Real-Time Motion Planning for Aerial Videography With Dynamic Obstacle Avoidance and Viewpoint Optimization

Tobias Nägeli, Javier Alonso-Mora, Alexander Domahidi, Daniela Rus, and Otmar Hilliges

***Abstract*—We propose a method for real-time trajectory generation with applications in aerial videography. Taking framing objectives, such as position of targets in the image plane, as input, our method solves for robot trajectories and gimbal controls automatically and adapts plans in real time due to changes in the environment . We contribute a real-time receding horizon planner that autonomously records scenes with moving targets, while optimizing for visibility under occlusion and ensuring collision-free trajectories. A modular cost function, based on the reprojection error of targets, is proposed that allows for flexibility and artistic freedom and is well behaved under numerical optimization. We formulate the minimization problem under constraints as a finite horizon optimal control problem that fulfills aesthetic objectives, adheres to nonlinear model constraints of the filming robot and collision constraints with static and dynamic obstacles and can be solved in real time. We demonstrate the robustness and efficiency of the method with a number of challenging shots filmed in dynamic environments including those with moving obstacles and shots with multiple targets to be filmed simultaneously.**

***Index Terms*—Intelligent cinematography, MPC, path planning.**

I. Introduction

ROBOTICS and micro-aerial vehicles in particular are rapidly becoming an end-user facing technology. In particular, the application domain of filming with aerial vehicles currently receives great interest from industry and consumers. Professional camera teams leverage consumer-grade robots to create stunning visuals that previously required a helicopter and expensive camera gear. However, manually flying quadcopters remains a surprisingly hard task. Furthermore, automated flight modes in current commercial offerings are restricted to simple circling or target following.

Several algorithms have been proposed for the planning of quadcopter trajectories [1]–[3], taking both aesthetic objectives and the physical limits of the robot into consideration. These methods make the task of trajectory planning easier for non-experts. However, current approaches employ global optimization techniques, planning the entire trajectory a priori. While [2] allows to adjust the velocity along the planned trajectory at execution time, none of the existing methods are capable of re-planning a suitable trajectory in real-time for example, to avoid collisions with dynamic obstacles or to film targets that move in unpredictable ways (i.e., human actors).

In this letter we propose an approach to the automatic generation of quadcopter and gimbal controls in real-time while ensuring physical feasibility. In particular, we contribute a fast receding horizon planner based on numerical optimization for automatic aerial cinematography. The method takes high-level aesthetic objectives given by a cameraman as input and automatically records the scene while the targets move in an a priori unknown way. The system fulfils the high-level framing objective, in the least squares sense, while ensuring collision-free paths. The resulting appearance of targets is specified via set-points in screen space (extending ideas outlined in [5]) and the method minimizes the re-projection error alongside the viewing direction and scale of the target projections in real-time.

We formulate this cost minimization problem under constraints as a finite horizon model predictive control (MPC) nonlinear program with the following properties: (i) a main utility function to fulfill - as close as possible - the specifications from the user in a dynamic scene; (ii) input and state constraints to respect the dynamics and model constraints of the aerial vehicle; (iii) constraints for collision avoidance with obstacles and the targets being recorded. The resulting optimization problem can be solved numerically with state-of-the-art solvers in real-time. The inputs computed are applied for the first time step, and the optimization is repeated at the next sampling instance with updated information about the quadrotor state, obstacles and target positions. We believe that the method is general enough and could be adapted to other tasks for aerial vehicles which include collision avoidance with respect to moving obstacles.

Manuscript received September 10, 2016; accepted January 18, 2017. Date of publication February 8, 2017; date of current version May 16, 2017. This letter was recommended for publication by Associate Editor F. Kendoul and Editor J. Roberts upon evaluation of the reviewers' comments. This work was supported in part by the Microsoft Research, SMARTS N00014-09-1051, pDOT ONR N00014-12-1-1000, and the Toyota Research Institute.

T. Nägeli and O. Hilliges are with the AIT Lab, Department of Computer Science, ETH Zurich, Zurich 8092, Switzerland (e-mail: naegelit@inf.ethz.ch; otmar.hillige@inf.ethz.ch).

J. Alonso-Mora is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA, and also with the Delft Center for Systems and Control, Delft Technical University, Delft 2628CD, The Netherlands (e-mail: J.AlonsoMora@tudelft.nl).

A. Domahidi is with embotech GmbH, Zurich 8092, Switzerland (e-mail: domahidi@embotech.com).

D. Rus is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: rus@csail.mit.edu).

This letter has supplementary downloadable material available at http:// ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LRA.2017.2665693

II. Related Work

Quadrotor trajectory generation is a well studied problem and various approaches have been proposed, including early work on MPC for collision avoidance applied to aerial vehicles [6], [7], global forward planning approaches to generate minimum snap trajectories [8], for generating collision-free trajectories via convex optimization [9] or for state interception with a quadrotor [10] based on a convex MPC formulation. Outside of the field of aerial vehicles, constrained optimization methods have also been proven to be powerful tools for trajectory generation [11]. We also formulate our problem in the MPC framework to attain real-time performance, using the full non-linear dynamics of the quadrotor and extending trajectory generation to include collision avoidance with dynamic targets and to respect cinematographic objectives.

Automatic camera control has been studied extensively in the context of virtual environments in computer graphics. We refer to the comprehensive review in [12], with the majority of methods using discrete optimization formulations. Notably in virtual environments it has become somewhat of a standard to define viewing constraints in screen-space [5], [13], [14]

However, virtual environments are not limited by real-world physics and robot constraints and hence can produce arbitrary camera trajectories, velocities and viewpoints.

