Reactive Trajectory Planning for Robotic Operations in an Unstructured Environment

Simulated by Tecnomatix Process Simulate

Giulio Carpi Lapi

Université de Strasbourg

August 27, 2025

Outline

Context: Industry 4.0 & smart manufacturing

- **Industry 4.0**: **AI**, **IoT**, and **data analytics**.
- **Digital Twins & Simulation**: **Virtual testing**, **process optimization**, and **reducing costs**.
- **Automation**: **Competitiveness**, **precision**, and **safety**.

Internship Framework

- **Advanced Robotic Kinematics** team.
- **Kineo** department.
- **Siemens Digital Industries Software**, Toulouse.

Internship Framework: Kineo presentation

- Founded in **2001** as a spin-off from **LAAS-CNRS**.
- Acquired by **Siemens PLM Software** in **2012**.
- **Automatic motion planning** and **collision detection**.
- **Software Development Kits** to be **integrated** into **third-party software**.

KineoWorks: Collision-free trajectories

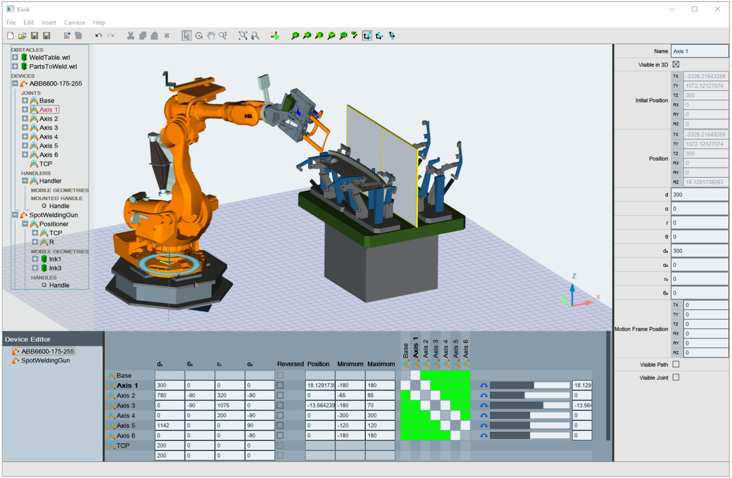

Figure: KineoWorks interface which computes collision-free trajectories [1]

Kineo Collision Detector: Collision detection

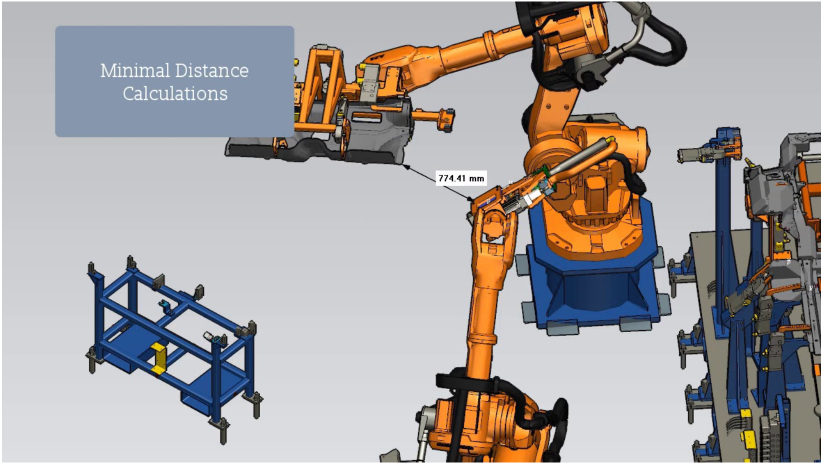

Figure: Kineo Collision Detector which performs collision detection [2]

