Free LSD: Prior-Free Visual Landing Site Detection for Autonomous Planes

Timo Hinzmann[1], Thomas Stastny[1], Cesar Cadena[1], Roland Siegwart[1], and Igor Gilitschenski[1,2]

***Abstract*— Full autonomy for fixed-wing unmanned aerial vehicles (UAVs) requires the capability to autonomously detect potential landing sites in unknown and unstructured terrain, allowing for self-governed mission completion or handling of emergency situations. In this work, we propose a perception system addressing this challenge by detecting landing sites based on their texture and geometric shape without using any prior knowledge about the environment. The proposed method considers hazards within the landing region such as terrain roughness and slope, surrounding obstacles that obscure the landing approach path, and the local wind field that is estimated by the on-board EKF. The latter enables applicability of the proposed method on small-scale autonomous planes without landing gear. A safe approach path is computed based on the UAV dynamics, expected state estimation and actuator uncertainty, and the on-board computed elevation map. The proposed framework has been successfully tested on photo-realistic synthetic datasets and in challenging real-world environments.**

I. Introduction

Small-scale autonomous planes promise to become a ubiquitous tool in the commercial, industrial, and scientific sectors due to reduced operational costs and ever increasing robustness. Especially the ability to map large areas and to carry out perpetual surveillance tasks, e.g. by using a solar-powered platform, makes this type of unmanned aerial vehicles (UAVs) interesting for various applications. While mission operation can already be completely automated [1], appropriate landing site detection (LSD) and the actual landing procedure still requires an experienced safety pilot. Furthermore, in future fully autonomous beyond visual line-of-sight (BVLOS) operation, finding an appropriate landing spot in unstructured terrain is essential for handling emergency scenarios.

Existing LSD systems focus on the cases of vertical takeoff and landing (VTOL) platforms, or large-scale planes, may rely on offline-computed data, or require prior knowledge about the environment. These approaches are not suited for small-scale autonomous planes operating in unknown environments which are constrained by potentially limited energy supply and computational power. Furthermore, their size and speed requires taking the wind into consideration, and due to their potential absence of landing gear, preferably landing in flat grass to not damage wings or fuselage.

The present work proposes *Free LSD*, a real-time visual landing site detection and approach path computation algorithm for autonomous fixed-wing UAVs. To keep the problem complexity manageable, potential landing sites are tracked and ranked over multiple frames. Only the most promising landing sites are forwarded for finer-grained, 3D processing. No a priori data such as markers, pre-classified Digital Surface Maps (DSM), or orthomosaics are utilized which allows the framework to be operated in completely unknown terrain as is exemplary shown in Fig. 1. To the best of the authors' knowledge, this paper presents the first such system, which is also suitable for application on small-scale UAVs. The work incorporates wind field and nearby obstacle consideration during approach path generation and decision making. Performance of the full framework is evaluated in unknown terrain using various synthetic datasets and real-world test flights.

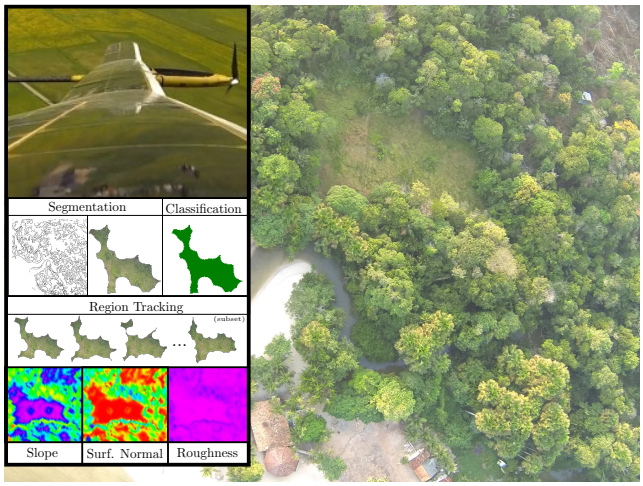


Fig. 1: The goal is to find the optimal landing spot while considering terrain shape, terrain texture, terrain roughness, terrain slope, surrounding obstacles, estimated local wind field, and UAV dynamics and their uncertainties.

II. Related Work

Automated landing of VTOL UAVs has been considered in a broad body of works. For instance, Desaraju et al. [2] propose a vision-based landing site evaluation framework to land on rooftops employing a Gaussian process to estimate the landing site confidence. Forster et al. [3] present an efficient way to compute a vision-based elevation map on-board of a quadrocopter. Johnson et al. [4] use a LIDAR-based elevation map to compute terrain smoothness, roughness, and incidence angles to determine safe landing spots for spacecrafts. Garcia-Padro et al. [5] introduce a contrast descriptor to land an autonomous helicopter far away from obstacles under the assumption that the terrain is flat.

[1]Autonomous Systems Lab, ETH Zurich, Switzerland.
[2]CSAIL, MIT, Cambridge, MA, USA.

Brockers [6], Cheng [7], and Bosch et al. [8] make use of homography decomposition for identifying planar landing spots. Theodore et al. [9] employ a stereo vision rig mounted on an unmanned helicopter to compute a range map and infer safe landing spots based on roughness, slope, and distance to closest obstacle. The above approaches have in common that the main criterion for VTOL UAVs is flatness of the landing spot. However, our application requires taking the plane dynamics and additional space requirements during landing into consideration.

For fixed-wing platforms, most research focuses on cases where the system recognizes modified environments or man-made structures, or where the landing site is pre-defined. Visual servoing is employed by Huh et al. to steer a small fixed-wing UAS into a red dome-shaped airbag located in an obstacle-free area [10]. Similarly, the framework proposed by Laiacker et al. [11] recognizes a runway from the UAV and compares it to a known model. Given a designated, obstacle-free landing site, the height above the ground plane can be estimated using monocular visual-inertial [12] or biologically inspired stereo vision [13].

