EAVE: Emotional Aerial Vehicle Evaluator

Marc Lieser[1], Ulrich Schwanecke[1], Jörg Berdux[1]

Abstract: Today, semi-autonomous quadrotors are already available at affordable prices and have the potential to become part of everyday life due to the variety of possible applications. To ensure that people feel safe around quadrotors and to optimize flight times, their size should be kept to a minimum which results in their appearances remaining purely functional. This reduces the possibility of adding anthropomorphic or zoomorphic features that are typically used in order to increase acceptability by conveying the robot's intent or emotion. Constrained by mechanical appearance, other non-verbal communication channels can be exploited instead, in particular robot motion. The application *EAVE* presented in this paper was developed with the idea to design and evaluate trajectories that breathe life into inanimate, mechanical quadrotors in order to improve interaction in human-robot companionships. It extends our existing quadrotor testbed *ICARUS*, which is capable of tracking arbitrary trajectories of real and simulated quadrotors that were designed using *EAVE*. We demonstrate that applying some of the established principles of character animation to the design of quadrotor trajectories opens up the possibility of conveying intent and improving interaction, though the appearance of the quadrotor remains purely functional.

Keywords: human-quadrotor interaction; emotional aerial vehicles; non-verbal communication; motion anticipation; quadrotor companion; principles of animation; quadrotor testbed; social robots.

1 Introduction

Quadrotors experienced a huge gain in popularity over the past twenty years. Thanks to their mechanical simplicity and affordability, they have found their way from the hobby sector via research into everyday applications. Most of them take advantage of their elevated view, whether for locating victims in disaster scenarios, for inspection of buildings or for filming movies. Private consumers mainly utilize quadrotors as self-flying cameras that accompany them during various leisure activities. Today, semi or even fully autonomous quadrotors are available and follow humans while avoiding obstacles and receiving commands via smartphones or gestures. This also enables them to be helpful companions in home environments, where in future scenarios they could accompany people through daily life as smartphones already do today. Whatever the exact purpose, the number of robot assistants — be it ground-based or of aerial nature — will most certainly increase and require research on possible communication channels for human-quadrotor interaction.

Even with today's state of technology and miniaturization, it is important and challenging to keep the size of quadrotors used in interaction scenarios to a minimum. Larger rotors

[1] Department of Computer Science, RheinMain University of Applied Sciences, Wiesbaden, Germany {marc.lieser,ulrich.schwanecke,joerg.berdux}@hs-rm.de

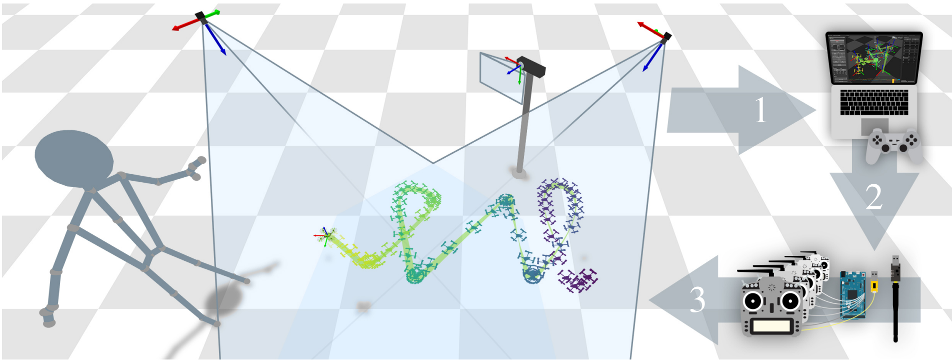

Fig. 1: Overview of our quadrotor testbed *ICARUS*, an affordable, portable indoor testbed: After optical pose estimation of the individual quadrotors (1), control variables are determined (2) and sent via different radio control systems (3) to the quadrotors. The depicted schematic shows a quadrotor in a human-quadrotor interaction scenario following a trajectory designed with *EAVE*.

of larger platforms, the generated noise and downwash are more likely to be perceived by people as disturbing [Ye17]. This introduces an inhibition threshold which is not desirable in companionships. For this reason, the quadrotors used in our scenarios are lightweight and cannot carry sensors or other electronics beyond ensuring their flight capability.

