Rapidly Exploring Random Trees-based Initialization of MPC Technique Designed for Formations of MAVs

Zdeněk Kasl, Martin Saska and Libor Přeučil
Department of Cybernetics, Czech Technical University, Technická 2, 166 27 Prague, Czech Republic

Keywords: Trajectory Planning, Model Predictive Control, Micro Aerial Vehicles, Rapidly Exploring Random Trees.

Abstract: Motion planning techniques suited for initialization of Model Predictive Control based methodology applied for complex maneuvering and stabilization of formations of Micro Aerial Vehicles are proposed in this paper. Two approaches to initialization of the formation driving method will be described, experimentally verified, evaluated and compared. The first proposed method is based on multiobjective optimization of the trajectory guess obtained by a Rapidly Exploring Random Trees technique. It represents an easy to implement and robust method suited for off-line initialization of the formation driving algorithm. The second proposed method is based on sequential processing of parts of the obtained trajectory. This method is well scalable and thus applicable in large workspaces with complex obstacles. In addition, the second method enables a significant reduction of computational time as is shown by comparison of series of simulations in different environments.

1 INTRODUCTION

The group cooperation of autonomous robots became an intensively studied topic in the past years. Most recently, the research got focused on unmanned Micro Aerial Vehicles (MAVs), quadrotors, and their cooperation. Control mechanisms and computational power onboard of small-size MAVs enable their deployment in teams with close mutual interaction. The precisely stabilized MAV formations of compact shapes are especially appealing for cooperative manipulation (Mellinger et al., 2013) carrying of objects too heavy for single MAV (Maza et al., 2010) or cooperative surveillance (Merino et al., 2007).

In this paper, we focus on a formation driving mechanism suited for most of these applications. The Model Predictive Control (MPC) based method is ready to use in environments without precise external localization system and in situations where reliability and precision of global navigation satellite systems is insufficient. Our developed formation driving system (Saska et al., 2014) relies on precise relative localization of neighbouring MAVs using onboard image processing system with simple monocular cameras (Krajník et al., 2014). The proposed paper brings a crucial contribution to the formation driving system, which enables its deployment in complex large workspaces due to an appropriate initial guess of the trajectory of the formation.

In most of the path tracking and formation stabilization approaches, e.g. see (Abdessameud and Tayebi, 2011; Chen et al., 2010) and references references cited therein, it is supposed that the desired trajectory followed by the formation is given as an input of these methods. Our method goes beyond these works, as it does not rely on following a given trajectory and the global trajectory planning is directly integrated into the formation control mechanism. Therefore, the method can continuously respond to changes in the MAVs vicinity, while keeping the cohesion of the immediate control inputs with the directions of movement of the formation in the future.

In addition, the proposed novel initialization of this MPC concept enables to extend applicability of the method into arbitrarily complex environments and it significantly increases its robustness. The MPC method relies on optimisation process that can be time consuming if improperly initialized. In order to enable the trajectory optimization on-board, the optimization process should be initialized with a feasible initial trajectory.

The motion planning methods presented in this paper are primarily suited for planning of the initial trajectory for MAV formations, but may be efficiently used also for planning of smooth trajectories of individual quadrotors, since MAV motion constraints are satisfied in the proposed algorithms. Both of the proposed methods are based on the Rapidly

Kasl Z., Saska M. and Přeučil L..
Rapidly Exploring Random Trees-based Initialization of MPC Technique Designed for Formations of MAVs.
DOI: 10.5220/0005058504360443
In *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2014), pages 436-443
ISBN: 978-989-758-040-6

exploring Random Trees (RRT) technique (LaValle, 1998). The RRT algorithm has been chosen for the required motion planning due to its computational efficiency and possibility to integrate the motion constraints of MAVs as well as motion constraints of formations of MAVs, which depend on possibly changing shape of the group. In addition, RRT provides the output trajectory as a vector of control inputs. Therefore, it can be directly employed as the initialization of the MPC optimization process. In the proposed methods, the optimality (mainly smoothness and length) of the trajectory found by the RRT algorithm is improved by consequent multiobjective optimization. This allows us to define the objectives, which are similar to those used in the MPC formation driving method, already within the initialization. The proposed approach to initialization of the MPC is necessary for real deployment of the MAVs, where the on-board available computational power is limited.

