2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

# Reconstruction of 3D flight trajectories from ad-hoc camera networks

Jingtong Li[1*], Jesse Murray[1*], Dorina Ismaili[2], Konrad Schindler[1] and Cenek Albl[1]

***Abstract*— We present a method to reconstruct the 3D trajectory of an airborne robotic system only from videos recorded with cameras that are unsynchronized, may feature rolling shutter distortion, and whose viewpoints are unknown. Our approach enables robust and accurate outside-in tracking of dynamically flying targets, with cheap and easy-to-deploy equipment. We show that, in spite of the weakly constrained setting, recent developments in computer vision make it possible to reconstruct trajectories in 3D from unsynchronized, uncalibrated networks of consumer cameras, and validate the proposed method in a realistic field experiment. We make our code available along with the data, including cm-accurate groundtruth from differential GNSS navigation.**

## I. Introduction

Unmanned aerial vehicles (UAVs) are becoming ubiquitous: they are nowadays used for a wide range of tasks such as recording movies, delivering parcels, and collecting samples from inaccessible locations. A basic capability of any mobile, airborne robotic system is to localise itself in the 3D environment. Navigation with on-board sensors (e.g., GNSS receivers, IMUs, cameras) has reached a high level of maturity, still the proliferation of UAVs calls for a method to track them *outside-in*, i.e., without relying on on-board observations. The need for external tracking arises whenever on-board sensors fail (e.g., GNSS-denied environments) or are not accurate enough (e.g., the well-known drift of visual/inertial odometry). Moreover, outside-in tracking is the only option for tracking UAVs operated by others (e.g., to enforce airspace restrictions), as we do not have access to the on-board systems.

What properties should an external tracking system for UAVs have? We suggest that, besides of course being accurate and reliable, it should also be reasonably cheap, which translates to only using standard sensors that can be mass-produced; and that it should be easy to deploy, without complicated setup and calibration procedures.

Here, we propose a visual UAV tracking system consisting of a number of standard cameras (e.g., mobile phones) placed on, or near, the ground. We only assume per-camera calibration of the intrinsic parameters such as focal length and radial distortion, which can easily be done with a calibration target. All other parameters of the camera setup are unknown and can be recovered during operation, including the camera poses, the synchronisation between different cameras, and the influence of the rolling shutters. This approach allows us to use an arbitrary CMOS sensor camera network to solve problems of significantly greater complexity than standard structure-from-motion (SfM, [1], [2]) with a static scene.

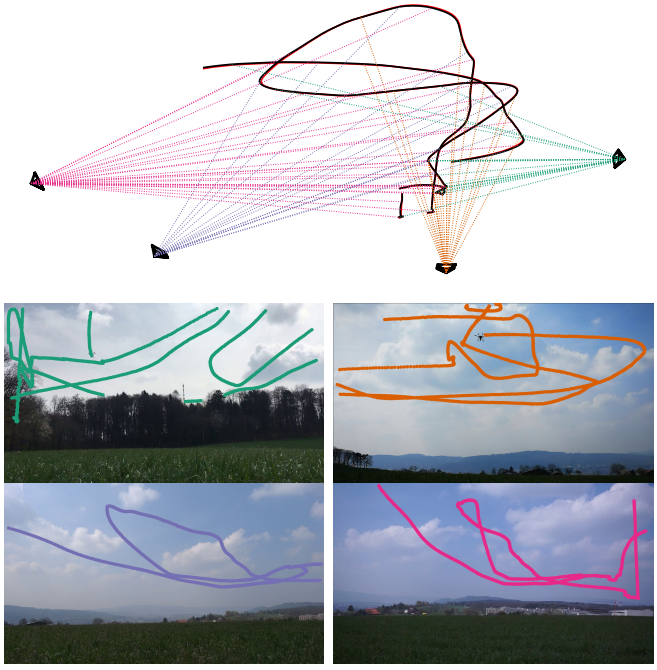

Fig. 1: A UAV is tracked using multiple unsynchronized cameras (bottom) with unknown poses. Our method robustly retrieves the 3D trajectory (black), camera poses and camera synchronization. The trajectory has a mean error of 7.6 cm compared to the ground truth (red).

In a realistic scenario the unknown camera poses can hardly be recovered a-priori from feature correspondences, as the cameras are placed far apart and observe mostly the sky, where one can neither find matching natural image features nor place a calibration target [3], [4]. Instead, we will use the observed flying object itself as a "feature" to calibrate the camera poses.

Another challenge is that the cameras will usually not be synchronized, since hardware synchronization complicates the installation and increases the cost, especially when the distances between cameras is large. Consequently, there will be a time shift between the triggers of any two cameras, and often also a time drift due to small deviations from their nominal frame-rates. For a dynamic target, such as a UAV, that effect is significant. For instance, an offset of just 1 frame at 25 fps corresponds to 40 ms, corresponding to a displacement of 33 cm for a UAV travelling at 30 km/h.

[1] Photogrammetry and Remote Sensing, ETH Zurich, 8093 Zurich, Switzerland {firstname.lastname}@geod.baug.ethz.ch
[2] Department of Mathematics, Technical University Munich, 85748 Garching bei München, Germany dorina.ismaili@tum.de
∗ Equal contribution

Any accuracy below that value can only be achieved with sub-frame synchronisation.

Moreover, the vast majority of cameras today are equipped with electronic rolling shutters (RS) [5]. The sensor array is exposed line by line with a constant delay between adjacent lines, leading to time delays *within* each "frame" that are proportional to the line number. Commonly the entire frame-to-frame interval is used for one RS pass, hence also the offsets between lines in the same "frame" reach a significant magnitude for fast-moving objects.

### A. Previous work

Detection and tracking of flying vehicles in images has recently been studied for a ground based, moving camera [6], [7] and also for cameras mounted on another flying vehicle [8]. These methods pave the way for the 3D trajectory reconstruction of a flying object by detecting its image coordinates. A theodolite system augmented by a CCD camera, QDaedalus [9], has been developed to automatically and precisely measure target location. It has been since used in automatic tracking of aircraft approaching an airport [10]. This system presents an accurate, yet very expensive method with limited applicability due to the theodolites not being suited for fast motions.

