Computer Vision and Image Understanding 170 (2018) 92–108

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Active learning for designing detectors for infrequently occurring objects in wide-area satellite imagery

Tanmay Prakash*, Avinash C. Kak

Purdue University, 465 Northwestern Ave, West Lafayette, IN 47906, USA

ARTICLE INFO

Keywords:
Object detection
Satellite imagery
Active learning
Distributed computing
Feature selection
Pattern recognition

MSC:
41A05
41A10
65D05
65D17

ABSTRACT

Generating ground truth to design object detectors for large geographic areas covered by hundreds of satellite images poses two major challenges: one algorithmic and the other rooted in human–computer interaction considerations. The algorithmic challenge relates to minimizing the human annotation burden by collecting only those ground truth samples that are likely to improve the classifier. And the human–computer interaction challenge relates to the temporal latencies associated with scanning all the images to find those ground truth samples and eliciting annotations from a user. We address the algorithmic challenge by using the now well-known concepts from Active Learning, albeit with a significant departure from how Active Learning has traditionally been presented in the literature: we present a human-operated active learning framework, rather than relying on previously collected fully labeled datasets for simulated experiments. And, we address the human-computer interaction challenge by using a distributed approach that relies on multiple virtual machines working in parallel to carry out randomized scans in different portions of the geographic area in order to generate the active-learning based samples for human annotation. We demonstrate our wide-area framework for two infrequently occurring objects over large geographic areas in Australia. One is for detecting pedestrian crosswalks in a region that spans 180,000 sq. km, and the other is for detecting power transmission-line towers in a region that spans 150,000 sq. km. Using randomly selected *unseen* regions for measuring detector performance, the crosswalk detector works with 92% precision and 72% recall, and the transmission-line tower detector with 80% precision and 50% recall.

1. Introduction

Object detection in remote sensing imagery may be used in various applications: planning cities, mapping regions for disaster relief, monitoring the changes in a region, and so on. Our particular motivation is that of geolocalization, by which we mean the problem of localizing a photograph or a video within an ROI (Region of Interest) that may span hundreds of thousands of square kilometers. The photographs and videos may contain images of any number of objects: arboreal arrangements, road markings, buildings, utility towers etc. While no single object may suffice for geolocalizing a photograph (or a video), a configuration of a small number of such objects can create strong constraints on candidate locations for the photograph (or the video). However, before attempting such solutions for geolocalization, one must create a database of geolocalized objects as seen in the satellite images.

Since geolocalization problems often involve large areas — areas that may span hundreds of thousands of square kilometers and may be covered by hundreds of satellite images — creating a database of geolocalized objects in the images presents several challenges, not the least of which is the difficulty of a manual identification of the positive and the negative training instances of the objects as they appear in the images. The marked objects instances must obviously capture all of the diversity in their appearance over the entire region. Since it is unlikely for a human to possess a good grasp of this diversity, one errs on the side of caution and collects as many samples as humanly possible. Consequently, it is frequently the case that many of the training samples thus annotated by a human are redundant, in the sense that they do not add to the class discriminatory power provided by the other samples.

Active Learning seeks to provide significant mitigation against the above mentioned annotation burden on a human. The core notion of Active Learning is that the learning framework is initially supplied with only a small number — typically just a couple dozen — of strongly positive and negative example patches. Subsequently, as the learning framework scans the satellite images, whenever it runs into a pixel blob for which a classification decision based on the training so far yields a

* Corresponding author.
E-mail address: tprakash@purdue.edu (T. Prakash).

https://doi.org/10.1016/j.cviu.2018.03.004
Received 24 June 2017; Received in revised form 28 February 2018; Accepted 15 March 2018
Available online 24 March 2018
1077-3142/

point in the feature space that is too close to the decision surface, the framework asks the human for help. In this manner, the human input is required only when absolutely necessary. We apply this central idea of active learning to the detection of objects that occur infrequently in satellite imagery. The detection of such objects requires exceptionally low-error classifiers to ensure that both the precision and the recall are sufficiently high.

The work we present in this paper should be seen as an attempt at the development of a more generic framework for designing object detectors than the approach we presented in Prakash et al. (2015). That work presented a road-following framework in which the images are scanned along OpenStreetMap (2017) (OSM) roads for the detection of pedestrian crosswalks. The approach described in that publication was custom designed for the detection of crosswalks — in the sense that the features that were extracted from the pixels were tuned for detecting the alternating black and white strips of the crosswalks. The Active Learning based framework we present here, while motivated primarily by our desire to mitigate the training-data annotation burden on humans, is a more generic framework compared to the one presented in Prakash et al. (2015) in the sense we use the same fundamental logic for discovering the best features to use for detecting pedestrian crosswalks as we do for detecting an entirely different object — transmission-line towers.

It is perhaps obvious that any object-detection framework that is meant to be generic with regard to object types must possess a rich vocabulary of low-level features — features that, to the extent possible, would be a superset of the features actually needed for any particular object. Our work here also demonstrates that it is possible to specify such a generic set of low-level texture and multispectral features. The generic low-level texture features that we use in our work include a large number of Haar-like features and those based on Local Binary Patterns (LBP). And the low-level multispectral features that we add to the texture features are based on the ratios of several spectral differences — along the lines of the well-known NDVI (Normalized Difference Vegetation Index) feature. More specifically, in addition to NDVI, we use NDWI (Normalized Difference Water Index), NDSI (Normalized Difference Soil Index), NHFD (Non-Homogeneous Feature Difference), and NSVDI (Normalized Saturation Value Difference Index).[1] As the reader would expect, such a superset of low-level generic features is bound to be very large. As a matter of fact, the set that we have used in the research reported here contains more than 8000 low-level features. Obviously, not all of the different features types listed here would be needed for all types of objects. As a case in point, the Active Learning based crosswalk detector we describe in this paper does not use any spectral features when the detector is applied to panchromatic imagery.

