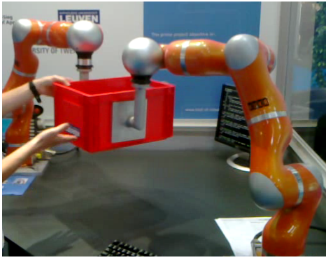
Task specification, control and estimation (plus some "lessons learned")

Herman Bruyninckx
Dept Mechanical Engineering, K.U.Leuven, Belgium
http://www.orocos.org

Paris, November 10, 2010

Overview

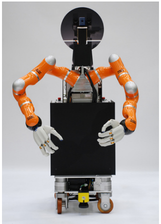
- **Ambition**: to integrate planning, sensing, control, and reasoning, at all levels of abstraction, and supported by FOSS[1]
- The **Task – Skill – Motion** paradigm (a.k.a.: *How to do this kind of manipulation?)*



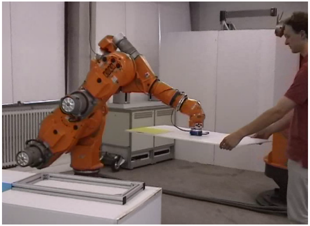
[1] *Free & Open Source Software*

Overview (2)

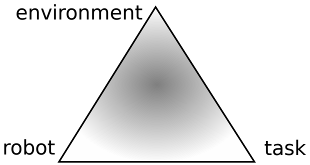
Medium-term ambition: to make these. . .

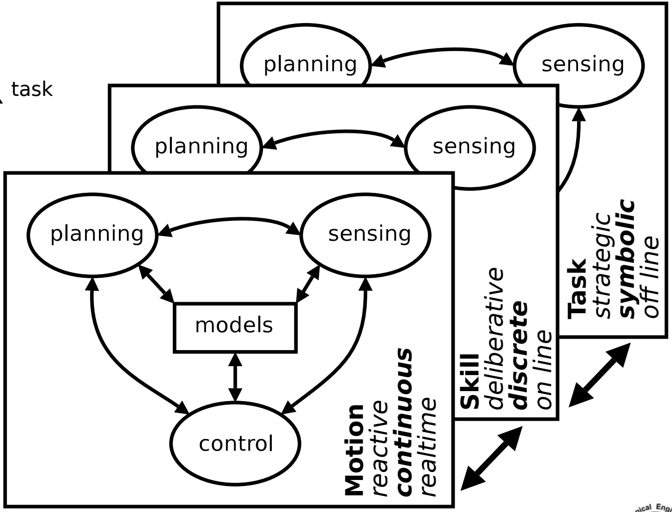


. . . move like this:



Robot systems in two figures





Software frameworks for robotics

The big four in FOSS:[2]

- Orocos ("Europe")
- OpenRTM ("Japan")
- ROS ("USA")
- OPRoS ("Korea")

The French one:

- Genom

[2] *Free & Open Source Software*

Software frameworks for robotics

The big four in FOSS:[2]

- Orocos ("Europe")
- OpenRTM ("Japan")
- ROS ("USA")
- OPRoS ("Korea")

The French one:

- Genom

Lessons learned:

- very similar in scope, goals and design :-)
- mostly non-interoperable :-(
- strong **Not Invented Here** reflexes :-(

[2] *Free & Open Source Software*

Robotic motion & manipulation

Lessons learned:

- "planning people" want to solve it by planning; "control guys" by control; "3D perceptionists" by sensing,. . .

$\Rightarrow$ different "domains" should **know where to stop**, and start using the **other** domains

Robotic motion & manipulation

Lessons learned:

- "planning people" want to solve it by planning; "control guys" by control; "3D perceptionists" by sensing,. . .

⇒ different "domains" should **know where to stop**, and start using the **other** domains

- most important *showstoppers*:
  - lack of discrete & **continuous** *Coordination*
  - too large-grained software modularity.
    E.g., there **is** planning & sensing in most of control software, and **vice versa**

Most robots move like a robot. . .

Because current approach is **still mostly traditional** **Sense–Plan–Act**:

- emphasis on (only) **geometric, static** **planning**
- not well connected to "traditional" **control**
- **uni-directional, input–output, hard set-point** "stack" hierarchies

Most robots move like a robot. . .

Because current approach is **still mostly traditional** **Sense–Plan–Act**:

- emphasis on (only) **geometric, static planning**
- not well connected to “traditional” **control**
- **uni-directional, input–output, hard set-point** “stack” hierarchies

Lesson learned:

- software/design/**specification** are not ready because of unawareness about **“4C” separation of concerns**: Computation, Communication, Configuration, & Coordination

4C best practice[3]

C1 **Computation**

C2 **Communication**

C3 **Configuration**

C4 **Coordination**

[3] Radestock & Eisenbach, *Coordination in evolving systems*, 1996

4C best practice[3]

C1 **Computation**: the useful functionality within the components that (hopefully) *pays the bill*

C2 **Communication**: overhead of supporting components/nodes to exchange data

C3 **Configuration**: which components have to interact with which other components?

