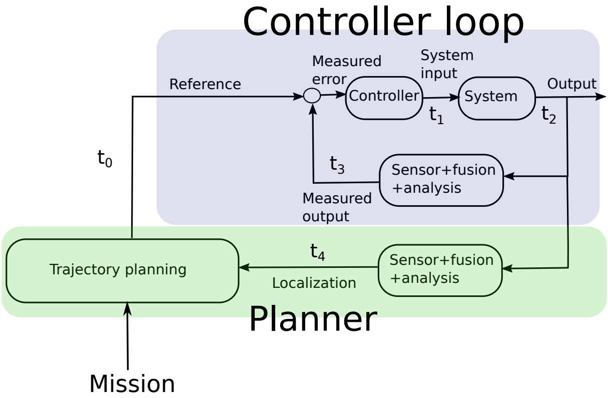
Revisting Box-RRT algorithm with DynIBEX

Alexandre Chapoutot

joint work with Julien Alexandre dit Sandretto and Olivier Mullier, François Pessaux
U2IS, ENSTA ParisTech, Palaiseau, France

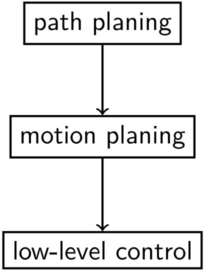
DGA MRIS meeting
November 24, 2016

Autonomous vehicle



Goal of the project try to understand main pieces of the system to validate their behavior and the behavior of the overall system.

A hierarchical control



- **Path planing** generates a set of way points (does not take into account the dynamics of the vehicle) from a map (totally or partially) known, take into account obstacles (static)
- **Motion planing** generates a set of trajectories feasible for the dynamics considered and take into account obstacles (static and dynamic)
- **Low-level controller** tries to follow the (discretized) trajectory w.r.t. the dynamic of the vehicle

This talk

We focus on **trajectory planing algorithms** taking into account

- A model of the vehicle
- A model of a map with obstacles (static)
- Bounded uncertain information on the position/orientation, etc.

This work is based on

Reliable robust path planning. Romain Pepy, Michel Kieffer, Eric Walter. Journal Applied Mathematical Computing. 2009.

More precisely, we implement the **BoxRRT algorithm** with **DynIBEX**.

DynIBEX in few words

A library combining of **CSP solver** (IBEX[1]) with **validated numerical integration methods** à la Runge-Kutta.

What can we simulate?

Our main tool for set-based simulation of dynamical systems

- Ordinary differential equations (ODE)
- Algebraic-differential equations (DAE) of index-1

$$S \equiv \begin{cases} \dot{\mathbf{y}} = F(t, \mathbf{y}, \mathbf{x}, \mathbf{p}, \mathbf{u}) \\ 0 = G(t, \mathbf{y}, \mathbf{x}, \mathbf{p}, \mathbf{u}) \\ 0 = H(t, \mathbf{y}, \mathbf{p}, \mathbf{u}) \\ \mathbf{y}(0) \in \mathcal{Y}_0, \mathbf{x}(0) \in \mathcal{X}_0, \mathbf{p} \in \mathcal{P}, \mathbf{u} \in \mathcal{U}, t \in [0, t_{\text{end}}] \end{cases} .$$

How checking temporal properties on S?

[1] Gilles Chabert (EMN) et al. `http://www.ibex-lib.org`

QCSDP

Let S be a differential system and $t_{\text{end}} \in \mathcal{R}_+$ the time limit. A QCSDP is a CSP defined by

- a set of variables $\mathcal{V}$ including *at least* t, a vector $\mathbf{y}_0$, $\mathbf{p}$, $\mathbf{u}$
  We represent these variables by the vector $\mathbf{v}$;
- an initial domain $\mathcal{D}$ containing *at least* $[0, t_{\text{end}}]$, $\mathcal{Y}_0$, $\mathcal{U}$, and $\mathcal{P}$;
- a set of constraints $\mathcal{C} = \{c_1, \ldots, c_e\}$ composed of predicates over sets, that is, constraints of the form

$$c_i \equiv Q\mathbf{v} \in \mathcal{D}_i . f_i(\mathbf{v}) \diamond \mathcal{A}, \qquad \forall 1 \leqslant i \leqslant e$$

  with $Q \in \{\exists, \forall\}$, $f_i : \wp(\mathcal{R}^{|\mathcal{V}|}) \to \wp(\mathcal{R}^q)$ stands for non-linear arithmetic expressions defined over variables $\mathbf{v}$ and solution of differential system S, $\mathbf{y}(t; \mathbf{y}_0, \mathbf{p}, \mathbf{u}) \equiv \mathbf{y}(\mathbf{v})$, $\diamond \in \{\subseteq, \cap_\emptyset\}$ and $\mathcal{A} \subseteq \mathcal{R}^q$ where $q > 0$.

