Object and Animation Display with Multiple Aerial Vehicles

Object and Animation Display with Multiple Aerial Vehicles

Javier Alonso-Mora, Marcel Schoch, Andreas Breitenmoser, Roland Siegwart and Paul Beardsley

***Abstract*— This paper presents a fully automated method to display objects and animations in 3D with a group of aerial vehicles. The system input is a single object or an animation (sequence of objects) created by an artist. The first stage is to generate physical goal configurations and robot colors to represent the objects with the available number of robots. The run-time system includes algorithms for goal assignment, path planning and local reciprocal collision avoidance that guarantee smooth, fast and oscillation-free motion. The presented algorithms are tested in simulations and verified with real quadrotor helicopters and scale to large robot swarms.**

I. INTRODUCTION

Screens that can display visual content in the third dimension and are no longer limited to flat surfaces remain an interesting challenge. In this paper we present a method to display visual content with a swarm of aerial vehicles. In [1] a method and setup to display images and animations with multiple ground vehicles was presented. We build on that work and extend it to a 3D display with aerial vehicles. In this line, [2] proposed a large swarm of micro helicopters which act as three-dimensional pixels in free space. The work still appears to be in concept phase, but the animations demonstrate the exciting potential of flying displays. Although when using a low amount of robots no complex objects can be formed, if used for specific purposes in the right environment, such a flying display can very well show fascinating effects.

The last five years have seen an increasing research interest in agile multi-rotor aerial vehicles. An overview of an aerial vehicle testbed, state of the art methods and challenges is given in [3]. In [4] a centralized and costly optimization method was presented to create agile motions for groups of quadrotors (including obstacle avoidance). In parallel, [5] presented algorithms to produce choreographic motions for a group of quadrotors, synchronizing them to music. In contrast to our objective, in these works, the desired trajectories are not only precomputed, but also specified individually for each vehicle, which may not scale well to large systems. When controlling large groups of aerial vehicles collision avoidance is of paramount importance, in particular, [6] presented a method for reciprocal collision avoidance for simple airplanes. We choose a similar strategy to guarantee collision-free motions and build on the 3D extension of [7].

J. Alonso-Mora, M. Schoch, A. Breitenmoser and R. Siegwart are with the Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland {jalonso,mschoch,andrbrei,rsiegwart}@ethz.ch

J. Alonso-Mora, P. Beardsley are with Disney Research Zurich, 8092 Zurich, Switzerland {jalonso,pab}@disneyresearch.com

Like in our experiments, most of the previously mentioned works rely on an external motion tracking system. In contrast, [8] presented an on-board vision based localization method that could allow for fully autonomous navigation.

Given an input object, both surface and volumetric representations are possible. In this work we transform the object, which is defined through a surface mesh, into a volumetric representation. This choice is discussed in more detail in Section III-E. The alternative of using a surface mesh as direct representation of the surface is common in computer graphics. Mesh simplification and remeshing operations like those in [9] and [10] are furthermore based on a tessellation of the surface. This is related to distributing a group of robots over a surface mesh, as demonstrated in [11] which applies a Centroidal Voronoi tessellation for covering surfaces with a group of climbing robots.

The remainder of the paper is structured as follows. Section II provides an overview of the system. Section III describes the goal generation for the object display. In Section IV the real-time controller is presented, including goal assignment, path planning and collision avoidance. The extension from static objects to animated displays is described in Section V. Section VI contains experimental results with physical robots and in simulation. Section VII concludes the paper and indicates future work.

