A Robust Vision-based Runway Detection and Tracking Algorithm for Automatic UAV Landing

Khaled Abu Jbara, Wael Alheadary, Ganesh Sundaramorthi, and Christian Claudel
Department of Electrical Engineering, KAUST, Saudi Arabia

***Abstract*—This work presents a novel real-time algorithm for runway detection and tracking applied to the automatic takeoff and landing of Unmanned Aerial Vehicles (UAVs).**

The algorithm is based on a combination of segmentation based region competition and the minimization of a specific energy function to detect and identify the runway edges from streaming video data. The resulting video-based runway position estimates are updated using a Kalman Filter, which can integrate other sensory information such as position and attitude angle estimates to allow a more robust tracking of the runway under turbulence.

We illustrate the performance of the proposed lane detection and tracking scheme on various experimental UAV flights conducted by the Saudi Aerospace Research Center (KACST). Results show an accurate tracking of the runway edges during the landing phase, under various lighting conditions, and suggest that such positional estimates would greatly improve the positional accuracy of the UAV during takeoff and landing phases. The robustness of the proposed algorithm is further validated using Hardware in the Loop (HIL) simulations with diverse takeoff and landing videos generated using a commercial flight simulator.

I. Introduction

Unmanned Aerial Vehicles (UAVs) have recently become more and more prevalent, enabling tasks that are too dirty, too dull or too dangerous to be undertaken by manned flying vehicles. UAVs have for instance been used in surveillance applications including fire detection [7], event detection [13] or object tracking [21]. They also play an increasing role in military operations [9]. one of the main practical challenges associated with the operation of fixed wing UAVs is to guarantee a safe takeoff and landing [20], particularly since UAVs are more sensitive to turbulence than heavier, manned aircrafts. Automatic takeoff and landing is typically done using navigation sensors, such as absolute positioning systems (e.g. GPS), accelerometers, gyrometers and magnetometers. While the fusion between GPS measurements and inertial measurement unit-based measurements greatly improve positional accuracy, the residual positional errors are too large to allow reliable UAV landings on a practical UAV airstrip. Positioning accuracy on practical UAV airstrips could be improved by using enhanced GPS systems systems (such as differential GPS), though such systems require additional equipment (reference stations), which is expensive. In contrast, the ever decreasing cost and ever increasing performance of cameras make them particularly suitable to this application. Since 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. In order to correct for the GPS positioning errors, we propose a new algorithm leveraging image processing and control systems theory to detect the runway edges, and use this information to correct the longitudinal and lateral position of the UAV during the landing phase. The main objective of the proposed algorithm is to allow a very robust (under all lighting conditions or runway configurations) detection and tracking of the runway edges, to guarantee an accurate landing on narrow airstrips. Numerous computer vision algorithms have been proposed to help the navigation of fixed wing and rotary UAVs. In [5], the authors use morphological image processing and Hough Transforms (HTs) to identify the horizon and estimate the attitude of a UAV. The authors of [14] propose a template matching algorithm to detect and track the position of a runway. Active methods can also be thought of, for instance in the article [6], 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 [15] where vehicles are detected using HT and Robert filters. The authors of [23], use splines curve fitting, HT, canny edge detector, and vanishing points to estimate the boundary of the road. While effective, this method is complex and does not run in real time on commercially available embedded platforms. In article [12] 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 [9] 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 [17], in which the authors use Random Sample Consensus (RANSAC) and cubic spline curve fitting to extract the lane position from a video stream. A linear parabolic lane tracking system with Kalman filter is proposed in [11]. On the other hand, many approaches [2], [3], [16] 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 variety of lighting conditions and a variety of runway configurations is currently available. Currently available methods, such as [10]) 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 [23]) have to be excluded from the tracking problem. Our objective is to develop such a robust scheme that uses a

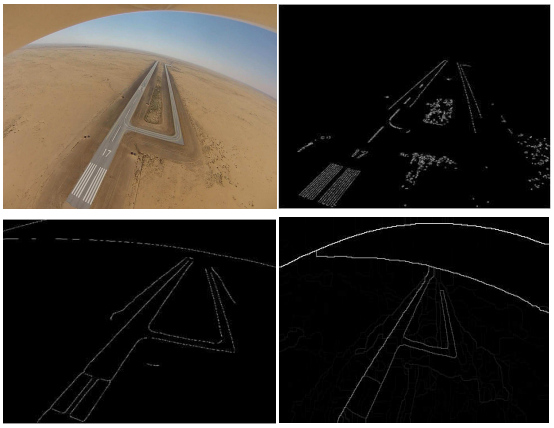

Fig. 1: **Canny edge detection results on experimental landing video image.** The upper left subfigure shows the actual video data, while the upper right, lower left and lower right subfigures correspond to the outputs of the Canny Edge detector applied to this image, using the parameters σ=1 (upper right), σ=10 (lower left) and σ=20 (lower right), in which σ denotes to the standard deviation of the Gaussian filter . All these results where fixed to a threshold [0.1-0.4]. As can one see from these pictures, a small variation of parameters dramatically affects the features that can be extracted from the image. Recent edge detectors [1] exhibit similar behavior.

minimal number of assumptions on the shape of the runway, and robust video detection and tracking methods coupled with a Kalman Filter which can fuse additional information (such as gyrometer data) in its prediction step. We illustrate this sensitivity to model parameters on Figure 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 detector (applied to a UAV landing video) due to minor changes in its parameters.

