Optimal Control and Optimization in Robotics

Mengda Li

August 20, 2019

ENS Paris-Saclay

Introduction

My internship is co-supervised by Justin Carpentier (Willow, Inria Paris) and Nicolas Mansard (Gepetto, LAAS/CNRS) in the Willow research group at INRIA Paris in France.

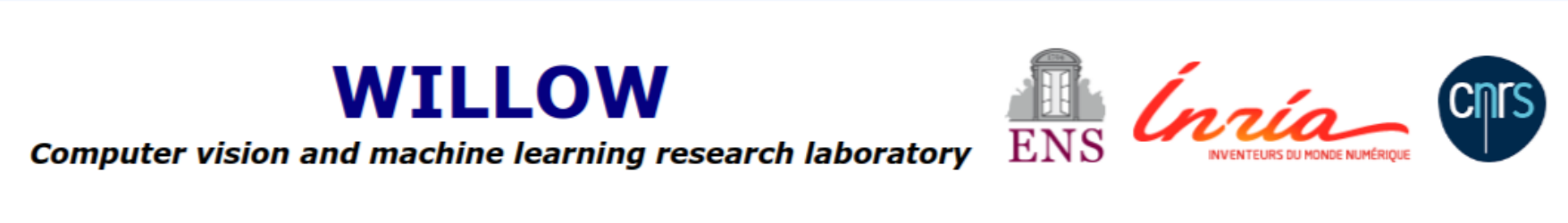
Figure 1: Justin Carpentier

Figure 2: Nicolas Mansard

Hosting institution

L'**Institut national de recherche en informatique et en automatique (Inria)** est un établissement public à caractère scientifique et technologique français spécialisé en mathématiques et informatique, placé sous la double tutelle du ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation et du ministère de l'Économie et des Finances créé le 3 janvier 1967 dans le cadre du « plan calcul ».

Research teams



WILLOW is based in the Laboratoire d'Informatique de l'École Normale Superiéure (CNRS/ENS/INRIA UMR 8548) and is a joint research team between INRIA Rocquencourt, École Normale Supérieure de Paris and Centre National de la Recherche Scientifique.

Their research is concerned with representational issues in visual object recognition and scene understanding.

Motivations and Problems in a general context

I want my robot to move one thing somewhere and pass some point at some moment with a lowest cost.

Optimization: lowest cost

Control: from some point to another point

and *Constraints*

Outline

Optimal Control Problem

Optimal Control Problem

Goal 1: Controllability

$$\begin{aligned} &\text{find} && u \\ &\text{subject to} && x(0) = x_0,\ x(T) = p, \\ &&& \dot{x}(t) = f(x(t), u(t)). \end{aligned} \tag{1}$$

Goal 2: Optimal Control

$$\begin{aligned} &\underset{u}{\text{minimize}} && \int_0^T l(x(t), u(t))dt \\ &\text{subject to} && x(0) = x_0,\ x(T) = p, \\ &&& \dot{x}(t) = f(x(t), u(t)). \end{aligned} \tag{2}$$

Transformation of the problem

Adding penalty to the terminal lost:

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \int_{[0,T[} l(x(t),u(t))dt + l_T(x(T)) \\ \text{subject to} \quad & x(0) = x_0, \\ & \dot{x}(t) = f(x(t),u(t)). \end{aligned} \tag{3}$$

Discretization of functions and variables:

$$\begin{aligned} \underset{x\in\ell^\infty_{N+1}, u\in\ell^\infty_N}{\text{minimize}} \quad & J(x,u) = \sum_{i=0}^{N-1} L(x_i,u_i) + L_T(x_N) \\ \text{subject to} \quad & x(0) = x_0, \\ & x_{i+1} = F(x_i,u_i)\ \forall i \in [0..N-1] \end{aligned} \tag{4}$$