In contrast to the aforementioned works, this paper aims at actively selecting appropriate landing spots in an *unknown* environment. This requires generation and assessment of potential candidate areas which has, to the best of our knowledge, only been discussed in two publications: Fitzgerald et al. [14] seek to find suitable areas for crash-landing an airplane in case of emergency. This is achieved by detecting areas without edges on a low-quality image from a defined height of $2500\,\mathrm{ft}$, before classifying them in order to retrieve large grass fields. However, relying on a fixed height makes this approach disadvantageous in case of emergencies. The closest approach to ours is presented by Warren et al. [15]. However, we see the following caveats that we address with the present work: Firstly, the terrain classification is derived from stored data. Secondly, the approach trajectory and height of nearby obstacles is only considered indirectly by the Principal Component Analysis (PCA). Thirdly, wind is not considered which has a large effect on smaller and light-weight planes. Finally, the approach by Warren et al. [15] does not run in real-time.

III. The Approach

An overview of our proposed algorithm for the detection of landing sites is shown in Fig. 2: The raw image is segmented into homogeneous regions (Sec. III-C) and classified into `grass` or $\neg$`grass` using a binary Random Forest (RF) classifier (Sec. III-D). In parallel, the on-board EKF of the *Pixhawk* autopilot estimates UAV pose and local wind field (Sec. III-A), and depending on the provided image rate and overlap of subsequent frames, a tracker or matcher is employed to connect consecutive camera frames via feature tracks. Resulting coarse depth measurements (Sec. III-B) are used in the region manager to track region of interests (ROI) based on geometry. The region manager (Sec. III-E) accumulates all information about the regions and ensures consistency and uniqueness by merging regions. Based on these metrics, a coarse grade determines which region is passed on as a candidate to the fine, 3D evaluation backend. This backend is periodically updated by the frontend with the n most promising ROIs. All observations of a ROI, UAV pose estimates, and previously generated feature tracks are used to perform key-frame based bundle adjustment (BA) and dense 3D reconstruction (Sec. III-F). Metrics such as terrain slope and roughness (Sec. III-G) are derived from the classification results and 3D model. A distance-to-hazard map determines the landing spot with maximum distance to the next hazard. Based on this touch down point, the estimated local wind field (Sec. III-H, III-I), and the 2.5D elevation map, a collision-free approach path is computed. The final decision module outputs the landing site location, optimal approach vector, and statistics about the final landing site. The actual tracking of the final approach path is described in [1]. The metrics of the best landing sites are stored to be able to land quickly in the case of an emergency.

A. State Estimation

The state estimator on the *Pixhawk* autopilot estimates body poses, velocities, IMU biases, and the wind field using GNSS, IMU, magnetometer, and pressure measurements [16]. The camera pose estimates are forwarded to the on-board computer which associates camera poses to the corresponding images based on the pre-calibrated camera-IMU transformation [17]. These camera pose estimates are used as priors in the bundle adjustment if an area was marked as potential landing spot. Additionally, feature tracks are generated using, depending on the provided framerate, a Kanade-Lucas-Tomasi (KLT) [18] feature tracker or matcher. These feature tracks are used to generate coarse depth measurements (cf. Sec. III-B) for region tracking in unknown terrain and in the bundle adjustment of the backend thread (cf. Sec. III-F).

B. Coarse Depth Measurements

To obtain a segmentation that is robust to height changes as well as for geometric region tracking, coarse depth measurements are required in the frontend (cf. Sec. III-E). Since our system is designed to operate in unknown terrain without a priori data, the depth measurements need to be retrieved at runtime[1]. One possibility would be to triangulate a few features at every step and build up a mesh by using, for instance, Delaunay triangulation [19]. However, to obtain depth measurements at a given pixel, computationally expensive ray-casting queries would be required. Furthermore, a depth image obtained from two views from a virtual stereo rig based on unoptimized camera pose estimates is prone to errors since we assume a noisy low-level state estimator. Instead, we take advantage of the feature tracking thread that is running in parallel: To map from 2D to 3D coordinates, the N feature tracks closest to the queried keypoint location are determined. The final height

[1]Depending on the application and flight altitude, the coarse depth measurements could alternatively be obtained from a ground plane approximation.

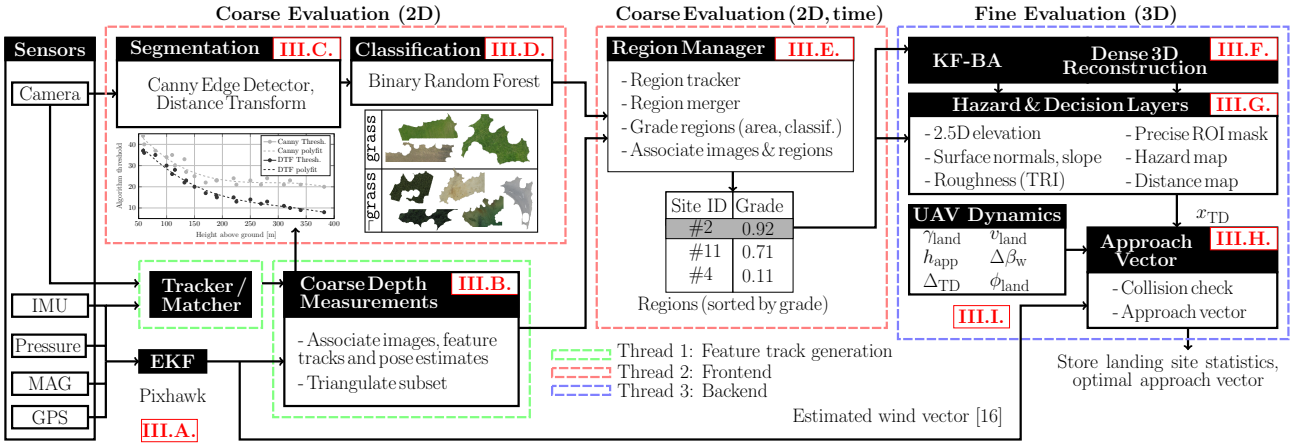
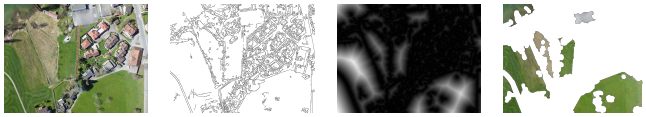


Fig. 2: Proposed framework for prior-free landing site detection. The frontend segments, classifies, and manages the potential region of interests (ROIs). The most promising site is forwarded to the backend for finer 3D analysis and computation of approach path.

of the requested keypoint location is obtained by performing multi-view triangulation of the N nearby tracks and inversely weighting the resulting triangulated landmark heights by their distance to this keypoint.

