Planning Sequences of Motion Primitives for humanoid robots

Florent Lamiraux, Sébastien Dalibard, Alireza Nakhaei and J.-P. Laumond

LAAS-CNRS

May 9, 2011

Objective

- Plan motions for humanoid robot with manipulation of simple objects like:
  - doors,
  - windows,
  - drawers.
- Execute these motions using sensor feedback control.

Objective

- Plan motions for humanoid robot with manipulation of simple objects like:
  - doors,
  - windows,
  - drawers.
- Execute these motions using sensor feedback control.

Approach

- Reduce complexity of manipulation planning:
  - “documented object”
- Plan motions as sequences of motion primitives

Motion primitive

- Definition
  - Motion produced by a controller.
- Examples
  - walking along a curve,
  - walking on foot prints,
  - reaching with a hand.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Documented objects

- Manipulating some objects requires some knowledge
  - opening a door requires
    - to grasp the handle,
    - to turn the handle,
    - to pull the handle along a circular path,
    - to cross the doorway,
    - to grasp the other handle and release the first one,
    - to pull the handle along a circular path,
    - to turn the handle.
- We put this knowlegde into the object thus defining the notion of
  - *documented object*.
- For each object, motion primitives manipulating the object can be precomputed and inserted in a global roadmap.

Example: going through a door

- Ideally, the knowledge is stored into the object as a sequence of time-varying tasks.
- Our implementation:
  - system: robot bounding box + door (3+1 dof),
  - algorithm: plan a sequence of motions satisfying successive constraints:
    1. no constraint,
    2. left hand is close to the handle,
    3. previous constraint + right hand close to other handle,
    4. right hand close to other handle,
    5. no constraint
  - generate step sequence along box path,
  - build a whole-body motion defined by
    - step sequence,
    - grasping constraints.

Example: going through a door

- Ideally, the knowledge is stored into the object as a sequence of time-varying tasks.
- Our implementation:
  - system: robot bounding box + door (3+1 dof),
  - algorithm: plan a sequence of motions satisfying successive constraints:
    1. no constraint,
    2. left hand is close to the handle,
    3. previous constraint + right hand close to other handle,
    4. right hand close to other handle,
    5. no constraint
  - generate step sequence along box path,
  - build a whole-body motion defined by
    - step sequence,
    - grasping constraints.

Example: going through a door

- Ideally, the knowledge is stored into the object as a sequence of time-varying tasks.
- Our implementation:
  - system: robot bounding box + door (3+1 dof),
  - algorithm: plan a sequence of motions satisfying successive constraints:
    1. no constraint,
    2. left hand is close to the handle,
    3. previous constraint + right hand close to other handle,
    4. right hand close to other handle,
    5. no constraint
  - generate step sequence along box path,
  - build a whole-body motion defined by
    - ⋆ step sequence,
    - ⋆ grasping constraints.

Example: going through a door

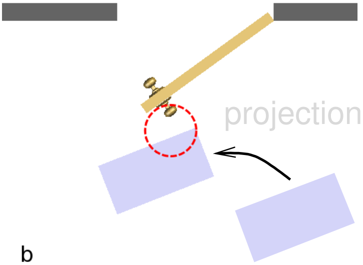
- Ideally, the knowledge is stored into the object as a sequence of time-varying tasks.
- Our implementation:
  - system: robot bounding box + door (3+1 dof),
  - algorithm: plan a sequence of motions satisfying successive constraints:
    1. no constraint,
    2. left hand is close to the handle,
    3. previous constraint + right hand close to other handle,
    4. right hand close to other handle,
    5. no constraint
  - generate step sequence along box path,
  - build a whole-body motion defined by
    - step sequence,
    - grasping constraints.

Example: going through a door

- Ideally, the knowledge is stored into the object as a sequence of time-varying tasks.
- Our implementation:
  - system: robot bounding box + door (3+1 dof),
  - algorithm: plan a sequence of motions satisfying successive constraints:
    1. no constraint,
    2. left hand is close to the handle,
    3. previous constraint + right hand close to other handle,
    4. right hand close to other handle,
    5. no constraint
  - generate step sequence along box path,
  - build a whole-body motion defined by
    - step sequence,
    - grasping constraints.

