Welcome!

Flexible programming of Industrial Robots for Agile Production environments
Laurent Cavazzana

researchXchange BFH-TI
Towards Agile Manufacturing

- Late 20th century: 3rd Industrial revolution (Digital Revolution)
  - more powerful, intelligent robots in production lines
  - production lines remained "static"

- 21st century:
  - new technologies (data exchange, cloud computing, 3D Printing, IoT, AI, etc)
  - step into the Industry 4.0 era
Towards Agile Manufacturing

- Rapidly evolving markets
- Hyper-personalization of products
Towards Agile Manufacturing

- Rapidly evolving markets
- Hyper-personalization of products

Major challenges for a manufacturing industry
Towards Agile Manufacturing

- Rapidly evolving markets
- Hyper-personalization of products

Major challenges for a manufacturing industry

- reduce time to market to stay competitive
- constant reprogramming of production tools and robots ?!
Agile Manufacturing

Organization which has the processes, tools, training enabling:

- Quick response to customer needs and market changes
- Cost control and quality
Agile Manufacturing

Organization which has the processes, tools, training enabling:

- Quick response to customer needs and market changes
- Cost control and quality

- Fine, but concretly what does it mean to production line in practice?
Agile Manufactoring
Agile Manufacturing

- flexible production concept:
  - Robotic system capable of self-adapting to varying production needs
  - Flexible reprogramming
  - Human Robot Collaboration (HRC) to combine:
    - Human's cognitive and creative power
    - Robots' high precision and repetitive tasks abilities
  - Digital simulation
The ACROBA Project

1 large company
- CABKA

7 SMEs
- nutai
- moses
- STAM
- ikor
- SteriPack

5 research centers
- BIBA
- ailtip
- vicomtech
- sigma
- HH Manufacturing Research

2 clusters
- EMC2
- ROBOCOAST
- DIH

2 universities
- Deusto
- Fh
The ACROBA Project

- **H2020 Innovation action**
- **Coordinator:** BFH
- **17 partners**
  - 9 countries
- **~8M€ budget**
  - ~7M€ EU funding
- **42 months**
- **5 industrial pilots**
- **2 ACROBA On-Site Lab**
- **Reference Architecture COPRA-AP**
- **12 hackathons**
The Project

- Design a novel concept of cognitive robotic platform enabling fast and cost-efficient deployment of robot systems.

  Reduction of:

  - robot programming time
  - commissioning time
  - software engineering
The ACROBA Project

Platform will be demonstrated by 5 industrial scenarios:

- Lights out manufacturing
  - STERIPACK: processing of medical 3D printed parts
  - CABKA/MOSES: Defects removal and quality control
- Collaborative assembly lines
  - IKOR: PTHs on PCBs
  - ICPE: Electric motors parts
Moses:

The ACROBA Project

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
The ACROBA Project

Moses:
The ACROBA Project

- Cabka:
The ACROBA Project

IKOR:
The ACROBA Project

Knowledge Transfer

ACROBA Platform

Integration Layer: Pilot Robotic Cell

Cyberphysical model

Collaborative Assembly Pilot Lines & Lights Out Pilot Lines
Flexible Programming of Industrial Robots
Flexible Programming of Industrial Robots
User Interface

- Task design:
  - Flexible and simple enough
  - New task from scratch
  - Save, load, visualize created task(s)
  - Allow for the creation of new skills / actions
  - From collaboration (learning from Demonstration)
  - CAD (step files, etc)

- Task Execution
  - Monitoring: know actions the robot did or is going to perform
  - Control the robot if needed (pause, restart, repeat some action(s), skip some, etc)
Flexible Programming of Industrial Robots

User Interface

- **Typical user of the UI?**
  - Lower skill workers (operator, etc): no knowledge of robotics
  - Robotics experts (roboticists, robot designer, etc)

- **What kind of user interface?**
Flexible Programming of Industrial Robots

User Interface

- Typical user of the UI?
  - Lower skill workers (operator, etc): no knowledge of robotics
  - Robotics experts (roboticists, robot designer, etc)

- What kind of user interface?

  researchXchange Robot Task Model and Notation
  Congyu Zhang Sprenger
Robot programming is time consuming and costly and requires high levels of expertise.