Kineo Flexible Cables: Deformable cables simulation

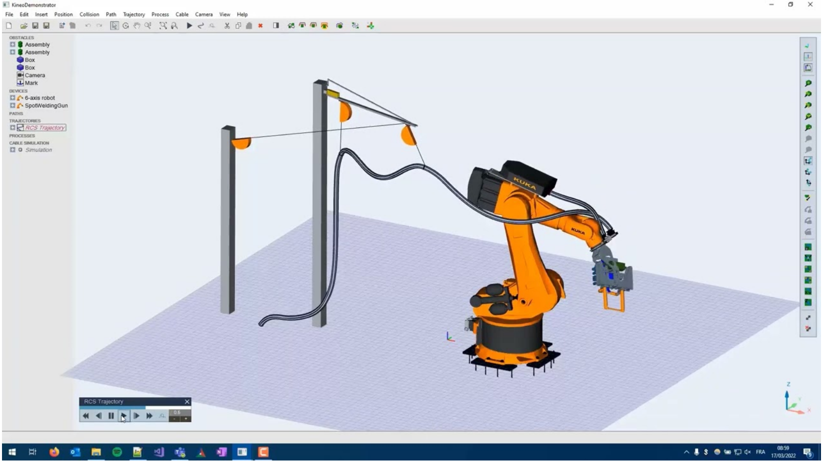

Figure: Kineo Flexible Cables which simulates the behavior of deformable cables [3]

Kineo 3D Nesting: Parts arrangement optimization

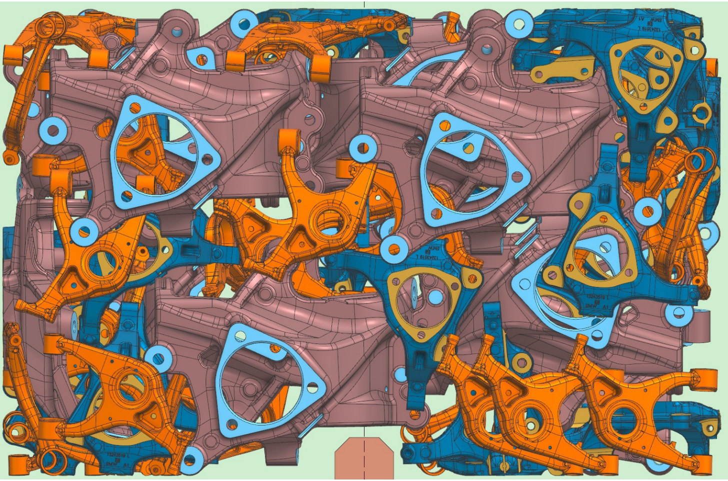

Figure: Kineo 3D Nesting which optimizes the arrangement of parts within a defined space [4]

Workflow & Tools

- **Scrum** methodology.
- **2-3 week sprints** with **daily stand-up meetings**.
- **Polarion** application lifecycle management.
- **Git**, **Git Extensions** for version control.

Project Objectives

- **Proof-of-concept** of **reactive trajectory planner** for **Tecnomatix Process Simulate**.
- Reactive **dynamic obstacles avoidance** based on real-time virtual **camera data**.
- Safe collaboration between **robots**, **inspection drones**, and **AGVs**.

The Environment: Tecnomatix Process Simulate

- Siemens **digital manufacturing** software.
- **Design**, **simulate**, and **validate manufacturing processes**.
- Test **control logic** in a **virtual setting** before **physical deployment**.

The Environment: A robotic workcell

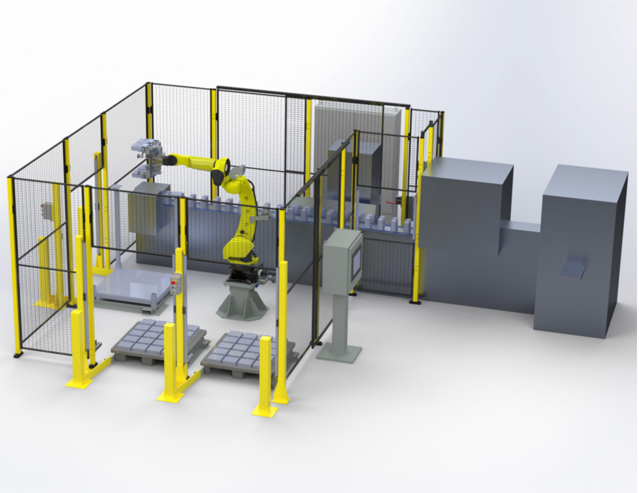

Figure: A typical robotic workcell [5]