Extending the work on trajectory generation [8], several algorithms for planning aerial video shots of (mostly) static scenes [1]–[3] exist. Joubert *et al.*'s method [2] allows an operator to adjust the velocity of the MAV at execution time to keep moving targets in-frame. However, the method cannot re-plan the trajectory in cases where the target does deviate from an a-priori known path or in cases were dynamic obstacles temporarily obstruct the original trajectory. Lino *et al.* [15] propose a method to generate camera paths in virtual environments that ensure visibility of the faces of two actors in the image. This concept was later extended to quadrotors [16], albeit limited to obstacle free environments. To record a single moving target with multiple quadrotors, [17] introduced a particle swarm method for formation control. The method is again limited to obstacle-free environments and only tested in simulation. We propose a general constrained optimization approach for automatic viewpoint and trajectory computation allowing for multiple moving targets and obstacles in the environment. The approach can be solved in real-time.

Our method also accounts for occlusions and loosely relates to the fields of target tracking [18], visual servoing [19], active vision [20] and persistent monitoring [21], where our focus is on videographing a set of moving targets.

III. Preliminaries

A. Cinematographic Objectives

While film making is a form of art and relies on human creativity and intuition, many aspects of it have been studied and categorized forming a 'grammar' of film [4]. Fig. 2 summarizes the most important aspects. We later formalize these

Fig. 1. We propose a real-time method to film shots of dynamic scenes automatically. The user specifies target positions on screen. We solve for robot trajectories and gimbal controls, producing footage that adheres to aesthetic constraints, minimizes visual occlusion and produces collision-free trajectories in dynamic scenes.

Fig. 2. Illustration of cinematographic framing constraints: size, viewing angle and position on screen (from left to right).

mathematically for use in an cost minimization algorithm. In particular, we are interested in framing objectives – that is formal rules that specify how objects should appear on the screen. The first important notion is that of distance. Shots can be categorized into five types of shots (close-up, close-medium, full and long shots), see Fig. 2 , left. A further important aspect is that of the relative viewing angle, which can also be categorized into ranges of pan- and tilt-angles), see Fig. 2, middle. Finally, the screen position of a filmed target is important in order to create aesthetically pleasing footage. In particular the rule of thirds (cf. [22]) prescribes the placement of significant vertical and horizontal elements along the horizontal and vertical thirds (Fig. 2, right).

1) Shot Framing Requirements: From the above film grammar we can extract objectives for an optimization method. In particular, we consider (i) the *screen position* of targets, where we seek to minimize the distance between the desired position on screen and the projection of the actual (3D) target position onto the screen. (ii) To relate shot distances to positions, we seek to optimize the *projected size* of a given target. (iii) Furthermore, we require the algorithm to consider the *relative viewing angle* between target and camera center, for example to keep the face of a person in view. (iv) Finally, we require the algorithm to account for an externally set *camera*

pose, which could be specified directly by the user or a high-level global planning algorithm.

B. Notation and Coordinate Frames

We denote the set of positive definite matrices of size n by $\mathbb{S}^n_{++}$ and the set of positive semi-definite matrices of size n by $\mathbb{S}^n_{+}$. Given a vector $\boldsymbol{x} \in \mathbb{R}^n$, we define the norm square of the norm as $\|\boldsymbol{x}\|_P \triangleq \boldsymbol{x}^T P \boldsymbol{x}$ for $P \in \mathbb{S}^n_{++}$. Vectors are denoted in bold. If not indicated differently, vectors are expressed in the standard ENU coordinate system. We denote points in 3D as $\boldsymbol{p}$ with a name as subscript (e.g., $\boldsymbol{p}_h$ for $\boldsymbol{p}_{\text{head}}$). A relative vector between two points $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ is denoted as $\boldsymbol{r}_{ab}$. A supperscript $\boldsymbol{r}^c_{ab}$ indicates the vector $\boldsymbol{r}_{ab}$ is expressed in frame C (without subscript the vector is expressed in the inertial frame I. Pixel coordinates are denoted as μ. Desired setpoints are indicated with a subscript $(.)_d$.

Rotation matrices performing rotations from frame A to frame B are denoted by $\boldsymbol{R}_{BA} = \boldsymbol{R}(\bar{q}_{BA}) \in SO(3)$, where $\bar{q}_{BA}$ is the corresponding quaternion. We adhere to the JPL quaternion definition [23] and denote a quaternion by $\bar{q} = [q_x \mathbf{i} + q_y \mathbf{j} + q_z \mathbf{k} + q_w] = [\boldsymbol{q}, q_w]^T$. Quaternion multiplication is denoted by $\otimes$. Orientation errors are described in $so(3)$, the tangent space of $SO(3)$, and are written as $\delta\theta$.

C. Robot and Camera Model

Our method is agnostic to the quadrotor hardware. However, in our experimental setup we use a Parrot Bebop 2 and therefore we will use its model in our derivations. In general, the pose of a quadrotor can be described by its position $\boldsymbol{p}_q \in \mathbb{R}^3$ and orientation $\bar{q}_q \in SO(3)$. While the camera is not directly attached to the center of the quadrotor, experimentally we have found it to be sufficient to assume the camera position $\boldsymbol{p}_c$ and the quadrotor's position $\boldsymbol{p}_q$ to be identical. In the following consider $\boldsymbol{p}_c = \boldsymbol{p}_q$. The set of feasible states is denoted by $\mathcal{X}$. The state of the system is defined by

$$x = [\boldsymbol{p}_c, v_x, v_y, \bar{q}_q]^T \in \mathbb{R}^5 \times SO(3), \tag{1}$$

where v_x, v_y are the velocities of the quadrotor along the x and y axis.

The inputs accepted by Parrot's SDK are roll angle ϕ_q, pitch angle θ_q, angular speed around the body-z axis ω_{q_z} and velocity in the body-z axis $v_z = \dot{p}_{q_z}$. The camera is attached to the robot via a pan-tilt gimbal (in case of the Bebop this is a software gimbal) where in this letter only the tilt part is used. Due to pre-stabilized gimbal dynamics, it can be modeled using a separate pitch angle θ_g.