Differential Dynamic Programming

Dynamic Programming

Optimize one by one:

$$\min_{U} J(U) = \min_{u_0} \min_{u_1} \dots \min_{u_{N-1}} J(U) \tag{5}$$

Definitions of Value Function and Q-functions:

$$\begin{aligned} V_i(x_i) &= \min_{u_i} L(x_i, u_i) + V_{i+1}(x_{i+1}) \\ V_N(x_N) &= L_T(x_N) \end{aligned} \tag{6}$$

$$\begin{aligned} Q_i(x_i, u_i) &= L(x_i, u_i) + V_{i+1}(x_{i+1}) \\ &= L(x_i, u_i) + V_{i+1}(f(x_i, u_i)) \\ &= L(x_i, u_i) + \min_{u_{i+1}} Q_{i+1}(f(x_i, u_i), u_{i+1}), \ \ i \leq N-2 \end{aligned} \tag{7}$$

$$V_i(x_i) = \min_{u_i} Q_i(x_i, u_i) \tag{8}$$

Linear Quadratic Regulator (LQR)

The LQR is an algorithm that solves the problem 4 *in one iteration* in case L, L_T are **quadratic** and F is **linear** (or quadratic).

Two phases in the algorithm:

- **Backward and Forward pass**:
  Compute V_x, V_{xx} and Q_u, Q_{uu}, Q_{ux} alternately
- **Roll-out**:
  Compute δ_u^* by V_x, V_{xx} and Q_u, Q_{uu}, Q_{ux}

Backward and Forward pass

Backward pass: from the partial derivatives of V_{i+1} back to the partial derivatives of Q_i

$$\begin{aligned} Q_u &= L_u + \underline{V'_x}\, F_u \quad {}^{1} \\ Q_{uu} &= L_{uu} + \underline{V'_x} \cdot F_{uu} + F_u^T\, \underline{V'_{xx}} F_u \\ Q_{ux} &= L_{ux} + \underline{V'_x} \cdot F_{ux} + F_u^T\, \underline{V'_{xx}} F_x \end{aligned} \tag{9}$$

Forward pass: from the partial derivatives of Q_i back to the partial derivatives of V_i

$$\begin{aligned} V_x &= Q_x - Q_u Q_{uu}^{-1} Q_{ux} &&= Q_x + Q_u K \\ V_{xx} &= Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{ux} &&= Q_{xx} + Q_{xu} K \end{aligned} \tag{10}$$

[1] "Q, V'" mean "Q_i, V_{i+1}".

Roll-out

Roll-**out**: determine the new trajectory x, u by computing the best control change δ_u.

$$\begin{aligned} \delta_u^*(i) &:= \arg\min_{\delta_u(i)} Q_i(x_i + \delta_x(i), u_i + \delta_u(i)) \\ \delta_u^* &= -Q_{uu}^{-1}(Q_u + Q_{ux}\delta_x) \quad \forall i \in [0..N-1] \end{aligned} \tag{11}$$

$$\begin{aligned} u^* &:= u + \delta_u^*, \\ x_0^* &= x_0, \\ x_{i+1}^* &= F(x_i^*, u_i^*) \quad \forall i \in [0..N-1] \end{aligned} \tag{12}$$

Sequential Quadratic Programming (SQP)

What if the F, L are not quadratic? Use the Sequential Quadratic Programming to solve general non-linear problems.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c(x) = 0 \end{aligned} \tag{13}$$

$$A(x)^T = [\nabla c_1(x), \nabla c_2(x), ..., \nabla c_m(x)] \tag{14}$$

Goal: KKT necessary conditions

$$F(x, \lambda) = \begin{bmatrix} \nabla f(x) - A(x)^T \lambda \\ c(x) \end{bmatrix} = 0 \tag{15}$$

Resolve the quadraticalized problems sequentially to find an optimal step and update the trajectory with a roll-out policy (like line search). Partie technique, voir la chapitre 18 de [1] and [2]

Augmented Lagrangian method

Penalty method

The *Penalty method* transform a *constrained optimization problem*:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c_i(x) = 0, \quad i \in \mathcal{E} \end{aligned} \tag{16}$$

to an *optimization problem without constraint*:

$$\underset{x}{\text{minimize}} \quad f(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x), \quad \mu > 0 \tag{17}$$

μ is the penalty parameter. We solve sequentially the 17 with an increasing sequence[2] of μ_k until the final convergence test [3]is satisfied.