Motion Planning constraint

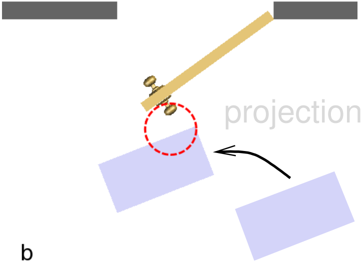
- **Definition**: sub-manifold or region of the configuration space.
- **Sampling**: projection function from configuration space to domain satisfying the constraint.



- projection functions move the bounding box in order to put door handles into reaching area of robot hands.

Motion Planning constraint

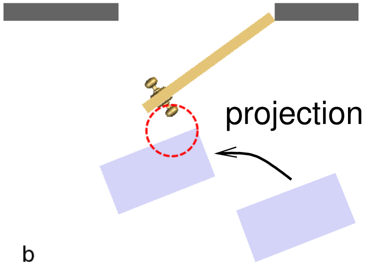
- **Definition**: sub-manifold or region of the configuration space.
- **Sampling**: projection function from configuration space to domain satisfying the constraint.



- projection functions move the bounding box in order to put door handles into reaching area of robot hands.

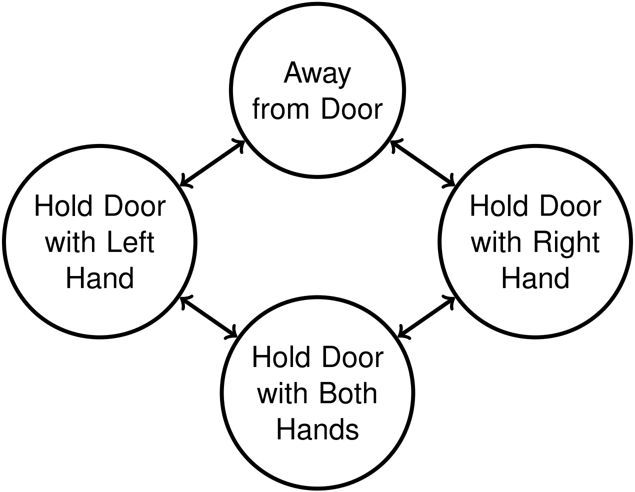
Motion Planning constraint

- **Definition**: sub-manifold or region of the configuration space.
- **Sampling**: projection function from configuration space to domain satisfying the constraint.



- projection functions move the bounding box in order to put door handles into reaching area of robot hands.

Going through a door: constraint transition graph



Going through a door: algorithm

- Configuration space:
  $SE(2) \times [\alpha_{min}, \alpha_{max}] \times \{\text{free, left hand, right hand, both hands}\}$
- Classical RRT algorithm with dedicated methods
  - steering method: connect two configurations if
    - states are adjacent in constraint graph,
    - motion of object is consistent,
    - enforce weaker constraint,
  - distance function
    - return ∞ when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.

Going through a door: algorithm

- Configuration space:
  $SE(2) \times [\alpha_{min}, \alpha_{max}] \times \{\text{free, left hand, right hand, both hands}\}$
- Classical RRT algorithm with dedicated methods
  - steering method: connect two configurations if
    - states are adjacent in constraint graph,
    - motion of object is consistent,
    - enforce weaker constraint,
  - distance function
    - return ∞ when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.

Going through a door: algorithm

- Configuration space:
  $SE(2) \times [\alpha_{min}, \alpha_{max}] \times \{\text{free}, \text{left hand}, \text{right hand}, \text{both hands}\}$
- Classical RRT algorithm with dedicated methods
  - steering method: connect two configurations if
    - states are adjacent in constraint graph,
    - motion of object is consistent,
    - enforce weaker constraint,
  - distance function
    - return ∞ when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.

Going through a door: algorithm

- Configuration space:
  $SE(2) \times [\alpha_{min}, \alpha_{max}] \times \{\text{free}, \text{left hand}, \text{right hand}, \text{both hands}\}$
- Classical RRT algorithm with dedicated methods
  - steering method: connect two configurations if
    - states are adjacent in constraint graph,
    - motion of object is consistent,
    - enforce weaker constraint,
  - distance function
    - return ∞ when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.