Introducing anthropomorphic features is often used in order to make the robot's inner state more identifiable for humans. This may be hard or even impossible to implement without adding to the already critical payload of quadrotors. Mediating motion intent without social cues such as gestures or gaze can be achieved by Augmented Reality (AR) applications [Wa18], but requires additional hardware. Instead of adding extra hardware, other channels of non-verbal communication can be further explored, in particular robot motion [HJ14], which is capable of transporting information beyond motion intent, namely the robot's emotions. Hence, for quadrotors that have no moving parts other than rotors, the key to convey intent lies in their trajectories and their augmentation with motions familiar to humans. In order to bring the purely mechanical remaining robots to life and to increase acceptance, we apply some of Disney's well established principles of character animation [TJ81] that have already been adapted for robots other than quadrotors [RP12, GT11]. The desire to animate quadrotors exists [Ka17], but published research remains on a conceptual level [De18] or lacks detailed implementation and parameterization [Ca16].

In this paper we present the application *EAVE* (Emotional Aerial Vehicle Evaluator) that allows to design and evaluate quadrotor trajectories. Following animation, trajectories are defined by keyframes, each consisting of position, velocity, acceleration and heading angle. Some principles of animation can be applied and parameterized in order to increase the expressiveness of the quadrotor's motion with the objective to enhance user interaction by enabling trajectories to communicate the quadrotor's emotion or intent. Based on the defined keyframes, smooth trajectories are generated and tracked by quadrotors within our testbed *ICARUS* that allows for safe test flights of arbitrary quadrotor models in a simulated environment but also controls real quadrotors.

In the next section we give a short overview of the current state of our quadrotor testbed and point out where *EAVE* is extending the infrastructure. In Section 3 we describe the trajectory generation and provide some examples of applied principles of animation. Finally, we summarize our work in Section 4 and give an outlook on future research.

2 Testbed

This section describes the current state of our quadrotor testbed *ICARUS*, that is extended by *EAVE* through software. We outline the core components of the system and give an insight of the libraries used. A detailed description of *ICARUS* can be found in [Li17]. This low-cost quadrotor testbed was continuously expanded and improved over the past years. A schematic of the testbed is given in Fig. 1. The quadrotors used in our experiments and the overall data flow of *ICARUS* and *EAVE* are shown in Fig. 2.

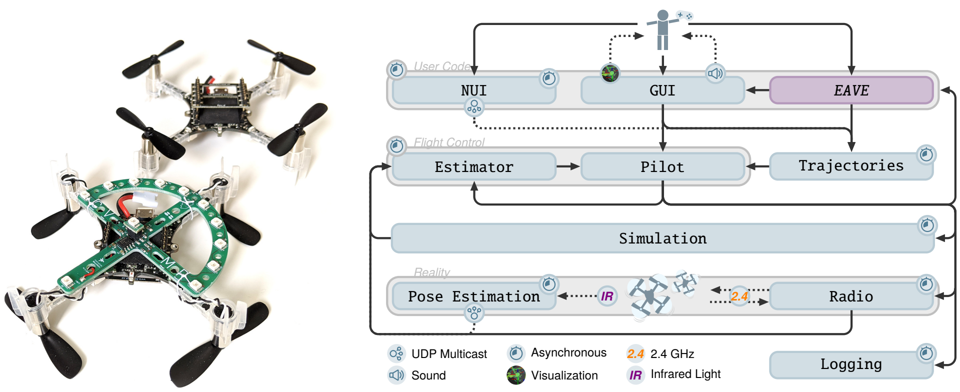

Fig. 2: Bitcraze Crazyflie 2.0 with the rotors mounted upside down so they do not obscure the LEDs of the attached tracking marker and an unmodified platform for comparison in the background (left). Data flow of *ICARUS* with *EAVE* as a component that extends the testbed by custom user code (right).