Existing solutions for external UAV tracking use commercial motion capture systems that work with active illumination and targets placed on the UAVs [11], [12], [13]. They reach high accuracy (down to millimeters), but require precise calibration and reasonably controlled lighting; and are therefore limited to small indoor areas of at most a few tens of meters (such systems are also more expensive than many of the UAVs we may want to track).

Calibration of ad-hoc camera network has been investigated in [14] using a static scene where synchronization is not needed, contrary to the case of calibration using dynamic objects. In [15] it was shown that a simple moving target (laser pointer dot) is a convenient tool to accurately calibrate camera networks where the cameras are synchronized. Simultaneous calibration and synchronization from silhouettes has been proposed in [16]. That approach, while robust, suffers from high complexity, as it involves a brute-force search over possible time shifts.

An early attempt to calibrate and synchronize cameras using the trajectory of a flying object was described in [17]. They employ a linear approximation of the 2D trajectory in each image to jointly estimate two-view geometry and a temporal offset. That work has been extended by [18], who use a cubic spline, and additionally estimate the difference in frame rate. [19] develops a minimal solver to estimate two-view camera geometry and time offset from eight points per view. In combination with RANSAC that method has been shown to robustly recover large offsets even in the presence of outliers. Bundle adjustment of unsynchronized cameras observing moving objects has been investigated in [20]. The paper discusses several priors to regularise motion trajectories, and finds that, for human-induced motion, minimising the associated kinetic energy works particularly well. [21] present a framework for reconstructing the trajectory of a quad-rotor from ground-based cameras. They model the specific flight dynamics of their system and explicitly fit the corresponding latent motion variables like angular velocity, throttle, and moments of inertia.

### B. Motivation and Contribution

Driven by the search for a convenient and practical external tracking system, this work asks the following question: can we reconstruct the 3D trajectory of a flying robot from multiple, unsynchronized videos recorded from different, unknown viewpoints?

Our aim is to assemble a complete system and demonstrate it in a real outdoor experiment. Individual components of the problem have been studied [17], [18], [19], [20], but they are sensitive to noise, outliers, and temporal offsets. We are not aware that trajectory reconstruction from such an ad-hoc camera network has ever been demonstrated. Perhaps the closest work to ours is [21], but it requires synchronized camera streams, and an existing 3D reconstruction of the scene, to which the cameras can be registered. Moreover, the method includes a dedicated motion model for quadrotors and thus cannot be used directly for other targets such as fixed-wing planes. To our knowledge, none of the existing works has considered the temporal offsets within a frame caused by the ubiquitous rolling shutter.

In the following, we present a complete, fully automatic pipeline that is able to calibrate a network of independently recording cameras and to reconstruct the 3D trajectory of a flying target, using only the image locations of the target as input. While, for clarity, we base the explanation on image coordinates of the target, we point out that those were in fact determined using an off-the-shelf automatic object detection method available in OpenCV [22] based on background subtraction and Gaussian Mixture Models [23], which works well for our simple case of a UAV against a sky background.

Our approach incrementally builds up an initial geometry estimate by chaining two-view geometries [19], while at the same time establishing temporally consistent $2D \leftrightarrow 2D$ and $2D \leftrightarrow 3D$ correspondences along a spline curve. The subsequent bundle adjustment [24] is extended such that it also optimally estimates the temporal shifts and drifts between videos and the rolling shutter readout speeds, while constraining the 3D trajectory to a cubic spline, optionally utilizing the force/energy-based priors from [20]. Note that these priors are valid for a wide class of targets, and also simpler than a platform-specific dynamic model, as in [21].

The proposed method is robust against outliers, thanks to the use of a minimal solver inside a RANSAC loop for the initial geometry. Furthermore it reaches high accuracy by properly taking into account synchronization and rolling shutter. In our experiments we achieve a mean reconstruction error of $<$40 cm at $\approx$50 m flying height, using a network of four to seven different low-cost cameras (phones, compact cameras and action cams).

## II. Problem Statement

We are interested in a broadly applicable scenario, where several different cameras are placed independently to observe the region of the sky where the UAV will operate. Note that in such a setup each camera will observe very little of the ground and baselines will be large, hence we cannot rely on background feature points for pose estimation. The camera intrinsics are assumed to be calibrated, since this is a simple off-line procedure that can be individually performed for each camera. Recovering the intrinsics online would in principle be possible, too, but suffers from a number of geometric instabilities [25], [26]. For clarity we formulate the problem for a single moving object, the extension to multiple targets is trivial.

The projection of a calibrated, perspective camera [1] can be described as

$$\mathtt{x} = \mu(\mathtt{PX}) = \mu(\left[\, \mathtt{R} \;\middle|\; -\mathtt{C} \,\right] \mathtt{X}) \tag{1}$$

where $\mathtt{x} = \mu \begin{bmatrix} x & y & z \end{bmatrix}^\top = \begin{bmatrix} \frac{x}{z} & \frac{y}{z} \end{bmatrix}^\top$ are the 2D image point coordinates, $\mathtt{P}$ is the $3 \times 4$ camera matrix, $\mathtt{X}$ are the homogeneous 3D point coordinates, $\mathtt{R}$ is a $3 \times 3$ rotation matrix and $\mathtt{C}$ are the 3D coordinates of the camera center.

In our network, $N$ cameras with projection matrices $\mathtt{P}_i, i \in [0, \ldots, N-1]$ each capture $M_i$ frames at times

$$t_i^j = \alpha_i j + \beta_i. \tag{2}$$

Here, $\beta_i$ is an offset and $\alpha_i$ is a scale factor, which together map the frame index $j \in [0, \ldots, M_i - 1]$ to a global time. Without loss of generality, we can use the first camera to anchor the global time, such that $\alpha_0 = 1$, $\beta_0 = 0$ and $t_0^j = j$.

The input of our method are the available 2D detections $\mathtt{x}_i^j$ from all cameras. The task is to estimate the camera matrices $\mathtt{P}_i$, synchronization parameters $[\alpha_1, \beta_1] \ldots [\alpha_{N-1}, \beta_{N-1}]$, and the UAV trajectory, expressed as a list of 3D coordinates $\mathtt{X}_i^j$, each time-stamped to a frame index $j$ in camera $i$.