Going through a door: algorithm

- Configuration space:
  $SE(2) \times [\alpha_{min}, \alpha_{max}] \times \{\text{free}, \text{left hand}, \text{right hand}, \text{both hands}\}$
- Classical RRT algorithm with dedicated methods
  - steering method: connect two configurations if
    - states are adjacent in constraint graph,
    - motion of object is consistent,
    - enforce weaker constraint,
  - distance function
    - return ∞ when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.

Going through a door: algorithm

- Configuration space:
  $SE(2) \times [\alpha_{min}, \alpha_{max}] \times \{\text{free}, \text{left hand}, \text{right hand}, \text{both hands}\}$
- Classical RRT algorithm with dedicated methods
  - steering method: connect two configurations if
    - states are adjacent in constraint graph,
    - motion of object is consistent,
    - enforce weaker constraint,
  - distance function
    - return ∞ when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.

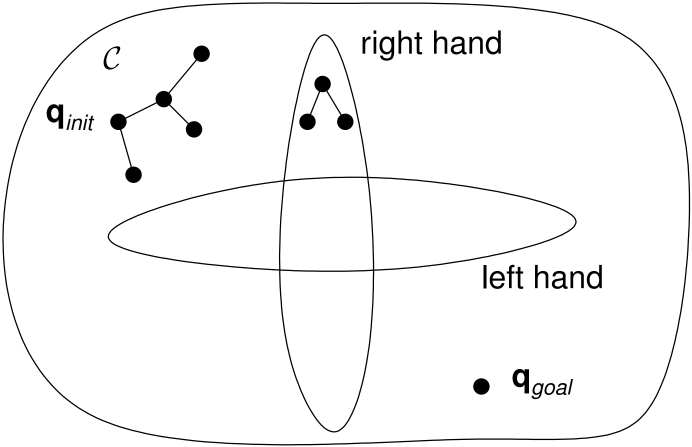
Going through a door: algorithm

- Configuration space:
  $SE(2) \times [\alpha_{min}, \alpha_{max}] \times \{\text{free}, \text{left hand}, \text{right hand}, \text{both hands}\}$
- Classical RRT algorithm with dedicated methods
  - steering method: connect two configurations if
    - states are adjacent in constraint graph,
    - motion of object is consistent,
    - enforce weaker constraint,
  - distance function
    - return ∞ when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.

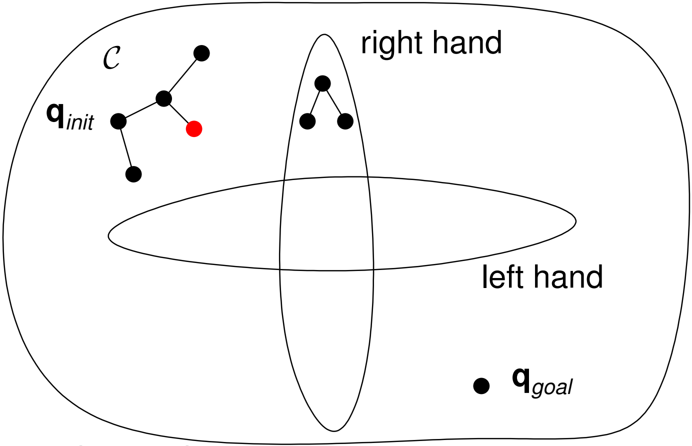
Going through a door: algorithm

- Configuration space:
  $SE(2) \times [\alpha_{min}, \alpha_{max}] \times \{\text{free}, \text{left hand}, \text{right hand}, \text{both hands}\}$
- Classical RRT algorithm with dedicated methods
  - steering method: connect two configurations if
    - states are adjacent in constraint graph,
    - motion of object is consistent,
    - enforce weaker constraint,
  - distance function
    - return ∞ when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.