At this point, the reader may wonder that if our framework must work with several thousands of low-level features for characterizing pixel blobs, would that not make the framework computationally inefficient. It would seem that, regardless of the top-level learning strategy used, it would be a formidable exercise for our framework to zero in on just that feature subset that would give us the required discriminatory power needed *for a specific object type*. We address this issue by interposing a layer of AdaBoost between the top-level active-learning framework and the low-level features. Treating each feature as a weak classifier, AdaBoost helps us select the best N features for the first cycle of Active Learning with the user-supplied strong positive and negative examples of the object under consideration. After every M annotations, we invoke AdaBoost again on all of the training data annotated so far to select a fresh list of N features. The parameter N is typically set to 200 and M to 50.

Despite the fact that active learning in conjunction with AdaBoost as a feature selector provides us with an efficient framework for significantly reducing the annotation burden on a human, we must still worry about the human-computer interaction latencies when creating a detector for an ROI covered by hundreds of satellite images. *A conventional scan of the satellite images for a large ROI would be much too serial an exercise to quickly capture all of the diversity across the ROI.* As we demonstrate in this paper, this problem is best handled with a distributed implementation of the framework in a cloud-based platform in which several virtual machines simultaneously carry out randomized scans in different portions of the geographic area in order to generate active-learning based samples for human annotation.

In addition to the distributed implementation as described above, the efficiency (and also, as the reader will see later, the overall performance) of the active learning based detector design can be further improved by invoking object-specific considerations that either make it unnecessary to look in those portions of the satellite images where the absence of the objects is more or less guaranteed or provide constraints for eliminating false positives. For example, when looking for pedestrian crosswalks, by the very definition of such objects, it makes sense to only look for them along the roads. So if we could project the relevant portions of, say, OSM into the satellite images, we should be able to ignore large portions of the satellite images when we look for pedestrian crosswalks. For a different example of an object-specific consideration, consider a detector for electric power transmission-line towers. We know that these towers are constructed on the ground in stretches of long straight lines. Thus we can refine an initial set of detections by applying the Hough transform to the detected towers and using the straight lines thus formed in the final accept/reject decisions for the towers.

We demonstrate our active learning based framework for the detection of two different infrequently occurring objects in a large geographic area of Australia. One is for detecting pedestrian crosswalks in a region of Australia that spans over 180,000 sq. km and is covered by 222 satellite images. The other is for detecting electric power transmission-line towers in an area in Australia that spans over 150,000 sq. km and is covered by 606 satellite images. For each of these detector types, we also exploit detector-specific domain knowledge in the overall design framework. Using randomly selected test regions for measuring detector performance, the crosswalk detector works with 92% precision 72% recall and transmission tower detector with 80% precision and 50% recall.

The remainder of the paper is organized as follows. Section 2 details the prior work in active learning, both in theory and in applications to remote sensing. Section 3 describes briefly the satellite data we used for the research reported here. Sections 4 and 5 discuss the particulars of the low-level features and the AdaBoost-trained classifier, whereas Section 6 discusses the components of the active learning framework at a higher level. Section 7 describes the object-specific considerations that were used to improve the detection of transmission towers and crosswalks. Section 8 discusses the results of the experiments on transmission tower detection and crosswalk detection. And finally the paper concludes in Section 9.

2. Related literature

Active learning has attracted much attention for solving image classification problems in remote sensing, as evidenced by the survey papers (Crawford et al., 2013; Tuia et al., 2011). In remote sensing, image classification refers to the problem of assigning a class label, such as building, water, vegetation, etc., to each of the pixels. The methods surveyed are pool-based and collect batches of unlabeled query samples. Queries are selected from the pool by ranking each unlabeled

[1] We do not mean to imply that these features constitute an exhaustive set for all of the different types of objects one may be able to detect in satellite imagery. Our goal is only to present a scalable active-learning based framework in which the set of low-level features can easily be added to, if that is made necessary by a new object type, without any changes to the main logic of active learning.Having said that, the reader should note that our low-level feature mix works well for two very different object types—pedestrian crosswalks and power transmission-line towers.

sample by a given criterion/heuristic and selecting a set of highly ranked samples. Similar samples are likely to have similar rank, but are also likely to provide redundant information; therefore constraints are put in place to ensure a diverse set of samples. When the underlying classification strategy is based on large-margin classifiers, such as SVMs, the unlabeled samples for which the oracle is consulted are those that fall within the margins.

Active learning research in remote sensing has often focused on developing different heuristics for active learning algorithms and comparing them based on the number of labeled samples that are necessary to achieve the same classification performance. This comparison is often facilitated by *simulating* the human oracle annotation process. A fully labeled dataset is stripped of its labels then input to the active learning algorithm. When the machine requests a label from the human, it is instead provided with the label from the original dataset, enabling researchers to run any number of experiments with minimal human effort. Researchers can thus compare a large number of active learning algorithms but this comes at the cost of two subtle side effects: the dataset must be small enough to have been labeled in its entirety and the consequences for the human operator of the latencies between the machine's request for labels and the human getting around to supplying them are less apparent to the researcher.