## III. Initial Geometry Computation

As in classical SfM, respectively visual SLAM, the first step is to compute the coarse, initial camera poses and object coordinates, which are then refined through a subsequent bundle adjustment. The initial geometry is built up incrementally, by first computing the relative orientation of two views, then alternating between triangulating the 3D structure and registering new cameras with perspective-n-point (PnP).

In contrast to conventional SfM we are dealing with a dynamic object, which every camera observes at a (slightly) different time and, hence, a different 3D position. That is, we do not obtain direct $2D \leftrightarrow 2D$ correspondences between images, and similarly we cannot match existing 3D points to 2D image points observed in a new camera. Throughout our reconstruction pipeline, we use the following simple, but effective scheme.

### A. Correspondence between unsynchronized cameras

To obtain $2D \leftrightarrow 2D$ correspondences between two cameras $(i, k)$ we use a spline approximation of the trajectory in 2D image space. Given an image point $\mathtt{x}_i^j$, we check whether there is a temporally overlapping set of $Q > 3$ consecutive points $\{\mathtt{x}_k^l, \mathtt{x}_k^{(l+1)}, \ldots, \mathtt{x}_k^{(l+Q)}\}$ in the other camera. If this is the case, we fit a spline to those points,

$$\mathtt{s}_k^p(t) = \sum_m \mathtt{b}_m(t) K_{km}^p \;, \tag{3}$$

where $\mathtt{s}_k^p(t)$ are the coordinates on the approximated trajectory at time $t$, $\mathtt{b}_m(t)$ are the basis functions and $K_{km}^p$ the spline coefficients. Evaluating the spline at the recording time $t_i^j$ of $\mathtt{x}_i^j$ yields a correspondence $\mathtt{x}_i^j \leftrightarrow \mathtt{s}_k^p(t_i^j)$, or shortly $\mathtt{x}_i^j \leftrightarrow \hat{\mathtt{x}}_i^j$, see Figure 2b.

$3D \leftrightarrow 2D$ correspondences for registering new cameras are obtained in much the same way, by fitting the 3D trajectory with a 3D spline curve $\mathtt{T}_r(t)$. Each $3D \leftrightarrow 2D$ correspondence is then obtained as $\mathtt{x}_i^j \leftrightarrow \mathtt{T}_r(t_i^j)$. See Fig. 2c and Sec. III-E for details on 3D spline fitting.

### B. Two-view geometry and time shift estimation

There are two main challenges to be addressed in our scenario: outliers due to detection errors and the unknown temporal offset between the cameras. A minimal solver for fitting two-view relative pose together with a constant time offset has recently been developed [19]. Since it only needs 8 correspondences, that method is suitable for outlier rejection with RANSAC.

We obtain putative correspondences $\mathtt{x}_i^j \leftrightarrow \hat{\mathtt{x}}_k^j$ as described in Sec. III-A, where as initial synchronisation parameters we set $\alpha_i$ according to the nominal frame-rate and $\beta_i$ to zero. We sample 8 of those correspondences randomly to solve the generalized epipolar equation under a linear approximation $\mathtt{v}_k^j$ of the 2D image point trajectory,

$$(\hat{\mathtt{x}}_k^j + \beta_{ik} \mathtt{v}_k^j)^\top \mathtt{F} \mathtt{x}_i^j \;. \tag{4}$$

This yields the relative time shift $\beta_{ik}$ and the fundamental matrix [1] $\mathtt{F}_{ik}$ between cameras $i$ and $k$, from which the camera matrices $\mathtt{P}_i$ and $\mathtt{P}_k$ can be readily extracted [1].

We exhaustively solve the two-view geometry for all camera pairs that have a sufficient number of correspondences, so as to obtain improved estimates of $\beta_i$ for all cameras.

### C. 3D points triangulation

Having recovered the pairwise relative poses and time shifts, we pick the pair with most inlier correspondences, recompute the correspondences $\hat{\mathtt{x}}_i^j \Leftrightarrow \mathtt{x}_k^j$ with the improved $\beta_i, \beta_k$, and triangulate 3D points with the standard linear projective method [1]. Note that by knowing the offsets we can globally time-stamp the triangulated 3D points $\mathtt{X}^j$. Note also, once the spline approximation of the trajectory has been computed we need not save the triangulated 3D points, as their (spline-smoothed) coordinates can be recovered from the spline parameters $\mathtt{T}(t)$ and the global time-stamp $t$.

### D. Registration of remaining cameras

Absolute camera pose is computed from $3D \leftrightarrow 2D$ correspondences with the standard P3P algorithm [27], using RANSAC for robustness. Correspondences for a new camera $k$ are obtained by finding the temporal overlap with the already reconstructed 3D trajectory and evaluating the spline curves $\mathtt{T}_r(t)$ at the appropriate times $t_k^j$, see Sec. III-A.

### E. Building up the trajectory

The UAV trajectory is gradually built up as a sequence of 3D spline curves parametrized by the global time $t$, such that

$$\mathtt{T}_r(t) = \sum_u^V \mathtt{b}_u(t) K_u \tag{5}$$

Only the spline parameters are carried over to subsequent steps of the pipeline. Every time a new camera $k$ has been registered, we try to fill in previously missing parts of the 3D trajectory. To that end the algorithm looks for image points $\mathtt{x}_k$ outside the (temporal) range of the already reconstructed parts, constructs $2D \leftrightarrow 2D$ correspondences for those points as described in Sec. III-A, triangulates them, and extends the spline. With a similar mechanism, existing portions of the spline are periodically updated as additional $3D \leftrightarrow 2D$ correspondences become available from new images.

## IV. Rolling Shutter Effect

The large majority of cameras are nowadays equipped with rolling shutters. We must account for the fact that, while the first line $l_0$ in frame $j$ is recorded at time $t_i^j$, a different line $l_k$ of the same frame is recorded at $t_i^j + r_i l_k$, with $r_i$ the RS scan speed expressed in lines/s.