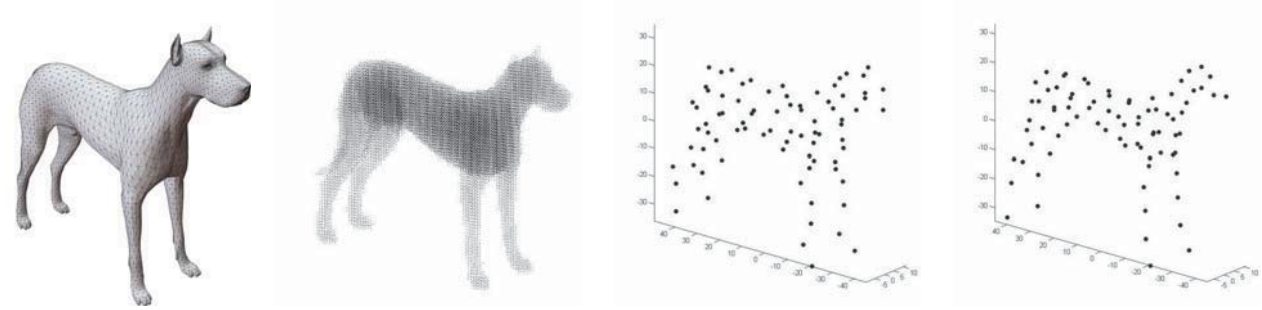
II. SYSTEM OVERVIEW

The presented system extends our work [1] for image and animation display in 2D to flying displays in 3D. The system consists of a set of N aerial vehicles, a motion tracking system, and a central computer that wirelessly sends motion commands to the robots. Due to the limitations in space and cost, experiments with a large swarm of robots are provided in simulation using an identified dynamic model of a real quadrotor helicopter. Experiments with two real quadrotors are also presented as proof of concept.

Given an input object and N robots of radius $\{r_1, \ldots, r_N\}$ the method is divided into two steps, first goal generation, followed by a real-time controller that drives the robot to the goal positions.

Goal generation is the computation of the robot positions in order to represent a desired object given a specified number of robots; the algorithm for the goal generation is described in Section IV. The goal positions for a given object are computed independently of the current position of the robots and thus can be pre-computed.

At run time, a real-time controller (described in Section V) drives the robot pixels to the computed goal positions. In each time step the following three computations take place.



(a) (b) (c) (d)

Fig. 1. Goal generation for a dog using 80 agents. (a) Rendered obj-file. (b) Volumetric point cloud. (c) Initial samples using a Poisson distribution. (d) Goal positions after k-means optimization.

Firstly, the robots are assigned to the goal positions by an auction algorithm. Secondly, a preferred velocity towards its assigned goal position is computed for each robot independently. Finally, a distributed reciprocal collision avoidance algorithm finds a collision-free velocity for each robot, taking its dynamics and the current positions and velocities of its neighbors into account. The new velocities are close to the robots' preferred velocities, and enable a safe motion update. Although the dynamics of the vehicles are considered in the collision avoidance algorithm, the robots are treated to be of spherical shape.

As shown in [12] it is intractable to compute the optimal motions of N robots interacting in a common workspace because the search space is exponential in N. Thus a distributed scheme, like the one that we propose, is needed for scalability to large swarms of robots.

III. Goal generation

A. Preprocessing of object files

In computer graphics there exists a variety of formats on how to represent an object in 3D space. Most of them use vertices, edges and faces to describe the surface of an object and do not include details about the volume inside, since it is not needed for rendering. In this work, the input object is given in the wavefront *obj*-file format, which is relatively widespread and available as an import/export option in most 3D design programs. In this format, an object is given by vertices and faces forming a surface mesh. Furthermore, texture coordinates can be stored.

The first step of the method is to convert the surface representation of the object into a uniformly distributed point cloud of its volume (see Figure 1 (a) and (b)). This is done by generating a grid of points and keeping those points that lie inside the volume enclosed by the object's surface mesh. The generated volumetric point cloud $\mathcal{V}$ is then used to obtain the goal positions of the robots.

B. Initial samples

To obtain the goal positions, an iterative optimization is used (see following section). This optimization distributes the goal positions in the volume by converging to a local optimum. Therefore a near-optimal distribution of initial samples is vital to obtain a balanced representation of the object upon convergence of the iterative optimization. Initial goal positions are sampled within the volumetric point cloud following a Poisson distribution [13].

