Sensor-based trajectory optimization

ABB Robotics

Master thesis
Martin Biel

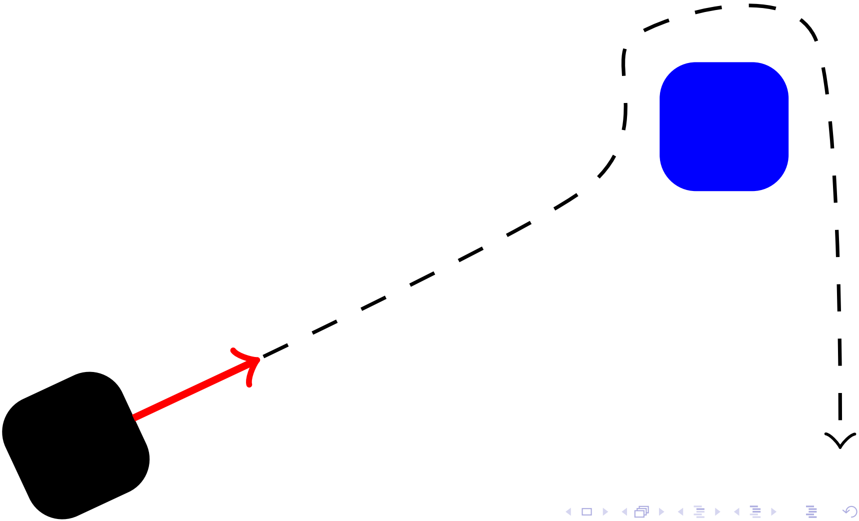
Supervisor: Mikael Norrlöf
Examiner: Xiaoming Hu

June 9, 2016

Outline

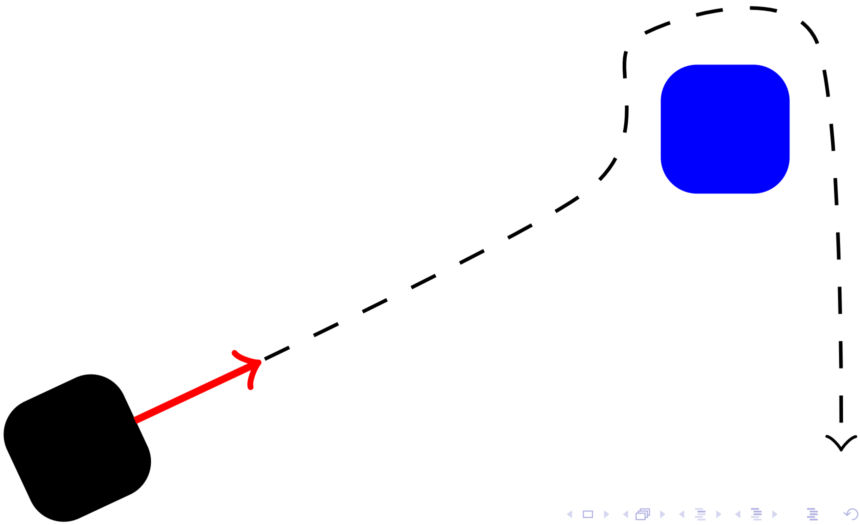
Traditional approach

- Geometric path computed on before hand.



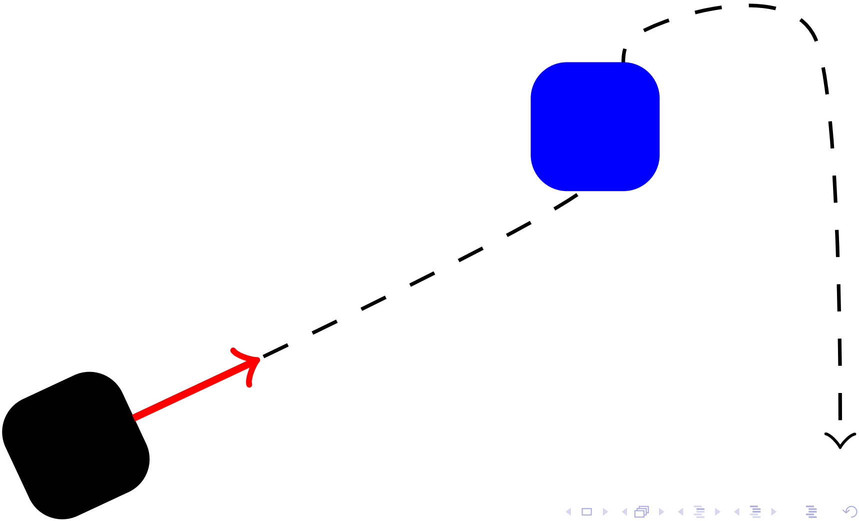
Traditional approach

- Geometric path computed on before hand.
- Optimal path following along the computed path.



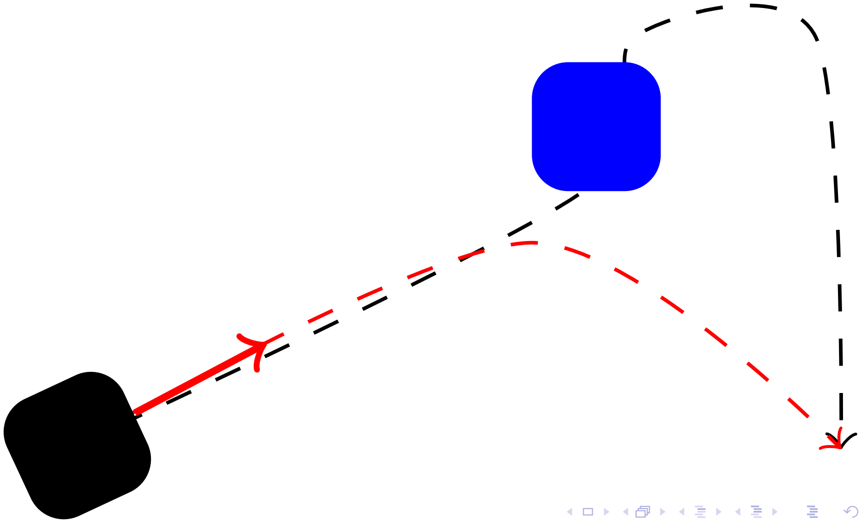
Traditional approach

- Geometric path computed on before hand.
- Optimal path following along the computed path.