2 METHODS OVERVIEW

The methods proposed in this article are devised for suitable initialization of the MPC-based approach designed for guidance of formations of MAVs (Saska et al., 2014), which is described in the first subsection 2.1. In subsection 2.2, the RRT algorithm is described. The RRT is used for finding an initial guess of the input trajectory, which is further optimized before using as the initialization vector for the MPC.

2.1 Model Predictive Control

Model Predictive Control is a control algorithm originally designed for industrial applications (Mayne et al., 2000). The method became widely used in robotic applications with increasing computational power of available computers.

The MPC method relies on the mathematical model of the controlled system. The control problem, in this case integrated formation control and trajectory planning, is coded as a vector of control inputs. Employing MPC, a solution of the control problem is found by optimization of the vector of control inputs, which is evaluated based on the system response prediction.

The optimization vector consists of N control inputs, each with constant duration Δt, in original MPC methods. It is forming so called prediction horizon. We proposed an extension of the standard MPC method for purposes of the trajectory planning in (Saska et al., 2013). In the approach, the planning horizon is extended by additional M control inputs in order to be able to code the entire trajectory from starting to goal state. Those M inputs have variable time duration δ_i, $i = N+1 \ldots N+M$. Therefore, it is possible to code long trajectories with small number of control inputs. This setup enables the MPC to optimize the entire trajectory, using the optimization vector in a form $(u_1, \ldots, u_N, u_{N+1}, \ldots, u_{N+M})$, shown in Fig. 1. In the optimization vector, $u_i = (v_{\tau i}, v_{\nu i}, k_i, t_i)$ is the i-th control input, $v_{\tau i}$ and $v_{\nu i}$ denote horizontal and vertical velocity of the MAV, k_i represents the curvature of the given trajectory segment and $t_i \in \{\Delta t, \delta_i\}$ is the duration of the segment.

More detailed description of the trajectory planning for formations of MAVs under MPC scheme can be found in (Saska et al., 2014).

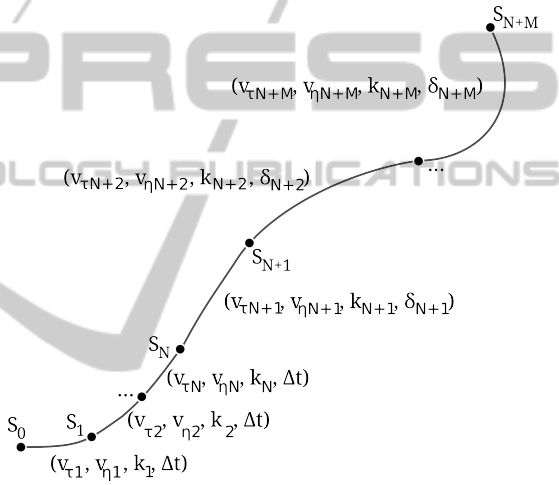


Figure 1: Trajectory represented as the optimization vector. Each control input $u_i = (v_{\tau i}, v_{\nu i}, k_i, t_i)$, $i = 1, \ldots, N+M$, drives the MAV between consecutive states S_{i-1} and S_i.

2.2 Rapidly Exploring Random Trees

Rapidly Exploring Random Tree is a motion planning algorithm designed for search in high dimensional spaces by filling them with a tree. The space–filling tree is rooted in the starting state S_0 and it is constructed in an iterative manner. The tree construction is terminated if $|S_{new} - S_{goal}| < \varepsilon$ or if the number of iterations exceeds the predefined limit. The constant ε is user–defined precision of the planning. It is convenient to define maximal number of iterations, which will terminate the search in case that the desired precision cannot be reached, e.g. when ε is too small. Finally, the trajectory can be constructed from parental references, starting from state S_{N+M} which is closest to the goal state. The detailed description of the algorithm can be found in (LaValle, 1998).

The main advantage of this algorithm lays in its

ability to accommodate kinematic and obstacle feasibility constraints to the planning process. Thus guaranteeing the feasibility of all the trajectories contained in the tree, e.g. when employed for nonholonomic or kinodynamic motion planning. Moreover, the coverage of the free space $\mathcal{C}_{free}$ can be performed in a short time even for environments with complex obstacles. Since the control inputs u are added along with the newly generated states, the outcome of RRT planning is a trajectory, which is already coded as an MPC optimization vector. However, it is not possible to guarantee optimality of the resulting trajectory, due to randomized approach to the tree construction.

