P_{cr} = \left( \| K_{cr}^{-1} \Delta F_{cr} \| / \Delta t \right) \]

S2: \[ P_{cr} = \| K_{cr}^{-1} \Delta F_{cr} \| \]

S3: \[ P_{cr} = 0 \]
Conclusions

On robotic cell programming:
- Task model able to take into account both physical and functional constraints
- Automatic task oriented programming based on the task model
- Collaboration with UNIVPM to integrate the task programming approach with the OTE methodology
- Verification of the methodology for realistic robotic cells

On service algorithms:
- Improvement of the robot dynamic model using a new framework for friction identification
- Adaptive collision detection algorithm (implemented in COMAU controller)
- Payload check (implemented in the COMAU controller)
- Post collision reaction
- Manual guidance
Publications

- **Task-Oriented Programming**
  2. Integration of a production efficiency tool with a general robot task modeling approach, Indri, Marina; Trapani, Stefano; Bonci, Andrea; Pirani, Massimiliano, IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA 2018)
  3. Programming robot work flows with a task modeling approach, Indri, Marina; Trapani, Stefano, 44th Annual Conference of the IEEE Industrial Electronics Society (IECON 2018)

- **General procedures and service algorithms of general validity**
  4. Development of a Virtual Collision Sensor for Industrial Robots, Indri, Marina; Trapani, Stefano; Lazzero, Ivan, journal SENSORS, 17(5), 1-23, 2017
  5. A general procedure for collision detection between an industrial robot and the environment , Indri, Marina; Trapani, Stefano; Lazzero, Ivan, 20th IEEE International Conference on Emerging Technologies and Factory , Automation (ETFA 2015)
  6. Development of a general friction identification framework for industrial manipulators, Indri, Marina; Trapani, Stefano; Lazzero, Ivan, IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society
  8. Smart Sensors Applications for a New Paradigm of a Production Line, Indri, Marina; Lachello, Luca; Lazzero, Ivan; Sibona, Fiorella; Trapani, Stefano. - In: SENSORS. - ISSN 1424-8220. - ELETTRONICO. - 19:3, 650(2019).
Thanks