The Environment: 3D bin picking

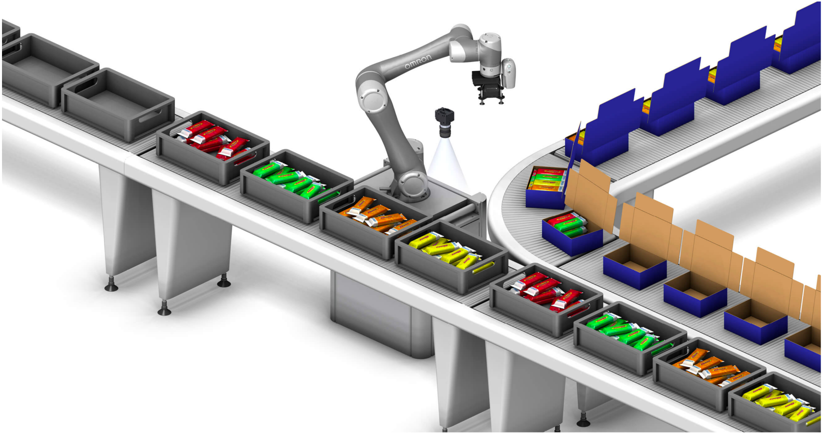

Figure: A bin picking scenario [6]

Video demonstration of a **bin-picking operation** in **Tecnomatix Process Simulate**.

Reactive Control Loop: The perception pipeline

1. **Capture & Generate**: From **depth data** from **virtual cameras** to **3D point cloud**.
2. **Filter Stage 1**: **Filter robot's geometries**.
3. **Filter Stage 2**: **Filter static environment**.
4. **Detect Collisions**: between **robot** and **point cloud**.
5. **Path Planning**: Generate a **collision-free path**.

Reactive Control Loop: State machine

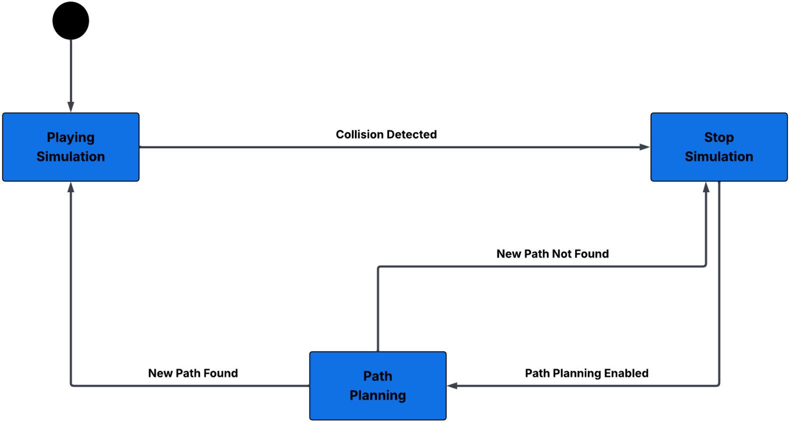

Figure: State machine for the complete reactive control loop.

Automatic Path Planning

- My work extends the core **Automatic Path Planning** functionality.
- **Integration** of **Kineo's core technologies** in **Tecnomatix Process Simulate**.

A Multi-Layered Architecture

- **Presentation (C#/WPF)**: The User Interface.
- **Application (C#)**: Orchestrates the workflow.
- **Bridge (C++/CLI)**: Connects managed C# to native C++.
- **Core Engine (C++)**: High-performance simulation and algorithms.

Test Scenario: Bin picking operation

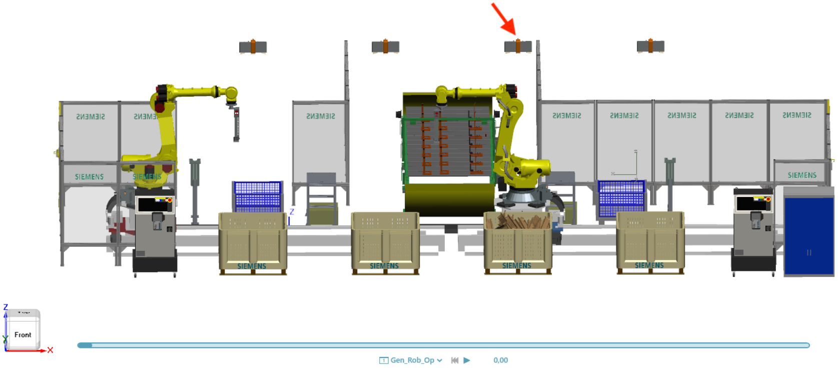

Figure: Initial state before the operation begins.