Accordingly, we incorporate the RS effect in our optimization, by modifying equation 2 to

$$t_i^j = \alpha_i j + \beta_i + r_i x_{2i}^j \tag{6}$$

where $x_{2i}^j$ is the $y$-coordinate (the line index) of the image coordinate, $\mathtt{x}_i^j = \begin{bmatrix} x_{1i}^j & x_{2i}^j \end{bmatrix}^\top$.

## V. Spatio-Temporal Bundle Adjustment

As in conventional SfM, our incremental procedure is not statistically optimal and prone to error accumulation. We thus optimize the estimated parameters by minimizing the image reprojection errors through an extended bundle adjustment [24]. The inputs of the optimization are the 2D observations $\mathtt{x}_i^j$ and initial estimates for all unknown parameters, as obtained from the initial reconstruction described above. That is, the camera poses $\mathtt{P}_i$, 3D trajectory parameters $T_r(t)$, as well as time offsets $\beta_1, \ldots, \beta_n$, time scales $\alpha_1, \ldots, \alpha_n$ and RS scan speeds $r_i$ for each camera. For the latter, initialising them as $r_i = 0$ is sufficient.

The splines used so far are a convenient, but somewhat arbitrary choice of trajectory model. For the refinement we therefore also experiment with other ways of regularizing the trajectory. Namely, minimizing respectively the kinetic energy and force associated with the UAV motion.

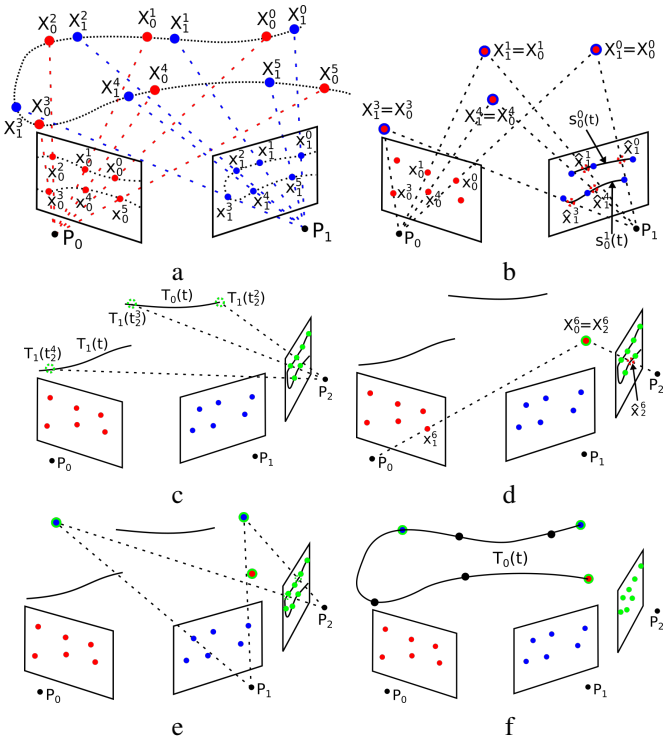


Fig. 2: Incrementally reconstructing the trajectory. Given 2D trajectory points (a) we start with two cameras with the largest time overlap, interpolate $2D \leftrightarrow 2D$ correspondences, compute relative orientation and triangulate 3D points (b). The 3D trajectory is represented by spline curves and new cameras are added by sampling the splines to obtain $3D \leftrightarrow 2D$ correspondences (c). New points are triangulated by combining data from existing and newly added cameras (d,e), and the splines are updated (f).

### A. Spline trajectory

This variant simply retains the cubic splines $T_r(t)$ already fitted before, and optimizes their parameters $K_r$:

$$\begin{aligned} &\underset{\mathtt{P}_i, \alpha_i, \beta_i, r_i, K_r}{\arg\min} \sum_{i=0}^{N} \sum_{j=0}^{M_i} \| \mathtt{x}_i^j - \mu(\mathtt{P}_i T_r(t_i^j)) \|^2 \\ &\text{with} \quad t_i^j = \alpha_i j + \beta_i + r_i x_{2i}^j \,. \end{aligned} \tag{7}$$

### B. Least kinetic energy trajectory

This model was found in [20] to work well for dynamic 3D objects. The kinetic energy of an object with mass $m$ is $E_k = \frac{1}{2} m v^2$. Considering the object a point mass with $m = 1$, and approximating its instantaneous velocity at time $t_i^j$ as $||(\mathtt{X}_i^{j+1} - \mathtt{X}_i^j)/(t_i^{j+1} - t_i^j)||$ leads to

$$\begin{aligned} &\underset{\mathtt{P}_i, \alpha_i, \beta_i, r_i, \mathtt{X}_i^j}{\arg\min} \quad e_r + e_m \qquad \text{with} \\ &e_r = \sum_{i=0}^{N} \sum_{j=0}^{M_i} \| \mathtt{x}_i^j - \mu(\mathtt{P}_i \mathtt{X}_i^j) \|^2 \\ &e_m = \sum_{i=0}^{N} \sum_{j=0}^{M_i} \left\| \frac{\mathtt{X}_i^{j+1} - \mathtt{X}_i^j}{t_i^{j+1} - t_i^j} \right\|^2 . \end{aligned} \tag{8}$$

The times $t_i^j$ are again expanded as in equation (6). In our experiments, however, we found that constraining the motion purely based on the instantaneous velocity does not work at all for the problem of UAV trajectory reconstruction. The reason for this is that contrary to [20], who used abundant static features visible in their scenes to constrain their camera poses, the UAV scenes contain mostly uniform sky from which few static features can be identified.

Therefore, we decided to keep the spline approximation and add the motion regularizers on top of it, such that

$$e_r = \sum_{i=0}^{N} \sum_{j=0}^{M_i} \|\mathtt{x_i^j} - \mu(\mathtt{P}_i T_r(t_i^j))\|^2 \tag{9}$$

as in equation 7.

### C. Least force trajectory

This variant was found in [20] to perform similar to least kinetic energy, for human motion. For UAVs, least force is arguably a physically more meaningful assumption: the least kinetic energy prior penalizes high velocities, but for a flying robot maintaining even a rather high velocity may be less effort than applying a high force to accelerate or decelerate.