Going through a door: algorithm

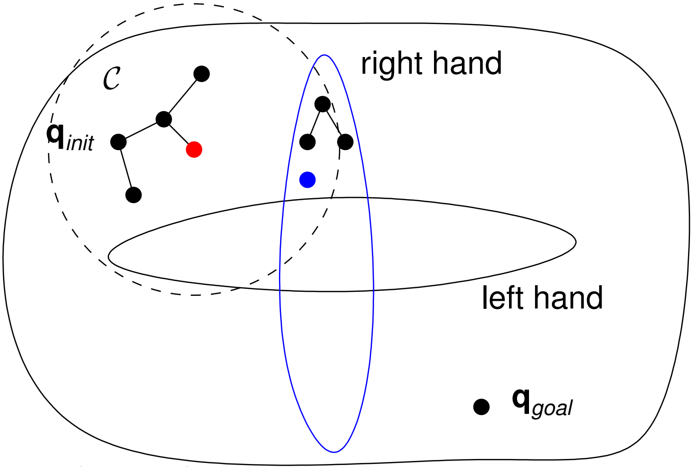


Going through a door: algorithm



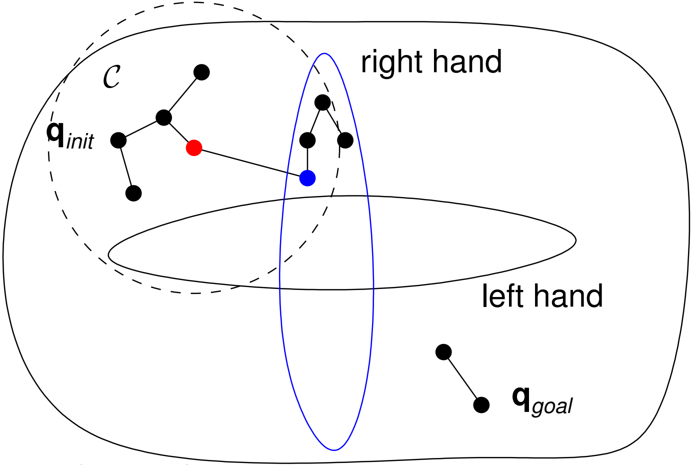
1. pick a random node

Going through a door: algorithm



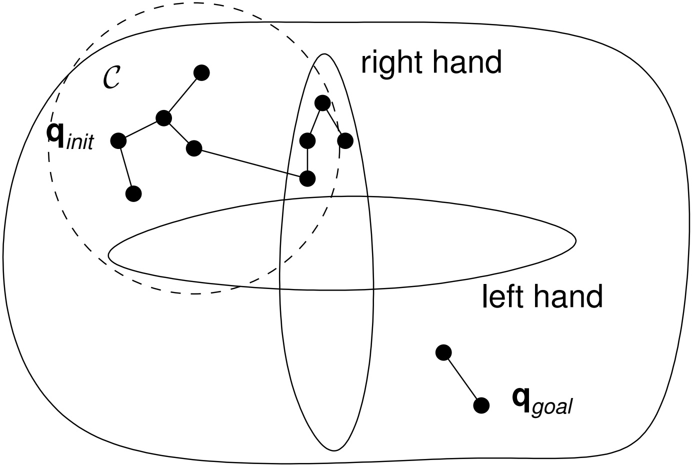
1. pick a random node
2. randomly sample a config about the node with the same object config

Going through a door: algorithm

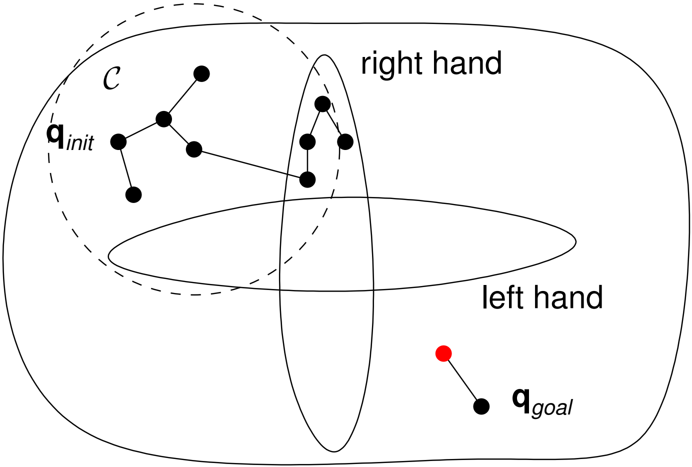


1. pick a random node
2. randomly sample a config about the node with the same object config
3. expand each connected component