In general, all testbed software components are developed in `C++` using the `Boost` libraries for multithreading and networking. Serialization of data structures for UDP messages is implemented with `Protobuf`. The *User Code* layer consists of a classical Graphical User Interface (GUI) and an `OpenGL` visualization implemented with `Qt` and `glm`; a screenshot is shown in Fig. 3. Also, manual control using a joystick (implemented with `SDL`) and a gesture-based Natural User Interface (NUI) using a Microsoft Kinect 2 with the `Kinect 2 SDK` are implemented in the top layer. At this point, the existing infrastructure can also be extended by custom user code like *EAVE*. Next to visual feedback, the user interface also generates synthetic rotor sound. For this purpose, the sound of a single rotor was recorded during hover and is played back spatially during the simulation using SFML. The sound of each rotor is pitched by the ratio of rotor revolutions to the revolutions required for hovering. Upper and lower limits were chosen by experiment. Below hover revolutions, the playback volume fades in linearly and remains constant above.

The implementation of the *Flight Control* layer uses a predictor–corrector estimator similar to the one described in [Lu14]. It filters noise from pose estimation and latency-compensates the quadrotors' states. Data from the estimator is used by the pilot implementation to hover in position or to track trajectories defined by the user. The control loop runs at a frequency of 100 Hz. For trajectory control a Model-Predictive Controller (MPC) with a discretization time of 0.1 s and a time horizon of 2 s is used. The implementation is based on the open-source code from [Fa18] who is using ACADO to set up the optimization problem and qpOASES to solve it. The hover controller was presented in [MK11]. Control variables are sent to the radio or simulation and back to the estimator for the next control step. A configurable range of data is also sent to an asynchronous logging implementation. Our *Simulation* is based on the quadrotor dynamics summarized in [Mi10, Lu14] and is implemented using Eigen.

For off-the-shelf (brushless) quadrotors in the *Reality* layer we use the Arduino-based serial remote control library serialrc[2] [Li17] in conjunction with a FrSky Taranis X9D radio system. Less effort but shorter flight times come with the brushed Bitcraze Crazyflie 2.0, which we integrated using the radio library that is part of [HA17]. The in-house developed OpenCV-based optical pose estimation is described in [Tj19]. The markers utilized in our system were optimized for the use with quadrotors and can be seen attached to a Crazyflie in Fig. 2. Pose estimation runs on a dedicated machine and multicasts UDP messages that consist of a six degrees of freedom pose and a timestamp. The estimator receives position and attitude from pose estimation and determines velocity and angular velocities by numerical differentiation. Telemetry data is also forwarded to the estimator and includes the current battery voltage to compensate the thrust command for varying voltages.

3 Trajectories

For most robots, it is completely sufficient to determine a trajectory on the shortest path between two points. This results in calculated straight or plain "mechanical" trajectories that render the identification of the robots intent or emotion impossible in a human-robot companionship. In our scenario, in which the appearance of the quadrotors is kept mechanical, their trajectories should distract from that design and be able to convey certain feelings to revalue quadrotors for humans in human-robot interaction. This section describes the representation of trajectories and our design approach to applying some of the basic principles of animation, with the goal of making quadrotors more alive. A screenshot of the user interface including the trajectory editor can be seen in Fig. 3.

[2] https://github.com/cvmrgroup/serialrc

3.1 Trajectory Representation

We describe trajectories as piecewise polynomial functions (see for example [Co13]). Quintic (fifth-order) polynomials are a common choice for the representation of trajectories, since their first and second temporal derivatives — velocity and acceleration — are continuous and thus result in smooth trajectories. Furthermore boundary conditions, such as position, velocity, acceleration and time, can easily be set. A scalar trajectory quintic polynomial together with its first and second order derivates, given as

