2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Viewpoint and Trajectory Optimization for Animation Display with Aerial Vehicles

Marcel Schoch[1], Javier Alonso-Mora[1,2], Roland Siegwart[1] and Paul Beardsley[2]

Abstract— **This paper presents a method to optimize the position and trajectory of each aerial vehicle within a large group that displays objects and animations in 3D space. The input is a single object or an animation created by an artist. In a first step, goal positions for the given number of vehicles and representing the object are optimized with respect to a known viewpoint. For displaying an animation, an optimal trajectory satisfying the dynamic constraints of each vehicle is computed using B-splines. Finally, a trajectory following controller is described, which provides the preferred velocity, later optimized to be collision-free with respect to all neighboring vehicles.**

I. Introduction

The last couple of years have seen great advancements in small unmanned aerial vehicles, particularly with quadrotor systems. Following our previous work to display objects and animations using a large group of aerial vehicles [1] where a volumetric representation was used, in this paper we present optimized methods to take into account the viewpoint in the goal generation step, compute optimized trajectories for animation display that take into account dynamic constraints, and a trajectory tracking controller that provides improved performance. Since a proof of concept of the original method with real vehicles was provided in [1], the analysis of the novel algorithms in this paper is limited to simulation, but could be directly applied in a system such as the one in [2].

Several approaches for controlling a group or aerial robots rely on a central optimization algorithm which may not scale well for large swarms. A testbed for multiple micro quadrotors was described in [3], and [4] presented a Mixed Integer optimization method to create collision-free trajectories for a set of quadrotors. In [5], an approach for choreography of quadrotors with music was presented.

The mentioned papers rely on path calculation where collision avoidance is guaranteed beforehand. This problem is computationally very expensive for a large group of robots. Our approach, on the contrary, does not precompute collision-free trajectories but relies on a real-time collision avoidance algorithm based on the Reciprocal Velocity Obstacles [6]. In this work an extension to 3D accounting for the dynamics of the vehicles [7] is used.

The remainder of the paper is structured as follows: Section II provides an overview of the system. Section III introduces the optimization of the positions given a viewpoint. Section IV describes the trajectory optimization and Section V presents the trajectory tracking controller providing the preferred velocity to the collision avoidance algorithm. Section VI contains experimental results and discussions. Finally, Section VII concludes the paper.

II. System overview

The approach presented in this paper builds on our previous work [1] on object and animation display in 3D with a large team of aerial vehicles.

Given n robots of radii $r_1, ..., r_n$ and height $h_1, ..., h_n$ and an input object or animation, the method is divided into several steps. First, the goal positions of all robots are computed and optimized for a certain viewpoint. Second, an optimization over the whole animation is performed to obtain optimal trajectories, given by B-splines, that satisfy the velocity and acceleration constraints of the robots. This two steps can be precomputed. Third, in each time-step, a preferred velocity is obtained given by a real-time controller then enables synchronous trajectory following. Finally, collision-free inputs are obtained following [7]. This allows for real-time adaptation to disturbances.

Regarding goal generation (calculation of the goal positions that represent the object), a method based on optimal 3D coverage and relying on the K-means algorithm was presented in [1]. In that work goal positions were computed such that an optimal volumetric representation was obtained. However, in such a display, spectators may be located at a specified location. This viewpoint is taken into account in this paper to optimize the distribution of the goal positions.

Synchronous and smooth movement of the robots was shown to be crucial for good visual representation during an animation. In previous work, robots transitioned between subsequent goal positions (computed to optimally represent subsequent frames of the animation) via a position controller. In this work, for each robot, a B-spline is computed that interpolates its goal positions through-out the animation whilst satisfying velocity and acceleration constraints. This is formulated as a high dimensional quadratic optimization.

To display the animation, each robot is initially assigned to a trajectory (starting position) minimizing the total travelled squared distance [1]. Then, in each time step, a preferred velocity is calculated that provides successful tracking of the assigned trajectory. Finally, a distributed reciprocal collision avoidance algorithm computes a collision-free velocity for each robot that takes into account the current positions and velocities of the neighboring robots, as well as the dynamic constraints [7]. This very last step lays outside of the scope of this paper.

[1]M. Schoch, J. Alonso-Mora and R. Siegwart are with the Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland {mschoch, jalonso, rsiegwart}@ethz.ch
[2]J. Alonso-Mora and P. Beardsley are with Disney Research Zurich, Switzerland {jalonso, pab}@disneyresearch.com

III. Viewpoint-dependant Position Optimization

An object is given by a set of points $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$, $\mathbf{x}_i \in \mathbb{R}^3$, which form its volumetric representation (see [1] for details). First, an initial goal set $\mathcal{G}_0 = \{\mathbf{g}_{0,1}, ..., \mathbf{g}_{0,n}\}$, $\mathbf{g}_{0,i} \in \mathbb{R}^3$ that provides the centroidal Voronoi tessellation of the original set $\mathcal{X}$ is computed via the k-means algorithm [1]. Given the viewpoint $\mathbf{v}$, the goal positions are optimized by first projecting them onto a plane perpendicular to the vector pointing from the viewpoint to the object centre, followed by an iterative optimization (adjusted k-means algorithm) and finally projected back to 3D to their original depth.