We compared this method with random, farthest first and subset initialization [14] and in all cases Poisson sampling provides the best results, both, visually and in computational time (see Figure 1 (c), and Table I for the case of the dog object). The grid to generate the point cloud for this experiments has dimensions 20 x 80 x 60, 80 goal positions are computed, the methods are repeated 200 times and the average computational time and cost W of the final distribution are obtained. See Section III-C for details.

C. Iterative optimization

Given the initial goal positions and the volumetric point cloud, the final goal positions for good representation of the input object are found by an optimization using a k-means clustering algorithm [13] (see Figure 1 (d)). The algorithm iteratively performs the following two steps. First, the point cloud is clustered with respect to the current goal positions and second, goal positions are updated to the centroids of each one of the clusters. This method asymptotically converges to a local optima of the cost measure

$$W = \sum_{i=1}^{N} \sum_{z \in T_i} \|g_i - z\|^2, \tag{1}$$

where g_i are the goal positions and $T_i \subset \mathcal{V}$ are the clusters of a partition of $\mathcal{V}$ given by

$$T_i = \{z \in \mathcal{V} \,|\, ||z - g_i|| \leq ||z - g_j||, j \in [1, N], j \neq i\}. \tag{2}$$

This method is equivalent to the Voronoi coverage method used in [1] to distribute N robots in a planar image.

D. Post-processing of goal positions

First, the resulting goal positions are rescaled to ensure a minimal distance between them.

Second, color can be added. In object files the color of the surface mesh is given by a texture. In our implementation, for each goal position its color is selected equal to that of

Initialization	Init. [s]	Optim. [s]	W/W_{max}	Iterations
Random	< 0.01	2.03	0.969	91.2
k-means++	0.09	1.51	0.967	75.6
Farthest first	0.27	2.18	0.972	101.0
20 subsets	0.9273	2.08	1.000	35.7

TABLE I

COMPARISON OF THE COMPUTATION TIME, COST AND NUMBER OF ITERATIONS FOR VARIOUS INITIALIZATION METHODS.

the closest surface point. This adds an extra cue to identify an object represented with a low number of robots.

E. Volumetric vs. surface representation

Objects can be displayed either by their surface or their volume. Given enough robots, an object may be displayed more accurately if the points are placed on the surface manifold since inner points are not necessary to define the shape. Nevertheless, consider now representing long and thin objects (such as the rod of Figure 2) with a low number of robots, due to cost limitations for instance. In this case, if the robots are distributed on the surface manifold, their positions are not aligned, leading to a bad representation, as shown in the right of Figure 2. On the other hand, if the robots are distributed over the volume, a clear representation of the long and thin object can be obtained, as shown in the middle of Figure 2. Arbitrary objects in general include long and thin parts, such as legs like the ones of the dog in Figure 1. For this reason and the limitation in the number of robots, the volumetric representation is used.

IV. REAL-TIME CONTROL

A real-time controller that scales well with the number of robots is implemented to drive the robots to the set of goal positions representing the input object. In each iteration the robots are first uniquely assigned to the goal positions; secondly, a preferred velocity is computed without taking the other vehicles into account; thirdly, a collision-free velocity is computed via a distributed local collision avoidance method. The collision-free velocity is then tracked by a controller that is based on an identified model of the vehicle dynamics. This method is an extension to aerial vehicles of the method presented in [1] for image display. Therefore we refer to [1] for some more details.

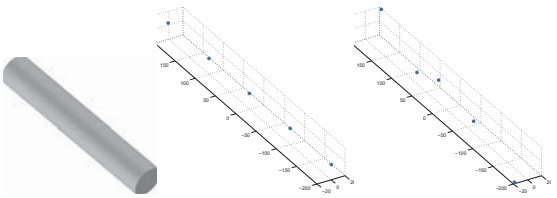


Fig. 2. Comparison between representation methods by the example of a rod using 5 robots. Left: Rendered obj-file. Middle: Volumetric representation. Right: Surface representation

A. Goal assignment

In order to minimize the convergence time, the robots at current position $\mathbf{p}_i$ are uniquely assigned to the goal positions $\mathbf{g}_j$ by minimizing the sum of squared distances $||\mathbf{p}_i - \mathbf{g}_j||^2$. This minimization is solved via the auction algorithm presented in [15] and previously used in [1], which scales very well with the number of robots.