The force needed to accelerate an object with mass $m$ by $a$ is $F = ma$. Again setting $m = 1$, without loss of generality, the instantaneous acceleration is approximately $a = ||(\mathtt{X}_i^{j+1} - \mathtt{X}_i^j)/(t_i^{j+1} - t_i^j) - (\mathtt{X}_i^j - \mathtt{X}_i^{j-1})/(t_i^j - t_i^{j-1})||$. The corresponding objective is the same as in section V-B, except that the prior is replaced by

$$e_m = \sum_{i=0}^{N} \sum_{j=0}^{M_i} \left\| \frac{\mathtt{X}_i^{j+1} - \mathtt{X}_i^j}{t_i^{j+1} - t_i^j} - \frac{\mathtt{X}_i^j - \mathtt{X}_i^{j-1}}{t_i^j - t_i^{j-1}} \right\|, \tag{10}$$

with the times $t_i^j$ again expanded as above. The bundle adjustment can be performed periodically in the pipeline and as early as we obtain the first geometry estimate from two cameras. We perform the optimization every time a new camera has been added.

## VI. Implementation Strategies

The proposed framework allows for different modes of operation, according to application requirements. A minimal requirement is that the observed trajectory is long enough, and deviates sufficiently from a straight line, the latter being a degenerate configuration [28]. For offline processing, where the trajectory reconstruction is done after recording, one can simply use all collected data (subject to memory limits). On the contrary, in an online application one will first use a limited number of detections/frames per camera to calibrate the system. Once the camera parameters have stabilized one can freeze them and switch to tracking mode, where one only triangulates new 3D points to extend the trajectory, as described in Sec. III-C. As in many SfM and visual SLAM systems [2], [29], bundle adjustment (Sec. V) can be re-run periodically in an asynchronous thread when a sufficient amount of new evidence has accumulated, in both calibration and tracking mode.

| Camera # | Model | Resolution | Frame rate |
|---|---|---|---|
| 1 | Huawei Mate 7 | 1920x1080 | 30.02 |
| 2 | Huawei Mate 10 | 3840x2160 | 29.72 |
| 3 | Sony NEX-5N | 1440x1080 | 25 |
| 4 | Sony Alpha 5100 | 1920x1080 | 29.97 |
| 5 | Sony DSC-HX20V | 1920x1080 | 50 |
| 6 | GoPro 3 | 1920x1080 | 59.94 |
| 7 | Huawei P20 Pro | 3840x2160 | 29.83 |

TABLE I: Cameras used to record our datasets.

## VII. Experiments

We use a Yuneec hexacopter designed for cinematographic recording and photogrammetric mapping. To acquire ground truth, the UAV was equipped with a real-time kinematic (RTK) GNSS system from Fixposition [30], with an estimated localization accuracy of $\pm 1$ cm [31]. The UAV flies within a region of $\approx 100 \times 100$ m, at heights up to $\approx 50$ m.

Around that region we set up 4-7 cameras, in such a way that the UAV is almost always visible in $\geq 2$ views. The cameras are of various types – smartphones, compact cameras and a GoPro action cam – with varying resolutions, frame rates and fields of view, see Table I. All of them have a rolling shutter. For dataset 3, ground truth camera synchronization was obtained with a radio-synchronized network of LEDs. One LED was placed in the field of view of each camera, such that synchronous flashes are periodically visible in all videos (40 flashes in total). To evaluate also the estimated camera poses, we measured the camera locations for dataset 3 directly with a survey-grade GNSS receiver (Trimble R8). The estimated positioning accuracy is 9 mm, but since we could not make the antenna centre coincide with the projection centres of the different consumer devices, we conservatively estimate that the ground truth coordinates of the camera centres have an accuracy better than 5 cm.

Altogether, we created 4 datasets with various properties, see table II. Datasets 1 and 2 represent standard cases, where the UAV flies with moderate speed and changes direction smoothly. Dataset 3 poses greater challenges in terms of speed (and thus also trajectory length) and acceleration. We strove to cover a variety of flight modes and movement types, from straight stretches to sharp changes of direction and fast ascents/descents. Dataset 4 is even more challenging, with the most aggressive flight path and also a larger number of misdetections, caused by fast-moving clouds that interfered with our detector. Dataset 5 was provided by [21]. Unfortunately its ground truth accuracy is not known. Off-the-shelf systems without differential GNSS typically reach at best 0.5 m, so the results cannot be compared directly to those of datasets 1-4. We refrain from any preprocessing or filtering of the videos to test that all parts of our pipeline are robust to outliers.

### A. Reconstruction accuracy

Our pipeline achieves remarkable accuracy for both the reconstructed trajectory and the camera poses. Quantitative evaluations of the 3D error are given in Table IV for datasets 1-4, and in Table V in dataset 5. To compute the error we

| Dataset # | # cameras | duration [s] | source | Velocity [km/h] | |
|---|---|---|---|---|---|
| | | | | Mean | Max |
| 1 | 4 | 120 | ours | 7 | 21 |
| 2 | 4 | 120 | ours | 14 | 26 |
| 3 | 6 | 600 | ours | 20 | 35 |
| 4 | 7 | 600 | ours | 24 | 37 |
| 5 | 6 | 500 | [21] | 12 | 29 |

TABLE II: Datasets and their parameters.

| | $\beta_{init}$ | 5 | 10 | 15 | 20 | 50 |
|---|---|---|---|---|---|---|
| Full pipeline | $\beta_F$ error | 0.6 | 0.7 | 0.7 | 0.8 | 0.7 |
| | $\beta_{opt}$ error | 0.5 | 0.6 | 0.6 | 0.7 | 0.6 |
| | Mean err. [cm] | 17.3 | 17.1 | 16.8 | 16.4 | 16.3 |
| w/o $\beta_F$ | $\beta_{opt}$ error | 1 | | 7.7 | | |
| | Mean err. [cm] | 19.7 | | 83.8 | | |
| w/o $\beta_F$, w/o $\beta_{opt}$ | Mean err. [cm] | 137.4 | | | | |

TABLE III: Importance of pairwise synchronisation. The table shows the mean error of estimated time shifts (unit: frames) for increasing initial de-sync $\beta_{init}$. Missing numbers indicate failures that did not return a reasonable trajectory. See text for details.

follow [21]: we first fit a similarity transform to align the reconstructed trajectory to the ground truth and compensate possible systematic offsets, then compute distances between the aligned trajectories. For dataset 3 we also measured the ground truth camera locations, which we again align to the estimated ones. On top of the mean and median absolute errors and the RMSE, we also show the outlier ratio, computed as the portion of the trajectory outside the confidence interval of 3×RMSE (99.7% of the probability mass for 0-mean Gaussian inliers).

