Fusing Vision and Inertial Sensors for Robust Runway Detection and Tracking

Khaled Abu-Jbara* and Ganesh Sundaramorthi*
KAUST, Thuwal, Saudi Arabia
and
Christian Claudel†
University of Texas, Austin 78712, Texas

DOI: 10.2514/1.G002898

This work presents a novel real-time algorithm for runway detection and tracking applied to unmanned aerial vehicles (UAVs). The algorithm relies on a combination of segmentation-based region competition and minimization of a particular energy function to detect and identify the runway edges from streaming video data. The resulting video-based runway position estimates can be updated using a Kalman filter (KF) that integrates additional kinematic estimates such as position and attitude angles, derived from video, inertial measurement unit data, or positioning data. This allows a more robust tracking of the runway under turbulence. The performance of the proposed lane detection and tracking scheme is illustrated on various experimental UAV flights conducted by the Saudi Aerospace Research Center (KACST), by the University of Texas, Austin, and on simulated landing videos obtained from a flight simulator. Results show an accurate tracking of the runway edges during the landing phase, under various lighting conditions, even in the presence of roads, taxiways, and other obstacles. This suggests that the positional estimates derived from the video data can significantly improve the guidance of the UAV during takeoff and landing phases.

Nomenclature

E	=	energy function
Ω	=	domain of the image
θ	=	angle of line segment, rad
∇	=	gradient

I. Introduction

UNMANNED aerial vehicles (UAVs) have become increasingly prevalent in the past decade, enabling monitoring tasks that are too dirty, too dull, or too dangerous (DDD) to be undertaken by manned flying vehicles. UAVs have, for instance, been used in surveillance applications, including fire detection [1], event detection [2], or object tracking [3]. They also play an increasing role in military operations [4] for both reconnaissance and strike. While rotorcraft UAVs are most commonly used, fixed-wing UAVs have in general a higher autonomy and endurance, owing to their relatively high lift-to-drag ratio, and are thus preferred for certain types of operations (e.g., the surveillance of a target, or search and rescue operations). However, takeoff and landings of fixed-wing UAVs are particularly risky [5], particularly because lightweight UAVs are much more sensitive to turbulence than heavier manned aircraft, and thus have much lower time constants. In currently operated UAVs, takeoffs and landings are typically carried out using navigation sensors, such as absolute positioning systems (e.g., Global Positioning System [GPS]), accelerometers, gyrometers, and magnetometers. While the fusion between these sensors greatly improves their accuracy, the residual positional errors are still too large to allow reliable UAV landings on narrow UAV airstrips. Positioning accuracy on practical UAV airstrips could be improved by using enhanced GPS systems (such as differential GPS, or RTK GPS), though such systems require additional equipment (reference stations, phase sensing antennas), which is expensive. The use of high-accuracy positioning systems also requires the UAV airstrips to be precisely mapped, though this positioning information is most of the time unavailable. In contrast, the ever decreasing cost of cameras makes them particularly suitable to this positioning application. Because UAVs are mainly used for surveillance applications, the vast majority of UAV flights are conducted during good weather conditions, during which the visibility is high. To help the guidance of UAVs during takeoff and landing phases, we propose a new algorithm leveraging computer vision and control systems theory to detect the runway edges. The resulting position estimates can then be used to more accurately correct the lateral position of the UAV during the landing phase, and use higher gains in the UAV control loop, resulting in better tracking performance during the takeoff and landing phases. The main objective of the proposed algorithm is to build a very robust detection and tracking of the runway edges, which performs well under diverse lighting and environmental conditions. Because video data of UAV landings are scarce, supervised learning could yield poor performance. Numerous computer vision algorithms have been proposed to help the navigation of fixed-wing and rotary UAVs. In [6], the authors use morphological image processing and HTs to identify the horizon and estimate the attitude of a UAV. The authors of [7] propose a template matching algorithm to detect and track the position of a runway. Active methods can also be thought of, for instance, in [8], which uses active infrared emitters to guide the UAV during its landing. In other contexts, computer vision has been successfully used to detect road lanes (for autonomous vehicle applications), for instance, in [9], where roads are detected using HT and Robert filters. The authors of [10] use splines curve fitting, HT, Canny edge detector, and vanishing points to estimate the boundary of the road. Although effective, this method is complex and does not run in real time on commercially available embedded platforms. In paper [11] a line segment detector (LSD) is used to detect the main line features, and K-means clustering is applied to sort and select specific lines. The authors of [4] have a different approach, relying on a Sobel filter and K-means to find the edges of the road from a fixed surveillance camera. Other approaches for lane or road detection exist, such as in [12], in which the authors use Random Sample Consensus (RANSAC), and cubic spline curve fitting are used to extract the lane position from a video stream. A linear parabolic lane tracking system with

Received 13 March 2017; revision received 8 February 2018; accepted for publication 11 March 2018; published online 25 June 2018.

*Department of Electrical Engineering.

†Department of Civil, Architectural and Environmental Engineering, Austin 301 E. Dean Keeton St. Stop C1761.

Kalman filter (KF) is proposed in [13]. On the other hand, many approaches, such as [14,15], tend to use convex methods in PDE optimization to segment the image into N regions based on global intensity statistics or local intensity statistics.

To the best of the authors knowledge, no robust runway detection and tracking scheme capable of working under a wide variety of lighting conditions and a variety of runway configurations (including the presence of obstacles and taxiways) is currently available. Currently available methods, such as [16], require the user to specify in advance a very large number of parameters associated with the current tracking problem or to specify in advance which features (for instance, landmarks or shadows [10]) have to be excluded from the tracking problem. These requirements significantly complicate operations and require recalibration before each flight and before changing the takeoff and landing locations. The present algorithm uses a minimal number of assumptions on the features associated with all runways and relies on robust video detection and tracking methods, which offer sufficient performance in practice to be used standalone. In the present paper, we also investigate a KF approach that can fuse additional information (such as gyrometer data) in the prediction step, to enhance tracking accuracy.

We illustrate this problem of sensitivity to model parameters on Fig. 1, which shows the difficulty of robustly selecting the contours of the image that are relevant to the proposed problem. This figure illustrates the considerable modification of the output of a Canny edge detector (applied to a UAV landing video) caused by minor changes in its parameters. The upper left subfigure shows the actual video data, whereas the upper right, lower left, and lower right subfigures correspond to the outputs of the Canny edge detector applied to this image, using the parameters $\sigma = 1$ (upper right), $\sigma = 10$ (lower left), and $\sigma = 20$ (lower right), in which σ denotes to the standard deviation of the Gaussian Filter, with upper and lower thresholds of 0.4 and 0.1, respectively. As can be seen from these images, a small variation of the parameters can dramatically affect the features extracted from the image. Recent edge detectors [17] exhibit a similar sensitivity to parameters.

The rest of the paper is organized as follows. Section II.A gives a high-level overview of the proposed runway detection and tracking algorithm. The core vision-based feature extraction algorithm is described in Sec. II.B, and formulated as an optimization problem. Section III introduces the KF formulation of the runway edge detection and tracking algorithm. We validate in Sec. IV the performance of this algorithm using experimental landing videos obtained by the Saudi Department of Aerospace and from the University of Texas, Austin. In this section, we show that runway detection and tracking is achieved in a very vast variety of initial conditions and lighting conditions, without any modification of the parameters of the algorithm. Refer to Fig. 2 for the algorithm structure.

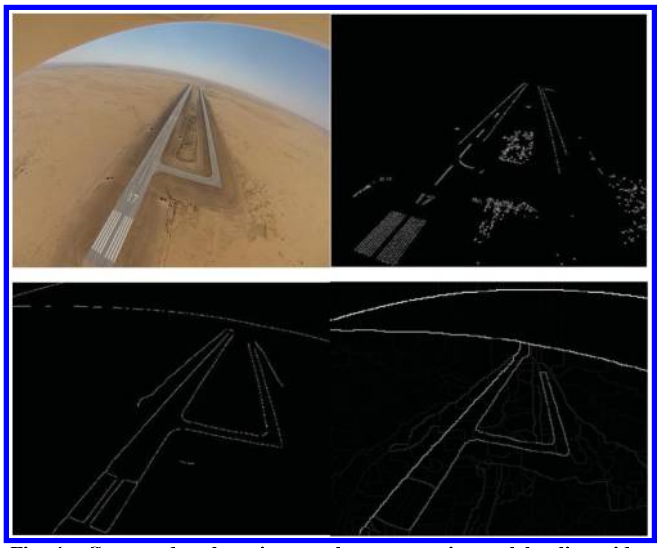


Fig. 1 Canny edge detection results on experimental landing video image: the original image (top left), along with the output of the Canny edge detector for various parameter settings. Small changes in the parameters of the detector could yield drastically different results.

II. Runway Detection and Tracking Algorithm Structure

In this section, we formulate a visual model of runways in aerial images. We then use the model to formulate an optimization problem to detect and segment the runway on a given image, based on the minimization of an energy function. Our key innovations are a model and optimization methods that are specifically designed to lead to efficient computational algorithms that can be implemented on embedded platforms from off-the-shelf UAVs. Furthermore, the algorithms are designed to be robust to the background clutter, illumination conditions (e.g., day and night), and noise. These features are prevalent in aerial imagery obtained during the takeoff and landing phases and make the detection and tracking of a runway significantly more complex.

A. Model and Optimization Problem

We model the runway as two line segments (one for each edge of the runway). Typical runways are flat enough to be approximated as two parallel straight lines, which due to perspective projection may appear nonparallel in the image. Although runways may consist of multiple intersecting runways, in which case the two-line-segment assumption may fail, we wish to detect the runway localized to a specific region around the longitudinal axis of the aircraft, where the two-line-segment assumption is largely true, and any deviations from it will be corrected with a filtering strategy described in the next section. Such a model is simple and robust, which makes the image processing faster and more reliable. We also assume that the image intensity statistics are *maximally* different in a neighborhood of each of the edges delimiting the runway. This translates the fact that the color and intensity associated with the runway is very different from the color and intensity associated with the neighborhood of the runway. Figure 3 illustrates this example, in the case of a runway located in a desert. This is the main assumption used by the proposed algorithm and is true in most settings. We make no assumption to the rest of the image, such as the background and the interior of the runway, as runways can have very different features, including ground markings, or the presence of taxiways.

For simplicity, we assume that the aircraft is approximately aligned with the runway at the initial time. This assumes that either the pilot flies toward the runway or that the autopilot flies the UAV in the general direction of the runway. This hypothesis is not restrictive in practice because commercially available GPS and inertial sensors have a positioning uncertainty of less than a few tens of meters (typically less than 10 m for the GPS), and uncertainty in heading measurement of fewer than 10 deg. In this case, within a rectangle around the runway, we may assume that both line segments representing the runway intersect the top and bottom of the rectangle. The rectangle corresponds to the approximate localization of the runway within the image and can be initialized with a guess of the approximate position of the runway, obtained, for example, using the navigation system of the UAV, from approximate coordinates of the runway.

The line segments representing the runway on the image can thus be described by four coordinates d_i, $i = 1, \ldots, 4$, representing the distance between the corners of the rectangle and the endpoints of the line segments, as illustrated in Fig. 3.

We now formulate an optimization problem to determine the values of the coordinates d_i, $i = 1, \ldots, 4$ that best represent the current position of the runway, within the image. As noted earlier, the runway is such that the neighboring regions on either side of the line segments have maximally differing image intensity statistics. For simplicity, we assume that the statistics are the mean RGB values with the neighborhoods, but any other statistics could be used, such as intensity histograms, statistics of filter bank responses, or statistics on a specific color hue (e.g., if the runway is known to be in a grassy, snowy, or desert environment). Let the image be denoted by

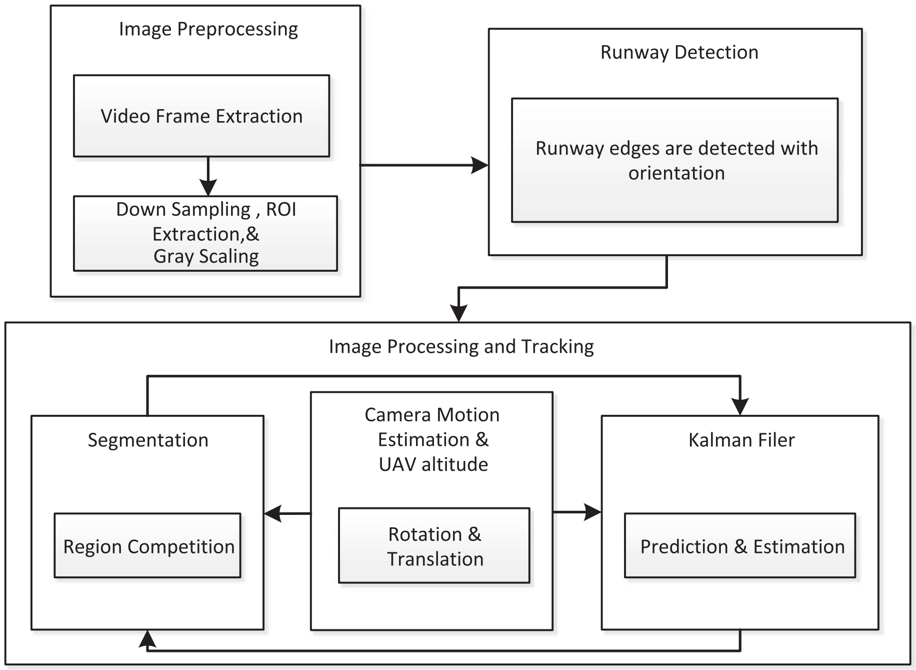


Image Preprocessing
Video Frame Extraction
Down Sampling , ROI Extraction,& Gray Scaling
Runway Detection
Runway edges are detected with orientation
Image Processing and Tracking
Segmentation
Region Competition
Camera Motion Estimation & UAV altitude
Rotation & Translation
Kalman Filer
Prediction & Estimation

Fig. 2 Overview of the runway detection and tracking algorithm developed in this paper.