The rest of the article is organized as follows. Section II gives a high level overview of the proposed runway detection and tracking algorithm. The core vision-based feature extraction algorithm is described in section III, and formulated as an optimization problem. Section IV introduces the Kalman Filter formulation of the runway edge detection and tracking algorithm. We validate in section V the performance of this algorithm using experimental landing videos obtained by the Saudi Department of Aerospace. To further illustrate the robustness of the proposed scheme, we develop an hardware-in-the-loop (HIL) framework around a commercial flight simulator and the proposed runway tracking algorithm in section VI. In this section, we show that runway detection and tracking is achieved in a very wide variety of runway configurations, lighting conditions and obstacles, without any modification of the parameters of the algorithm.

II. Runway detection and tracking algorithm overview

We now present the high level overview of the runway edge detection and tracking algorithm. This algorithm is based on the combination of computer vision and classical Kalman Filtering, which is illustrated in Figure 2 below.

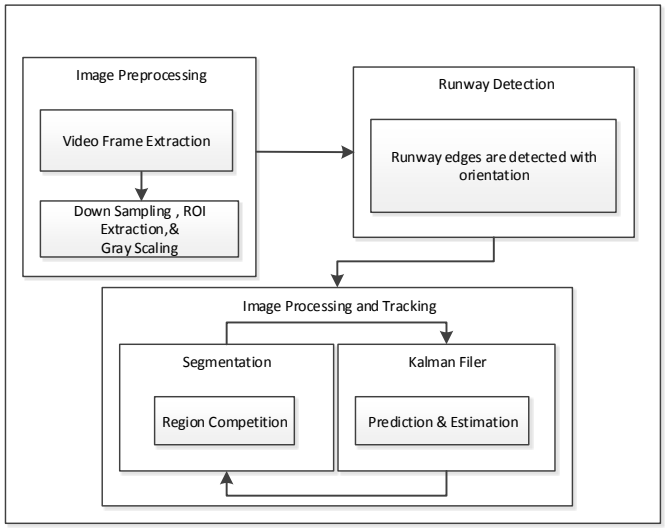

Fig. 2: **Computer vision algorithm diagram.**

The streaming video data is first converted into a sequence of frames, which are downsampled (both in the color space and spatially). The regions of interest (ROI) of the image (containing the runway) is detected at this step. Once the ROI is identified, the edges of the runway are detected through computer vision techniques detailed in section III. The edge location estimates are then fed to a Kalman Filter, which can fuse additional data (if available) to estimate the runway location. Then runway edges are then displayed on the real-time video, or can be used to feed the UAV autopilot (the location of the runway with respect to the UAV can be computed modulo a conversion of the video image coordinates to absolute coordinates).

We now describe the core image processing algorithm.

III. Runway detection and tracking algorithm structure

In this section, we formulate a model for aerial images containing a runway. We then use the model to formulate an optimization problem to detect and segment the runway given an image. We then derive optimization methods for the energy. 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 on off-the-shelf UAVs. Further, the algorithms are designed to be robust to background clutter, illumination conditions (e.g., day and night), and noise - all important visual nuisances that are prevalent in aerial imagery.

A. Model and Optimization Problem

Our model for a runway is 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 non-parallel in the

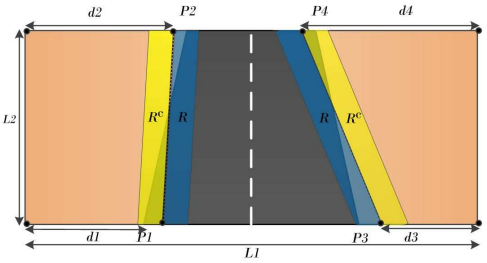

Fig. 3: **Runway Model**. The runway edges are straight line segments such that the corresponding regions R and R_c have maximally different image statistics. The other labeled quantities are lengths used through the manuscript.

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 certain region below the aircraft, where the two lane segment assumption is largely true. The advantage of such a model is that the model of the runway is finite dimensional and leads to fast algorithms. Our model of the image is that it consists of two segments (the runway), and that for each line segment, a neighborhood region on each side of the segment consists of *maximally* different image intensity statistics. See Figure 3. We make no assumptions on the rest of the image (e.g., the background and the interior of the runway) as the images may vary significantly in this region depending on the environment.

For simplicity, we assume that the aircraft is approximately aligned to the runway, which is true in our application since the navigation system based on GPS is able to position the aircraft within a few meters (typically less than 15 meters), and an heading uncertainty of a few degrees (typically less than 10 degrees). In this case, within a rectangle around the runway, we may assume that each line segment (representing the runway) intersects the top and bottom of the rectangle. The rectangle corresponds to the approximate localization of the runway within the image. Therefore, the line segments may be specified by four distances d_i, $i = 1, \ldots, 4$ between the corners of the rectangle and the endpoints of the line segments. See Figure 3.

We now formulate an optimization problem defined on line segments (specified by d_i) such that the minimizer corresponds to the line segments that are aligned with the edges of the runway. 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 is the mean RGB values with the neighborhoods, but any other statistics may be used (e.g., intensity histograms, statistics of filter bank responses, etc). Let the image be denoted by $I : \Omega \to \mathbb{R}^k$ ($k = 3$ for RGB) where $\Omega \subset \mathbb{R}^2$, and let $R, R^c \subset \Omega$ denote the inside and outside neighborhood regions of the line segments, respectively. See Figure 3. The mean statistics $u, v \in \mathbb{R}^k$ for the inside and outside neighborhoods are then

$$u = \frac{1}{|R|}\int_R I(x)dA, \quad v = \frac{1}{|R^c|}\int_{R^c} I(x)dA, \tag{1}$$

where $|R|$ denotes the area of R, and dA is the area element. Since we would like the line segments positioned so that u and v are maximally different, we define the following energy to be minimized:

$$E(d_1, d_2, d_3, d_4) = -\frac{1}{2}|u - v|^2. \tag{2}$$

Note that u and v are functions of the position of the line segments since R and R^c depend on the line segments, which are in turn specified by d_i. Therefore, the energy E is a function of d_i.

