Smooth Spline-based Trajectory Planning for Semi-Rigid Multi-Robot Formations*

Tobias Recker[1], Henrik Lurz[1] and Annika Raatz[1]

***Abstract*— This paper presents an approach for smooth trajectory planning in semi-rigid nonholonomic mobile robot formations using Beziér-splines. Unlike most existing approaches, the focus is on maintaining a semi-rigid formation, as required in many scenarios such as object transport, handling or assembly. We use a Relaxed A* planner to create an optimal collision-free global path and then smooth this path using splines. The smoothed global path serves to create target paths for every robot in the formation. From these paths, we then calculate the trajectories for each robot. In an iterative process, we match the velocities of the robots so that all trajectories are synchronized, and the dynamic limits of all robots are maintained. We provide experimental validation, which confirms no violation of the dynamic limits and shows an excellent control performance for a system of three robots moving at 0.3 m/s.**

I. INTRODUCTION

Robot formations are often used to extend the capabilities of a single robot or to break down complex tasks into simpler subtasks [1]. In this way, formations consisting of multiple mobile robots can transport objects that would otherwise be too heavy, big, or delicate for a single mobile robot. In many cooperative object transport scenarios, the distances between the individual robots must be constant. Any violation of the specified distances leads to the introduction of unwanted forces into the carried object. To keep the distances constant, the robots must drive in a semi-rigid formation from the start to the target position. Here the word formation describes a certain geometric arrangement/configuration of a group of robots. A semi-rigid formation means that the positions of the robots within a formation-fixed coordinate system do not change during the locomotion while the orientations can change.

Maintaining a semi-rigid formation proves to be a complex task when using nonholonomic mobile robots. The nonholonomic motion constraints limit the robots to forward and turning motions. Therefore, the formation's target path has to be planned in a way that each mobile robot can follow its target path, only requiring a combination of forward and turning motions while maintaining a defined (non-zero) velocity.

For industrial application, additional factors have to be considered. First, even small changes in the formation's orientation can lead to very high velocities and accelerations for the outer robots due to leverage. Hence, it is crucial to consider the velocity of every individual robot at the trajectory planning stage. Second, the global path should be efficient in terms of length and time while avoiding all obstacles. Third, very tight curves in the planned trajectory require the inner robots to move backwards. Backwards movements pose a risk to human workers, as humans may find it difficult to predict these movements. Consequently, the minimum curve radius should be limited. Considering all previously mentioned requirements we set the goal to develop a trajectory planning algorithm that creates smooth and C^2 continuous trajectories for use in semi-rigid multi-robot formations. The algorithm calculates a smooth trajectory to a given target location while considering predefined velocity and acceleration limits for every mobile robot in the formation and prevents reversing. We have already developed the underlying algorithm in [2] and refer the reader to this publication for more details. The following paper adds a formation controller, an algorithm to plan synchronized trajectories for all robots, and provides additional experimental evaluation.

The proposed multi-robot formation path and trajectory planning approach provides several advantages, with the main ones being:

1) smooth and synchronised trajectories with near-optimal length.
2) high-velocity transport, as (at least one) robot's defined dynamic limits are always fully utilized.
3) fast computation due to an efficient global planner and analytical algorithm.
4) convenient operation thanks to integration into the ROS navigation stack.
5) unlimited scalability in terms of the number of robots.

We start our work with a short introduction to the state of art in multi-robot formations and path planning. In section 3 we showcase the developed algorithm and follow up with section 4, where we present an experimental evaluation and results. This paper ends with a summary of our work and an outlook.

II. RELATED WORK

In multi-robot systems, path planning and formation control play an essential role. While there are many approaches in each of the two individual areas [3][4], papers that examine both areas in combination are much rarer. In the shared area, which we refer to as multi-robot formation path planning, there are three different subcategories. In the first category, the goal is to change the formation from an initial configuration to a target configuration without maintaining a fixed formation. An example of this is [5], where the

*This work was not supported by any organization

[1]All authors are with the Institute of Assembly Technology, University of Hanover, 30823 Garbsen, Germany Recker@match.uni-hannover.de

robots follow a target path but leave this path to avoid collisions. In [1], this principle is extended that even the role and position of the robots in the formation are no longer predefined. The proposed method plans the formation as a set, and afterwards determines which robot takes which role. The second category of path planning approaches for multi-robot formations includes works like [6] and [7]. In these works, a target path is planned for each mobile robot. The robots follow this path moving in a formation. However, in contrast to the third category, the formation is not strictly maintained especially in curves.