B. Preferred velocity

For each vehicle, a preferred velocity $\mathbf{v}_i^p$ is given by the vector pointing towards its goal position $(\mathbf{g}_j - \mathbf{p}_i)$ and with magnitude proportional to the distance. This preferred velocity is saturated to the preferred speed V_{pref} of the vehicle.

C. Reciprocal collision avoidance

The reciprocal collision avoidance algorithm is fully distributed. All vehicles are considered to have spherical shape. For each robot, given its preferred velocity $\mathbf{v}_i^p$ and the current velocities $\mathbf{v}_j$ and positions $\mathbf{p}_j$ of its neighbors, a new collision-free velocity $\mathbf{v}_i$ is computed. In order to avoid collisions while guaranteeing smooth motions in the case of complex dynamics, the optimal reciprocal collision avoidance (ORCA) method presented in [7] is extended. ORCA is an optimization with linear constraints (one per neighboring robot) based on velocity obstacles [16] and exploits the fact that all controlled robots in the environment react following the same scheme, thus avoiding oscillations. This method guarantees oscillation-free and smooth motions in multi-robot scenarios but assumes the robots to be holonomic and does not take acceleration constraints into consideration. Following the line of thought of [17], we extend it to verify the dynamics of the aerial vehicles used in our experiments.

1) Idea: For each vehicle and at each time step, a straight-line constant-velocity reference trajectory (given by $\mathbf{v}_j^{new}$) is computed as if the robot was holonomic following [7]. In this optimization, an extra constraint is added (IV-C.3) to account for the kino-dynamic capabilities of the vehicle. Furthermore, the radius of the vehicle is enlarged (IV-C.4) to account for the tracking error of the reference trajectory when using the specified controller (IV-D).

2) Reachable velocities: For simplicity (although not required for our method), we consider a maximum acceleration equal in all directions and independent of the initial conditions of the vehicle. For the identified model of the quad-rotor used in the experiments, this was a reasonable assumption. We further consider a maximum time δ to reach the desired velocity and we obtain the sphere

$$S_{\mathbf{v}_i} = \{\mathbf{v} \mid \quad ||\mathbf{v} - \mathbf{v}_i|| < \delta a_{max}\}. \tag{3}$$

3) Tracking error: The space of feasible states and reachable velocities is discretized. The trajectory of the quad-rotor is simulated (note the symmetries) using an identified model and the LQR trajectory tracker for the quad-rotor (which is also used in the experiments) of Section IV-D. The maximum tracking errors are computed and for each set of initial conditions, the inner-sphere $S_{\mathbf{v}_i,E} \subset S_{\mathbf{v}_i}$ of the

velocities with tracking error lower than E is stored. This can be precomputed for the dynamics of the robot.

4) Extended radius: If the radius of the robots is extended by E and the constraint $v_i^{new} \in S_{\mathbf{v}_i^{current},E}$ is added to the optimization with linear constraints, the trajectories of the robots are collision-free, for the identified model and design trajectory tracker. Note that E is variable and may decrease if two robots are close to each other.

5) Avoiding deadlocks due to symmetry: The method presented in [7] can lead to deadlocks in symmetric and ideal cases (for example when two robots are in direct collision course in a noise-free scenario), this can be avoided by adding an inner cone of aperture α to the Velocity Obstacle formed by two vehicles preventing the linear constraint to be perpendicular to the axis of the cone, which would have led to an asymptotic decrease in speed to zero in this particular case.

