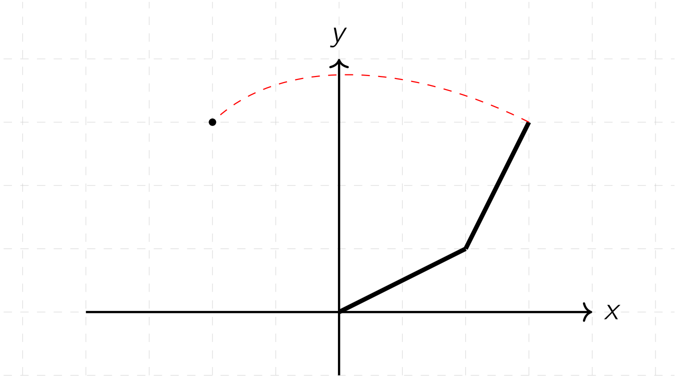
Efficient Trajectory Reshaping in a Dynamic Environment

Martin Biel, Mikael Norrlöf

KTH - Royal Institute of Technology, Linköping University, ABB

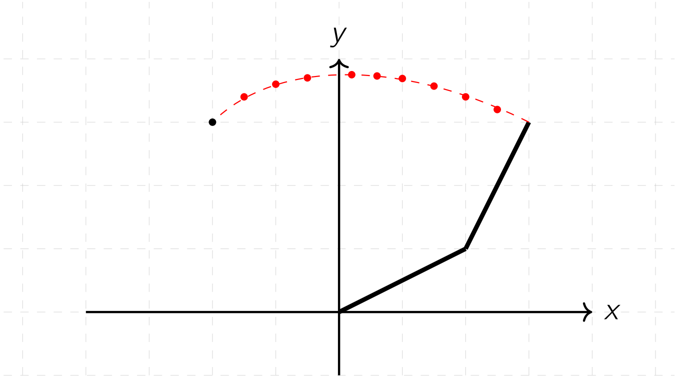
MARCH 10, 2018

Motivation: Standard Approach



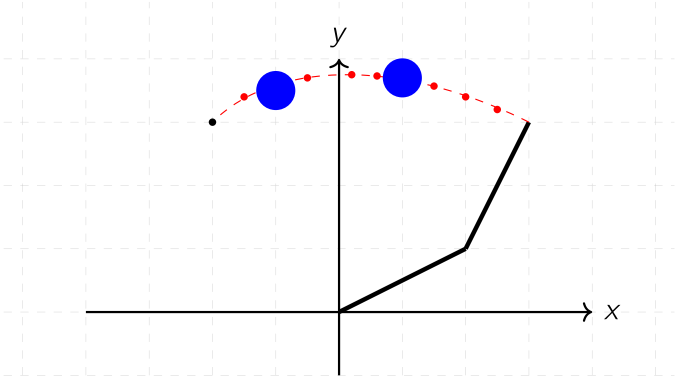
Geometric path planning

Motivation: Standard Approach



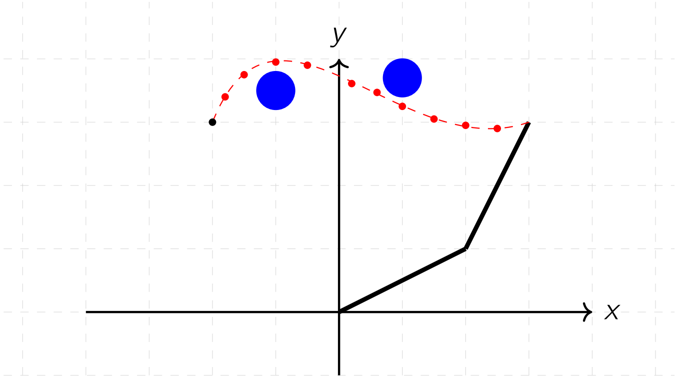
Optimal trajectory along path

Motivation: Standard Approach



Not viable in a dynamic environment

Motivation: New Approach



A combined approach is required!