[2]For example, $\mu_{k+1} = 2\mu_k$ if some residual tolerance is satisfied.

[3]For example, the residual $\|c(x)\| < 10^{-8}$

Augmented Lagrangian

Augmented Lagrangian function is modified version of the *penalty function*:

$$\mathcal{L}^A(x) := f(x) + \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x) \tag{18}$$

The λ is the dual variable called *Lagrangian multiplier* who has the same dimension as the constraints.

While solving sequentially the 18, at each iteration,

- if the tolerance is satisfied, update $\lambda^{k+1} := \lambda^k + \mu^k c(x_k)$
- else, increase the penalty μ
- Stopping criteria: KKT necessary optimality conditions

If the algorithm finally finds a numerical optimum, the final λ would also be the dual optimum numerically.

Central problem in the internship

$$
\begin{aligned}
\underset{x \in \mathbb{X}^{T+1}, u \in \mathbb{U}^T}{\text{minimize}} \quad & J(x,u) = \sum_{i=0}^{T-1} L_i(x_i, u_i) + L_T(x_T) \\
\text{subject to} \quad & x(0) = x_0, \\
& x_{i+1} = F(x_i, u_i) && \forall i \in [0..T-1] \\
& c_j(x,u) = 0 && \forall j \in [0..M-1]
\end{aligned}
\tag{19}
$$

Let $c : \mathbb{X}^{T+1} \times \mathbb{U}^T \to \mathbb{R}^M$ [4]be the function for the set of constraints. Suppose the constraints are *independent*.

[4]Let $\mathbb{X}$ be the state space, for example $\mathbb{R}^n$ and $\mathbb{U}$ be the control space, for example $\mathbb{R}^m$. We can use $\mathcal{X} := \mathbb{X}^{T+1}, \mathcal{U} := \mathbb{X}^T$ to denote the state and control trajectory space.

Augmented Lagrangian on DDP

$$\mathcal{L}_i^A(x_i, u_i, \lambda, \mu) := L_i(x_i, u_i) + \sum_{j \sim i} \lambda_j c_j(x_i, u_i) + \frac{\mu}{2} \sum_{j \sim i} c_j^2(x_i, u_i) \tag{20}$$