The third category introduces the theory of a virtual structure, which considers the individual mobile robots as particles embedded into a rigid body [8]. Due to the rigid body properties, the distances between all robots always stay the same, and the motion of one individual robot can be derived from the movement of the formation. While all three approaches control a formation in some way, only the virtual structure allows an object to be placed onto the robots without falling off [9]. Some works introduce a further distinction in this respect: Formations in which the orientation and spacing of all robots remain the same during motion are called strictly or perfectly rigid formations, while formations with constant spacing but different orientations are called flexible or semi-rigid [9]. Strictly rigid formations are only sensible with omnidirectional robots, as a formation of nonholonomic robots can only keep a constant orientation on a straight path. The nonholonomic constraints in the robot kinematics impose that a nonholonomic robot must always be oriented in the direction of travel. At the same time, these constraints also mean that the outer robots in a formation are subject to high velocities and accelerations whenever the formation changes its orientation. This strong excitation of the formation controller has a detrimental effect on the control performance and occurs increasingly with unsmooth formation paths [10]. For this reason, [11] developed an approach that bridges the aforementioned gap between path planning and formation control while also generating smooth trajectories. In their approach, an individual path is planned for each robot in the formation and passed to a decentralized path controller in the respective robot. The path controller has the task of moving the robot along the defined path. The target path for the path controller is generated by combining splines, which results in a smooth target path. A second centralized controller is added to determine which point on their path each robot should approach next. This control approach enables the robots to move synchronized, but it does not allow for true formation control. Also, the generation of smooth trajectories requires the manual input of spline parameters and can not be done in a user-friendly way (e.g. through RViz).

In this paper, we propose an approach to multi-robot formation path and trajectory planning that extends the idea of generating smooth and collision-free trajectories for multi-robot formations using splines. Similar to [12], the support points for all splines are taken from a global path to ensure easy planning and smooth paths. Selecting support points based on the formation geometry generates short paths while adding little computational overhead. After the path planning stage, the required target trajectories are generated based on the planned paths and user-defined formation dynamics.

III. Global path planner

In this section, we introduce our algorithm that computes a collision-free path for every robot in a multi-robot formation controlled by a leader-follower control. While the target pose of every following robot could be computed online from the leader pose, we compute the paths offline to check for collisions and ensure adherence to dynamic limits. To determine a path that allows the whole formation to traverse from a start to a target position without colliding with the environment, we modified the existing navigation stack for a single robot. The modification and the resulting three-stage process are briefly described in this section[1].

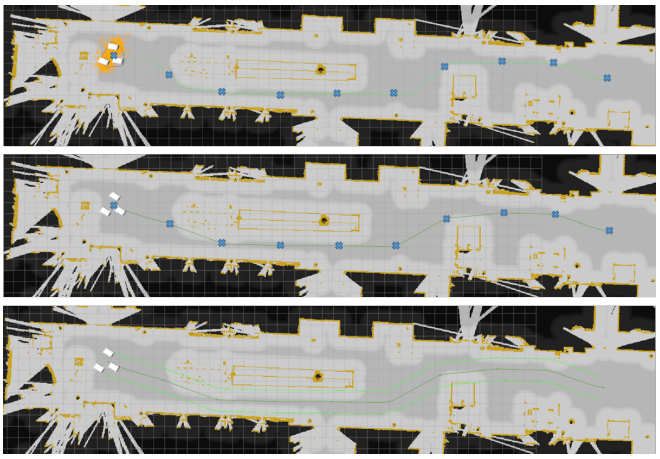

Fig. 1. Overview of the proposed spline-based planner in three stages. Obstacles are shown in orange; the global costmap is shown in light grey (the inflation radius had to be reduced due to the cramped environment). Stage 1 (top): The Relaxed A* path is plotted in light green. The positions of control points are marked with a blue cross. Stage 2 (middle): Interpolated Bezièr-spline in dark green. Stage 3 (bottom): formation trajectory target in dark green, following robot target trajectories in light green.

A. Stage 1: Initial global planning

The main objective in stage 1 is to calculate the initial global path for the whole formation (see Fig 1 top). To account for the increased size of the formation compared to a singular robot, we increase the costmap's inflation radius. The new radius is calculated based on the formation's footprint and equals the minimal enclosing circle around the center of rotation that includes all robots. With the inflated costmap, the path planner can treat the formation as a point, enabling navigation through the map using a standard path planning algorithm like Relaxed A*. We chose Relaxed A* for its efficient use of system resources and fast computation [13]. However, as Relaxed A* is based on a grid overlaying the map, it does not produce continuous curves. Instead, the orientation of the path changes abruptly either by 45° or

[1]for more information please refer to [2]

90°. This path causes an almost infinite angular acceleration and cannot be followed by a nonholonomic robot formation. Therefore, we use the initial path only to extract control points, which later support the generation of a smooth path (stage 2).

