5-2011

Self-Supervised Segmentation of River Scenes

Supreeth Achar  
*Carnegie Mellon University*

Bharath Sankaran  
*University of Pennsylvania*

Stephen Nuske  
*Carnegie Mellon University*

Sebastian Scherer  
*Carnegie Mellon University*

Sanjiv Singh  
*Carnegie Mellon University*

Follow this and additional works at: [http://repository.cmu.edu/robotics](http://repository.cmu.edu/robotics)  
Part of the [Robotics Commons](http://repository.cmu.edu/robotics)

Published In  

This Conference Proceeding is brought to you for free and open access by the School of Computer Science at Research Showcase @ CMU. It has been accepted for inclusion in Robotics Institute by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.
Abstract—Here we consider the problem of automatically segmenting images taken from a boat or low-flying aircraft. Such a capability is important for autonomous river following and mapping. The need for accurate segmentation in a wide variety of riverine environments challenges the state of the art vision-based methods that have been used in more structured environments such as roads and highways. Apart from the lack of structure, the principal difficulty is the large spatial and temporal variations in the appearance of water in the presence of nearby vegetation and with reflections from the sky. We propose a self-supervised method to segment images into ‘sky’, ‘river’ and ‘shore’ (vegetation + structures) regions. Our approach uses assumptions about river scene structure to learn appearance models based on features like color, texture and image location which are used to segment the image. We validated our algorithm by testing on four datasets captured under varying conditions on different rivers. Our self-supervised algorithm had higher accuracy rates than a supervised alternative, often significantly more accurate, and does not need to be retrained to work under different conditions.

I. INTRODUCTION

We are interested in a minimal sensor suite based on passive vision and inertial sensing that could be used to autonomously explore rivers, mapping their width as it proceeds. Given sensor pose and camera calibration segmented images can be turned into river maps. In some cases, the canopy can be so thick and high around a river so as to block GPS signals and the problem becomes one of simultaneous localization and mapping. In this paper, we focus on the key subproblem of automatically segmenting images so that the extents of the river can be found accurately. Since rivers can look like roads, one idea is to use fairly well developed methods that have been used to track roads and highways with passive vision. Apart from the lack of structure such as clearly defined road edges, the principal difficulty of vision-based tracking of rivers has to do with large spatial and temporal variations in appearance of water in the presence of nearby vegetation and with reflections from the sky. See for example, Fig. 1 and Fig. 2.

Such problems are often solved with supervised learning, an approach where a system is trained based on representative data ahead of time. In our case, supervised learning would require that we train our river detector with instances of image regions that correspond to water in rivers and then use the trained system online. Given the lack of a representation that would render the appearance of rivers as a constant under varying conditions, we find that it is necessary that any learning be self-supervised. That is, the system must train itself, adapting to local appearance as influenced by terrain, flora and lighting conditions. A natural question arises in being able to develop such a self-supervised method—whence the representation? There is a common intuition that features of color, texture and vertical location in an image might help. Still, at the outset it is not clear how to encode/combine different features.

Here we propose a two step self-supervised method to perform such segmentation. In the first step we use a simple heuristic (knowledge of the horizon line) to automatically determine the relative correlation between features (a grab bag of color and texture based features). This allows us to score each image patch on its likelihood of belonging to a river. Patches with high confidence labels are used to train a Support Vector Machine. In the second step, the output of the SVM segments the image into distinct regions. The

Fig. 1. An example image illustrating the variation of river appearance within a single image.

Fig. 2. This image was taken from almost the same location as Fig. 1 at a different time of day. The appearance of the river has changed dramatically.
advantage of such an approach is that it assumes no prior to 
determine segmentation other than simple geometric assump-
tions such as that the river lies below the horizon, and, that
river features are significantly different than other features 
below the horizon. All other training happens automatically 
as often as at every frame.

We have tested our algorithms with image sequences from 
two different rivers, the image sequences have different envi-
ronmental conditions; sunny and dusk, summer and fall. We 
show how supervised learning by training with hand labeled 
instances of segmentation, provides a reasonable solution but
does not generalize well to novel environments. We show how performance and adaptability to novel environments can 
be improved with a self-supervised algorithm.

II. RELATED WORK

Work related to the river segmentation algorithm presented 
in this paper falls into three main categories: techniques for 
inferring geometry from appearance, road following methods 
and water hazard detection systems for autonomous ground 
vehicles.