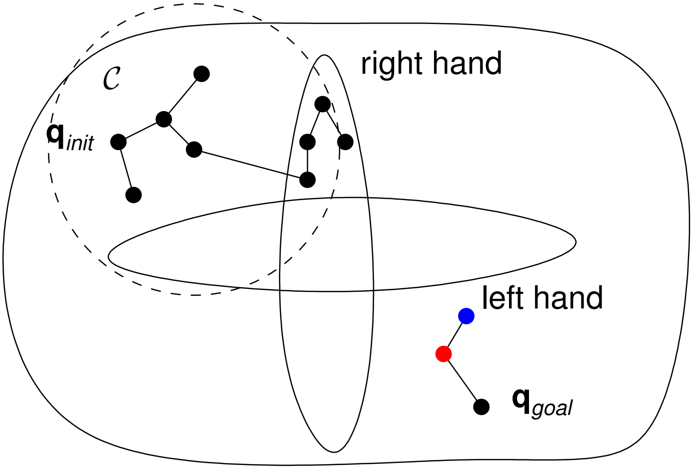
Going through a door: algorithm



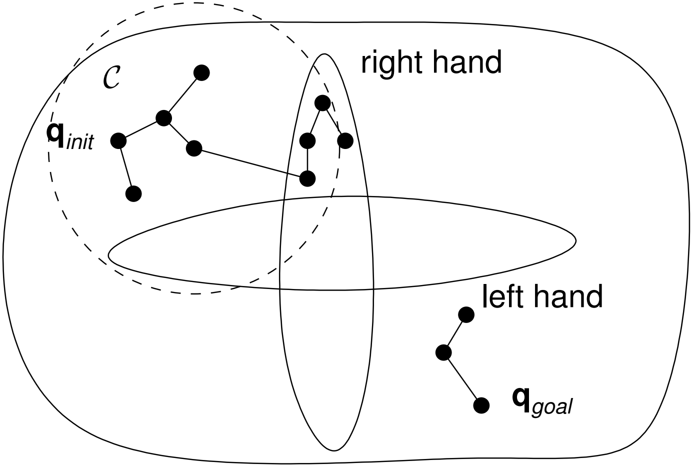
Going through a door: algorithm



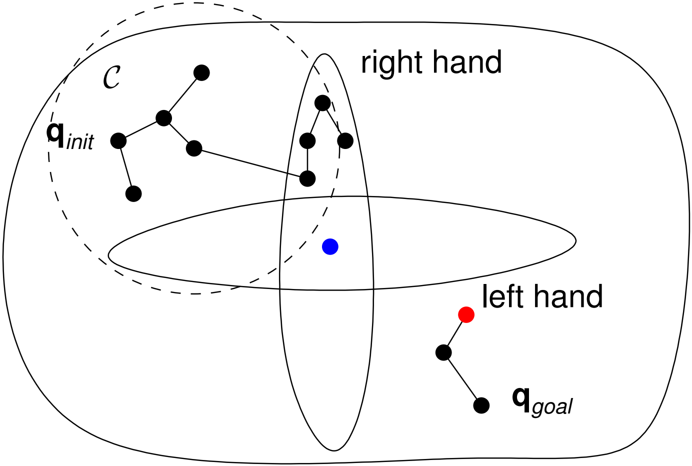
Going through a door: algorithm



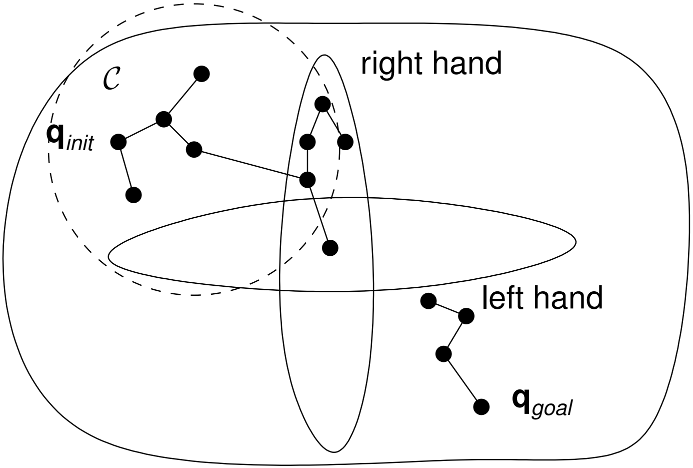
Going through a door: algorithm



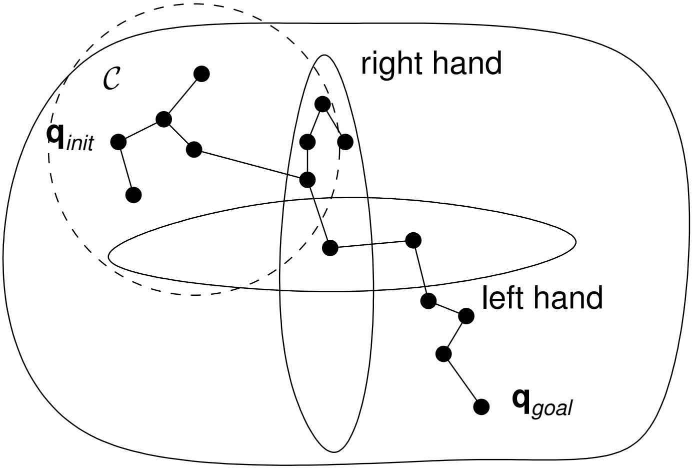
Going through a door: algorithm



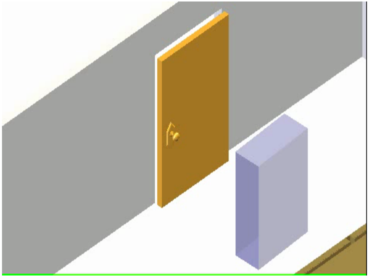
Going through a door: algorithm



Going through a door: algorithm



Going through a door: motion planning results



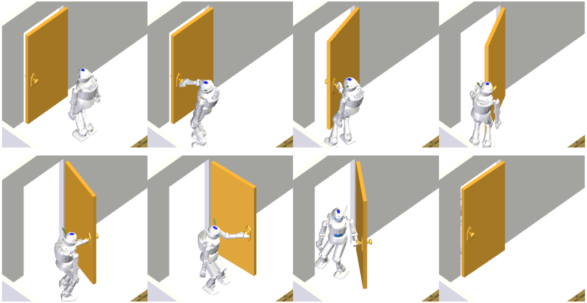
Going through a door: whole-body animation

- Motion of the bounding box is converted into time parameterized
  - step sequence,
  - motions of end effectors
- Resolution
  - step sequence → COM: preview control,
  - feet and upper body: inverse kinematics.

Going through a door: whole-body animation

- Motion of the bounding box is converted into time parameterized