Traditional approach

- Geometric path computed on before hand.
- Optimal path following along the computed path.



Problem formulation

General problem: Investigate the possibility of constructing a real-time capable trajectory planner, where:

- The underlying path should be allowed to change dynamically.

Problem formulation

General problem: Investigate the possibility of constructing a real-time capable trajectory planner, where:

- The underlying path should be allowed to change dynamically.
- The planner should be able to react to sensor events, and deform the trajectory accordingly.

Problem formulation

General problem: Investigate the possibility of constructing a real-time capable trajectory planner, where:

- The underlying path should be allowed to change dynamically.
- The planner should be able to react to sensor events, and deform the trajectory accordingly.
- The trajectory should be consistent with some given system dynamics.

Problem formulation

General problem: Investigate the possibility of constructing a real-time capable trajectory planner, where:

- The underlying path should be allowed to change dynamically.
- The planner should be able to react to sensor events, and deform the trajectory accordingly.
- The trajectory should be consistent with some given system dynamics.

Target application: Conveyor tracking

Problem formulation

General problem: Investigate the possibility of constructing a real-time capable trajectory planner, where:

- The underlying path should be allowed to change dynamically.
- The planner should be able to react to sensor events, and deform the trajectory accordingly.
- The trajectory should be consistent with some given system dynamics.

Target application: Conveyor tracking

- Pick and place.

Problem formulation

General problem: Investigate the possibility of constructing a real-time capable trajectory planner, where:

- The underlying path should be allowed to change dynamically.
- The planner should be able to react to sensor events, and deform the trajectory accordingly.
- The trajectory should be consistent with some given system dynamics.

Target application: Conveyor tracking

- Pick and place.
- Collision avoidance.

Problem formulation

General problem: Investigate the possibility of constructing a real-time capable trajectory planner, where:

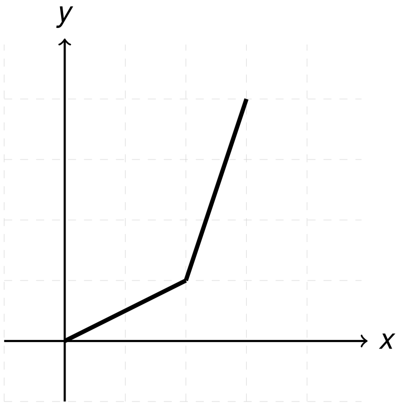
- The underlying path should be allowed to change dynamically.
- The planner should be able to react to sensor events, and deform the trajectory accordingly.
- The trajectory should be consistent with some given system dynamics.

Target application: Conveyor tracking

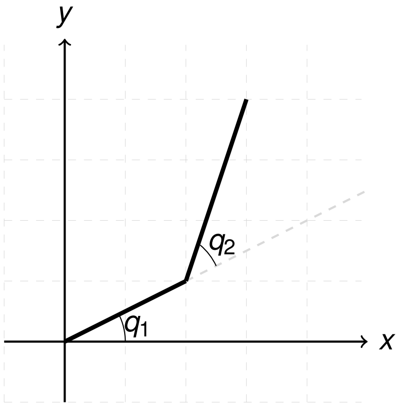
- Pick and place.
- Collision avoidance.
- Track moving targets.

Outline

Preliminaries - Robot modelling

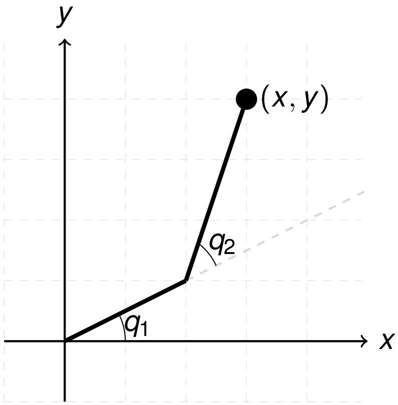


Preliminaries - Robot modelling



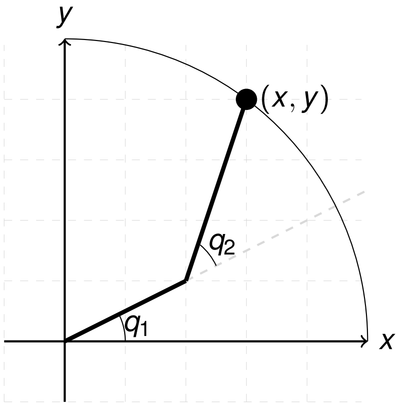
- $\mathcal{Q}$ - *Configuration space*

Preliminaries - Robot modelling



- $\mathcal{Q}$ - *Configuration space*
- $\mathcal{O}$ - *Operational space*

Preliminaries - Robot modelling



- $\mathcal{Q}$ - *Configuration space*
- $\mathcal{O}$ - *Operational space*
- $\mathcal{W}$ - *Workspace*