As we will see, 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 [8], [18], [19] in which the energies are defined on the set of (infinite dimensional) regions. The energy is most closely related to [24], where the objective is to divide the image into two disjoint regions that have maximally different intensity means. Our energy formulation has two major modifications with respect to previously used formulations: first, the energy is defined on a simplified representation of regions defined by line segments to support real-time applications needed for the UAV and second, the use of neighborhoods rather than the entire image to avoid background clutter typical in aerial imagery. One problem with the methods of image segmentation using partial differential equations is they typically rely on local descent methods because the energies are non-convex (as is our energy of interest). Therefore, they require the user to design an initialization algorithm. Our model of the runway is designed to achieve a computationally tractable real-time detection algorithm that localizes the runway. This supports a *fully automatic real-time* runway detection and segmentation algorithm, a key advance with respect to existing image segmentation methods, and the key computer vision innovation in our method.

In the next subsections, we describe an automatic detection algorithm that roughly localizes the runway by approximating an optimizer of E, and then we present an optimization scheme that refines the detection to achieve a more accurate localization of the runway.

B. Detection Method

The optimization of E directly by evaluating E for every d_i (or a sampling of the space) would be computationally prohibitive, certainly prohibitive for real-time applications. Moreover, without an initialization near the desired runway, local optimization techniques, which are the only candidates for optimization of E due to non-convexity, would fail to capture the runway. Therefore, in this section, we construct a coarse-to-fine algorithm that searches efficiently over the space of d_i. The algorithm prunes away irrelevant line segments in the search space of d_i and proposes a few candidate segments as segments corresponding to the runway. The energy of these few candidates can then be evaluated, and the energetically most favorable candidate can be selected. This leads to a real-time fully automatic algorithm that is also robust to clutter.

First, to efficiently propose candidate pairs of line segments, we decouple the pair and focus on generating single line segments as candidates. Pairs of line segments corresponding to candidates of the runway will be proposed from the single line segment candidates and will be described later in this subsection. Thus, our algorithm initially searches for candidate line segments representing any edge of the runway. This is accomplished by representing a line segment by an angle θ ($\theta \in [-\pi/2, \pi/2)$, which corresponds to the angle of the line segment with respect to the $y-$axis, and a location x (x-coordinate of the center of the line segment). Our algorithm efficiently computes minimizers of

$$E'(x,\theta) = -\frac{1}{2}(u'(x,\theta) - v'(x,\theta))^2, \tag{3}$$

where u' and v' are the mean values of the image in the neighborhoods on either side of the line segment specified by x, θ.

Evaluating E' for all x, θ is still too computationally prohibitive, and thus we introduce a hierarchical coarse-to-fine search. Instead of computing u', v' for each θ, i.e., the mean intensity on either side of a slanted line segment, we design a scheme whereby the energy E' is low provided there is some $\theta \in [-\pi/2, \pi/2)$ for which E' is low, and only a *single* computation of means is required (rather than computation of means for all θ). A diagram illustrating the neighborhoods (called r_+, r_-) of the line segment, in which u', v' are computed, for various θ is shown in Figure 4. Consider the intersection of the neighborhoods r_+ and r_- for each θ between some range of angles $|\theta| < \pi/2$. The intersected regions are shown in the bottom right in Figure 4. Note that E' is low for some angle θ when the difference of means in the intersections of r_+ and the intersections of r_- is large. We call the difference of means in the intersected regions squared the *response* $\mathcal{R}$, which is a function of the location x.

The candidate locations of line segments corresponding to edges of the runway are local maxima of $\mathcal{R}$ with the highest responses. Note that since the intersection of regions r_+ has fewer pixels than any particular r_+ for a fixed angle, the means are more susceptible to noise, and therefore, computing

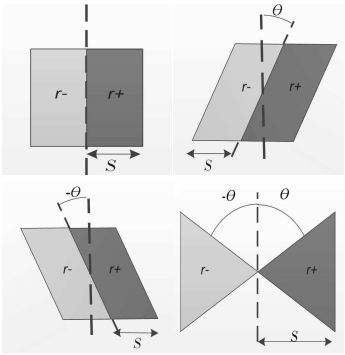

Fig. 4: Edge detectors for various orientations. The lower right regions is the intersection of all $r+$ and r_- within a specified range $[-\theta_m, \theta_m)$; it is able to detect lines with any orientation without searching within the range.

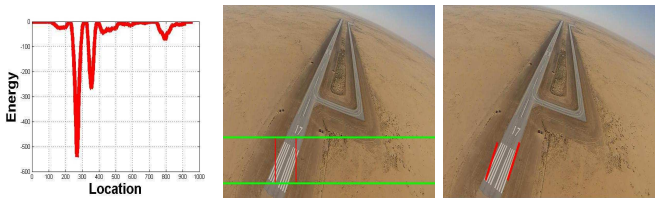

Fig. 5: **Energy profile and candidate lines.** Left: Energy profile for the image between the green lines. Middle: Red lines are chosen among the candidate edges based on the correlation matrix. Right: Red lines are edges with correct orientation.

local minima may generate false positives. However, these false positives are pruned out in the next step. Clearly, false positives are the cost for efficiency, i.e., avoiding the search over all θ at all locations. Since locating local maxima of $\mathcal{R}$ gives only candidate locations, the energy E' is computed at multiple θ (we choose $\theta \in \{-\pi/4, -\pi/8, 0, \pi/8, \pi/4\}$) *only* at the local maxima locations. This gives approximations for the orientation of the line segments. We have thus avoided an expensive search over two variables (x, θ) and only a search over x to find local minima of $\mathcal{R}$ is needed with this procedure. Fortunately, computation $\mathcal{R}$ for all x is efficient enough to be implemented in real-time. This procedure gives candidates for line segments of the runway, (x_i, θ_i), $i = 1, \ldots, N$ where N is chosen small (in our experiments, $N = 5$). This is illustrated in Figure 5.