Virtual Camera Output: RGB image

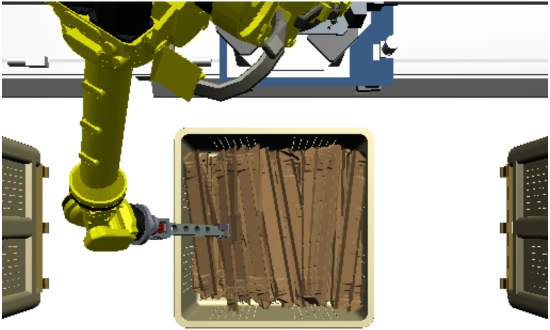

Figure: RGB virtual camera output at initial state.

Virtual Camera Output: Depth map

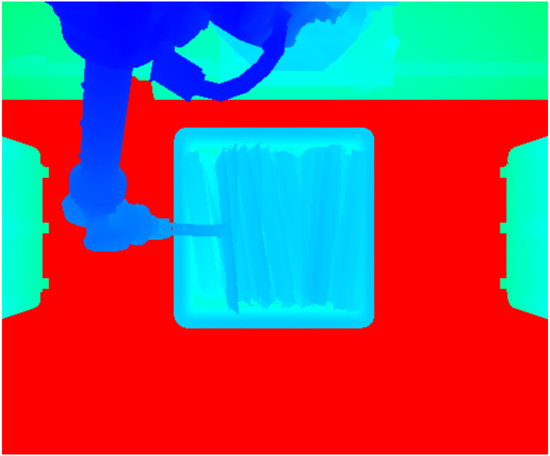

Figure: RGB-D virtual camera output at initial state.

Tecnomatix Snapshot API Benchmark

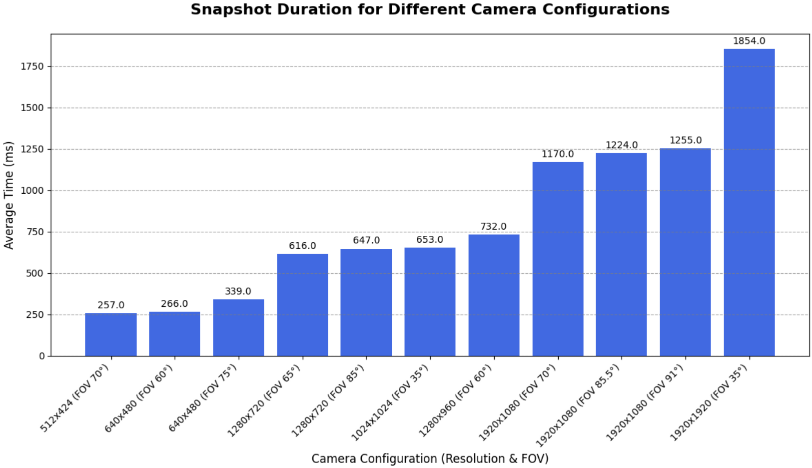

Figure: Snapshot duration vs. camera configuration.

Virtual Camera Configuration

- 512×424 and 70° FOV (Microsoft Kinect v2).
- **Depth buffer**: 217,088 values.
- Snapshot operation averaged 250 ms.

Depth Data Processing: Depth buffer

- **Depth buffer**: 2D array stored as 1D float list in row-major order.
- Each value represents the **distance** from the **camera** to an **object** at **pixel (u, v)**.
- Access **pixel (u, v)** at index: $v \times \text{width} + u$.

