Superpixel Analysis for Object Detection and Tracking with Application to UAV Imagery

Christopher Rasmussen

Dept. Computer & Information Sciences
University of Delaware
cer@cis.udel.edu

Abstract. We introduce a framework for object detection and tracking in video of natural outdoor scenes based on fast per-frame segmentations using Felzenszwalb's graph-based algorithm. Region boundaries obtained at multiple scales are first temporally filtered to detect stable structures to be considered as object hypotheses. Depending on object type, these are then classified using *a priori* appearance characteristics such as color and texture and geometric attributes derived from the Hough transform. We describe preliminary results on image sequences taken from low-flying aircraft in which object categories are relevant to UAVs, consisting of sky, ground, and navigationally-useful ground features such as roads and pipelines.

1 Introduction

The problem of segmenting an image into semantically meaningful units is a fundamental one in image processing. Watersheds [1], mean-shift clustering [2], and graph theoretic [3–6] techniques which group pixels based on proximity, color, texture, and other Gestalt characteristics are powerful tools, but none are silver bullets. Besides practical issues of running time or the necessity of some user interaction [4], the key issue is what objects, if any, correspond to the segmented regions (aka *superpixels*). Given the frequency of oversegmentation, in which objects are broken in multiple superpixels, or the opposite phenomenon, a realistic hope of resolving this issue is only possible in constrained scenarios in which much is known *a priori* about the objects in the scene. A sample of recent work tackling superpixel grouping and classification includes the domains of human body pose recognition [7], categorizing scene surface layout [8], and finding trees, vehicles, and buildings in aerial images [9].

In this paper we take inspiration from such work in proposing a multi-stage framework for object detection and tracking that is based on the output of the graph-based image segmentation algorithm of Felzenszwalb and Huttenlocher [6] (aka the *FH segmenter*). One major advantage of the FH segmenter is its speed: on an Intel Core2 Extreme X6800, it segments 360×240 color images at over 30 Hz, compared to several minutes per image using spectral graph cut methods

[3, 5]. This makes it practical to use for real-time video. Thus, a chief novelty of our approach is that we can analyze superpixels over time for *consistency* according to the intuition that temporally transient borders are likely spurious. Candidate superpixels that pass this restrictive test are further examined using region characteristics such as color, texture, and shape to see if they match any of the objects we are searching for.

The task area we apply the framework to here is navigation and control for unmanned aerial vehicles (UAVs) flying at very low altitudes (under 500 m). There are many possible image processing tasks for UAVs; we focus on horizon detection for attitude determination and recognizing ground features for navigation. At such low altitudes, terrain relief means that the horizon may be quite jagged (see Fig. 1) rather than straight as most work assumes [10–12]. Raw segmentations of mountainous images are shown in the classification approach of [13], but there is no discussion of explicit recovery of the boundary curve in a single image or tracking it over time.

Other object categories of interest besides sky and ground are navigationally-useful linear features along the ground, which we term *trails*. These include engineered artifacts such as dirt and paved roads and above-ground pipelines, but also natural phenomena like rivers and streams. While vision-based road segmentation is a well-studied topic for unmanned ground vehicles [14–16], the relatively large size and stable centroid of the road region in their camera view allows a robust approach to shape estimation from edges and easy sampling of on-road pixels that is not possible from a low-altitude UAV perspective. From the air, trail pixels do not dominate the image, making mere detection of the trail among many other features a difficult aspect of the problem. Also, the camera platform is often anything but stable due to wind gusts and pitching and rolling as the UAV maneuvers, causing the horizon and trail to go in and out of view. Unfortunately, because of the wide range of shapes and appearance characteristics possible, aerial trail finding is not amenable to traditional machine learning-based object detection methods such as the face- and person-finders of [17, 18].