B. Stage 2: Formation Path Planning and smoothing

In this work, we use Beziér-splines to generate a continuous path. The main reason being that a Beziér-spline provides a smooth path and passes through the first and last of its control points. In this way, the spline can be controlled much more precisely, especially to avoid collisions. To fit a collision-free spline through a complex environment, a high number of control points is needed. These control points can be combined into a singular spline in two different ways. The first way is to interpolate n points with a spline of $(n-1)th$-degree. However, this is inefficient and can lead to numerical instability [14]. The second method utilizes multiple splines of a lower degree, which must be connected to create a complete interpolation.

To merge all splines seamlessly without creating a non-continuous transition, it is necessary to use at least quintic Beziér-Splines [15]. This spline type requires six control points ($\mathbf{p}_0 - \mathbf{p}_5$) for its analytic form, shown in (1).

$$\mathbf{s}(\lambda) = \begin{bmatrix} x(\lambda) \\ y(\lambda) \end{bmatrix} = \begin{bmatrix} (1-\lambda)^5 \\ 5(1-\lambda)^4\lambda \\ 10(1-\lambda)^3\lambda^2 \\ 10(1-\lambda)^2\lambda^3 \\ 5(1-\lambda)\lambda^4 \\ \lambda^5 \end{bmatrix}^T \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \\ \mathbf{p}_5 \end{bmatrix} \tag{1}$$

The function $\mathbf{s}(\lambda)$ defines a quintic Beziér-spline dependent on the iterator variable λ in the range of λ_0 to λ_1. The control points $\mathbf{p}_0$ and $\mathbf{p}_5$ define the points that the Beziér-Spline has to pass at the respective iterator variable. The remaining control points $\mathbf{p}_1$ to $\mathbf{p}_4$ define the first and second derivative of the spline at $s(\lambda_0)$ and $s(\lambda_1)$ and allow for an adjustment of the spline's curvature (see Fig. 2). The first step in the spline interpolation process is to select the first and last control point of each spline segment ($\mathbf{p}_0$,$\mathbf{p}_5$, ..., $\mathbf{p}_m$) through which the combined spline should pass exactly (the endpoint of spline i is the start point of spline $i+1$). We want to use as many control points as possible to maintain a similar path to the initial plan while not introducing oscillation due to too many control points. Additionally, we need a minimum distance between adjacent control points to limit the minimum curve radius, as we do not want the outer robots to move backwards in tight curves. To prevent the outer robots from performing a backwards motion in a curve, the formation's linear velocity $v_\mathrm{F}(\lambda)$ needs to be higher than its angular velocity $\omega_\mathrm{F}(\lambda)$ times the robot's distance from the formation's center of rotation $|\mathbf{d}_i| = |(d_{x,i}\,, d_{y,i})|$:

$$v_\mathrm{F}(\lambda) \overset{!}{>} |\vec{d}_i| \cdot \omega_\mathrm{F}(\lambda) \cdot \cos\left(\arctan\left(\frac{d_y}{d_x}\right)\right) \tag{2}$$

The linear and angular velocities are calculated from the formation's center position $\mathbf{x}_\mathrm{F}(\lambda) = (x_\mathrm{F}(\lambda), y_\mathrm{F}(\lambda), \theta_\mathrm{F}(\lambda))^T$:

$$\omega_\mathrm{F}(\lambda) = \frac{v_\mathrm{F}(\lambda)}{r} = \frac{\sqrt{x'_\mathrm{F}(\lambda)^2 + y'_\mathrm{F}(\lambda)}}{r} \tag{3}$$

With (2) and (3) we can calculate the limit for the minimum curve radius r_min in our interpolation:

$$r_\mathrm{min} \overset{!}{>} |\vec{d}_i| \cdot \cos\left(\arctan(\frac{d_y}{d_x})\right) \tag{4}$$

Given r_min we can define the maximum curvature κ_max and the minimum distance $d_{s,\mathrm{min}}$ between $\mathbf{p}_0$ and $\mathbf{p}_5$.

$$d_{s,\mathrm{min}} = 4 \cdot r_\mathrm{min} = 4 \cdot \frac{1}{\kappa_\mathrm{max}} \tag{5}$$