The set of feasible inputs is denoted by $\mathcal{U}$. The inputs to the system are

$$\boldsymbol{u} = [\phi_q, \theta_q, \omega_{q_z}, v_z, \theta_g] \in \mathbb{R}^5. \tag{2}$$

The sets $\mathcal{X}$ and $\mathcal{U}$ are given by the physical limits of the environment (e.g. $p_z > 0$) and by the internal constraints of the Parrot Bebop 2 (e.g. maximal vertical and horizontal velocities, maximal roll pitch angle). The limits are described in the documentation of the Parrot SDK [1] The full non-linear model and its states are not exposed via the SDK, hence we use the following approximation for the dynamics. The system model of the flying camera is given by non-linear ordinary differential equations for the acceleration in the x and y axis, $\ddot{\boldsymbol{p}}_{c_{x,y}}$, and the velocity in the body-z axis, $\dot{p}_{c_z}$, i.e.

$$\ddot{\boldsymbol{p}}_{c_{x,y}} = R_\psi(\bar{q}_q) \begin{bmatrix} -\tan(\theta_q) \\ \tan(\phi_q) \end{bmatrix} g \quad \text{and} \qquad \dot{p}_{c_z} = v_z, \tag{3}$$

where $g = 9.81 \frac{m}{s^2}$ is the earth's gravity and $R_\psi(\bar{q}_q) \in SO(2)$ is the rotation matrix only containing the yaw rotation of the quadrotor.

The camera's attitude is given by the quaternion

$$\dot{\bar{q}}_c = \bar{q}_g(\theta_g) \otimes (\bar{q}_q \otimes \bar{q}(\omega_{q_z})), \tag{4}$$

where $\bar{q}_g(\theta_g)$ is the quaternion defined by the gimbal pitch angle θ_g and $\bar{q}(\omega_{q_z})$ is defined as

$$\bar{q}(\omega_{q_z}) = \frac{1}{2\epsilon} \begin{bmatrix} 0 \\ 0 \\ \omega_{q_z} \sin(\epsilon) \\ 2\cos(\epsilon) \end{bmatrix} \quad \text{with} \quad \epsilon = \|\omega_{q_z}\|. \tag{5}$$

In the remainder of the letter the continuous non-linear quadrotor model $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ is discretized using a standard forward Euler approach: $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$ [24].

D. Target

Model We denote the position of each actor, or moving target, by $\boldsymbol{p}_t \in \mathbb{R}^3$, with K the number of targets. We then assume that the human motion is based on a constant velocity model $\ddot{\boldsymbol{p}}_t = \boldsymbol{q}_\eta$ where $\boldsymbol{q}_\eta \sim \mathcal{N}(0, \sigma_t)$ is Gaussian noise. Given the current observations, we predict future positions of the actor with a standard linear Kalman filter [24].

IV. Trajectory Generation for Viewpoint Optimization

The goal of our algorithm is to find a feasible and locally optimal trajectory for the quadrotor and gimbal control inputs in real-time. The method produces imagery that minimizes the deviation with respect to the high-level image space specifications of Section III-A1 and generates trajectories that guarantee collision-free motion with respect to dynamic obstacles.

A scene is formed by K targets to be filmed, e.g. the faces of actors as seen in Fig. 3 . Each target $i \in \{1, \ldots, K\}$ is modeled as a sphere at position $\boldsymbol{p}_h \in \mathbb{R}^3$, with orientation ψ and of radius h_h. Note that for clarity we typically omit the additional subscript i unless explicitly dealing with several targets. Without loss of generality, in the following we only consider the orientation ψ around the z-axis, to model the gaze direction of the face (see Fig. 3). For collision avoidance and visibility, we consider that each target (not only the face) occupies a physical space defined by an ellipsoid S_{Ω_s} centered at $\boldsymbol{p}_t = \boldsymbol{p}_h + (\frac{h_h}{2} - \frac{h}{2})\boldsymbol{e_z}$,

[1] http://developer.parrot.com/

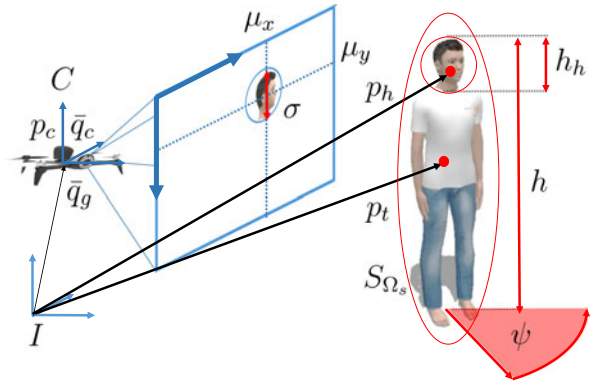

Fig. 3. Coordinate frames and physical quantities used in our method, here shown on the example of a human actor. The pose of the flying camera is given by its position $\boldsymbol{p}_c$, its orientation $\bar{q}_c$ and the orientation of the gimbal $\bar{q}_g$. The target is modeled by an ellipsoid S_{Ω_s} centered at $\boldsymbol{p}_t$. The flying camera avoids collisions with this volume. In this case the objective is to record the head of the target, modeled as an oriented sphere with direction given by ϕ, size h_h and position $\boldsymbol{p}_h$. The projection of the head onto the camera image space has size σ and is centered at μ_x, μ_y.

where h denotes the physical height of the target from the ground and $\boldsymbol{e_z}$ the vertical vector, illustrated in Fig. 3 for the case of a human actor .

A. Method Overview

We propose a receding horizon MPC formulation for trajectory generation given by the following variables, cost terms and constraints. Denote by Δ_t the time step and by N the time horizon of the MPC problem . At each time step a local trajectory of duration $N\Delta_t$ is computed. The first input is then applied. The optimization problem is re-solved at every sampling instance, leading to closed loop performance. This makes the approach robust against model uncertainties and unpredictable disturbances. At the next timestep $k+1$ we use the estimated states $\hat{\boldsymbol{x}}_0$ of the quadrotor and all targets as the initial state $\boldsymbol{x}_0$ and use the trajectory from timestep k as an initialization for the solver.