Contribution

- C++ Framework for efficient trajectory reshaping

Contribution

- C++ Framework for efficient trajectory reshaping
- Heuristic strategy for solving optimal control problems in real-time

Contribution

- C++ Framework for efficient trajectory reshaping
- Heuristic strategy for solving optimal control problems in real-time
- Simulation environment. Evaulation on SCARA type robot

Content

- Overview of reshaping procedure

Content

- Overview of reshaping procedure
- Extensions

Content

- Overview of reshaping procedure
- Extensions
- Demo

Content

- Overview of reshaping procedure
- Extensions
- Demo
- Final Remarks

Timed Elastic Band (TEB)

$$
\underset{\boldsymbol{u}(.)}{\text{minimize}}\, t_f \quad \text{s.t.} \quad
\begin{cases}
\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\
\boldsymbol{y}(t) = g(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\
\boldsymbol{x}(t) \in \mathcal{X}_t \\
\boldsymbol{u}(t) \in \mathcal{U}_t \\
\boldsymbol{y}(t) \in \mathcal{Y}_t \\
\phi_0(\boldsymbol{x}(t_0), \boldsymbol{u}(t_0), \boldsymbol{y}(t_0), t_0) = 0 \\
\phi_f(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), \boldsymbol{y}(t_f), t_f) = 0
\end{cases}
$$

Timed Elastic Band (TEB)

- Discretized states and inputs are collected into a *TEB* set:

$$\mathcal{B} := \{\boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \dots, \boldsymbol{x}_{n-1}, \boldsymbol{u}_{n-1}, \boldsymbol{x}_n, \Delta T\}$$

Timed Elastic Band (TEB)

- Discretized states and inputs are collected into a *TEB* set:

$$\mathcal{B} := \{\boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \ldots, \boldsymbol{x}_{n-1}, \boldsymbol{u}_{n-1}, \boldsymbol{x}_n, \Delta T\}$$

- Approximated system dynamics:

$$\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta T} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$$

Timed Elastic Band (TEB)

- Discretized states and inputs are collected into a *TEB* set:

$$\mathcal{B} := \{\boldsymbol{x}_1, \boldsymbol{u}_1, \boldsymbol{x}_2, \boldsymbol{u}_2, \ldots, \boldsymbol{x}_{n-1}, \boldsymbol{u}_{n-1}, \boldsymbol{x}_n, \Delta T\}$$

- Approximated system dynamics:

$$\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta T} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$$

- The inclusion of the time increment allows the trajectory to be reshaped in both space and time

Problem reformulation in TEB space

$$
\begin{aligned}
\underset{\mathcal{B}}{\text{minimize}} \quad & (n-1)\Delta T \\
\text{s.t.} \quad & \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta T} - f(\boldsymbol{x}_k, \boldsymbol{u}_k) = 0 \\
& \boldsymbol{x}_k \in \mathcal{X}_k \\
& \boldsymbol{u}_k \in \mathcal{U}_k \\
& g(\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{Y}_k \\
& \phi_s(\boldsymbol{x}_1, \boldsymbol{u}_1, g(\boldsymbol{x}_1, \boldsymbol{u}_1)) = 0 \\
& \phi_f(\boldsymbol{x}_n, \boldsymbol{u}_n, g(\boldsymbol{x}_n, \boldsymbol{u}_n)) = 0 \\
& \Delta T > 0, \;\; k \in [1, n-1]
\end{aligned}
$$

Switching Strategy

- No convergence guarantees in TEB formulation

Switching Strategy

- No convergence guarantees in TEB formulation
- Switch to standard NMPC when target is *close enough*

$$
\begin{aligned}
\underset{\mathcal{B}\setminus\{\Delta T\}}{\text{minimize}} \quad & \sum_{k=1}^{n-1} \|\boldsymbol{x}_k - \boldsymbol{x}_f\|^2 \\
\text{s.t.} \quad & \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta \bar{T}} - f(\boldsymbol{x}_k, \boldsymbol{u}_k) = 0 \\
& \boldsymbol{x}_k \in \mathcal{X}_k \\
& \boldsymbol{u}_k \in \mathcal{U}_k \\
& g(\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{Y}_k \\
& \phi_s(\boldsymbol{x}_1, \boldsymbol{u}_1, g(\boldsymbol{x}_1, \boldsymbol{u}_1)) = 0 \\
& \phi_f(\boldsymbol{x}_n, \boldsymbol{u}_n, g(\boldsymbol{x}_n, \boldsymbol{u}_n)) = 0
\end{aligned}
$$