Choosing $d_{s,\mathrm{min}}$ according to (5) allows the interpolation to connect any two points with an S-shaped spline, without exceeding κ_max [2]. After choosing m control points with the distance $d_{s,\mathrm{min}}$ and hence defining $\mathbf{p}_0$ and $\mathbf{p}_5$, we calculate the control points $\mathbf{p}_1$ and $\mathbf{p}_4$. As visualized in Fig. 2, the direction of the spline is defined by the orthogonal tangent to the bisecting angle between the connecting line of three adjacent control points. The direction of the tangent at the start and the end position is defined by the formation's orientation at the start and target position respectively. Most importantly, the tangent vectors for two Beziér-Splines at the same control point (see $\mathbf{p}_5$ in Fig. 2) must be in the same direction and length to create a continuous path. We use an approach presented in [15] to optimize the position of $\mathbf{p}_1$ and $\mathbf{p}_4$. We refer the reader to [16] for a much more detailed description of the optimization process.

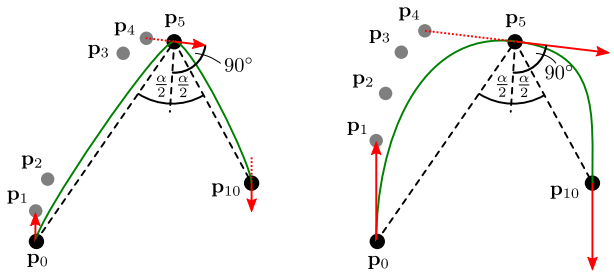

Fig. 2. Demonstration of the usage of two splines while optimizing the curvature of a Beziér-Spline through the control points $\mathbf{p}_1$-$\mathbf{p}_4$

Lau et al. [15] also describe the calculation of $\mathbf{p}_2$ and $\mathbf{p}_3$, which are determined by the second derivative of the Beziér-Spline at $\ddot{s}(\lambda_0)$ and $\ddot{s}(\lambda_1)$. With $\mathbf{p}_0, \ldots, \mathbf{p}_n$ we use (1) to compute the continuous functions $x(\lambda)$ and $y(\lambda)$ from which we can derive $\theta(\lambda)$.

$$\theta(\lambda) = \mathrm{atan}(\frac{y'(\lambda)}{x'(\lambda)}) \tag{6}$$

ROS navigation stack: As noted before, we want to include our algorithm in the ROS navigation stack (ROS-NS) to improve modularity and usability. This also means the algorithm has to adhere to the interfaces defined in the

navigation stack. For example, as ROS-NS requires a discrete global path, we convert the continuous $\mathbf{s}(\lambda)$ to a global path

$$\mathbf{X}_{\text{global}} = (\mathbf{x}_{\text{g},j}, \mathbf{x}_{\text{g},j+1}, \ldots, \mathbf{x}_{\text{g},n}) = \begin{pmatrix} x_{\text{g},j} & x_{\text{g},j+1} & \ldots & x_{\text{g},n} \\ y_{\text{g},j} & y_{\text{g},j+1} & \ldots & y_{\text{g},n} \\ \theta_{\text{g},j} & \theta_{\text{g},j+1} & \ldots & \theta_{\text{g},n} \end{pmatrix} \tag{7}$$

with $j = 0, 1, 2, \ldots, n$ discrete poses, where the number of points n equals the length of the spline divided by the user-defined distance l between two poses.

C. Stage 3: Formation Path Planning

In the third stage, we compute the path of each individual participating robot from the global path using a rigid body transformation. To compute the robot paths, we define a formation coordinate system CS_F located in the formation's center (see Fig. 3).

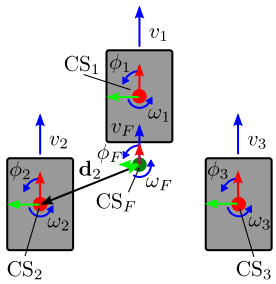

Fig. 3. Overview of a formation containing three mobile robots where each mobile robot is placed with a vector d_i relative to the formation coordinate system.

In the CS_F, every robot maintains a fixed position (semi-rigid formation):

$${}_{(F)}\vec{r}_i = const = (d_{x,i}, d_{y,i})^T, \quad i = 0, 1, 2, \ldots \tag{8}$$

By adjusting $d_{x,i}$,$d_{y,i}$, and ϕ_i the desired formation geometry can be set. For a given formation geometry, the path of robot i is calculated using (9).

$$\mathbf{x}_{i,j} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \left(\mathbf{x}_{\text{g},j} + R_z(\phi_{\text{g},j}) \cdot {}_{(F)}\vec{r}_i\right) + \begin{pmatrix} 0 \\ 0 \\ \arctan(\frac{\dot{y}_{\text{g},j}}{\dot{x}_{\text{g},j}}) \end{pmatrix} \tag{9}$$

Figure 4 shows an example of two following robots following the leader-robot ($i = 0$) in a parallel formation. In this example, the leader moves in the center of the formation resulting in $\text{CS}_0 = \text{CS}_\text{F}$ and $\mathbf{d}_0 = 0$.