1) Variables: The optimization variables are the following. (a) The states $\boldsymbol{x}_{0:N}$, which include the initial state x_0, the state trajectory $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-1}\}$ and the final state x_N. (b) The control inputs $\boldsymbol{u}_{0:N-1} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-1}\}$. And (c) the additional slack variables $\boldsymbol{s}_{0:N} = \{\boldsymbol{s}_0, \ldots, \boldsymbol{s}_N\}$, one for each collision constraint and described in the forthcoming Section IV-D. The set of optimization variables is denoted by

$$\mathbf{z} = \{\boldsymbol{x}_{0:N}, \boldsymbol{u}_{0:N-1}, \boldsymbol{s}_{0:N}\}. \tag{6}$$

2) Cost Terms: The final quantity we seek to minimize is the weighted sum of cost terms consisting of image framing, visibility and pose costs over all N stages of the planned trajectory.

3) Constraints: We include constraints to avoid collisions with moving obstacles and moving targets and to respect the dynamic model of the quadrotor introduced in Section III-C.

B. Viewpoint Optimization Problem

At the core of our algorithm lie a number of cost terms $c_{\text{image}}(\boldsymbol{x})$, $c_{\text{angle}}(\boldsymbol{x})$ and $c_{\text{size}}(\boldsymbol{x})$ which describe the deviation from the desired position of each target's projection on the image plane $\mu_{x,y\,d}$, from the viewing angle $\boldsymbol{a}_d$ and from the projection size σ_d . Each of these cost terms is computed for each state $k \leq N$ of the planning horizon and for each target $i \leq K$. For simplicity of exposition, the derivation of the costs is described for a general target. We also define the vector $\boldsymbol{r}_{\text{ch}}$ which is the relative vector between the target and the camera as well as the rotation into the camera frame of $\boldsymbol{r}_{\text{ch}}$ which is denoted as $\boldsymbol{r}^c_{\text{ch}}$:

$$\boldsymbol{r}_{\text{ch}} = \boldsymbol{p}_h - \boldsymbol{p}_c \quad \text{and} \quad \boldsymbol{r}^c_{\text{ch}} = R(\bar{q}_c)\boldsymbol{r}_{\text{ch}},$$

1) Image Space Positions: Given the desired position of each target's projection on the image plane $\mu_{x,y\,d}$, the pixel coordinates $\boldsymbol{m}_d$ are computed via the camera intrinsics with focal point $C_{x,y}$ and focal length $\mathbf{f} = [f_x, f_y]$:

$$\boldsymbol{m}_d = \begin{bmatrix} (\mu_{x\,d} - C_x)\, f_x \\ (\mu_{x\,d} - C_y)\, f_y \end{bmatrix} \text{ with } \mu_{(x,y)\,d} \in \{0, 2C_{(x,y)}\},$$

Consider the vector $\boldsymbol{r}^c_d = [\boldsymbol{m}_d, 1]^T \in \mathbb{R}^3$ pointing from the camera center through the desired pixel location $\boldsymbol{m}_d$ in the image. We compute the quadratic image space location cost $c_{\text{image}}(\boldsymbol{x})$ using the residual given by the difference between the ray $\boldsymbol{r}^c_{\text{ch}}$ from the camera to the target and the desired direction $\boldsymbol{r}^c_d$,

$$c_{\text{image}}(\boldsymbol{x}) = \|\boldsymbol{\rho}_m\|_{Q_m} \quad \text{with} \quad \boldsymbol{\rho}_m = \frac{\boldsymbol{r}^c_{\text{ch}}}{\|\boldsymbol{r}^c_{\text{ch}}\|} - \frac{\boldsymbol{r}^c_d}{\|\boldsymbol{r}^c_d\|} \tag{7}$$

2) Projection Size: The size σ of the target in the image plane is computed by projecting the 3D sphere of diameter h_h to a circle in the 2D image plane. The target-size cost $c_{\text{size}}(\boldsymbol{x})$ is computed using the difference between the projected σ and the desired size σ_d.

$$c_{\text{size}}(\boldsymbol{x}) = \|\sigma - \sigma_d\|_{Q_\sigma} \quad \text{with} \quad \sigma = \frac{h_h \|\mathbf{f}\|}{\|\boldsymbol{r}^c_{\text{ch}}\|}. \tag{8}$$

3) Relative Viewing Angle: Given the desired viewing angles θ_d and ψ_d , and the current orientation ψ of the target, we define the desired viewing orientation $\boldsymbol{a}_d$ between the center of the camera and the target:

$$\boldsymbol{a}_d = \left[\sin\theta_d \cos(\psi_d + \psi),\ \sin\theta_d \sin(\psi_d + \psi),\ \cos\theta_d\right]^T$$

Because the angle ψ_d is relative to the target we add the current rotation of the target ψ to obtain the setpoint. The view angle cost $c_{\text{angle}}(\boldsymbol{x})$ is computed using the error between the desired (relative to the target's normal) and the current viewing angle:

$$c_{\text{angle}}(\boldsymbol{x}) = \|\boldsymbol{\rho}_a\|_{Q_a} \quad \text{with} \quad \boldsymbol{\rho}_d = -\frac{\boldsymbol{r}^c_{\text{ch}}}{\|\boldsymbol{r}^c_{\text{ch}}\|} - \frac{\boldsymbol{a}_d}{\|\boldsymbol{a}_d\|}. \tag{9}$$

4) Camera Pose: In addition to local viewpoint optimization we can also consider - but not required - the residual in the camera's position and orientation dynamics. Here we assume that either the user or a global path planning algorithm can

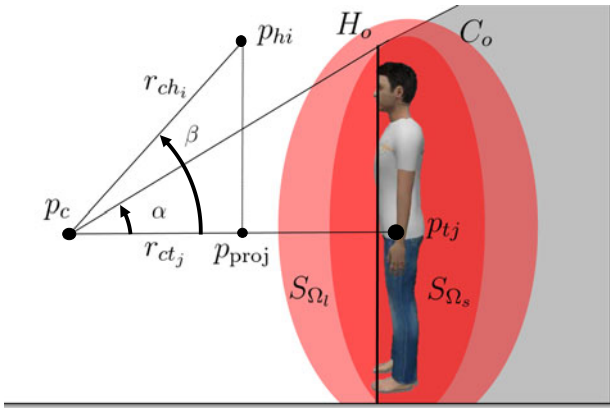

Fig. 4. Schematic illustration of occlusion minimization based on viewpoint dependent horizon culling.