Preliminaries - Robot modelling

- *Forward kinematics*: $\boldsymbol{y} = \chi_y(\boldsymbol{q})$
- *Inverse kinematics*: $\boldsymbol{q} = \chi_y^{-1}(\boldsymbol{y})$
- *Velocity Jacobian*: $\boldsymbol{v} = J(\boldsymbol{q})\dot{\boldsymbol{q}}$
- *Dynamics*: $M(\boldsymbol{q}(t))\ddot{\boldsymbol{q}}(t) + C(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))\dot{\boldsymbol{q}}(t) + g(\boldsymbol{q}(t)) = \boldsymbol{\tau}(t)$

Preliminaries - Optimal control problem

Time minimizing formulation

$$\min_{\boldsymbol{\tau}(.)} T \quad \text{s.t.} \quad \begin{cases} M(\boldsymbol{q}(t))\ddot{\boldsymbol{q}}(t) + C(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))\dot{\boldsymbol{q}}(t) + g(\boldsymbol{q}(t)) = \boldsymbol{\tau}(t) \\ \boldsymbol{q}(t) \in \mathcal{Q} \\ \boldsymbol{\tau}_- \leq \boldsymbol{\tau}(t) \leq \boldsymbol{\tau}_+ \\ \boldsymbol{y}(t) = \chi_y(\boldsymbol{q}(t)) \\ \boldsymbol{y}(0) = \boldsymbol{y}_0, \dot{\boldsymbol{y}}(0) = \dot{\boldsymbol{y}}_0 \\ \boldsymbol{y}(T) = \boldsymbol{y}_T, \dot{\boldsymbol{y}}(T) = \dot{\boldsymbol{y}}_T \end{cases}$$

Preliminaries - Timed elastic band

- Introduce the state vector

$$\boldsymbol{x}(t) = \begin{pmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{pmatrix}$$

as a solution trajectory to the optimal control problem.

Preliminaries - Timed elastic band

- Introduce the state vector

$$\boldsymbol{x}(t) = \begin{pmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{pmatrix}$$

  as a solution trajectory to the optimal control problem.
- Discretize the trajectory into a so called *Timed Elastic Band* (TEB) set $\mathcal{B} := \{\boldsymbol{x}_1, \boldsymbol{\tau}_1, \boldsymbol{x}_2, \boldsymbol{\tau}_2, \dots, \boldsymbol{x}_{n-1}, \boldsymbol{\tau}_{n-1}, \boldsymbol{x}_n, \Delta T\}$. Note that n and ΔT are NOT fixed.

Preliminaries - Timed elastic band

- Introduce the state vector

$$\boldsymbol{x}(t) = \begin{pmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{pmatrix}$$

as a solution trajectory to the optimal control problem.

- Discretize the trajectory into a so called *Timed Elastic Band* (TEB) set $\mathcal{B} := \{\boldsymbol{x}_1, \boldsymbol{\tau}_1, \boldsymbol{x}_2, \boldsymbol{\tau}_2, \dots, \boldsymbol{x}_{n-1}, \boldsymbol{\tau}_{n-1}, \boldsymbol{x}_n, \Delta T\}$. Note that n and ΔT are NOT fixed.
- Determine the system dynamics for $\boldsymbol{x}(t)$ and approximate them using forward Euler,

$$\frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta T} = A\boldsymbol{x}_k + B(f(\boldsymbol{x}_k) + h(\boldsymbol{x}_k)\boldsymbol{\tau}_k)$$

Preliminaries - Timed elastic band

$$
\begin{aligned}
\min_{\mathcal{B}} \quad & (n-1)\Delta T \\
\text{s.t.} \quad & \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta T} - A\boldsymbol{x}_k + B(f(\boldsymbol{x}_k) + h(\boldsymbol{x}_k)\boldsymbol{\tau}_k) = 0 \quad (k = 1, 2, \dots, n-1) \\
& \boldsymbol{\tau}_- \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_+ \quad (k = 1, 2, \dots, n-1) \\
& \boldsymbol{x}_1 = \boldsymbol{x}_s, \;\; \boldsymbol{x}_n = \boldsymbol{x}_f, \;\; \Delta T > 0
\end{aligned}
$$

$$
\left(\boldsymbol{x}_s = \begin{pmatrix} \boldsymbol{q}_s \\ \dot{\boldsymbol{q}}_s \end{pmatrix}, \quad \boldsymbol{x}_f = \begin{pmatrix} \chi_y^{-1}(\boldsymbol{y}_T) \\ \boldsymbol{0} \end{pmatrix} \right)
$$

Preliminaries - Timed elastic band

$$\min_{\mathcal{B}} \quad (n-1)\Delta T$$

$$\text{s.t.} \quad \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta T} - A\boldsymbol{x}_k + B(f(\boldsymbol{x}_k) + h(\boldsymbol{x}_k)\boldsymbol{\tau}_k) = 0 \quad (k = 1, 2, \ldots, n-1)$$

$$\boldsymbol{\tau}_- \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_+ \quad (k = 1, 2, \ldots, n-1)$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_s, \;\; \boldsymbol{x}_n = \boldsymbol{x}_f, \;\; \Delta T > 0$$

$$\left(\boldsymbol{x}_s = \begin{pmatrix} \boldsymbol{q}_s \\ \dot{\boldsymbol{q}}_s \end{pmatrix}, \quad \boldsymbol{x}_f = \begin{pmatrix} \chi_y^{-1}(\boldsymbol{y}_T) \\ \boldsymbol{0} \end{pmatrix} \right)$$

The optimization problem is solved on-line using non-linear model predictive control techniques, in the timed elastic band framework.