D. Trajectory tracking

In our simulated experiments we use an identified second order model of the dynamics of the AscTec Hummingbird quadrotor [18]. Both a PID and an LQR trajectory controller were derived in [18] for the aforementioned vehicle. These controllers guarantee smooth convergence to the desired trajectory, in our case given by the velocity v_i. In the experiments presented in this paper, the LQR controller is used.

V. Animation display

The previous sections describe a method to display a single object with multiple aerial robots. This section extends the method to display 3D animations specified by a sequence of input object files or *keyframes*. Modifications of the methods of Sections III and IV are described below.

For each frame (given by an object file) of the animation, a set of goal positions is obtained using the method in Section III. In parts of the animation where there is sufficient correlation, the goal positions computed in the previous frame serve as initial positions for the computation of the goal positions in the current frame, replacing the Poisson sampling described in Section III-B. This initialization reduces computation time and disparities between consecutive goal sets.

The controller of Section IV drives the robots through the given goal positions, representing the frames in sequence at given time-instances with $K_f \Delta t$ separation. $K_f \in \mathcal{N}$ is a design constant and Δt is the time-step of the controller. In order to achieve smoother motions, the velocities of the robots are synchronized when the set of goal positions is changed

$$||\mathbf{v}_i^{p,f}|| = \min(V_{max}, ||\mathbf{p}_i - \mathbf{g}_i^f||/(K_f \Delta t)), \qquad (4)$$

where $\mathbf{v}_i^{p,f}$ and $\mathbf{g}_i^f$ are the preferred velocity and goal position of robot i at frame f.

With this method, an artist can create an animation with standard software and export the keyframes as object files that are imported in our framework.

VI. Experimental results

In this section we present experiments of object and 3D animation display with a simulated swarm of quad-rotors. We further verify the algorithms in experiments with two real quad-rotors. A video containing the experiments accompanies this paper.

A. Experimental setup

AscTec Hummingbird quad-rotors[1] are used in the experiments. The quad-rotors communicate via XBee modules[1b] with a central computer. A commercial motion tracking setup[1c] is used to accurately localize the vehicles. In the simulations, a second order identified model of the AscTec Hummingbird quad-rotor is used (see [18]).

Given the input object files, the goal positions are precomputed using the algorithms of Section III. The control loop explained in Section IV is performed at $25Hz$ and an internal rate controller[2] on the quad-rotors is further used.

Due to the reduced size of the experimental space ($5.5m$ x $5m$ x $2.5m$), the maximum velocity of the quad-rotors is limited and the walls of the room are added as linear constraints in the local collision avoidance.

In the simulations, the aerial vehicles are displayed as spheres able to change color, although their behavior follows the same dynamics of the quad-rotors used in the experiments.

The following parameters are used: radius of the quad-rotor $r = 0.3$m, desired radius enlargement $E = 0.2$m, maximum speed $V_{max} = 3$m/s, preferred speed $V_{pref} = 2.4$m/s, maximum acceleration $a_{max} = 2.5$m/s^2, time to reach a new velocity $\delta = 0.3$s.

B. Object display

The object display is demonstrated with three different object files: a star, an ant and a dog (see Figure 3 top and middle right and in Figure 1). In the first experiment, eleven simulated quad-rotors lift off from ground and display a star. In the second experiment, 80 simulated quad-rotors display the ant starting from the dog. The vehicles' trajectories are depicted on the right-most images. In the third experiment (bottom row), 80 simulated quad-rotors lift off the ground and display the dog. The minimum distance between vehicles is displayed in the right-most graphic. Note that although robots may come close (due to their dynamics) no collisions happen.

C. Animation display

This method allows an artist to design a 3D animation that is then displayed by aerial vehicles in a fully automated way. In Figure 4 we present an experiment where 50 simulated quadrotors display an animation of a human walking. Images are captured every 5s and distances are displayed in meters. The input animation is shown in the top row, the 3D-frontal view of the aerial display in the middle row and a side view in the bottom row.