On our data we achieved mean errors well below 20 cm for the first three datasets, and even for the hard dataset 4 below 40 cm (about the diameter of the UAV). For dataset 3 the mean error of the reconstructed camera positions was 17cm and the biggest error was 68cm, a very good localisation accuracy considering the baselines in the order of 100 m. Visual example of the camera poses and trajectories are shown in Fig. 3.

| | | Spline | | Least kinetic | | Least force | |
|---|---|---|---|---|---|---|---|
| | | w/o RS | RS | w/o RS | RS | w/o RS | RS |
| Dataset 1 | Mean | 7.6 | 7.5 | 7.5 | 7.4 | 7.5 | **7.3** |
| | RMSE | 8.7 | 8.6 | 8.6 | **8.5** | 8.7 | **8.5** |
| | Median | 6.3 | 6.3 | 6.2 | 6.2 | 6.3 | **6.1** |
| | Outl. % | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Dataset 2 | Mean | 14.4 | 14.3 | 14.2 | **14.1** | 14.3 | 14.2 |
| | RMSE | 16.9 | 16.8 | 16.7 | **16.5** | 16.9 | 16.7 |
| | Median | 12.1 | 12.2 | **12.0** | 12.1 | 12.2 | 12.3 |
| | Outl. % | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Dataset 3 | Mean | 16.8 | 16.2 | 16.6 | **16.1** | 16.6 | 16.4 |
| | RMSE | 24.5 | 22.2 | 23.7 | **22.0** | 22.7 | **22.0** |
| | Median | 11.4 | 11.5 | **11.3** | 11.6 | 11.8 | 12.1 |
| | Outl. % | 2.2 | 1.9 | 2.2 | 1.8 | 1.8 | **1.7** |
| Dataset 4 | Mean | 38.9 | 38.2 | 38.6 | 38.0 | 37.6 | **35.9** |
| | RMSE | 54.1 | 54.0 | 57.0 | 54.5 | 54.4 | **49.7** |
| | Median | 31.7 | 30.2 | 29.8 | 30.1 | **29.6** | 29.7 |
| | Outl. % | **1.5** | **1.5** | 1.8 | 1.6 | **1.5** | **1.5** |

TABLE IV: 3D trajectory errors on our datasets with accurate ground truth, in centimeters.

| | | Spline | | Least kinetic | | Least force | |
|---|---|---|---|---|---|---|---|
| | | w/o RS | RS | w/o RS | RS | w/o RS | RS |
| Dataset 5 | Mean | 77.2 | 75.2 | 76.5 | 74.5 | **70.5** | 73.0 |
| | RMSE | 94.4 | 88.2 | 94.9 | 87.8 | **81.7** | 85.8 |
| | Median | 66.8 | 64.5 | **60.5** | 61.1 | **60.5** | 61.9 |
| | Outl. % | 0.8 | 0.2 | 1.4 | **0.0** | **0.0** | **0.0** |

TABLE V: 3D trajectory errors on the dataset of [21], in centimeters.

On dataset 5 we obtained an RMSE slightly above 80 cm, about half the error reported in [21]; a significant improvement, especially taking into account that in [21] the camera poses are pre-calibrated and synchronised, and the method is restricted to a specific type of aircraft dynamics. We do point out that the comparison is not exact, because we only reconstructed approximately 90% of the trajectory (our current detector assumes a sky background and missed the start and end of the trajectory where the UAV is near the ground).

### *B. Contribution of motion regularizers and RS*

Overall, the different versions of trajectory regularization did not differ significantly in terms of trajectory and camera pose errors, see Table IV. For the easy datasets 1 and 2, the results were almost identical. For the more challenging datasets, a small improvement becomes apparent. On dataset 3, the RMSE drops by 1 cm with the kinetic energy regularizer and by 2 cm with the least force regularizer. The number of outliers was also reduced from 2.2% to 1.8%. On dataset 4, the kinetic energy regularizer performed slightly worse in terms of RMSE than the spline, which could be attributed to the fact that the UAV speed was the highest in this dataset and imposing constraints on the velocity is counterproductive. On the other hand, least force prior together with the RS term improved the mean error by 3 cm and RMSE by almost 5 cm compared to other approaches.

Incorporating the RS model in the bundle adjustment provides negligible error reduction in the easy cases, but does on average bring a small decrease in both mean error and RMSE for complex trajectories. Especially apparent was the use of the RS term together with the least force regularizer in dataset 4. The impact of the RS term depends on the drone positions in image space. E.g., if the trajectory only spans a narrow horizontal strip, the inter-row delay will be small and its influence might be negligible. The interplay of the motion priors and the RS term is of interest for future investigation.

On dataset 5, the results of the motion priors and RS correction were not entirely consistent with datasets 1-4, but due to the questionable accuracy of the ground truth for dataset 5 we do not consider these deviations significant.

Although not apparent in the statistical indicators, the motion regularizers and RS model locally improve parts of the trajectory significantly compared to the base solution with spline trajectory. In figure 4 we highlight exemplary cases with blue circles where the trajectory with the full model is clearly more plausible.