We now 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. The energy E is thus evaluated N^2 times. Since N is small, this is a considerable time savings compared with evaluating E at all d_i. In fact, we demonstrate in the experiments that our entire detection method is real-time. The right image in Figure 5 shows an example result of the final minimum energy line segment pair.

The algorithm for this detection procedure is summarized in Algorithm 1.

1) for each $x \in \{0, 1, \ldots, L_1 - 1\}$

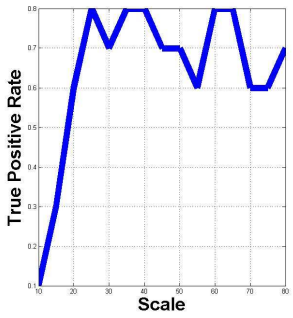

Fig. 6: True positive rate in function of the scale. This figure shows that the true positive edge detection rate is high on a wide range of scales, which illustrate the robustness of the method with respect to the scale parameter.

- compute u and v with regions defined by the mask in Fig. 4 (bottom, right) and centered at $(x, L_2/2)$ for $\theta = \pi/4$
- compute $\mathcal{R}(x) = -(u-v)^2/2$

2) compute the 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, θ_i)
5) output the pair of line segments $(x_i, y_i), (x_j, y_j)$ that have minimum E

C. Local Optimization for Refined Segmentation

We now derive a local optimization method for E given an initialization close to the desired runway. This step is needed for two reasons. First, the detection procedure in the previous section only gives a coarse approximation of the runway, which needs to be refined. Second, the algorithm will be used to track the runway in a video, and thus the lane is already roughly localized (using the result from the previous frame), and the line segments only need to be slightly refined. Local optimization is computationally fast, and less costly than running the global detection procedure described in the previous section. Therefore, we present a gradient descent method to optimize E.

One may interpret the energy E as defined on a closed contour, i.e., a trapezoid formed by the line segments. The advantage of this interpretation is that, we may use results in the PDE image segmentation literature [8], [18], [19], which have derived methods for optimizing arbitrary energies defined on closed contours that are allowed to deform in arbitrary ways that may change shape. However, in optimizing E, we would like to enforce the constraint that the trapezoid remains a trapezoid. Fortunately, such a constraint can be enforced by restricting the deformations of the contour to trapezoid preserving deformations [22].

We summarize the computation of the gradient of E, $\nabla E = (\partial E/\partial d_i)_{i=1}^4$, in the proposition below. As seen below, each partial derivative in the gradient can be efficiently computed by a line integral of the line segments.

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:*

$$\frac{\partial E}{\partial d_1} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_1}(t)) - u)dt + \frac{1}{|R^c|}\int_0^1 (t-1)(I(C_{L_1}(t)) - v)dt\right] \quad (4)$$

$$\frac{\partial E}{\partial d_2} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 t(I(C_{L_1}(t)) - u)dt + \frac{1}{|R^c|}\int_0^1 t(I(C_{L_1}(t)) - v)dt\right], \quad (5)$$

$$\frac{\partial E}{\partial d_3} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_2}(t)) - u)dt + \frac{1}{|R^c|}\int_0^1 (t-1)I((C_{L_2}(t)) - v)dt\right], \quad (6)$$

$$\frac{\partial E}{\partial d_4} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 t(I(C_{L_2}(t)) - u)dt + \frac{1}{|R^c|}\int_0^1 t(I(C_{L_2}(t)) - v)dt\right], \quad (7)$$

where C_{L_i} is the arc-parameterization of the i^{th} line segment corresponding to $(d_1, \ldots, d_4)$, L_2 is the image width, and u and v are the means inside the regions R and R^c, respectively.

The interested reader may find the derivation of the expressions in Appendix A.

The gradient descent algorithm to minimize this energy is then given by

$$\vec{d}_{k+1} = \vec{d}_k - \Delta t \nabla E(\vec{d}_k), \quad (8)$$

where $\vec{d}_k$ is the vector of d_is at each iteration of the gradient descent, and $\Delta t = 0.5/|\nabla E(\vec{d}_k)|$ is the time step size. Note that $\vec{d}_0$, the initialization, is the result of the detection described in the previous section at the beginning of the video. For subsequent frames, it is the prediction from the previous frame (see next section for details).

IV. Tracking

We use Kalman filtering (KF) [4] to predict and the initial location of the lines before they are processed by the gradient descent, and then to arrive at a final estimate. This allows us to process fewer frames, since we may predict ahead a few frames, and increase the computational speed of the runway detection algorithm, to enable this process to run in real time. Moreover, Kalman Filtering increases the robustness against clutter and outliers that are in the field of view of the camera. Finally, it can also be used to fuse inertial measurement data (such as rotation rates measured by the onboard gyrometers) to enable video-inertial data fusion. We assume a constant velocity plus noise model, which gives a first order approximation to the dynamics of the line segments

seen through the camera. We define the state space X_k to include the vector $\vec{d}_k$ and their velocities, which makes X_k an 8-dimensional vector. We assume the measurement $y_k \in \mathbb{R}^4$ is a noisy measurement of the first components of X_k ($\vec{d}_k$), and these measurements will be obtained by the converged result of the gradient descent, described in the previous section, with initialization the predicted relevant part of the state.