- **Focal Length (f)**: Derived from field of view (FOV) and image dimensions.

$$f = \frac{\sqrt{\text{width}^2 + \text{height}^2}}{2\tan\left(\frac{\text{FOV}}{2}\right)}$$

- **Principal Points** (c_x, c_y): Define the **optical center** of the image.

$$c_x = \frac{\text{width} - 1}{2}, \quad c_y = \frac{\text{height} - 1}{2}$$

Depth Data Processing: Pixel to 3D point

1. **Depth**: $d = \text{depthBuffer}[v \times \text{width} + u]$

2. **Normalization factor**:

$$t = \sqrt{\left(\frac{u - c_x}{f}\right)^2 + \left(\frac{v - c_y}{f}\right)^2 + 1}$$

3. **3D coordinates** in camera space:

$$X = \frac{(v - c_y) \times d}{f \times t}, \quad Y = \frac{(c_x - u) \times d}{f \times t}, \quad Z = \frac{d}{t}$$

Point Cloud Generation: Camera to world coordinates

Apply the **transformation matrix**:

$$P_{world} = T_{camera \rightarrow world} \times P_{camera}$$

Point Cloud Generation: Generated point cloud

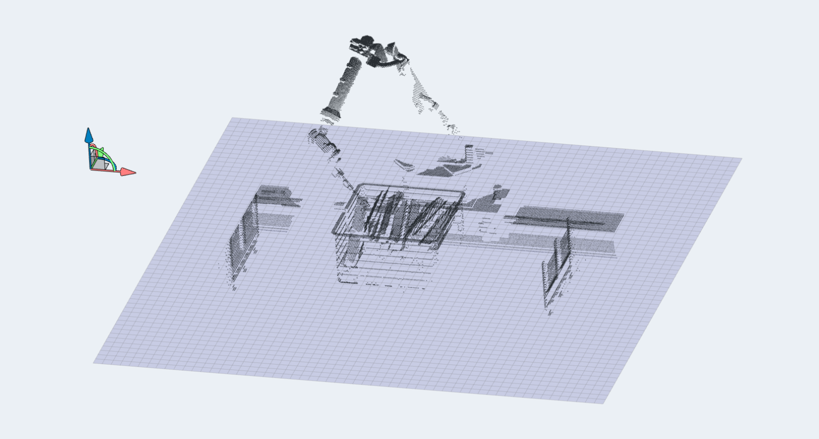

Figure: Generated point cloud (includes robot).

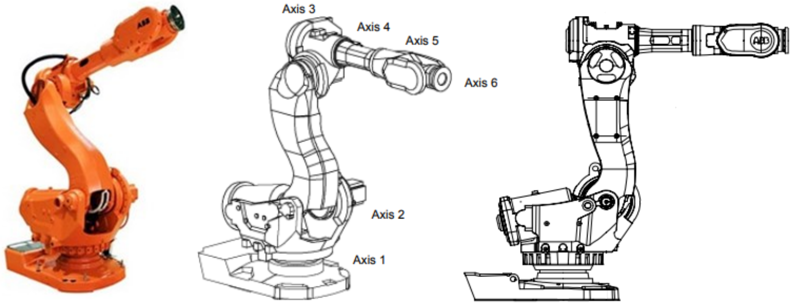

Figure: Robot's geometries [7].

Robot's Geometries Filtering: Oriented bounding boxes

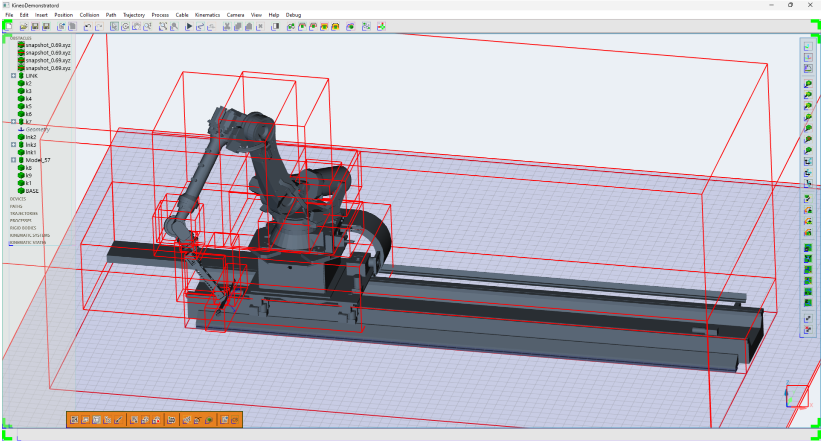

Figure: Device's Oriented Bounding Box.