specify a desired pose $\boldsymbol{p}_{cd}$ and $\bar{q}_{qd}$ for the flying camera, which is compared to the actual pose of the camera given by $\boldsymbol{p}_c$ and $\bar{q}_c$. The position residual is given by the euclidean difference. The orientation residual $\boldsymbol{\rho}_\theta$ is computed building the error quaternion and projecting it into the tangent space $so(3)$ of the $SO(3)$ manifold:

$$\boldsymbol{\rho}_\theta = \delta\theta = \begin{bmatrix} \bar{q}_{e_x} & \bar{q}_{e_y} & \bar{q}_{e_z} \end{bmatrix} \in so(3) \;\text{ with }\; \bar{q}_e = \bar{q}_c \otimes \bar{q}_{c_d}^{-1}.$$

where the error quaternion is defined as $\bar{q}_e = \bar{q}_c \otimes \bar{q}_{ref}^{-1} \in SO(3)$, with $\bar{q}_{ref}$ the desired camera orientation. This yields the camera pose cost c_p: $c_{\text{pose}}(\boldsymbol{x})$:

$$c_{\text{pose}}(\boldsymbol{x}) = \|\boldsymbol{p}_{cd} - \boldsymbol{p}_c\|_{Q_p} + \|\boldsymbol{\rho}_\theta\|_{Q_\theta}. \tag{10}$$

C. Occlusion Minimization

To allow for scenes containing objects that can move into the line-of-sight, we have to account for target visibility. Dynamic objects are modelled as ellipsoids S_{Ω_s} and we observe that point visibility can be decided by 'horizon culling', illustrated in In Fig. 4. The lines from the camera center $\boldsymbol{p}_c$ to the ellipsoid's tangent define the horizon plane H_o. All points on the intersection of H_o and S_{Ω_s} are on the horizon, all points in the shaded region C_o are below the horizon and therefore not visible from the camera's viewpoint. To this end we adopt a fast visibility test [25], which can be summarized as: (i) determine if a point is in front H_o (visible) and (ii) for those behind H_o whether they are within the infinite cone formed by the tangent lines (C_o).

To determine if a target $\boldsymbol{t}_i$ is in front of H_o we compute:

$$\begin{aligned} \boldsymbol{r}_{\text{ch}_i} &= \boldsymbol{p}_{h_i} - \boldsymbol{p}_c, \\ \boldsymbol{r}_{\text{ct}_j} &= \boldsymbol{p}_{tj} - \boldsymbol{p}_c, \\ p_{\text{proj}} &= \boldsymbol{r}_{\text{ch}_i}^T \boldsymbol{r}_{\text{ct}_j}. \end{aligned}$$

Where $\boldsymbol{r}_{\text{ch}_i}$ is the vector to the target, $\boldsymbol{r}_{\text{ct}_j}$ the vector to the center of ellipsoid S_{Ω_s} and p_{proj} is the component of $\boldsymbol{r}_{\text{ch}_i}$ in the direction of $\boldsymbol{r}_{\text{ct}_j}$. If $p_{\text{proj}} > \boldsymbol{r}_{\text{ct}_j}^T \boldsymbol{r}_{\text{ct}_j} - 1$ then the point is behind the plane.

For these cases we can determine if the point falls into the cone by comparing the opening angles α and β. Avoiding costly trigonometric functions we use directly the approach given in [25]. This gives us the visibility score:

$$d_v = \frac{p_{\text{proj}}}{\boldsymbol{r}_{\text{ch}_i}^T \boldsymbol{r}_{\text{ch}_i}} > \boldsymbol{r}_{\text{ct}_j}^T \boldsymbol{r}_{\text{ct}_j}.$$

And we define the visibility cost $c_{\text{vis}}(\boldsymbol{x})$:

$$c_{\text{vis}}(\boldsymbol{x}) = \begin{cases} \|d_v\|_{Q_v} & \text{if } d_v > 0 \text{ and } p_{\text{proj}} > \boldsymbol{r}_{\text{ct}_j}^T \boldsymbol{r}_{\text{ct}_j} - 1, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

D. Collision Avoidance

A final concern of the planning algorithm is to generate collision-free trajectories. Here we assume known geometry of the scene and real-time position information of moving obstacles (targets are also considered as moving obstacles). To improve safety and at the same time guarantee performance and responsiveness of the system we adopt a two layered approach: (i) a potential field to repel the robot from obstacles and (ii) a hard constraint to stay outside a smaller enclosing ellipsoid S_{Ω_s} to enforce collision-free motion. We denote the relative vector between the center of the target and the camera by $\boldsymbol{r}_{\text{ct}} = \boldsymbol{p}_t - \boldsymbol{p}_c$.

1) Collision Avoidance Potential Field: First, we employ a potential field cost which becomes active as soon as the camera enters an ellipsoid S_{Ω_l} containing the target and with a buffer zone. This term helps to maintain a safe distance from moving obstacles and leaves some buffer for the un-modeled dynamics of the quadrotor and human motion. Formally, the distance to this ellipsoid is given by

$$d_c = \boldsymbol{r}_{\text{ct}}^T \Omega_l \boldsymbol{r}_{\text{ct}} - 1.$$

Which produces the cost term:

$$c_{\text{coll}}(\boldsymbol{x}) = \begin{cases} \|d_c\|_{Q_c} & \text{if } d_c > 0 \\ 0 & \text{else} \end{cases}. \tag{12}$$

2) Collision Avoidance Constraint: Second, we introduce a non-linear constraint which becomes active within the collision ellipsoid S_{Ω_s} . This constraint guarantees collision-free motion, i.e. the position of the quadrotor must remain outside of the S_{Ω_s} at all times. Formally,

$$\boldsymbol{r}_{\text{ct}}^T \Omega_s \boldsymbol{r}_{\text{ct}} > 1 - \boldsymbol{s}_c, \tag{13}$$

We introduce the slack variables $\boldsymbol{s}_c$ and, therefore, we adopt a soft-constrained approach for the collision-avoidance constraints to ensure that the optimizer always returns an answer in practice. It can be shown that under sufficiently high penalization of (a linear norm of) the slack variables $\boldsymbol{s}_c$, the solution of the hard-constrained problem is recovered when it exists; otherwise, a plan with minimum deviation will be computed by the optimizer [26].