[1] www.asctec.de; www.digi.com; www.vicon.com

[2] www.ros.org/wiki/asctecmavframework

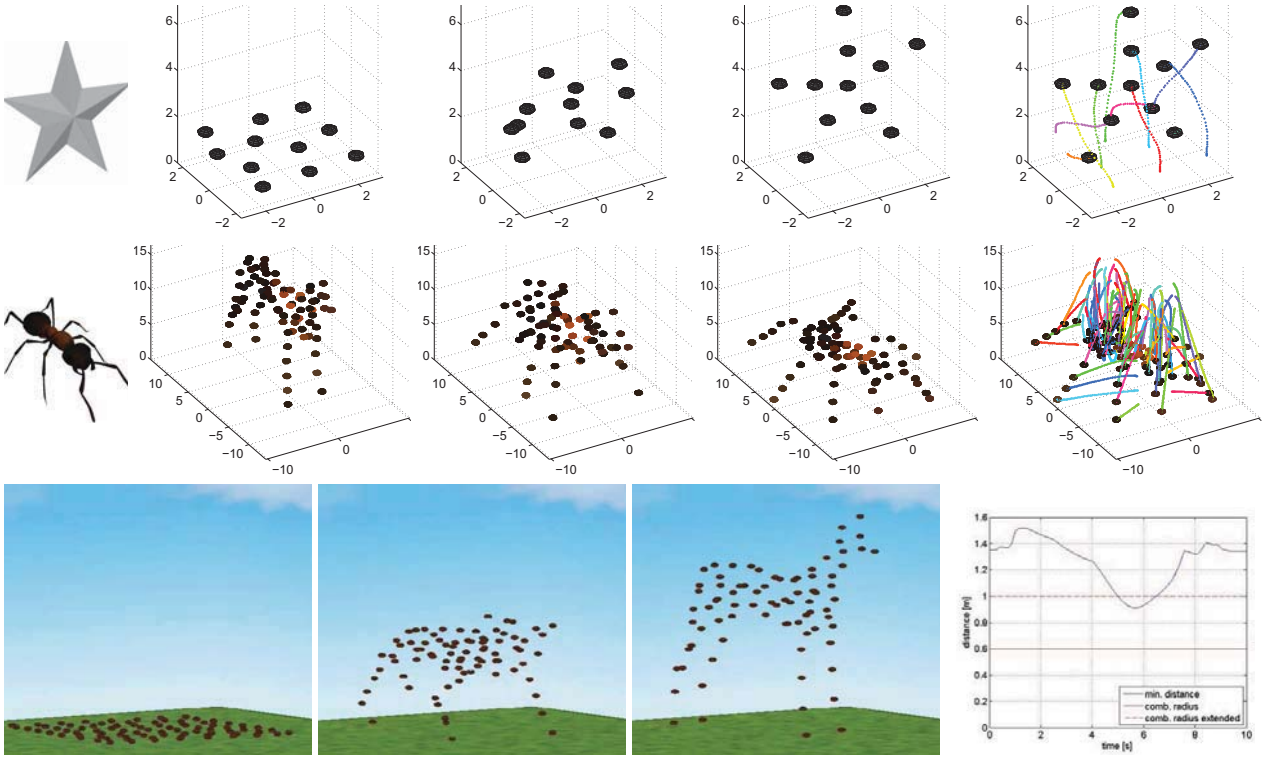


Fig. 3. Top row and left to right: Input object star; eleven simulated quad-rotor helicopters in their initial configuration, followed by an intermediate capture after 2s and in their final positions (4s) displaying the star; trajectory of each quad-rotor in different colors. Middle row and left to right: Input object ant; 80 simulated quad-rotor helicopters in their initial configuration displaying a dog, followed by an intermediate capture after 3s and in their final positions (7s) displaying the ant; trajectory of each quad-rotor in different colors. Bottom row: 80 simulated quad-rotors take off and display the dog (three screenshots at 0s, 4s and 9s; the minimum distance between quad-rotors in each time instance is shown in the right-most image. The minimum distance stays over twice the radius of the vehicles and thus collisions are avoided. Realistic background is added to visualize the effect of such a display.