3 INITIALIZATION METHODS

The core of the designed MPC method is the optimization process, which needs to be initialized with a guess of the trajectory before the first MPC step. The initial guess should be a feasible trajectory in order to assure that the method will find a solution which is also feasible. The RRT algorithm satisfies the conditions for the MPC initialization. However the resulting trajectories are not optimal and often composed of many control inputs, which would significantly slow down the trajectory planning process under the MPC scheme. The initialization methods proposed in this article take over the advantages of the RRT algorithm and produce suitable initial trajectories for the devised trajectory planning method.

Having the raw trajectory estimate as a vector of control inputs from the RRT algorithm (see Fig. 1), its optimality can be improved by optimization. Subsequently, it is possible to reduce the number of control inputs by merging consequent control inputs with similar values.

3.1 Trajectory Optimization

The optimization of the initial trajectory, which results in reduction of the amount of control inputs, relies on properly designed objective function. It evaluates the quality of the trajectory by user-defined criteria in the optimization process. In order to assure that the trajectory found by RRT algorithm is feasible for the formation guidance, several constraints have to be defined. Both, objective and constraint functions are described in the following sub-subsections.

3.1.1 Objective Function

In the proposed objective function, each objective is evaluated and the final value of the objective function is denoted as a weighted sum of costs of the individual objectives as

$$\mathcal{F} = \sum_{i=1}^{6} w_i \cdot \mathcal{F}_i. \tag{1}$$

The weights w_i are set by the user according to the importance of the respective objectives. In this case, the objectives that have to be minimized are:

Trajectory length function penalizes the length of the trajectory as

$$\mathcal{F}_1 = \sum_{i=1}^{N+M} t_i \sqrt{v_{\tau i}^2 + v_{\nu i}^2}, \tag{2}$$

where $v_{\tau i}$ and $v_{\nu i}$ are constant velocities in the i-th control input of length t_i.

Obstacle proximity function penalizes the trajectories that approach to obstacles closer than the safety radius r_s. The obstacle proximity function is determined as

$$\mathcal{F}_2 = \sum_{i=1}^{N+M} \left[\min\left(0, \frac{d_i - r_s}{d_i - r_a}\right) \right]^2, \tag{3}$$

where d_i is minimal distance from all obstacles over the i-th interval of the trajectory and r_a is critical radius. The value of the proximity function grows to infinity for $d_i \to r_a$. The trajectory is considered infeasible if it approaches to any of the obstacles closer than r_a.

Input curvature penalty is employed with the aim to prefer straight trajectories. This function is defined as

$$\mathcal{F}_3 = \sum_{i=1}^{N+M} k_i^2. \tag{4}$$

Consecutive Input Difference is the key aspect for achieving the simplification of the trajectory for the initialization. The aim of this function is to obtain trajectories having the consecutive control input interval values as similar as possible. Thus they can be merged together, making the trajectory more suitable for the initialization of the MPC method. This function is defined for each of the control inputs as follows:

$$\mathcal{F}_4 = \sum_{i=1}^{N+M} \left(v_{\tau i} - \bar{v}_\tau\right)^2, \tag{5}$$

$$\mathcal{F}_5 = \sum_{i=1}^{N+M} \left(v_{\nu i} - \bar{v}_\nu\right)^2, \tag{6}$$

$$\mathcal{F}_6 = \sum_{i=1}^{N+M} \left(k_i - \bar{k}\right)^2, \tag{7}$$

where $\bar{\cdot}$ denotes the mean value of the individual values over all $N+M$ control inputs.

3.1.2 Constraint Functions

The above described fitness function ensures optimality of the resulting trajectory, but the feasibility of the trajectory is not guaranteed. Therefore, it is required to apply the following set of constraints:

Motion constraints are applied to each of the control inputs u_i in order to keep them in a feasible interval. The bounds u_{min} and u_{max} are given by the controlled vehicle. The control inputs are limited as

$$u_{j,\min} < u_{j,i} < u_{j,\max},\ i = 1 \ldots N+M, \tag{8}$$

where $j = 1 \ldots 4$ denotes the individual control input (v_τ, v_ν, k, t) being evaluated and its respective limits.