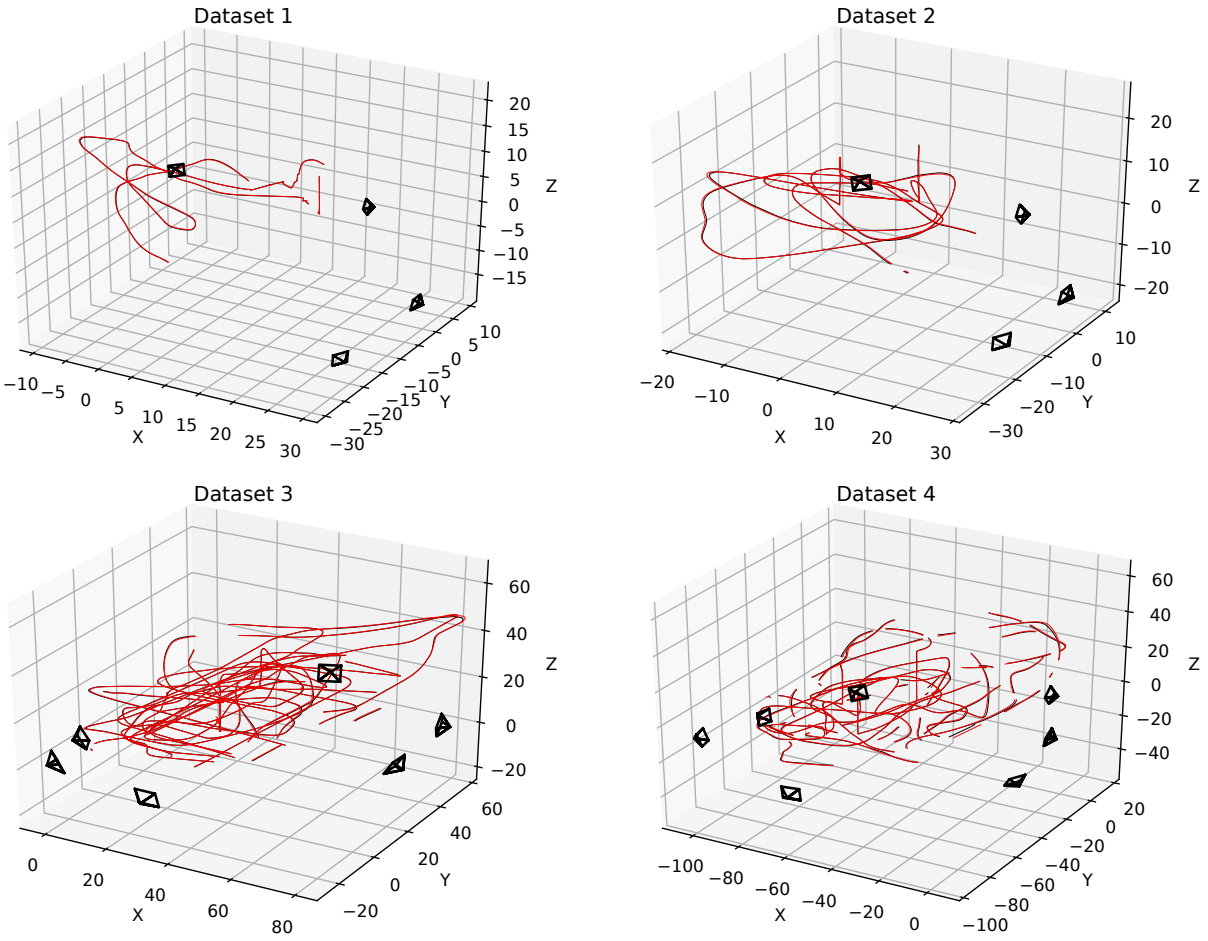


Fig. 3: Visualization of the reconstructed trajectories (black) from our datasets with accurate ground truth (red). The missing parts of the trajectory are where the drone was detected in $\leq 1$ camera.

### C. Robustness to de-synchronization

One of the major advantages of our pipeline for practical purposes is that the input videos need not be synchronized. Table III summarizes our ablation study to test the effect of individual steps when the input sequences are out of sync. Time shifts between input videos are modelled in two places: first, when fitting the two-view geometries (Sec. III-B) we estimate a time shift $\beta_F$, then during bundle adjustment we refine it to obtain a final estimate $\beta_{opt}$. The mean errors of these parameters are shown in the table for different amounts $\beta_{init}$ of desynchronization. We add two baselines. The naive one completely ignores the time shift and assumes the cameras are in sync (or the impact of desynchronization is negligible). The second one drops the time shift parameter only from the initial two-view fitting, thus testing the assumption that the initialisation will nevertheless be good enough to recover the time shift during bundle adjustment.

Table III shows that the the two-view time shift estimation is extremely robust and recovers sub-frame $\beta_F$ even with a large initial shift of 50 frames. The $\beta_{opt}$ after BA are only marginally better. The mean error of the reconstructed trajectory is not increased at all, i.e., our method fully compensates for the influence of desynchronization (the slight decrease with stronger de-sync is an artefact of the implementation, which then reconstructs slightly shorter trajectories). Without $\beta_F$, just optimizing $\beta_{opt}$ in the BA part of the pipeline, the mean reconstruction error and the resulting time shift error after BA immediately increase already at an initial time shift of 5 frames. For 10 frames shift, the pipeline fails, as the optimization gets stuck in a local minimum. For 15 frames shift the pipeline recovers the trajectory but with a large mean error of 83.8 cm and 7.7 frames mean error in the time shift. With neither $\beta_F$ nor $\beta_{opt}$, ignoring time shift altogether, the resulting trajectory has $>1$ m error already at 5 frames time shift, and fails for any larger shift.

## VIII. Conclusions

We have described a practical method for reconstructing the trajectories of flying objects using multiple external cameras. Contrary to previous work, our method needs neither synchronized cameras nor pre-calibrated camera poses. Furthermore, it accounts for the influence of rolling shutters, and can handle gross errors of the preceding object detector. We have shown that, thanks to these properties, a cheap and easy-to-use system can be built with off-the-shelf cameras, smart-