IV. Trajectory Generation and Formation Control

The formation control is used to keep the formation on its given path and to maintain the distances between robots. The performance of this feedback controller can be improved by adding a feed-forward term $\hat{\mathbf{c}}_i$[2]. In this section, we first describe the generation of the target trajectory, which is later used to compute the feed-forward term and then introduce the formation controller.

[2]we denote target values with a circumflex in this paper

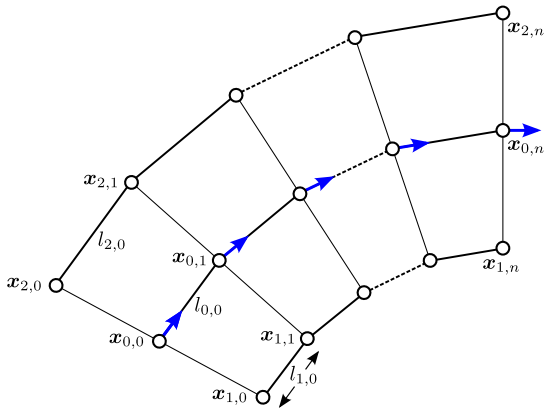

Fig. 4. Target path for a parallel formation of three robots moving in a curve. The equidistant leader path is computed using the aforementioned spline-based planner. The follower paths are derived from the leader path using (9).

A. Trajectory generation

The main purpose of trajectory generation is to convert the global path, which consists of n poses with a fixed distance into a trajectory. A trajectory $\mathbf{V}_i = (\mathbf{x}_{i,k}, \mathbf{x}_{i,k+1}, \ldots, \mathbf{x}_{i,m})^T$ is defined analog to (7), whereas the distance of the poses represents the target velocity for the trajectory. For a given trajectory $s_{i,k}$ denotes the distance along the trajectory at the index k. The distance $s_{i,k}$ is computed iteratively by applying a target velocity $\hat{v}_{i,k}$, acceleration $\hat{a}_{i,k}$, and jerk $\hat{j}_{i,k}$ between the individual points of a given global path:

$$\hat{s}_{i,k} = \hat{s}_{k-1,i} + \hat{v}_{i,k} \cdot \Delta k + \frac{\hat{a}_{i,k} \cdot \Delta k^2}{2} + \frac{\hat{j}_{i,k} \cdot \Delta k^3}{6}. \tag{10}$$

The time difference Δk between two indices is defined by the interpolation cycle time T of the formation controller. Given T we can also calculate the linear and angular velocities based on the discrete trajectory.

In the following section, we want to focus on the synchronization of all robot trajectories, as this is of particular interest for controlling semi-rigid multi-robot formations. Looking at Fig. 1 and Fig. 5 it is evident that the total length of each robots trajectory is slightly different depending on its position in the formation. If the distance $\Delta\hat{s}_{i,k}$ each robot moves in the current time step would be equal for all robots, the formation geometry could not be maintained in curves. Instead, $\Delta\hat{s}_{i,k}$ must be coordinated between all robots, meaning the inner robots should reduce their velocity in a curve. This coordination is achieved by normalizing $\Delta\hat{s}_{i,k,n}$ (11), where $l_{i,j}$ is the distance between the current and next point on the global path (see Fig. 4).

$$\Delta\hat{s}_{i,k,n} = \Delta\hat{s}_{i,k} \cdot \frac{l_{i,j}}{l_{j,\max}} \tag{11}$$

with

$$l_{i,j} = \|\mathbf{x}_{i,k+1} - \mathbf{x}_{i,k}\| \quad l_{j,\max} = \max\{l_{i,j}\,;\, i \in \mathbb{Z}\} \tag{12}$$

and

$$\hat{s}_{i,k+1} = \hat{s}_{i,k} + \Delta\hat{s}_{i,k,n} \tag{13}$$

By applying the same normalization from (11) to $\hat{v}_{i,k}$, $\hat{a}_{i,k}$, and $\hat{j}_{i,k}$, we get the target dynamics for the next waypoint. Given $\hat{s}_{i,k+1}$ and the global path $\mathbf{X}_{global}$, we calculate the new target pose $\hat{\mathbf{x}}_{i,k+1}$ as described in [17]. Using $\hat{\mathbf{x}}_{i,k+1}$, the linear target velocity $\hat{v}_{i,k}$ and angular target velocity $\hat{\omega}_{i,k}$ for each robot result to

$$\hat{\mathbf{c}}_i = \begin{pmatrix} \hat{v}_{i,k} \\ \hat{\omega}_{i,k} \end{pmatrix} = \begin{pmatrix} \sqrt{(\hat{x}_{i,k+1} - \hat{x}_{i,k})^2 + (\hat{y}_{i,k+1} - \hat{y}_{i,k})^2} \\ \frac{d\,\mathrm{atan2}(\hat{y}_{i,k+1} - \hat{y}_{i,k}, \hat{x}_{i,k+1} - \hat{x}_{i,k})}{dt} \end{pmatrix}. \quad (14)$$