Outline

Trajectory Planner - Deformation

Deformation in time

Deformation in space

Trajectory Planner - Deformation

Deformation in time

During each control cycle, the following TEB update is performed $\bar{I}_{TEB}$ times

$$\text{TEB update i}-\begin{cases} \text{Insert a new state if } \Delta T_i > \Delta \bar{T}_{ref} + \Delta \bar{T}_{hyst} \wedge n_i < \bar{n}_{max} \\ \text{Remove a state if } \Delta T_i < \Delta \bar{T}_{ref} - \Delta \bar{T}_{hyst} \wedge n_i > \bar{n}_{min} \\ \text{Leave the TEB unchanged otherwise} \end{cases}$$

Deformation in space

Trajectory Planner - Deformation

Deformation in time

During each control cycle, the following TEB update is performed $\bar{I}_{TEB}$ times

$$\text{TEB update i} - \begin{cases} \text{Insert a new state if } \Delta T_i > \Delta \bar{T}_{ref} + \Delta \bar{T}_{hyst} \wedge n_i < \bar{n}_{max} \\ \text{Remove a state if } \Delta T_i < \Delta \bar{T}_{ref} - \Delta \bar{T}_{hyst} \wedge n_i > \bar{n}_{min} \\ \text{Leave the TEB unchanged otherwise} \end{cases}$$

Deformation in space

After each TEB update, the trajectory is improved by running an optimization solver for $\bar{I}_{SQP}$ iterations with respect to the underlying non-linear optimization problem.

Trajectory Planner - Deformation

Deformation in time

During each control cycle, the following TEB update is performed $\bar{I}_{TEB}$ times

$$\text{TEB update i} - \begin{cases} \text{Insert a new state if } \Delta T_i > \Delta \bar{T}_{ref} + \Delta \bar{T}_{hyst} \wedge n_i < \bar{n}_{max} \\ \text{Remove a state if } \Delta T_i < \Delta \bar{T}_{ref} - \Delta \bar{T}_{hyst} \wedge n_i > \bar{n}_{min} \\ \text{Leave the TEB unchanged otherwise} \end{cases}$$

Deformation in space

After each TEB update, the trajectory is improved by running an optimization solver for $\bar{I}_{SQP}$ iterations with respect to the underlying non-linear optimization problem.

- The underlying solver is based on *Sequential Quadratic Programming* and employs line-search through an l_1 merit function.

Trajectory Planner - Deformation

Deformation in time

During each control cycle, the following TEB update is performed $\bar{I}_{TEB}$ times

$$\text{TEB update i} - \begin{cases} \text{Insert a new state if } \Delta T_i > \Delta \bar{T}_{ref} + \Delta \bar{T}_{hyst} \wedge n_i < \bar{n}_{max} \\ \text{Remove a state if } \Delta T_i < \Delta \bar{T}_{ref} - \Delta \bar{T}_{hyst} \wedge n_i > \bar{n}_{min} \\ \text{Leave the TEB unchanged otherwise} \end{cases}$$

Deformation in space

After each TEB update, the trajectory is improved by running an optimization solver for $\bar{I}_{SQP}$ iterations with respect to the underlying non-linear optimization problem.

- The underlying solver is based on *Sequential Quadratic Programming* and employs line-search through an l_1 merit function.
- Automatic differentiation is used to compute the required gradients and Jacobians.

Trajectory Planner - Deformation

Deformation in time

During each control cycle, the following TEB update is performed $\bar{I}_{TEB}$ times

$$\text{TEB update i} - \begin{cases} \text{Insert a new state if } \Delta T_i > \Delta \bar{T}_{ref} + \Delta \bar{T}_{hyst} \wedge n_i < \bar{n}_{max} \\ \text{Remove a state if } \Delta T_i < \Delta \bar{T}_{ref} - \Delta \bar{T}_{hyst} \wedge n_i > \bar{n}_{min} \\ \text{Leave the TEB unchanged otherwise} \end{cases}$$

Deformation in space

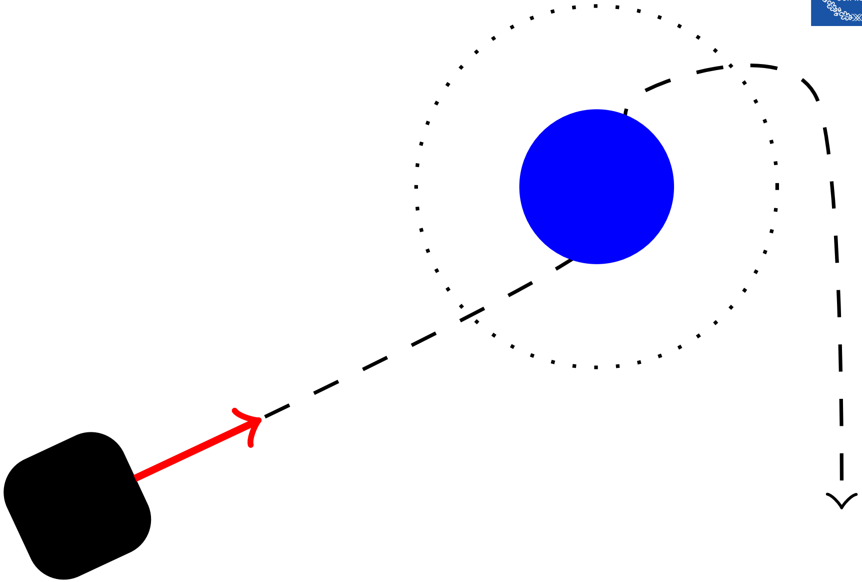
After each TEB update, the trajectory is improved by running an optimization solver for $\bar{I}_{SQP}$ iterations with respect to the underlying non-linear optimization problem.