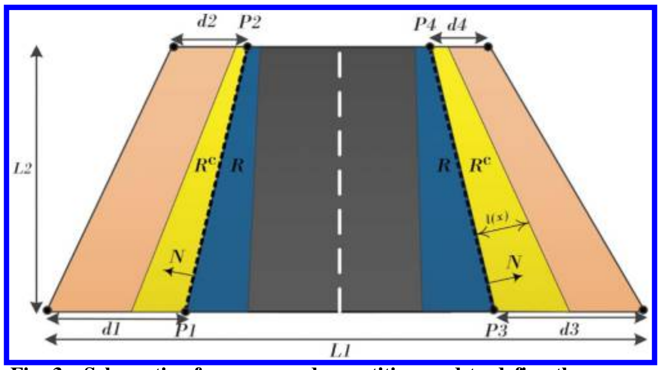


Fig. 3 Schematic of runway and quantities used to define the energy for runway detection considering the perspective projection using measurements of the orientation and position of the camera.

$I{:}\Omega \to \boldsymbol{R}^k$ (with $k = 3$ for RGB color encoding), where $\Omega \subset \boldsymbol{R}^2$, and let R, $R^c \subset \Omega$ denote the inside and outside neighborhood regions of the line segments, respectively, as in Fig. 3. The mean statistics $u, v \in \boldsymbol{R}^k$ for the inner and outer neighborhoods defined at a certain point x on the line segment become

$$u(x) = \frac{1}{\ell(x)} \int_0^{\ell(x)} I(x - sN)\,\mathrm{d}s,$$
$$v(x) = \frac{1}{\ell(x)} \int_0^{\ell(x)} I(x + sN)\,\mathrm{d}s \tag{1}$$

where N is the unit outward normal to the line segment corresponding to either side (right or left) of the trapezoid, $\ell(x)$ is the length of the line segment, ds is the arclength differential element of the segment defined on $[x, x - \ell(x)N]$ or $[x, x + \ell(x)N]$ for u, v respectively, and we would define that as the spatially variant mean calculation.

As a result of the perspective projection from three-dimensional (3D) to two-dimensional (2D), $\ell(x)$ would be a function of the coordinate x in the image. Consequently, this gives rise to the trapezoidal neighborhood around the line segments using measurements of the orientation and position of the camera. Refer to Sec. III.B for details on computing $\ell(x)$.

For the special case, having R, R^c nonspatially variant, the mean statistics expression can be computed as the means within the regions rather than the line segments:

$$u = \frac{1}{|R|} \int_R I(x)\mathrm{d}A, \qquad v = \frac{1}{|R^c|} \int_{R^c} I(x)\,\mathrm{d}A \tag{2}$$

where $|R|$ denotes the area of R, and dA is an area element.

Because we would like to define the optimal line segments such that the two statistics, u and v, are maximally different; our objective is to minimize the following energy function:

$$E(d_1, d_2, d_3, d_4) = -\frac{1}{2} \int_{\partial R} |u(x) - v(x)|^2\,\mathrm{d}s(x) + \gamma \sum_{i=1}^{4} |d_i - d_i'|^2 \tag{3}$$

where ∂R is the boundary of R that is allowed to vary, and $ds(x)$ is the arclength differential element of the segment defined on $[0, L_2]$.

Note that u and v are functions of the position of the line segments because R and R^c depend on the line segments, which are in turn specified by d_i. Therefore, the energy E itself is only a function of the coordinates d_i, $i = 1, \ldots, 4$.

Because the image is not expected to move too much between frames, we add a regularization term to penalize abrupt changes of the coordinates d_i, $i = 1, \ldots, 4$ from the initial guess denoted by d_i', $i = 1, \ldots, 4$, which will be obtained from a prediction specified in the next section. This prevents over segmentation if the region around the runway include outliers to our model. Higher values of γ will limit the freedom of the lines' displacement toward convergence and vice versa. For practical implementations, γ can be set a priori, or adaptive, depending on the amount of clutter in the neighborhood of the runway.

As will be shown later in the paper, the use of region statistics in the design of the energy leads to a robust runway detection algorithm. Local derivatives, which are widely used in edge detection, are sensitive to noise and extraneous features in the images. Thus, using statistics that are more robust to such disturbances will lead to an algorithm that is less sensitive to irrelevant image features. The larger the neighborhoods, R and R^c, the more robust the statistics are to local disturbances. However, larger neighborhoods encompass more of the background where our assumptions of maximally differing statistics of u and v may not hold due to irrelevant clutter. This tradeoff in size is important and will be analyzed in the subsequent sections.

The energy above is related to a large body of literature in image segmentation by partial differential equations–based methods [18–20] in which the energies are defined on the set of (infinite dimensional) regions. The energy is most closely related to [21], where the objective is to divide the image into two disjoint regions that have maximally different intensity means. Our energy formulation has two major improvements with respect to previously developed formulations:

1) The energy is defined on a simplified representation of regions defined by line segments to support real-time applications needed for UAV landing.

2) We rely on neighborhoods rather than the entire image to avoid background clutter typical in aerial imagery.

One problem with image segmentation methods based on partial differential equations is that they usually rely on local gradient descent methods, because the energy function is nonconvex (as is our energy of interest). Therefore, they require the user to initialize the algorithm properly so that the algorithm converges toward a desirable local minimum of the energy function. Our model of the runway is designed to achieve a computationally tractable detection algorithm that localizes the runway in real time on standard embedded computers.

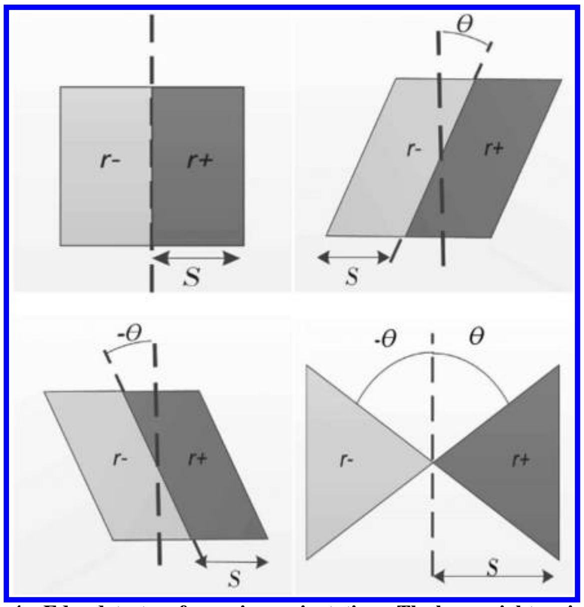


Fig. 4 Edge detectors for various orientations. The lower right region is the intersection of all $r+$ and r_- within a specified range $[-\theta_m, \theta_m]$; it is able to detect lines with any orientation without searching within the range.

B. Detection Method

The evaluation of $E(d_1, \ldots, d_4)$ for every $(d_1, \ldots, d_4)$ on a four-dimensional computational grid would be computationally prohibitive and too slow for real-time runway tracking applications. To minimize the number of candidate grid points, we need to have a good initial guess of the position of the runway. In the present application, we cannot assume that the runway is approximately located initially, because its coordinates may be unavailable, or position sensors may be malfunctioning (e.g., in GPS-denied environments). Therefore, local optimization techniques, which are the only candidates for optimization of E due to nonconvexity, could fail to initially detect the runway and possibly lead to a crash if the runway position information is used in the control loop. To address this issue, we construct an algorithm that efficiently searches per the set of candidates $(d_1, \ldots, d_4)$. The algorithm prunes away irrelevant line segments in this set and proposes a few candidate segments as potential initialization points. The energy of these few candidates can then be evaluated, and the energetically most favorable candidate can be selected. This allows us to detect the runway immediately, without any convergence period, and without initializing the algorithm with the estimated initial position of the runway.

To identify candidate pairs of line segments outlining the runway, we first focus on generating candidate single line segments separating the runway with the rest of the image. Each line segment is described by two coordinates: θ ($\theta \in [-\pi/2, \pi/2]$), which corresponds to the angle of the line segment with respect to the vertical axis, and the horizontal position x of the center of the line segment. The present algorithm efficiently computes minimizers of

$$E'(x, \theta) = -\frac{1}{2}(\bar{u}(x, \theta) - \bar{v}(x, \theta))^2 \tag{4}$$

where $\bar{u}$ and $\bar{v}$ are the mean values of the chosen statistics on either side of the segment described by x, θ.

Evaluating E' for all x, θ on a 2D computational grid would be too slow for real-time applications. Therefore, we introduce a hierarchical search, to first determine candidate locations x, and then determine the second coordinate θ. For a fixed x, a diagram illustrating the neighborhoods (called $r_+(\theta)$, $r_-(\theta)$) of the line segments corresponding to different orientations θ is shown in Fig. 4.

Let us now consider the intersections $r_{+,\text{inter}} = \cap_{\theta\in\mathcal{D}}\, r_+(\theta)$ and $r_{-,\text{inter}} = \cap_{\theta\in\mathcal{D}}\, r_-(\theta)$ of the neighborhoods r_+ and r_- for θ ranging in $\mathcal{D}$. These intersected regions are shown in the bottom right in Fig. 4. If the difference between $r_{+,\text{inter}}$ and $r_{-,\text{inter}}$ is small, then the coordinate x is not a good candidate location for a runway edge. Such locations are thus eliminated from the search, allowing us to quickly determine candidate locations x. We define the *response* $\mathcal{R}$ as $(r_{+,\text{inter}} - r_{-,\text{inter}})^2$. The response is only a function of the coordinate x. An example of response function is shown in Fig. 5. The left subfigure shows the response function of the image, associated with coordinates centered between the two horizontal green lines of the center subfigure, whereas the right subfigure depicts the determination of the correct candidate coordinates x_i, θ_i.

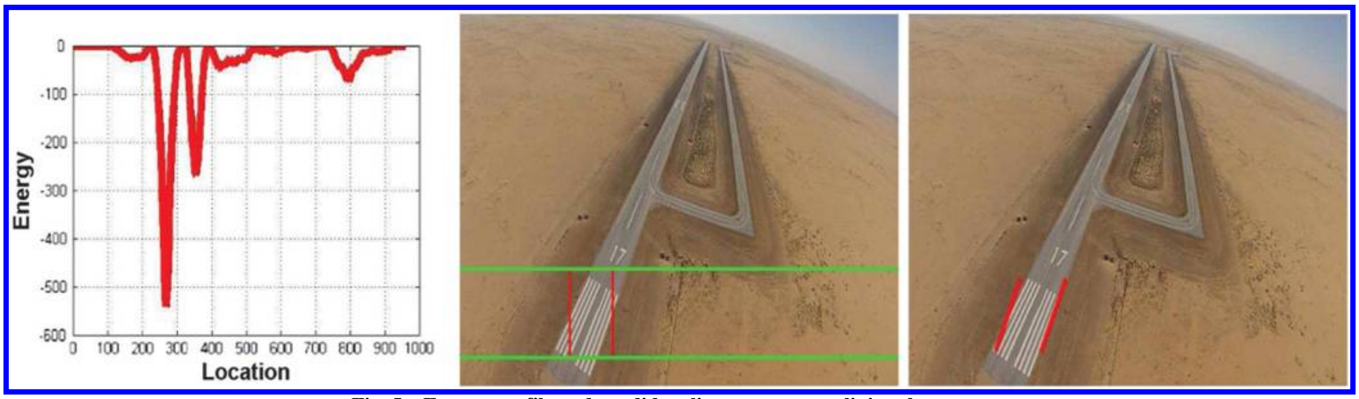


Fig. 5 Energy profile and candidate line segments outlining the runway.

Algorithm 1: Runway detection.

```
1) For each $x \in \{0, 1, \ldots, L_1 - 1\}$
     a) compute $u$ and $v$ with regions defined by the mask in Fig. 4 (bottom, right) and centered at $(x, L_2/2)$ for $\theta = \pi/4$
     b) compute $\mathcal{R}(x) = -(u - v)^2/2$
2) compute the $N$ lowest local minima of $\mathcal{R}$, called $x_i$, $i = 1, \ldots, N$
3) compute the $\theta \in \{-\pi/4, -\pi/8, 0, \pi/8, \pi/4\}$ corresponding to $x_i$ that minimizes $E$ for $i = 1, \ldots, N$
4) compute $E$ for all pairs of line segments determined by $(x_i, \theta_i)$
5) output the pair of line segments $(x_i, y_i)$, $(x_j, y_j)$ that have minimum $E$
```

The best candidate locations of line segments corresponding to edges of the runway are local maxima of $\mathcal{R}$, that is, the locations associated with the highest local responses. Because $r_{+,\mathrm{inter}}$ has fewer pixels than any particular r_+, the means $r_{+,\mathrm{inter}}$ and $r_{-,\mathrm{inter}}$ are more susceptible to noise, and therefore computing local minima may generate false positives. We eliminate these false positives by doing a search over a coarse grid $\theta \in \{-\pi/4, -\pi/8, 0, \pi/8, \pi/4\}$, *only* at the candidate local maxima locations, and retain only candidate locations that minimize the energy function. The actual orientation is then determined by performing a search on a fine grid. This procedure gives candidate coordinates (x_i, θ_i), $i = 1, \ldots, N$ for line segments of the runway, where N corresponds to the number of candidates retained by the algorithm. In subsequent implementations we choose $N = 5$.

Once we have obtained the candidate line segments, we compute the energy E for pairs of line segments in the candidate list (x_i, θ_i). The pair with the lowest energy is selected as the detected runway, after running $N(N-1)$ energy computations. Because N is small, this procedure is considerably faster than evaluating the energy of all possible pairs of line segments, corresponding to a search over a four-dimensional grid.