Note: we follow the same approach that Goldsztejn et al.[2]

[2] Including ODE Based Constraints in the Standard CP Framework, CP10

Box-QCSDP as abstraction of QCSDP

Box-QCSDP

Let S be a differential system and $t_{\text{end}} \in \mathcal{R}_+$ the time limit A Box-QCSDP is defined by

- a set of variables $\mathcal{V}$ including *at least* t, a vector $\mathbf{y}_0$, $\mathbf{p}$, $\mathbf{u}$
  We represent these variables by the vector $\mathbf{v}$;
- an initial box $[\mathbf{d}]$ containing *at least* $[0, t_{\text{end}}]$, $[\mathbf{y}_0]$, $[\mathbf{u}]$, and $[\mathbf{p}]$;
- a set of interval constraints $\mathcal{C} = \{c_1, \ldots, c_e\}$ composed of predicates over sets, that is, constraints of the form

  $$c_i \equiv Q\mathbf{v} \in [\mathbf{d}_i].[f_i](\mathbf{v}) \diamond \alpha(\mathcal{A}), \qquad \forall 1 \leqslant i \leqslant e$$

  with $Q \in \{\exists, \forall\}$, $[f_i] : \mathcal{IR}^{|\mathcal{V}|} \to \mathcal{IR}^q$ stands for non-linear arithmetic expressions defined over variables $\mathbf{v}$ and interval enclosure solution $[\mathbf{y}](t; \mathbf{y}_0, \mathbf{p}, \mathbf{u}) \equiv [\mathbf{y}](\mathbf{v})$, $\diamond \in \{\subseteq, \cap_{\emptyset}\}$ and $\alpha \in \{\text{Hull}, \text{Int}\}$

Note: using boxes is not so straightforward to preserve soundness
TODO: A more formal definition of the abstraction has to be defined!

DynIBEX: a Box-QCSDP solver with restrictions

Solving arbitrary quantified constraints is hard!

We focus on particular problems of robotics involving quantifiers

- Robust controller synthesis: $\exists \mathbf{u}$, $\forall \mathbf{p}$, $\forall \mathbf{y}_0$ + temporal constraints
- Parameter synthesis: $\exists \mathbf{p}$, $\forall \mathbf{u}$, $\forall \mathbf{y}_0$ + temporal constraints
- etc.

We also defined a set of temporal constraints useful to analyze/design robotic application.

Verbal property	QCSDP translation
Stay in $\mathcal{A}$	$\forall t \in [0, t_{\text{end}}]$, $[\mathbf{y}](t, \mathbf{v}') \subseteq \text{Int}(\mathcal{A})$
In $\mathcal{A}$ at τ	$\exists t \in [0, t_{\text{end}}]$, $[\mathbf{y}](t, \mathbf{v}') \subseteq \text{Int}(\mathcal{A})$
Has crossed $\mathcal{A}$	$\exists t \in [0, t_{\text{end}}]$, $[\mathbf{y}](t, \mathbf{v}') \cap \text{Hull}(\mathcal{A}) \neq \emptyset$
Go out $\mathcal{A}$	$\exists t \in [0, t_{\text{end}}]$, $[\mathbf{y}](t, \mathbf{v}') \cap \text{Hull}(\mathcal{A}) = \emptyset$
Has reached $\mathcal{A}$	$[\mathbf{y}](t_{\text{end}}, \mathbf{v}') \cap \text{Hull}(\mathcal{A}) \neq \emptyset$
Finished in $\mathcal{A}$	$[\mathbf{y}](t_{\text{end}}, \mathbf{v}') \subseteq \text{Int}(\mathcal{A})$

Box-RRT Algorithm

Goal

- "quickly" find a trajectory going from an initial configuration $\mathbf{s}_i$ to a final configuration $\mathbf{s}_f$
- while avoiding obstacles $\mathbf{s}_o$
- and taking into account bounded uncertainties.

Main ingredients

- used a model of the vehicle
- based on RRT (Rapid Explore Tree) Algorithm
- combined with interval analysis tools (*e.g.*, guaranteed numerical integration)

Cinematic of a Robot in 2D

We consider an unicycle model of a robot

$$\begin{aligned}
\dot{x} &= v\cos(\theta) \\
\dot{y} &= v\sin(\theta) \\
\dot{\theta} &= v\tan\left(\delta + [-\varepsilon, \varepsilon]\right)
\end{aligned}$$

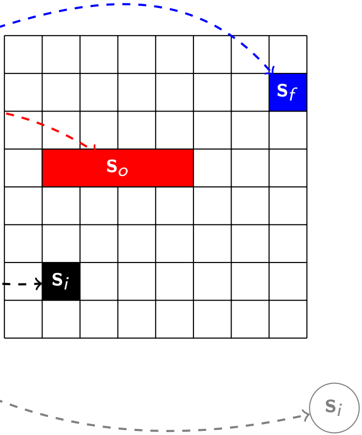
with constraints

- $v \in [-10, 10]$ and $\omega \in [-\pi/6, \pi/6]$
- $\varepsilon = 1e^{-3}$

How does Box-RRT work?