A. Projection into image plane

The object center is denoted by $\bar{\mathbf{x}} = \frac{\sum_i^m \mathbf{x}_i}{m}$. Both the object points $\mathcal{X}$ and the initial goal set $\mathcal{G}_0$ are projected on a plane passing through the point $\bar{\mathbf{x}}$ and with normal vector $\mathbf{n} = \frac{\bar{\mathbf{x}} - \mathbf{v}}{\|\bar{\mathbf{x}} - \mathbf{v}\|}$. The projected sets are denoted by $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_0, ..., \bar{\mathbf{x}}_m\}$, $\bar{\mathbf{x}}_i \in \mathbb{R}^2$ and $\bar{\mathcal{G}}_0 = \{\bar{\mathbf{g}}_{0,1}, ..., \bar{\mathbf{g}}_{0,n}\}$, $\bar{\mathbf{g}}_{0,i} \in \mathbb{R}^2$.

B. Optimized goal set

The method to optimize the goal set with respect to the viewpoint is an extension of the k-means algorithm. The point set is partitioned minimizing the distance to the cluster center, but the update step for the cluster centers is slightly modified. The cluster centers (projected goal positions) $\bar{\mathbf{g}}_i \in \mathbb{R}^2$, $1 \le i \le n$ are obtained by minimizing the following adjusted within-cluster sum of squares

$$C = \sum_{i=1}^{n} \left[\sum_{x_j \in \bar{\mathcal{X}}_i} \|\bar{\mathbf{x}}_j - \bar{\mathbf{g}}_i\|^2 + c\|\bar{\mathbf{g}}_{0,i} - \bar{\mathbf{g}}_i\|^2 \right], \quad (1)$$

where $\bigcup \bar{\mathcal{X}}_i$ is a partition of $\bar{\mathcal{X}}$ with each cluster given by

$$\bar{\mathcal{X}}_i = \{\bar{\mathbf{x}}_m \in \bar{\mathcal{X}} : \|\bar{\mathbf{x}}_m - \bar{\mathbf{g}}_i\| \le \|\bar{\mathbf{x}}_m - \bar{\mathbf{g}}_j\|, j \in [1, n], j \ne i\}. \quad (2)$$

The additional term in Eq. (1) with constant factor c determines how much the new cluster center $\bar{\mathbf{g}}_i$ deviates from the initial center $\bar{\mathbf{g}}_{0,i}$ and provides a tradeoff between the optimal distribution in 3D and the optimal distribution on the 2D manifold.

This optimization is solved via a two step algorithm with initial cluster centers $\bar{\mathbf{g}}_i = \bar{\mathbf{g}}_{0,i}$.

- **Assignment step:** Create new clusters from Eq. (2).
- **Update step:** Calculate new centers given by

$$\bar{\mathbf{g}}_i = \frac{1}{|\bar{\mathcal{X}}_i| + c} \left[\sum_{\bar{\mathbf{x}}_j \in \bar{\mathcal{X}}_i} \bar{\mathbf{x}}_j + c\bar{\mathbf{g}}_{0,i} \right]. \quad (3)$$

Both steps minimize the cost function until a local minimum is reached. A sketch of the proof is given in the Appendix. The factor c determines how much the final solution deviates from the initialization:

- $c = 0$: optimal coverage of the 2D projected set $\bar{\mathcal{X}}$.
- $c \to \infty$: optimal coverage of the original 3D set $\mathcal{X}$.

$$\bar{\mathbf{g}}_i = \lim_{c \to \infty} \left(\frac{1}{\|\bar{\mathcal{X}}_i\| + c} \left[\sum_{\bar{\mathbf{x}}_j \in \bar{\mathcal{X}}_i} \bar{\mathbf{x}}_j + c\bar{\mathbf{g}}_{0,i} \right] \right) = \bar{\mathbf{g}}_{0,i}$$

- $0 < c < \infty$: The minimum lies between the initialization (optimal 3D coverage) and the 2D optimal coverage.