$$\begin{aligned} p(t) &= at^5 + bt^4 + ct^3 + dt^2 + et + f \\ \dot{p}(t) &= 5\,at^4 + 4\,bt^3 + 3\,ct^2 + 2\,dt + e \\ \ddot{p}(t) &= 20\,at^3 + 12\,bt^2 + 6\,ct + 2\,d \end{aligned}$$

result in a linear system

$$\begin{pmatrix} p(0) \\ \dot{p}(0) \\ \ddot{p}(0) \\ p(T) \\ \dot{p}(T) \\ \ddot{p}(T) \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ T^5 & T^4 & T^3 & T^2 & T^1 & 1 \\ 5\,T^4 & 4\,T^3 & 3\,T^2 & 2\,T^1 & 1 & 0 \\ 20\,T^3 & 12\,T^2 & 6\,T^1 & 2 & 0 & 0 \end{bmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} \tag{1}$$

with start boundary time $t = 0$ and end boundary time $t = T$, where T is the duration of the trajectory segment. For $T \neq 0$ the linear system (1) can be uniquely solved for the polynomial coefficient vector $\mathbf{c} = (a, b, c, d, e, f)^\top$. A trajectory $\mathcal{K}$ is represented by a list of $N + 1$ boundary conditions $\mathbf{p}_i(t_i)$ at prescribed time t_i, i.e. $\mathcal{K} = (\mathbf{p}_0(t_0), \ldots, \mathbf{p}_N(t_N))$. Scalar trajectories can be extended to the multi-dimensional vector case in a straightforward way. Three-dimensional vectors of position, velocity and acceleration are pooled with time and quadrotor heading into keyframes, a term borrowed from animation. When a trajectory is finally sampled for the controller, the piecewise polynomials are interpolated independently in each dimension and joined to obtain the final trajectory. The heading angle is interpolated linearly. As sampling rate we use 100 Hz according to the update rate of the controller used.

3.2 Trajectory Design

It is difficult, if not impossible, to predict the intent of a mechanically designed quadrotor or of robots in general. Disney's twelve basic principles of character animation [TJ81] have proven beneficial in robot motion design [HJ14] but have not yet been applied to quadrotors. These principles are the result of several decades of work of animators and serve the main purpose of creating the illusion that cartoon characters adhere to the basic laws of physics, but they also deal with more abstract topics such as emotional timing and character appeal.

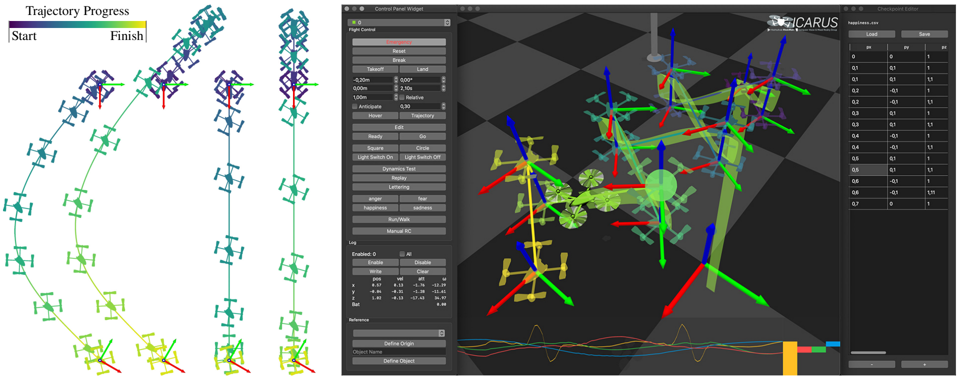

Fig. 3: Two pairs of trajectory segments, with and without anticipation (velocity and heading), which differ by the velocity at the target keyframe (left). *EAVE*'s GUI with the control panel on the left side, trajectory editor on the right side and visualization in the middle (right).