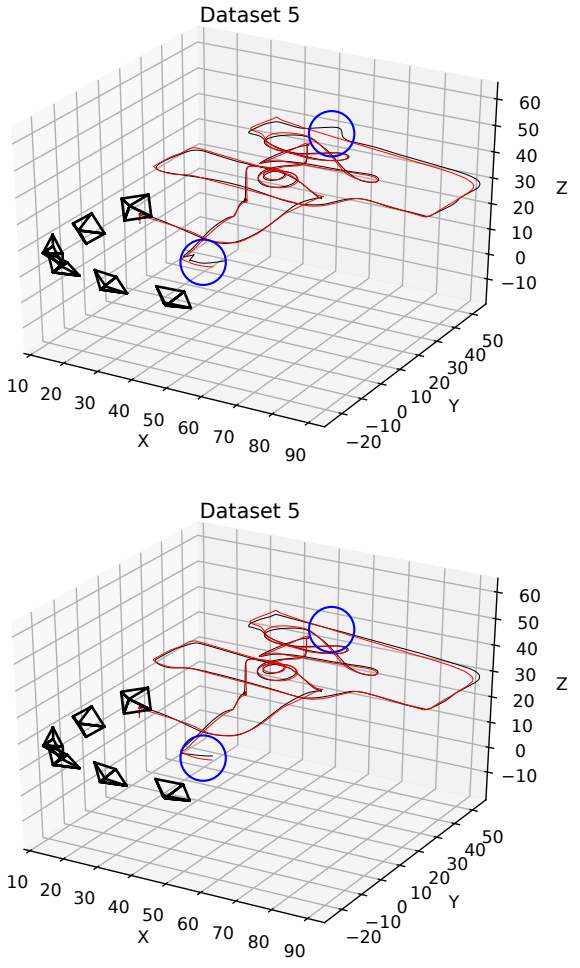


Fig. 4: Reconstructed trajectory from the data of [21]. Note that the spline parameterization (top) itself exhibited some artefacts (in blue circles), which the least force regularizer (bottom) was able to remove.

phones or surveillance cameras. Moreover, we have created a dataset with cm-accurate ground truth UAV trajectories, measured with differential GNSS. In our experiments we were able to achieve mean trajectory errors <40 cm relative to that ground truth. The dataset and the software pipeline are publicly available at `https://github.com/CenekAlbl/drone-tracking-datasets`, respectively `https://github.com/CenekAlbl/mvus`. So far we have only explored the ability to reconstruct UAV trajectories in post-processing, after collecting all the data. A logical next step is to reconstruct the trajectory online, as video data comes in. Moreover, it would be interesting to implement multi-target tracking.

## REFERENCES


[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo Tourism: exploring photo collections in 3D," in *SIGGRAPH*, 2006.

[3] Z. Zhang, "A flexible new technique for camera calibration," *TPAMI*, vol. 22, no. 11, 2000.

[4] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *IROS*, 2013.

[5] M. Meingast, C. Geyer, and S. Sastry, "Geometric Models of Rolling-Shutter Cameras," *Computing Research Repository*, 2005.

[6] A. Rozantsev, V. Lepetit, and P. Fua, "Detecting Flying Objects Using a Single Moving Camera," *TPAMI*, vol. 39, no. 5, 2017.

[7] Y. Wu, Y. Sui, and G. Wang, "Vision-Based Real-Time Aerial Object Localization and Tracking for UAV Sensing System," *Access*, vol. 5, 2017.

[8] A. Nussberger, H. Grabner, and L. V. Gool, "Aerial object tracking from an airborne platform," in *ICUAS*, 2014.

[9] S. Guillaume, B. Burki, S. Griffet, and H. Mainaud Durand, "QDaedalus : Augmentation of Total Stations by CCD Sensor for Automated contactless High-Precision Metrology," in *FIG Working Week*, 2012.

[10] S. Guillaume, A. Geiger, and M. Scaramuzza, "Automated High Precision Optical Tracking of Aircrafts and non-cooperative Flying Objects," in *ION GNSS+*, 2016.

[11] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The Flying Machine Arena," *Mechatronics*, vol. 24, no. 1, 2014.

[12] J. P. How, B. Behihke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *Control Systems Magazine*, vol. 28, no. 2, 2008.

[13] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP Multiple Micro-UAV Testbed," *Robotics Automation Magazine*, vol. 17, no. 3, 2010.

[14] D. Devarajan, R. J. Radke, and H. Chung, "Distributed metric calibration of ad hoc camera networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 3, pp. 380–403, 2006.

[15] T. Svoboda, D. Martinec, and T. Pajdla, "A Convenient Multicamera Self-Calibration for Virtual Environments," *Presence*, vol. 14, no. 4, pp. 407–422, Aug. 2005.

[16] S. Sinha and M. Pollefeys, "Synchronization and calibration of camera networks from silhouettes," in *ICPR*, 2004.

[17] M. Noguchi and T. Kato, "Geometric and Timing Calibration for Unsynchronized Cameras Using Trajectories of a Moving Marker," in *WACV*, 2007.

[18] M. Nischt and R. Swaminathan, "Self-calibration of asynchronized camera networks," in *ICCV Workshops*, 2009.

[19] C. Albl, Z. Kukelova, A. W. Fitzgibbon, J. Heller, M. Smid, and T. Pajdla, "On the Two-View Geometry of Unsynchronized Cameras." in *CVPR*, 2017.

[20] M. Vo, S. G. Narasimhan, and Y. Sheikh, "Spatiotemporal bundle adjustment for dynamic 3D reconstruction," in *CVPR*, 2016.

[21] A. Rozantsev, S. N. Sinha, D. Dey, and P. Fua, "Flight Dynamics-Based Recovery of a UAV Trajectory Using Ground Cameras," in *CVPR*, 2017.

[22] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[23] P. KaewTraKulPong and R. Bowden, "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection," in *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*, 2002.

[24] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment — A Modern Synthesis," in *Vision Algorithms: Theory and Practice*, Lecture Notes in Computer Science.

[25] C. Wu, "Critical Configurations for Radial Distortion Self-Calibration," in *CVPR*, 2014.

[26] F. Kahl, B. Triggs, and K. Åström, "Critical Motions for Auto-Calibration When Some Intrinsic Parameters Can Vary," *JMIV*, vol. 13, no. 2, 2000.

[27] R. M. Haralick, D. Lee, K. Ottenburg, and M. Nolle, "Analysis and solutions of the three point perspective pose estimation problem," in *CVPR*, 1991.

[28] R. Hartley and F. Kahl, "Critical Configurations for Projective Reconstruction from Multiple Views," *IJCV*, vol. 71, no. 1, pp. 5–47, June 2006.

[29] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *ISMAR*, 2007.

[30] "Fixposition. www.fixposition.ch," 2017. [Online]. Available: https://www.fixposition.ch/

[31] Z. Su, "Single-frequency RTK GNSS positioning," Doctoral thesis, ETH Zurich, 2018.