  - step sequence,
  - motions of end effectors
- Resolution
  - step sequence → COM: preview control,
  - feet and upper body: inverse kinematics.

Going through a door: animation



From motion of bounding box to motion primitives

- Objective
  - transform result of path planning into sequence of controllers.

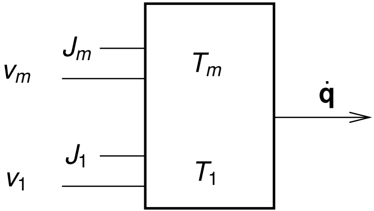
Hierarchical task based control framework

- Task
  1. Function of configuration to be controlled to 0,
  2. Jacobian of the function.
- Stack of tasks
  - tasks in decreasing order of priority,
  - compute velocity by cascade of pseudo inverses. [Siciliano, Slotine 1991]

Hierarchical task based control framework

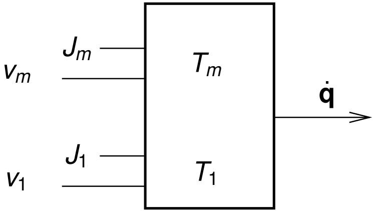
- Task
  1. Function of configuration to be controlled to 0,
  2. Jacobian of the function.
- Stack of tasks
  - tasks in decreasing order of priority,
  - compute velocity by cascade of pseudo inverses. [Siciliano, Slotine 1991]