Make3D [1] and Photo Popup [2] infer information about 
scene structure from appearance. These methods work well 
on a wide variety of images but tend to have problems 
dealing with reflections and shadows which are common 
in riverine environments. Also, both these methods are 
computationally intensive which precludes their use in time 
sensitive applications.

Our work is similar in purpose to road following algo-

was proposed that used a laser scanner to find a small 
region of road directly in front of the vehicle which was 
used to build a road appearance model and classify the 
rest of the image. The underlying assumption is that road 
appearance is fairly uniform, but as is clear in Fig. 1 river 
appearance is inhomogeneous so an appearance model learnt 
on a small patch near the bottom of an image would not be 
representative of appearance of the entire river region.

There has also been work in detecting water hazards for 
a ground plane (river plane) and exploits the knowledge of 
the horizon to create self-supervised training sets from the 
current image.

III. FEATURE SELECTION

To enable effective river segmentation a discriminative 
feature vector \(X \in \mathbb{R}^3\) for describing local appearance 
is required. Color is a useful feature for detecting water as 
demonstrated in [7] & [8]. Texture is another useful cue, 
for example, parts of an image containing the river tend to 
be less textured than regions of foliage. Position in an image 
provides a strong prior, the higher up in an image a region 
is the less likely it is to be part of the river.

We selected features by empirically measuring the per-
formance of different feature vectors describing color and 
texture in a supervised learning setup. Each candidate feature 
vector was used to train a two class (river and shoreline) 
linear support vector machine over a set of training images 
and then tested by using the SVM to segment the river 
in images from a small evaluation dataset. Performance 
of a feature vector was quantified by the percentage of 
correctly labeled pixels averaged over the evaluation dataset.

All features were calculated over \(5 \times 5\) pixel patches.

To choose a color descriptor for the feature vector we 
used the RGB, Lab and HSV color spaces individually and 
in various combinations to train classifiers. The performance 
of these classifiers on labeling river regions in the evaluation 
dataset is shown in Table I. A combination of RGB and Lab 
colorspaces gave the best performance and was selected as 
our color descriptor. Although both RGB and LAB encode 
the same color information, combining them allows a linear 
classifier to learn a more complex decision boundary.

We evaluated the performance of three texture descriptors 
using a similar methodology. The first was the response to 
a set of Laws’ masks [10]. We used 8 of the \(9 \times 3\) Laws’ 
Masks leaving out the mask that performs low pass filtering. 
Responses to each of these filters was calculated over 4 
scales on the 3 channels of the Lab image, so the resulting 
texture descriptor had length \(4 \times 3 \times 8 = 96\). The second 
texture descriptor considered was the unnormalized DAISY 
descritor [11]. The third alternative was a bank of Gabor 
filters with 4 scales and 6 orientations. For all the texture 
descriptors, filter responses were normalized with respect 
to intensity. All three filter sets performed almost equally 
(Table I) so we chose the Laws’ masks because it was the 
fastest to compute.

When combined, color and texture features perform better 
than either cue alone. Image position, specifically the height 
of a region in image coordinates, provides a strong contextual 
cue for segmentation and is included in the feature vector. 
The bottom of Table I shows the labeling error rate for the 
combined color and texture descriptor and for our final 
choice of feature descriptor with a fully supervised classifier 
on the images in the evaluation dataset.

IV. SELF-SUPERVISED RIVER SEGMENTATION

Riverine environments vary widely in appearance which 
makes building a single classifier that works well in different
conditions difficult. Such a classifier would require labeled training images captured in many environments under different conditions. Its performance in an environment dissimilar to those seen earlier would not be assured. Instead of attempting to learn offline a universal model of river appearance, we automatically learn a new appearance model online for each new input image.

To do this we utilize knowledge about the position of the horizon in each image and assume that anything appearing above the horizon line is not part of the river. The assumption that the river can not be above the horizon is valid except for situations in which the river has a significant upward slope (these are unusual scenarios in which we do not expect to operate like upstream on a rapid or at a waterfall). In our application domain, the inertial measurement unit on the vehicle would be used to estimate the horizon. Another assumption we make is that the appearance of the above horizon regions is fairly representative of the appearance of the entire shore area in that image.