Image processing techniques for road or pipeline tracing in higher altitude imagery [19–21] also do not carry over directly. The low-altitude UAV perspective is unlikely to be straight down, leading to perspective effects like non-parallel road edges. On the positive side, though, enough pixels can typically be resolved within the trail region to gather detailed appearance statistics. Examples of existing techniques that are more tracing-oriented than region-based include [22], which demonstrates tracking of a short, straight runway with relatively high contrast from a UAV, and [23] from the same authors, which tracks video of a slightly curving canal. The tracker in the latter paper initializes automatically under the assumption that the ground feature is homogeneous, only gently curved, and occupies a fairly large fraction of the image. [13] has some results on finding straight roads in low-altitude aerial images using a Hough transform on static images only, with no tracking.

The organization of the paper is as follows. First, we detail our methods for discriminating between the sky and ground objects reliably and tracking the border between them. Second, we discuss preliminary results on extending our superpixel analysis method to trail detection for several trail types. Finally, we present a separate section of results processed offline on three diverse sequences: a winding canyon road in the desert, a flyover of a section of the Alaska pipeline, and a helicopter flight along a mountain stream.

2 Sky/Ground Discrimination

The FH segmenter works by greedily splitting regions when the intensity difference along the candidate child regions' border exceeds their internal intensity variations by a certain threshold (color image segmentations result from merging the segmentations of their channels). Two key parameters govern the operation of the segmenter: k, which sets the threshold for splitting, with larger k leading to a preference for the retention of larger regions; and min, which sets an absolute minimum on the area of any region that is output.

Applying Felzenszwalb's code for the FH segmenter to the canyon sequence images in the first row of Figure 1 with standard parameters $k = 100$ and min $= 100$ (explanation below) yields the segmentations shown in the second row of the figure. The colors of the segmentation are randomly generated to show superpixel memberships. While the sky-ground border is clearly visible, it is but one of many edges generated by this oversegmentation. Because the object we are looking for, sky, is expected to be a large region, we can obtain a cleaner segmentation with a different set of FH segmenter parameters. We bias the segmenter to find much larger regions by changing k to 500 and setting min $= 5\%$ of the area of the image. This results in the segmentations shown in the third row of the figure. Over all of our data, these parameters yield 2-5 superpixels in almost all images of the canyon sequence. The sky-ground border is generally very stable, with only transient "extra" regions due mostly to cloud formations, hazy distant terrain, and intermittent video transmission interference when the capture device was not onboard the aircraft (as seen in the right column of Figure 1).

We were able to temporally filter these superpixel borders directly to get quite good results for the sky-ground dividing line, but because of the mountainous terrain in the canyon sequence and the steep banking of the UAV, it is not always clear from geometry alone which side of the line is sky and which is ground. It is thus desirable to differentiate the two based on region characteristics such as color. There are many options here, but we chose to use the *surface layout* classifier of [8, 24]. This method divides images into geometric classes corresponding to three kinds of surfaces: *support* (aka ground), *vertical*, or *sky*. Within the vertical class are several subclasses corresponding to different normal directions if the surface is planar and the designations *solid* or *porous* if it is not. A particular pixel is labeled by a learned classifier using features such as as image location, color, texture, and any association with vanishing points.