- The underlying solver is based on *Sequential Quadratic Programming* and employs line-search through an l_1 merit function.
- Automatic differentiation is used to compute the required gradients and Jacobians.

In total, $\bar{I}_{TEB} \cdot \bar{I}_{SQP}$ optimization iterations are performed each cycle.

Outline

Trajectory Planner - Collision avoidance



Trajectory Planner - Collision avoidance

$$\min_{\mathcal{B}} \quad (n-1)\Delta T - \sum_{j=1}^{m} \sum_{k \in K_{j,\bar{\sigma}_{op}}} ||\chi_y(C\boldsymbol{x}_k) - \mathcal{O}_j||^2$$

$$\text{s.t.} \quad \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\Delta T} - A\boldsymbol{x}_k + B(f(\boldsymbol{x}_k) + g(\boldsymbol{x}_k)\boldsymbol{\tau}_k) = 0 \quad (k = 1, \ldots, n-1)$$

$$\boldsymbol{\tau}_- \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_+ \quad (k = 1, \ldots, n-1)$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_s, \ \boldsymbol{x}_n = \boldsymbol{x}_f, \ \Delta T > 0$$

Outline

Trajectory Planner - Track moving targets

- The objective is to track some moving target, represented here by the curve $\boldsymbol{y}^{tg}(t)$ in operational space.

Trajectory Planner - Track moving targets

- The objective is to track some moving target, represented here by the curve $\boldsymbol{y}^{tg}(t)$ in operational space.
- In configuration space, the end-condition becomes

$$\boldsymbol{q}_f = \chi_y^{-1}(\boldsymbol{y}^{tg}(T))$$
$$\dot{\boldsymbol{q}}_f = J(\boldsymbol{q}_f)^{-1}\dot{\boldsymbol{y}}^{tg}(T)$$

Trajectory Planner - Track moving targets

- The objective is to track some moving target, represented here by the curve $\boldsymbol{y}^{tg}(t)$ in operational space.
- In configuration space, the end-condition becomes

$$\boldsymbol{q}_f = \chi_y^{-1}(\boldsymbol{y}^{tg}(T))$$
$$\dot{\boldsymbol{q}}_f = J(\boldsymbol{q}_f)^{-1}\dot{\boldsymbol{y}}^{tg}(T)$$

- Alternatively, the target state is replaced with the prediction

$$\boldsymbol{y}_i^{tg} + (n_{i-1} - 1)\Delta T_i \boldsymbol{v}$$

Outline

Algorithm 1 Trajectory Planning

Input: $\boldsymbol{q}_s$ - current state; $\dot{\boldsymbol{q}}_s$ - current velocity; $\boldsymbol{y}_f$ - target; $\dot{\boldsymbol{y}}_f$ - target velocity; $\mathcal{O}$ - obstacle information

Output: (Sub-)optimal control input $\boldsymbol{\tau}$

procedure PlanTrajectory
    repeat
        $(\boldsymbol{q}_s, \dot{\boldsymbol{q}}_s, \boldsymbol{y}_f, \dot{\boldsymbol{y}}_f) \leftarrow$ ReadSensorInput
        $\mathcal{O} \leftarrow$ InformAboutObstacles
        for each iteration 1 to $\bar{I}_{TEB}$ **do**
            $\mathcal{B} \leftarrow$ DeformInTime($\mathcal{B}$)
            $P \leftarrow$SetupUnderlyingProblem($\mathcal{B}, \mathcal{O}, \boldsymbol{q}_s, \dot{\boldsymbol{q}}_s, \boldsymbol{y}_f, \dot{\boldsymbol{y}}_f$)
            for each iteration 1 to $\bar{I}_{SQP}$ **do**
                $\mathcal{B} \leftarrow$ SQPSolve($\mathcal{B}$,P)
            end for
        end for
        $\tau \leftarrow$ ApplyControl($\mathcal{B}$)
    until target has been reached
end procedure

Trajectory Planner - Implementation

- The trajectory planner is implemented as a software package in C++.

Trajectory Planner - Implementation

- The trajectory planner is implemented as a software package in C++.
- Notable third-party libraries:

Trajectory Planner - Implementation

- The trajectory planner is implemented as a software package in C++.
- Notable third-party libraries:
  - *Eigen* - for matrix/vector storage and linear algebra operations.

Trajectory Planner - Implementation

- The trajectory planner is implemented as a software package in C++.
- Notable third-party libraries:
  - *Eigen* - for matrix/vector storage and linear algebra operations.
  - *qpOASES* - for solving the arising quadratic subproblems during the SQP procedure.

Trajectory Planner - Implementation

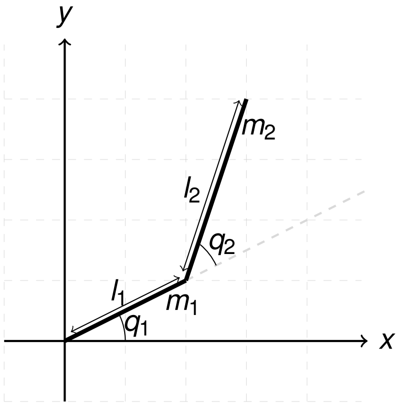
- The trajectory planner is implemented as a software package in C++.
- Notable third-party libraries:
  - *Eigen* - for matrix/vector storage and linear algebra operations.
  - *qpOASES* - for solving the arising quadratic subproblems during the SQP procedure.
  - *CppAD* - for automatic differentiation. Used to compute gradients and Jacobians.

Trajectory Planner - Implementation

- The trajectory planner is implemented as a software package in C++.
- Notable third-party libraries:
  - *Eigen* - for matrix/vector storage and linear algebra operations.
  - *qpOASES* - for solving the arising quadratic subproblems during the SQP procedure.
  - *CppAD* - for automatic differentiation. Used to compute gradients and Jacobians.
- In specific applications, the trajectory planner is extended in a subclass that configures the planner and provides the appropriate system dynamics.