As the influence of c strongly depends on the size of $\bar{\mathcal{X}}$, it is suggested to redefine the constant as: $c = \hat{c} \cdot \tilde{c}$ with $\tilde{c} = \frac{|\bar{\mathcal{X}}|}{n}$, so $\tilde{c}$ equals the average cluster size if the points are distributed equally. The constant $\hat{c}$ is defined as a positive number, $0 < \hat{c} < \infty$, but does not depend on the the cluster size anymore. We observed that $c \approx 0.5$ yields good results. The choice $c = 0$ might not be too favorable as the goal positions are optimized such that the projected view results in an optimal 2D coverage and could potentially falsify the 3D impression of the shape.

C. Projection back to 3D

After convergence, the optimized projected goal positions $\bar{\mathbf{g}}_i$ are projected back into 3D to their original depth along the normal vector $\mathbf{n}$ by $\mathbf{n}^T(\mathbf{g}_{0,i} - \bar{\mathbf{x}})$, resulting in the goal positions $\mathbf{g}_i \in \mathbb{R}^3$, $1 \le i \le n$. The back projection does not affect the view from the specified viewpoint and maintains the depth constant with respect to the initial goal positions.

D. Evaluation

An example of the optimization is shown in Fig. 1. The stronger the optimization for a viewpoint, the more it deviates from it's initialized distribution (optimal volumetric representation). A second example shows a human in Fig. 2. The influence on the extremities is minimal, as they are long and thin volumes; however, a slight improvement in the recognition of the torso is observed after view-point optimization. In general, only the parts with high volume of an object are significantly affected by the optimization, as in those parts the goal positions potentially cover each other in the 2D view.

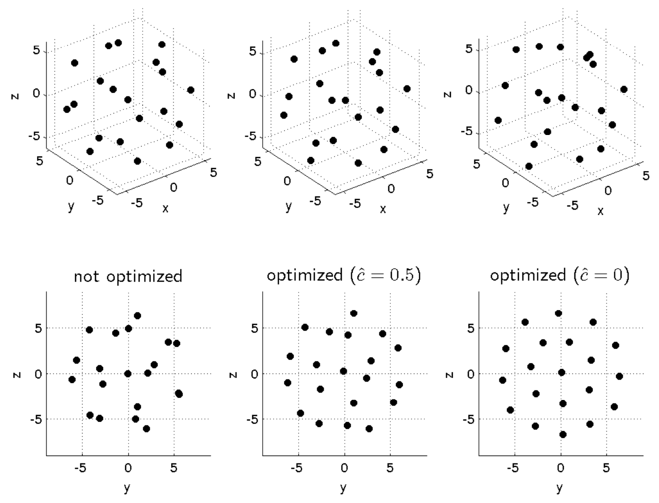

Fig. 1. Comparison of 20 goal positions displaying a sphere with optimized view along the x-axis. Left: initial goal positions (3D optimal), middle: optimized with $\hat{c} = 0.5$, right: optimized with $\hat{c} = 0$ (2D optimal). The 3D view (top row) shows that the view from any viewpoint other than the one used for the optimization is not greatly affected.

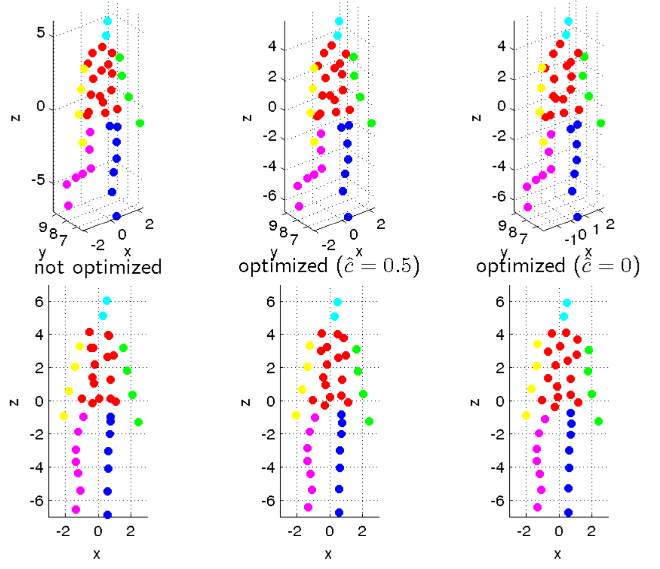

Fig. 2. Comparison of 40 goal positions displaying a human with optimized view along the y-axis. Left: initial goal positions, middle: optimized with $\hat{c} = 0.5$, right: optimized with $\hat{c} = 0$. The 3D view (top row) shows that the view from any viewpoint other than the one used for the optimization is not greatly affected. The effect of the view-point optimization is most visible in the wide areas.

IV. B-Spline Trajectory Optimization

In the previous step, goal positions are obtained for every frame of the animation. The trajectory for each robot is defined by goal positions in consecutive frames. Depending on the animation, the trajectory might contain sharp turns with accelerations that are out of reach for the quadrotor. The trajectory is optimized to account for the additional constraints due to dynamic limitations.