Our own work, as reported in this paper, focuses on object detection and not on image classification in the sense described above. While one could cast object detection as a problem of image classification, i.e. identifying those pixels that belong to the object class, the practical demands of the two problems differ enough that we consider them to be separate. The pixel blobs corresponding to the objects we are interested in occur at a frequency that is far lower than what is the case for the different types of pixels in image classification. So while a false positive rate of 1% may suffice for image classification, in object detection it would likely mean that majority of the detections are false. The rarity of positive samples also complicates the use of active learning when applied to object detection. Obviously, a sufficiently large set of positive samples is necessary for accurate supervised learning.

We should also mention that one of the concerns in the previously reported research related to active learning in the image classification context is that of capturing adequate diversity in the positive and the negative samples. While other methods attempt to impose diversity explicitly, in our cloud-based distributed implementation, samples are collected by multiple "workers" using randomized sample selection strategies over large areas; therefore the wide spatial distribution of samples should by itself provide diversity.

The application of active learning in remote sensing owes its origins to much theoretical work that has been done on the subject (Hanneke, 2014; Settles, 2012). Much of the theoretical research aims to find active learning approaches that are guaranteed to converge to the optimal hypothesis while requiring fewer labeled samples than passive learning. Early research grappled only with the *realizable* case, in which it is assumed the set of hypotheses being considered contains a hypothesis that can perfectly separate the two classes. More recent research considers the *non-realizable* case, which abandons that assumption and instead seeks to find the hypothesis with the minimal error. A popular paradigm that has received much attention in theoretical research is disagreement-based active learning (Hanneke, 2014). The core idea here is to select only those samples for the oracle that have conflicting predicted labels among the hypotheses under consideration, and use this conflict to eliminate hypotheses. Disagreement based active learning can provide theoretical guarantees regarding the optimality of the final classifications. For instance, the agnostic active (A^2) algorithm (Balcan et al., 2006) is guaranteed to converge to the optimal hypothesis, and its labeling complexity can be shown to be exponentially smaller than what is the case for passive learning in some cases, and not significantly larger in the worst case.

For the case when the number of candidate hypotheses that the learner must maintain becomes too large to explicitly store in memory, creative applications of an empirical risk minimization (ERM) oracle can bypass that need altogether. ERM based strategies have been implemented for both the *realizable* and *non-realizable* cases. Under *realizable*, a hypothesis can be rejected if it misclassifies any labeled samples, but under *non-realizable*, even the optimal hypothesis can misclassify a sample; therefor more evidence is necessary to reject a hypothesis in the non-realizable case. The CAL algorithm (Cohn et al., 1994) for the realizable case detects a disagreement on an unlabeled sample by using the ERM oracle to train two hypotheses: one with all of the labeled samples and the new sample labeled positively, and one with all of the labeled samples and the new sample labeled negatively. If in both cases the hypotheses have no error, these two hypotheses disagree on the unlabeled sample, and the learner queries the oracle for a label for the unlabeled sample. For the more complex non-realizable case, see the work reported in Dasgupta et al. (2007) and Beygelzimer et al. (2011).

Disagreement based active learning frameworks that have been proposed have limited practical utility for the large-scale research we report here. Instead of sampling the hypothesis space or using an ERM oracle (which can be computationally expensive), we have chosen to use AdaBoost as a dynamic feature selector that can make strong discriminations between the pixel blobs corresponding to the objects and the background pixels in the satellite images. We say that our AdaBoost based logic is dynamic because it has the ability to change the features for characterizing the pixel blobs as new positive and negative samples are acquired through active learning.

With regard to using AdaBoost for feature selection, the SEVILLE system presented in Abramson and Freund (2005) also uses such a feature selection in an active learning framework to train pedestrian detectors. And the authors in Grabner et al. (2008) use an online approximation to AdaBoost in order to train an aerial-imagery-based car detector through active learning. As long as we are on the subject of AdaBoost, we should also mention the previous work on query-by-boosting (Abe and Mamitsuka, 1998).

The active learning algorithm we employ is based what is referred to as *uncertainty sampling*. This process begins by approximating an intermediate decision boundary from a small initial set of labeled positive and negative samples. Using this intermediate decision boundary, the learner can identify unlabeled samples whose predicted labels are uncertain. Once identified, the learner can present these unlabeled samples to a human oracle for labeling, and then use them to update the decision boundary. This process of labeling samples and updating the classifier can be iterated numerous times until the detector achieves desirable performance. The number of samples labeled by the human user during this process should ideally be significantly fewer than the case when traditional passive learning is used for training the detectors.

3. The satellite data used in our evaluation of object detectors

The work we have reported in this paper on the detection of pedestrian crosswalks is based on the satellite images for an ROI covering a 180,000 sq. km area southeast of Australia. The crosswalk experiments are performed using GeoEye-1 panchromatic images at a spatial resolution of 0.4 - 0.5 meters per pixel.

And the work we have reported on the detection of electric power transmission-line towers is based on 0.4 - 0.5 meters per pixel 8-band multispectral imagery pansharpened from WorldView-2 0.4 - 0.5 meters per pixel panchromatic and 2 meters per pixel 8-band multispectral imagery, covering a 150,000 sq. km subset of the southeast Australia ROI.