1. Retrieve **current robot's kinematic configuration**.
2. Get **oriented bounding boxes**.
3. Discard any point inside the OBBs.

Prevents from **detecting** the robotic system as an obstacle.

Filtered Point Cloud

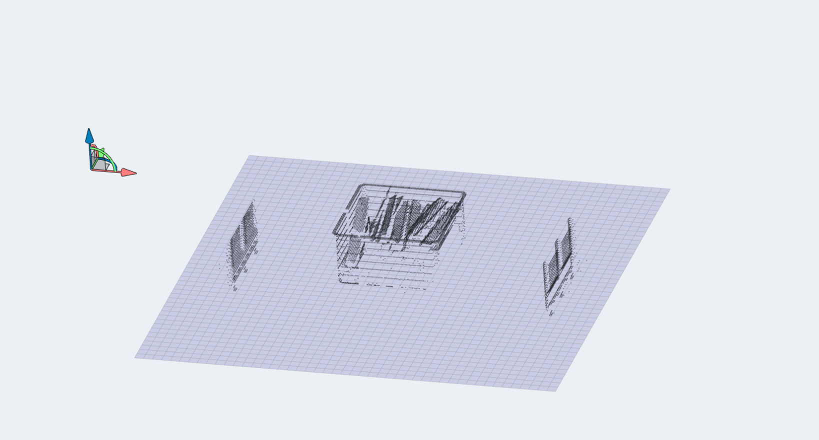

Figure: Filtered point cloud (After robot points are removed).

Object Flow Operation

- Box representing **dynamic obstacle**
- **Object flow operation** moves the **box** to **collide** with **robotic system**.
- Simulates **automated guided vehicle** or **inspection drone** entering the **robot's workspace**.

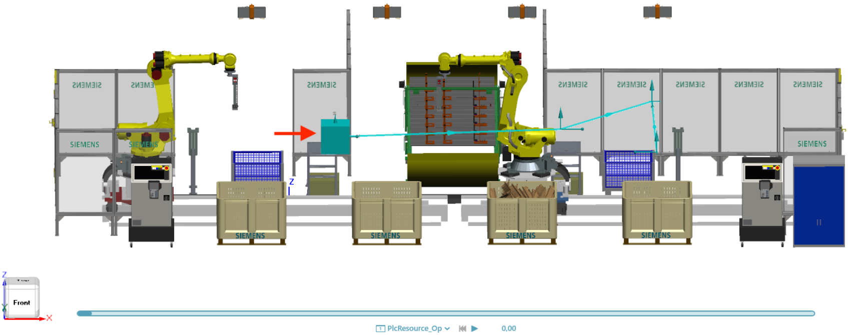

Figure: The dynamic obstacle (box) and its planned trajectory (blue).

Virtual Camera Output: RGB image

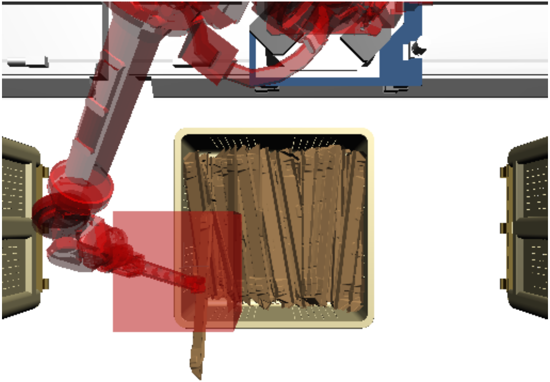

Figure: RGB virtual camera output at the moment of detected collision.