The last step of each iteration is to check if $\hat{\omega}_{i,k}$ exceeds the angular velocity limit $\omega_{\max}$ or angular acceleration limit $\dot{\omega}_{\max}$ (15). If any robot exceeds the limits, $\Delta\hat{s}_{i,k,n}$ is reduced proportionally for all robots to maintain the formation geometry.

$$\Delta\hat{s}_{i,k,n} := \begin{cases} \Delta\hat{s}_{i,k,n} & \text{for } \hat{\omega}_{i,k} < \omega_{\max} \vee \hat{\dot{\omega}}_{i,k} < \dot{\omega}_{\max} \\ \Delta\hat{s}_{i,k,n} \cdot \frac{\hat{\omega}_{i,k}}{\omega_{\max}} & \text{for } \hat{\omega}_{i,k} > \omega_{\max} \vee \hat{\dot{\omega}}_{i,k} < \dot{\omega}_{\max} \\ \Delta\hat{s}_{i,k,n} \cdot \frac{\hat{\omega}_{i,k}}{\dot{\omega}_{\max}} & \text{for } \hat{\omega}_{i,k} < \omega_{\max} \vee \hat{\dot{\omega}}_{i,k} > \dot{\omega}_{\max} \\ \Delta\hat{s}_{i,k,n} \cdot \frac{\hat{\omega}_{i,k}}{\omega_{\max}} \cdot \frac{\hat{\dot{\omega}}_{i,k}}{\dot{\omega}_{\max}} & \text{for } \hat{\omega}_{i,k} > \omega_{\max} \vee \hat{\dot{\omega}}_{i,k} > \dot{\omega}_{\max} \end{cases} \quad (15)$$

The final result of this process can be seen in Fig. 6, where all robots move at the desired velocity on straights and reduce their velocity in curves depending on their position in the formation and the curve radius.

B. Formation controller

The basic control law we are using in this work is derived from [18] and slightly modified to account for a predefined target trajectory. The control law (17; 18) calculates the controller output for every following robot ($v_{i,c}$, $\omega_{i,c}$) based on the control error of the leading robot $\mathbf{e_0}$.

$$\mathbf{e_0} = \begin{pmatrix} e_{0,x} \\ e_{0,y} \\ e_{0,\theta} \end{pmatrix} = \begin{pmatrix} \hat{x}_0 - x_0 \\ \hat{y}_0 - y_0 \\ \hat{\theta}_0 - \theta_0 \end{pmatrix} \quad (16)$$

$$v_{i,c} = \hat{v}_{i,k} \cdot \cos(\hat{\theta}_i - \theta_i) + K_{p,x} \cdot (\hat{x}_{i,k} - x_{i,k} + e_{0,x}) \quad (17)$$

$$\omega_{i,c} = \hat{\omega}_{i,k} + \hat{v}_{i,k} \cdot \big(K_{p,y} \cdot (\hat{y}_{i,k} - y_{i,k} + e_{0,y}) + K_{p,\theta} \cdot \sin(\hat{\theta}_{i,k} - \theta_{i,k} + e_{0,\theta})\big) \quad (18)$$

A more detailed description and evaluation of the presented controller can be found in [10] and [19].

V. Evaluation

The goal of this section is to evaluate the developed algorithm based on the previously defined objectives, starting with the trajectory smoothing and compliance to dynamic limits. For the evaluation, we used the triangular formation and 45 m long path shown in Fig. 1 and Fig 5. A video of the experiments can be found at *bit.ly/3xmnGpi*. We set the controller parameters and dynamic limits according to Table I. Figure 6 shows the linear and angular velocities of the virtual leader (robot 0) and the two physical follower robots. Comparing the measurements to the limits from Table I it can be seen, that the formation controller maintains the desired linear velocity until the maximum angular velocity or acceleration is reached with only slight overshoots due to a lower lever controller. The figure also shows, that the linear error $e_{i,l}$ (19) converges to around 5-10 cm after an initial spike caused by a misalignment in the starting orientations (see Fig. 5).