In both cases, a top-of-atmosphere correction was applied to the satellite images in order to remove some of the imaging effects that introduce variations to the appearance of the target objects.

4. Features

Since our goal is to create a generic framework for constructing object detectors — generic to the maximum extent possible — how we specify the features to be extracted from the satellite images becomes a very important issue. For contrast, in our earlier approach in Prakash et al. (2015), the set of features we used were engineered specifically for detecting the alternating black and white stripes of pedestrian crosswalks.

A generic framework requires a set of features that has the potential of working for different types of objects. These object types may be as different as the two we have considered in this paper: pedestrian crosswalks and electric power transmission-line detectors. There is virtually nothing in common between these two object types.

We must obviously choose low-level features that are invariant to small deformations and changes in illumination. Even more importantly, the features must disregard any specific relationship between the different visual components of the objects, since such relationships would be object specific.

In the subsections that follow, we will first introduce the reader to the data abstractions "Scanning Window", "Image Patch" and "Block." The low-level features are meant for the characterization of the pixels in a scanning window and a scanning window resides in a larger abstraction called "Image Patch," and some features require dividing the "Scanning Window" into "Blocks."

4.1. Image patch, scanning window, and block

Our framework uses three main data abstractions for designing a detector: an *image patch*, a *scanning window*, and a *block*.

The system looks for the presence/absence of the object at each position of a moving *scanning window* in an *image patch*.[2] Unfortunately, a scanning window, meant to be a rectangular enclosure for the object being detected, is much too small a data abstraction for the human-computer interaction for eliciting a label for the pixels inside the window. When deciding whether or not an object is present at a given location of the scanning window, a human also needs to see the surrounding context for the window. Toward that end, we use the notion of an *image patch* that is several times larger in size than that of a scanning window. Fig. 1 illustrates how an image patch can provide additional context for a window.

That brings us to the last of the three abstractions, *block*. In order to make feature measurements inside a scanning window, it is frequently necessary to divide the scanning window into an array of *blocks* that may or may not be overlapping. Subsequently, the relationship between the feature values in the different blocks inside a scanning window captures some discriminating aspect of the object to be detected.

At training time, image patches are extracted from the satellite data covering a geographic region, *with the patches located randomly in the geographic region of interest.* Each patch is scanned with a scanning window, with each location of the scanning window yielding one of three answers obtained with the current state of the detector in the active-learning based framework: (1) the object is definitely inside the scanning window at that location; (2) the object is definitely absent; or (3) the detector cannot be sure because the feature values for the pixels inside the scanning window are too close to the decision surface in the feature space. All of the locations in the image patch with the third outcome are marked and shown to the human for classification decision.

Fig. 1. Illustration of a scanning window within an image patch. What is enclosed by the blue border is an example image patch extracted as described in Section 6.2. The green rectangle is an example location of the moving scanning window. In this case, the scanning window happens to enclose the shadow of an electric transmission-line tower. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

At testing time, those image patches for which the ground truth is available are extracted by the same procedure and scanned with the trained object detector. Each image patch is examined with a moving scanning window, and, at each location of the scanning window, the detector decides whether or not the object is present in the window. This result is subsequently compared with the ground truth.

4.2. Haar-like features

Haar-like features have been widely used in computer vision since they were successfully demonstrated for face detection by Viola and Jones (2004). Haar-like features are simple, robust, and computationally efficient. In this subsection, we will show how our detector design framework represents the pixel contents of a scanning window with a large number of Haar-like features.[3]

Haar-like features for characterizing a scanning window in a satellite image are calculated through the five operators shown in Fig. 2. The two operators shown in (a) and (b) are for calculating approximately the first-order horizontal and vertical derivatives at a pixel. The operators in (c) and (d) are for approximating the second-order horizontal and vertical derivatives. Finally, the operator in (e) is for approximating the second-order cross-derivative. An operator is moved to each pixel inside the scanning window and, at each pixel, one calculates the output of the operator as the difference of the sum of the "red" pixels and the sum of the "blue" pixels.

In Viola and Jones (2004), the authors introduce the notion of an integral image for a fast computation of Haar-like features. The integral image $II(x, y)$, defined as shown below, is a look-up table whose value at column and row (x, y) is equal to the sum of all the pixels in the image $I(x, y)$ that are above and to the left of the pixel at (x, y):

[2] Since the purpose of a scanning window is to detect an object, its size must depend on the size of the object. As the reader will see, for tall object that cast shadows, one may want to use scanning windows that are aligned with the shadows, something that is easy to do since the metadata associated with a satellite image includes the sun azimuth. Recognizing tall objects using the information in the shadows cast by them is best accomplished by orienting the image patches so that one of the patch axes—say the x-axis—aligns with the direction of the shadow. This makes it possible to use a regular unoriented scanning window inside the image patch for detecting the objects. We will get into the specifics of the scanning window parameters in a later section in this paper.

[3] To give the reader a sense of how many such feature values are calculated, for the implementation presented in Section 8.1, for the case of power transmission-line towers a scanning window is of size 36×120 and each is represented with 7263 Haar-like features. A reader not familiar with such features may think that that is way too many. Consider the fact that a 24×24 scanning window in the Viola and Jones face detector is represented with 173,000 Haar-like features. As will be mentioned shortly, it takes a very small number of lookup operations, typically of the order of unity, to calculate each Haar-like feature.