The algorithm used for the runway detection procedure is summarized in Algorithm 1.

To link the image processing in 2D and the data fusing in 3D, a rough location estimate in 3D is given via GPS (runway is in the vicinity). Hence, the 2D image processing-lane detection starts to work, only by sweeping horizontally, between the green lines in lower part of the image where originally the runway starts to appear. This distance between the green lines is related to the elevation of the UAV above the ground; with lower altitude it increases. It can be roughly estimated based on observations from landing videos, and then to be saved as a look-up table.

C. Local Optimization for Refined Segmentation

Once a candidate runway has been identified, we run a local optimization method to minimize E on an image, given an initialization close to the desired runway. We choose a local optimization approach for multiple reasons:

1) The detection procedure outlined in the previous section only gives us a coarse approximation of the position and orientation of the UAV, since it is obtained by performing a search over a relatively coarse computational grid.

2) Assuming that the runway was correctly localized on the previous frame, we can process the current frame with a very good initial guess (assuming that the frame rate of the camera is sufficient, and the dynamics of the UAV are sufficiently slow). Therefore, local optimization is sufficient to solve the problem, and allows both higher precision and faster computational time than a search over the entire image.

3) A local optimization approach allows us to regularize the solutions with respect to time, if multiple runway candidates are possible (e.g., if in the presence of multiple parallel runways or taxiways).

Local optimization is relatively fast, and less computationally costly than running the global detection procedure outlined in the previous section. Therefore, a gradient descent method [22,23] is used to minimize E. One may interpret the energy E as defined on a closed contour, that is, a trapezoid formed by the line segments. This description allows us to use results from the partial differential equation (PDE) image segmentation literature [18–20], which have derived methods for optimizing arbitrary energies defined on closed contours that are allowed to deform arbitrarily. However, in optimizing E, we would like to enforce the constraint that the trapezoidal shape of the runway section remains a trapezoid on the subsequent frame. This type of constraint can be enforced by restricting the possible contour deformations to trapezoid preserving deformations [24].

The gradient of the cost function is the sum of two terms, one term associated with the energy E: $(\nabla E)_i = (\partial E/\partial d_i)_i$, for $i \in \{1, \ldots, 4\}$ (this term can be computed as line integrals, as shown in Proposition 1), and a second term, $2\gamma(d_i - d_i')$, associated with the regularization term of the objective function.

The gradient descent algorithm thus optimizes the parameters d using a recursive equation

$$d_{k+1} = d_k - \Delta d \nabla E(d_k) \tag{5}$$

where d_k are the coordinates computed at step k. We initialize d_0 with the coordinates of the runway associated with the previous frame, and choose an adaptive step size $\Delta d = 0.5/\|\nabla E(d_k)\|$ to speed up convergence. With these assumptions, we can now determine the gradient as follows:

Proposition 1: The gradient of E is given by $\nabla E = (\partial E/\partial d_i)_{i=1}^4$, where the partial derivatives are given by:

$$\frac{\partial E}{\partial d_1} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_1}(t)) - u_{L_1}(t))\,\mathrm{d}t + \frac{1}{|R^c|}\int_0^1 (t-1)(I(C_{L_1}(t)) - v_{L_1}(t))\,\mathrm{d}t\right] \tag{6}$$

$$\frac{\partial E}{\partial d_2} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 t(I(C_{L_1}(t)) - u_{L_1}(t))\,\mathrm{d}t + \frac{1}{|R^c|}\int_0^1 t(I(C_{L_1}(t)) - v_{L_1}(t))\,\mathrm{d}t\right] \tag{7}$$

$$\frac{\partial E}{\partial d_3} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_2}(t)) - u_{L_2}(t))\,\mathrm{d}t + \frac{1}{|R^c|}\int_0^1 (t-1)I((C_{L_2}(t)) - v_{L_1}(t))\,\mathrm{d}t\right] \tag{8}$$

$$\frac{\partial E}{\partial d_4} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 t(I(C_{L_2}(t)) - u_{L_2}(t))\,\mathrm{d}t + \frac{1}{|R^c|}\int_0^1 t(I(C_{L_2}(t)) - v_{L_2}(t))\,\mathrm{d}t\right] \tag{9}$$

$$u_{L_i}(t) = \frac{1}{\ell(t)}\int_0^{\ell(t)} I(C_{L_i}(t) - s\cdot N)\,\mathrm{d}s \tag{10}$$

$$v_{L_i}(t) = \frac{1}{\ell(t)} \int_0^{\ell(t)} I(C_{L_i}(t) + s \cdot N)\,\mathrm{d}s \tag{11}$$

where C_{L_i} is the arc-parameterization of the ith line segment corresponding to $(d_1, \ldots, d_4)$; L_2 is the image width; u and v are the means inside the regions R and R^c, respectively; N_i is the unit outward normal to C_{L_i}; and ds is the arclength differential element of the segment defined on $[C_{L_i}(t), C_{L_i}(t) - \ell(t)N]$, $[C_{L_i}(t), C_{L_i}(t) + \ell(t)N]$, for $u_{L_i}(t), v_{L_i}(t)$, respectively.

III. Tracking

A. Dynamical Model of the Camera

Because virtually all UAVs are equipped with positioning and inertial systems, we can leverage additional data generated by the IMU and positioning systems of the UAV to better initialize the gradient descent algorithm in highly turbulent conditions, or when the camera has a low frame rate (e.g., in poor lighting conditions). In earlier work [25], we assumed that the apparent velocity of the runway was approximately constant. This approximation holds only if the attitude of the UAV does not change too abruptly over time. In this paper, we fuse the inertial and positional data generated by the UAV sensors to better estimate the position of the runway on a new frame, allowing a faster convergence of the gradient descent method outlined in the previous section. Because our dynamic model is the runway position in the 3D world coordinates of a reference frame, the method arising from the model also allows one to estimate the relative position between the UAV and the runway, which is important in closed-loop operation, when the relative position is fed into a controller of the UAV.

We now describe our representation for the runway, which then allows us to define its dynamic model. Instead of representing the runway as four sets of coordinates (positions of the start and end points of the segments limiting the runway), we assume that the runway is described by two infinite parallel lines. We also enforce geometrical constraints on the runway that the runway can be described by two parallel lines in the Earth frame. These two previous assumptions allow us to reduce the dimensionality of the problem, with only four independent coordinates instead of eight:

$$\left\{\begin{array}{ll} ax + by + c = 0, & \text{Line 1} \\ ax + by + d = 0, & \text{Line 2} \end{array}\right\} \tag{12}$$

where (x, y) describe points on the ground $(z = 0)$ on the lines, and $a, b, c, d \in \boldsymbol{R}$ are parameters of the lines. Because a given line of equation $ax + by + c = 0$ can be equivalently described as $\alpha ax + \alpha by + \alpha c = 0$ for nonzero α, we further impose

$$\sqrt{a^2 + b^2} = 1 \tag{13}$$

to enforce uniqueness.

Our dynamical model for the runway is simply that the runway remains stationary with respect to the world frame, which implies that a, b, c, d remain constant across time. To fuse all sources of information (camera, inertial, and positioning sensors), we rely on a KF approach, in which the position and attitude of the UAVs are updated from the inertial and positioning sensors, allowing us to predict the coordinates of the runway at a future time in the image, using current runway coordinates. This requires us to track the attitude of the UAV (through its Euler angles: pitch, roll, and yaw), and compute the position of the runway on the image through a mapping that is a function of rotation, translation, and camera distortion. We can thus write the runway position estimation problem as a norm-constrained KF [26], with the following dynamical model:

$$x_k = (a, b, c, d)^T \in \boldsymbol{R}^4 \tag{14}$$

$$x_{k+1} = Ax_k + Bu_k + \zeta_k \tag{15}$$

$$y_{k+1} = C_k x_k + \eta_k \tag{16}$$

$$\zeta_k \sim \mathcal{N}(0, Q) \tag{17}$$

$$\eta_k \sim \mathcal{N}(0, U) \tag{18}$$

$$\text{s.t.} \|x^T \tilde{A} x\| = 1 \tag{19}$$

$$a \geq 0 \tag{20}$$

where the state x_k consists the parameters of the runway lines, y_k is measurements of the projection of the lines in the imaging plane, and ζ_k, η_k are model and measurement noise. The noise in the model could result due to, for example, deviations from the infinite parallel line assumption, and the measurement noise could arise because the measurement is obtained through a segmentation process. The last two constraints enforce uniqueness of the our representation for the lines. The matrices above are defined as

$$A = \left[\, \mathrm{id}_{4\times4} \,\right] \tag{21}$$

$$B = 0 \tag{22}$$

$$U = 0.01 \times \mathrm{id}_{4\times4} \tag{23}$$

$$Q = 0.01 \times \mathrm{id}_{4\times4} \tag{24}$$

$$\tilde{A} = \begin{bmatrix} \mathrm{id}_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix} \tag{25}$$

In the above definitions, the matrix $\mathrm{id}_{i\times i}$ denotes the identity matrix of dimension i. Therefore, our dynamic model is that the parameters of the line do not change (up to modeling noise). The initial state estimate x_0 is chosen to be the output of the runway detection procedure outlined in Sec. II.B, which can be projected to the frame of the world by estimates of the altitude and orientation of the UAV. The crucial element of this dynamic model is the observation matrix C_k, which maps the Earth frame coordinates to the camera coordinates. In the following section, we derive the observation matrix C_k as a function of the attitude, position and camera parameters, which give rise to the matrices R_k, T_k and the projection Π.

B. Geometric Camera Model, Parameters, and Measurement Matrix

Cameras have an intrinsic geometric model [27] that describes the mapping between the Earth frame coordinates and the camera coordinate system. This mapping depends on the optical properties, orientation, and position of the camera, as illustrated in Fig. 6.

The transformation between the Earth frame coordinates and the camera coordinate system involves multiple elementary transformations: rotation, denoted by a 3×3 rotation matrix, and translation, denoted by a 3D offset vector t with respect to a reference point in the Earth frame. More precisely, the mapping between any point of coordinates $p = (x, y, z)^T$ in the Earth frame and its counterpart

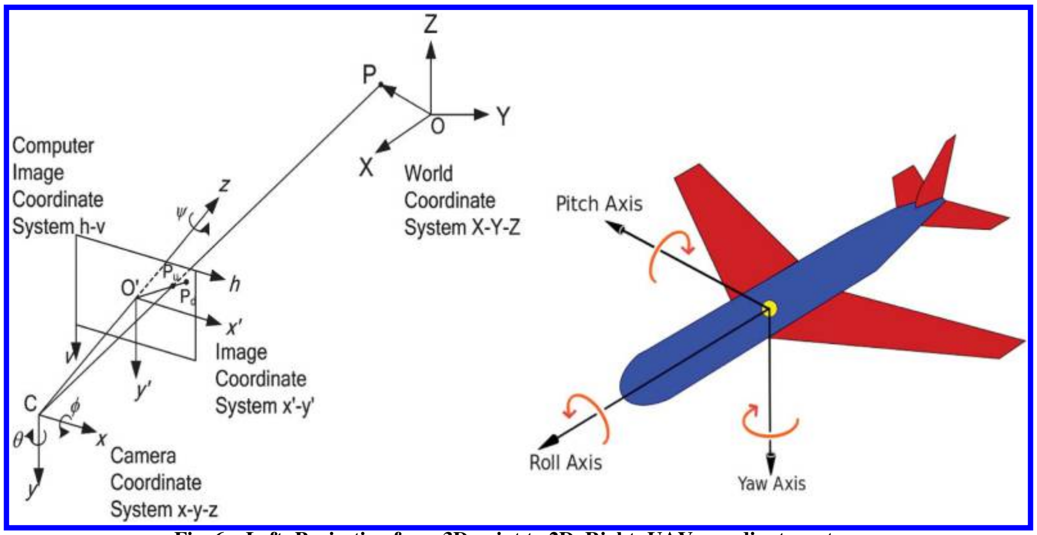

Computer Image Coordinate System h-v
World Coordinate System X-Y-Z
Image Coordinate System x'-y'
Camera Coordinate System x-y-z
Z
Y
X
O
P
O'
h
x'
y'
v
C
x
y
z
Pitch Axis
Roll Axis
Yaw Axis

Fig. 6 Left: Projection from 3D point to 2D. Right: UAV coordinate system.

coordinates $p' = (x', y')^T$ in the image is described by the following equation:

$$\begin{pmatrix} x_{\text{cam}} \\ y_{\text{cam}} \\ z_{\text{cam}} \end{pmatrix} = \kappa[R|t]\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \kappa\left[R\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}\right] \tag{26}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \Pi\begin{pmatrix} x_{\text{cam}} \\ y_{\text{cam}} \\ z_{\text{cam}} \end{pmatrix} = \begin{pmatrix} x_{\text{cam}}/z_{\text{cam}} \\ y_{\text{cam}}/z_{\text{cam}} \end{pmatrix} \tag{27}$$

In the above equation, the coordinates (x', y') of a point on the image is obtained by a projection Π, which denotes the projection of points represented in the camera frame to the plane perpendicular to the direction of the camera with distance 1 away from the optical center of the camera. It assumes that z_{cam} is the distance from this imaging plane. By an argument of similar triangles, the projection is given by division by the last coordinate as seen above. The matrix κ is a function of the intrinsic camera parameters: focal length, pixel size, and position of the principal point. Such parameters can be obtained by calibrating the camera [28], for example, using a calibration tool box [29]. The matrix κ corresponds to the composition of three transformations: $\kappa = \kappa_1 \times \kappa_2 \times \kappa_3$, respectively corresponding to translation, scaling, and shear.

$$\kappa_1 = \begin{bmatrix} 1 & 0 & x_o \\ 0 & 1 & y_o \\ 0 & 0 & 1 \end{bmatrix}, \qquad \kappa_2 = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix},$$
$$\kappa_3 = \begin{bmatrix} 1 & sf_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{28}$$

Using (12) one can find the equivalent transformation that could be applied to the vector (a, b, c) instead of the point $p = (x, y, z)^T$ using (26).