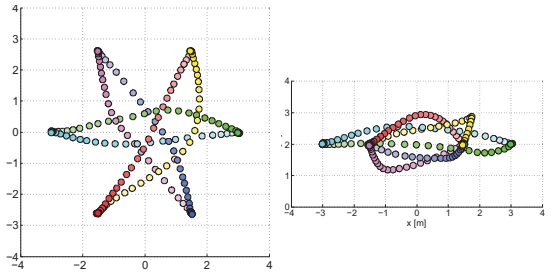


Fig. 5. Six simulated quadrotors (with identified dynamic model and measurement noise) exchange positions on a circle. Left: Top view of the trajectories [m]. Right: Frontal view.

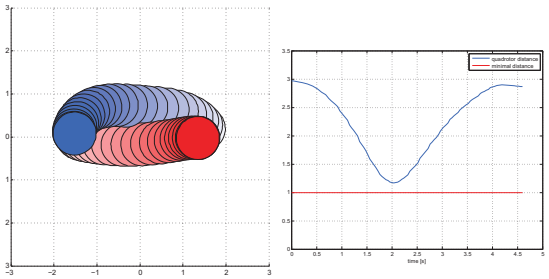


Fig. 6. Two real quadrotors exchange positions. Left: Top view of the trajectories. Right: Distance between the quadrotors at each time instance (blue) and desired minimal distance (red).

D. Collision avoidance

We present two experiments to show the performance of the collision avoidance algorithm for aerial vehicles. The first one consists of six simulated quadrotors (with identified dynamic model and measurement noise) exchanging positions on a circle. The trajectories are shown in Figure 5. The results of the second experiment are shown in Figure 6, where two real quadrotors exchange their positions in flight.

VII. Conclusion and Future Work

In this paper we present a method to create 3D displays with aerial vehicles. An object or 3D animation created by an artist is taken as input, and is first transformed into a set of goal positions. In real time, the aerial vehicles are assigned to the goal positions and driven towards them. Resulting trajectories are collision-free. The underlying local reciprocal collision avoidance algorithm can be distributed and enables the method to scale well to large groups of vehicles. Simulated experiments are presented with up to 80

