Motion planning for every-day tasks in human environment

Alireza Nakhaei   Sébastien Dalibard
Florent Lamiraux   Jean-Paul Laumond

LAAS-CNRS

October 4, 2010
Introduction

Objectives
- Develop algorithms to enable a humanoid robot to perform every-day tasks in a human environment.
- We focus on geometric motion planning.

Context
- Work in collaboration with Toyota Motor Europe.
Introduction

Objectives
- Develop algorithms to enable a humanoid robot to perform every-day tasks in a human environment.
- We focus on geometric motion planning.

Context
- Work in collaboration with Toyota Motor Europe.
Types of tasks addressed

- From
  - grasping an object,
- to
  - opening doors, windows and drawers,
- making computation time reasonable.
Types of tasks addressed

- From
  - grasping an object,
- to
  - opening doors, windows and drawers,
- making computation time reasonable.
Motion planning

Definition

- Given
  - a robot,
  - obstacles in the workspace,
  - an initial and a goal configurations,

- compute
  - a collision-free path from the initial to the goal configuration.
Definition

- Given
  - a robot,
  - obstacles in the workspace,
  - an initial and a goal configurations,
- compute
  - a collision-free path from the initial to the goal configuration.
Motion planning

- Also called
  - the piano mover’s problem
Most popular method:

- Randomly Rapidly exploring Tree (RRT)

expand a tree in the configuration space of the robot.
Motion planning: resolution

- Most popular method:
  - Randomly Rapidly exploring Tree (RRT)
- expand a tree in the configuration space of the robot.
RRT algorithm

- RRT: develop a roadmap in configuration space
RRT algorithm

- RRT-extend
  - pick a random configuration $q_{rand}$
RRT algorithm

- RRT-extend
  - find closest node in current tree $q_{near}$
RRT algorithm

- RRT-extend
  - try to connect $q_{near}$ to $q_{rand}$, thus defining $q_{new}$
RRT algorithm

- RRT-connect
  - try to connect $q_{new}$ to goal configuration
RRT algorithm

- RRT-connect
  - try to connect $q_{new}$ to goal configuration
RRT algorithm

- RRT extend
RRT algorithm

- RRT extend

![Diagram showing RRT algorithm with initial configuration (q_init) and goal configuration (q_goal) connected by a line in the configuration space (C).]
RRT algorithm

- RRT extend

![Diagram of RRT algorithm](image)
RRT algorithm

- RRT connect
RRT algorithm

- RRT connect
RRT algorithm

- RRT extend
RRT extend

$C$

$q_{init}$

$q_{goal}$
RRT algorithm

- RRT extend
RRT algorithm

- RRT extend
RRT algorithm

- RRT extend
RRT algorithm

- RRT extend
RRT algorithm

- RRT extend
RRT algorithm

- RRT connect
Main characteristics of humanoid robots

- walk
  - contact with the ground defines submanifolds of the configuration space: $f(q) = 0$,
  - motion should be dynamically balanced.
- goal is a task (submanifold) rather than a configuration $g(q) = 0$. 
Main characteristics of humanoid robots

- walk
  - contact with the ground defines submanifolds of the configuration space: $f(q) = 0$,
  - motion should be dynamically balanced.
- goal is a task (submanifold) rather than a configuration $g(q) = 0$. 
Grasping an object

- Cast humanoid motion planning problem into classical RRT method:
  - contact and balance constraints,
  - goal task.
RRT-extend

- try to link two configurations
Constraints define a submanifold of $C$

- link two configurations remaining on a manifold defined by
  - fixed foot position,
  - fixed projection of COM
- using hierarchical constraint solver: hpp-gik.
RRT-extend

Constraints define a submanifold of $C$

- link two configurations remaining on a manifold defined by
  - fixed foot position,
  - fixed projection of COM
- using hierarchical constraint solver: hpp-gik.
RRT-extend

Constraints define a submanifold of $C$

- link two configurations remaining on a manifold defined by
  - fixed foot position,
  - fixed projection of COM

- using hierarchical constraint solver: hpp-gik.
RRT-extend

Constraints define a submanifold of $C$

- link two configurations remaining on a manifold defined by
  - fixed foot position,
  - fixed projection of COM
- using hierarchical constraint solver: hpp-gik.
Hierarchical constraint solver

- input:
  - initial configuration,
  - ordered list of tasks

- output:
  - motion from initial configuration to configuration satisfying as many tasks as possible.
Add a posture constraint as lowest priority task,
numerically integrate velocity computed by \texttt{hpp-gik}.

\begin{itemize}
\item $q_{\text{near}}$ \rightarrow new configurations
\item $q_{\text{rand}}$
\item $q_{\text{new}}$
\item Constrained manifold
\end{itemize}
Goal task vs goal configurations

- goal of motion is a task
- we randomly sample goal task to generate goal configurations
  - using \texttt{hpp-gik}: hierarchical nonlinear task solver.

- random trees can be expanded from each goal configuration.
Example

- HRP2 robot grasping an object in a cluttered environment.
Path optimization

- The result of the motion planning step needs to be optimized
- Two steps
  - final posture optimization,
  - path optimization.
Path optimization

- The result of the motion planning step needs to be optimized
- Two steps
  - final posture optimization,
  - path optimization.
Goal configuration optimization

- Random-connect optimization

  - blue: \texttt{hpp-gik} with half-sitting posture as lower priority task,
  - magenta: random sampling of configurations.
Path optimization

- Random constrained short-cut
Path optimization

- Random constrained short-cut
Walking for grasping

- Goal configuration of walk motion is randomly sampled
Manipulating doors, windows, drawers...