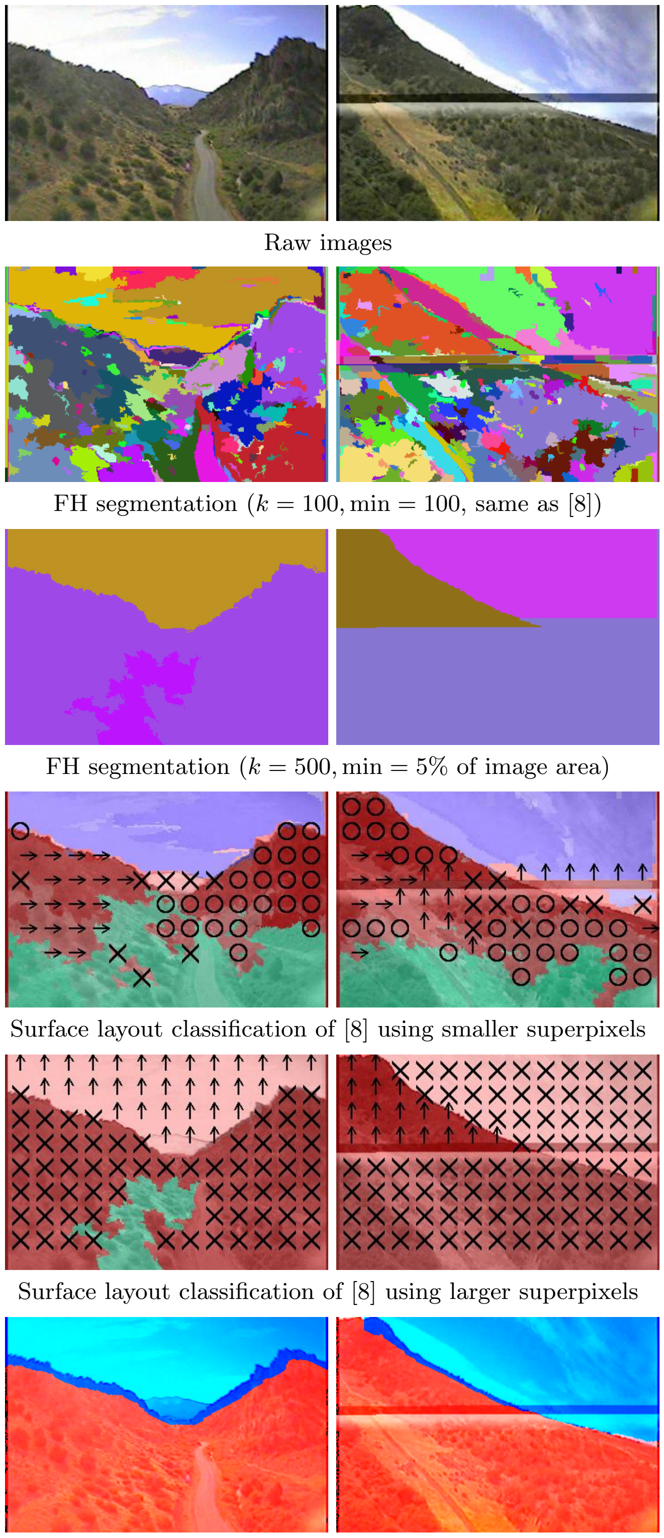
Raw images
FH segmentation ($k = 100, \min = 100$, same as [8])
FH segmentation ($k = 500, \min = 5\%$ of image area)
Surface layout classification of [8] using smaller superpixels
Surface layout classification of [8] using larger superpixels
Our filtered sky segmentation (after dilation)

Fig. 1. Sky-ground object detection steps for two images. The horizontal bar in the right image is a wireless video transmission artifact

An important aspect of the classification process is that pixels are not labeled independently, but rather in groups by superpixel. They use the FH segmenter with the parameters $k = 100$ and $\min = 100$ for all of their results, which is why we show what is obtained with these parameters in the second row of Figure 1).

Feeding these to the author-provided implementation of the scene layout algorithm (the later version from [24]) with the outdoor-trained version of their classifier yields the classifications shown in the third row of the figure. The scene layout labels are indicated by tinting: blue for sky, red for vertical, and green for support. Subclasses within vertical are indicated by overlaid icons: "O" for porous, "X" for solid, or an arrow for the normal direction of planar surfaces. One can see that the support and vertical class labels, as well as vertical's subclass labels, are mostly non-informative. The sky label is only reasonably accurate, with a sample from the entire sequence indicating that nontrivial fractions of the sky are mislabeled in over 50% of the frames.

We were able to significantly improve the accuracy of the scene layout sky labels by rerunning the segmenter with the parameters that yield larger regions. The new scene layout labels with these parameters are shown in row four of the figure. The erroneous results shown are not representative: the scene layout classifier actually does quite well overall. A manual tally of the entire canyon sequence (2500 frames) shows that a majority of sky pixels are mislabeled in only 9% of the images—most commonly as vertical—with a non-zero minority of the sky mislabeled in about 2%. Practically none of the ground pixels are mislabeled as sky. These numbers agree quite well with the confusion matrix in [24].