Minimal obstacle distance ensures that the trajectory will not lead to a collision with obstacles. It also guarantees that the denominator in (3) is always greater than zero. The constraint is defined as

$$r_a - d_i < 0,\ i = 1 \ldots N+M, \tag{9}$$

where r_a is the minimal allowed distance to obstacles and d_i is the minimal distance to all obstacles of the i-th part of the trajectory.

Final position constraint assures that the goal region is reached if the trajectory is followed. The goal region is defined by its central point $\mathbf{X}_g$ and radius r_g:

$$|\mathbf{X}_g - \mathbf{X}_{N+M}| < r_g. \tag{10}$$

Vector $\mathbf{X}_{N+M}$ denotes the last point of the followed trajectory.

If the sequential trajectory processing is employed (see sec. 3.4), an additional constraint

$$|\gamma_0 - \gamma| < \varepsilon_\gamma \tag{11}$$

has to be considered. In the constraint, γ is the yaw Euler angle, γ_0 is the desired yaw Euler angle and ε_γ is the user–defined tolerance of the final angle.

3.2 Trajectory Simplification

The definition of the consecutive input difference objective function ensures that the consecutive control inputs defining the trajectory will be as similar as possible after the optimization. Subsequently, the similar successive control inputs are merged into one with adjusted parameters and prolonged time duration. It leads to shortening the length of the optimization vector and reducing the time needed for optimization of the trajectory. Such an initial trajectory will improve the time needed for finding the optimal solution by the MPC method, which is the main purpose of the proposed methods.

Having two consecutive control inputs $u_1 = (v_{\tau 1}, v_{\nu 1}, k_1, t_1)$ and $u_2 = (v_{\tau 2}, v_{\nu 2}, k_2, t_2)$, it is possible to merge them into one input, if they fulfil three conditions. The first two conditions concern the velocities of the consecutive control inputs:

$$|v_{\tau 1} - v_{\tau 2}| < \varepsilon_{v\tau} \tag{12}$$

and

$$|v_{\nu 1} - v_{\nu 2}| < \varepsilon_{v\nu}, \tag{13}$$

where $\varepsilon_{v\tau}$ and $\varepsilon_{v\nu}$ denote the user-defined velocity tolerance. This tolerance has to be defined with respect to the difference of the final state, which may be caused by merging of the two consecutive control intervals. The difference is proportional to the respective tolerance and to the duration of the interval. In this case, the difference in horizontal or vertical movement may be compensated by shortening or prolonging the duration of the interval and by appropriate adjusting the average velocity values. However, an extra attention has to be paid to respecting the limits $u_{\min}$ and $u_{\max}$.

The third condition is defined as

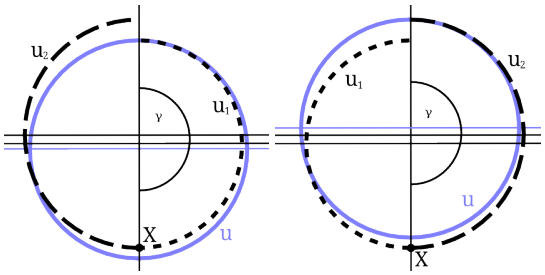
$$|k_1 - k_2| < \varepsilon_k. \tag{14}$$

The analysis of influence of curvature difference between two consecutive segments to the final state after their merging is not as straightforward as in the preceding case. The worst case scenario is depicted in Fig. 2. Applying the control input u_1 moves the MAV to $\gamma = 180^\circ$ along a circular trajectory and applying the control input u_2 moves it by another 180° along a circular trajectory with different curvature. The position error is then equal to double of the difference

$$\Delta r = \frac{k_2 - k_1}{k_1 k_2}. \tag{15}$$

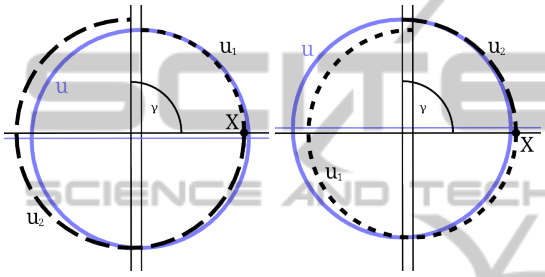
In this case, the tolerance should be determined with respect to the biggest allowed curvature. It is shorter trajectory to travel 180° around a circular trajectory with maximal curvature and therefore it is more likely that the worst case scenario will occur. With decreasing curvature, the crossing point moves towards 90° (see Fig. 3) and 45° (see Fig. 4) and the resulting difference becomes smaller than the double of radius difference.