The dynamical system of interest can be defined as follows:

$$X_k = (\vec{d}_k^T, \vec{v}_k^T)^T \in \mathbb{R}^8 \tag{9}$$

$$X_{k+1} = AX_k + Bu_k + \zeta_k \tag{10}$$

$$y_{k+1} = CX_k + \eta_k \tag{11}$$

$$\zeta_k \sim \mathcal{N}(0, Q) \tag{12}$$

$$\eta_k \sim \mathcal{N}(0, U) \tag{13}$$

where $\vec{v}_k$ is the velocity corresponding to the horizontal distances $\vec{d}_k$, S is the measurement noise covariance matrix, and U the process (or model) noise covariance matrix. In our specific problem (constant velocity plus noise), these matrices are defined as follows [1]:

$$A = \begin{bmatrix} \mathrm{id}_{4\times4} & \mathrm{id}_{4\times4} \\ 0_{4\times4} & \mathrm{id}_{4\times4} \end{bmatrix},\ B = 0,\ C = \begin{bmatrix} \mathrm{id}_{4\times4} & 0_{4\times4} \end{bmatrix} \tag{14}$$

$$U = \begin{bmatrix} 0.01 \times \mathrm{id}_{4\times4} \end{bmatrix},\ Q = \begin{bmatrix} 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & 0.01 \times \mathrm{id}_{4\times4} \end{bmatrix}. \tag{15}$$

Note that X_0 is chosen to be the output of the detection procedure, and zero velocity.

The Kalman Filter involves two steps: prediction and estimation. We use the "hat" notation to denote predicted and estimated quantities.

1) Prediction:

$$\hat{X}_{k|k-1} = A\hat{X}_{k-1|k-1}, \tag{16}$$

and the predicted estimated covariance:

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \tag{17}$$

2) Estimation:

- Measurement error/innovation covariance:

$$S = HPH^T + U \tag{18}$$

- Optimal Kalman Gain:

$$K = PH^T S^{-1} \tag{19}$$

- Updated state estimate:

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K(y_k - C\hat{X}_{k|k-1}) \tag{20}$$

where y_k is obtained by running gradient descent (8) with initialization the first four components of $\hat{X}_{k|k-1}$.

[1]In this section, we do not assume to have any available inertial or positional measurements. Such measurements could be used to modify the prediction step of the Kalman Filter (for instance the rotation rates could be used in the prediction step to recompute the predicted runway edge locations, enabling a more accurate and robust tracking).

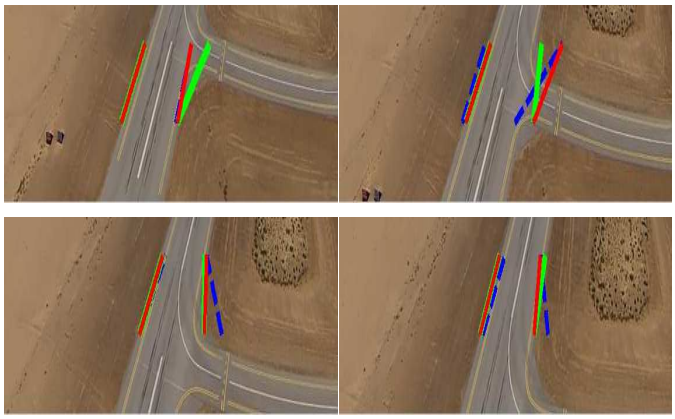

Fig. 7: **Visualization of the Kalman Filter performance in tracking the runway edges.** We consider on this picture four consecutive time steps (Left to right, and up to down). Blue line: predicted runway edge locations (from previous step). Green line: runway edge locations (measurements) using vision based segmentation (outlined in section III of the present article). Red lines: estimation (output of the Kalman filter).

- Updated estimate covariance:

$$P_{k|k} = P_{k|k-1} - KHP_{k|k-1} \tag{21}$$

where P_k is the state variance matrix which can be initialized with zeros.

Figure 7 shows an example of the filtering procedure above.

V. Experimental Results

A. Scale Sensitivity

An important parameter of the runway edge detection and tracking algorithm presented in this article is the scale parameter. Which scale should be used, and how robust would this choice be to experimental landing conditions? On the two extremes, a small scale would be too sensitive to image noise, while a large scale may be unable to capture the location of the runway. If no altitude or positional information is available to the UAV, the scale of the runway (on the image) is unknown. If this information is available, one can specify the approximate scale of the runway, which would facilitate the detection of the runway. The area of the neighborhood included in the statistical analysis presented in section III is also another difficulty, since increasing this area could lead to the inclusion of outliers or clutter located around the edges of the runway. This could lead to another line being falsely detected as a runway edge (false positive), which is undesirable. We ran multiple experiments over ten representative frames including different views of the lane. Accordingly, there should be a scale by which we can detect the edges easily and accurately and choose it to be part of the lane correctly. This is considered as true positive result. Running an experiment, over ten representative frames of different views of the lane with some clutters in the surrounding over a scale

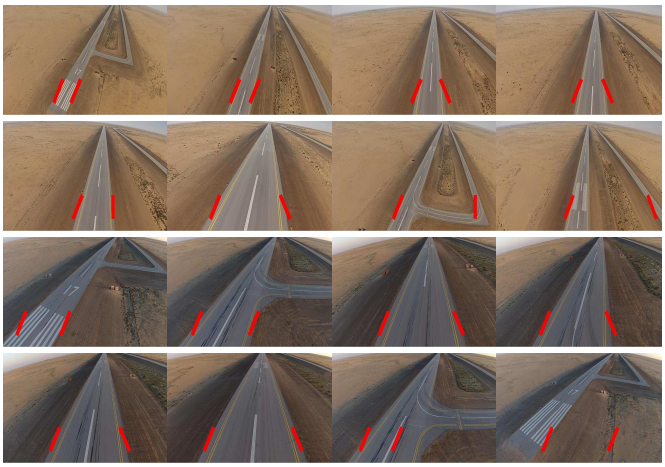

Fig. 8: **Output of the runway detection algorithm.** This figure shows the detection output for different views for one video sequence.

range $S \in [10, 15, .., 75, 80]$, and $\theta = 45°$), gave us a good indicator on how we can choose the scale of the filter. Hence, the detection process is very sensitive to the scale. Figure 6 shows that the best scale is $S = 40$. On the other hand, $S = 60$ gave the same results, however, it is considered to be big scale. Figure 6 shows the true positive rate (TPR) as a function of the chosen scale.