C. Region Segmentation

The Canny edge detector [20] is applied to the grayscale spectrum of the raw image (cf. Fig. 3a). The result is shown in Fig. 3b. Next, the distance transform [21] is applied to compute for every pixel the distance to the closest non-zero pixel or Canny edge. The distance map, as shown in Fig. 3c, is then thresholded to obtain homogeneous regions (cf. Fig. 3d). Note that high contrast obstacles, such as the trees in the lower right section of the images, are often already identified at this early stage. The threshold in the Canny edge detector and the distance transform is computed from a function of height, to ensure that the same areas are segmented independently of the UAV's altitude above ground[2]. The thresholds are derived from Google Earth imagery and span an altitude range of 58-382 m above ground. For reference, the nominal flight altitude of the deployed UAVs in this publication is between 50 and 250 m.

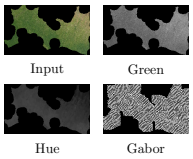


(a) **(b)** **(c)** **(d)**

Fig. 3: Region segmentation: (a) Original input image, (b) Canny edges, (c) Distance transformation, (d) Segmented regions. Regions with a small area are rejected already at this step.

D. Region Classification

The segmentation module presented in the previous section only ensures that the extracted area is homogeneous. In the classification step, the texture and color properties of the homogeneous area are extracted to classify the regions into `grass` or `¬grass` as illustrated in Fig. 2. For this purpose, we employ a binary Random Forest (RF) [22] classifier which takes the segment from Fig. 3d and predicts the binary label. The classifier is trained based on a set of features extracted from the homogeneous regions. The parameters of the classifier, that is, the maximal tree depth and the number of samples needed per branch, are optimized on the training data by 10-fold cross-validation. The ground truth for the classification is established as follows: Homogeneous regions are obtained by the described segmentation algorithm. After visual inspection, the region is manually labeled as `grass` or `¬grass`.



Color	Texture
$\mu_{\text{B}}, \mu_{\text{G}}, \mu_{\text{R}}, \sigma_{\text{B}}, \sigma_{\text{G}}, \sigma_{\text{R}}$	$\mu_{\text{G0.5}}, \mu_{\text{G1.0}}, \mu_{\text{G5.0}}$
$\mu_{\text{H}}, \mu_{\text{S}}, \mu_{\text{V}}, \sigma_{\text{H}}, \sigma_{\text{S}}, \sigma_{\text{V}}$	$\sigma_{\text{G0.5}}, \sigma_{\text{G1.0}}, \sigma_{\text{G5.0}}$

Fig. 4: Features used for binary classification of homogeneous regions. Note that the Gabor feature applied in form of a convolutional filter expands the area, but only the information within the ROI mask is used to compute mean and standard deviation.

1) Feature Space: For each segmented ROI, twelve color and six texture features, are extracted as summarized in Fig. 4.

a) Color: For each sub-image, the mean and standard deviation for all three color channels are computed across the complete segmented ROI. This is performed not only in the standard RGB color space, but also in the HSV space. In many computer vision applications, the HSV space has proven to be less sensitive to lighting conditions, when comparing to RGB [23]. While the classifier performs better using the HSV color space than RGB only, it performs even slightly better when using the features extracted from both: The classification error for only using RGB is 15.36 %, 14.26 % for HSV, and 14.12 % for RGB and HSV. We hence get a total of (2 color spaces) × (3 colors) × (2 features per color) = 12 color features which are computed for every sub-image. While more advanced color features can be extracted,

[2]The height-dependent thresholds were approximated by $p_{\text{canny}}(h) = -1.72e-06h^3 + 0.00148h^2 - 0.43h + 62.97$, and $p_{\text{dtf}}(h) = -1.23e-06h^3 + 0.0011h^2 - 0.39h + 56.82$ as shown in Fig. 2.

e.g. using various combinations of color histograms [24], we here only rely on these very simple features for low computational costs.

b) Texture: Color features are sensitive to illumination and viewing angle. To better assess the spatial arrangement of intensities in an image patch we additionally compute texture descriptors. For this task, we employ Gabor filters [25], linear filters related to the Gabor Wavelet that extract texture features from gray-scale imagery [26] more efficiently than alternatives, such as Local Binary Patterns (LBP) [27], [28]. The following parameters for phase offset φ, standard deviation of the Gaussian function σ, and spatial aspect ratio γ are used: $\varphi = 0$, $\sigma = 4$ and $\gamma = 0.02$. The orientation θ in which the edges are detected is not important in our case, since we try to detect rotation-independent descriptors. The Gabor filtered images are computed by applying a convolutional filter in four directions $\theta \in \{0, \pi/4, \pi/2, 3\pi/4\}$ and taking the mean of the extracted values. This approach yields three Gabor filtered images for the wavelengths $\lambda \in \{0.5, 1, 5\}$. The final descriptors used in the classifier correspond to the mean and standard deviation of each of these Gabor filtered images, hence a total of six texture descriptors.

E. Region Manager: Tracking, Merging and Updating

1) Tracking and Updating of ROIs: As illustrated in Fig. 5, the classification and segmentation module forwards the contours of a fine classification mask, defined by a set of 2D points, to the region manager. To simplify tracking and to increase the robustness with respect to impairing factors[3], the fine classification mask is approximated by the minimum-area enclosing rectangle using the rotating caliper method [29], [30]. Next, the 2D positions of the four corners and centroid of the rectangle are projected into 3D based on the available coarse depth measurements (cf. Sec. III-B). As depicted in Fig. 6, two cases are distinguished for initializing and updating ROIs: In the first case, the ROI is fully visible, i.e. all 2D corners are within the current image. If so, the corresponding 3D corners are fixed and ROI statistics (n_{grass}, n_{obs}, grade, cf. Sec. III-E.3) are set. In a subsequent frame, a re-detection is triggered if the centroid of the ROI in the current frame is within the corners of an existing ROI, which is determined by the winding number method [31]. In this event, only the tracked ROI statistics are updated. In a second case, if the ROI is not fully visible, i.e. one or more corners are on the border of the image, the 3D corners are set but not fixed. If, in a subsequent re-detection, the ROI is again not fully visible the corners are updated by the vertices of the rectangle that incorporates all 8 corners [29], [30] until the ROI is fully visible and the first case applies.