Outline

Simulations - PlanarElbow/SCARA model



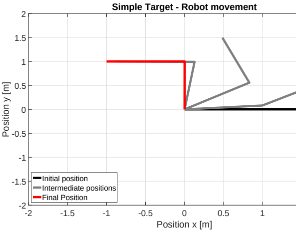
Outline

Scenario 1: Pick and place

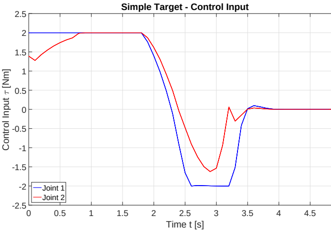
Demonstration

Scenario 1: Simple target - Time

(a) Snapshots of the intermediate planned trajectories, taken every 0.5s, together with the actual realized trajectory.



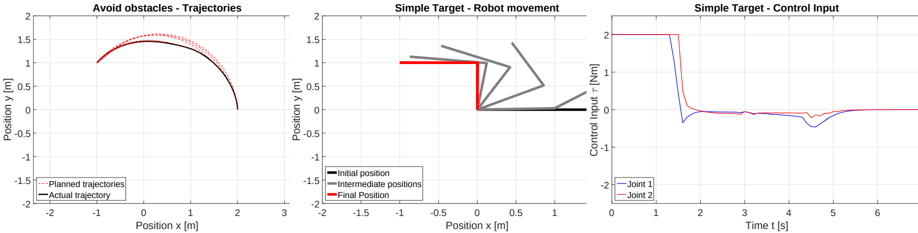
(b) The movement pattern of the robot



(c) The control input signal that was applied during the procedure.

Figure: Trajectory planning procedure for the *PlanarElbow* model with a simple stationary target at $(-1, 1)$ and aiming to minimize transition time. The planner was configured with the default values.

Scenario 1: Simple target - Energy



(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.

(b) The movement pattern of the robot.

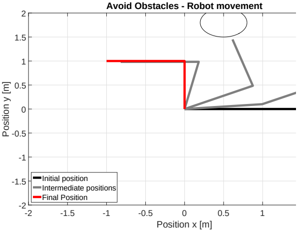
(c) The control input signal that was applied during the procedure.

Figure: Trajectory planning procedure for the *PlanarElbow* model with a simple stationary target at $(-1, 1)$ and aiming to minimize energy. The planner was configured with the default values.

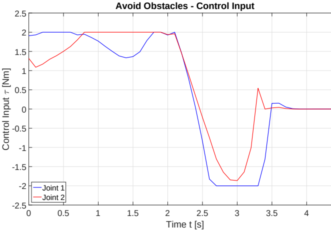
Outline

Scenario 2: Avoid obstacles

(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.



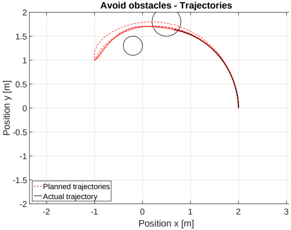
(b) The movement pattern of the robot.



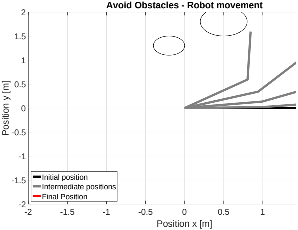
(c) The control input signal that was applied during the procedure.

Figure: A single obstacle with radius $0.3\mathrm{m}$ is placed at $(0.5, 1.8)$. The planner was configured with the default values.

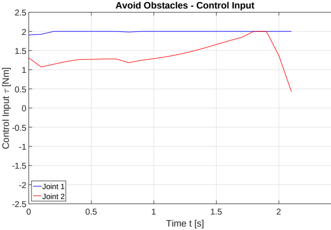
Scenario 2: Avoid obstacles



(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.



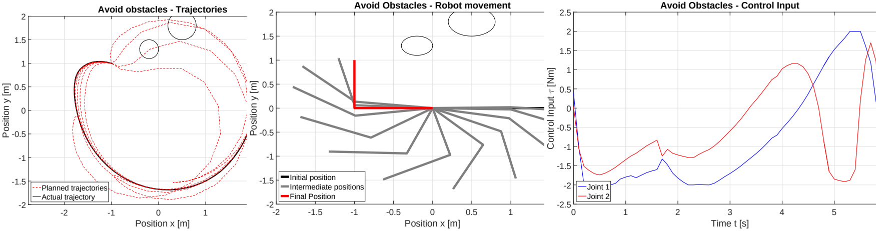
(b) The movement pattern of the robot.



(c) The control input signal that was applied during the procedure.

Figure: Two obstacles are added to the workspace: one at $(0.5, 1.8)$ with radius $0.3\mathrm{m}$ and one at $(-0.2, 1, 3)$ with radius $0.2\mathrm{m}$. The planner was configured with the default values.

Scenario 2: Avoid obstacles



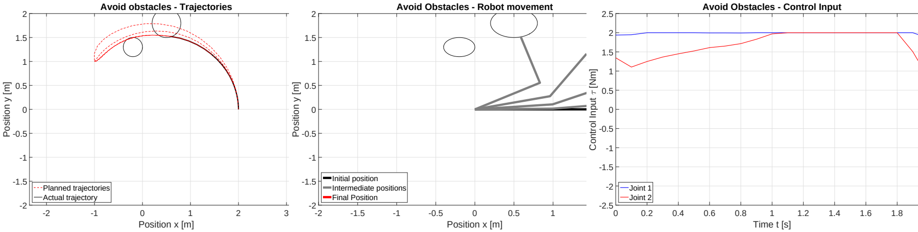
(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.