Because the points that we track are on the ground, we can assume that their altitude is zero $Z = 0$ (by setting the altitude reference to the runway altitude). Therefore, we can also write a linear relationship between the runway equation coefficients in the Earth frame and in the camera frame:

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = E\begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{29}$$

given that r_i and k_i are the ith column of R^t and κ, respectively. Then

$$E = \begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix}^{-T} \tag{30}$$

where

$$e_1 = \begin{pmatrix} k_1 \cdot r_1 \\ k_2 \cdot r_1 \\ k_3 \cdot r_1 \end{pmatrix}, \quad e_2 = \begin{pmatrix} k_1 \cdot r_2 \\ k_2 \cdot r_2 \\ k_3 \cdot r_2 \end{pmatrix}, \quad e_3 = \begin{pmatrix} k_1 \cdot t \\ k_2 \cdot t \\ k_3 \cdot t \end{pmatrix} \tag{31}$$

It can be written as dot product between the columns of the intrinsic parameters matrix and the extrinsic parameters matrix (i.e., rotation and translation).

Now, because we have two parallel lines that represent the runway in the world coordinate system as in Eq. (12), we get for "Line 1" and "Line 2", respectively,

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = E\begin{pmatrix} a \\ b \\ c \end{pmatrix}, \qquad \begin{pmatrix} d' \\ e' \\ f' \end{pmatrix} = E\begin{pmatrix} a \\ b \\ d \end{pmatrix} \tag{32}$$

By concatenating the two vectors at the left-hand side, we can find a total transformation:

$$\begin{pmatrix} a' \\ b' \\ c' \\ d' \\ e' \\ f' \end{pmatrix} = C_k\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \tag{33}$$

where C_k is a 6×4 matrix given by

$$C_k = \begin{pmatrix} e_1 & e_2 & e_3 & 0_{(3\times1)} \\ e_1 & e_2 & 0_{(3\times1)} & e_3 \end{pmatrix} \tag{34}$$

where e_i is the ith column of matrix E. The detailed derivation of C_k is shown in the Appendixes.

To conclude, matrix C_k will be used in the KF to project the lines from 3D to 2D, using the lines' parameters. Afterward, these lines would be refined using the segmentation based on the energy model. Because a neighborhood around the lines should be estimated efficiently, the matrix C_k can be used to determine the neighborhood. At each instance k, the lines parameters c and d particularly are shifted with a proper distance in 3D, to generate the neighborhood as straight lines around the edges of the runway in 3D. Then, they are projected into the image, using C_k, resulting a trapezoidal region parameterized by $\ell(x)$ defined in Sec. II.A. $\ell(x)$ would increase from up to bottom around the runway edges, due to the perspective projection, as originally $\ell(x)$ is uniform in 3D.

C. Camera Model Construction

The measurement matrix C_k derived in the previous section relies on matrices R and t, which are a function of the attitude and position of the UAV. We use a North, East, and Up orientation convention for the Earth frame, which requires some sign changes for angles and translation coefficients, in order to transform coordinates and rotations expressed in a right-handed coordinate system into a left-handed coordinate system. The orientation of the camera with respect to the Earth frame is the composition of a rotation of $-\pi/2$ with respect to the x axis of the UAV, with the rotation transforming the UAV coordinates to the Earth frame coordinates $R = Ry(-\theta)Rx(\phi)Rz(\Psi)$, where Ψ, ϕ, θ represent the Euler angles (respectively, roll, pitch, and yaw angles). Each elementary rotation is associated with a rotation matrix:

$$R_z(\Psi) = \begin{bmatrix} \cos(\Psi) & -\sin(\Psi) & 0 \\ \sin(\Psi) & \cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$R_y(-\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix},$$

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{35}$$

The translation vector mapping the origin of the Earth frame coordinates to the origin of the UAV coordinates requires the following measurement data:

1) The altitude of the UAV, denoted by a_t, and the altitude of the origin of the Earth frame, a_0

2) The latitude and longitude of the UAV, denoted by θ_{lat_t}, θ_{lng_t}

3) The latitude and longitude of the origin of the Earth frame, denoted by θ_{lat_0}, θ_{lng_0}

4) The radius of the Earth ρ

At time t, the translation vector mapping the Earth frame and the UAV frame origins is given by

$$t = \begin{bmatrix} \rho \cdot (\theta_{\mathrm{lng}_t} - \theta_{\mathrm{lng}_0}) \cdot \cos\left(\frac{\theta_{\mathrm{lat}_t} + \theta_{\mathrm{lat}_0}}{2}\right) \\ a_0 - a_t \\ \rho \cdot (\theta_{\mathrm{lat}_t} - \theta_{\mathrm{lat}_0}) \end{bmatrix} \tag{36}$$

Because the Euler angles estimated by the IMU are not very accurate, we rely on the images captured by the camera to estimate the translation and rotation changes more finely between consecutive frames. This approach is detailed in Sec. III.D. In an actual implementation of the system, we can rely on a fusion of inertial-camera-based measurements to estimate the rotation and translation changes.

Now, we mention how the state of the EKF initialized. In 2D (i.e., the image is used), the detection algorithm gives the location of the runway throughout the points $p1, \ldots, p4$, and then the lines parameters are retrieved using Eqs. (12) and (13). Then by using the reading of the GPS, altitude, and rotation of the UAV, for the detection moment, Eq. (33) is used to find the lines parameters in 3D, which initializes the filtering process.

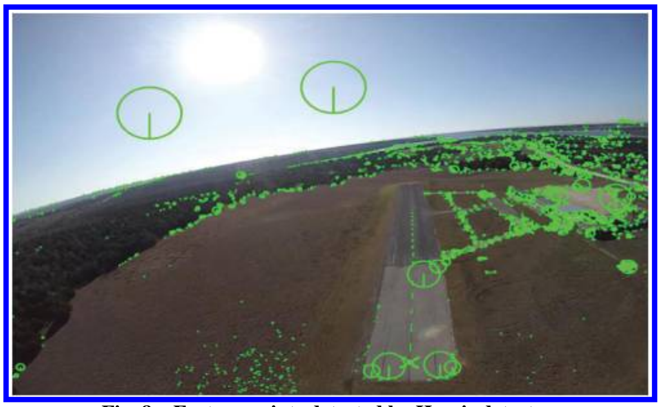


Fig. 8 Feature points detected by Harris detector.

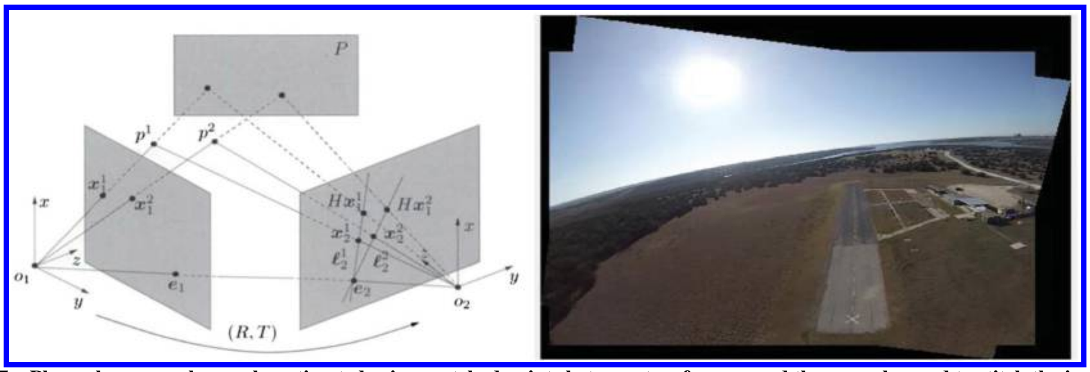


Fig. 7 Planar homography can be estimated using matched points between two frames, and then may be used to stitch the images.

Algorithm 2: Relative motion recovery

```
Input: $I_1$, $I_2$
Output: $R_{12}$, $t_{12}$
    1. Extract Harris features from $I_1$ and $I_2$ using $vl_{\mathrm{cov\,det}}$
    2. Find matched points $p_{1,i}^{\prime T}$, $p_{2,i}'$ using $vl_{\mathrm{ubcmatch}}$
    3. Use RANSAC to find $\tilde{H}$
    4. Decompose $\tilde{H}$ into $R_{12}$, $n$, $t_{12}/d'$
    5. Choose $\tilde{H}$ which satisfies the positive depth constraint
    6. Back project $p_{1,i}'$ into the Earth frame, then transform it into the first
       camera frame and find $p_{1,i}$
    7. Compute $d'$ by averaging $d_i'$ where $n^t p_{1,i} = d_i'$ for all $i$
    8. Compute $t_{12}$ by multiplying the translation from the homography
       decomposition by $d'$
```

D. Motion Recovery from Images

The camera is a visual sensor that can be fused in implementation to estimate the motion at a certain time. In other words, it can partially substitute using the IMU and GPS, where it suppress any errors due to turbulence in the flight or bad synchronization with video. Accordingly, an image-based algorithm is used to estimate translation and rotation changes between frames, and then composing such resultant transformation to find the final motion estimate at a certain time of the flight with respect to the Earth frame.

Our image-based algorithm for rotation and translation estimation uses the camera pinhole model. Images of the same scene can be stitched together if there exists one planar homography H that maps between adjacent images [27] (refer to Fig. 7). The left subfigure shows that a planer homography maps the left image to the right image under the assumption that the underlying scene is flat. Whereas the right subfigure shows the two frames stitched together using the estimating planar homography. This assumption holds as long as the scene in the Earth frames appears to be flat in the image plane, which is true because the ground is approximately flat compared to the altitude. A set of points in I_1, called feature points (refer to Fig. 8) should match with some points in the next image I_2, and these correspondences can be used to determine H. Feature points are detected such that they are discriminative and can be matched across frames, where in such aerial images, the corners of the runway on the ground, for instance, can be used as features, as well as other corners (refer to Fig. 8). Among different feature detectors, we found that the Harris detector can provide features of the borders of the runway, which makes sense for our application. We denote the feature locations or set of interest points in I_1 and I_2, after the matching process, by $p_{1,i}'$, $p_{2,i}'$, respectively, where i indexes the interest points. Note we assume that $p_{1,i}'$ in I_1 corresponds to $p_{2,i}'$ in I_2. Because the process of feature matching typically produces errors, we use RANSAC to remove possible inconsistent matches so that an accurate estimate of the homography can be determined from the inliers. The following system can be solved to estimate the homography from the corresponding locations of the interest points,

$$\hat{p}_{1,i}' H p_{2,i}' = 0, \quad \text{for all } i \tag{37}$$

where the hat operator denotes the skew symmetric matrix formed from the vector under the hat. Since the former formula assumes an identity calibration matrix, the resulting H should be premultiplied by κ^{-1} and post multiplied by κ, and this matrix is denoted by $\tilde{H}$. Afterward, it can easily be verified that $\tilde{H}$ transforms the first frame to the second one. Refer to Fig. 7. $\tilde{H}$ can be decomposed as follows:

$$\tilde{H} = R_{12} + \frac{t_{12} n^T}{d'} \tag{38}$$

where R_{12} and t_{12} are the rotation and translation between two adjacent frames, respectively, $n = [n_1, n_2, n_3]^T$ is the unit normal vector of the 3D plane in the Earth frame in the coordinates of the first camera frame, and d' denotes the distance from the 3D plane to the optical center of the camera.

Decomposition is not unique and yields four solutions, and we choose the one that satisfies the positive depth constraint (see [27]) that all the homogeneous representations of the points should result in positive depth (in front of the camera) for the desired homography. Knowing that t_{12}/d' is retrieved from the decomposition, one can compute d using $n^T p_{1,i} = d_i$, and we take an average of the d_i to compute the depth d, where $p_{1,i}$ is the 3D point corresponding to p_1' in the first camera frame. Note that although we can estimate d from the altitude reading, this is often too noisy, and so we instead use the procedure described next.

%related by $p_i = \lambda_i p_i'$, and λ_i states the ambiguity inhibited when projecting points from 3D points in 2D, according to Eq.-(C1).

We back project p_1' to find its corresponding coordinates in the Earth frame, using Eqs. (C6) and (C7), where the assumption is the third component (with respect to the ground frame) is zero since the point lies on the runway, which is on the ground. After that, a transformation is applied to the point in the Earth frame, using the rotation R_1^T and translation $-R_1 t_1$ associated with image frame containing p_1', to find p_1, where the last component of the translation vector is set to be the altitude measured by the altimeter, and then d' is computed. For more details, the reader may refer to [27]. The algorithm for relative motion estimation between adjacent frames is summarized in Algorithm 2.

The above discussion focused on computing the rotation and translation between two consecutive times. We can now obtain the rotation and translation between time 0 and another time, by composing the mappings between frames. If R_{12} is the rotation and t_{12} is the translation between two adjacent frames, then given the motion (R_1, t_1) up to frame 1, we can find the motion (R_2, t_2) between time 0 and the second frame at the second frame as follows:

$$t_2 = R_1 t_{12} + t_1, \qquad R_2 = R_1 R_{12} \tag{39}$$

which is obtained by composition. This procedure can be continued to find the rotation and translation at any frame, given the initial motion $R_0 = \mathrm{id}_{3\times 3}$ and $t_0 = 0$ at the initial location in the Earth frame. Hence, the measurement matrix can be computed and used in our dynamical model through out KF.