2) Merging of tracked ROIs: It can occur that two tracked regions of interest correspond in fact to the same landing area. An example for this would be if only half of the area is detected in a series of subsequent frames, while the other half is detected later on in other frames. However, in following

[3]E.g. the depth approximation introduced by the coarse depth measurements utilized for the 2D to 3D projection.

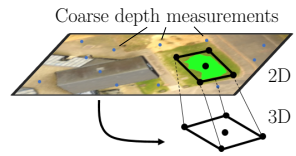


Fig. 5: ROI initialization

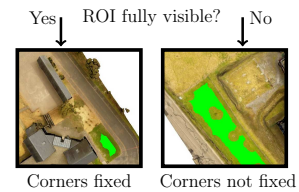


Fig. 6: ROI tracking

images, the UAV could detect the complete area. Without any merging, the pipeline would update values such as the corner position or the area size for one of these two areas because the projection of the newly detected center point is placed inside it, while the other half would remain unchanged. To avoid this duplication, every time the corner positions of a tracked ROI get updated, we verify for each ROI in the tracker if it belongs to that area, i.e. if the center of the newly updated area is in-between the four corners of the tracked ROI. If that is the case, the two ROIs are merged: the corner positions are set to the ones of the largest area and the grade is updated accordingly.

3) Grading of tracked ROIs: The tracked ROIs are ranked according to a cost function assigning a grade to each landing spot. The grading function makes use of metrics computed for each tracked region: The area A spanned by the four projected corners, the number of images in which the ROI has been classified as grass n_{grass}, and the total number of images in which it has been observed n_{obs}. The grade is zero if $\text{A} < \text{A}_{\text{min}}$ or $n_{\text{obs}} < n_{\text{obs,min}}$ and $n_{\text{grass}} n_{\text{obs}}^{-1}$ otherwise. In order to reduce the computational load, only the 20 ROIs with the highest grade are retained. This is implemented in form of a FIFO buffer in order to first remove regions which have not been detected or recognized recently.

F. Dense 3D Reconstruction

To reduce the computational burden, the subset of frames is iteratively selected for pose refinement and dense reconstruction as follows: The first pose is set as key-frame (KF). Then the next frame for which the feature track connection count first drops below 30 is determined. The predecessor to this frame is the next KF if the baseline is larger than a minimal baseline. Next, for every KF, the best suited stereo-pair is selected based on baseline, epipolar and viewing cost [32]. The selected set of poses is refined by incorporating pre-computed pose priors and feature tracks (cf. Section III-A). Finally, the optimized poses are used for planar rectification [33], [32] in combination with Semi-Global Block-Matching (SGBM [30]). As described in Section III-G, inverse distance weighting (IDW) is used to convert from 3D point cloud to 2.5D elevation map which smoothes the depth estimates.

G. Hazard and Decision Layers

This module uses the dense point cloud as input in order to evaluate the landing spot with respect to potential hazards such as terrain slope and terrain roughness. The data flow is presented in Fig. 7, for a sample visualization we refer to Fig. 10. To avoid the high memory load introduced by

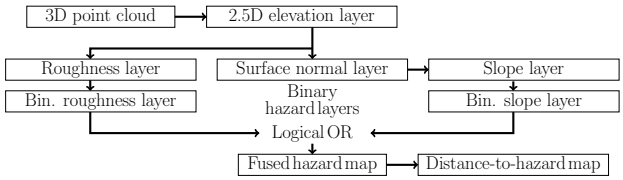


Fig. 7: Hazard and Decision Layers.

calculations involving the dense 3D point cloud, we convert to a 2.5D grid-based elevation map [34] (Fig. 10c). The elevation of each cell is computed using KD-tree based [35] IDW in a radius around the cell. From the 2.5D elevation layer, the surface normal in z-direction n_z of cell c_{ij} is computed based on the current cell and the 8-nearest neighbor cells using PCA. From the surface normal layer, the cell's slope α_{ij} with respect to the ground plane is obtained from $\alpha_{ij} = \arccos(n_{z,ij})$ (Fig. 10d). Terrain roughness is identified as a second hazard. The terrain ruggedness index (TRI) [15] is computed based on the elevation difference to the 8 adjacent cells and allows, for instance, to differentiate between flat grass, crops, or forest regions (Fig. 10e). A fine classification mask is computed as logical OR operation of all fine grass classification masks associated with this ROI. All hazard layers are only evaluated in the cells that have been classified as grass in at least one observation. Next, the hazard layers are transformed into binary layers using thresholds that are acceptable for the UAV (Fig. 10f). The binary hazard layers are then fused using the logical OR operation. In order to find safe and contiguous landing paths, we then apply the distance transform (Fig. 10g) to the fused binary hazard map. This yields, for every cell, the distance to the closest hazard. Further decision layers, such as a probabilistic point cloud or classification uncertainty layer, could easily be incorporated.