TABLE I
Controller parameterization

Parameter	Value	Unit
$d_{x,1}$; $d_{y,1}$; $d_{\theta,1}$	1; 0.75; 0	m; m; rad
$d_{x,2}$; $d_{y,2}$; $d_{\theta,2}$	0; 1.5; 0	m; m; rad
$K_{p,x}$, $K_{p,y}$, $K_{p,\theta}$	0.1, 0.3, 0.25	
$\hat{v}_i$	0.3	$\mathrm{m\,s^{-1}}$
$\hat{\dot{v}}_i$	1	$\mathrm{m\,s^{-2}}$
ω_{max}	1	$\mathrm{rad\,s^{-1}}$
$\dot{\omega}_{max}$	1	$\mathrm{rad\,s^{-2}}$
T	0.01	s

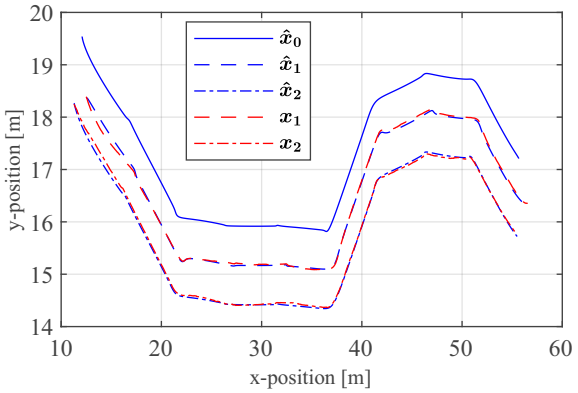

Fig. 5. Target and actual poses for robots 1 and 2 as well as the target pose for the virtual leader.

$$e_{i,l} = \sqrt{e_{i,x}^2 + e_{i,y}^2} \quad (19)$$

More importantly, the measurements show no visible impact of the angular velocity/acceleration on the linear error for a smoothed trajectory in great contrast to previous measurements using unsmooth trajectories [10]. For this reason, a significantly higher formation speed can be achieved without reducing the formation control quality at the same time.

VI. Summary and Outlook

We have presented a planning algorithm that calculates collision-free trajectories for semi-rigid multi-robot formations while maintaining dynamic constraints. Our approach first plans an unsmooth collision-free global path and then smoothes the curvature of the path using splines to allow nonholonomic mobile robots to traverse along the path without reversing. We previously successfully simulated the global path planning algorithm in various environments with different formations [2]. In this work, we added a formation controller as well as trajectory generation and included

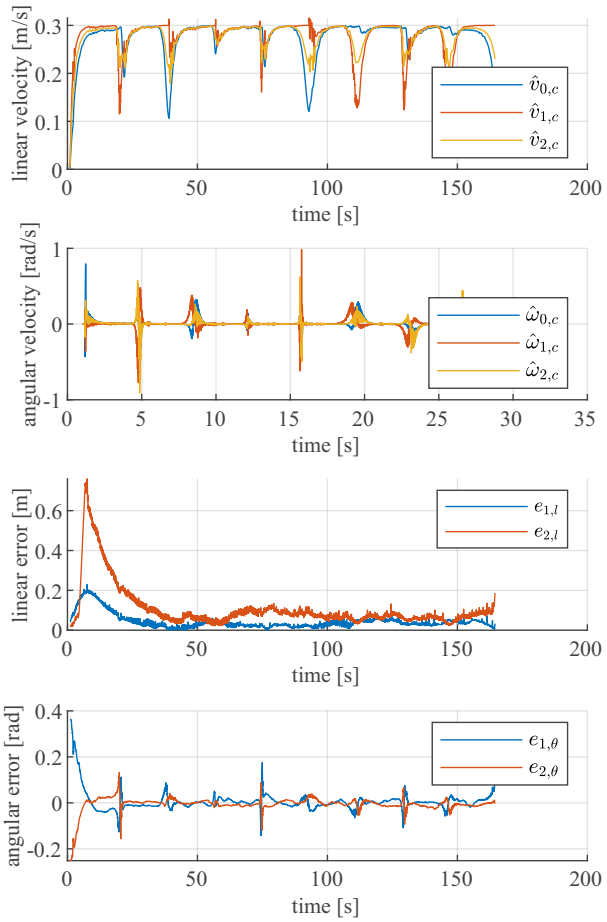

Fig. 6. Linear and angular velocities for a triangular formation of three robots. Target linear velocity: $0.3\frac{m}{s}$; angular velocity limit: $1\frac{rad}{s}$

Fig. 7. Triangular formation of three MiR industrial robots. The three robots from left to rigth are: virtual leader, robot 1, robot 2

further experiments using two industrial mobile robots following a third virtual robot. The experiments showed that, given smooth trajectories, even a simple formation controller can compensate for disturbances (wrong starting orientation) and maintain the target velocity for the formation. Given the good control performance the curvature of the initial global path and distance to obstacles becomes the limiting factor for the linear velocity. As the Relaxed A* plans the global path as close to obstacles as possible to optimize the path in terms of length, it tends to create a path with sharp curves and close proximity to obstacles. In future work we plan to investigate other global planners (e.g. Voronoi-based planners) that prioritises the distance to obstacles and create a smoother initial path.