Our point in showing error examples is to motivate our use of temporal filtering: simply trusting a single-image classifier, however good, will occasionally result in nonsensical output. We know more: the sky region does not disappear and reappear from frame to frame—it has continuity of appearance and shape.

Before describing our tracking framework, we should mention that we also modified how the scene layout classifier's output is interpreted. We found that misclassifications of the sky as vertical most often resulted from random fluctuations in the two class likelihoods when they were nearly the same magnitude. We observed that the sky likelihood distribution was quite bimodal, and that instead of choosing the class with the highest likelihood for a superpixel, setting a low threshold on the sky likelihood for classification as simply sky or not sky eliminated almost all such errors.

2.1 Temporal filtering

Going beyond performing segmentations and scene layout classifications *de novo* on each frame is fairly straightforward. Let $\mathbf{B}_t$ be the sky-ground boundary obtained after the steps of the previous subsection for frame t. $\mathbf{B}_t$ is simply represented as a set of points along the border of the sky region (after some smoothing and cropping to eliminate image edge effects and capture artifacts). There are no restrictions on connectivity or indeed any attempt made to parametrize the curve.

We want to mitigate the effects of transient classification errors and segmentation artifacts by temporal smoothing. We choose a majority filter: given n successively segmented borders $\mathbf{B}_{t-n+1}, \ldots, \mathbf{B}_t$ registered with one another (i.e., motion stabilized), an aberrant border segment or even an entire missing border can be detected by treating each $\mathbf{B}_t$ as voting for a particular hypothesis and only accepting those that achieve consensus. We do this by spreading each border out with a linear falloff, summing, and thresholding.

In order to register each sky-ground boundary $\mathbf{B}_t$ with its predecessor $\mathbf{B}_{t-1}$, we assume that a 2-D rigid transformation is a good approximation of horizon movement over a short time scale (in fact, we found that translation alone is good enough). RANSAC [25] does quite well at registering successive curves in the face of possible large outliers. Over the long term, nonlinear deformation occurs as new mountains come into view; this is handled implicitly with the finite "memory" of the border voter buffer, with $n = 5$.

Example filtered sky-ground regions are shown in the last row of Figure 1). Some small misregistrations remain because of the implicit lag of the majority filter.

3 Trail Detection

Besides the control benefits of horizon orientation information for a UAV, finding the sky region is also critical for successful trail detection. First, since we know the trail is on the ground, sky pixels can be masked out, reducing the image area to search and helping efficiency. Second, accuracy is also improved: since the horizon line is often the highest-strength edge in the image, removing it makes voting techniques for identifying trail edges less susceptible to distraction.

We make several simplifying assumptions. First we assume that there is a trail visible over nearly all of the image sequence—we are not trying to decide *whether* there is a trail or not, only to find and parametrize it in a way that could be passed to a flight control system. This does allow for the trail to briefly disappear from view as the aircraft maneuvers. A second assumption is that there is only one trail visible over the sequence. Parallel roads, forks, and intersections are not accounted for, as they introduce ambiguity about which trail is to be followed. Finally, we assume that the off-trail terrain is *natural* rather than urban, as our primary means of recognizing trails is through the rarity of the structural regularity associated with them.

Our approach to trail detection is similar to that for sky segmentation: first the image is segmented into superpixels, of which we expect trail edges to be relatively stable. Applying a region classifier along the lines of the scene layout method described above is problematic because trails have much more appearance variation than sky and we do not know *a priori* which kind of trail we are tracking. Rather, we narrow the candidates using a geometric test based on the intuition that trails often have linear or smoothly curving borders, unlike the natural background that occupies the rest of the image. Surprisingly, even the river trail shown in Figure 4, which has somewhat a stochastic boundary shape, fits this profile for much of its length.

This test currently consists of a Hough transform applied to the superpixel borders (using a different set of segmentation parameters $k = 200$, min $= 1000$). The top line candidates associated with the trail should cluster by vanishing point, which is trackable with filtering. Using the superpixel borders rather than low-level edge features such as Canny or gradient magnitude allows more sophisticated grouping information such as color and texture to play a role, if at some cost of smoothness.