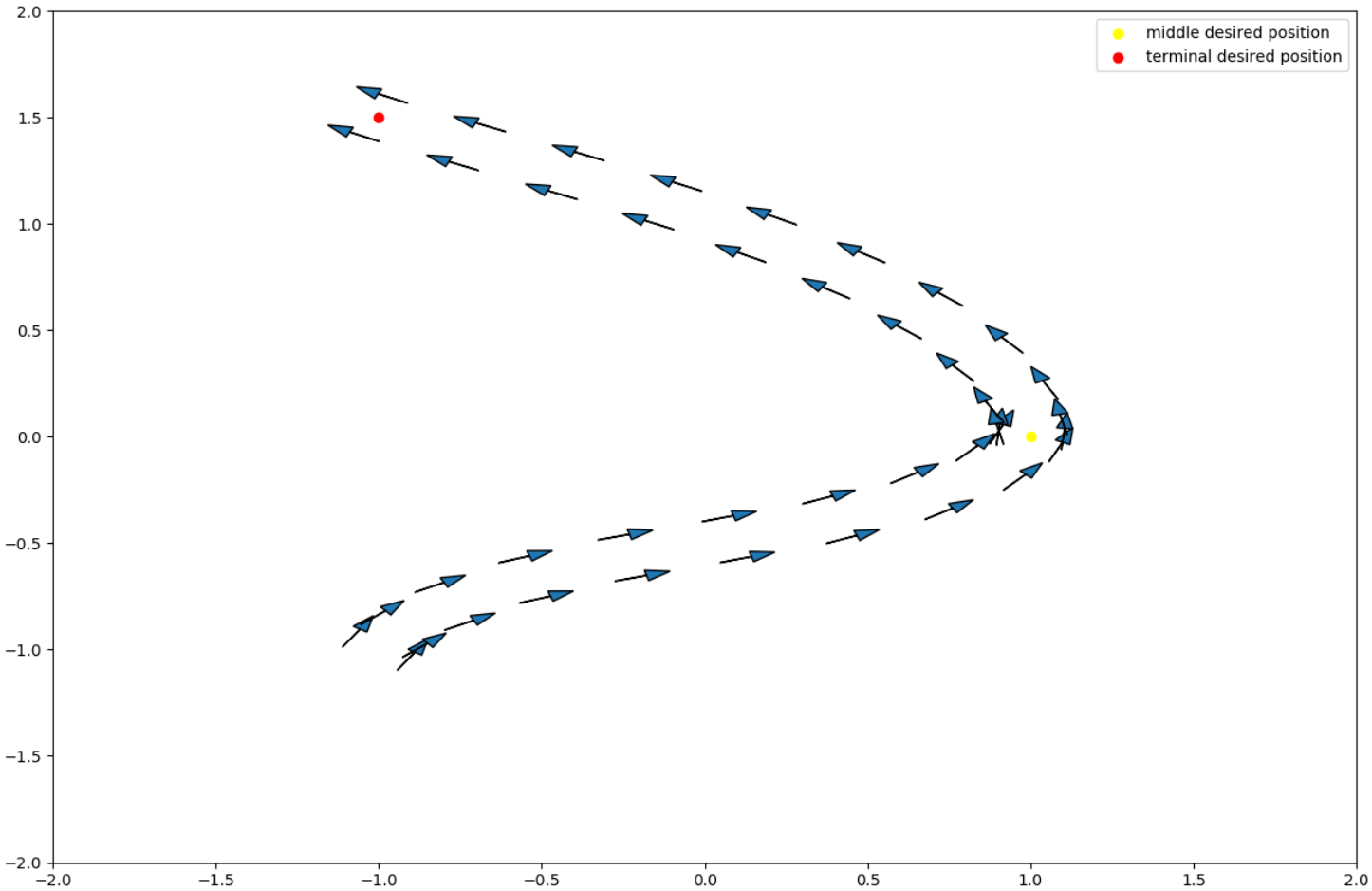
(Cette partie est ma contribution)

$$\begin{aligned} \underset{x \in \mathbb{X}^{T+1}, u \in \mathbb{U}^T}{\text{minimize}} \quad & J(x, u) = \sum_{i=0}^{T-1} \mathcal{L}_i^A(x_i, u_i, \lambda, \mu) + \mathcal{L}_T^A(x_T, \lambda, \mu) \\ \text{subject to} \quad & x(0) = x_0, \\ & x_{i+1} = F(x_i, u_i) \end{aligned} \tag{21}$$

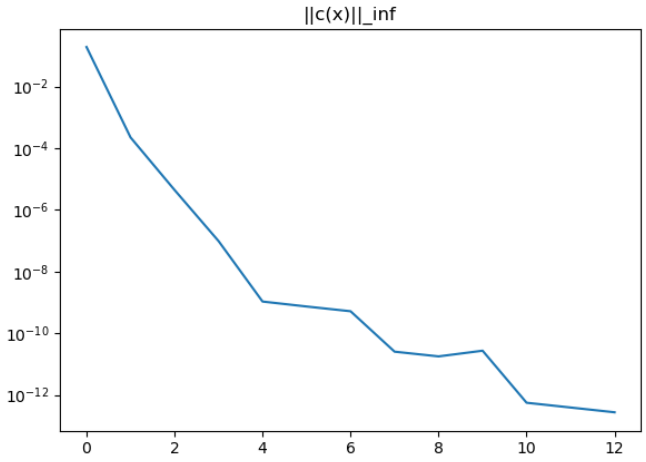
For all $\lambda \in \mathbb{R}^M, \mu \in \mathbb{R}^{+*}$, the problem 21 can be solved by a DDP (or FDDP) solver ([2])