Current systems are difficult to understand and control by operators.

Non-programming experts are facing challenges in the fast-growing agile production industry.

Source: Congyu Zhang Sprenger - researchXchange Robot Task Model & Notation
Robot programming is time consuming and costly and requires high levels of expertise.

Current systems are difficult to understand and control by operators.

Non-programming experts are facing challenges in the fast-growing agile production industry.

Need for an intuitive way of modeling and programming that enables nonexperts to plan and program robot tasks.

Source: Congyu Zhang Sprenger - researchXchange Robot Task Model & Notation
Research - Questionnaire for ACROBA User Interface

1. What are the users of the ACROBA platform?
15 responses

- Business users: 1 (6.7%)
- Engineers: 14 (93.3%)
- Programmers: 8 (53.3%)
- Manager: 4 (26.7%)
- Operators: 2 (13.3%)
- Operator: 1 (6.7%)
Research - Questionnaire for ACROBA User Interface

4. What do you expect for a user interface of ACROBA?
15 responses

- One user interface for both platforms: 4 (26.7%)
- Separate user interface for platforms: 7 (46.7%)
- Graphical user interface: 7 (46.7%)
- Simple texts and buttons: 5 (33.3%)
- Easy to use: 11 (73.3%)
- No user interface needed: 0 (0%)
- No licensing required: 1 (6.7%)
Source: Congyu Zhang Sprenger - researchXchange Robot Task Model & Notation

Research - Questionnaire for ACROBA User Interface

- Which task representation do you like the most?

73.3% vote for this one
The RTMN Model - User Interface
The RTMN Model - User Interface
The RTMN Model - User Interface

Source: Congyu Zhang Sprenger - researchXchange Robot Task Model & Notation
The RTMN Model - User Interface
Flexible Programming of Industrial Robots
Task Planner

UI
HMI
Execution control
GUI
Task design
Rest API
ROS
Task Planner

Virtual Gym ROS Interface
Virtual Gym (simulation engine)
Virtual scene
Perception Module ROS Interface
Sensors, Robot controller, Hardware
Flexible Programming of Industrial Robots
Task Planner

HMI
Execution control

GUI
Task design

UI

Rest API

ROS

Virtual Gym ROS Interface
Virtual Gym (simulation engine)
Virtual scene

Perception Module
ROS Interface

Sensors, Robot controller, Hardware
Flexible Programming of Industrial Robots

Task Planner

- Generate a task from the UI representation and existing skills/primitives.
- Performs the task execution/control.
- Automatic online replanning
Flexible Programming of Industrial Robots

Task Planner

UI

HMI
Execution control

GUI
Task design

Rest API

ROS

Virtual Gym ROS Interface

Virtual Gym
(simulation engine)

Virtual scene

Perception Module
ROS Interface

Sensors,
Robot controller,
Hardware
Flexible Programming of Industrial Robots Skills
# Flexible Programming of Industrial Robots Skills