The trajectory is given by a B-spline due to its advantageous characteristics, such as the local influence of changes. B-splines do not visit each point exactly but the control points are used to shape the B-spline. This is a favorable characteristic as it avoids oscillating behaviours. For a general introduction to splines, refer to [8].

For the optimization, the special case of uniform cubic B-splines is used due to the following reasons:

- easily differentiable
- first derivative (velocity) continuous and differentiable
- second derivative (acceleration) continuous but not differentiable
- min/max of second derivative at $t = 0$ and $t = 1$

A uniform cubic B-spline segment is defined by:

$$S_i(t) = \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}' \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_i \\ x_{i+1} \\ x_{i+2} \\ x_{i+3} \end{bmatrix} = TM\mathbf{x}_i, \tag{4}$$

for $t \in [0, 1]$, and the trajectory (spline) consists of the consecutive connection of segments, $S = \{S_0, S_1, ...\}$. The B-spline is defined for each dimension x, y, z separately.

The velocity and the acceleration when following the trajectory are calculated by taking the first and second derivative of the B-spline: the velocity is $\frac{d}{dt}S_i = T_1 M\mathbf{x}_i$ with $T_1 = \frac{dT}{dt}$ and the acceleration $\frac{d^2}{dt^2}S_i = T_2 M\mathbf{x}_i$ with $T_2 = \frac{d^2T}{dt^2}$.

To account for the quadrotor limitations, two B-splines S and $\tilde{S}$ are defined, where S is the unconstrained one and $\tilde{S}$ the constrained one with new control points

$$\tilde{\mathbf{x}}_i = \begin{bmatrix} x_i \\ x_{i+1} \\ x_{i+2} \\ x_{i+3} \end{bmatrix} + \begin{bmatrix} \Delta x_i \\ \Delta x_{i+1} \\ \Delta x_{i+2} \\ \Delta x_{i+3} \end{bmatrix} = \mathbf{x}_i + \Delta\mathbf{x}_i \tag{5}$$

The problem is formulated as a quadratic optimization in $\Delta\mathbf{x} = \bigcup \Delta\mathbf{x}_i$, where the distance of the constrained B-spline to the unconstrained B-spline is minimized subject to linear and quadratic constraints.

For ease of exposition, the optimization is first shown for a single segment and the synthesis for the whole trajectory is described at the end of this section.

A. *Time scaling*

The goal positions of each frame have a timestamp. It is assumed that the timestamps between the frames are regular and fixed and given by $\Delta\tilde{t}$. This is the case for an animation. The time of a B-spline segment is defined to be $\Delta t = 1$. To account for this difference, the velocity and acceleration have to be transformed from the real to the B-spline time with $\beta = \frac{\Delta t}{\Delta \tilde{t}} = \frac{1}{\Delta \tilde{t}}$

$$t = \beta\tilde{t}, \qquad v = \frac{1}{\beta}\tilde{v}, \qquad a = \frac{1}{\beta^2}\tilde{a} \tag{6}$$

B. *Minimization term*

As an abbreviation, the subscript d is used to denote the dimension (x, y and z). The term to be minimized is the difference f_i between two trajectory segments

$$\begin{aligned} f_i &= \int_0^1 \sum_d \left(S_{d,i} - \tilde{S}_{d,i}\right)^2 dt \\ &= \int_0^1 \sum_d \left(TM\mathbf{x}_{d,i} - TM\tilde{\mathbf{x}}_{d,i}\right)^2 dt \\ &= \int_0^1 \sum_d \left(TM\Delta\mathbf{x}_{x,i}\right)^T \cdot \left(TM\Delta\mathbf{x}_{x,i}\right) dt \\ &= \sum_d \Delta\mathbf{x}_{d,i}^T M^T \left[\int_0^1 T^T T dt\right] M\Delta\mathbf{x}_{d,i} \end{aligned} \tag{7}$$

And substituting $T_{int} = \int_0^1 T^T T dt$,

$$f_i = \sum_d \Delta\mathbf{x}_{d,i}^T M^T T_{int} M\Delta\mathbf{x}_{d,i} \tag{8}$$

C. *Acceleration constraints*

The minimum / maximum acceleration is found at $t = 0$ or $t = 1$ due to the linearity of each segment. Furthermore, due to the continuity of the acceleration, $A_i(1) = A_{i+1}(0)$, therefore it is sufficient to analyze only $A_i(0)$. For $A_i(0) \leq a_{max}$ the constraint is given by

$$\begin{aligned} T_2(0)M_2 \cdot \tilde{\mathbf{x}}_{d,i} &\leq a_{max} \\ T_2(0)M_2\Delta\mathbf{x}_{d,i} &\leq a_{max} - T_2(0)M_2\mathbf{x}_{d,i} \end{aligned} \tag{9}$$

And analogously for $A_i(0) \geq -a_{max}$

$$-T_2(0)M_2\Delta\mathbf{x}_{d,i} \leq a_{max} + T_2(0)M_2\mathbf{x}_{d,i} \tag{10}$$

Therefore, for each dimension and segment, two linear constraints are obtained.