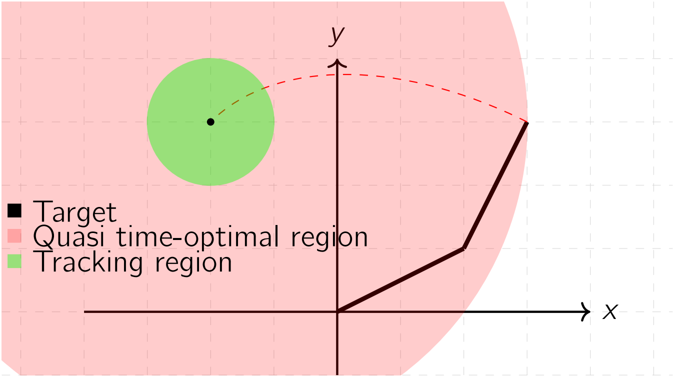
Switching Strategy

- No convergence guarantees in TEB formulation
- Switch to standard NMPC when target is *close enough*

$$
\begin{aligned}
\underset{\mathcal{B}\setminus\{\Delta T\}}{\text{minimize}} \quad & \sum_{k=1}^{n-1} \|\boldsymbol{x}_k - \boldsymbol{x}_f\|^2 \\
\text{s.t.} \quad & \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta \bar{T}} - f(\boldsymbol{x}_k, \boldsymbol{u}_k) = 0 \\
& \boldsymbol{x}_k \in \mathcal{X}_k \\
& \boldsymbol{u}_k \in \mathcal{U}_k \\
& g(\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{Y}_k \\
& \phi_s(\boldsymbol{x}_1, \boldsymbol{u}_1, g(\boldsymbol{x}_1, \boldsymbol{u}_1)) = 0 \\
& \phi_f(\boldsymbol{x}_n, \boldsymbol{u}_n, g(\boldsymbol{x}_n, \boldsymbol{u}_n)) = 0
\end{aligned}
$$

- Gives convergence under certain assumptions

Switching Strategy



Trajectory Reshaping Procedure

input: $\mathcal{B}$ - Current trajectory as TEB set
$\mathcal{O}$ - Environment information

Trajectory Reshaping Procedure

input: $\mathcal{B}$ - Current trajectory as TEB set
$\mathcal{O}$ - Environment information
output: $\mathcal{B}^*$ - Reshaped trajectory

Trajectory Reshaping Procedure

input: $\mathcal{B}$ - Current trajectory as TEB set
$\mathcal{O}$ - Environment information
output: $\mathcal{B}^*$ - Reshaped trajectory

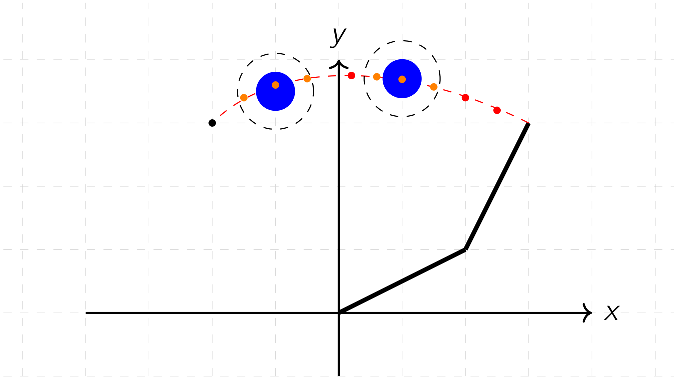
procedure ReshapeTrajectory
    $\tilde{\mathcal{B}} \leftarrow$ DeformInTime($\mathcal{B}$)
    $\mathcal{P} \leftarrow$ FormulateOptimizationProblem($\tilde{\mathcal{B}}$,$\mathcal{O}$)
    $\mathcal{B}^* \leftarrow$ DeformInSpace($\mathcal{P}$,$\tilde{\mathcal{B}}$)
    return $\mathcal{B}^*$
end procedure

