3DSSC (3D Smart Sense and Control)

Deliverable D1.2 Early Integration

3D Smart Sensing and Flexible Task Programming for On-Line Trajectory Adaptation in Fast Surface Treatment

Vanthienen Dominick (Flexible Robotic Solutions, Belgium)
Vander Elst Karel (Flexible Robotic Solutions, Belgium)
Aertbeliën Erwin (KU Leuven University, Belgium)
De Schutter Joris (Flexible Robotic Solutions, Belgium)
Delforge Philippe (Flexible Robotic Solutions, Belgium)

1 Hardware Integration

Two tools have been developed.

A first tool (fig 1b) consists of the planing tool, I.e. a mill, and a set of three laser sensors mounted before the mill, I.e. in the direction of the mill movement The mill consists of a rotating axis to which a set of two knives are attached. The three lasers are used to measure the cheese surface and build a local model of the cheese surface. A second tool (fig 1a) replaces the planing tool with a laser. The latter is used during the development process to verify the accuracy of the control algorithms by measuring the actually realized distance between the robot end-effector and the position where the mill should remove material. Figure 1 shows a schematic of both setups with indication of the important frames, between which control constraints are imposed.

As reported in RAP3, we identified and optimized the communication delay of different laser products, and optimized the robustness and the frequency of the laser sensor readings.

Furthermore, we modeled the robot control characteristics and identified its parameters (control and measurement delays, and time constant of the position controllers).

We designed a planing tool specifically for the application and integrated it in the setup, as shown in Figure 2. The planing tool prototype was tested on wooden blocks (mock-ups for the cheese blocks), which allowed us to find the appropriate process parameters. The tool also contains an outlet port connected to an industrial vacuum cleaner to evacuate the removed material.

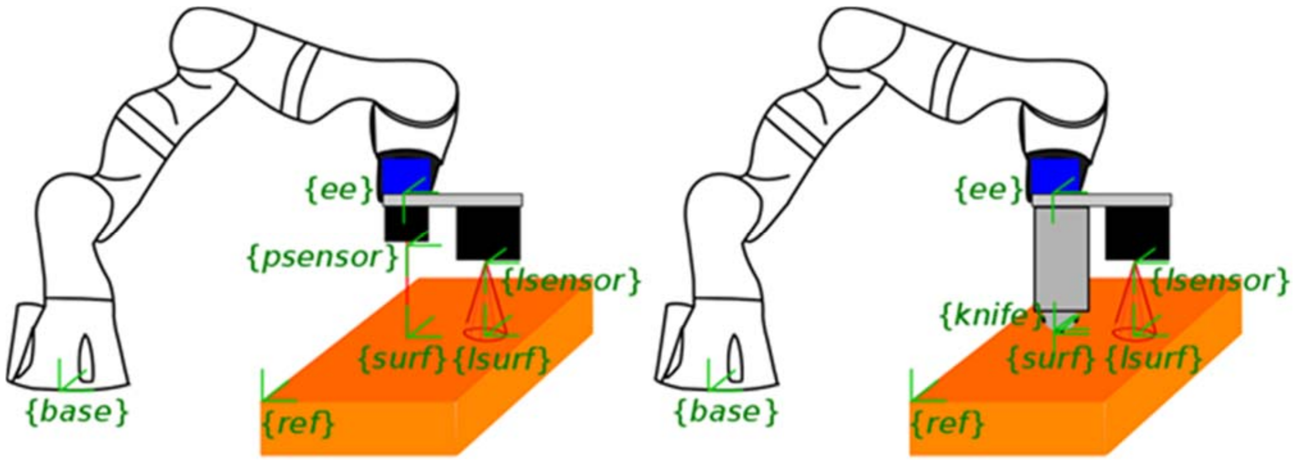

Figure 1: Setups: {psensor} indicates the sensor used for verification, {lsensor} indicates the pose of one of the three lasers (the middle one) used to measure the surface measurement, and {knife} indicates the frame connected to the point on the tool where the knife touches the cheese.