Since quadrotors are already subject to the laws of physics and have no moving parts other than rotors, not all principles are applicable, or at least not in their entirety. On the other hand, some principles can easily be applied by just setting the according boundary conditions of the linear system (1). The *Slow In and Slow Out* principle creates realistic acceleration of objects and can be adjusted by reducing the acceleration at keyframes. Velocity information generates arched trajectories and thus more natural motion, as described by the *Arcs* principle. On the contrary, a mechanical robot motion would be the direct path between two points. *Timing* can also be modified with a boundary condition and is probably the most powerful tool to change emotions. Velocity and acceleration between keyframes can determine whether a character is lethargic, excited or nervous. According to [TJ81], personalities are defined more by their movements than by their appearance, which substantiates our idea of distracting from a required mechanical design by natural motion. A *Secondary Action* is often used to emphasize the main action and thus give more life to a scene. Applied to quadrotor motion, it could yaw its heading to the next keyframe to give a more life-like impression. *Exaggeration* presents character features in a more extreme form in order to overcome the static appearance of drawings that perfectly imitate reality. A quadrotor could perform a tornado-like yaw spin right before a faster movement or through the whole trajectory. *Anticipation* helps the audience preparing for actions and to make them appear more natural. This is done by special moves that anticipate the upcoming action, just as a bow draw anticipates the arrow shot.

In order to avoid that users are surprised or frightened by sudden movements of the quadrotor and thus possibly intimidated in their interaction, we applied the principle of *Anticipation* along with the principles of *Slow In and Slow Out* and *Arcs*. The result is shown in Fig. 3, where the motion of the quadrotor is anticipated by extrapolating the first piecewise polynomial by a user-defined time coefficient. Thus a counter movement is created right before the start of the actual motion. Extrapolating by large numbers is not predictable, but low coefficients achieve results well suited to the intend of anticipation. Using polynomial extrapolation over linear extrapolation does not only transport information about the next keyframe's position but also adds velocity and acceleration information. Beyond that, a counter rotation is added by extrapolating the heading angle. Evaluating improvement of interaction quality by allowing the user to anticipate the quadrotor's next movement as well as adding further animation principles will be part of our future work.

Rotor sound plays an important but subordinate role and should be kept in mind during the design process. Parameter tuning of the MPC influences smoothness, aggressiveness and accurateness of how a trajectory is tracked. With certain controller settings, it is even possible to fly a smooth trajectory with irritating, unsteady rotor noise, which can be unpleasant for humans. In order to evaluate this during the design stage, we have introduced synthetic sound as part of the user interface as described in Section 2.

3.3 Evaluation

While evaluation of trajectory controllers address the robots's spatial and temporal deviation from the planned trajectory, there are several options for the evaluation of trajectories in terms of quality of user experience and quadrotor acceptance. User tests can be carried out directly in the testbed using real hardware or remotely using videos. Testing users on site is more elaborate to conduct, but gives the users a more realistic experience than online surveys. Online surveys on the other hand are able to reach a much larger test group. A test video can be filmed from the user's or a third person's perspective or rendered from the visualization. Videos can also be extended by AR to gain immersion for more elaborate tests, where the use of real hardware would be too dangerous, costly or complex. How well the trajectory design described in Section 3.2 is understood and helps to improve interaction could be evaluated by rendering all the trajectory information of our 3D visualization during a flight in the testbed on the screen of an AR device, such as a head-mounted display.

4 Conclusion & Future Work

In this paper we presented *EAVE*, a software extension to design and evaluate trajectories along with an updated description of our quadrotor testbed *ICARUS* that is used to control real and simulated quadrotors. We provided an updated description of the core components of said testbed. *EAVE* was implemented with the aim to design trajectories that are able to convey intent and thus improve quality of human-quadrotor interaction. As an example, we have applied an animation technique called *Anticipation* to a given trajectory. Future work will include user tests on how well the interaction experience can be improved by adding anticipation along with further principles of animation to quadrotor trajectories. Also, we are currently working on a dynamic, situation-dependent human-quadrotor interaction scenario where the quadrotor's response is triggered by user behavior.