<table>
<thead>
<tr>
<th>Primitives</th>
<th>Input (Goals)</th>
<th>Outputs (Results)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CADLoading</td>
<td>String: path&lt;br&gt;String: frame_id</td>
<td>PointCloud2 P2</td>
<td>Primitive that loads a CAD file (stl, STL, obj, pcD or ply). Then converts it into a point cloud.</td>
</tr>
<tr>
<td>DetectMarkers</td>
<td>None</td>
<td>None</td>
<td>Primitive that detects specific markers (ARUCO, CHARUCO, QRCODE, datamatrix...) to outputs its associated data.</td>
</tr>
<tr>
<td>ExecuteTrajectory</td>
<td>RobotTrajectory: tr</td>
<td>None</td>
<td>Primitive that execute a given trajectory and check the proper execution.</td>
</tr>
<tr>
<td>GenerateGraspPose</td>
<td>bool: visualize (optional)&lt;br&gt;Pose: id_frame</td>
<td>Pose: id_frame</td>
<td>Primitive that outputs a optimal grasping position, of an object, from a rbgd image.</td>
</tr>
<tr>
<td>GenerateTrajectory</td>
<td>string: id_frame&lt;br&gt;Pos: [t: pos p: float j: articulate p: bool: cartesian]</td>
<td>RobotTrajectory: tr</td>
<td>Primitive that generates a robot trajectory between actual robot state and one or several given points, with a cartesian or articulate movement.</td>
</tr>
<tr>
<td>GetMaxDist</td>
<td>PointCloud2: point_cloud_in&lt;br&gt;Float: max_dist</td>
<td>Float: max_dist</td>
<td>Primitive that outputs the biggest distance between 2 points of a pointcloud.</td>
</tr>
<tr>
<td>Grasp</td>
<td>Int: gap=255 (by default)&lt;br&gt;Bool: finish</td>
<td></td>
<td>Primitive that controls the closure of a Robotiq 2F or Robotiq 3F (cinch) gripper, with object detection. More gripper could be added later on.</td>
</tr>
<tr>
<td>Matching</td>
<td>PointCloud2: P4&lt;br&gt;PointCloud2: P3&lt;br&gt;String: id_frame</td>
<td>Pose: P&lt;br&gt;String: id_frame</td>
<td>Primitive that matches a source pointcloud to an object in the target scene.</td>
</tr>
<tr>
<td>OutlierRemoval</td>
<td>PointCloud2: P4&lt;br&gt;Float: Deviation&lt;br&gt;Int: Mean</td>
<td>PointCloud2: P5</td>
<td>Primitive that removes outliers from a pointcloud. Based on remove_statistical_outlier from open3d library.</td>
</tr>
<tr>
<td>PlaneFiltering</td>
<td>PointCloud2: P3&lt;br&gt;Float: Threshold</td>
<td>PointCloud2: P4</td>
<td>Primitive that filters a plane from a pointcloud. It identifies the biggest possible planes that fits the pointcloud using ransac method (<a href="https://pypi.org/project/pyransac3d/">https://pypi.org/project/pyransac3d/</a>). It then filters all points that are at a given distance of this plane.</td>
</tr>
<tr>
<td>Point3DTo2Dpixels</td>
<td>Pose: p=Pose: camera_info&lt;br&gt;Int: x&lt;br&gt;Int: y</td>
<td></td>
<td>Primitive that takes a 3D point in the camera Frame and convert it to 2D pixel coordinates in the image.</td>
</tr>
<tr>
<td>Release</td>
<td>Int: gap=0 (by default)</td>
<td>Bool: finish</td>
<td>Primitive that controls the opening of a Robotiq 2F or Robotiq 3F gripper. More gripper could be added later on.</td>
</tr>
<tr>
<td>ROISelection</td>
<td>PointCloud2: P1&lt;br&gt;Float: X&lt;br&gt;Float: Y&lt;br&gt;Max</td>
<td>PointCloud2: P2</td>
<td>Primitive that crop the pointcloud to a given region of interest.</td>
</tr>
<tr>
<td>Subsampling</td>
<td>PointCloud2: P2&lt;br&gt;Float: size_leaf</td>
<td>PointCloud2: P3</td>
<td>Primitive to downsample the pointcloud using open3d.geometry.voxel_down_sample</td>
</tr>
</tbody>
</table>
## Flexible Programming of Industrial Robots Skills

<table>
<thead>
<tr>
<th>Skills</th>
<th>Goals</th>
<th>Results</th>
</tr>
</thead>
</table>
| PointCloudProcessing | PointCloud2: P1  
Float[]: Min  
Float[]: Max  
Float: size_leaf  
Float: Th  
Float: Deviation  
Int: Mean | PointCloud2: P5 |
| Pick | string: id_frame  
Pose[]: pose  
Float[]: articular_pose  
bool: cartesian | None |
| Place | string: id_frame  
Pose[]: pose  
Float[]: articular_pose  
bool: cartesian | None |
| CADMatching | String: path  
String: frame_id  
PointCloud2: P4 | Pose: P  
string: id_frame |
| MoveTo | string: id_frame  
Pose[]: pose  
Float[]: articular_pose  
bool: cartesian | None |
| Locate | Float: timeout  
String: rgb_image_topic_name  
String: depth_image_topic_name  
String: pointCloud_topic_name  
String: camera_info_topic_name  
String: robot_frame  
String: depth_frame  
String: color_frame  
Pose: points  
Float: articular_pose  
bool: cartesian  
PointCloud2: point_cloud_in  
Float: std  
Int: mean  
Float: thresh  
Float: leaf_size  
Float: min  
Float: max  
String: path  
Float: offset  
bool: visualize | Pose: grasping_pose  
String: frame_id  
bool: cartesian |
## Flexible Programming of Industrial Robots Skills