Extensions

- Track moving targets
- Obstacle avoidance
- Multiple trajectories

Extension: Obstacle avoidance

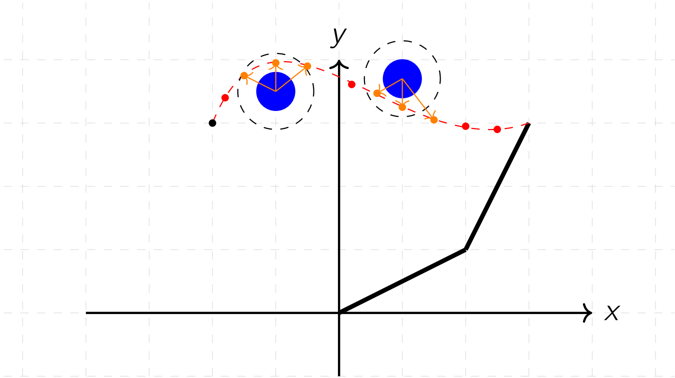
$$K_{j,\bar{\sigma}_{op}} := \left\{ k : \left\| g(\boldsymbol{x}_k, \boldsymbol{u}_k) - \mathcal{O}_j \right\|^2 \leq (\bar{\sigma}_{op} + r_j)^2 \right\}, \quad j = 1, \dots, m$$



Extension: Obstacle avoidance

$$
\begin{aligned}
\underset{\mathcal{B}}{\text{minimize}} \quad & (n-1)\Delta T - \sum_{j=1}^{m} \sum_{k \in K_{j,\bar{\sigma}_{op}}} \left\| g(\boldsymbol{x}_k, \boldsymbol{u}_k) - \mathcal{O}_j \right\|^2 \\
\text{s.t.} \quad & \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta T} - f(\boldsymbol{x}_k, \boldsymbol{u}_k) = 0 \\
& \boldsymbol{x}_k \in \mathcal{X}_k \\
& \boldsymbol{u}_k \in \mathcal{U}_k \\
& g(\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{Y}_k \\
& \phi_s(\boldsymbol{x}_1, \boldsymbol{u}_1, g(\boldsymbol{x}_1, \boldsymbol{u}_1)) = 0 \\
& \phi_f(\boldsymbol{x}_n, \boldsymbol{u}_n, g(\boldsymbol{x}_n, \boldsymbol{u}_n)) = 0 \\
& \Delta T > 0, \; k \in [1, n-1]
\end{aligned}
$$

Extension: Obstacle avoidance



Extension: Multiple Trajectories

- Non-convex problem in general

$\Rightarrow$ The reshaper might get stuck in local optima

Extension: Multiple Trajectories

- Non-convex problem in general

$\Rightarrow$ The reshaper might get stuck in local optima

Solution: *Reshape multiple trajectory candidates*

Extension: Multiple Trajectories

- Non-convex problem in general

$\Rightarrow$ The reshaper might get stuck in local optima

Solution: *Reshape multiple trajectory candidates*

- The trajectories are independent and can be planned in parallel

Extension: Multiple Trajectories

- Non-convex problem in general

$\Rightarrow$ The reshaper might get stuck in local optima

Solution: *Reshape multiple trajectory candidates*

- The trajectories are independent and can be planned in parallel
- Choose trajectory according to some performance critera
  (e.g. transition time + collisions)

Extension: Multiple Trajectories

- Non-convex problem in general

$\Rightarrow$ The reshaper might get stuck in local optima

Solution: *Reshape multiple trajectory candidates*

- The trajectories are independent and can be planned in parallel
- Choose trajectory according to some performance critera (e.g. transition time + collisions)
- Eventually commit to traversed trajectory and drop others

Implementation

- Framework for trajectory reshaping implemented in C++

Implementation

- Framework for trajectory reshaping implemented in C++
- Derivative information is computed using automatic differentiation

$\Rightarrow$ Supports arbitrary system dynamics

Implementation

- Framework for trajectory reshaping implemented in C++
- Derivative information is computed using automatic differentiation

⇒ Supports arbitrary system dynamics

- In-house SQP solver
  - *qpOASES* for QP subproblems
  - Exploits the sparsity in the emerging Hessians & Jacobians

Implementation

- Framework for trajectory reshaping implemented in C++
- Derivative information is computed using automatic differentiation

⇒ Supports arbitrary system dynamics

- In-house SQP solver
    - *qpOASES* for QP subproblems
    - Exploits the sparsity in the emerging Hessians & Jacobians
- Simulator implemented in the Julia programming language

Demo: SCARA Model

Final Remarks

Summary

- C++ Framework for efficient trajectory reshaping
- Heuristic strategy for solving optimal control problems in real-time
- Simulation environment

Final Remarks

Summary

- C++ Framework for efficient trajectory reshaping
- Heuristic strategy for solving optimal control problems in real-time
- Simulation environment

Ongoing work

- Proof of concept. Initial results are promising.
- Suitable for a pick-and-place scenario on a conveyor
- Next step is to employ the procedure in an embedded setting