With all merging criteria defined, it is possible to loop through the trajectory and merge together the consecutive control intervals that fulfil the conditions of similarity. The control inputs applied on



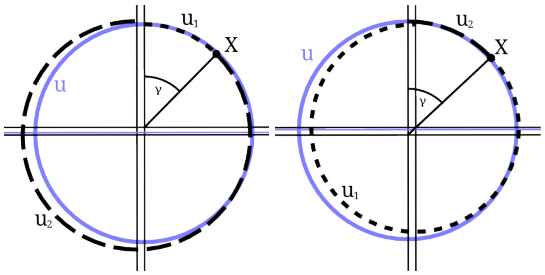
(a) u_1 to u_2 transition (b) u_2 to u_1 transition

Figure 2: The worst case scenario occurs when merging two circular trajectories that cross at $\gamma = 180°$. The difference in the final state is equal to the radius difference Δr. The lighter color denotes the average control input u, replacing control inputs u_1 and u_2.



(a) u_1 to u_2 transition (b) u_2 to u_1 transition

Figure 3: Crossing at $\gamma = 90°$. The difference in the final state is smaller than the radius difference Δr. The lighter color denotes the average control input u, replacing control inputs u_1 and u_2.



(a) u_1 to u_2 transition (b) u_2 to u_1 transition

Figure 4: Crossing at $\gamma = 45°$ and less, which is the scenario most likely to occur for control inputs with small curvatures. The difference in the final state is also smaller than the radius difference Δr. The lighter color denotes the average control input u, replacing control inputs u_1 and u_2.

the merged consecutive intervals are obtained as follows:

$$u = \left(\bar{v}_\tau, \bar{v}_\nu, \bar{k}, t_1 + t_2\right), \tag{16}$$

where the operation $\bar{\cdot}$ denotes the average value of the two inputs u_1 and u_2.

3.3 Naive Approach

This approach to a suitable initialization of the MPC method is an intuitive employment of the described optimization approach and the proposed trajectory simplification method to a trajectory found by the RRT algorithm. The main idea is that the whole trajectory obtained from the RRT algorithm is simplified together, using the previously described methods.

The trajectory found by the RRT algorithm is composed of many control inputs and it is usually far from the optimal result. The approach to construction of the space-filling tree suggests that the trajectory will often change direction and thus it is difficult to simplify. This is the reason why the trajectory found by the RRT is pre-optimized, using the approach described in section 3.1. The objective function of the optimization process was designed with the aim of unification of the consecutive intervals of the trajectory. Therefore, it is possible to employ the trajectory simplification method described in section 3.2 and merge all control inputs with similar values together. In a general case, the simplified trajectory is optimal with respect to the objective function defined in section 3.1 and it is described by less control inputs. Thus it is suitable for the initialization of the MPC.

The main weakness of this approach is that it suffers form the length of the optimization vector generated by the RRT algorithm. In some cases, the optimization may take minutes to complete. This is why the naive approach is suitable merely for applications that allow offline pre-calculation of the initial trajectory with no time constraints.

3.4 Sequential Processing

The sequential processing approach exploits the fact that the time consumed by the optimization grows as $O(n^2)$, where n is the number of control inputs describing the trajectory, i.e. $n = |u| \cdot (N+M)$. Therefore, piecewise processing of the trajectory speeds up the optimization process.

In order to preserve start and end state of the whole trajectory, start and end states of each of the processed parts of the trajectory should be fixed. Moreover, the yaw Euler angle γ in each start and end configuration has to be fixed as well, in order to enable putting the trajectory smoothly back together. The optimization of the pieces of the trajectory is implemented according to section 3.1 and the consequent simplification is performed as described in section 3.2.

Once the simplification of smaller pieces of

the trajectory is complete, the overall optimization has to be performed in order to improve the trajectory to its final state. Subsequently, the simplification is done producing the final trajectory.

In some cases, the piecewise reduction of the trajectory does not affect the result and the number of control inputs remains the same. Then, it is efficient to replan the RRT trajectory estimate. The probability of occurrence of this problem can be reduced by planning several trajectories, exploiting the time efficiency of the RRT algorithm in comparison with the time demanding trajectory optimization in the post-processing phase. From the obtained set, the best candidate trajectory can be selected e.g. by evaluating with subset of objectives defined in optimization objective function in section 3.1.