As a preprocessing step, we first segment out the sky and ignore it during all further computations. To build an online appearance model for discriminating between river and shore regions, we extract two types of patches from the image, those that are part of the shore region and those that are likely to be part of the river. Generating shore patches is trivial, we just sample patches lying above the horizon. We find the patches below the horizon line that are most dissimilar in appearance to those above the horizon and use them as candidate river patches. We then use these patches to train a linear SVM which is then used to classify all parts of the image. Details of each step in the algorithm are given below.

### A. Feature Extraction

As described in the Section III, the image is divided into square patches and a feature vector \( X \in \mathbb{R}^n \) is computed for each patch which describes its color, texture and position.

### B. Sky Detection

A linear support vector machine trained over a set of labeled images is used to detect the sky in each image. The sky is relatively easy to segment out with a globally trained classifier as its appearance is not as variable as that of the river. Discarding the sky before further processing reduces computation and prevents confusion with shiny, mirror like parts of the water’s surface which often appear similar to the sky. Fig. 3(d) shows an example of sky detection. The remainder of the image needs to be classified as either being part of the river or part of the shore.
C. Appearance Modeling

The goal of the appearance modeler is to assign to each patch a probability of being part of the river (R) or being part of the shore (¬R) on the basis of the feature vector (X) that describes it. By Bayes’ Rule we have

\[ P(¬R | X) = \frac{P(X | ¬R)P(¬R)}{P(X)} \]  

Since the labels (R and ¬R) are what we are trying to determine, \( P(X | ¬R) \) can not be calculated directly. We define a boolean variable \( H \) which is true for regions below the horizon and false for regions above. The value of \( H \) at each point in the image is known because the horizon is known. We assume that the appearance of the shore region above the horizon is representative of the entire shore region in the image or \( P(X | ¬R) \approx P(X | ¬H) \) which gives

\[ P(¬R | X) \approx \frac{P(X | ¬H)P(¬R)}{P(X)} \]  

\[ = \frac{P(X | ¬H)P(¬R)}{P(X | ¬H)P(¬H) + P(X | H)P(H)} \]  

\( P(¬H) \) and \( P(H) \) are determined from the relative sizes of the above and below horizon regions in the image. \( P(¬R) \) which we arbitrarily set to 0.5 is the prior probability of a patch being part of the shore region. The appearance distributions \( P(X | ¬H) \) and \( P(X | H) \) are modeled using Chow Liu trees [12]. A Chow Liu tree is a method for approximating the joint probability distribution of a set of discrete random variables by factoring it into a product of second order distributions. A Chow Liu tree is optimal in the sense that it is the distribution of its class that minimizes the KL divergence to the real distribution.

The feature vector \( X \) contains color, texture and image position information and has high dimensionality as it contains many texture filter responses. It is computationally intractable to use the full length feature vector at this point because of the way Chow Liu modeling scales with dimensionality. Therefore, we use an abridged feature vector \( \tilde{X} \in \mathbb{R}^d \) in which the texture descriptor subvector is replaced by its \( L_2 \) norm. Furthermore, we do not include the image position in \( \tilde{X} \) because we do not want to model the effect of image position on appearance at this stage.

The features in feature vector \( \tilde{X} \) are continuous valued while Chow Liu trees work over discrete valued random variables. To solve this problem, each \( \tilde{X} \in \mathbb{R}^d \) is converted into a discretized feature vector \( \tilde{X} \in \mathbb{S}^d \). Each feature in \( \tilde{X} \) is assigned to one of 16 levels (\( \mathbb{S} = \{0, 1, 2, \ldots, 15\} \)) using equally sized bins that span the range of values taken by that feature.

Two Chow trees are built, one using the feature vectors of patches above the horizon (\( \tilde{X} \notin H \)) to model \( P(X | ¬H) \) and another using the feature vectors of below horizon patches (\( \tilde{X} \in H \)) to model \( P(X | H) \). We use subscripts to denote individual features in a feature vector so \( \tilde{X}_i \) is the \( i \)th feature in feature vector \( \tilde{X} \). A Chow Liu tree is a maximal spanning tree of the mutual information graph. The mutual information graph has one node for each feature (\( \tilde{X}_i \)) and an edge between every pair of nodes (\( \tilde{X}_i \) and \( \tilde{X}_j \)) which is weighted by their mutual information \( I(\tilde{X}_i; \tilde{X}_j) \).