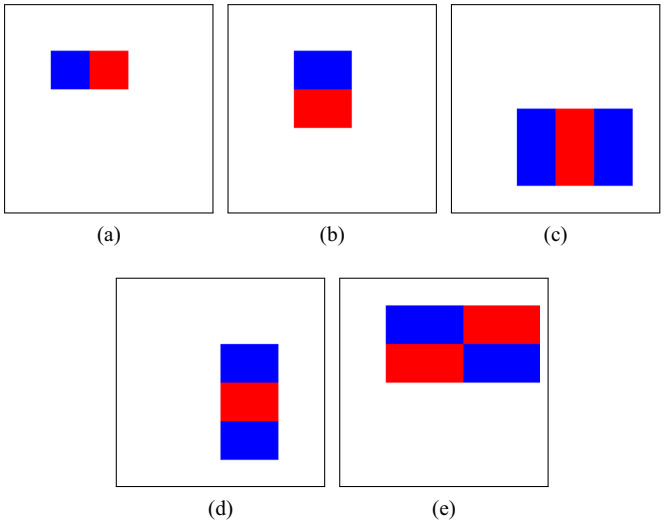


Fig. 2. The different placements of the Haar operators in the enclosing window are simply meant to convey the idea that the operators have no fixed locations in a scanning window (in the training phase). A scanning window is itself scanned with each operator, with the operator translated from pixel to pixel, and at each location, the corresponding Haar feature value extracted. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

$$II(x, y) = \sum_{x_i \le x, y_i \le y} I(x_i, y_i) \tag{1}$$

The integral image can be calculated recursively because

$$II(x, y) = II(x-1, y) + II(x, y-1) - II(x-1, y-1) \tag{2}$$

As will be explained in Section 4.5, the integral image is calculated for each *image patch* at a time. Once the integral image is calculated, the sum of the pixels in either the red rectangles or the blue rectangles of the operators shown in Fig. 2 can be calculated with just four calls to the data stored in the integral image. To calculate the sum $S(A)$ of the pixels at locations in the rectangular region $A = \{(x, y): x \in [x_0, x_1], y \in [y_0, y_1]\}$, we use

$$S(A) = II(x_1, y_1) - II(x_0 - 1, y_1) - II(x_1, y_0 - 1) + II(x_0 - 1, y_0 - 1) \tag{3}$$

Thus using this approach any Haar-like feature can be calculated in constant time.

4.3. Local binary patterns

Local binary patterns (LBP) features were originally designed to characterize the texture in images (Kak, 2016; Ojala et al., 2002). While there are now several different version of LBP in the literature, we make use of the rotationally invariant uniform version introduced in Ojala et al. (2002). More recently LBP features have also been used for face recognition (Ahonen et al., 2006) and car detection in aerial images (Grabner et al., 2008). In this subsection we show how each scanning window is represented with LBP features. As with the Haar-like features, the number of LBP features used to represent a scanning window is again large.[4]

With LBP, the local texture at a pixel in a single-channel image is characterized with a local binary pattern by observing the pixel values at P locations that are evenly spaced on a circle of radius R pixels surrounding the pixel in question. Fig. 3 illustrates an example of these

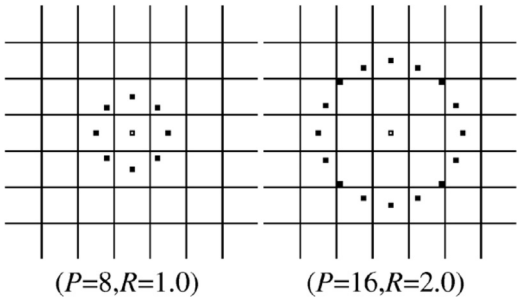


Fig. 3. Examples of sampling points (black squares) on a circle around a given center pixel (white squares). Below each example is indicated the number of sampling points P and the radius of the circle R in pixels. Image from Ojala et al. (2002).

sampling points. We compare the center pixel to each of the points on the circle and represent these comparisons with a binary pattern in which a '1' stands for the pixel on the circle being equal to or greater than the one at the center and '0' for the case when that is not true. Since the binary pattern involves differences between pixel values, it is invariant to small changes in illumination.

The binary patterns are made rotationally invariant by shifting them circularly until the integer value of the pattern is the least. This happens when the largest number of zeros occupy the most significant bit positions in a pattern. The original authors of LBP have recommended that only those binary patterns be retained as features that are "uniform," meaning that consist of a single string of 0's followed by a single string of 1's (Ojala et al., 2002). For example 00001111, 00000001, 00000000, and 11111111 are uniform, but not 00101111. A uniform binary pattern is assigned a label equal to the number of 1's in the pattern. The labels of uniform patterns can range from 0 to P. Each such label is used to characterizes the texture at the pixel that is at the center of the circle. A pixel whose binary pattern is nonuniform is assigned the label $P + 1$. Thus, the pixel-level local texture can be encoded using $P + 2$ possible labels.

Following Ojala et al. (2002) and Ahonen et al. (2006), we form a histogram of the LBP labels assigned to all pixels in each block of an array of blocks inside a scanning window. Subsequently, we concatenate the histograms for all the blocks in order to form a descriptor. The histogram for each block has $P + 2$ bins. So if we have $M \times N$ blocks inside a scanning window, that yields a descriptor of length $M \times N \times (P + 2)$. Since our design framework is meant to be generic, we need an LBP descriptor that would yield a texture characterization for a couple of different choices for P and R, for a small number of choices for the block size, and for a small number of choices regarding the arrangement of the blocks in a scanning window.