4 Results

Our results are summarized with sample object detections on all three sequences in Figure 4. Each sequence exhibits artifacts of some kind: the canyon sequence has a fair degree of video transmitter interference, and the pipeline and river sequences have significant compression artifacts. Nonetheless, the algorithm presented here performs extremely well on all three at sky-ground border tracking. Trail detection results are somewhat preliminary, as further filtering could be done to counter noise in the segmentation, but still quite promising, especially for the pipeline sequence. Counter to one of our assumptions above, there is ambiguity in the pipeline sequence about what the trail is: the pipe itself, the road, or both? This is reflected in the trail object finder shifting around a bit between these alternative interpretations.

We note that the sky-ground tracker's ability to handle the disappearance and reappearance of the horizon is shown in the pipeline images. Also seen in the river sequence is the phenomenon of flipping back and forth between sky and ground classifications for distant mountains, which through haze have a similar color to the sky. The filtering mechanism smooths these transitions.

5 Conclusion

We have demonstrated a promising approach to object detection and tracking in video that is based on appearance and geometric analysis of quickly-segmented superpixels. The competence of the technique was shown through application to several realistic UAV tasks and over diverse image sequences.

Our current work focuses on the trail detection component. While the linear Hough transform is a reasonable start, further refinement to estimate trail curvature and smoothness is necessary. Rather than approximating the trail boundaries with parametric curves as we currently do, we are investigating how to extract irregular boundaries directly from the segmentation.

References

1. L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, 1991.

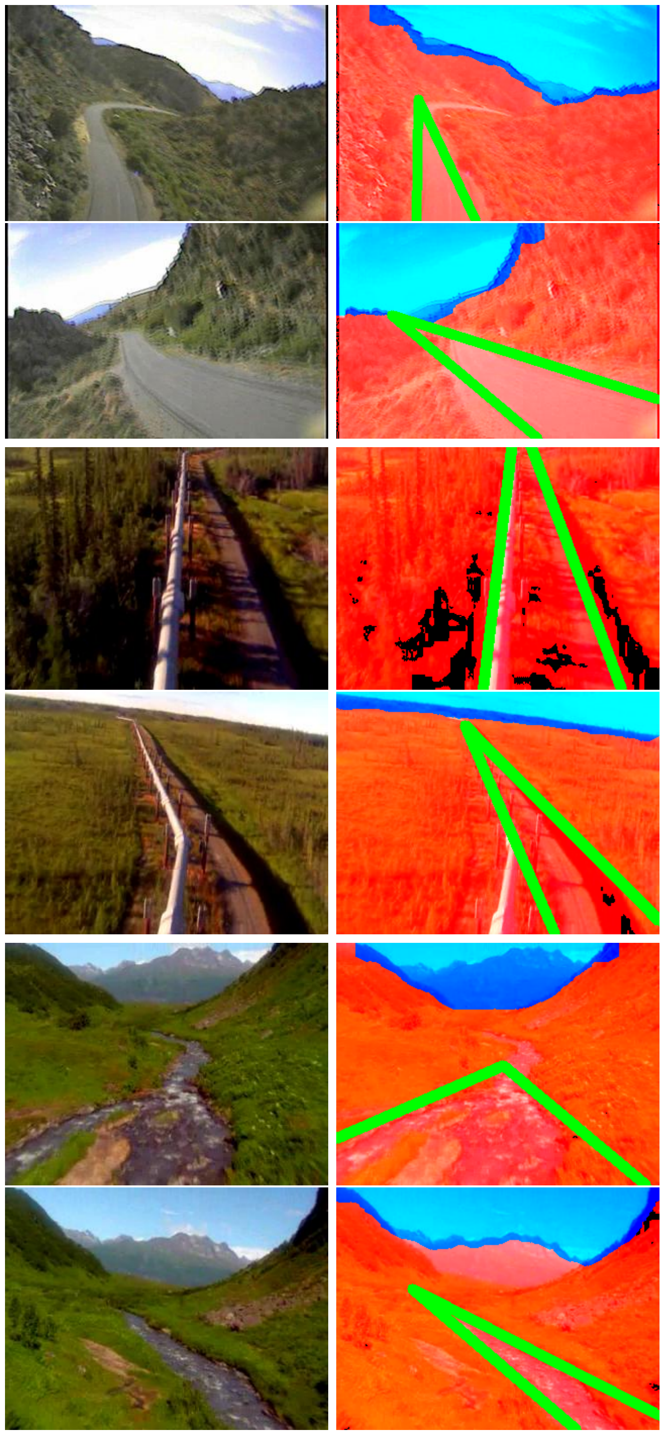

Fig. 2. Sample object detections. The envelopes of trail detections are demarcated in green; dark spots in the ground regions are compression artifacts