E. MPC formulation

The locally optimal trajectory and inputs for the quadrotor are then computed by solving a constrained optimization problem, which consists of the cost terms introduced in (7)–(12) and the constraints of (13) , stacked together over all targets, obstacles and time-steps of the controller. Without loss of generality, we consider the obstacle set to be equal to the set of targets, since an obstacle can be treated as a target with only the collision avoidance constrained active - and no other cost term. The full constrained optimization problem solved as MPC non-linear program is then given by:

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{s}} \quad & w_N^T c(\boldsymbol{x}_N, \boldsymbol{u}_N) + \Sigma_{k=0}^{N-1} w^T c(\boldsymbol{x}_k, \boldsymbol{u}_k) + \lambda \|\boldsymbol{s}_k\|_\infty \\
& \boldsymbol{x}_0 = \hat{\boldsymbol{x}}_0 && \text{(Initial State)} \\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), && \text{(Dynamics)} \\
& \boldsymbol{r}_{\mathrm{ct}}^T \Omega_s \boldsymbol{r}_{\mathrm{ct}} > 1 - \boldsymbol{s}_k, && \text{(Collision Avoidance)} \\
& \boldsymbol{r}_{\mathrm{ct}} = g(\boldsymbol{x}_k), \\
& \boldsymbol{x}_k \in \mathcal{X}, && \text{(State Constraints)} \\
& \boldsymbol{u}_k \in \mathcal{U}, && \text{(Input t Constraints)} \\
& \boldsymbol{s}_k \geq 0, && \text{(Slack Constraints).} \quad (14)
\end{aligned}
$$

where the stage cost function $c(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is defined as:

$$
(\boldsymbol{x}_{k\,k}) = \begin{bmatrix} c_{\text{image}} & c_{\text{size}} & c_{\text{angle}} & c_{\text{coll}} & c_{\text{vis}} & c_{\text{pose}} \end{bmatrix}^T_{(\boldsymbol{x}_k, \boldsymbol{u}_k)}
$$

Where w are weights that can be set interactively by the user to control influence of the different framing constraints. This problem can readily be formulated in standard software, e.g. FORCES Pro [27], [28], and efficient code be generated to solve it in real-time.

V. Experimental Results

A. Hardware Setup

We use a Parrot Bebop2 with an integrated electronic gimbal. Robot and targets are tracked with a Vicon motion capturing system. Experiments were conducted on a standard desktop PC (Quadcore Intel i7 CPU@3.5 GHz) .

B. Experiments

We conducted five experiments to evaluate the performance of our proposed method under dynamic conditions. In our experiments we used a maximum number of three moving targets, although this is not an inherent limitation of the method . In the following results the boxes rendered on the images represent the setpoints. The size of a box determines the desired size of the projected target ellipse. The color coding is: Target 1 (red), Target 2 (purple) and Target 3 (blue).

Experiment 1 (Setpoint Change): First, we illustrate the effect of different framing objectives. A single target is filmed with different setpoints, including transitions from sitting to standing. Horizon length $N = 25$.

Experiment 2 (Single target with occlusion): Our second experiment demonstrates effectivity of the framing objectives (see (9)) and occlusion handling (see (11)). One face is set as target and two further actors move in and out of the line-of-sight. The goal is then to keep the main target always in view while minimizing occlusions from the other two targets. Horizon length $N = 25$.

Experiment 3 (Single target with collision avoidance): In the third experiment we modify above setup so that the two other targets walk randomly, including directly towards the quadrotor. Again the algorithm will try to keep the target in frame while avoiding collisions (see (12)). Horizon length $N = 25$.

Experiment 4 (Multi target framing objective): In our fourth setup we declare all three actors to be targets and the algorithm tries to keep all three faces appearing along a single line in the image. Horizon length $N = 25$.

Experiment 5 (Execution time): To evaluate the real-time performance we conducted two experiments. We measure the execution time per timestep along a full trajectory (solving over the entire horizon per timestep). First, we use a constant number of targets $K = 2$ and vary horizon length ($N = 25, N = 40, N = 55$). Second, we keep the horizon fixed $N = 25$ and vary the number of targets.

C. Results

Experiment 1: Fig. 5 shows the algorithm working with a single target $K = 1$ and a horizon length of $N = 25$. With a single target we change the setpoints to: 3/4 Frontal right, 3/4 frontal left, right screen position, left screen position. The accompanying video also shows the target moving.

Experiment 2: The sequence in Fig. 6 illustrates how at time $t = 0$ s the initial condition is met and the face is visible according to the framing objective. At time $t = 1$ s target 3 moves into the line-of-sight and occludes target 2. The robot moves smoothly to the closest pose in terms of framing objectives but restores line-of-sight $t = 3$ s.

Experiment 3: Fig. 7 shows representative views with the framing objective of creating a frontal shot of target 2. Although the targets 1 and 3 force the robot to perform multiple collision avoidance moves, the target remains in view and its screen space position remains relatively stable. Due to safety reasons collision avoidance takes the highest priority. Furthermore, collision avoidance (hard constraint) restricts the quadrotor motion stronger than the occlusion minimization (soft constraint).

Experiment 4: Fig. 8 shows representative frames from a shot with multi-target framing objective. Although there exists, except of some degenerated cases, a camera position with a zero re-projection error, the algorithm has to respect state constraints. In this case the quadrotor has to stay inside the physical room limits, reducing the range of motion drastically. Nevertheless all targets remain in view and the framing requirements are fulfilled as closely as possible.