The choices made for P, R, M, and N can be thought of as belonging to the set of tunable parameters of our generic detector design framework. You would need to make appropriate choices for these parameters for each object type. Shown in Table 1 are the choices that have worked for us for the case of the detectors for pedestrian crosswalks and for electric transmission-line towers. These choices result in a total of 952 LBP features for crosswalks and 700 LBP features for transmission-line towers.[5]

[4] For the implementation presented in Section 8.1 for the case of electric transmission-line towers, a scanning window is of size 36 × 120 pixels and each such window is represented with 700 LBP features.

[5] As to how we arrive at these numbers, as mentioned each block yields a histogram of $P + 2$ bins. So a concatenation of the histograms for an arrangement of 3 × 3 blocks yields a descriptor with $9 \times (P + 2)$ elements. For the first row of Table 1 for the case of crosswalks, we have $P = 8$ and a total of 9 blocks in an 3 × 3 array. So this row will contribute $10 \times 9 = 90$ elements to the LBP descriptor. The second row of the table yields an additional $10 \times 25 = 250$ elements to the LBP descriptor. Similarly, the third row yields 162 elements and the last row 450 elements. When we add all these elements together, we get a descriptor with 952 elements. So we say that we represent a scanning window for crosswalk detection with 952 LBP features. In a similar manner, the part of the table devoted to transmission-line towers says that we represent a scanning window in that case with 700 LBP features.

Table 1
LBP parameters selected for two different object types.

	P	R	Block size (pixels)	Grid (blocks)
Crosswalk	8	1	8 × 8	3 × 3
	8	1	4 × 4	5 × 5
	16	2	8 × 8	3 × 3
	16	2	4 × 4	5 × 5
Trans. tower	8	1	24 × 36	2 × 5
	8	1	12 × 24	3 × 5
	16	2	24 × 36	2 × 5
	16	2	12 × 24	3 × 5

The calculation of the histograms used in the LBP features is made computationally efficient through the use of integral images constructed from the binary masks associated with the different LBP labels, as we now explain. A bin of the histogram is just the number of pixels within a rectangular region that have the given LBP label. To get this number, we first create a binary image with a pixel set to 1 if it has the given LBP label and 0 otherwise, and then calculate the integral image of this binary image. The value of a pixel in this integral image $II_{\mathrm{LBP}}(x, y)$ gives the number of pixels from the original image that have the given LBP label and fall in the rectangular region extending from the origin to (x, y), inclusive. Once such an integral image is calculated for every possible LBP label, we can quickly calculate the frequency of any label in any rectangular region in the image, from which we can generate the necessary histograms.

4.4. Spectral features

When multispectral data is available, the spectral signature recorded at a pixel in a satellite image is another source of discriminative power with respect to the material on the ground being imaged. Our experience with object detection has shown that, when using multispectral data, it is best to calculate such features separately in a small number of blocks inside the scanning window. Just to illustrate what that means for a specific case, for the implementation of the detector for electric transmission-line towers, the scanning window is divided into 4 blocks and a characterizing 13-dimensional spectral signature calculated separately for each block.

With regard to how the characterizing spectral signatures are calculated for each block, note that the WorldView-2 multispectral imagery used in this paper has 8 spectral bands: Coastal, Blue, Green, Yellow, Red, Red Edge, Near Infrared 1 (NIR1), and Near Infrared 2 (NIR2). One commonly employed method of using pixel spectral signatures is through "normalized" differences between pairs of bands (Gao, 1996; Ma et al., 2008; Myneni et al., 1995; Wolf, 2012); this approach can enhance spectral differences and remove gain factors that would otherwise vary from image to image. We use four of these quantities, referred to as "index" values, as adapted to WorldView-2 imagery byWolf (2012). We also include one additional index, Normalized Saturation Value Difference Index, that is based on "saturation" considerations in Ma et al. (2008). Shown below are these five different indexes derived from the spectral signature at each pixel:

- Normalized Difference Water Index (NDWI)

$$\mathrm{NDWI} = \frac{\mathrm{Coastal} - \mathrm{NIR2}}{\mathrm{Coastal} + \mathrm{NIR2}} \tag{4}$$

for identifying standing water

- Normalized Difference Vegetation Index (NDVI)

$$\mathrm{NDVI} = \frac{\mathrm{NIR2} - \mathrm{Red}}{\mathrm{NIR2} + \mathrm{Red}} \tag{5}$$

for identifying vegetation

- Normalized Difference Soil Index (NDSI)

$$\mathrm{NDSI} = \frac{\mathrm{Yellow} - \mathrm{Green}}{\mathrm{Yellow} + \mathrm{Green}} \tag{6}$$

for identifying soil

- Non-Homogeneous Feature Difference (NHFD)

$$\mathrm{NHFD} = \frac{\mathrm{Red\ \ Edge} - \mathrm{Coastal}}{\mathrm{Red\ \ Edge} + \mathrm{Coastal}} \tag{7}$$

for identifying man-made structures

- Normalized Saturation Value Difference Index (NSVDI)

$$\mathrm{NSVDI} = \frac{\mathrm{Saturation} - \mathrm{Value}}{\mathrm{Saturation} + \mathrm{Value}} \tag{8}$$

for identifying shadow pixels.