<table>
<thead>
<tr>
<th>Skills</th>
<th>Input (Goals)</th>
<th>Outputs (Results)</th>
<th>Sub-Program (Primitive or Skill)</th>
<th>Type</th>
<th>Input (Goals)</th>
<th>Outputs (Result)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MonitoringDepthImage</td>
<td>Primitive</td>
<td>String: topic_name&lt;br&gt;Float: timeout</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MonitoringPointCloud</td>
<td>Primitive</td>
<td>String: topic_name&lt;br&gt;Float: timeout</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MonitoringRGBImage</td>
<td>Primitive</td>
<td>String: topic_name&lt;br&gt;Float: timeout</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MoveTo</td>
<td>Skill</td>
<td>string: id_frame&lt;br&gt;Pose[]: pose&lt;br&gt;Float[]: articular_pose&lt;br&gt;Bool: cartesian</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WorkspaceDetection</td>
<td>Primitive</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DataProcessing</td>
<td>Skill</td>
<td>PointCloud2: P1&lt;br&gt;float[]: plane_eq&lt;br&gt;float: th&lt;br&gt;float[]: leaf_size&lt;br&gt;float: min&lt;br&gt;float[]: max</td>
<td>PointCloud2: P2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CADMatching</td>
<td>Skill</td>
<td>String: path&lt;br&gt;String: frame_id&lt;br&gt;PointCloud2: P4</td>
<td>Pose: P&lt;br&gt;string: id_frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MoveTo</td>
<td>Skill</td>
<td>string: id_frame&lt;br&gt;Pose[]: pose&lt;br&gt;Float[]: articular_pose&lt;br&gt;Bool: cartesian</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GenerateGraspPose</td>
<td>Primitive</td>
<td>bool: visualize (optional)</td>
<td>Pose: id_frame</td>
</tr>
</tbody>
</table>

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
Flexible Programming of Industrial Robots

Task Planner

- World Knowledge,
- Cell description/setup,
- domain definition,
- Primitive & Skills,
- etc.

ROS

Virtual Gym ROS Interface

Virtual Gym (simulation engine)

Virtual scene

HMI
Execution control

GUI
Task design

UI

start/stop

Rest API

Ontology

Sensors, Robot controller, Hardware

Perception Module ROS Interface

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
Flexible Programming of Industrial Robots

Task Planner

• Generate a task from the existing skills/primitives
• Performs the task execution/control
• Automatic online replanning
Flexible Programming of Industrial Robots

Behavior Trees

- Generalization of Finite State Machine
- Tree of hierarchical nodes control the flow of decision making
- Flexible and powerful
- Used in games to emulate AI, intelligent agent/behavior
Flexible Programming of Industrial Robots

Behavior Trees

Source: https://www.behaviortree.dev/docs/learn-the-basics/BT_basics
Flexible Programming of Industrial Robots
Behavior Trees

- Tick signal sent to the root of the tree
- Propagate it to through the tree till it reaches a leaf node
- Tree node receiving a tick executes a callback, which returns either:
  - SUCCESS
  - FAILURE
  - RUNNING
- If a node has multiple children, it is responsible to propagate to its children
# Flexible Programming of Industrial Robots

## Behavior Trees

<table>
<thead>
<tr>
<th>Type of TreeNode</th>
<th>Children Count</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ControlNode</td>
<td>1...N</td>
<td>Usually, ticks a child based on the result of its siblings or/and its own state.</td>
</tr>
<tr>
<td>DecoratorNode</td>
<td>1</td>
<td>Among other things, it may alter the result of the children or tick it multiple times.</td>
</tr>
<tr>
<td>ConditionNode</td>
<td>0</td>
<td>Should not alter the system. Shall not return RUNNING.</td>
</tr>
<tr>
<td>ActionNode</td>
<td>0</td>
<td>This is the Node that &quot;does something&quot;</td>
</tr>
</tbody>
</table>