D. Velocity constraints

The velocity constraints can not be added continuously as it would lead to quartic constraints. Therefore, the velocity is limited at k intermediate points of the interval $t = [0, 1]$ and given by $t = \frac{i}{k}$, $i = [0, ..., k-1]$, $k \in \mathbb{N}^+$.

$$\begin{aligned} V_i(t)^2 \leq v_{max} \quad \Rightarrow \quad \sum_d V_{d,i}(t)^2 &\leq v_{max}^2 \\ \sum_d (T_1 M \tilde{\mathbf{x}}_{d,i})^2 &\leq v_{max}^2 \\ \sum_d \tilde{\mathbf{x}}_{d,i}^T M^T T_1^T T_1 M \tilde{\mathbf{x}}_{d,i}^T &\leq v_{max}^2 \\ \sum_d (\mathbf{x}_{d,i}^T + \Delta\mathbf{x}_{d,i}^T)\hat{Q}(\mathbf{x}_{d,i}^T + \Delta\mathbf{x}_{d,i}^T) &\leq v_{max}^2 \end{aligned} \tag{11}$$

With $\hat{Q} = M^T T_1^T T_1 M$ and therefore $\hat{Q}^T = \hat{Q}$, this term can be expanded and rearranged:

$$\sum_d \left[2\mathbf{x}_{d,i}^T \hat{Q} \Delta\mathbf{x}_{d,i} + \Delta\mathbf{x}_{d,i}^T \hat{Q} \Delta\mathbf{x}_{d,i} \right] \leq v_{max}^2 - \sum_d \mathbf{x}_{d,i}^T \hat{Q} \mathbf{x}_{d,i}$$

Note that $\hat{Q}$ depends on t and therefore, k quadratic constraints are obtained for each segment.

E. Synthesis for the whole trajectory

A detailed formulation is left to the reader in the sake of space, but it can be obtained by concatenation from the above formulation for each single segment. Given the constraints and terms for each single segment, a trajectory with $(n+1)$ control points and d dimensions is formed by $(n-2)$ segments (with 4 control points for each single segment), leading to:

- Minimization terms: 1
- Linear constraints: $2d(n-2)+2$
- Quadratic constrains: $k(n-2)+1$

Note that the minimization of the difference between the two trajectories is done over the whole trajectory, therefore only one minimization term (composed by the cost of all segments) is present, whereas the linear and the quadratic constraints are formulated separately for each segment.

F. Evaluation

An example is shown in Fig. 3. The initial trajectory is given at regular intervals, with constant speed on each segment and with instantaneous changes in direction. The constrained trajectories do not present constant velocity anymore since the control points are shifted in space but not in time. This characteristic is advantageous as the timestamps are crucial for synchronous movement of the robots.