Saturation and Value in the definition of NSVDI are calculated through

$$\mathrm{Saturation} = \frac{\max\{bands\} - \min\{bands\}}{\max\{bands\}} \tag{9}$$

$$\mathrm{Value} = \max\{bands\} \tag{10}$$

where *bands* is the set of values, normalized to [0,1], in each of the eight bands at the given pixel. Note that in the original definition of NSVDI in Ma et al. (2008), Saturation and Value were calculated using just the Red, Green, and Blue bands through the "RGB-to-HSV' transformation formulas found commonly in the literature dealing with color in computer vision and computer graphics (Kak, 2016). We have extended that definition to take into account all of the WorldView-2 bands.

Fig. 4 is meant to give the reader a sense of the usefulness of all five indexes. Since our goal is to create a generic object detector design framework, we must be reasonably exhaustive in extracting the material characterizing information from the bands.

We combine the eight WorldView-2 spectral bands with the five indexes in Eqs. (4)–(8) to create a 13-dimensional spectral feature vector at each pixel. Subsequently, as an image patch is scanned with a scanning window for evidence of an object, we aggregate these 13-dimensional pixel-based characterizations into an overall 13-dimensional characterization of a block of pixels inside the scanning window as explained below.

For the purpose of multispectral characterization, a scanning window is divided into four overlapping blocks, the second of which is shown in Fig. 5. In each of the four blocks, we aggregate the 13-dimensional spectral signatures at the pixels with a Gaussian weighting function that peaks at the center of the block. In order to see the reason for Gaussian weighting, note that our aim is to characterize a block from the standpoint that the object we are looking for is at the center of the block. The Gaussian weighting emphasizes the central pixels in relation to the pixels near the boundary of the block.[6] This aggregation results in a 13-dimensional characterization of each block.

4.5. Patch preprocessing for efficient computation

In this section, we review some of the finer aspects of how a patch is processed so that the calculations required for the measurement of the features inside a moving scanning window do not contribute unacceptable latencies to the human–computer interaction. In particular, we want to take advantage of the redundancies between any two successive positions of the scanning window. Note that the scanning window is displaced by only two pixels from one location to the next.

As described in Section 4.2, an integral image representation is used to facilitate fast Haar-like feature calculation. Integral image representation (as derived from binary masks) are used as well to

[6] Such Gaussian weighting is common to several image descriptors such as SIFT and SURF.

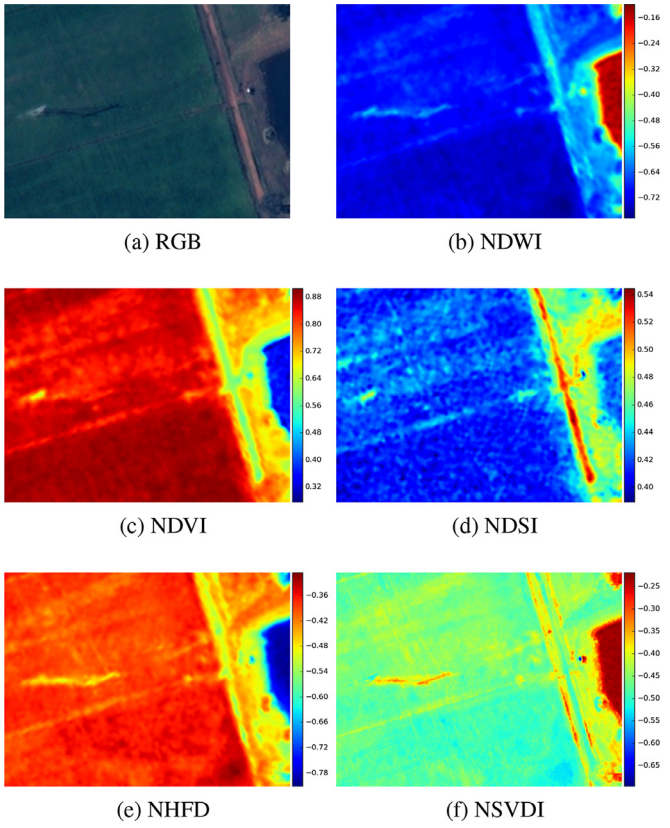


Fig. 4. An illustration of the usefulness of the five index values derived from the spectral signature at each pixel. (a) The RGB image shows a transmission tower on the left, and a dirt road and a pond on the right. The pond has a high response in the NDWI image (b), the grass has a high response in the NDVI image (c), the dirt road has a high response in the NDSI image (d), and, although not exclusively, the shadow of the tower has a relatively high response in the NSVDI image (f). The NHFD image in (e) may seem anomalous since it shows grass with the greatest response. That is because the scene is devoid of typical man-made structures like buildings. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

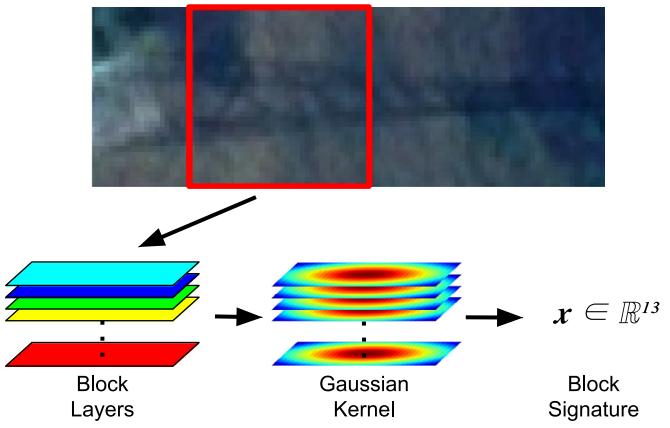


Fig. 5. Flowchart of spectral signature calculation for a single block of pixels. The second of four overlapping blocks is highlighted in the top image. Gaussian weighting is applied to each of the 13 spectral attributes at each pixel in the block, resulting in a 13-dimensional signature for the block. The spectral signature of a scanning window is the concatenation of the spectral signatures for each of its blocks.