(b) The movement pattern of the robot.

(c) The control input signal that was applied during the procedure.

Figure: Two obstacles are added to the workspace: one at $(0.5, 1.8)$ with radius $0.3\mathrm{m}$ and one at $(-0.2, 1, 3)$ with radius $0.2\mathrm{m}$. The planner was configured with the default values, but with `"obstacleCloseProximity" : 1`.

Scenario 2: Avoid obstacles



(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.

(b) The movement pattern of the robot.

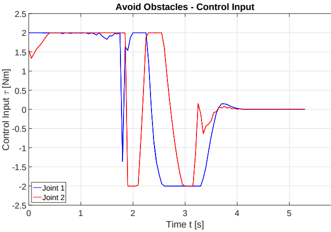
(c) The control input signal that was applied during the procedure.

Figure: Two obstacles are added to the workspace: one at $(0.5, 1.8)$ with radius $0.3\mathrm{m}$ and one at $(-0.2, 1, 3)$ with radius $0.2\mathrm{m}$. The planner was configured with the default values, but with `"obstacleCloseProximity" : 0.1`.

Scenario 2: Avoid obstacles

(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.

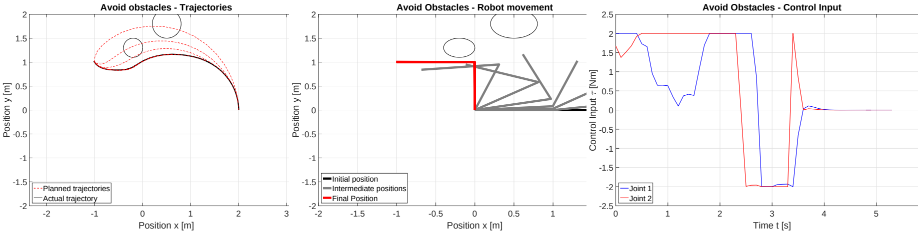
(b) The movement pattern of the robot.



(c) The control input signal that was applied during the procedure.

Figure: Two obstacles are added to the workspace: one at $(0.5, 1.8)$ with radius $0.3\mathrm{m}$ and one at $(-0.2, 1, 3)$ with radius $0.2\mathrm{m}$. The planner was configured with the default values, but with `"referenceTime" : 0.05` and `"Iteb" : 3`. The simulation was run with a sample time of $0.05\mathrm{s}$.

Scenario 2: Avoid obstacles



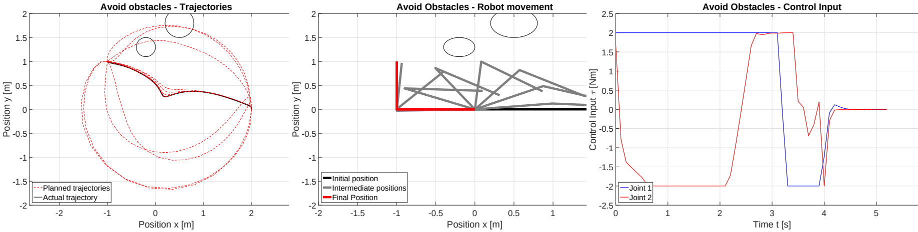
(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.

(b) The movement pattern of the robot.

(c) The control input signal that was applied during the procedure.

Figure: Two obstacles are added to the workspace: one at $(0.5, 1.8)$ with radius $0.3\mathrm{m}$ and one at $(-0.2, 1, 3)$ with radius $0.2\mathrm{m}$. The planner was configured with the default values, but with `"Isqp" : 4`.

Scenario 2: Avoid obstacles



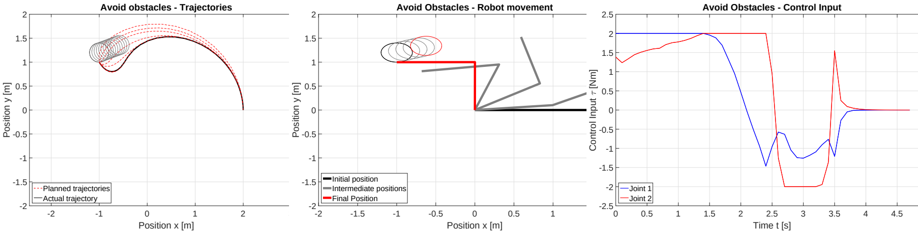
(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.

(b) The movement pattern of the robot.

(c) The control input signal that was applied during the procedure.

Figure: Two obstacles are added to the workspace: one at $(0.5, 1.8)$ with radius $0.3\mathrm{m}$ and one at $(-0.2, 1, 3)$ with radius $0.2\mathrm{m}$. The planner was configured with the default values, but with `"Isqp" : 4` and `"multipleTrajectories": true`.

Scenario 2: Avoid obstacles



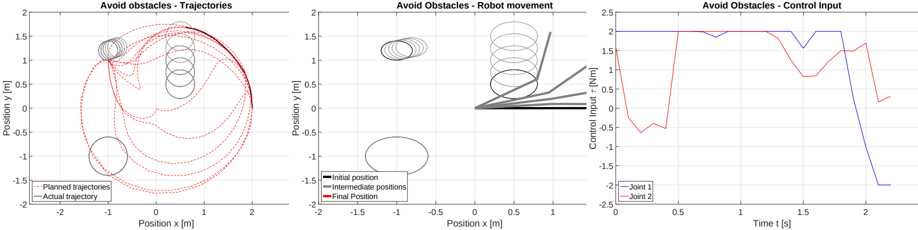
(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.