If a multi-core processor is available, it is possible to run this method as a parallel processing. Running the optimization of the parts of the trajectory in parallel threads fully exploit the benefits of the processor.

4 EXPERIMENTS

The experimental verification of the properties of the proposed methods is described in this section. The RRT algorithm uses kinematic model described in (Saska et al., 2014) employing constraints shown in tab. 1. Each state S_{new} is selected from a set of candidate states that are generated by applying a set of control inputs $\mathcal{U}$. The set $\mathcal{U}$ contains combinations of control inputs with horizontal velocity $v_\tau = 0.6\,\mathrm{m\cdot s^{-1}}$, normal velocities $v_\nu \in \{-0.3, 0.0, 0.3\}\,\mathrm{m\cdot s^{-1}}$ and curvatures $k \in \{-1.0, 0.0, 1.0\}\,\mathrm{m^{-1}}$. The duration of the control inputs is reduced with increasing number of iterations, as $t = 2.5\,\mathrm{s}$ for $1 < i < 2000$; $t = 0.5\,\mathrm{s}$ for $2001 < i < 5000$ and $t = 0.1\,\mathrm{s}$ for $5001 < i < 10000$. The decreasing length of control inputs helps to maintain rapid coverage of the free space and to reduce size of the tree. Moreover, the decreasing length of the control inputs helps to find trajectories that are composed of fewer–longer control inputs, rather than more shorter control inputs. A counterexample with fixed control input duration $t = 0.1\,\mathrm{s}$ is also presented. Maximal number of iterations of the RRT algorithm is set to $m_{iter} = 10000$.

The choice of the duration of the control input influences the level of detail of planning. Therefore, it is crucial to pay attention to its choice if the algorithm is used for planning in an environment with narrow passages with respect to the formation size.

The trajectory merging constants were set according to section 3.2 as follows: $\varepsilon_{v\tau} = 0.01\,\mathrm{m\cdot s^{-1}}$; $\varepsilon_{v\nu} = 0.01\,\mathrm{m\cdot s^{-1}}$ and $\varepsilon_k = 0.01\,\mathrm{m^{-1}}$.

Table 1: Kinematic constraints of an MAV.

	$v_\tau\,[\mathrm{m\cdot s^{-1}}]$	$v_\nu\,[\mathrm{m\cdot s^{-1}}]$	$k\,[\mathrm{m^{-1}}]$	$t\,[\mathrm{s}]$
$u_{\min}$	0	−0.3	−1	0
$u_{\max}$	0.6	0.3	1	10

4.1 Tree and Trajectory

An example of the space-filling tree generated by the RRT algorithm is shown in Fig. 5. The bold line denotes the resulting trajectory found by the algorithm.

As you may notice, the trajectory does not end in the desired goal point exactly, which is fine for purposes of testing. In real deployment, if the distance between the final point of the trajectory and the desired goal point is higher than a predefined threshold, the trajectory has to be replanned.

The time efficiency of the RRT algorithm can be exploited by planning several trajectories and taking the most suitable one for the optimization. For example, a number of control inputs, final distance to goal point, existence of a loop within the trajectory or length of the trajectory can be taken as a criterion.

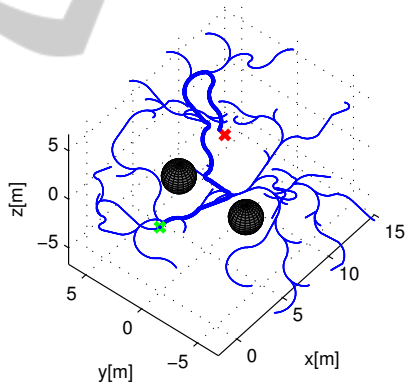


Figure 5: Space-filling tree, with resulting trajectory (bold).

4.2 Trajectory Optimization

The comparison of the raw trajectory obtained from the RRT algorithm and the optimization result is shown in Fig. 6. The initial trajectory (dark) was coded using 16 control inputs. The number of control inputs was reduced to 10 (light).