H. Landing Approach Vector

The question remains from which direction the landing spot is to be approached while circumventing the surrounding hazard(s). The local wind field, which is estimated in real-time by the on-board EKF, constrains the approach vector as illustrated in Fig. 8. Small-size fixed wing UAVs need to land against the wind direction in order to minimize the distance required for landing and to remain in a safe ground velocity region. Furthermore, we consider nearby obstacles obscuring the landing field based on the maximum descent rate of the UAV as well as obstacles in the landing region which are encoded in the distance map. Based on these considerations, the landing approach path is computed [1]:

$$x_{\text{app}} = \frac{v_{\text{land}}\cos(\gamma_{\text{land}}) - w\cos(\Delta\beta_w)}{v_{\text{land}}\sin(\gamma_{\text{land}})} h_{\text{app}}$$
$$R_{\text{loit}} = \frac{(v_{\text{land}}\cos(\gamma_{\text{land}}) + w)^2}{g\tan(\phi_{\text{land}})} \tag{1}$$

with

x_{TD} : touch down point	Δ_{TD} : touch down uncertainty (10 m)
w : wind magnitude (estimated)	v_{land} : airspeed ref. (13 m/s)
γ_{land} : flight path angle ref. (4 deg)	$\Delta\beta_w$: crosswind uncertainty (30 deg)
h_{app} : altitude approach (12 m)	ϕ_{land} : maximum bank angle ref. (11 deg),

and approach vector $\mathbf{x}_{\text{app}} = x_{\text{TD}} + \left[x_{\text{app}}\tilde{w}_x,\ x_{\text{app}}\tilde{w}_y,\ h_{\text{app}}\right]^\top$, where $\tilde{\mathbf{w}} = \left[\tilde{w}_x,\ \tilde{w}_y\right]^\top$ is the normalized 2D wind vector. The numbers in parentheses are sample values used for the research UAV *Techpod*. Based on the distance-to-hazard map we efficiently take the touch down uncertainty into consideration. In particular, starting from the safest touch down point, we check all cells traversed by the linear approach path and loiter-down circle for collision to obstacles based on the elevation map and incorporating a safety margin. Note that approach path optimization, in this context, is only used for a more meaningful scoring of potential landing sites and the provision of an informed approach vector. It should not be seen as a replacement for local re-planners which are still necessary for real-time corrections upon the actual landing attempt.

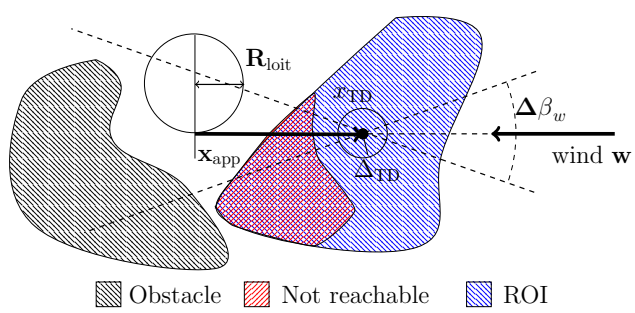


Fig. 8: Computation of the landing approach vector while considering the local wind field as well as hazards surrounding and within the landing region (ROI).

I. Wind Vector Estimation

The UAV's state estimator provides online estimates of the local wind field [16]. All measurements taken within a certain distance to the center of a ROI are associated with a landing spot as shown in Fig. 12, denoted by the black circle. To counteract slowly changing wind fields, we compute the final wind vector using the exponentially weighted moving average.

IV. Results

A. Region Classification

Fig. 9 shows the binary classification of homogenous regions into `grass` or `¬grass`. The results from random forest [30] and the multi-layer perceptron (MLP) based artificial neural network (ANN) [30] are plotted in form of the true positive rate (TPR) vs. the false positive rate (FPR) on the left (ROC chart) and the precision recall curve on the right. While ANN performs slightly better in the validation set, the measured computational cost to predict a binary label is $2.4e-3 \pm 7.8e-4$ ms for RF and $1.7e-2 \pm 9.0e-3$ ms for ANN. Since ROIs are tracked over several frames (cf. Sec. III-E) the influence of a single false prediction is mitigated by the probabilistic score as seen in Fig. 10 and Fig. 11. Hence RF was employed in all further experiments for computational speed-up.

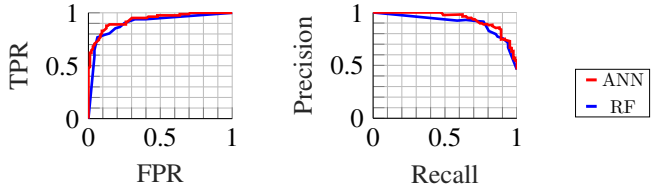


Fig. 9: Binary classification of homogenous regions into `grass` or `¬grass` for artificial neural network (ANN) and random forest (RF)

B. Computational Costs

The runtime evaluated on the real-world experiment "Switzerland" is shown in Table I. In the frontend, most of the time is spent on classification, in particular, to compute Gabor features. The frontend can run at 12.49 Hz with Gabor features and at 21.82 Hz when only relying on color cues. One could speed up the Gabor filter by only retrieving few samples from the image patch instead of using a convolution over the whole patch. However, since the incoming image rate is 4 Hz, the frontend (including Gabor features) is more than three times faster than real-time. Note the efficiency of the geometric region managing. BA, dense reconstruction and terrain analysis introduce, depending on the grid resolution, a certain delay and are available in *near real-time*.

	samples	mean ± stdev
Segmentation	250	7.20 ± 0.64
Class. w. Gabor	250	70.30 ± 22.76
- feature vector	2217	3.56 ± 4.95
- predict	2217	$2.4e{-3} \pm 7.8e{-4}$
Class. w/o Gabor	250	36.89 ± 20.28
- feature vector	2217	0.21 ± 0.27
- predict	2217	$2.1e{-3} \pm 8.4e{-4}$
Region Manager	250	1.57 ± 0.39
Bundle Adjustm.	23	20.1 ± 3.2
Dense Reconstr.	23	16.7 ± 2.3
Decision Layers	10	1083 ± 21.09
Approach Vector	10	527.48 ± 20.65

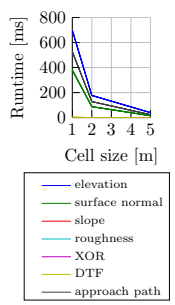


TABLE I: Runtime in ms for the "Switzerland" dataset. Note that the frontend (light gray) and backend (dark gray) run in separate threads. Runtime of frontend, BA, and dense reconstruction is measured per frame. Runtime of decision layers and approach vectors is computed for a ROI point cloud consisting of 4.2×10^6 points (300×300 cells, 1.0 m resolution). Evaluated on Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz.