Source: https://www.behaviortree.dev/docs/learn-the-basics/BT_basics
Flexible Programming of Industrial Robots
Behavior Trees

- Presentation of the PCB
- Presentation of the PTHs
- Pick and place of the PTHs
Flexible Programming of Industrial Robots

Behavior Trees
Flexible Programming of Industrial Robots

Task Planner

- Generate a task from the existing skills/primitives
- Performs the task execution/control
- **Automatic online replanning**
Flexible Programming of Industrial Robots

AI Planning

- Planning Problem:
  - Initial starting state, I
  - Goal state, G
  - Set of actions, A. (Each action has Preconditions, Effects)

- From this, a planner generates a plan:
  - Series of actions from A that turn I into G
Flexible Programming of Industrial Robots

AI Planning

PDDL (Planning Language definition)

- Lisp-like syntax
- Two parts:
  - Domain: abstract predicate definitions, actions
  - Problem: initial state, goal state
- Standardized!
  - write one model, solve it with any planner
  - Easy benchmarking across solvers (same input)
- Domain independent planning
Flexible Programming of Industrial Robots
AI Planning

Example:

Domain:
Two rooms, four balls and two robot arms
Actions:
- robot can move between rooms
- pick up a ball
- drop a ball

Problem:
Initial State: All balls and the robot are in the first room
Goal: All balls must be in the second room
Flexible Programming of Industrial Robots

AI Planning

• Domain definition:

```
(define (domain <domain name>)
  <PDDL code for predicates>
  <PDDL code for first action>
  [...]
  <PDDL code for last action>
)
```

```
(:predicates (ROOM ?x) (BALL ?x) (GRIPPER ?x)
  (at-robby ?x) (at-ball ?x ?y)
  (free ?x) (carry ?x ?y))
```
Flexible Programming of Industrial Robots
AI Planning

- Domain definition (actions):

```prolog
(:action move :parameters (?x ?y)
 :precondition (and (ROOM ?x) (ROOM ?y)
 (at-robby ?x))
 :effect (and (at-robby ?y)
 (not (at-robby ?x))))

(:action pick-up :parameters (?x ?y ?z)
 :precondition (and (BALL ?x) (ROOM ?y) (GRIPPER ?z)
 (at-ball ?x ?y) (at-robby ?y) (free ?z))
 :effect (and (carry ?z ?x)
 (not (at-ball ?x ?y)) (not (free ?z))))
```
Flexible Programming of Industrial Robots

AI Planning

• Problem definition:

```
(define (problem <problem name>)
  (:domain <domain name>)
  <PDDL code for objects>
  <PDDL code for initial state>
  <PDDL code for goal specification>
)

(:objects rooma roomb
  ball1 ball2 ball3 ball4
  left right)
```
Flexible Programming of Industrial Robots

AI Planning

• Problem definition:

\[
(:\text{init} \ (\text{ROOM \ rooma}) \ (\text{ROOM \ roomb})
(BALL \ ball1) \ (BALL \ ball2) \ (BALL \ ball3) \ (BALL \ ball4)
(GRIPPER \ left) \ (GRIPPER \ right) \ (\text{free \ left}) \ (\text{free \ right})
(at-\text{robbby \ rooma})
(at-ball \ ball1 \ rooma) \ (at-ball \ ball2 \ rooma)
(at-ball \ ball3 \ rooma) \ (at-ball \ ball4 \ rooma))
\]

\[
(:\text{goal} \ (\text{and} \ (at-ball \ ball1 \ roomb)
(at-ball \ ball2 \ roomb)
(at-ball \ ball3 \ roomb)
(at-ball \ ball4 \ roomb)))
\]
Flexible Programming of Industrial Robots

AI Planning

- Solving: Forward chain planning
Flexible Programming of Industrial Robots
AI Planning