In Algorithm 2, Step 2, the interest points and features are computed using VLFeat [30]. We use the Harris interest point detector and the SIFT descriptor to describe the regions of the image local to the interest points. We use the matching procedure proposed by Lowe [31] to match SIFT features; that is, for each feature in I_1 the distance between it and each feature in I_2 is computed. The best match (with the smallest distance) is selected as a match if the ratio of the lowest distance to the second smallest distance is greater than a given threshold, so as to reject ambiguous matches. Note that for speed, one can use SURF [32] for the descriptor and FAST corner detector [33] to achieve speed ups of the traditional SIFT and Harris detectors, respectively, at real-time speeds.

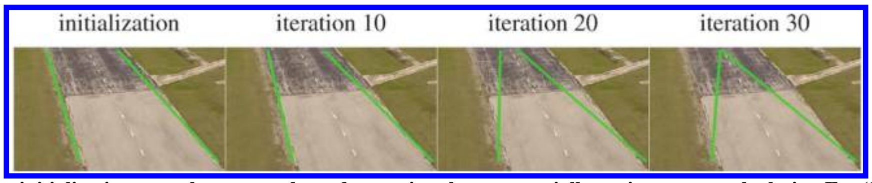


Fig. 9 GD evolution, given initialization near the runway boundary, using the non-spatially variant mean calculation Eq. (3). This illustrates a failure case for the non-spatially variant mean, and justifies the need for the spatially variant mean.

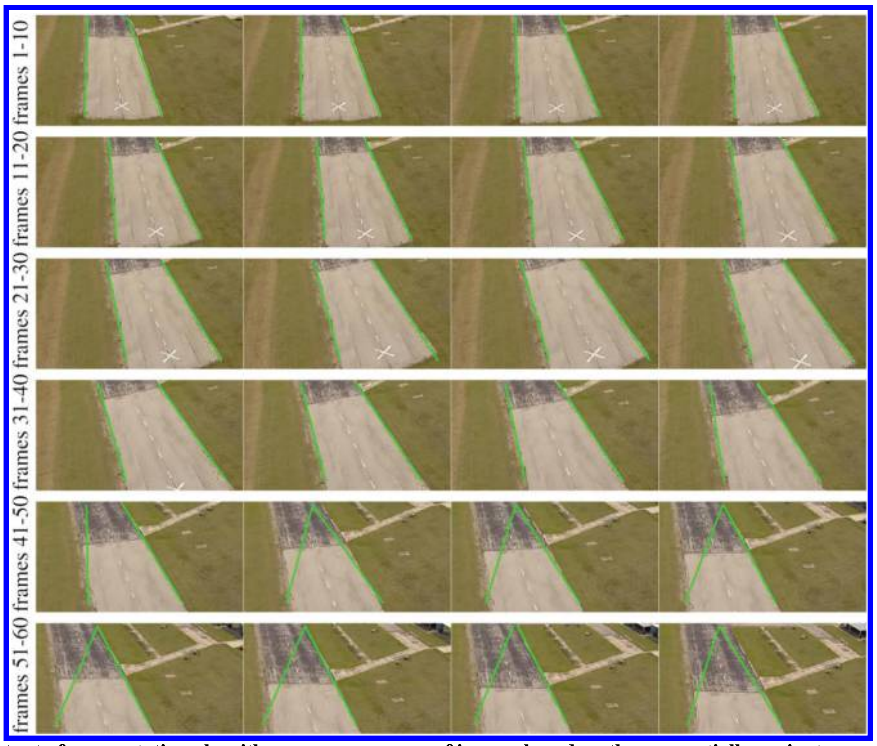


Fig. 10 Output of segmentation algorithm over a sequence of images based on the nonspatially variant mean calculation.

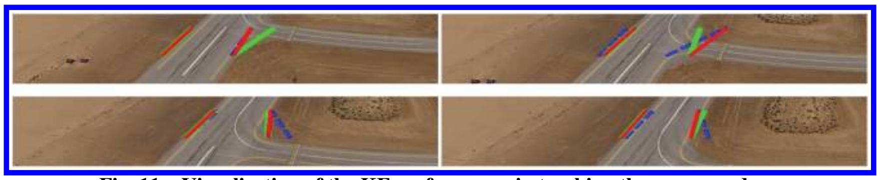


Fig. 11 Visualization of the KF performance in tracking the runway edges.

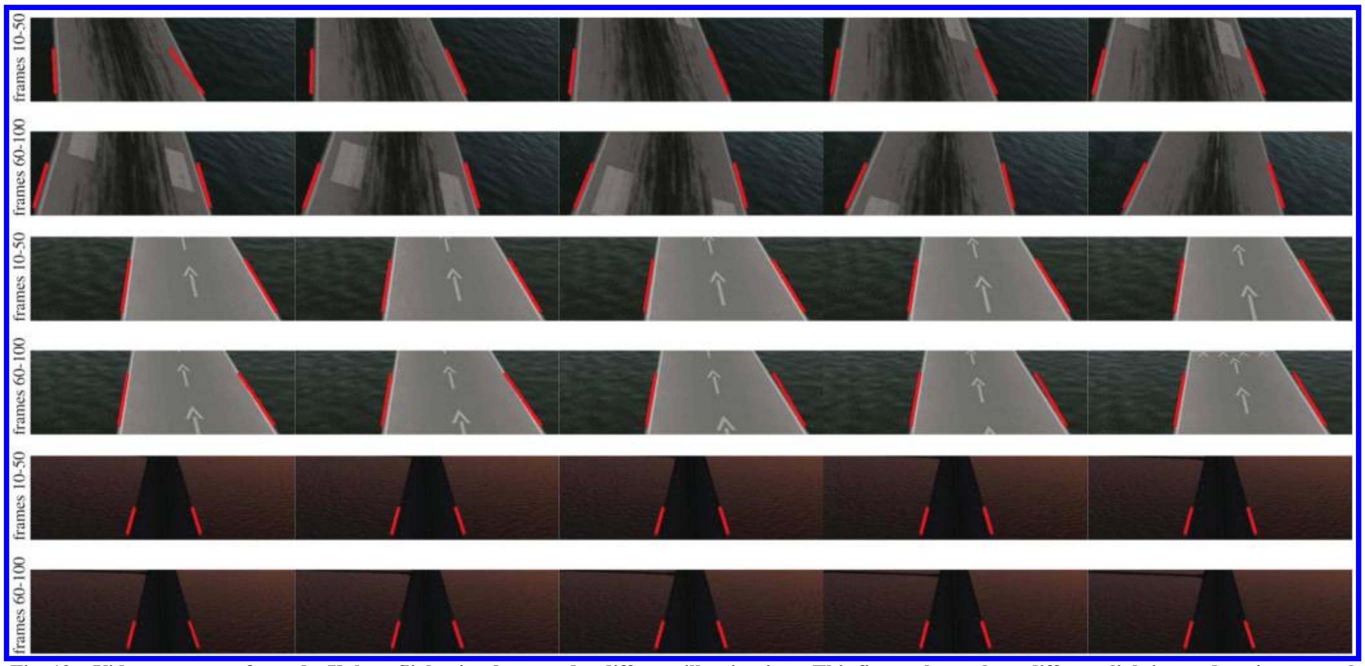


Fig. 12 Video sequences from the Xplane flight simulator under different illuminations. This figure shows three different lighting and environmental conditions, each represented over two rows.

IV. Experimental Results

A. Experimental Validation

We now evaluate the performance of the algorithm described in this paper over different video samples generated by UAV flying to the Austin Radio Control Associates (ARCA) runway in Austin, TX. These samples are obtained using an electric-powered `Bormatec Maja` fixed-wing UAV, from the Distributed Sensing Systems Laboratory of the University of Texas at Austin.

B. Segmentation Results

The gradient descent evolution is illustrated in Fig. 9 using one image over a number of iterations. This figure shows that minimizing the energy function using nonspatially variant mean (2) is not able to correctly segment the runway and thus justifies the minimization based on spatially variant mean (1). The latter more robustly tracks the runway since it is more robust to nonuniformities in the axis of the runway, such as ground markings or tire tracks, as we will see later. Of course, the non-spatially varying mean algorithm fails since the statistics of the runway changes in this particular frame. However, in Fig. 10, we show that the segmentation over the entire sequence. We only apply the runway segmentation algorithm over a sequence of images (without prediction from the dynamic model), where the initialization at a frame is the result of the segmentation algorithm from the previous frame. This sequence shows that energy based on nonspatially variant mean calculation (2) fails in segmenting the runway whenever the variations in image statistics along the runway axis are variant.

C. Tracking Results

Visualization of the KF performance in tracking the runway edges is shown in Fig. 11. We consider four consecutive time steps (left to right and up to down): blue line, predicted runway edge locations (from previous step); green line, runway edge locations (measurements) using vision-based segmentation (outlined in Sec. III of the present paper); red lines, estimation (output of the KF).

Figure 12 shows the performance of the tracker in [25], while Figs. 13 and 14 show how the runway is tracked under different lighting conditions, demonstrating the robustness of the proposed approach. The red lines show the output of the KF (which fuses the dynamics of the UAV to the observations of the runway detection algorithm) over a sequence of images. Reader may refer to [34] for videos.

Figure 15 shows a comparison of the performance of the two energy functions investigated in this paper. The motion information of the UAV helps the algorithm converge, though the constant velocity model (without Kalman filtering) performs adequately in low-turbulence conditions. The red lines show the estimated location of the runway using KF where an infinite weight on the prediction, whereas the measurements are in green and it is not considered in the filtering process although it is perfect. The 3D-based dynamical model outperforms the 2D-based model.

Setting infinite weight on the prediction in both models and having a perfect measurement, which its uncertainty, is extremely high so that only the prediction is affecting performance of the KF. In this experiment, although the measurement is perfect, its noise is set to be high only to show the robustness of the prediction model of tracking in 3D.

V. Quantitative Evaluation

In Table 1, a comparison is conducted between the ground truth runway position and the performance of the nonspatially variant mean–based model (2), the spatially variant mean–based model (1), and a HT with Canny edge detector approach. The performance metric used is the root mean square error in pixel position, using the ground truth as a reference, averaged over sequences of 50 consecutive frames.

The computational time results differ according to the hardware implementation. For the segmentation process, it boils down to a line integral over line segment, in addition to Kalman filtering that involves

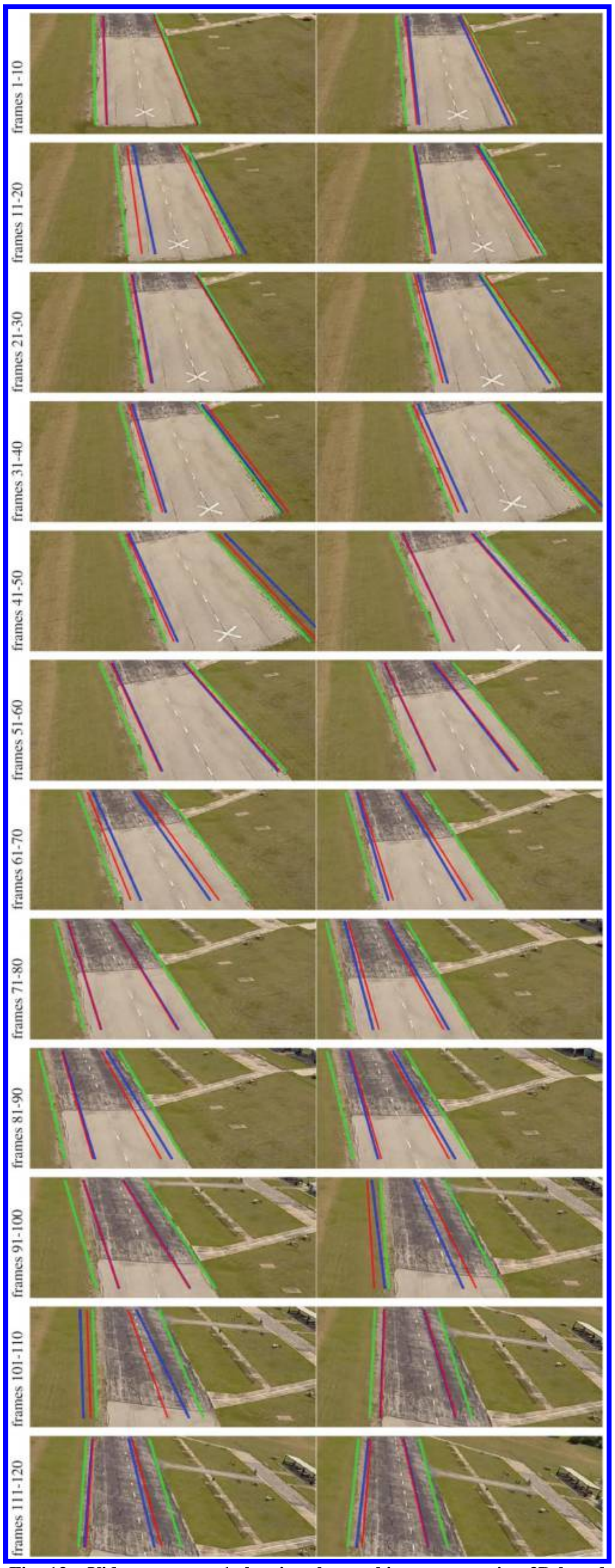


Fig. 13 Video sequence 1 showing the tracking output using 3D-based tracking model with spatially variant mean energy model for segmentation.

only matrix multiplications. This complete process requires 30 m · s using an Odroid U3X, which is considered as a small computer. Hence, the algorithm is fast enough to run at 20 Hz (or slightly more), which is comparable to the frequencies used by the high- and low-level control