Fig. 5. Exp. 1: The effect of viewpoint optimization under varying setpoints: 3/4 Frontal right, 3/4 frontal left, right screen position, left screen position (from left to right).

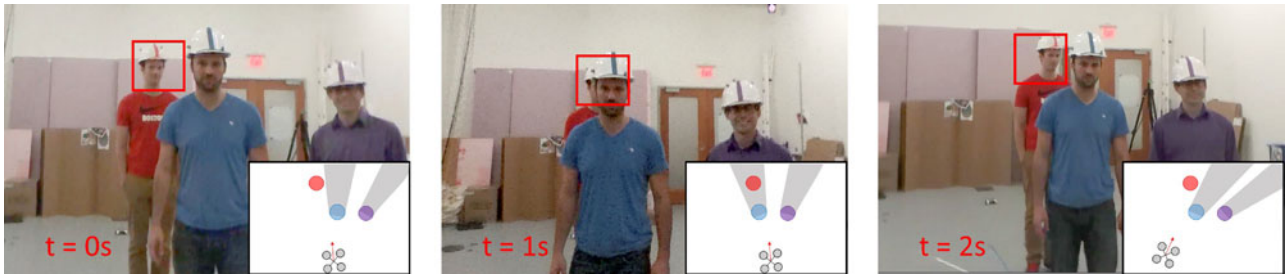

Fig. 6. Exp. 2: Effect of occlusion handling (onboard view). Entire duration is 3s. Inset shows robot position and non-visible areas.

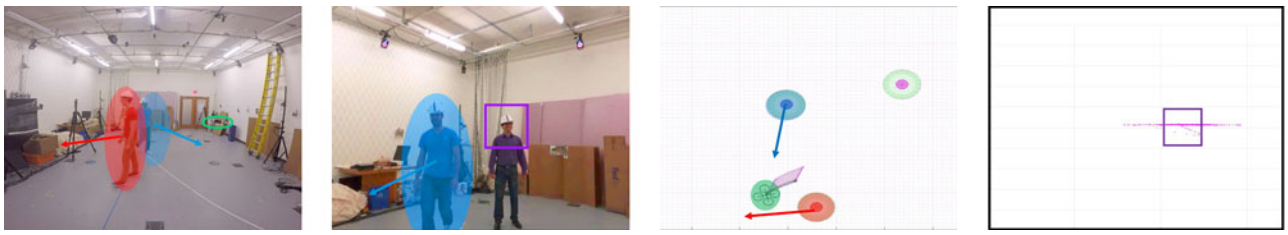

Fig. 7. Exp. 3: Framing & collision avoidance. From left to right: Offboard view (robot in green). Onboard view (target in purple). Visualization of collision ellipsoids. Target position residual, gimbal reduces error in x but yawing the robot is slow (large error in y).

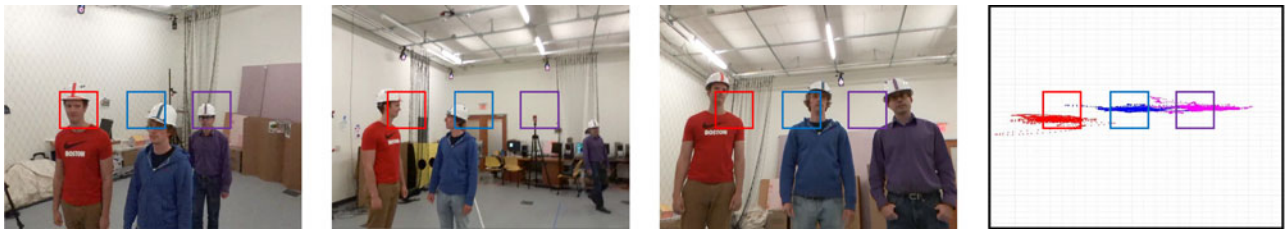

Fig. 8. Exp. 4: The Camera tries to keep the three targets in view. Distribution of the re-projection is shown (right).

Experiment 5: Fig. 9 plots the computation time for a trajectory with 2000 timesteps at each of which we solve (14) over the horizon length resulting in a total of $2000 \times N$ calls to the solver. We randomly vary the setpoints, ensuring that we don't initialize with unrealistic values to close to the solution. Furthermore, collision avoidance is turned on. The two experiments are parametrized: (i) First, a constant target number $K = 2$ and variable horizon length Fig. 9, top. The mean solve-times are: 0.01 s, 0.015 s and 0.025 s respectively. (ii) Second, a constant horizon $N = 25$ and a variable number of targets K Fig. 9, bottom. The mean solve-times are: 0.009 s, 0.011 s and 0.011 s. The computational time grows approximately linear with the length of the horizon. This result is expected according to the design of FORCES Pro. In general we see that the computation time for longer horizons and more targets also tend to have more variability in the solve-time. We think this is due to the existence of a unique solution for a single target, while for multiple targets, depending on the setpoints, the solution may only exist in the least squares sense.

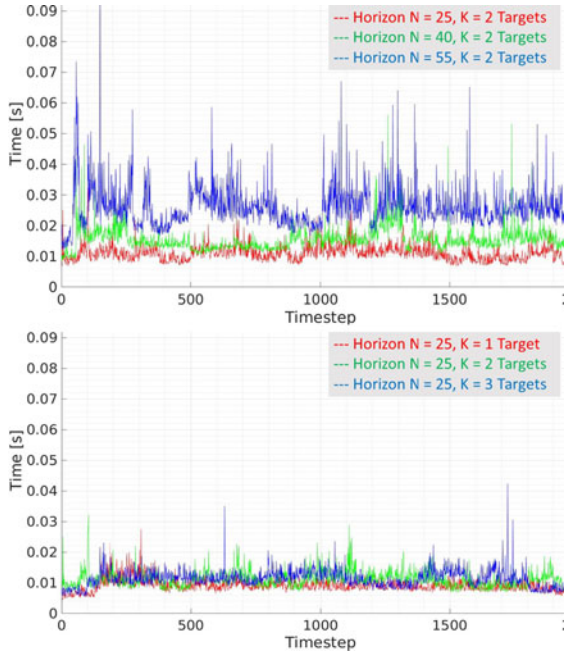

Fig. 9. Plots show solve time per evaluation of the MPC-problem (see (14)) as function of horizon length (N) and number of targets (K). Note that we solve over N stages for each timestep (y-axis). *Top*: We change desired setpoints randomly throughout the trajectory but keep number of targets fixed $K = 2$. Colors indicate different horizon length. *Bottom*: Fixed horizon length, while changing number of targets K.