(b) The movement pattern of the robot and the moving obstacle.

(c) The control input signal that was applied during the procedure.

Figure: A single obstacle with radius $0.2\mathrm{m}$ is placed at $(1, 1.2)$, and moving in the direction $(0.94, 0.35)$ with speed $0.1\mathrm{m/s}$. The planner was configured with the default values.

Scenario 2: Avoid obstacles



(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.

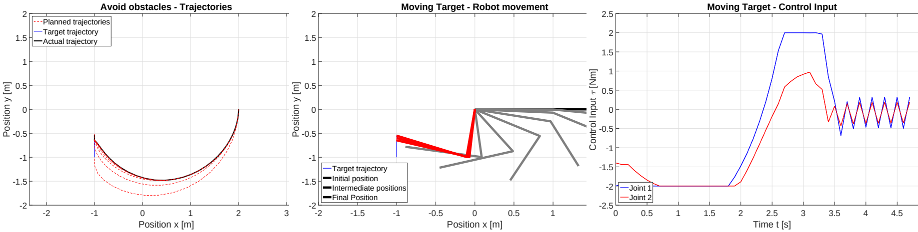
(b) The movement pattern of the robot and the moving obstacles.

(c) The control input signal that was applied during the procedure.

Figure: Three obstacles are added to the workspace: One at $(1, 1.2)$ with radius $0.2\mathrm{m}$, moving in the direction $(0.94, 0.35)$ with speed $0.1\mathrm{m}/\mathrm{s}$; One stationary obstacle at $(-1, -1)$ with radius $0.4\mathrm{m}$; and finally one at $(0.5, 0.5)$ with radius $0.3\mathrm{m}$, moving in the direction $(0, 1)$ with speed $5\mathrm{m}/\mathrm{s}$. The planner was configured with the default values, but with `"multipleTrajectories" : true`.

Outline

Scenario 3: Track moving target



(a) Snapshots of the intermediate planned trajectories together with the actual realized trajectory.

(b) The movement pattern of the robot and the moving target.

(c) The control input signal that was applied during the procedure.

Figure: Trajectory planning procedure for the *PlanarElbow* model with a moving target initially located at $(-1, -1)$ and moving in the direction $(0, 1)$ with speed $0.1\mathrm{m/s}$. The aim is to minimize transition time. The planner was configured with the default values.

Outline

Demonstration

Outline

Evaluation of results

- The planner is, in many different scenarios, successful in generating a feasible trajectory in relation to the different goals and the provided constraints.

Evaluation of results

- The planner is, in many different scenarios, successful in generating a feasible trajectory in relation to the different goals and the provided constraints.
- When employing the time-minimizing strategy, the resulting trajectories often appear to be quasi time-optimal.

Evaluation of results

- The planner is, in many different scenarios, successful in generating a feasible trajectory in relation to the different goals and the provided constraints.
- When employing the time-minimizing strategy, the resulting trajectories often appear to be quasi time-optimal.
- The results of the pick-and-place scenario shows the planners potential as an alternative to the two-step approach when operating in dynamic environments.

Evaluation of results

- The planner is, in many different scenarios, successful in generating a feasible trajectory in relation to the different goals and the provided constraints.
- When employing the time-minimizing strategy, the resulting trajectories often appear to be quasi time-optimal.
- The results of the pick-and-place scenario shows the planners potential as an alternative to the two-step approach when operating in dynamic environments.

- The results of the more complicated examples indicate that many of the employed strategies are too primitive, and need to be explored further.

Evaluation of results

- The planner is, in many different scenarios, successful in generating a feasible trajectory in relation to the different goals and the provided constraints.
- When employing the time-minimizing strategy, the resulting trajectories often appear to be quasi time-optimal.
- The results of the pick-and-place scenario shows the planners potential as an alternative to the two-step approach when operating in dynamic environments.

- The results of the more complicated examples indicate that many of the employed strategies are too primitive, and need to be explored further.
- Many of the examples, and the appararent parameter sensitivity, indicate that the planner needs to be tailored for use in specific applications.

Evaluation of results

- The planner is, in many different scenarios, successful in generating a feasible trajectory in relation to the different goals and the provided constraints.
- When employing the time-minimizing strategy, the resulting trajectories often appear to be quasi time-optimal.
- The results of the pick-and-place scenario shows the planners potential as an alternative to the two-step approach when operating in dynamic environments.

- The results of the more complicated examples indicate that many of the employed strategies are too primitive, and need to be explored further.
- Many of the examples, and the appararent parameter sensitivity, indicate that the planner needs to be tailored for use in specific applications.
- More work required to make the planner real-time capable.

Outline

Improvements and future work

- Utilize the sparsity of the underlying optimization problem.

Improvements and future work

- Utilize the sparsity of the underlying optimization problem.
- Explore trust-region methods as an alternative to line-search in the SQP procedure.

Improvements and future work

- Utilize the sparsity of the underlying optimization problem.
- Explore trust-region methods as an alternative to line-search in the SQP procedure.
- Robust MPC techniques to counter issues that arise from inaccurate models.

Outline

Final remarks

- A trajectory planning procedure has been implemented and presented in this work, and it looks promising for future use.

Final remarks

- A trajectory planning procedure has been implemented and presented in this work, and it looks promising for future use.
- The intended application has been robotic manipulators, but the planner can be extended to other optimal control problems.

Final remarks

- A trajectory planning procedure has been implemented and presented in this work, and it looks promising for future use.
- The intended application has been robotic manipulators, but the planner can be extended to other optimal control problems.
- The planner is successively applied in simple scenarios, but needs further work before it can be used in a real applications.

Questions?