B. Experimental validation

We evaluated the efficiency of our algorithm over video samples downloaded from a UAV onboard camera filming landing procedures from different altitudes, positions, and circumstances. These samples obtained using Saker 4, which is one of the medium range UAVs that that has been developed by King Abdulaziz City for Science and Technology (KACST). tested video has a frame resolution of 1920x1080 pixels. The algorithm achieves an average speed of 30 frames per second on this high resolution video. The code shows promising results to detect the runway. It is accurate, simple, efficient and supporting real time application. Moreover, to check the robustness of our code. We applied it under deferent brightness conditions using day and evening video samples.

C. Detection Results

Figure 8 shows the output of sample detection results on set out of 20 images of the runway taken at different lighting conditions and multiple views of the runway. The results show that our detection algorithm is able to correctly detect the lane in 90% of the cases, and is thus robust to UAV position and lighting conditions. All detection results were obtained with a fixed scale parameter of $S = 40$ (determined empirically from a training set not used in the evaluation).

D. Segmentation Results

The gradient descent evolution is depicted for different types of initialization in Figure 11. This shows how robustly the segmentation algorithm can converge towards the runway for any type of initialization around the it.

Fig. 9: **Saker 4 UAV.** This UAV has a maximum takeoff weight of 25 kg and can carry a payload of 5 kg. Its wingspan is 3.75 meters and can fly at maximum speed of 120 kms per hour, and a maximal altitude of 5,000 meters.

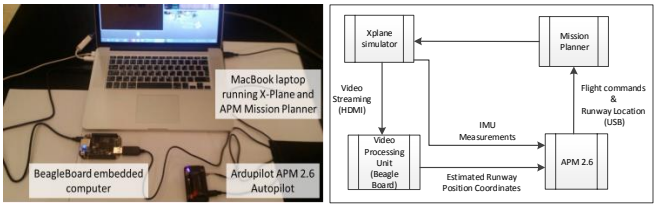

Fig. 10: **Hardware in the Loop simulation system.**

E. Tracking Results

Figure 12 and Figure 13 show how the lane is tracked under different light conditions by which robustness is proved. The red lines show the estimated location of the runway using KF over a sequence of images.

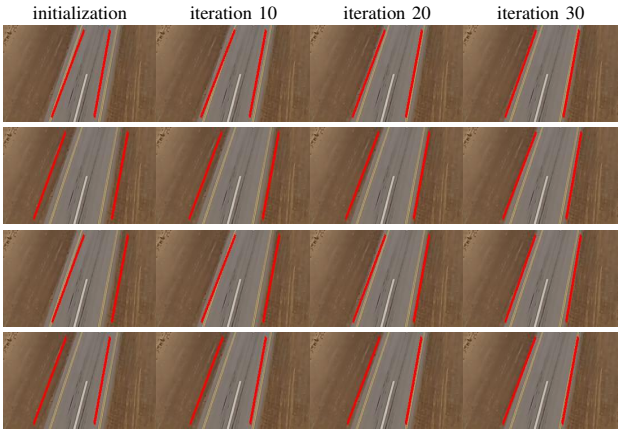

Fig. 11: **GD evolution for different initializations with all possible types of initialization.** From the top row to the bottom: outer lines, inner lines, right shift, and left shift, respectively, are the different initializations.

VI. Hardware in the loop simulation

Hardware in the Loop (HIL) consists in interfacing a simulator (presently a flight simulator) to a sensing and control system (presently the navigation and control system of a UAV) to evaluate its performance under simulated reality conditions.

In the present case, our HIL framework consists in the following. The `X-Plane` flight simulator simulates the view that a UAV in a landing phase would see, and feeds this video data to an embedded computer (in the present case a `TI Beagle Board`). The embedded computer has a ARM processor operating at 1GHz, and 512MB of DDR3 RAM. This computer receives the video data through a HDMI cable (though in reality the video would be acquired by a high resolution video camera), and processes this streaming data using the methods outlined in section III. The resulting edge estimates are sent to an autopilot (currently a `Pixhawk` autopilot from `3D robotics`), which fuses this information with positional and inertial measurements (if available) using the Kalman Filter outlined in section IV.

We validated the robustness of the algorithm on various video sequences generated by the X-Plane flight simulator. Examples of runway detections in various environmental conditions are illustrated in Figure 14 below.

VII. Conclusion

We have presented a robust, fully automatic and real-time runway detection and tracking algorithm for UAV landing applications. Unlike recent approaches to runway detection using computer vision, our approach requires no rendered computer model of the shape and appearance of the runway. This is particularly advantageous since such rendered models are only applicable to a pre-defined runway geometry and under specific lighting and weather conditions. Generating rendered computer models under all possible illumination / weather conditions and runway geometries is not scalable to the wide range of conditions that are observed on a UAV (even under a pre-defined runway). 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 dissimilarity to the immediate background. These simple assumptions are more widely applicable to general runways than a specific rendered model, and thus our approach is scalable to realistic conditions encountered by the UAV. We have demonstrated a new model for runways, created an associated optimization problem for the runway, created a robust and efficient coarse-to-fine detection algorithm to localize the runway, and we derived a local optimization approach to refine the results of detection and is used in tracking. The algorithms have been demonstrated experimentally to be robust to a wide range of lighting conditions and varied runways, and shown to be real-time.