Virtual Camera Output: Depth map

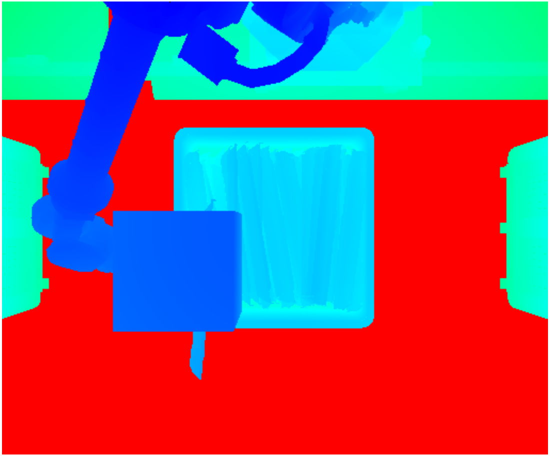

Figure: RGB-D virtual camera output at the moment of detected collision.

The Custom Simulator

- **Tecnomatix Process Simulate**'s **graphics engine**.
- Advances simulation time in **discrete steps**.
- Calculates and **updates the poses** of the **robot** and the **box**.
- Creates **synchronized**, **dynamic test environment**.

Static Environment Filter

1. Save **static baseline** point cloud.
2. **Compare** each subsequent **point cloud** to **baseline**.
3. **Discard** any point that is **close** to a point in the baseline.

Result: Point cloud containing only **dynamic** objects.

Collision Detection

1. Point cloud **wrapped** in custom **Kineo PointCloud** geometry object.
2. **Collision Set** with **robotic system** and **point cloud**.
3. **Kineo Collision Detector** runs **query** with the **collision set**.
4. **Collision** or **near-miss** stops the custom simulator.

WPF Dialog Box (C#/WPF)

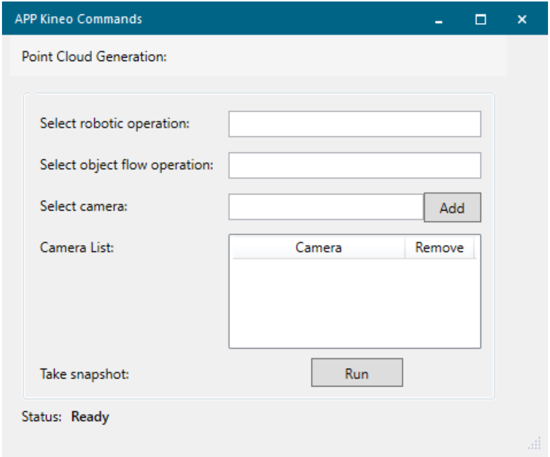

Figure: WPF dialog box for user interaction.

Video demonstration of a Robotic Simulation being stopped in **Tecnomatix Process Simulate**.

Conclusion

- **Integrated** a **reactive trajectory planner** into **Tecnomatix Process Simulate**.
- **Validated** in a bin picking scenario.
- **Key achievements**: Custom simulation loop, depth data capture and processing, two-stage filtering algorithm, collision detection and response.

Conclusion

- **Limitation** posed by **Tecnomatix snapshot API**.

- **Bottleneck** not important with **physical hardware**.

Future Work: Path planning

1. **Feed** point cloud to **KineoWorks**.
2. Compute **collision-free** path to **target**.
3. **Resume motion**.

Future Work: Performance optimization

- **Robot Geometry Filtering**:
  - **Current**: Checks each point against every robot geometry ($O(N \times M)$).
  - **Future**: Use spatial structure (**Octree**).
- **Static Environment Filtering**:
  - **Current**: Compares each new point to every baseline point ($O(N \times M)$).
  - **Future**: Store baseline points (**k-d tree**, **voxel grid**) efficient **nearest-neighbor search** ($O(N \log M)$).

The End

Thank you for your attention!

Siemens Digital Industries Software.
KineoWorks Version 7.0 Highlights, 2019.

Siemens Digital Industries Software.
Kineo components in robot simulation, 2018.

Siemens Digital Industries Software.
Kineo Flexible Cables | Versatile Robot Dresspack Modeling, 2022.

Silvia Peruch.
Introducing the new Kineo 2D and 3D Nesting software components from Siemens, 2025.

Control Engineering.
Using a modular robotic platform to enhance safety, efficiency for battery manufacturer, 2024.

Omron.
3D Bin picking: Pick up objects with random overlapping positions, 2025.

Kineo Wiki.

Robot's geometries, 2025.
Internal documentation, not publicly accessible. Accessed August 2025.