References

[1] S. Kloder and S. Hutchinson, Path planning for permutation-invariant multirobot formations, in IEEE Transactions on Robotics, vol. 22, no. 4, 2006, pp. 650-665.

[2] H. Lurz, T. Recker, A. Raatz, Spline-based Path Planning and Reconfiguration for Rigid Multi-Robot Formations, Procedia CIRP, vol. 106, 2022, pp. 174-179.

[3] Y. Q. Chen, Z. Wang, Formation control: A review and a new consideration, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 3181-3186.

[4] J. R. Sánchez-Ibáñez, C. Perez-del-Pulgar, A. Garcia, Path Planning for Autonomous Mobile Robots: A Review, Sensors 21(23), 7898, 2021, pp. 1-29.

[5] A. N. Asl, M. B. Menhaj, A. Sajedin, Control of leader–follower formation and path planning of mobile robots using Asexual Reproduction Optimization (ARO), Applied Soft Computing,vol 14, 2014, pp. 563-576.

[6] J. Alonso-Mora, S. Baker, D. Rus, Multi-robot formation control and object transport in dynamic environments via constrained optimization, The International Journal of Robotics Research, 2017, pp. 1000-1021.

[7] T. D. Barfoot, C. M. Clark, Motion planning for formations of mobile robots, Robotics and Autonomous Systems 46, 2004, pp. 65-78.

[8] M.A. Lewis, K.H. Tan, High Precision Formation Control of Mobile Robots Using Virtual Structures, Autonomous Robots 4, 1997, pp. 387–403.

[9] Y. Wang, M. Shan and D. Wang, A Comparative Analysis on Rigid and Flexible Formations of Multiple Differential-Drive Mobile Robots: A Motion Capability Perspective, 18th European Control Conference (ECC), 2019, pp. 2077-2082.

[10] T. Recker, M. Heinrich, Aa Raatz, A Comparison of Different Approaches for Formation Control of Nonholonomic Mobile Robots regarding Object Transport, Procedia CIRP, vol. 96,2021, pp. 248-253.

[11] Y. Hao, B. Laxton, S. K. Agrawal, E. Lee, E. Benson, Planning and control of UGV formations in a dynamic environment: a practical framework with experiments, IEEE International Conference on Robotics and Automation, 2003, pp. 1209-1214.

[12] H. Ali, G. Xiong, H. Wu, B. Hu, Z. Shen and H. Bai, Multi-robot Path Planning and Trajectory Smoothing, IEEE 16th International Conference on Automation Science and Engineering (CASE), 2020, pp. 685-690.

[13] A. Ammar, H. Bennaceur, I. Chaari, A. Koubaa, M. Alajlan, Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments, Soft Computing 20(10), 2015, pp. 1-23.

[14] L. Piegl, W. Tiller, The NURBS book (2nd ed.), Springer, 1997.

[15] Lau, Boris & Sprunk, Christoph & Burgard, Wolfram. (2009). Kinodynamic Motion Planning for Mobile Robots Using Splines. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009. 2427-2433. 10.1109/IROS.2009.5354805.

[16] C. Sprunk, Planning Motion Trajectories for Mobile Robots Using Splines, Technical Report, 2008, pp. 1-110

[17] Tobias Kunz and Mike Stilman, Turning Paths Into Trajectories Using Parabolic Blends, Technical Report, 2011, pp. 1-4

[18] Y. Kanayama, Y. Kimura, F. Miyazaki and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," Proceedings., IEEE International Conference on Robotics and Automation, 1990, pp. 384-389 vol.1, doi: 10.1109/ROBOT.1990.126006.

[19] T. Recker, M. Heinrich and A. Raatz, "A Hybrid Control Approach on Handling Orientation Constraints and Tracking Errors in Formation Control for Multiple Nonholonomic Mobile Manipulators," 2021 20th International Conference on Advanced Robotics (ICAR), 2021, pp. 891-896, doi: 10.1109/ICAR53236.2021.9659315.