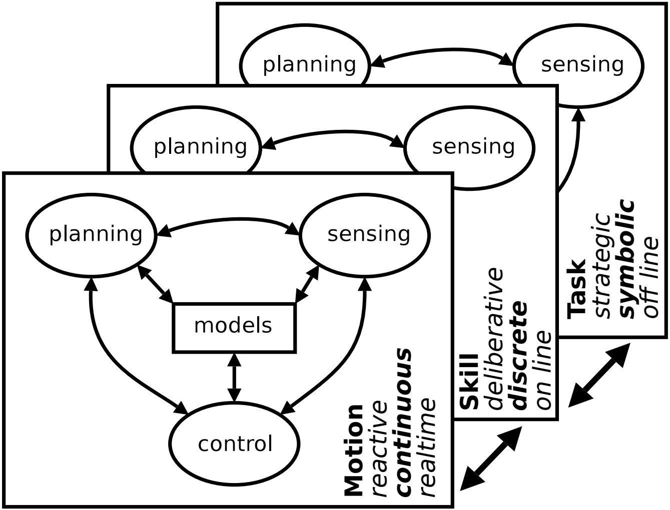
C4 **Coordination**: to help components in switching their functional behaviour in a coordinated way

Holds for hardware, algorithms, middleware,. . . !

[3] Radestock & Eisenbach, *Coordination in evolving systems*, 1996

Task—Skill—Motion

Best practice **three level** "architecture"



Three-level (meta) architecture

- first(?) **explicit** description: Saridis 1977
  - *Organization, Coordination, Direct control*
  - *Increasing order of intelligence, decreasing order of precision*
- is known under **various other names** (e.g., *strategic, deliberative, reactive*,. . .)
- research challenges for coming decade:
  - **more reasoning/intelligence** in all levels
  - to integrate all levels

My definitions

- **Motion**:
- **Skill**:
- **Task**:

My definitions

- **Motion**: a **continuous** **time/space activity** of a robot, moving its **joints and/or tool(s)** in a specified way, **until some constraint is violated** that can be **checked by sensors**.
- **Skill**:
- **Task**:

My definitions

- **Motion**: a **continuous time/space activity** of a robot, moving its **joints and/or tool(s)** in a specified way, **until some constraint is violated** that can be **checked by sensors**.
  (Extremely simple) examples:
  - a **force-controlled peg-in-hole** motion, terminated by reaching a force threshold in the insertion direction
  - a **force-guarded approach motion** in free space, terminated by sensing a non-zero approach force.
- **Skill**:
- **Task**:

My definitions

- **Motion**:
- **Skill**: a **discrete** state automaton (FSM), in which each State **runs** one single **Motion**, and each violation of a motion constraint (can) give rise to a **transition event**.
- **Task**:

My definitions

- **Motion**:
- **Skill**: a **discrete** state automaton (FSM), in which each State **runs** one single **Motion**, and each violation of a motion constraint (can) give rise to a **transition event**.
  (Extremely simple) examples:
  - **assemble a peg into a hole**: approach, find hole, align, insert
  - **opening a door**: locating the handle, reaching out to grasp it, grasping it, opening the door
- **Task**:

My definitions

- **Motion**:
- **Skill**:
- **Task**: **symbolic constraints** between sub-Tasks (= partial fulfilment of the whole Task), in which each transition between two such sub-Tasks (compatible with the constraints) is realised by one out of a set of appropriate Skills.

My definitions

- **Motion**:
- **Skill**:
- **Task**: **symbolic constraints** between sub-Tasks (= partial fulfilment of the whole Task), in which each transition between two such sub-Tasks (compatible with the constraints) is realised by one out of a set of appropriate Skills. (Not so extremely simple) example: **bring a bottle of beer from the fridge**

Intermediate reflections

- Skills are the **"glue"** between the symbolic and the real worlds
- **reasoning** can take place at all levels
- **hierarchy** can exist at all levels.
- main / major / inevitable **research error**: try to apply solutions fit for one level to problems at other levels.