- A discretized map
- A final state
- Some obstacles
- An initial state
- A tree of possible trajectories

Initialization



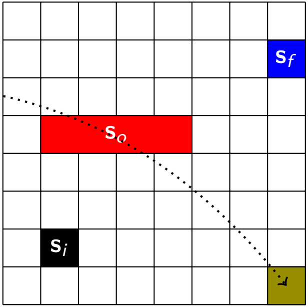
How does Box-RRT work?

Iteration 1

The building of the tree follows 3 main steps in a loop

1. Pick s_{random}
2. Find s_{nearest}
3. Compute s_{new}

until $\mathbf{s}_f$ is reached



Choose randomly a free state of the map

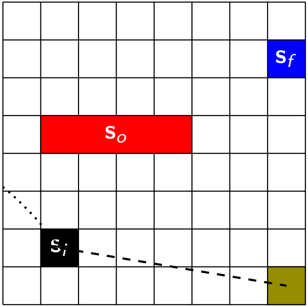
How does Box-RRT work?

Iteration 1

The building of the tree follows 3 main steps in a loop

1. Pick s_{random}
2. Find s_{nearest}
3. Compute s_{new}

until $\mathbf{s}_f$ is reached



Find the nearest node in the tree following a Hausdorff distance

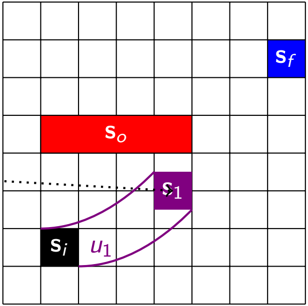
How does Box-RRT work?

Iteration 1

The building of the tree follows 3 main steps in a loop

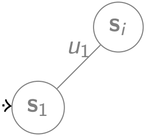
1. Pick s_{random}
2. Find s_{nearest}
3. Compute s_{new}

until $\mathbf{s}_f$ is reached



Predict the next state from a random control u

if no collision detected add $(\mathbf{s}_{\text{nearest}}, u, \mathbf{s}_{\text{new}})$ in the tree



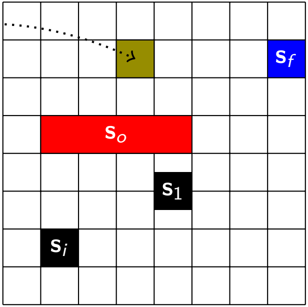
How does Box-RRT work?

Iteration 2

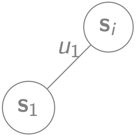
The building of the tree follows 3 main steps in a loop

1. Pick s_{random}
2. Find s_{nearest}
3. Compute s_{new}

until $\mathbf{s}_f$ is reached



Note that the choice of $\mathbf{s}_{\text{random}}$ can be biased to increase probability to be closer to $\mathbf{s}_f$



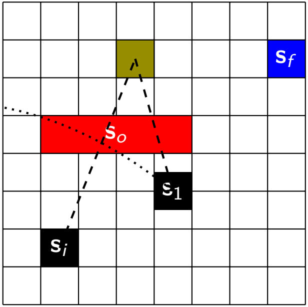
How does Box-RRT work?

Iteration 2

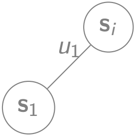
The building of the tree follows 3 main steps in a loop

1. Pick s_{random}
2. Find s_{nearest}
3. Compute s_{new}

until $\mathbf{s}_f$ is reached



Computing distances has a linear complexity w.r.t. the number of nodes in the tree



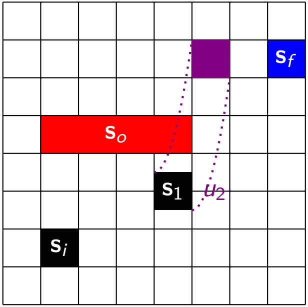
How does Box-RRT work?

Iteration 2

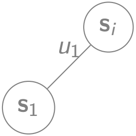
The building of the tree follows 3 main steps in a loop

1. Pick s_{random}
2. Find s_{nearest}
3. Compute s_{new}

until $\mathbf{s}_f$ is reached



The absence of collision is detected when the **tube** does not intersect an obstacle



Box-RRT in DynIBEX

Three temporal constraints have been used:

- `stay_in` state-space;
- `has_crossed` (in negative form) obstacles;
- `one_in` target;

and one contractor on tube has been used

- `get_tight(t)` to get final state.

So this algorithm can be quickly implemented in DynIBEX.

Conclusion

- Defined a framework to analyze robotic application: Box-QCSDP
- Presented a small example of autonomous vehicle
- Shown one algorithm in the control hierarchy: Box-RRT

Future work

- Define properties we wan/can prove: Viability Kernel, etc.
- Make Box-RRT deterministic ?
- Combine Box-RRT with low-level controller (PID)

Under development

- DynIBEX and contractor and predicate on tubes (J. Alexandre dit Sandretto)
- Extension to n-dimensional case of viability computation (O. Mullier)
- Model this system in an appropriate language, *e.g.*, Zelus (F. Pessaux)
- Combining OpenSMT2 and DynIBEX $\Rightarrow$ SMT modulo ODE (R. Morier)