We created a visualization of the whole setup in rviz [http://wiki.ros.org/rviz],

in order to verify the correct input from the sensors, to visualize the interaction of constraints, and to visualize the measured surface. Figure 3 shows the visualization during an experiment.

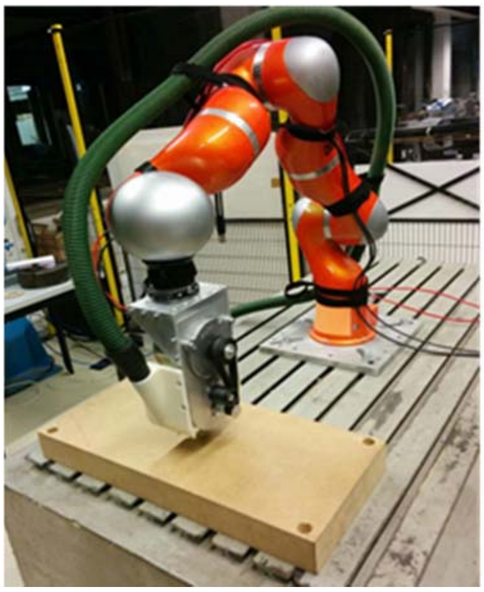
Figure 2: picture of the setup.

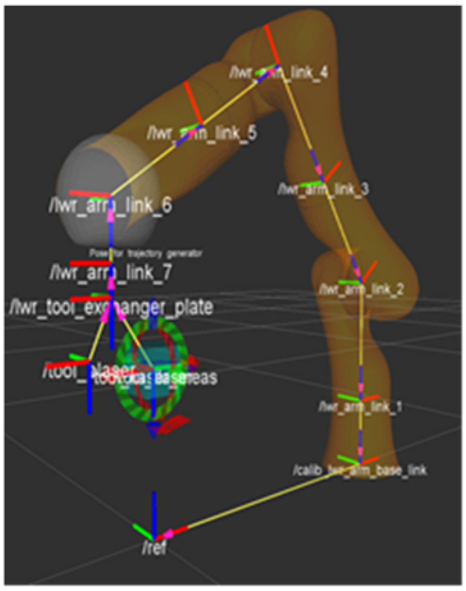

Figure 3: visualization of the setup using rviz, the red lines indicate the measured surface.

2 Software integration (including integration of the early results of Task 2

The Software architecture

The software architecture applies the Composition Pattern [1], which provides a practical way to implement the 5C's principle of separation of concerns. Most significantly, this design improved the scheduling and hence resulted in lower latencies and soft real-time performance on a soft real-time operating system. Figure 4 gives an overview.