Stack of tasks: collaboration with JRL -Tsukuba Japan



$$\dot{\mathbf{q}}_1 = -\lambda_1 J_1^+ v_1 \quad \rightarrow \quad \dot{v}_1 = -\lambda_1 v1$$

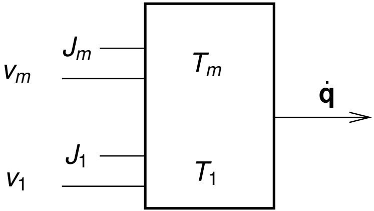
Stack of tasks



$$\dot{\mathbf{q}}_1 = -\lambda_1 J_1^+ v_1 \quad \rightarrow \quad \dot{v}_1 = -\lambda_1 v1$$

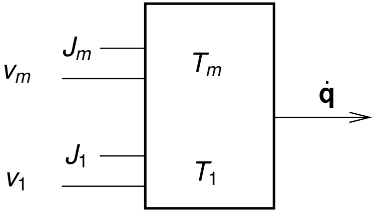
$$\dot{\mathbf{q}}_i = \qquad\quad - \lambda_i \; J_i \qquad\quad {}^+ \; v_i$$

Stack of tasks



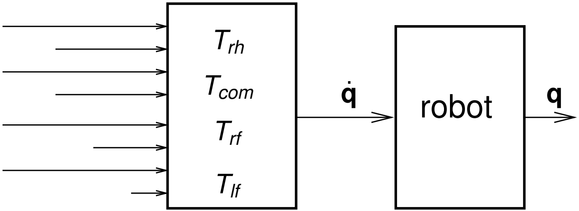
$$
\begin{aligned}
&\dot{\mathbf{q}}_1 = -\lambda_1 J_1^+ v_1 \quad \rightarrow \quad \dot{v}_1 = -\lambda_1 v1 \\
&\dot{\mathbf{q}}_i = \dot{\mathbf{q}}_{i-1} - \lambda_i (J_i P_{i-1})^+ (v_i - J_i \dot{\mathbf{q}}_{i-1}) \\
&P_i = P_{i-1} - (J_i P_{i-1})^+ J_i P_{i-1}
\end{aligned}
$$

Stack of tasks

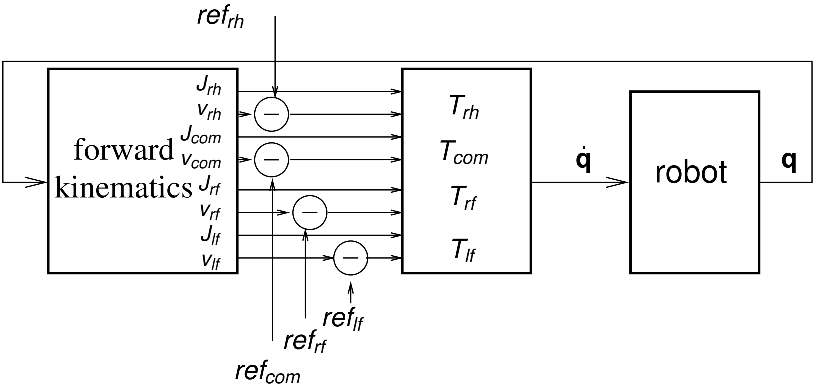


$$
\begin{aligned}
\dot{\mathbf{q}}_1 &= -\lambda_1 J_1^+ v_1 \quad \rightarrow \quad \dot{v}_1 = -\lambda_1 v1 \\
\dot{\mathbf{q}}_i &= \dot{\mathbf{q}}_{i-1} - \lambda_i (J_i P_{i-1})^+ (v_i - J_i \dot{\mathbf{q}}_{i-1}) \\
P_i &= P_{i-1} - (J_i P_{i-1})^+ J_i P_{i-1} \\
\dot{\mathbf{q}} &= \dot{\mathbf{q}}_m
\end{aligned}
$$