\[ I(\tilde{X}_i; \tilde{X}_j) = \sum_{x_i=0}^{1} \sum_{x_j=0}^{1} p(x_i,x_j) \log \frac{p(x_i|x_j)}{p(x_i)p(x_j)} \]  

Where \( p(x_i) \) is \( P(\tilde{X}_i = x_i) \), the probability that the \( i \)th feature in \( \tilde{X} \) takes on the value \( x_i \) and \( p(x_i|x_j) \) is the joint probability distribution \( P(\tilde{X}_i = x_i, \tilde{X}_j = x_j) \). These distributions are estimated directly from \( \tilde{X} \notin H \) and \( \tilde{X} \in H \) by counting feature value occurrences and co-occurrences. Edges are pruned from the mutual information graph to form a maximal spanning tree \( T = (\mathcal{V'}, \mathcal{E'}) \). The resulting tree encodes the approximate distribution as a product of pairwise conditionals. Figures 3(b) and 3(c) show the Chow Liu trees built for the appearance of above and below horizon regions for a given input image. Each graph contains seven nodes, three each for the RGB and Lab color encodings and one for texturized. We can calculate \( P(\tilde{X} = \tilde{x}) \), the probability of occurrence of a particular feature vector \( \tilde{x} \in \mathbb{S}^d = \{\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \ldots, \tilde{x}_d\} \) as follows

\[ P(\tilde{X} = \tilde{x}) \approx \prod_{(i,j) \in \mathcal{E}} \frac{p(x_i,x_j)}{p(x_i)p(x_j)} \prod_{j \in \mathcal{V'}} p(x_j) \]  

The Chow Liu trees for \( P(X | ¬H) \) and \( P(X | H) \) are used in (2) to calculate \( P(¬R | X) \) for each patch in an image. An example is shown in Fig. 3(d).

D. Classifier Training

For each patch, the probability of being part of the shore region \( P(¬R | X) \) is calculated using (2). The patches that are least likely to be on the shore (\( P(¬R | X) < \theta \)) are used as candidate river patches. Using a low value for \( \theta \) reduces the chance that shore patches are accidentally used as candidate river patches, but if \( \theta \) is set too low then the selected region will be too small to provide a good representation of the river region. In our experiments, \( \theta \) was set to 0.01. The shore patches above the horizon and the candidate river patches with \( P(¬R | X) < \theta \) are used to train a two class (river/shore) linear support vector machine.

The SVM uses the unabridged, continuous valued feature vectors \( X \), including image position and the complete texture descriptor. Since we are learning a new classifier for each new frame, while the appearance of the river is likely to remain fairly constant over short periods of time, we initialize the SVM training using the SVM learnt on the previous frame to reduce training time. Fig. 3(e) shows the river and shore training examples selected in an image.

E. Detecting Water

The trained SVM is used to classify all the patches in the image. As a post processing step, small holes in the labeling are filled using morphological operators. An example final segmentation result is shown in Fig 3(f)
V. RESULTS

A. Datasets and Groundtruth

Four datasets were used for evaluating our algorithm. The data collection setup was a tripod mounted camera (a SANYO VPC-CG102BK or Canon SX200IS) placed onboard a small motorboat. The cameras captured 1280×720 images which were downsampled to 640×360. Three of the datasets were collected on the Allegheny river at Washington’s Landing in Pittsburgh, PA, U.S.A, the fourth was collected on the Youghiogheny at Perryopolis, PA, U.S.A. The first Allegheny dataset (Allegheny Day) was collected on a summer afternoon, the second (Allegheny Dusk) was collected in the evening and the third (Allegheny Fall) was collected in the afternoon during autumn. The Youghiogheny dataset was collected around noon. Each dataset contains between 120 and 150 images spaced roughly 2 meters apart that were manually segmented for groundtruth information into river, shore (vegetation, structures etc.) and sky regions. An example ground truth labeling is shown in Fig. 4. Also a groundtruth horizon line was marked out on each image.

![Fig. 4. (a) image from the Allegheny Day dataset (b) groundtruth labeling with the river colored red, shore in green and the sky in blue.](image)

The performance metric used is the percentage of pixels misclassified by an algorithm when compared against ground truth. Since the classifiers generate only two output labels and we are not interested in differentiating between the shore and sky we treat them both as a single class (non-river) during performance evaluation. It should be noted that a naïve classifier that marked everything below the horizon as river and everything above the horizon as non-river would have an error rate of around 10% on these datasets.