VI. Conclusion

In this letter we propose a novel approach to the automatic generation of quadcopter and gimbal controls real-time videography. We introduce a trajectory generation method that solves for trajectory parameters from set-points defined in image-space, and a formulation of this problem that lends itself to an implementation as a receding horizon optimal control program with non-linear constraints which can be solved numerically with state-of-the art solvers in real-time. We have demonstrated in experiments with an aerial vehicle and multiple targets that the algorithm can satisfy framing constraints, derived from cinematographic rules, continuously minimize occlusions from dynamic objects in the environment and avoid collisions with these. Currently our algorithm accepts position and framing set-points from a user. In future work this could be handled by a global planning algorithm that generates inputs for entire scenes rather than individual shots. We believe that the algorithm is general and could be adapted to other tasks for aerial vehicles which include collision avoidance with respect to moving obstacles. We will release our code as open-source.

Acknowledgment

The authors would like to thank Parrot for hardware and tech support.

References

[1] C. Gebhardt, B. Hepp, T. Naegeli, S. Stevsic, and O. Hilliges, "Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, New York, NY, USA, 2016, pp. 2508–2519.

[2] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz, and P. Hanrahan, "An interactive tool for designing quadrotor camera shots," *ACM Trans. Graph.*, vol. 34, no. 6, 2015, Art. no. 238.

[3] M. Roberts and P. Hanrahan, "Generating dynamically feasible trajectories for quadrotor cameras," *ACM Trans. Graph.*, vol. 35, no. 4, 2016, Art. no. 61.

[4] Daniel Arijon, *Grammar of the Film Language*. New York, NY, USA: Hastings, 1976.

[5] M. Gleicher and A. Witkin, "Through-the-lens camera control," in *Proc. 19th Annu. Conf. Comput. Graph. Interact. Techn.*, Jul. 1992, vol. 26, pp. 331–340.

[6] D. S. US, H. Kim, and S. Sastry, "Decentralized reflective model predictive control of multiple flying robots in dynamic environment," in *Proc. 42nd IEEE Conf. Decis. Control*, 2003.

[7] A. Richards and J. How, "Decentralized model predictive control of cooperating UAVs," in *Proc. 47th IEEE Conf. Decis. Control*, 2004, vol. 4, pp. 4286–4291.

[8] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. 2011 IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2520–2525.

[9] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Auton. Robot.*, vol. 39, pp. 101–121, Jan. 2015.

[10] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *Proc. 2013 Eur. Control Conf.*, 2013, pp. 1383–1389.

[11] J. Schulman, Y. Duan, J. Ho, A. Lee, and I. Awwal, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, 2014.

[12] M. Christie, P. Olivier, and J. M. Normand, "Camera control in computer graphics," *Comput. Graph. Forum*, vol. 27, no. 8, pp. 2197–2218, 2008.

[13] S. M. Drucker and D. Zeltzer, "Intelligent camera control in a virtual environment," in *Proc. Graph. Interface*, 1994, pp. 190–199.

[14] C. Lino, M. Christie, R. Ranon, and W. Bares, "The director's lens: An intelligent assistant for virtual cinematography," in *Proc. 19th ACM Int. Conf. Multimedia*, New York, NY, USA, 2011, pp. 323–332.

[15] C. Lino and M. Christie, "Intuitive and efficient camera control with the toric space," *ACM Trans. Graph.*, vol. 34, no. 4, 2015, Art. no. 82.

[16] Q. Galvane, J. Fleureau, F.-L. Tariolle, and P. Guillotel, "Automated cinematography with unmanned aerial vehicles," in *Proc. Eurograph. Workshop Intell. Cinematography Editing*, The Eurographics Association, 2016.

[17] F. Poiesi and A. Cavallaro, "Distributed vision-based flying cameras to film a moving target," in *Proc. 2015 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 2453–2459.

[18] K. Hausman, G. Kahn, S. Patil, J. Müller, and K. Goldberg, "Cooperative occlusion-aware multi-robot target tracking using optimization." [Online]. Available: rll.berkeley.edu

[19] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. Robot. Autom.*, vol. 8, no. 3, pp. 313–326, Jun. 1992.

[20] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *J. Comput. Vis.*, vol. 1, pp. 333–356, 1988.

[21] S. L. Smith, M. Schwager, and D. Rus, "Persistent monitoring of changing environments using a robot with limited range sensing," in *Proc. 2010 IEEE Int. Conf. Robot. Autom.*, 2011, pp. 5448–5455.

[22] University of Dallas, "Elements of Cinematography: Camera." [Online]. Available: http://www.utdallas.edu/atec/midori/Handouts/camera.htm, Accessed on: Sep. 2, 2016.

[23] W. Breckenridge, "Quaternions proposed standard conventions," Jet Propul. Lab., Pasadena, CA, USA, Tech. Rep. Interoffice Memorandum IOM 343-79-1199, pp. 343–379, 1999.

[24] B. P. Gibbs, *Advanced Kalman Filtering, Least-Squares and Modeling: A Practical Handbook*. Hoboken, NJ, USA: Wiley, 2011.

[25] P. Cozzi and F. Stoner, "Gpu ray casting of virtual globes," in *Proc. SIGGRAPH*, 2010.

[26] E. C. Kerrigan and J. M. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," in *Proc. Int. Conf. Control*, 2000.

[27] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Proc. 47th IEEE Conf. Decis. Control*, 2012, pp. 668–674.

[28] A. Domahidi and J. Jerez, "FORCES Pro: Code generation for embedded optimization," Sep. 2016. [Online]. Available: https://www.embotech.com/FORCES-Pro