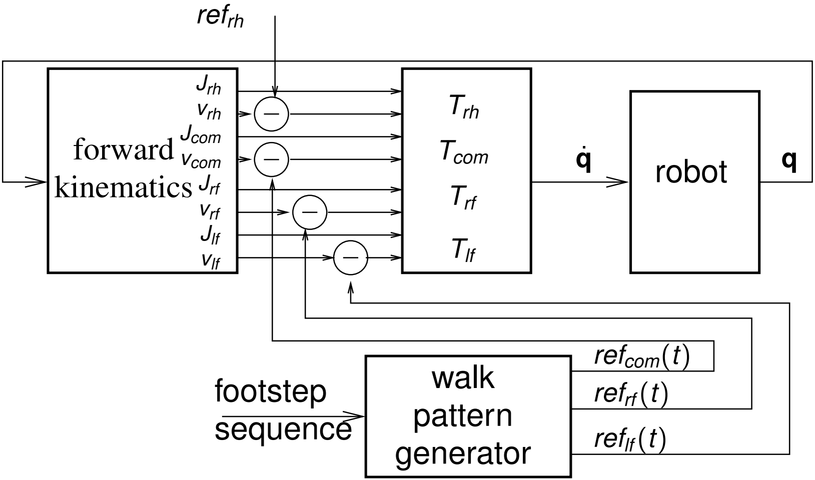
Stack of tasks



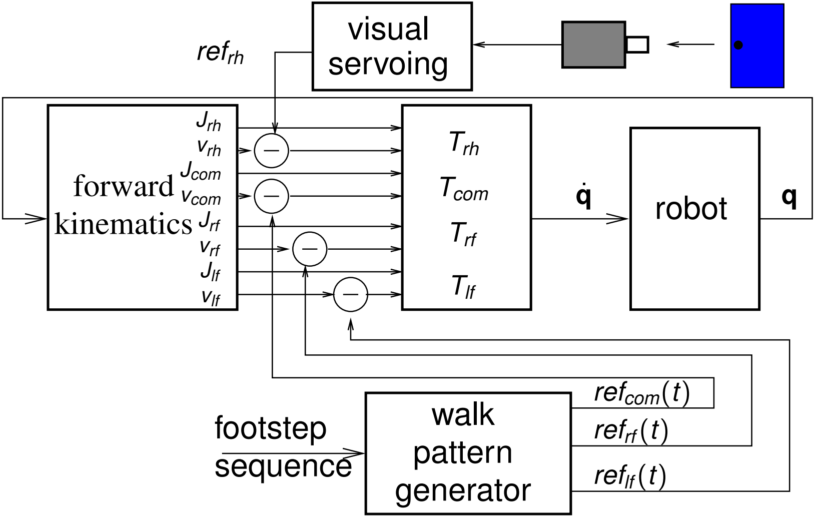
Stack of tasks



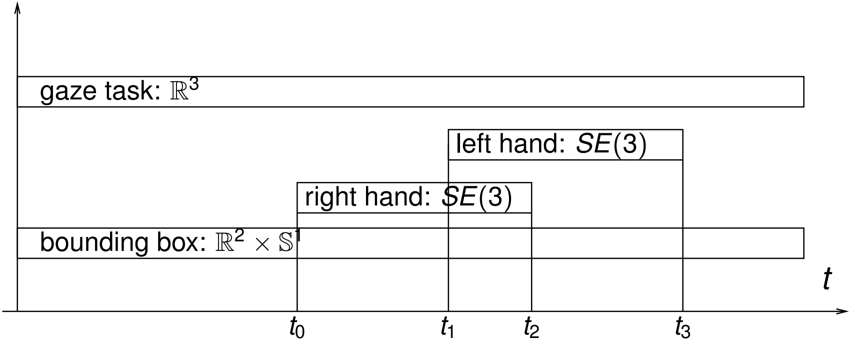
Stack of tasks



Stack of tasks

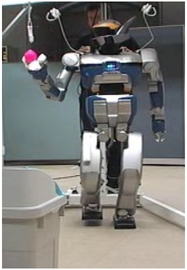


Motion plan



gaze task: $\mathbb{R}^3$
left hand: $SE(3)$
right hand: $SE(3)$
bounding box: $\mathbb{R}^2 \times \mathbb{S}^1$
t
t_0
t_1
t_2
t_3

Preview



- Grasping a ball while walking, N. Mansard, O. Stasse, JRL - Tsukuba 2007.

Conclusion

- Let us do it on the physical robot