C. Semi-Synthetic Dataset

The results obtained from a synthetic dataset are shown in Fig. 10. The images are rendered using *Blender* from poses computed by a simple lawn mower scan pattern generator. The underlying mesh was obtained from photogrammetry with images taken from a real camera, hence denoted as *semi*-synthetic. The overview mesh in Fig. 10 shows the scan pattern, corresponding frame indices on the left, and the three landing spots with the highest score. The right side of Fig. 10 shows the output of segmentation and classification for a sample frame. The second row of Fig. 10 plots the number of observations, classification certainty, score, and estimated area over time. Although there are some wrong classifications, the final classification certainty for all three regions is above 95 %. Fig. 10a to 10g show the backend, i.e. the dense reconstruction, decision, and hazard layers, the final touch down point, and linear approach path for the highest-scoring ROI #53. Note that the touch down points on the right side of Fig 10g show a high distance to the next hazards within the ROI but are rejected due to obstacles (house) along the planned linear approach path.

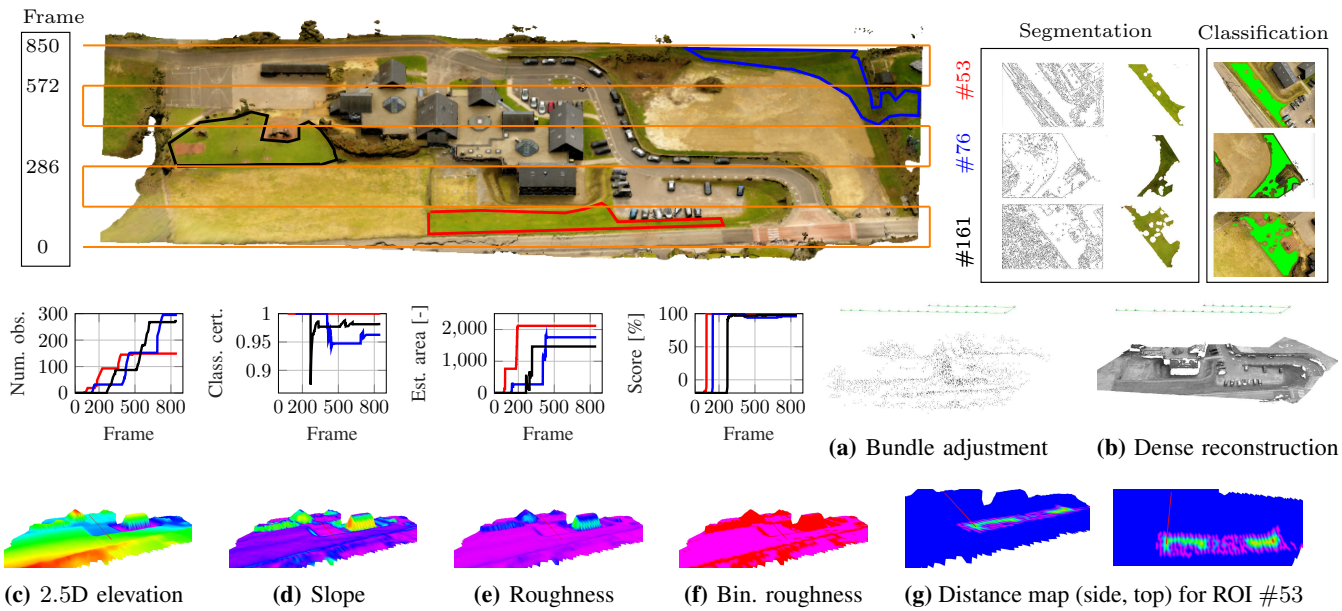
D. Experiments with Real-World Datasets

The real-world experiments are analysed using datasets recorded onboard of *AtlantikSolar* and *Techpod*. Details about the hardware setup and the employed platforms can be found in [1] and [36], respectively. The first experiment was conducted with the research platform *Techpod* in snowy scenery in Switzerland (cf. Fig. 11). Fig. 11b and 11e show the segmentation and classification of the landing region that received the highest score. The ROI is then forwarded to the backend thread which generates the decision layers based on a dense point cloud. The terrain slope and terrain roughness are used to compute the distance map (Fig. 11g) which encodes the distance to the next hazard in form of a memory-friendly grid map. From the score plots in Fig. 10 and 11 one can see that already the coarse grading can achieve a large separation between desired and undesired landing spots. Depending on the UAV characteristics and desired landing spot, the final ROIs can be compared based on the output of the fine landing site evaluation. In the next experiment, the distance to terrain elevation during the landing approach is given as an example for such a fine ROI output statistic.

This second real-world experiment was conducted with *AtlantikSolar* at the beach of Rio Pará, Brazil. Fig. 12a presents the overview mesh and camera poses for visualization, generated with *Pix4D*. Fig. 12b and 12c show the 2.5D elevation map, the estimated wind vector, and approach path to the selected landing site. The touch down point in the landing site is selected based on the maximum distance to nearby hazards while considering obscuring obstacles. The plots below show the path of the UAV with marked landing spot, wind speed measurements, and altitude profile during the approach path. For instance, the margin between UAV altitude and terrain elevation is predicted to drop to approx. 2 m, 35 m before touch down. As discussed in Section III-H, the algorithm is designed to land against the wind vector, here with a magnitude of ca. 5.5 m/s. This has the advantage of reducing the aircraft's forward ground velocity, allowing for shorter landing ground distances and thus increasing the perceived descent angle with respect to *ground*, yet still maintaining the chosen *airmass-relative* flight path angle γ_{land} (cf. Equation 1).