It may happen that the trajectory found by the RRT algorithm contains a loop. This situation is depicted in the horizontal plane projection in Fig. 7. Again, the dark trajectory is the raw trajectory obtained from the RRT algorithm and the light one is

the result of the optimization process. The control input reduction is not so significant as in the previous case due to the occurrence of the loop. Here, 25 original control inputs were reduced to 22. Therefore, it is important to detect and reject loopy trajectories.

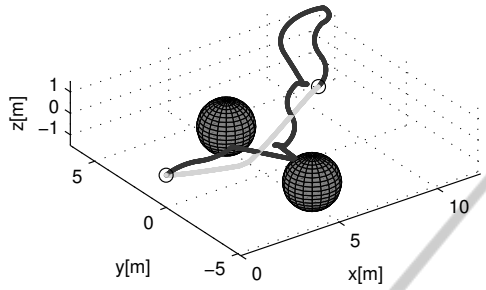


Figure 6: Comparison of trajectory generated by the RRT algorithm (dark) with trajectory after optimization (light). The circles denote start and end points of both trajectories.

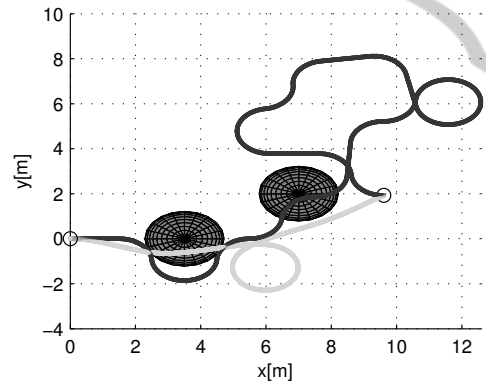


Figure 7: Example of trajectory containing loop before optimization (dark) and after optimization (light). The black circles denote start and end points of the trajectory.

4.3 Optimization Methods Comparison

The presented methods are compared concerning the time consumption in this sub-section. The first two experiments were performed in obstacle-free environment and in an environment with obstacles. The third experiment shows properties of presented algorithms when treating long trajectories (over 100 control inputs). This last scenario may happen if a detailed space coverage by the RRT algorithm is required, e.g. in a narrow passage. In all of the figures presented in this subsection, the measured data (rectangular and cross marks) were interpolated with an exponential function in order to illustrate the rate of time consumption increase.

4.3.1 Obstacle–free Environment

In this experiment, each of the proposed methods was run 1000 times with fixed start configuration and varying goal. All the time measurements were collected and the graph of average time consumption is shown in Fig. 8. The time consumption t is shown with respect to the number of control inputs n describing the trajectory. The length of the trajectory part for sequential processing was set to 4 control inputs.

The graph shows that the average time consumption in the obstacle-free environment is similar. The sequential method time consumption is influenced by the choice of number of control inputs for sub-optimization. The graph shows, that the sequential method finds solution faster than the naive method for trajectories with more control inputs.

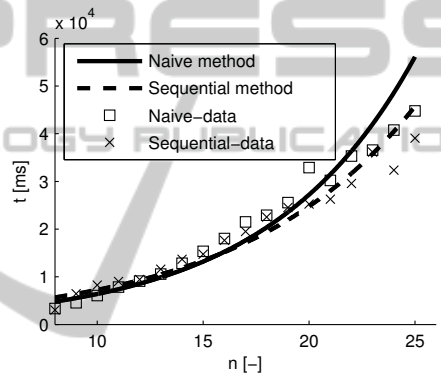


Figure 8: Comparison of performance of the proposed methods in obstacle-free environment. The variable n denotes the number of control inputs describing the trajectory.

4.3.2 Environment with Obstacles

The sequential processing method was used with the same setup as in the preceding experiment. An environment with 52 spherical obstacles was used for verification of the algorithm performance. Both methods were run 100 times and the time measurements were collected. The comparison in Fig. 9 shows that the maximal time consumption of the sequential method is significantly smaller in comparison to the naive method.