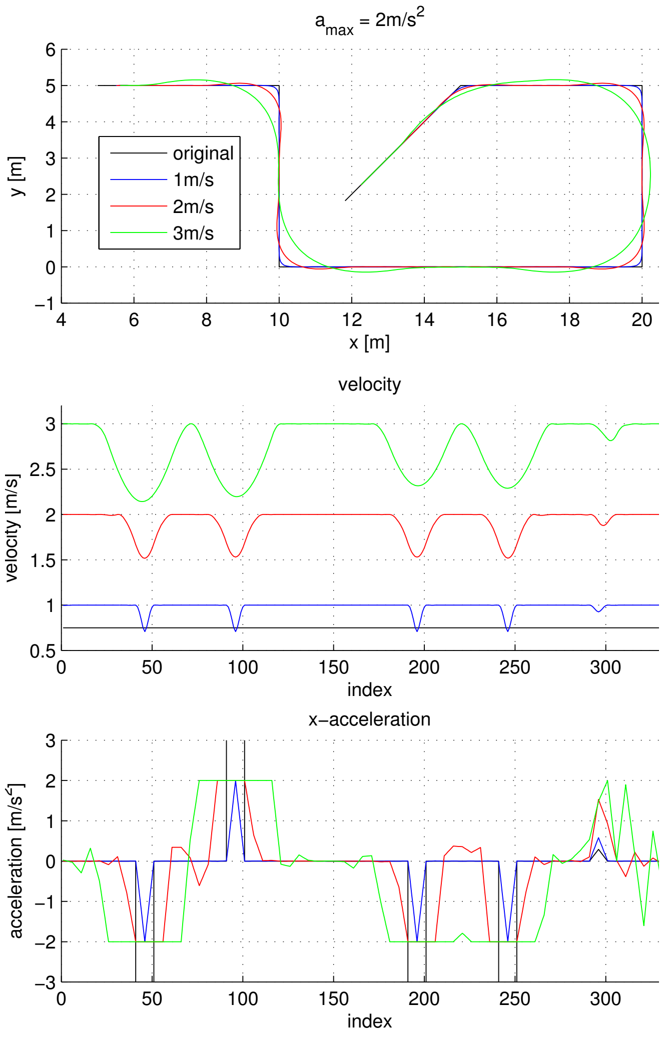

Fig. 3. Comparison of trajectory optimization for three different velocities and a maximum acceleration $a_{max} = 2\frac{m}{s^2}$. Top: trajectories, middle: linear velocity, bottom: acceleration in x dimension.

V. Velocity Control for Trajectory Following

The objective is to control n robots synchronously along the defined trajectories. It is important that the robots follow the trajectory closely, in order to keep the desired visual impression defined by the animation. Although the trajectories from the previous section are given as a continuous representation, for the reminder of this section, a discretized version is used to simplify the equations, however if the discretization is fine enough, the difference is negligible.

The approach for trajectory control in this paper is an adapted version of [9], but adjusted to suit the multi aerial vehicle framework with real time collision avoidance. Given a path $\mathcal{P} = [\mathbf{x}_0, ..., \mathbf{x}_m]$, $\mathbf{x}_i \in \mathbb{R}^3$ and the quadrotor position $\mathbf{x}(t)$, the two closest points on the trajectory $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are computed, see Fig. 4. $\mathbf{t}_i$ is the unit tangent vector in direction of the path from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$ and $\mathbf{n}_i$ is the unit vector perpendicular on $\mathbf{t}_i$ such that the plane defined by $\mathbf{t}_i$ and $\mathbf{n}_i$ contains $\mathbf{x}(t)$. Furthermore, a point $\mathbf{x}_j$ further ahead

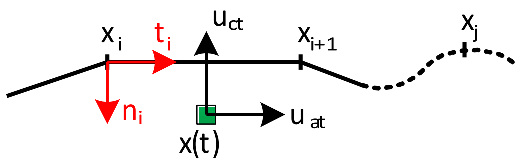

Fig. 4. Schema of trajectory following controller. $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are the closest waypoints, $\mathbf{x}_j$ is a waypoint ahead of the trajectory. u_{at} and u_{ct} show the along track and cross track velocity input.

on the trajectory is used to keep the quadrotors synchronized (chosen at identical time for all quadrotors). The following error terms are computed

$$\begin{aligned} \text{along track error: } e_{at} &= (\mathbf{x}_j - \mathbf{x}(t)) \cdot \mathbf{t}_i \\ \text{along track error rate: } \dot{e}_{at} &= v_i - \dot{\mathbf{x}}(t) \cdot \mathbf{t}_i \\ \text{cross track error: } e_{ct} &= (\mathbf{x}_i - \mathbf{x}(t)) \cdot \mathbf{n}_i \\ \text{cross track error rate: } \dot{e}_{ct} &= -\dot{\mathbf{x}}(t) \cdot \mathbf{n}_i \end{aligned} \tag{12}$$

The along track error denotes the error towards $\mathbf{x}_j$ ahead of the trajectory whereas the other three error terms are related to the closest point on the trajectory.

The preferred along track and cross track velocities u_{at} and u_{ct} are calculated using standard PD control (with K_* design constants),

$$\begin{aligned} u_{at} &= K_{atp} \cdot e_{at} + K_{atd} \cdot \dot{e}_{at} \\ u_{ct} &= K_{ctp} \cdot e_{ct} + K_{ctd} \cdot \dot{e}_{ct} \end{aligned} \tag{13}$$

Furthermore, a feed forward part is used so that changes in velocity can be corrected as fast as possible

$$\mathbf{u}_{ff} = K_{ff} \cdot \mathbf{a}_i \tag{14}$$

The preferred velocity for trajectory tracking is then given by a combination of these three terms together with the ideal velocity $\mathbf{v}_i$

$$\mathbf{v}_{pref} = v_i \cdot \mathbf{t}_i + u_{at} \cdot \mathbf{t}_i + u_{ct} \cdot \mathbf{n}_i + \mathbf{u}_{ff} \tag{15}$$

To avoid collisions with other vehicles, a new collision-free velocity minimizing the deviation to $\mathbf{v}_{pref}$ is obtained following an extension to 3D of the velocity obstacles based optimization with dynamic constraints of [1] and [7].

VI. Experimental Results

In this section experiments in simulation are presented, using a quadrotor model including its dynamics, attitude and position control.