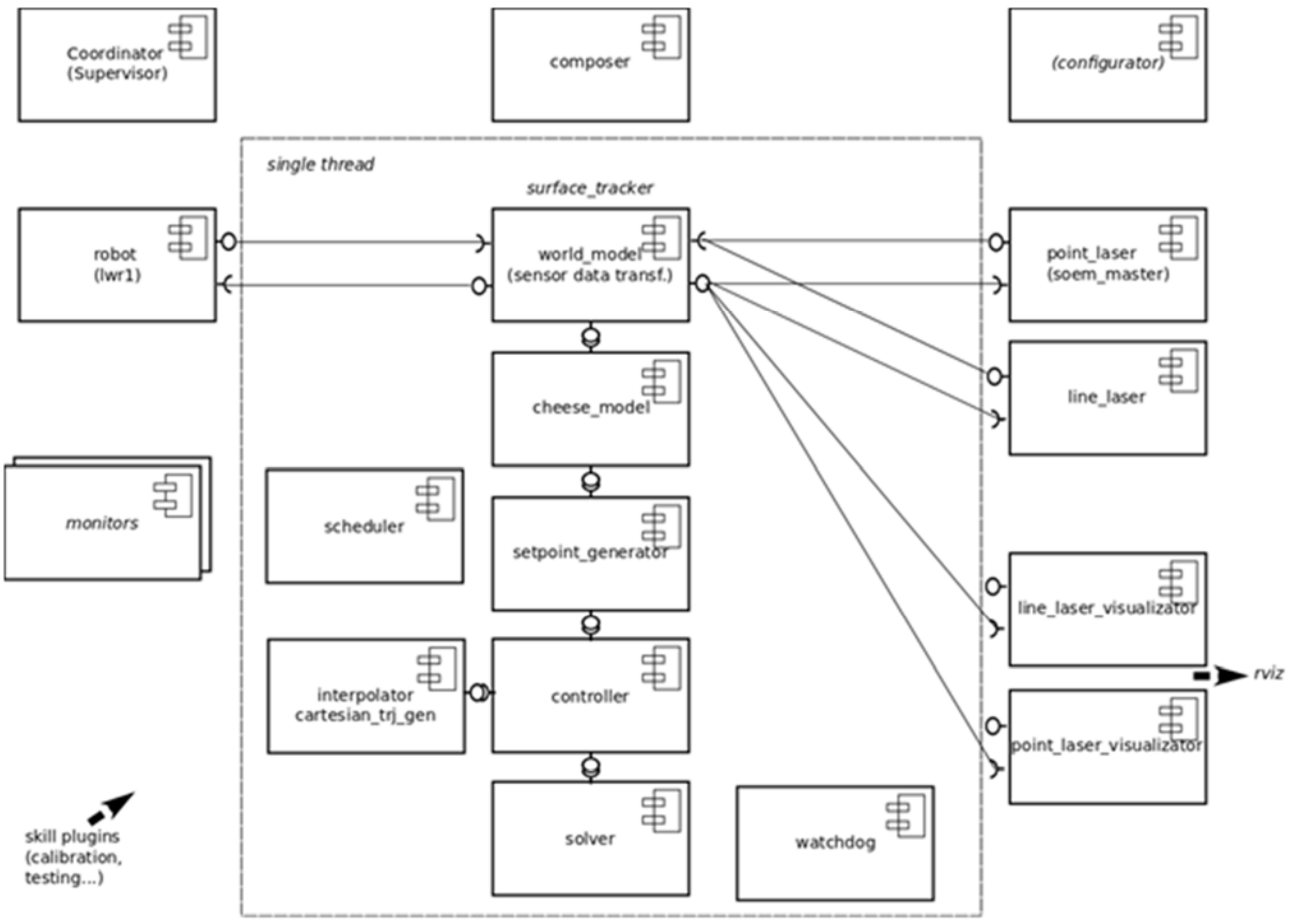

Figure 4: software architecture

Link with eTaSL

In the early simulations, reported in RAP1, we worked out all details of the controllers manually.

These details include the model-based formulation of feedforward and feedback, and task constraints.

This approach is time consuming and error prone due to the manual calculations to be made.

Therefore, we developed a higher-level task description in the expression-graph based task specification language (eTaSL)[2]. EtaSL is a Domain Specific Language to describe the relations between different features on the robots and objects involved in the application, and the constraints that are imposed on them.

The biggest advantages are that 1) the task programmer is less burdened with calculating the mathematical derivations involved in the description and specification of the aforementioned relations and constraints, for example the curvature and torsion of the local surface model, and 2) the task programmer is less burdened with the programming language details, since the underlying framework instantiates the required software components and algorithms from the task description.

Based on our iterations on the specification of the application in eTaSL, we defined the necessary extensions of eTaSL and the underlying software tools to make it possible to instantiate the application from this higher-level task description.

[1] Vanthienen, D., De Schutter, J. (sup.), Bruyninckx, H. (cosup.) (2015). *Composition Pattern for Constraint-based Programming with Application to Force-sensorless Robot Tasks, 285 pp.*

[2] Aertbelien, E., De Schutter, J. (2014), Etasl/eTC: *A constraint-based task specification language and robot controller using expression graphs.* 2014 IEEE/RSJ International Conference on