VIII. Acknowledgment

We are very grateful to the Aerospace division of King Abdulaziz City for Science and Technology (KACST) for supporting us with experimental flight videos of the SAKER 4 UAV. The algorithm developed as part of this project will eventually be integrated and tested with the navigation system of SAKER 4.

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 $\vec{d} = (d_1, d_2, d_3, d_4)$. Using the Chain Rule, we see that

$$\nabla E(\vec{d}) = -(u(\vec{d}) - v(\vec{d}))(\nabla u(\vec{d}) - \nabla v(\vec{d})), \qquad (22)$$

where ∇ deontes the gradient with respect to $\vec{d}$.

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

$$u(\vec{d}) = \frac{\int_R I(x)dA}{\int_R dA}, \quad v(\vec{d}) = \frac{\int_{R^c} I(x)dA}{\int_{R^c} dA}, \qquad (23)$$

where R is the banded region inside the trapezoid determined by $\vec{d}$, R^c is the banded region outside the trapezoid determined by $\vec{d}$, and dA is the area differential element. To compute the gradient, we may use the result [25] 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

$$\begin{aligned} \frac{\partial u}{\partial d_j} &= \frac{1}{|R|}\int_{C_{L_i}} (I-u)h_j.N ds \\ \frac{\partial v}{\partial d_j} &= -\frac{1}{|R^c|}\int_{C_{L_i}} (I-v)h_j.N ds \end{aligned} \qquad (24)$$

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 j^{th} 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

$$C_{L_i}(t) = \begin{cases} \binom{d_1}{0} + \binom{d_2-d_1}{L_2}t & \text{if } \ i = 1,2 \\ \binom{L_1-d_3}{0} + \binom{d_3-d_4}{L_2}t & \text{if } \ i = 3,4 \end{cases} \qquad (25)$$

and

$$h_j = \frac{\partial C(t)}{\partial d_j} = \begin{cases} \binom{1-t}{0} & if \quad j = 1,3 \\ \binom{t}{0} & if \quad j = 1,4 \end{cases}. \qquad (26)$$

Fig. 12: Video sequence 1 showing the tracking output.

Fig. 13: Video sequence 2 shows the tracking output

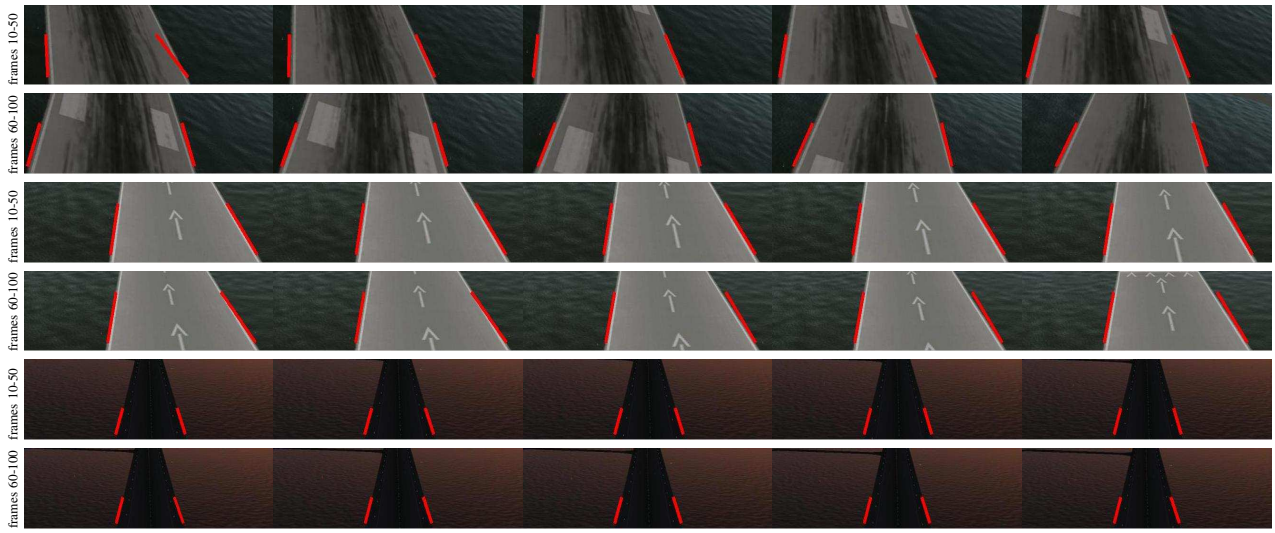

Fig. 14: Video sequences from Xplane under different illumination. Every two rows correspond to one environment

Since 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

$$J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \tag{27}$$

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

$$T(t) = \frac{C'_{L_j}(t)}{\left|C'_{L_j}(t)\right|} \tag{28}$$

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

length parameter,

$$ds = \left| C'_{L_j}(t) \right| dt, \tag{29}$$

and L_j will denote the length of C_{L_j}. Combining equations (25), (26), (28) and (29) results in

$$N(t).h(t)ds = L_2(t-1)dt. \tag{30}$$

Then, the gradient direction at vertex $j = 3$, for example, can be derived as follows

$$\frac{\partial E}{\partial d_3} = -L_2(u-v)[\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_2}(t)) - u)dt + \frac{1}{|R^c|}\int_0^1 (t-1)I((C_{L_2}(t)) - v)dt] \tag{31}$$