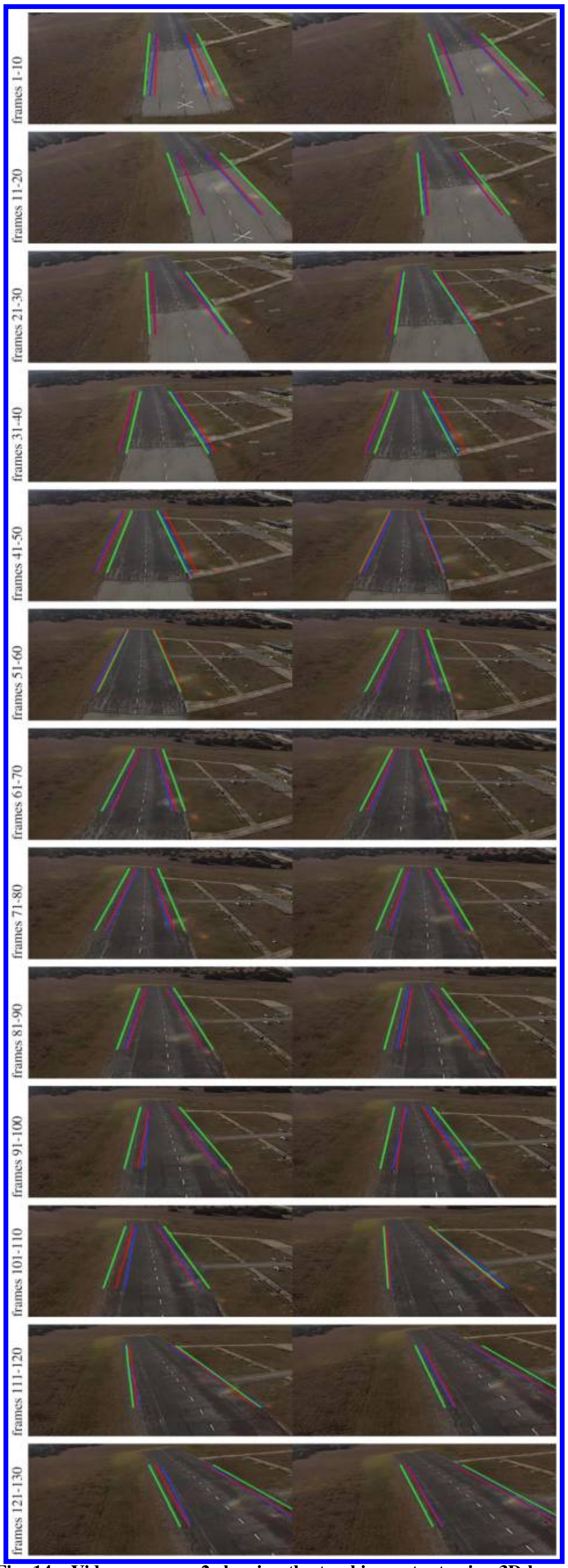


Fig. 14 Video sequence 2 showing the tracking output using 3D-based tracking model with spatially variant mean energy model for segmentation.

loops of actual consumer-grade UAVs (10 Hz for Pixhawk-based systems). For future work, a computer stick and NUC intel would be used to further improve the computational speed.

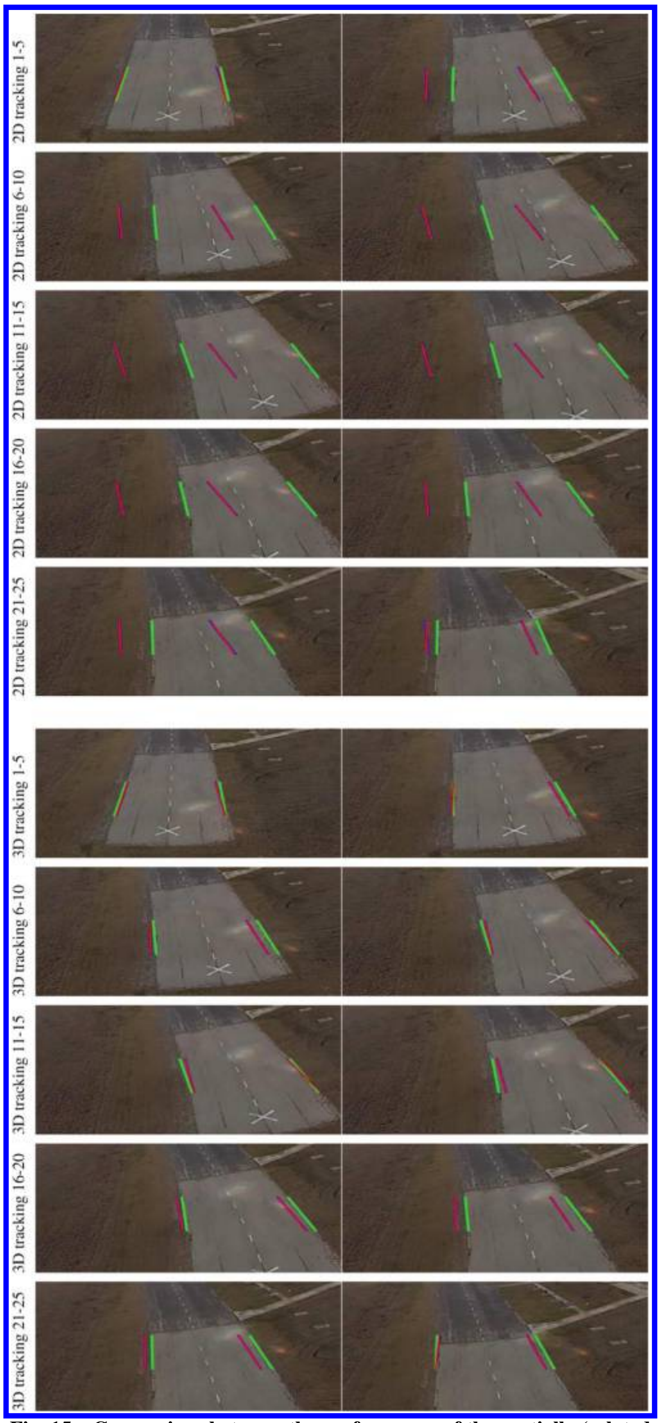


Fig. 15 Comparison between the performance of the spatially (related to 3D tracking) and nonspatially (related to 2D tracking) variant mean–based models.

VI. Discussion

The experimental comparison against HT and Canny edge detection can raise a question of a fairer comparison that would be against graph cut [35] or other appearance-based segmentation algorithms, which can also be quite fast. Looking at such based-

Table 1 Average pixel error in runway position estimation

Sequence	Spatially variant mean	Nonspatially variant mean	Hough transform and Canny edge detector
1	6.9 ± 3.6	30.9 ± 24.5	54.3 ± 99.4
2	7.3 ± 2.9	32.9 ± 22.6	57.9 ± 39.9

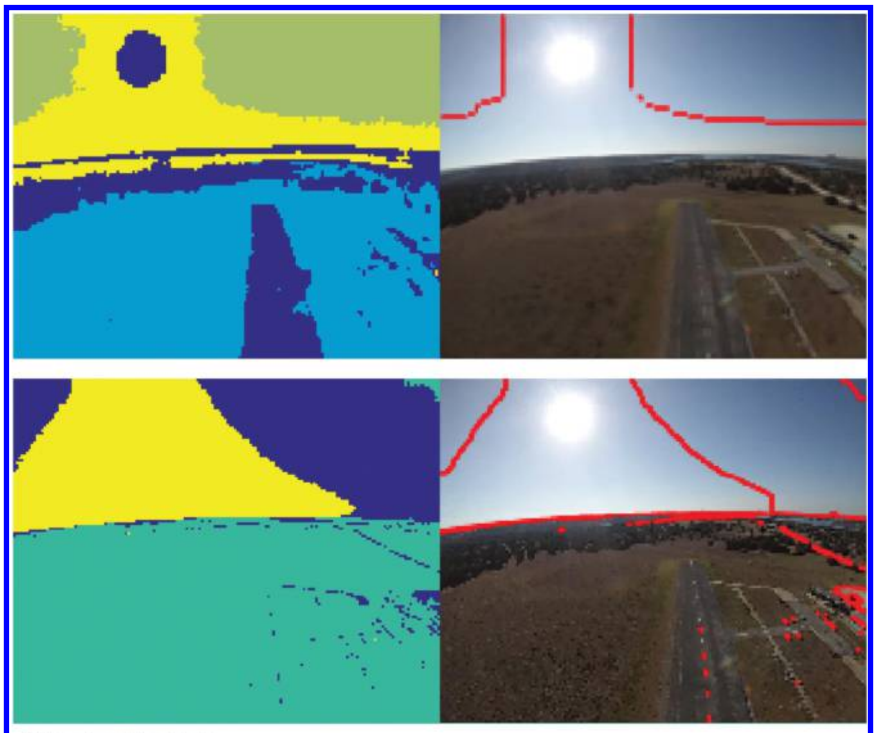
region segmentation algorithm was the first step to approach the runway detection. However, it requires user supervision by selecting a number of seed points to start clustering all pixels in the image (expensive) into groups based on their appearance similarities.

In more detail, this can be involved in the detection and segmentation steps. First, for the detection step. Our approach is generic enough to find edges directly without any other postprocessing step, such as edge detectors. Other methods, such as graph cut, would similarly use color-based segmentation energy function; with excellent conditions of not having any clutters, it would be subjected to loss of detection of the runway edges under difficult illumination conditions, whereas our method finds candidates of edges that could form the runway and then find their correlation response. If we assume that graph cuts are free of miss detections, then an extra step is required to find the edges of the region of interest of the runway (given that we restricted the search region for the graph cut method), and this would give many possible candidates, including the white land marks on the runway thresholds. To conclude, our detection method is robust to different illumination conditions, weather conditions, and runway marking layouts.

Second, for the segmentation. Graph cuts method requires from the users initial seeds, which are difficult to set. Hence, our method outperforms graph cuts in terms of user-free operation. Also, assuming that we are interested in two labeled segmentations, graph cuts based–color based segmentation methods, such as on k-means, histograms, or region competition, can end up at a local minima (causing a false detection). This can occur in the presence of landmarks, road marks, taxiways, clutter, or because of changes in illumination. Also, such results would violate the geometric constraint of having a trapezoidal region of the runway. On the other side, our algorithm is generic and can overcome the aforementioned nuisances and reduces the space of parameters to only four points, preserving the geometric properties of a trapezoidal shape.

The following figures show the challenge when using the graph cut method. On the left of Fig. 16a are a four regions output from graph cut, where the runway is clustered with the sun, part of the horizon, and part of the ground clutters. With no prior information, the runway cannot be extracted. On the right, the contoured output of the clustered image, based on a built-in tunable threshold, the runway cannot be located. Even when choosing to segment three regions, the runway is missed.

This illustrates a significant problem in making use of the graph cut algorithm in designing a fully automatic detection and segmentation algorithm; namely, it is difficult to choose a priori the number of


a) Graph cuts' output

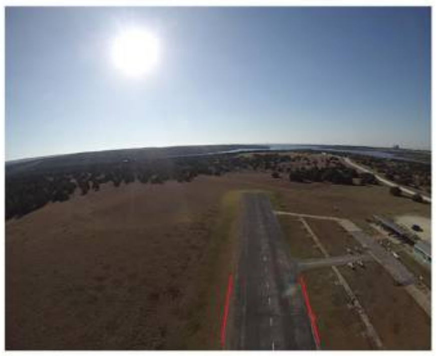

b) Our detection algorithm output

Fig. 16 Comparison of our segmentation and detection method with graph cuts. a) Result of graph cuts for four regions (top) and three regions (bottom). Graph cuts are highly sensitive to the number of regions input by the user. Thus, in some cases the runway is obtained and sometimes it is not. The images on the left are the segmentation and the images on the right are the corresponding boundaries. b) Our method correctly detects and segments the runway automatically—the main motivation for our approach.

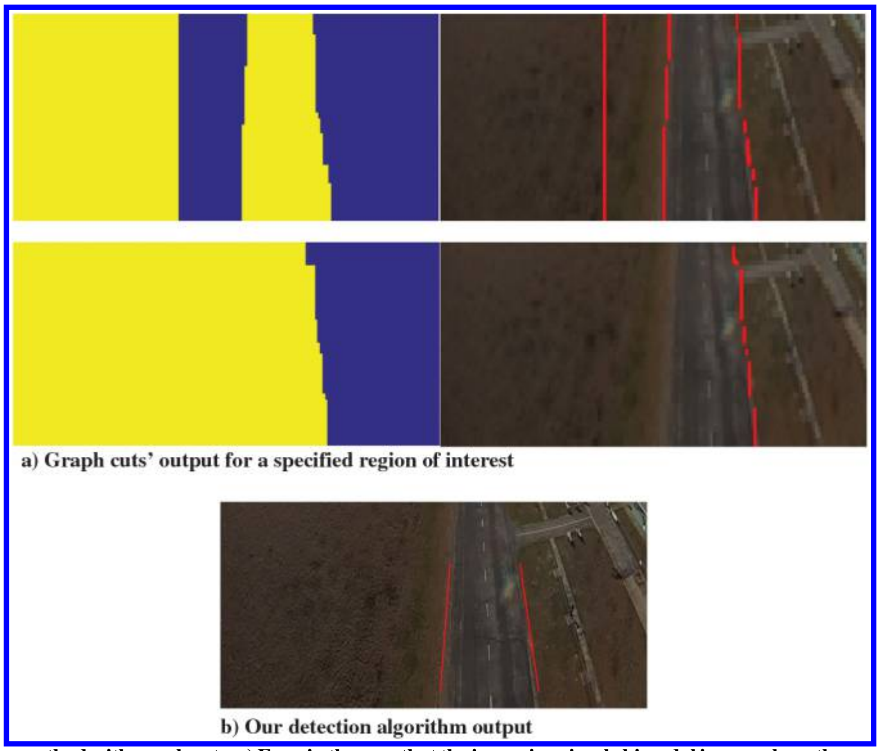

a) Graph cuts' output for a specified region of interest
b) Our detection algorithm output

Fig. 17 Comparison of our method with graph cuts. a) Even in the case that the image is a simple bimodal image where the correct number of regions is 2, and the graph cut algorithm is given the correct number, the results depend on the location of the seeds. The top and bottom show the results with two different initial seeds. b) Our detection and segmentation algorithms are able to achieve the correct result, because the detection process is a robust global search, which gives good initialization to the gradient descent for the segmentation scheme, which is also robust to local minima because the optimization is over a space that is very low dimensional.

regions (or seeds) to choose. Choosing a small number of regions leads to an undersegmentation, and a large number leads to an oversegmentation of the runway. In general, the number will vary with the scenery present, making it nearly impossible to choose the "correct number" a priori. Moreover, there is no guarantee that the runway will appear as a segment for any number of regions chosen. Even if it does, one would still need to ascertain which of the segments is the runway, requiring further postprocessing. These problems are the motivation for our model-based detection and segmentation scheme that essentially fits a simple geometric model of the runway to the image(s). As Figs. 16b and 17b show, our method obtains the correct result automatically.