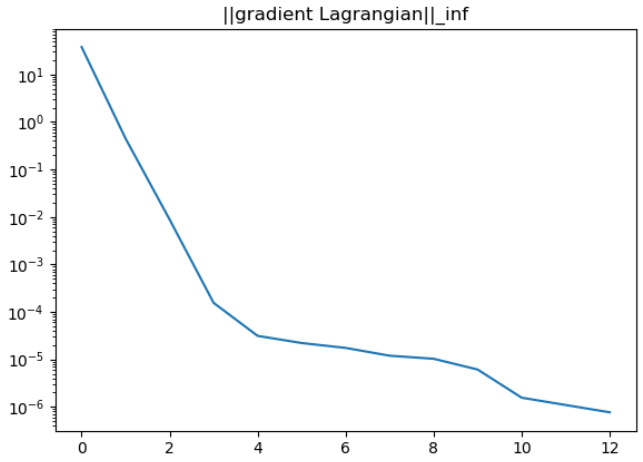
Example

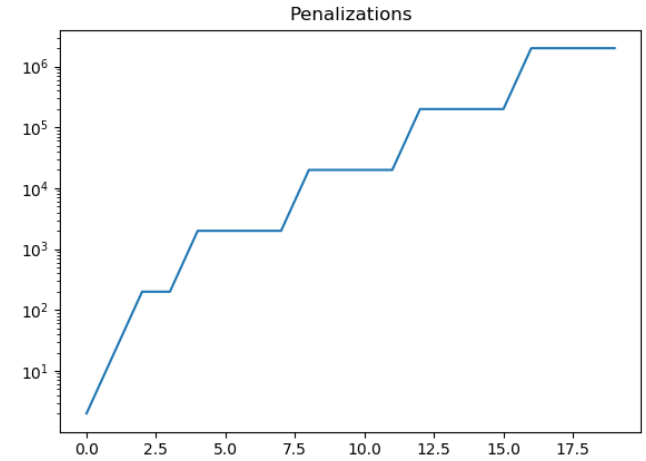
Example: Unicycle

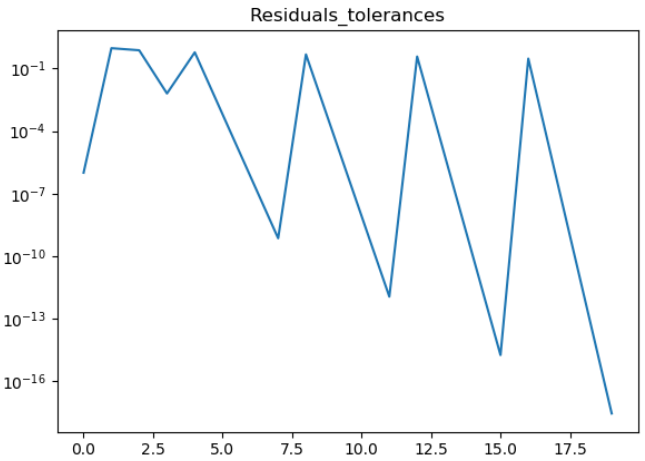


Graph of convergence









Difficulty

- The theory (of optimization) could be simple
- The implementation (and debug) could be complex
- ☐ No strong theorem for the convergence of algorithm(s) [5]

[5] There are indeed many theorems for optimization algorithms, but they are almost always in a weak form or not usable in practice. The special properties they state could be interesting in theory, but they will not directly give convergence guarantee.

Bibliography

Jorge Nocedal and Stephen J. Wright.
***Numerical Optimization*.**
Springer, 2006.

Carlos Mastalli, Rohan Budhiraja, Nicolas Mansard, and Justin Carpentier.
***Contact RObot COntrol by Differential DYnamic programming Library*.**