B. Supervised Segmentation Results

We investigated how well a supervised classifier would be able to generalize to previously unseen environments. From each dataset, 2 images and their groundtruth labelings were picked at random as training examples. Each dataset was classified using two supervised classifiers. The first classifier was trained on the 2 images from the same dataset (’Self’ in Table II), the second was a classifier trained on the 6 images from the other 3 datasets(’Leave One Out’ in Table II). These classifiers were then tested on all the images in the dataset. This process of picking images, training classifiers and testing on all the images was repeated 16 times for each dataset. Table II reports error rates averaged over these 16 trials. It can be seen that the supervised approach worked well when used on datasets it had been trained on but performance often degraded when run on new datasets that were not seen during training. This suggests that a supervised approach does not generalize well to new environments.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Supervised Error (%)</th>
<th>Our Method Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allegheny Day</td>
<td>3.80% (0.64%)</td>
<td>3.81% (0.63%)</td>
</tr>
<tr>
<td>Allegheny Dusk</td>
<td>3.10% (0.46%)</td>
<td>9.59% (1.53%)</td>
</tr>
<tr>
<td>Allegheny Fall</td>
<td>3.05% (0.51%)</td>
<td>10.57% (1.67%)</td>
</tr>
<tr>
<td>Youghiogheny</td>
<td>3.50% (0.34%)</td>
<td>4.47% (0.41%)</td>
</tr>
</tbody>
</table>

C. Self-supervised Segmentation Results

The self-supervised segmentation algorithm described in Section IV was evaluated on the four datasets. Error rates are shown in Table II. Because there is no training step for the self-supervised algorithm, standard deviations are not included with the results. On all four datasets, the self-supervised algorithm outperformed a supervised classifier that was tested on previously unseen datasets. Even when the supervised algorithm was trained on a subset of images from the same dataset it was tested on, it only outperformed the self-supervised method on Allegheny Dusk. This is significant considering how the self-supervised method uses no manually labeled input.

Fig. 5 shows examples of the detected extent of the river along with some intermediate steps of the algorithm. Some failure cases of our algorithm are explained in Fig. 6.

Since the river segmentation is for vehicle guidance, it is important that the algorithm be able to run at around 1 frame per second or faster. Table III shows a profile of the execution time for processing a 640 × 360 image split into 5 × 5 pixel patches with our current MATLAB implementation on an Intel Core2 Q9550 based desktop computer. The execution time indicates that with an optimized C/C++ implementation it should be possible to perform segmentation at 1fps for vehicle guidance.

<table>
<thead>
<tr>
<th>Step</th>
<th>Time taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Extraction (color)</td>
<td>0.49s</td>
</tr>
<tr>
<td>Feature Extraction (texture)</td>
<td>1.08s</td>
</tr>
<tr>
<td>Appearance Modeling</td>
<td>0.41s</td>
</tr>
<tr>
<td>Classifier Training</td>
<td>0.20s</td>
</tr>
<tr>
<td>Final Detection</td>
<td>0.14s</td>
</tr>
<tr>
<td>Total</td>
<td>2.33s</td>
</tr>
</tbody>
</table>

VI. CONCLUSIONS AND FUTURE WORK

We presented a method for using monocular vision to estimate the extent of a river in an image. We demonstrated that a supervised technique is unlikely to generalize well to different environments. We formulated and evaluated a self-supervised alternative that automatically generates a model of
river appearance when presented with an image by leveraging assumptions that can be made about the structure of the environment and the horizon.

Our method is completely memoryless and builds a new model for each input image. Using history of previous river appearance to build an adaptive model would provide robustness against failure. The most serious failure mode of our algorithm is when novel objects (like the boats and piers in Fig. 6(a)) appear below the horizon and are misclassified. Our algorithm is unable to handle this because it does not maintain a model of river appearance and because it assumes that the appearance of the above horizon region adequately models all non-river objects. Extending our approach to learn an appearance model from previous images would enable the detection of novel objects below the horizon.

We would also like to investigate the use of priors on likely river shapes and performing inference over multiple frames in a video sequence to increase accuracy.

REFERENCES