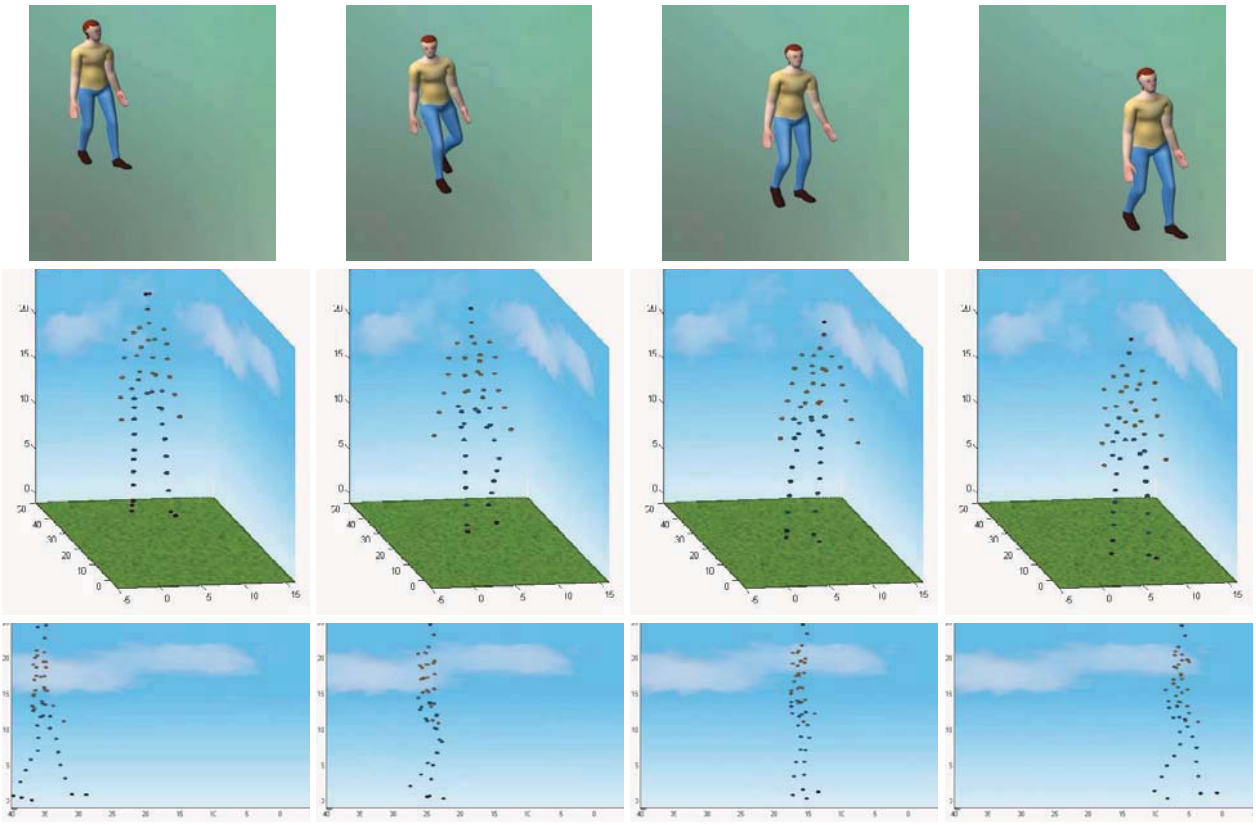


Fig. 4. 50 simulated quadrotors display an animation of a human walking. Images are captured every 10s and distances are in meters. The robot color is adjusted according to color from surface mesh texture. Realistic background is again added to visualize the effect of the display. Top: Input animation. Middle: 3D-frontal view. Bottom: Side view. The full animation is found in the accompanying video.

quad-rotors subject to their real dynamics, and the method is further verified with two real quadrotors.

It becomes apparent that when using a low number of aerial vehicles, the view-point of the object has an important effect on the accurate representation. In this work we have presented a general method, but one could benefit from view-point information in the future.

References

[1] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple robots," *Int. Journal of Robotics Research*, vol. 31, pp. 753–773, May 2012.

[2] Flyfire, "http://senseable.mit.edu/flyfire," 2010.

[3] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *Robotics Automation Magazine, IEEE*, vol. 17, pp. 56 –65, sept. 2010.

[4] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program (miqp) trajectory generation for heterogeneous quadrotor teams," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2012.

[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. Snape and D. Manocha, "Navigating multiple simple-airplanes in 3d workspace," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3974 –3980, may 2010.

[7] 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.

[8] S. Weiss, M. Achtelik, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012.

[9] C. Erikson, "Polygonal simplification: An overview," 1996.

[10] Q. Du, M. D. Gunzburger, and L. Ju, "Constrained centroidal voronoi tessellations for surfaces," *SIAM J. Sci. Comput.*, vol. 24, pp. 1488–1506, May 2002.

[11] A. Breitenmoser, J. Metzger, R. Siegwart, and D. Rus, "Distributed coverage control on surfaces in 3d space," in *Proc. of The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[12] J.-C. Latombe, *Robot Motion Planning*. Boston: Kluver, 1991.

[13] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Technical Report 2006-13, Stanford InfoLab, June 2006.

[14] P. S. Bradley and U. M. Fayyad, "Refining initial points for k-means clustering," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, (San Francisco, CA, USA), pp. 91–99, Morgan Kaufmann Publishers Inc., 1998.

[15] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, vol. 14, no. 1, pp. 105–123, 1988.

[16] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[17] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2012.

[18] M. Burri, J. Nikolic, J. Rehder, and R. Siegwart, "Aerial service robots for visual inspection of thermal power plant boiler systems," in *Int. Conf. on Applied Robotics for the Power Industry*, 2012.