In the upper row of Fig. 17a, the region of interest is specified for graph cut by the user. Two regions output from graph cut, and still the runway is not detected. In the lower row of Fig. 17a, there is same input as before, but with different output, due to the randomized initialization inherited in the graph cut algorithm.

VII. Conclusions

This paper introduces a robust runway detection and tracking algorithm that can be used for real-time UAV control in landing conditions. Unlike recent approaches to runway detection using computer vision, our approach requires no rendered computer model of the shape and appearance of the runway, making it very robust to different types of runways and lighting conditions. Generating rendered computer models under all possible illumination/weather conditions, runway geometries, and runway materials (grass, concrete, asphalt, dirt) is not scalable, especially because small UAVs can operate from a large number of small airstrips or private roads. Our approach uses a simple model of the *local* geometry of the runway and makes no assumption on the appearance of the runway other than the dissimilarity of the runway with respect to its surrounding and the assumption that the runway is straight. We show that a combination of runway position estimation (through the minimization of an energy function derived from the runway model) and Kalman filtering using inertial or visual attitude information allows us to efficiently and robustly track the runway in real time. The performance of the runway detection algorithm is sufficiently good to use it directly (without any inertial measurements) for most runway landing applications.

Appendix A: Computation of the Gradient of E

In this section, we show the details for the computation of the gradient of E with respect to $\boldsymbol{d} = (d_1, d_2, d_3, d_4)$. Using the chain rule, we see that

$$\nabla E(\boldsymbol{d}) = -(u(\boldsymbol{d}) - v(\boldsymbol{d}))(\nabla u(\boldsymbol{d}) - \nabla v(\boldsymbol{d})) \tag{A1}$$

where ∇ denotes the gradient with respect to $\boldsymbol{d}$.

It remains to compute the gradient of u and v with respect to $\boldsymbol{d}$, which are defined by

$$u(\boldsymbol{d}) = \frac{\int_R I(x)\,\mathrm{d}A}{\int_R \mathrm{d}A}, \qquad v(\boldsymbol{d}) = \frac{\int_{R^c} I(x)\,\mathrm{d}A}{\int_{R^c} \mathrm{d}A} \tag{A2}$$

in the non-spatially varying case. R is the banded region inside the trapezoid determined by $\boldsymbol{d}$, R^c is the banded region outside the trapezoid determined by $\boldsymbol{d}$, and dA is the area differential element. To compute the gradient, we may use the result [36], which states that the directional derivative of a functional

$$e(R) = \int_R f(x)\,\mathrm{d}A$$

is given by

$$de(R) \cdot h = \int_{\partial R} f(x)\,\mathrm{d}s$$

where h is a perturbation of the boundary of R, ∂R is the boundary of R that is allowed to vary, ds is the arclength differential element, and

$de(R)\cdot h$ is the change in e by perturbing R by a perturbation h (defined on the boundary of ∂R).

By applying the quotient rule and the previous result, one can show that

$$\begin{aligned}\frac{\partial u}{\partial d_j} &= \frac{1}{|R|}\int_{C_{L_i}} (I-u)h_j\cdot N\,\mathrm{d}s\\ \frac{\partial v}{\partial d_j} &= -\frac{1}{|R^c|}\int_{C_{L_i}} (I-v)h_j\cdot N\,\mathrm{d}s\end{aligned} \tag{A3}$$

where $C_L{:}[0,1]\to R^2$ is the line segment corresponding to either side (right or left) of the trapezoid, N is the unit outward normal to C_L, and h_j corresponds to the perturbation of C_{L_j} when the jth vertex is perturbed.

$i=1,2$ and $i=3,4$ correspond to line segments at the left side and the right side, respectively. While $j=1,2,3,4$ correspond to the four variables that should be updated. Note that

$$C_{L_i}(t)=\begin{cases}\begin{pmatrix}d_1\\0\end{pmatrix}+\begin{pmatrix}d_2-d_1\\L_2\end{pmatrix}t & \text{if } i=1,2\\ \begin{pmatrix}L_1-d_3\\0\end{pmatrix}+\begin{pmatrix}d_3-d_4\\L_2\end{pmatrix}t & \text{if } i=3,4\end{cases} \tag{A4}$$

and

$$h_j=\frac{\partial C(t)}{\partial d_j}=\begin{cases}\begin{pmatrix}1-t\\0\end{pmatrix} & \text{if } j=1,3\\ \begin{pmatrix}t\\0\end{pmatrix} & \text{if } j=1,4\end{cases} \tag{A5}$$

Because N is the outward normal to C_{L_j}, it can be related to the tangent T to C_{L_j} by $N=JT$, where J

$$J=\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix} \tag{A6}$$

is the rotation matrix with $\theta=-90^\circ$ counter clockwise, if the trapezoid is parameterized clockwise. Note also that

$$T(t)=\frac{C'_{L_j}(t)}{|C'_{L_j}(t)|} \tag{A7}$$

where $|C'_{L_j}(t)|$ is the speed of the point, s denotes the arc length parameter,

$$ds=|C'_{L_j}(t)|dt \tag{A8}$$

and L_j will denote the length of C_{L_j}. Combining Eqs. (A4), (A5), (A7), and (A8) results in

$$N(t)\cdot h(t)ds=L_2(t-1)dt \tag{A9}$$

for $i=1,2$ and $j=3$ (the other cases yield similar expressions). Then, the gradient direction at vertex $j=3$, for example, can be derived as follows:

$$\begin{aligned}\frac{\partial E}{\partial d_3} = -L_2(u-v)\Bigg[&\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_2}(t))-u)\,\mathrm{d}t\\ &+\frac{1}{|R^c|}\int_0^1 (t-1)I((C_{L_2}(t))-v)\,\mathrm{d}t\Bigg]\end{aligned} \tag{A10}$$

The same argument applies to the other d_i, and thus

$$\begin{aligned}\frac{\partial E}{\partial d_1} = L_2(v-u)\Bigg[&\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_1}(t))-u)\,\mathrm{d}t\\ &+\frac{1}{|R^c|}\int_0^1 (t-1)(I(C_{L_1}(t))-v)\,\mathrm{d}t\Bigg]\end{aligned} \tag{A11}$$

$$\begin{aligned}\frac{\partial E}{\partial d_2} = L_2(v-u)\Bigg[&\frac{1}{|R|}\int_0^1 t(I(C_{L_1}(t))-u)\,\mathrm{d}t\\ &+\frac{1}{|R^c|}\int_0^1 t(I(C_{L_1}(t))-v)\,\mathrm{d}t\Bigg]\end{aligned} \tag{A12}$$

$$\begin{aligned}\frac{\partial E}{\partial d_3} = L_2(v-u)\Bigg[&\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_2}(t))-u)\,\mathrm{d}t\\ &+\frac{1}{|R^c|}\int_0^1 (t-1)I((C_{L_2}(t))-v)\,\mathrm{d}t\Bigg]\end{aligned} \tag{A13}$$

$$\begin{aligned}\frac{\partial E}{\partial d_4} = L_2(v-u)\Bigg[&\frac{1}{|R|}\int_0^1 t(I(C_{L_2}(t))-u)\,\mathrm{d}t\\ &+\frac{1}{|R^c|}\int_0^1 t(I(C_{L_2}(t))-v)\,\mathrm{d}t\Bigg]\end{aligned} \tag{A14}$$

This derivation applies for the special case where u and v are nonspatially variant. Generalizing the previous expressions for spatially variant means would require to calculate the mean in a neighborhood, at each point on the line segment, represented by a line that lies in the normal direction. Hence,

$$\begin{aligned}u_{L_i}(t) &= \frac{1}{\ell(t)}\int_0^{\ell(t)} I(C_{L_i}(t)-s\cdot N)\,\mathrm{d}s,\\ v_{L_i}(t) &= \frac{1}{\ell(t)}\int_0^{\ell(t)} I(C_{L_i}(t)+s\cdot N)\,\mathrm{d}s\end{aligned} \tag{A15}$$

where ds is the arclength differential element of the segment defined on $[C_{L_i}(t), C_{L_i}(t)-\ell(t)N]$, $[C_{L_i}(t), C_{L_i}(t)+\ell(t)N]$ for $u_{L_i}(t)$, $v_{L_i}(t)$, respectively. The gradient expression is identical though, with u and v substituted with $u_{L_i}(t)$ and $v_{L_i}(t)$, respectively.

Appendix B: Computation of the Measurement Matrix C_k

We derive the mapping between the parameters of a line in the ground plane of the world coordinate system and the parameters of the same line in the image plane.

A line given in the imaging plane, a 2D coordinate system, can be written as

$$a'x'+b'y'+c'=0 \tag{B1}$$

where the $'$ indicates quantities in the imaging plane. The same line in the 3D coordinate system of the world ground frame is

$$\left\{\begin{pmatrix}x\\y\\1\end{pmatrix}\cdot\begin{pmatrix}a\\b\\c\end{pmatrix}=0 \atop z=0\right\} \tag{B2}$$

Note that we assume that $z=0$ is the ground, and that the runway is on the ground.

Any point $p = (x, y, z)^T$ in 3D in the world frame corresponds to a point $p = (x', y')^T$ in 2D in the imaging plane. As noted previously, the transformation is given by

$$\begin{pmatrix} x_{\text{cam}} \\ y_{\text{cam}} \\ z_{\text{cam}} \end{pmatrix} = \kappa \left[R \begin{pmatrix} x \\ y \\ z \end{pmatrix} + T \right] \tag{B3}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x_{\text{cam}}/z_{\text{cam}} \\ y_{\text{cam}}/z_{\text{cam}} \end{pmatrix} \tag{B4}$$

where κ is the intrinsic calibration matrix, and (R, t) describe the rotation and translation between ground and camera frame.

We write Eq. (B2) in vector form as

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \boldsymbol{a}_{\text{image}} \cdot \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = 0 \tag{B5}$$

Substituting Eq. (B4) into the previous equation, we get

$$\boldsymbol{a}_{\text{image}} \cdot \begin{pmatrix} x_{\text{cam}}/z_{\text{cam}} \\ y_{\text{cam}}/z_{\text{cam}} \\ 1 \end{pmatrix} = 0 \tag{B6}$$

and multiplying both sides of the equation by z_{cam}, we arrive at

$$\boldsymbol{a}_{\text{image}} \cdot \begin{pmatrix} x_{\text{cam}} \\ y_{\text{cam}} \\ z_{\text{cam}} \end{pmatrix} = 0 \tag{B7}$$

By substituting Eq. (B3) into Eq. (B7), we arrive at

$$\boldsymbol{a}^T_{\text{image}} \left[\kappa \left(R \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + t \right) \right] = 0 \tag{B8}$$

$$\left(\boldsymbol{a}^T_{\text{image}} \kappa R \right) \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \left(\boldsymbol{a}^T_{\text{image}} \kappa t \right) = 0 \tag{B9}$$

We can rewrite the previous expression as

$$\left[\boldsymbol{a}^T_{\text{image}} \kappa R \text{id}_{3\times 2} \quad \boldsymbol{a}^T_{\text{image}} \kappa t \right] \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0 \tag{B10}$$

where $\text{id}_{3\times 2}$ is the 3×2 matrix containing as columns unit vectors, $(1, 0, 0)^T$ and $(0, 1, 0)^T$. Note that the previous equation is now in the form of the line in 3D world coordinates, as in Eq. (B2). Therefore, we can take the left most factor to be the parameters of the line in the world coordinate system (this is unique up to a scale factor, and presently the scale factor is chosen to satisfy $a^2 + b^2 = 1$, as noted earlier):

$$\boldsymbol{a}^T_{\text{world}} = \left[\boldsymbol{a}^T_{\text{image}} \kappa R \text{id}_{3\times 2} \quad \boldsymbol{a}^T_{\text{image}} \kappa t \right] \tag{B11}$$

$$= \left[\boldsymbol{a}^T_{\text{image}} \kappa R \text{id}_{3\times 2} \quad \boldsymbol{a}^T_{\text{image}} \kappa t \right] \tag{B12}$$

$$= \boldsymbol{a}^T_{\text{image}} \left[\kappa R \text{id}_{3\times 2} \quad \kappa t \right] \tag{B13}$$

Transposing both sides of the equation above and then solving for $\boldsymbol{a}_{\text{image}}$, one obtains

$$\begin{pmatrix} a\prime \\ b\prime \\ c\prime \end{pmatrix} = E \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{B14}$$

where

$$E = \left[\kappa R \text{id}_{3\times 2} \quad \kappa t \right]^{-T} = \begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix}^{-T} \tag{B15}$$

and

$$e_1 = \begin{pmatrix} k_1 \cdot r_1 \\ k_2 \cdot r_1 \\ k_3 \cdot r_1 \end{pmatrix}, \quad e_2 = \begin{pmatrix} k_1 \cdot r_2 \\ k_2 \cdot r_2 \\ k_3 \cdot r_2 \end{pmatrix}, \quad e_3 = \begin{pmatrix} k_1 \cdot t \\ k_2 \cdot t \\ k_3 \cdot t \end{pmatrix} \tag{B16}$$

This same transformation can be also used to relate both parallel lines of the runway in the image; therefore, we get

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = E \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \quad \begin{pmatrix} d' \\ e' \\ f' \end{pmatrix} = E \begin{pmatrix} a \\ b \\ d \end{pmatrix} \tag{B17}$$

where d', e', f' are the parameters of the second line of the runway in the image plane. By putting both of those equations into a single equation, we can get our measurement model:

$$\begin{pmatrix} a' \\ b' \\ c' \\ d' \\ e' \\ f' \end{pmatrix} = C_k \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \tag{B18}$$

where C_t is a 6×4 matrix given by

$$C_t = \begin{pmatrix} e_1 & e_2 & e_3 & 0_{(3\times 1)} \\ e_1 & e_2 & 0_{(3\times 1)} & e_3 \end{pmatrix} \tag{B19}$$

This equation relates the parameters of the parallel lines in the world frame to the parameters of the two lines in the imaging plane (not necessarily parallel).