- 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 do not take into account forces here.
- We put this knowledge into the object thus defining the notion of
  - *documented object*.
- Humanoids 2010
Manipulating doors, windows, drawers...

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 do not take into account forces here.

We put this knowledge into the object thus defining the notion of documented object.

Humanoids 2010
Manipulating doors, windows, drawers...

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 do not take into account forces here.

We put this knowlegde into the object thus defining the notion of

- documented object.

Humanoids 2010
Manipulating doors, windows, drawers...

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 do not take into account forces here.

We put this knowledge into the object thus defining the notion of documented object.

Humanoids 2010
Manipulating doors, windows, drawers...

- 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 do not take into account forces here.

- We put this knowledge into the object thus defining the notion of
  - *documented object*.

- Humanoids 2010
Manipulating doors, windows, drawers...

- 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 do not take into account forces here.
- We put this knowledge into the object thus defining the notion of
  - documented object.
- Humanoids 2010
Manipulating doors, windows, drawers...

- 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 do not take into account forces here.
- We put this knowledge into the object thus defining the notion of
documentated object.
- Humanoids 2010
Manipulating doors, windows, drawers...

- 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 do not take into account forces here.
- We put this knowledge into the object thus defining the notion of
  - documented object.

Humanoids 2010
Manipulating doors, windows, drawers...

- 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 do not take into account forces here.

- We put this knowledge into the object thus defining the notion of documented object.

- Humanoids 2010
Manipulating doors, windows, drawers...

- 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 do not take into account forces here.
- We put this knowlegde into the object thus defining the notion of
  - documented object.

Humanoids 2010
Manipulating doors, windows, drawers...

- 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 do not take into account forces here.
- We put this knowledge into the object thus defining the notion of
  - *documented object*.
- Humanoids 2010
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.

Florent Lamiraux (LAAS)  Motion planning for every-day tasks  October 4, 2010  40 / 47
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.

![Diagram showing projection functions move the bounding box in order to put door handles into reaching area of robot hands.](image)
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

- Away from Door
- Hold Door with Left Hand
- Hold Door with Both Hands
- Hold Door with Right Hand

Motion planning for everyday tasks
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 \( \infty \) 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 \( \infty \) when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.
Going through a door: algorithm

- Configuration space:
  $SE(2) \times [\alpha_{\text{min}}, \alpha_{\text{max}}] \times \{\text{free, left hand, right hand, both hands}\}$

- 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 $\infty$ when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.
Going through a door: algorithm

- Configuration space:
  \( SE(2) \times [\alpha_{\text{min}}, \alpha_{\text{max}}] \times \{\text{free, left hand, right hand, both hands}\} \)

- 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 \( \infty \) when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.
Going through a door: algorithm

- Configuration space:
  \[ SE(2) \times [\alpha_{\text{min}}, \alpha_{\text{max}}] \times \{\text{free, left hand, right hand, both hands}\} \]
- 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 \( \infty \) 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 \(\infty\) 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 \( \infty \) 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 \(\infty\) when two configurations cannot be connected,
  - random configuration shooter: sample at intersections of constraint manifolds.
Going through a door: intermediate result
Going through a door: whole-body motion

- Motion of the bounding box is converted into time parameterized
  - step sequence,
  - motions of end effectors

- Resolution
  - step sequence $\rightarrow$ COM: preview control,
  - swing foot and upper body: hpp-gik.
Going through a door: whole-body motion

- Motion of the bounding box is converted into time parameterized
  - step sequence,
  - motions of end effectors

- Resolution
  - step sequence $\rightarrow$ COM: preview control,
  - swing foot and upper body: hpp-gik.
Perspectives

- Push further the notion of documented objects
  - clearly distinguish between data relative to
    - the object (door handle),
    - the robot (grasping area)
  - include the notion of consequence
    - passing through a door makes the robot move from one room to another one
- in order to make efficient and fast symbolic reasoning.
Push further the notion of documented objects
  - clearly distinguish between data relative to
    - the object (door handle),
    - the robot (grasping area)
  - include the notion of consequence
    - passing through a door makes the robot move from one room to another one
in order to make efficient and fast symbolic reasoning.
Push further the notion of documented objects

- clearly distinguish between data relative to
  - the object (door handle),
  - the robot (grasping area)
- include the notion of consequence
  - passing through a door makes the robot move from one room to another one

in order to make efficient and fast symbolic reasoning.
Perspectives

- Push further the notion of documented objects
  - clearly distinguish between data relative to
    - the object (door handle),
    - the robot (grasping area)
  - include the notion of consequence
    - passing through a door makes the robot move from one room to another one

- in order to make efficient and fast symbolic reasoning.
Perspectives

- Push further the notion of documented objects
  - clearly distinguish between data relative to
    - the object (door handle),
    - the robot (grasping area)
  - include the notion of consequence
    - passing through a door makes the robot move from one room to another one
- in order to make efficient and fast symbolic reasoning.