V. Conclusion

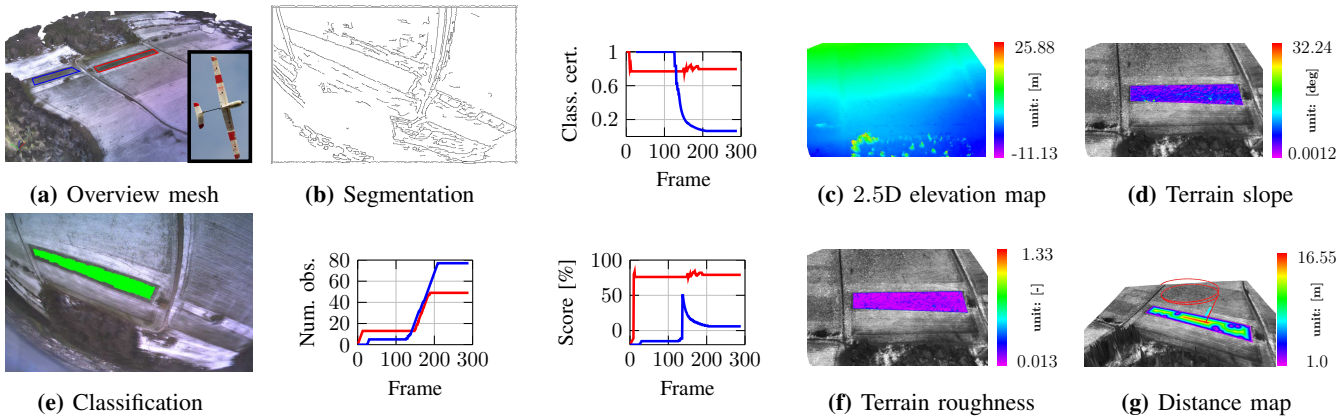
In this paper, we present a vision-based prior-free landing site detection algorithm which is designed for small UAVs, taking into account terrain texture, shape, roughness, and slope. The wind field, which is estimated online, and obscuring obstacles are taken into consideration when computing a suitable landing spot while regarding UAV dynamics and



(a) Bundle adjustment (b) Dense reconstruction

(c) 2.5D elevation (d) Slope (e) Roughness (f) Bin. roughness (g) Distance map (side, top) for ROI #53

Fig. 10: Semi-synthetic dataset illustrating the output of the segmentation, classification, tracking over time, and fine 3D terrain evaluation. The simulated camera is a down-looking *Aptina MT9v034* (0.36 MP). The (unscaled) mesh was downloaded from `https://skfb.ly/6o9Y7`.



(a) Overview mesh (b) Segmentation (c) 2.5D elevation map (d) Terrain slope

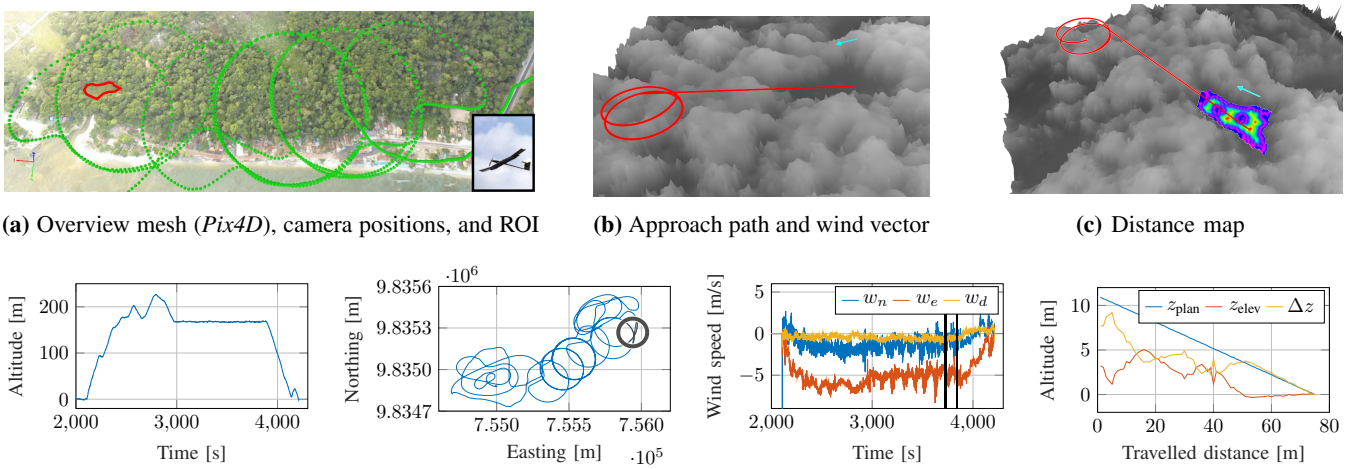
(e) Classification (f) Terrain roughness (g) Distance map

Fig. 11: Manual flight with *Techpod* in snowy scenery in Zurich, Switzerland. The employed camera is an *IDS UI-3241LE* (1.92 MP). The experiment underlines the performance in a challenging environment and with an obliquely mounted camera.

safety margins. To keep the problem complexity manageable, we segment the environment into regions and use a layered 2.5D grid map for decision making. The implemented multi-threaded framework combines a light-weight, real-time frontend with a backend which is periodically updated based on the host's resources. The linear approach path, which is one output of our method, can be tracked as demonstrated in [1]. The actual landing attempt should furthermore be supported by a perception system, local re-planners and low-level autopilot logic to avoid previously unmapped or moving obstacles. In this paper, a simplistic key-frame selection algorithm was employed. In a next step, an algorithm should be designed that guarantees complete coverage while minimizing the reconstruction uncertainty, utilized number of poses, and hence the computational costs. Inter-matches and free-space carving [37] could be incorporated into the reconstruction process. In future work, the classification and reconstruction uncertainty around a promising landing spot and approach path should be actively reduced by adapting the scanning pattern online and, in particular, by low-terrain flights to increase the ground resolution.

Acknowledgment

The research leading to these results has received funding from the *Federal office armasuisse Science and Technology* under project number n°050-45. The authors wish to thank Felix Renaut for an initial implementation of the presented frontend, Lucas Teixeira (Vision for Robotics Lab, ETH Zurich) for sharing scripts that bridge the gap between *Blender* and *Gazebo*, and Philipp Oettershagen.



(a) Overview mesh (*Pix4D*), camera positions, and ROI **(b)** Approach path and wind vector **(c)** Distance map

Fig. 12: Semi-automated flight at the beach of Rio Pará, Brazil, using *AtlantikSolar* and a down-looking *GoPro HERO3 Black* (12 MP): Takeoff and landing are performed in manual, the actual mission in automated GPS waypoint following mode. Subfig. (b) and (c) depict the 2.5D elevation map: The darker the pixel, the lower the height or z-value of the elevation map's cell. The plots illustrate the incorporation of wind estimates into the LSD framework: By landing against the wind vector, the required landing distance is drastically reduced.