A. Simulation setup

The quadratic optimization is solved using the IBM CPLEX library[1]. First the initial goal positions of the animation are computed using the approach described in [1]. The goal positions are then optimized using the approach of Sec. III and the trajectories generated following Sec. IV. The collision avoidance (see [1] and [7]) uses the preferred velocity as described in Sec. V as input. The following physical values are employed in the simulation: radius of the quadrotor $r = 0.35$m, height $h = 0.8$m, maximum speed $v_{max} = 3.5$m/s, maximum acceleration $a_{max} = 2$m/s^2.

[1]http://www.ibm.com/software/commerce/optimization/cplex-optimizer/ (2013)

B. Results

For all the animations, the trajectories have been optimized for quadrotors that have a maximum speed v_{max} and a maximum acceleration a_{max}, as employed for the dynamic simulation of the quadrotors.

The framework for animation display is first demonstrated with a sphere moving in a square with 15 quadrotors, see Fig. 5. In Fig. 6 the velocity of one of the quadrotors is shown and compared to the velocity of the ideal trajectory and the preferred velocity. The drop of velocity at $t = \{5, 10, 15\}$ seconds is due to the corners. Note that the preferred velocity (from the controller) is decreasing before the velocity given by the trajectory. This can be explained by the along track error part of the controller.

A second animation, in Fig. 7 shows a walking human with 15 quadrotors at regular intervals.

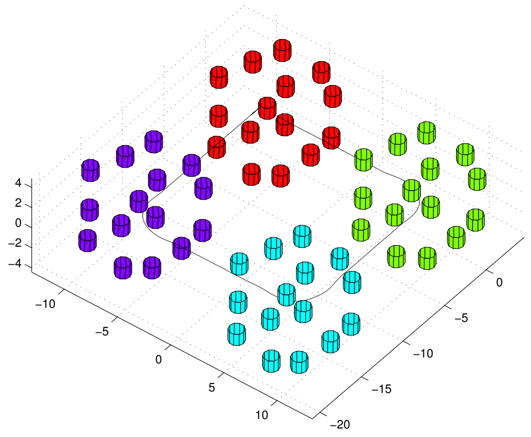

Fig. 5. Animated sphere partially optimized ($\hat{c} = 0.5$) for the viewpoint moving on a square path with 15 quadrotors, shown at $t = \{0, 5, 10, 15\}$ seconds. Different time steps are marked with different colors and the black line shows the average position of all quadrotors along the animation.

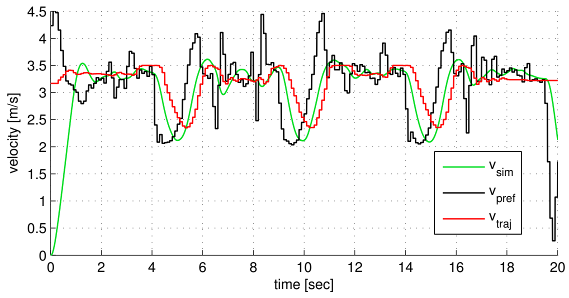

Fig. 6. v_{sim}: actual velocity of the simulated quadrotor; v_{pref}: controller velocity (input to the collision avoidance); v_{traj}: optimized velocity of the trajectory.

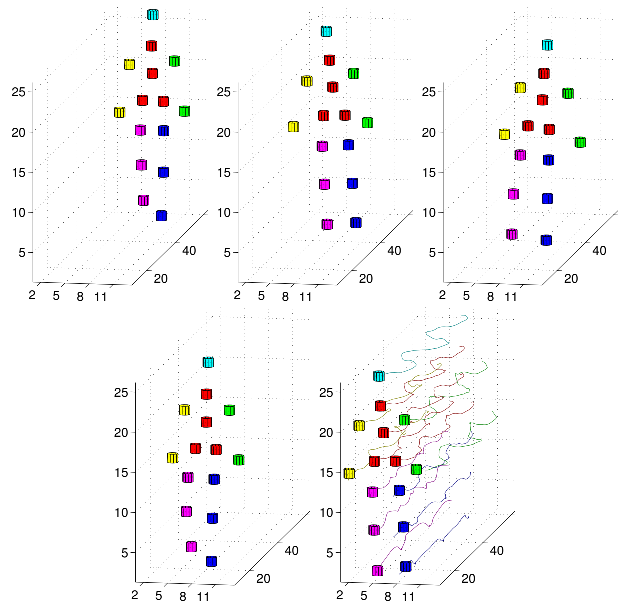
Fig. 7. Animated human with 15 quadrotors, shown at $t = \{0, 6, 12, 18, 24\}$ seconds, first one on the top left, last one on the lower right. The path of each quadrotor throughout the animation is shown in the lower right image.