Intermediate overview

Our research in Motion/Skill specification & execution:

- **Past**: (single) Task Frame Formalism **Motions**
- **(Recent) Past**: (multiple) Feature Frame Formalism **Motions** ("iTaSC")
- **Present**: **Skills**
- **Future**: **Tasks**

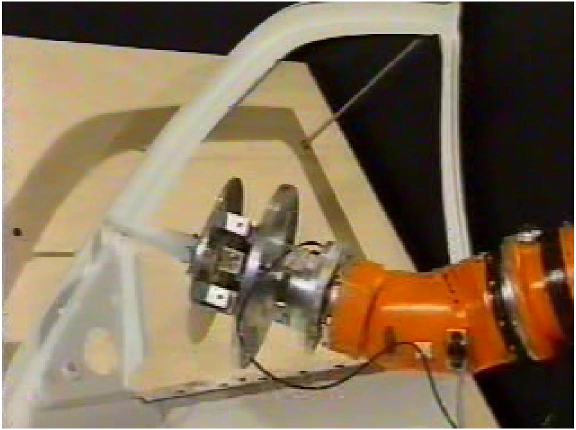
Intermediate overview

Our research in Motion/Skill specification & execution:

- **Past**: (single) Task Frame Formalism **Motions**
- **(Recent) Past**: (multiple) Feature Frame Formalism **Motions** ("iTaSC")
- **Present**: **Skills**
- **Future**: **Tasks**

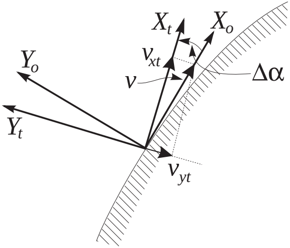
All the time: the search for *best practices* in **software** support

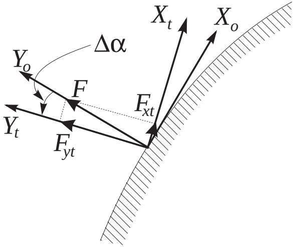
Past (1985–2000)
—Task Frame Formalism—



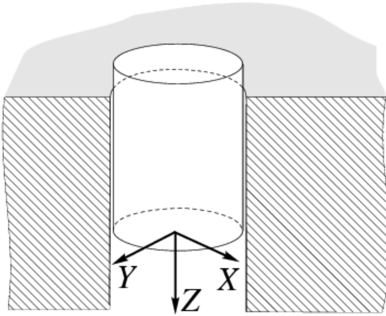
Task Frame Formalism (2)

- Single frame with six DOFs.
- Explicit setpoints (= hard, uni-directional constraints)
- Velocity + Force.
- Only serial “skill” logic.
- Sensor-based tracking. (E.g., force, vision.)

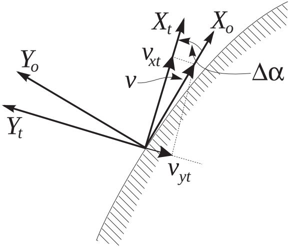


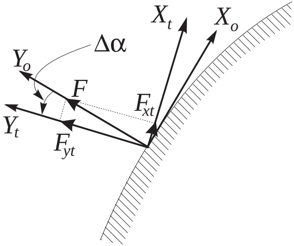


```
move compliantly {
 with task frame directions
      xt:  force 0 N
      yt:  force 0 N
      zt:  velocity v mm/sec
      axt: force 0 Nmm
      ayt: force 0 Nmm
      axt velocity 0 rad/sec
 } until zt force < -f N
```



```
move compliantly {
  with task frame directions
       xt: velocity v mm/sec
       yt: force f N
       zt: velocity 0 mm/sec
       axt: velocity 0 rad/sec
       ayt: velocity 0 rad/sec
       azt: track (on velocities)
  } until until distance > d mm
```



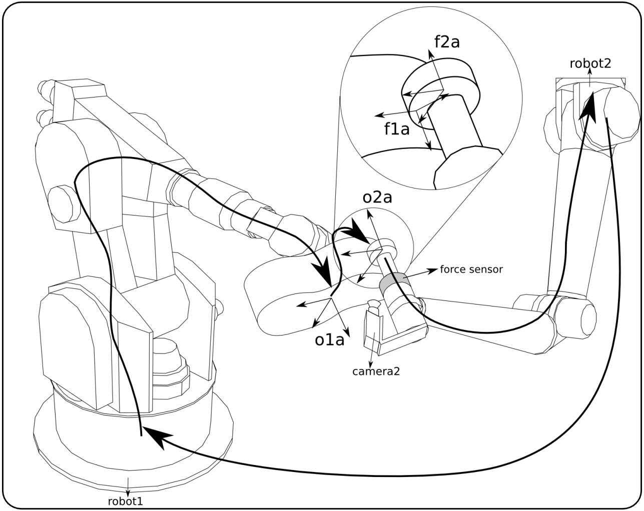


Past (2004–2009)
—iTaSC Motions—

"Instantaneous Task Specification with Constraints":

- **multiple** frames...
- ... with **partial** specification per frame...
- ... and **constraint-based** i.s.o. setpoints.
- **planning** & **estimation** can be included.

Modelling primitive: kinematic loops



Motions: methodology

Step 1.: "Scene graph" model

- geometric relationships between "feature frames"
- assign control, uncertainty,. . . to each feature.

Motions: methodology

Step 1.: "Scene graph" model

- geometric relationships between "feature frames"
- assign control, uncertainty,...to each feature.