References

[1] P. Oettershagen *et al.*, "Robotic technologies for solar-powered UAVs: Fully autonomous updraft-aware aerial sensing for multiday search-and-rescue missions," *Journal of Field Robotics*, 2017.

[2] V. R. Desaraju *et al.*, "Vision-based landing site evaluation and informed optimal trajectory generation toward autonomous rooftop landing," *Auton. Robots*, vol. 39, no. 3, pp. 445–463, 2015.

[3] C. Forster *et al.*, "Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles," in *Intern. Conf. on Robotics and Automation*, pp. 111–118, 2015.

[4] A. E. Johnson *et al.*, "Lidar-based hazard avoidance for safe landing on Mars," *J. of guidance, control, and dynamics*, vol. 25, no. 6, pp. 1091–1099, 2002.

[5] P. J. Garcia-Pardo *et al.*, "Towards vision-based safe landing for an autonomous helicopter," *Robotics and Autonomous Systems*, vol. 38, no. 1, pp. 19–29, 2002.

[6] R. Brockers *et al.*, "Autonomous landing and ingress of micro-air-vehicles in urban environments based on monocular vision," in *SPIE Defense, Security, and Sensing*, 2011.

[7] Y. Cheng, "Real-time surface slope estimation by homography alignment for spacecraft safe landing," in *Robotics and Automation, 2010 IEEE Intern. Conf. on*, pp. 2280–2286, IEEE, 2010.

[8] S. Bosch *et al.*, "Autonomous detection of safe landing areas for an UAV from monocular images," in *Intern. Conf. on Intelligent Robots and Systems, Beijing (China)*, 2006.

[9] C. Theodore *et al.*, "Flight trials of a rotorcraft unmanned aerial vehicle landing autonomously at unprepared sites," in *Annual Forum Proceedings-American Helicopter Society*, 2006.

[10] S. Huh *et al.*, *A Vision-Based Automatic Landing Method for Fixed-Wing UAVs*, pp. 217–231. Dordrecht: Springer Netherlands, 2010.

[11] M. Laiacker *et al.*, "Vision aided automatic landing system for fixed wing UAV," in *IROS*, pp. 2971–2976, IEEE, 2013.

[12] D. B. Barber *et al.*, "Autonomous landing of miniature aerial vehicles," *JACIC*, vol. 4, no. 5, pp. 770–784, 2007.

[13] S. Thurrowgood *et al.*, "A biologically inspired, vision-based guidance system for automatic landing of a fixed-wing aircraft," *J. of Field Robotics*, vol. 31, no. 4, pp. 699–727, 2014.

[14] D. Fitzgerald *et al.*, "A vision based forced landing site selection system for an autonomous UAV," in *Intelligent Sensors, Sensor Networks and Information Processing Conf.. Proc. of the 2005 Int. Conf. on*, pp. 397–402, IEEE, 2005.

[15] M. Warren *et al.*, *Enabling Aircraft Emergency Landings Using Active Visual Site Detection*, pp. 167–181. Springer Intern. Publishing, 2015.

[16] S. Leutenegger *et al.*, "Robust state estimation for small unmanned airplanes," in *2014 IEEE Conference on Control Applications (CCA)*, pp. 1003–1010, Oct 2014.

[17] P. Furgale *et al.*, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1280–1286, Nov 2013.

[18] B. D. Lucas *et al.*, "An iterative image registration technique with an application to stereo vision," in *In IJCAI81*, pp. 674–679, 1981.

[19] S. Weiss *et al.*, "Intuitive 3D Maps for MAV Terrain Exploration and Obstacle Avoidance.," *J. of Intelligent and Robotic Systems*, vol. 61, no. 1-4, pp. 473–493, 2011.

[20] J. Canny, "Finding Edges and Lines in Images," tech. rep., Cambridge, MA, USA, 1983.

[21] P. F. Felzenszwalb *et al.*, "Distance Transforms of Sampled Functions," *Theory of Computing*, vol. 8, no. 1, pp. 415–428, 2012.

[22] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.

[23] M. Agoston, *Computer Graphics and Geometric Modelling*. Computer Graphics and Geometric Modeling, Springer, 2005.

[24] J. R. Smith, *Color for Image Retrieval*, pp. 285–311. John Wiley & Sons, Inc., 2002.

[25] D. Gabor, "Theory of Communication," *J. IEE*, vol. 93, pp. 429–457, Nov. 1946.

[26] A. K. Jain *et al.*, "Unsupervised texture segmentation using Gabor filters," in *Intern. Conf. on Systems, Man, and Cybernetics Conference Proc.*, pp. 14–19, Nov 1990.

[27] M. Ghiasi *et al.*, "Fast semantic segmentation of aerial images based on color and texture," in *2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP)*, pp. 324–327, Sept 2013.

[28] M. Pietikinen *et al.*, "Color texture classification with color histograms and local binary patterns," in *In: IWTAS*, pp. 109–112, 2002.

[29] G. Toussaint, "Solving geometric problems with the rotating calipers," 1983.

[30] Itseez, "Open Source Computer Vision Library." `https://github.com/itseez/opencv`, 2015.

[31] K. Weiler, "An incremental angle point in polygon test," in *Graphics Gems IV* (P. S. Heckbert, ed.), pp. 16–23, Morgan Kaufmann, 1994.

[32] A. Jäger, "Real-Time Monocular Dense 3D Reconstruction for Robot Navigation," Master's thesis, ETH Zurich, 2015.

[33] A. Fusiello *et al.*, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.

[34] P. Fankhauser *et al.*, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) The Complete Reference*, vol. 1, Springer, 2016.

[35] J. L. Blanco *et al.*, "nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) wih KD-trees." `https://github.com/jlblancoc/nanoflann`, 2014.

[36] T. Hinzmann *et al.*, "Robust Map Generation for Fixed-Wing UAVs with Low-Cost Highly-Oblique Monocular Cameras," in *Intern. Conf. on Intelligent Robots and Systems*, October 2016.

[37] A. Hornung *et al.*, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Autonomous Robots*, 2013. Software available at `http://octomap.github.com`.