4.3.3 Fixed RRT Control Input Duration

In case that a more detailed coverage of the free space is required, the tree will produce much longer trajectory than in the preceding two scenarios, i.e. the trajectory is composed of over 100 control inputs. Both methods were run 100 times in this experiment in an environment with four spherical obstacles. The length of sub-optimization part of trajectory was

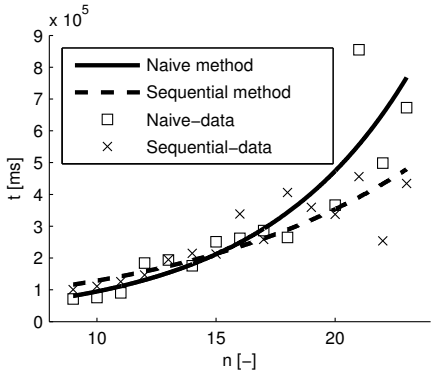


Figure 9: Comparison of performance of the proposed methods in environment with obstacles. The variable n denotes the number of control inputs describing the trajectory.

set to 10 control inputs. The comparison of average time consumption is shown in Fig. 10.

The graph shows that the time consumption of the naive method rapidly grows with increasing number of control inputs. The increase of time consumption of the sequential method is more gradual due to the reduction of the number of control inputs.

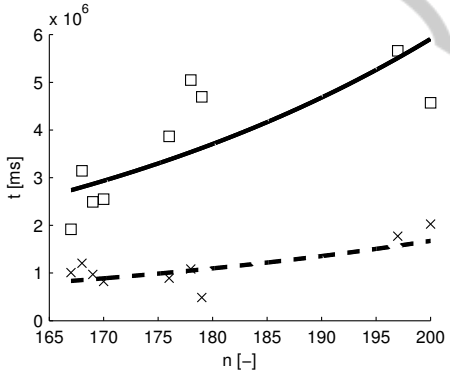


Figure 10: Comparison of performance of the proposed methods when planning long trajectories. The legend remains the same as in previous figures. The variable n denotes the number of control inputs describing the trajectory.

5 CONCLUSIONS

The motion planning methods presented in this paper respect kinematic constraints of the particular MAVs as well as the entire MAV formation. The resulting trajectory is coded as a vector of control inputs, i.e. in the form required by the MPC method. Therefore, these methods are suitable for initial trajectory planning for formations of MAVs. The proposed approaches increase the time efficiency of the method.

When employing the methods in a complex environment, an attention has to be paid to the choice of the merging constants as well as to the setup of the RRT algorithm. The experimental verification showed that the sequential processing method is more efficient, compared to the naive method. If the trajectory guess for the method is selected from a set of trajectories computed using the RRT algorithm, the time required to find a solution can be significantly reduced.

ACKNOWLEDGEMENTS

This work was supported by internal CTU grant no. SGS14/073/OHK3/1T/13, GAČR postdoc grant no. P103-12/P756 and MŠMT grant under Kontakt II no. LH11053.

REFERENCES

Abdessameud, A. and Tayebi, A. (2011). Formation control of vtol unmanned aerial vehicles with communication delays. *Automatica*, 47(11):2383 – 2394.

Chen, J., Sun, D., Yang, J., and Chen, H. (2010). Leader-follower formation control of multiple nonholonomic mobile robots incorporating a receding-horizon scheme. *Int. Journal Robotic Research*, 29:727–747.

Krajník, T., Nitsche, M., Faigl, J., Vaněk, P., Martin, S., Přeučil, L., Duckett, T., and Mejail, M. (2014). A practical multirobot localization system. *Journal of Intelligent and Robotic Systems*, pages 1–24.

LaValle, S. M. (1998). Rapidly-exploring random trees a new tool for path planning.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

Maza, I., Kondak, K., Bernard, M., and Ollero, A. (2010). Multi-uav cooperation and control for load transportation and deployment. *Journal of Intelligent and Robotic Systems*, 57(1-4):417–449.

Mellinger, D., Shomin, M., Michael, N., and Kumar, V. (2013). Cooperative grasping and transport using multiple quadrotors. In *Distributed autonomous robotic systems*, pages 545–558. Springer.

Merino, L., Caballero, F., Ferruz, J., Wiklund, J., Forssén, P.-E., and Ollero, A. (2007). Multi-uav cooperative perception techniques. In *Multiple Heterogeneous Unmanned Aerial Vehicles*, pages 67–110. Springer.

Saska, M., Kasl, Z., and Přeučil, L. (2014). Motion planning and control of formations of micro unmanned aerial vehicles. In *19th World Congress of IFAC*. To be published.

Saska, M., Mejía, J., Stipanovič, D. M., Vonásek, V., Schilling, K., and Přeučil, L. (2013). Control and navigation in manoeuvres of formations of unmanned mobile vehicles. *European Journal of Control*, 19(2):157–171.