Step 2a.: Global objective function(s)

Step 2b.: Constraint specification per feature

Motions: methodology

Step 1.: "Scene graph" model

- geometric relationships between "feature frames"
- assign control, uncertainty, . . . to each feature.

Step 2a.: Global objective function(s)

Step 2b.: Constraint specification per feature

Step 3.: Solve the resulting constrained optimization problem

Motions: methodology

Step 1.: "Scene graph" model

- geometric relationships between "feature frames"
- assign control, uncertainty,... to each feature.

Step 2a.: Global objective function(s)

Step 2b.: Constraint specification per feature

Step 3.: Solve the resulting constrained optimization problem

Step 4.: Update the "scene graph" and iterate Step 3.

Present: Skills

- **Modelling**
- **Configuration**
- **Computation**
- **Coordination**

Present: Skills

- **Modelling** ("scene graph")
- **Configuration**
- **Computation**
- **Coordination**

Present: Skills

- **Modelling** ("scene graph")
- **Configuration** ("constraints"):
  - motion: instantaneous, trajectory primitives, interaction,...
    including weights between motion primitives and objective functions
  - learning: model parameter priors
- **Computation**
- **Coordination**

Present: Skills

- **Modelling** ("scene graph")
- **Configuration** ("constraints"):
  - motion: instantaneous, trajectory primitives, interaction,...
    including weights between motion primitives and objective functions
  - learning: model parameter priors
- **Computation** ("weighted constrained optimization")
  - instantaneous motion solver
  - estimation/learning/reasoning calculations

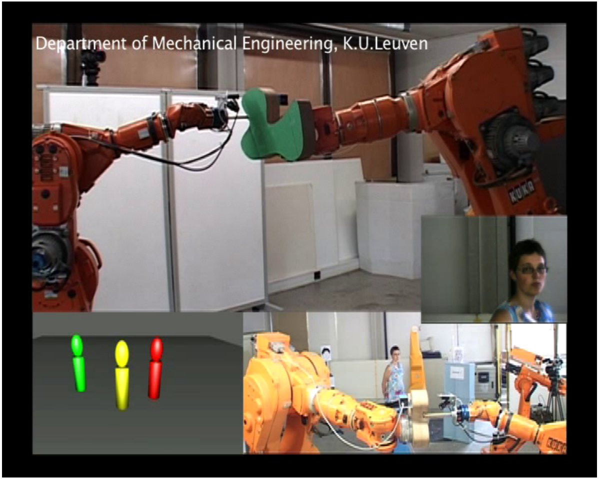
  includes monitoring of constraint **violations**!
- **Coordination**

Present: Skills

- **Modelling** ("scene graph")
- **Configuration** ("constraints"):
  - motion: instantaneous, trajectory primitives, interaction,...
    including weights between motion primitives and objective functions
  - learning: model parameter priors
- **Computation** ("weighted constrained optimization")
  - instantaneous motion solver
  - estimation/learning/reasoning calculations

  includes monitoring of constraint **violations**!
- **Coordination**: constraint violation event-driven FSM, including "discrete scheduling" of Computations

Example experiment
—Human-aware dual-arm Skill—



Summary

- our paradigm: a **methodological** way of **specifying** Skills and Motions
  methodology = **4C** + constrained optimiz.
- **constraint-based**:
  - (**soft**) constraints are **composable & bi-directional**
  - constraints = **knowledge relationships**
  - allow **single-concept** **integration** of cognition and reasoning at all three **levels of abstraction** (Task, Skill, Motion)
- **multi-frame**, **partial** specification
- **scene graph** is central **shared** **resource**
- **traditional** *Sense-Plan-Act* is **smooth** limit case

Summary (2)

Integration of

- **planning**: plan is just another constraint
- **behaviour based** approach: behaviours generate constraints, and not directly motion setpoints
- **high-level reasoning**: name of Skill states = **symbol grounding**

Summary (2)

Integration of

- **planning**: plan is just another constraint
- **behaviour based** approach: behaviours generate constraints, and not directly motion setpoints
- **high-level reasoning**: name of Skill states = **symbol grounding**

"To ROS or not to ROS...?"

Summary (2)

Integration of

- **planning**: plan is just another constraint
- **behaviour based** approach: behaviours generate constraints, and not directly motion setpoints
- **high-level reasoning**: name of Skill states = **symbol grounding**

"To ROS or not to ROS. . . ?"

- Wrong question!

$\Rightarrow$ *Let's make (open source) robotics software more professional, hence interoperable, worldwide!*

Summary of presented paradigm (3)

Lesson learned:

- currently *best practice* and most impressive[a] implementations of our paradigm: **DLR Justin**. . .
- . . . using Simulink/RTW, and no FOSS. . .

[a] *Coffee making* video does *not* need "×10" annotation. . .