2. D. Comaniciu and P. Meer, “Mean shift analysis and applications,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1999.
3. J. Shi and J. Malik, “Normalized cuts and image segmentation,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997.
4. Y. Boykov and M. Jolly, “Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images,” in *Proc. Int. Conf. Computer Vision*, 2001.
5. C. Fowlkes, D. Martin, and J. Malik, “Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.
6. P. Felzenszwalb and D. Huttenlocher, “Efficient graph-based image segmentation,” in *Int. J. Computer Vision*, 2004.
7. G. Mori, X. Ren, A. Efros, and J. Malik, “Recovering human body configurations: Combining segmentation and recognition,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004.
8. D. Hoiem, A. Efros, and M. Hebert, “Geometric context from a single image,” in *Proc. Int. Conf. Computer Vision*, 2005.
9. J. Kaufhold, R. Collins, A. Hoogs, and P. Rondot, “Recognition and segmentation of scene content using region-based classification,” in *Proc. Int. Conf. Pattern Recognition*, 2006.
10. S. Ettinger, M. Nechyba, P. Ifju, and M. Waszak, “Vision-guided flight stability and control for micro air vehicles,” in *Proc. Int. Conf. Intelligent Robots and Systems*, 2002.
11. T. McGee, R. Sengupta, and K. Hedrick, “Obstacle detection for small autonomous aircraft using sky segmentation,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 2005.
12. T. Cornall, G. Egan, and A. Price, “Aircraft attitude estimation from horizon video,” *IEE Electronics Letters*, vol. 42, no. 13, 2006.
13. S. Todorovic and M. Nechyba, “A vision system for intelligent mission profiles of micro air vehicles,” *IEEE Trans. Vehicular Technology*, vol. 53, no. 6, 2004.
14. G. Stein, O. Mano, and A. Shashua, “A robust method for computing vehicle ego-motion,” in *Proc. IEEE Intelligent Vehicles Symposium*, 2000.
15. B. Southall and C. Taylor, “Stochastic road shape estimation,” in *Proc. Int. Conf. Computer Vision*, 2001, pp. 205–212.
16. C. Rasmussen, “A Hybrid Vision + Ladar Rural Road Follower,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 2006.
17. P. Viola and M. Jones, “Robust real-time object detection,” in *Int. Workshop on Statistical & Computational Theories of Vision*, 2001.
18. P. Viola and M. Jones, “Detecting pedestrians using patterns of motion and appearance,” in *Proc. Int. Conf. Computer Vision*, 2003.
19. D. Geman and B. Jedynak, “An active testing model for tracking roads from satellite images,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, 1996.
20. A. Yuille and J. Coughlan, “Fundamental limits of Bayesian inference: Order parameters and phase transitions for road tracking,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 2, pp. 160–173, 2000.
21. P. Perez, A. Blake, and M. Gangnet, “Jetstream: Probabilistic contour extraction with particles,” in *Proc. Int. Conf. Computer Vision*, 2001, pp. 524–531.
22. E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta, “Vision-based road following using a small autonomous aircraft,” in *IEEE Aerospace Conference*, 2004.

23. S. Rathinam, Z. Kim, A. Soghikian, and R. Sengupta, “Vision based following of locally linear structures using an unmanned aerial vehicle,” in *IEEE Conference on Decision and Control*, 2005.
24. D. Hoiem, A. Efros, and M. Hebert, “Recovering surface layout from an image,” *To appear in Int. J. Computer Vision*, 2007.
25. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.