C. Comparison of viewpoint optimization

To compare the viewpoint optimizations with different $\hat{c}$, the coverage (sum of the squared distance of each point to the closest cluster centre) is computed

$$C = \sum_{i=1}^{n} \sum_{x_j \in \mathcal{X}_i} \|\mathbf{x}_j - \mathbf{g}_i\|^2, \tag{16}$$

as well as for the projected points $\bar{\mathbf{x}}_j$ and 2D cluster centers $\bar{\mathbf{g}}_i$. Table I shows a comparison for various objects. As expected, and visually shown in Figs. 1 and 2, the cost for the projected 2D view decreases as it is optimized for that viewpoint, whereas the cost in 3D slightly increases. As the initial distribution was optimized for the volumetric object, any derivation of that distribution causes an increase in the 3D coverage cost, although this can not be seen visually. The advantage of the view-point optimization is most notable for objects with high volume / surface ratio, such as the sphere, where the decrease in 2D cost is most notable.

Object	3D cost			2D cost		
$c =$	∞	0.5	0.0	∞	0.5	0.0
Sphere	17.52	17.68	18.06	10.32	9.09	8.78
Human	26.68	26.75	26.76	1.45	1.39	1.37
Cube	1.44	1.46	1.53	0.30	0.25	0.24
Rhino	21.23	21.47	21.65	13.50	12.79	12.55

TABLE I

COMPARISON OF THE COST FOR VARIOUS OBJECTS IN 2D (PROJECTED VIEW) AND 3D, FOR VARIOUS DEGREE OF OPTIMIZATION. ALL THE VALUES WERE DIVIDED BY 10^4.

VII. CONCLUSION

In this paper a extension of 3D animation display with a large group of aerial vehicles is described. The initial goal positions are optimized for a viewpoint and the trajectories of the animation optimized to take into account the vehicle's dynamic constraints. A real-time collision avoidance framework with an appropriate trajectory tracking controller ensures safe navigation along the trajectories. Since the approach is distributed, it scales well to a large swarm of vehicles.

VIII. APPENDIX

A sketch of the convergence proof towards a local minimum for the viewpoint optimization algorithm is presented. The algorithm consists of two steps that are repeated until a local minimum is reached. Both steps independently reduce the within-cluster sum of squares (Eq. (1)).

- Assignment step: The points $\bar{\mathbf{x}}_m$ are reassigned to cluster centers $\bar{\mathbf{g}}_i$ according to Eq. (2). This naturally minimizes the first part of the within-cluster sum of squares

$$\sum_{i=1}^{n} \sum_{x_j \in \bar{\mathcal{X}}_i} \|\bar{\mathbf{x}}_j - \bar{\mathbf{g}}_i\|^2, \tag{17}$$

  whereas the second part is not affected as the cluster centers are not moved.
- Update step: Eq. (3) results from minimizing Eq. (1) for each $\bar{\mathbf{g}}_i$:

$$\frac{d}{d\bar{\mathbf{g}}_i}\left(\sum_{i=1}^{n}\left[\sum_{x_j \in \bar{\mathcal{X}}_i} \|\bar{\mathbf{x}}_j - \bar{\mathbf{g}}_i\|^2 + c\|\bar{\mathbf{g}}_{0,i} - \bar{\mathbf{g}}_i\|^2\right]\right) \overset{!}{=} 0, \tag{18}$$

  therefore the update step decreases the overall cost function. The second derivative shows that it is a minimum

$$\frac{d^2}{d\bar{\mathbf{g}}_i^2} C = 2\left(|\bar{\mathcal{X}}_i| + c\right) \geq 0, \tag{19}$$

  where $|\bar{\mathcal{X}}_i| \geq 0$ and $c \geq 0$ by definition.

REFERENCES

[1] J. Alonso-Mora, M. Schoch, A. Breitenmoser, R. Siegwart, and P. Beardsley, "Object and animation display with multiple aerial vehicles," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[2] A. Electronica, "Spaxels / klangwolke - quadrocopter," 2012.

[3] A. Kushleyev, D. Mellinger, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Robotics: Science and Systems*, July 2012.

[4] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[5] A. Schollig, M. Hehn, S. Lupashin, and R. D'Andrea, "Feasiblity of motion primitives for choreographed quadrocopter flight," in *American Control Conference (ACC), 2011*, pp. 3843 –3849, 29 2011-july 1 2011.

[6] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *International Symposium on Robotics Research (ISRR)*, 2009.

[7] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance with motion continuity constraints for aerial vehicles in multi-agent scenarios," in *Autonomous Robots*, 2014 - submitted.

[8] C. de Boor, *A Practical Guide to Splines*. New York: Springer Verlag, 1978.

[9] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.