Bibliography

[Ca16] Cauchard, J. R.; Zhai, K. Y.; Spadafora, M.; Landay, J. A.: Emotion encoding in Human-Drone Interaction. In: HRI 2016. pp. 263–270, March 2016.

[Co13] Corke, Peter: Robotics, Vision and Control: Fundamental Algorithms in MATLAB. Springer Publishing Company, Incorporated, 1st edition, 2013.

[De18] Deng, Honghao; Li, Jiabao; Sayegh, Allen; Birolini, Sebastian; Andreani, Stefano: Twinkle: A Flying Lighting Companion for Urban Safety. In: TEI '18. pp. 567–573, 2018.

[Fa18] Falanga, D.; Foehn, P.; Lu, P.; Scaramuzza, D.: PAMPC: Perception-Aware Model Predictive Control for Quadrotors. In: IROS '18. pp. 1–8, Oct 2018.

[GT11] Gielniak, M. J.; Thomaz, A. L.: Generating anticipation in robot motion. In: 2011 RO-MAN. pp. 449–454, 2011.

[HA17] Hönig, Wolfgang; Ayanian, Nora: Flying Multiple UAVs Using ROS. In: Robot Operating System (ROS): The Complete Reference. Springer, pp. 83–118, 2017.

[HJ14] Hoffman, Guy; Ju, Wendy: Designing Robots with Movement in Mind. J. Hum.-Robot Interact., 3(1):91–122, February 2014.

[Ka17] Karjalainen, Kari Daniel; Romell, Anna Elisabeth Sofia; Ratsamee, Photchara; Yantac, Asim Evren; Fjeld, Morten; Obaid, Mohammad: Social Drone Companion for the Home Environment: A User-Centric Exploration. In: Proceedings of the 5th International Conference on Human Agent Interaction. HAI '17, p. 89–96, 2017.

[Li17] Lieser, M.; Tjaden, H.; Brylka, R.; Löffler, L.; Schwanecke, U.: A low-cost mobile infrastructure for compact aerial robots under supervision. In: 2017 European Conference on Mobile Robots (ECMR). pp. 1–6, Sept 2017.

[Lu14] Lupashin, Sergei; Hehn, Markus; Mueller, Mark W; Schoellig, Angela P; Sherback, Michael; D'Andrea, Raffaello: A platform for aerial robotics research and demonstration: The Flying Machine Arena. Mechatronics, 24:41–54, 2014.

[Mi10] Michael, Nathan; Mellinger, D.; Lindsey, Q.; Kumar, V.: The GRASP Multiple Micro-UAV Testbed. Robotics Automation Magazine, IEEE, 17(3):56–65, Sept 2010.

[MK11] Mellinger, D.; Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: ICRA 2011. pp. 2520–2525, May 2011.

[RP12] Ribeiro, T.; Paiva, A.: The illusion of robotic life: Principles and practices of animation for robots. In: HRI '12. pp. 383–390, 2012.

[TJ81] Thomas, F.; Johnston, O.: The Illusion of Life: Disney Animation. Disney Editions, 1981.

[Tj19] Tjaden, Henning: Robust Monocular Pose Estimation of Rigid 3D Objects in Real-Time. PhD thesis, Johannes Gutenberg University Mainz, Germany, 2019. `https://nbn-resolving.org/urn:nbn:de:hebis:77-diss-1000025478`, last accessed 06 May 2020.

[Wa18] Walker, Michael; Hedayati, Hooman; Lee, Jennifer; Szafir, Daniel: Communicating Robot Motion Intent with Augmented Reality. In: HRI '18. Association for Computing Machinery, New York, NY, USA, p. 316–324, 2018.

[Ye17] Yeh, Alexander; Ratsamee, Photchara; Kiyokawa, Kiyoshi; Uranishi, Yuki; Mashita, Tomohiro; Takemura, Haruo; Fjeld, Morten; Obaid, Mohammad: Exploring Proxemics for Human-Drone Interaction. In: HAI '17. ACM, New York, NY, USA, pp. 81–88, 2017.