- Solving: Forward chain planning
Flexible Programming of Industrial Robots

AI Planning

- Solving: Forward chain planning
Flexible Programming of Industrial Robots

AI Planning

• Solving: Forward chain planning

• Solution plan:
  - pickup ball1 rooma left
  - pickup ball4 rooma right
  - move rooma roomb
  - drop ball1
  - drop ball4
  - move roomb rooma
  - pickup ball3 rooma left
  - pickup ball2 rooma right
  - move rooma roomb
  - drop ball3
  - drop ball2
Flexible Programming of Industrial Robots
AI Planning

Search strategies:

- Breadth First Search
- Depth First Search

Planning problem is Hard:

Suppose:
20 actions to choose from
- Solution plan is 20 steps long.

- Worst case search $20^{20} = 1e26$
  1 million iterations per second, ETA $1e20$ seconds
  Age of the universe in seconds = $4.35e+17$ seconds!
Flexible Programming of Industrial Robots
AI Planning

Key challenge: search guidance!

Use of heuristics:
- function giving an estimate of how far we are from a goal.
- use simplified problem
- use heuristics that never over-estimates
Flexible Programming of Industrial Robots
AI Planning

Initial state

Goal state

Source: Introduction to AI Planning, Part 1 (Amanda Coles, EASSS 2013)
Flexible Programming of Industrial Robots
AI Planning

Heuristic: number of misplaced tiles

$H(\text{initia state}) = 6$

Source: Introduction to AI Planning, Part 1 (Amanda Coles, EASSS 2013)
Flexible Programming of Industrial Robots
AI Planning

Heuristic: sum of distance to correct position of misplaced tiles

\[ h(\text{initial state}) = 2 + 3 + 3 + 2 + 1 + 3 = 14 \]

Source: Introduction to AI Planning, Part 1 (Amanda Coles, EASSS 2013)
Flexible Programming of Industrial Robots

AI Planning

And then?

- Solution plan:
  - pickup ball1 rooma left
  - pickup ball2 rooma right
  - move rooma roomb
  - drop ball1
  - drop ball2

Behavior Tree generation, then task Execution

Flexible Programming of Industrial Robots
Deep Reinforcement Learning
Flexible Programming of Industrial Robots
How Flexible?

Task Planner

World Knowledge,
Cell description/setup,
domain definition,
Primitive & Skills,
etc.

Ontology

Rest API

HMI
Execution control

GUI
Task design

UI

Virtual Gym ROS Interface

Virtual Gym
(simulation engine)

Virtual scene

DRL

Perception Module
ROS Interface

Sensors,
Robot controller,
Hardware

DRL

ROS

start/stop

start/stop

Berner Fachhochschule | Haute école spécialisée bernoise | Bern University of Applied Sciences
Flexible Programming of Industrial Robots
How Flexible?

- Platform shipped with a set of skills & compatible hardware
- UI & BTs gives great flexibility.

Challenges:

- Ontology definitions (robotics engineer)
- Virtual scene definition (templates?)
- Hardware abstraction?
Flexible Programming of Industrial Robots

thank you
Next seminars

**Biel/Bienne**
Quellgasse 21, Aula

**25.11.22** Experimental heart rate variability characterization
Lars Brockmann, Assistant, Institute for Human Centered Engineering HuCE, BFH-TI

**09.12.22** Parylene-based encapsulation technology for wearable or implantable electronic devices
Dr. Andreas Hogg, CEO, Coat-X AG, La Chaux-de-Fonds

**13.01.23** Care@Home mit technischer Unterstützung
Prof. Dr. Sang-II Kim, Professor, Institute for Medical Informatics I4MI, BFH-TI

**Burgdorf/Berthoud**
Pestalozzistrasse 20, E 013

**02.12.22** Wie gefährlich ist ein Unfall mit einem Cabriolet?
Prof. Raphael Murri, Institutsleiter IEM, Institut für Energie- und Mobilitätsforschung IEM, BFH-TI

**16.12.22** Systemtechnologie für die Mikrobearbeitung mit Hochleistungs-UKP-Lasern
Prof. Dr. Beat Neuenschwander, Institutsleiter ALPS, Institute for Applied Laser, Photonics and Surface Technologies ALPS, BFH-TI