Appendix C: Runway Location in the Earth Frame Given Its Image Location

In a classical camera pinhole model, the alignment of the axes is described by the following. The camera Z axis is the depth axis, where the image is formed in the XY plane, and hence the depth component is the normalization factor to reach homogeneous coordinates.

Figure 6 shows relation a schematic illustrating the geometry of the image formation, and it relates a point in the camera coordinate system to the corresponding point in the world coordinate system. Given a 2D point in the image and the rigid body motion between the camera frame and the world frame, the 3D point located at the ground ($z = 0$ in the camera frame) can be obtained by solving a linear system. Starting with camera pinhole model, we have

$$\kappa\left[R\begin{pmatrix}x\\y\\z\end{pmatrix}+t\right]=\begin{pmatrix}x_{\text{cam}}\\y_{\text{cam}}\\z_{\text{cam}}\end{pmatrix}\tag{C1}$$

Hence, the normalized 2D point can be written as

$$\begin{pmatrix}x_{\text{cam}}\\y_{\text{cam}}\end{pmatrix}=z_{\text{cam}}\begin{pmatrix}x'\\y'\end{pmatrix}\tag{C2}$$

Now, the system can be decomposed into two parts:

$$\text{id}^T_{3\times2}\kappa R\begin{pmatrix}x\\y\\0\end{pmatrix}+\text{id}^T_{3\times2}\kappa t=z_{\text{cam}}\begin{pmatrix}x'\\y'\end{pmatrix}\tag{C3}$$

and

$$z_{\text{cam}}=\text{id}^T_{3\times1}\kappa R\begin{pmatrix}x\\y\\0\end{pmatrix}+\text{id}^T_{3\times1}\kappa t\tag{C4}$$

where $\text{id}_{3\times2}$ is the 3×2 matrix containing as columns unit vectors, $(1,0,0)^T$ and $(0,1,0)^T$, and $\text{id}_{3\times1}$ is the 3×2 matrix containing as column unit vector, $(0,0,1)^T$.

Combining terms and rearranging:

$$\text{id}^T_{3\times2}\kappa R\begin{pmatrix}x\\y\\0\end{pmatrix}-\begin{pmatrix}(\text{id}^T_{3\times1}\kappa R)\cdot x'\\(\text{id}^T_{3\times1}\kappa R)\cdot y'\end{pmatrix}\begin{pmatrix}x\\y\\0\end{pmatrix}=-\text{id}^T_{3\times2}\kappa t+\text{id}^T_{3\times1}\kappa t\begin{pmatrix}x'\\y'\end{pmatrix}\tag{C5}$$

Hence, solving the system $MP=b$ would give the 3D point, where M and b are given by

$$M=\text{id}_{2\times2}\kappa R-\begin{pmatrix}x'\\y'\end{pmatrix}\text{id}^T_{2\times1}\kappa R\tag{C6}$$

$$b=-\text{id}_{3\times2}\kappa t+\text{id}^T_{3\times1}\kappa t\begin{pmatrix}x'\\y'\end{pmatrix}\tag{C7}$$

Acknowledgment

We gratefully acknowledge the Aerospace Division of King Abdulaziz City for Science and Technology (KACST) for supporting us with experimental flight videos generated by the SAKER 4 UAV.

References

[1] Kontitsis, M., Valavanis, K. P., and Tsourveloudis, N., "A UAV Vision System for Airborne Surveillance," *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04*, Vol. 1, IEEE Publ., Piscataway, NJ, 2004, pp. 77–83.
doi:10.1109/ROBOT.2004.1307132

[2] Medioni, G., Cohen, I., Brémond, F., Hongeng, S., and Nevatia, R., *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 8, 2001, pp. 873–889.
doi:10.1109/34.946990

[3] Todorovic, S., and Nechyba, M. C., "Intelligent Missions for MAVs: Visual Contexts for Control, Tracking and Recognition," *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04*, Vol. 2, IEEE Publ., Piscataway, NJ, 2004, pp. 1640–1645.
doi:10.1109/ROBOT.2004.1308059

[4] Lai, A. H. S., and Yung, N. H. C., "Lane Detection by Orientation and Length Discrimination," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 30, No. 4, 2000, pp. 539–548.
doi:10.1109/3477.865171

[5] Tang, Y.-L., and Kasturi, R., "Runway Detection in an Image Sequence," *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, International Soc. for Optics and Photonics, San Jose, CA, 1995, pp. 181–190.
doi:10.1117/12.205479

[6] Dusha, D., Boles, W. W., and Walker, R., "Fixed-Wing Attitude Estimation Using Computer Vision Based Horizon Detection," *Proceedings of 12th Australian International Aerospace Congress*, April 2007.

[7] Meng, D., Yun-feng, C., and Lin, G., "A Method to Recognize and Track Runway in the Image Sequences Based on Template Matching," *1st International Symposium on Systems and Control in Aerospace and Astronautics, 2006. ISSCAA 2006*, IEEE Publ., Piscataway, NJ, 2006, p. 4.
doi:10.1109/ISSCAA.2006.1627585

[8] Gui, Y., Guo, P., Zhang, H., Lei, Z., Zhou, X., Du, J., and Yu, Q., "Airborne Vision-Based Navigation Method for UAV Accuracy Landing Using Infrared Lamps," *Journal of Intelligent & Robotic Systems*, Vol. 72, No. 2, 2013, pp. 197–218.
doi:10.1007/s10846-013-9819-5

[9] Phueakjeen, W., Jindapetch, N., Kuburat, L., and Suvanvorn, N., "A Study of the Edge Detection for Road Lane," *2011 8th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, IEEE Publ., Piscataway, NJ, 2011, pp. 995–998.
doi:10.1109/ECTICON.2011.5948010

[10] Wang, Y., Teoh, E. K., and Shen, D., "Lane Detection and Tracking Using B-Snake," *Image and Vision Computing*, Vol. 22, No. 4, 2004, pp. 269–280.
doi:10.1016/j.imavis.2003.10.003

[11] Liu, W., and Li, S., "An Effective Lane Detection Algorithm for Structured Road in Urban," *Intelligent Science and Intelligent Data Engineering*, Springer, Berlin, 2013, pp. 759–767.
doi:10.1007/978-3-642-36669-7

[12] Saripalli, S., Montgomery, J. F., and Sukhatme, G., "Vision-Based Autonomous Landing of an Unmanned Aerial Vehicle," *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA'02*, Vol. 3, IEEE Publ., Piscataway, NJ, 2002, pp. 2799–2804.
doi:10.1109/ROBOT.2002.1013656

[13] Lim, K. H., Seng, K. P., Ang, L.-M., and Chin, S. W., "Lane Detection and Kalman-Based Linear-Parabolic Lane Tracking," *International Conference on Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC'09*, Vol. 2, IEEE Publ., Piscataway, NJ, 2009, pp. 351–354.
doi:10.1109/IHMSC.2009.211

[14] Goldstein, T., Bresson, X., and Osher, S., "Geometric Applications of the Split Bregman Method: Segmentation and Surface Reconstruction," *Journal of Scientific Computing*, Vol. 45, Nos. 1–3, 2010, pp. 272–293.
doi:10.1007/s10915-009-9331-z

[15] Osher, S., Burger, M., Goldfarb, D., Xu, J., and Yin, W., "An Iterative Regularization Method for Total Variation-Based Image Restoration," *Multiscale Modeling & Simulation*, Vol. 4, No. 2, 2005, pp. 460–489.
doi:10.1137/040605412

[16] Laiacker, M., Kondak, K., Schwarzbach, M., and Muskardin, T., "Vision Aided Automatic Landing System for Fixed Wing UAV," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE Publ., Piscataway, NJ, 2013, pp. 2971–2976.
doi:10.1109/IROS.2013.6696777

[17] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J., "Contour Detection and Hierarchical Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 5, 2011, pp. 898–916.
doi:10.1109/TPAMI.2010.161

[18] Sofou, A., Evangelopoulos, G., and Maragos, P., "Coupled Geometric and Texture PDE-Based Segmentation," *IEEE International Conference on Image Processing, 2005. ICIP 2005*, Vol. 2, IEEE Publ., Piscataway, NJ, 2005, p. II-650.
doi:10.1109/ICIP.2005.1530139

[19] Kornprobst, P., Deriche, R., and Aubert, G., "Image Sequence Restoration: A PDE Based Coupled Method for Image Restoration and Motion Segmentation," *European Conference on Computer Vision*, Springer, Berlin, 1998, pp.-548–562.
doi:10.1007/BFb0054764

[20] Sofou, A., and Maragos, P., "Generalized Flooding and Multicue PDE-Based Image Segmentation," *IEEE Transactions on Image Processing*, Vol. 17, No. 3, 2008, pp. 364–376.
doi:10.1109/TIP.2007.916156

[21] Yezzi, A., Tsai, A., and Willsky, A., "A Fully Global Approach to Image Segmentation via Coupled Curve Evolution Equations," *Journal of Visual Communication and Image Representation*, Vol. 13, No. 1, 2002, pp. 195–216.
doi:10.1006/jvci.2001.0500

[22] Boyd, S., and Vandenberghe, L., *Convex Optimization*, Cambridge Univ. Press, Cambridge, England, U.K., 2004, pp. 466–474.

[23] Kennedy, J., "Particle Swarm Optimization," *Encyclopedia of Machine Learning*, Springer, Berlin, 2010, pp. 760–766.
doi:10.1007/978-0-387-30164-8_630

[24] Unal, G., Yezzi, A., and Krim, H., "Information-Theoretic Active Polygons for Unsupervised Texture Segmentation," *International Journal of Computer Vision*, Vol. 62, No. 3, 2005, pp. 199–220.
doi:10.1007/s11263-005-4880-6

[25] Abu-Jbara, K., Alheadary, W., Sundaramorthi, G., and Claudel, C., "A Robust Vision-Based Runway Detection and Tracking Algorithm for Automatic UAV Landing," *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE Publ., Piscataway, NJ, 2015, pp. 1148–1157.
doi:10.1109/ICUAS.2015.7152407

[26] Zanetti, R., Majji, M., Bishop, R. H., and Mortari, D., "Norm-Constrained Kalman Filtering," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 5, 2009, pp. 1458–1465.
doi:10.2514/1.43119

[27] Ma, Y., Soatto, S., Kosecka, J., and Sastry, S. S., *An Invitation to 3-d Vision: From Images to Geometric Models*, Vol. 26, Springer Science & Business Media, New York, 2012, pp. 109–170.

[28] Faugeras, O. D., Luong, Q.-T., and Maybank, S. J., "Camera Self-Calibration: Theory and Experiments," *Computer Vision—ECCV'92*, Springer, Berlin, 1992, pp. 321–334.
doi:10.1007/3-540-55426-2_37

[29] Bouguet, J.-Y., "Camera Calibration Toolbox for Matlab," 2004, http://www.vision.caltech.edu/bouguetj/calib_doc/.

[30] Vedaldi, A., and Fulkerson, B., "Vlfeat: An Open and Portable Library of Computer Vision Algorithms," *Proceedings of the 18th ACM International Conference on Multimedia, MM '10*, ACM, New York, 2010, pp. 1469–1472.
doi:10.1145/1873951.1874249

[31] Lowe, D. G., "Object Recognition from Local Scale-Invariant Features," *Proceedings of the 7th IEEE International Conference on Computer Vision, 1999*, IEEE Publ., Piscataway, NJ, Vol. 2, 1999, pp. 1150–1157.

[32] Bay, H., Tuytelaars, T., and Van Gool, L., "Surf: Speeded Up Robust Features," *Computer Vision—ECCV 2006*, Springer, Berlin, 2006, pp. 404–417.

[33] Rosten, E., and Drummond, T., "Machine Learning for High-Speed Corner Detection," *Computer Vision—ECCV 2006*, Springer, Berlin, 2006, pp. 430–443.

[34] Abu-Jbara, K., "Runway Detection and Trackingin 3d Algorithm's Videos," Feb. 2017, https://sites.google.com/a/kaust.edu.sa/projects-kf/3Dtracking-runway-autolanding [accessed 2 Feb. 2017].

[35] Irani, M., "Matlab Wrapper for Graph Cuts," Dec. 2006, http://www.wisdom.weizmann.ac.il/~bagon/matlab.html.

[36] Zhu, S. C., and Yuille, A., "Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 9, 1996, pp. 884–900.
doi:10.1109/34.537343