The same argument applies to the other d_i, and thus

$$\frac{\partial E}{\partial d_1} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_1}(t)) - u)dt + \frac{1}{|R^c|}\int_0^1 (t-1)(I(C_{L_1}(t)) - v)dt\right] \tag{32}$$

$$\frac{\partial E}{\partial d_2} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 t(I(C_{L_1}(t)) - u)dt + \frac{1}{|R^c|}\int_0^1 t(I(C_{L_1}(t)) - v)dt\right], \tag{33}$$

$$\frac{\partial E}{\partial d_3} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 (t-1)(I(C_{L_2}(t)) - u)dt + \frac{1}{|R^c|}\int_0^1 (t-1)I((C_{L_2}(t)) - v)dt\right], \tag{34}$$

$$\frac{\partial E}{\partial d_4} = L_2(v-u)\left[\frac{1}{|R|}\int_0^1 t(I(C_{L_2}(t)) - u)dt + \frac{1}{|R^c|}\int_0^1 t(I(C_{L_2}(t)) - v)dt\right], \tag{35}$$

REFERENCES

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898–916, 2011.

[2] Thomas Brox and Daniel Cremers. On local region models and a statistical interpretation of the piecewise smooth mumford-shah functional. *International journal of computer vision*, 84(2):184–193, 2009.

[3] Daniel Cremers, Mikael Rousson, and Rachid Deriche. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. *International journal of computer vision*, 72(2):195–215, 2007.

[4] Erik V Cuevas. Kalman filter for vision tracking. 2005.

[5] Damien Dusha, Wageeh W Boles, and Rodney Walker. Fixed-wing attitude estimation using computer vision based horizon detection. 2007.

[6] Yang Gui, Pengyu Guo, Hongliang Zhang, Zhihui Lei, Xiang Zhou, Jing Du, and Qifeng Yu. Airborne vision-based navigation method for uav accuracy landing using infrared lamps. *Journal of Intelligent & Robotic Systems*, 72(2):197–218, 2013.

[7] Michael Kontitsis, Kimon P Valavanis, and Nikos Tsourveloudis. A uav vision system for airborne surveillance. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 77–83. IEEE, 2004.

[8] Pierre Kornprobst, Rachid Deriche, and Gilles Aubert. *Image sequence restoration: A PDE based coupled method for image restoration and motion segmentation*. Springer, 1998.

[9] Andrew HS Lai and Nelson HC Yung. Lane detection by orientation and length discrimination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 30(4):539–548, 2000.

[10] Maximilian Laiacker, Konstantin Kondak, Marc Schwarzbach, and Tin Muskardin. Vision aided automatic landing system for fixed wing uav. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2971–2976. IEEE, 2013.

[11] King Hann Lim, Kah Phooi Seng, Li-Minn Ang, and Siew Wen Chin. Lane detection and kalman-based linear-parabolic lane tracking. In *Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC'09. International Conference on*, volume 2, pages 351–354. IEEE, 2009.

[12] Weirong Liu and Shutao Li. An effective lane detection algorithm for structured road in urban. In *Intelligent Science and Intelligent Data Engineering*, pages 759–767. Springer, 2013.

[13] Gérard Medioni, Isaac Cohen, François Brémond, Somboon Hongeng, and Ramakant Nevatia. Event detection and analysis from video streams. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(8):873–889, 2001.

[14] Ding Meng, Cao Yun-feng, and Guo Lin. A method to recognize and track runway in the image sequences based on template matching. In *Systems and Control in Aerospace and Astronautics, 2006. ISSCAA 2006. 1st International Symposium on*, pages 4–pp. IEEE, 2006.

[15] Worawit Phueakjeen, Nattha Jindapetch, Leang Kuburat, and Nikom Suvanvorn. A study of the edge detection for road lane. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2011 8th International Conference on*, pages 995–998. IEEE, 2011.

[16] Thomas Pock, Daniel Cremers, Horst Bischof, and Antonin Chambolle. An algorithm for minimizing the mumford-shah functional. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1133–1140. IEEE, 2009.

[17] Srikanth Saripalli, James F Montgomery, and Gaurav Sukhatme. Vision-based autonomous landing of an unmanned aerial vehicle. In *Robotics and automation, 2002. Proceedings. ICRA'02. IEEE international conference on*, volume 3, pages 2799–2804. IEEE, 2002.

[18] Anastasia Sofou, Georgios Evangelopoulos, and Petros Maragos. Coupled geometric and texture pde-based segmentation. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2, pages II–650. IEEE, 2005.

[19] Anastasia Sofou and Petros Maragos. Generalized flooding and multicue pde-based image segmentation. *Image Processing, IEEE Transactions on*, 17(3):364–376, 2008.

[20] Yuan-Ling Tang and Rangachar Kasturi. Runway detection in an image sequence. In *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, pages 181–190. International Society for Optics and Photonics, 1995.

[21] Sinisa Todorovic and Michael C Nechyba. Intelligent missions for mavs: visual contexts for control, tracking and recognition. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1640–1645. IEEE, 2004.

[22] Gozde Unal, Anthony Yezzi, and Hamid Krim. Information-theoretic active polygons for unsupervised texture segmentation. *International Journal of Computer Vision*, 62(3):199–220, 2005.

[23] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. Lane detection and tracking using b-snake. *Image and Vision computing*, 22(4):269–280, 2004.

[24] Anthony Yezzi, Andy Tsai, and Alan Willsky. A fully global approach to image segmentation via coupled curve evolution equations. *Journal of Visual Communication and Image Representation*, 13(1):195–216, 2002.

[25] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9):884–900, 1996.