calculate the histograms of LBP values in Section 4.3. In both cases, it is *not* the scanning window for which an integral image representation is created — it is the image patch. Creating an integral representation for an image patch reduces what would otherwise be redundant computations performed for the overlapping scanning windows. Given an integral representation for an image patch, the value of a feature for any block of pixels in a scanning window can be calculated by locating that block in the enclosing image patch. For example consider a scanning window whose top-left corner is at (x_w, y_w) with respect to the image patch, and a rectangular block whose top-left and bottom-right corners are at (x_0, y_0) and (x_1, y_1), respectively, with respect to the scanning window itself. The sum of the pixels within the block A would then be calculated as

$$\begin{aligned} S(A) &= II(x_1 + x_w, y_1 + y_w) - II(x_0 + x_w - 1, y_1 + y_w) \\ &\quad - II(x_1 + x_w, y_0 + y_w - 1) + II(x_0 + x_w - 1, y_0 + y_w - 1) \end{aligned} \tag{11}$$

With regard to the computation of the spectral features, recall that the extraction of spectral signatures involves the application of Gaussian weighting that, if implemented naively, can be computationally prohibitive. We consider each of the 13 spectral attributes (8 bands and 5 ratio attributes described in Section 4.4) calculated at each pixel in a patch as a layer. Each such layer for a patch is convolved with a Gaussian kernel. The 13-element spectral signature [7]associated with a block consists of the values drawn from each of the 13 layers at the center of the block. As mentioned earlier, the signature of a scanning window is simply the concatenation of the signature of each of the block signatures. In this manner, after preprocessing, the spectral features for each scanning window can be calculated in constant time.

The convolution itself is made efficient using separable filters (Smith, 1997). The Gaussian kernel can be separated into two 1D kernels $g_x(x)$ and $g_y(y)$

$$\begin{aligned} g(x, y) &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{2\sigma^2}(x^2 + y^2)\right) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}x^2\right) \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{1}{-2\sigma^2}y^2\right) \\ &= g_x(x) \cdot g_y(y) \end{aligned} \tag{12}$$

which allows us to split the computation into two separate but much faster convolutions with two 1D kernels

$$I(x, y) * g(x, y) = I(x, y) * g_x(x) * g_y(y) \tag{13}$$

Whereas a naive application of a $K \times K$ kernel to a $P \times P$ patch would require $O(K^2P^2)$ time, the separable filter approach requires only $O(KP^2)$.

5. AdaBoost

As the reader will see in Section 6.1, for the purpose of collecting training data, after a user has supplied a small number of strongly positive and negative examples of the object for which a detector is desired, the system randomly selects image patches, examines each image patch with a scanning window, and marks those scanning window locations where the current decision boundary in the active learning framework does not conclusively yield a decision regarding the presence/absence of the object. Such scanning window locations are then shown to the user for his/her input. This process yields a collection of scanning windows as positive and negative examples for the object.

Given the low-level feature characterizations for the specific scanning windows chosen for the training data, it is the job of the AdaBoost algorithm described in this section to find T best such characterization for discriminating between the positive examples and the negative examples. Take for example the transmission tower feature set. Each window is characterized by 7,263 Haar-like features, 700 LBP features,

[7] Note that we have used "spectral signature" for characterizing a pixel, a block, and a scanning window. The context should make it clear as to which data abstraction is being referred to in any given usage.

and four 13-dimensional spectral signatures for each of the four blocks in a scanning window, but the detector should utilize only a small set thereof.

As described in Freund and Schapire (1997) and Kak (2012), AdaBoost builds a strong classifier from an ensemble of weak classifiers by adding, at each iteration, a new weak classifier to the ensemble that is trained to correctly classify samples that were misclassified in previous iterations. In order to place more importance on the previously misclassified samples, AdaBoost maintains a set of weights on the samples that is updated at every iteration. In this paper, at each iteration, we select the best weak classifier from a large set of weak classifiers. The large set of weak classifiers that we consider are based on the individual low-level Haar-like or LBP features, and the 13-element spectral feature vector. For example, for the case of transmission-line towers, there are 7,263 weak classifiers for each of the 7,263 Haar-like features, 700 for each of the 700 LBP features, and 4 for each of the four 13-dimensional spectral signatures. That gives us a total of 7,967 weak classifiers.

More formally, we represent the labeled training data as a set of pairs $\{(x_i, y_i)\}_{i=1}^{m}$ of training samples x_i and labels $y_i \in \{-1, 1\}$. The strong classifier returned by AdaBoost is a function $h_f(x)$ that maps each training sample x to one of the labels $\{-1, 1\}$. The supervised learning problem here is one of induction; we want to use the labeled training samples to estimate a strong classifier $h_f(x)$ that will accurately classify unseen samples. Thus the classifier must generalize well, i.e. accurately classify samples not included in the training data, but that are drawn from the same distribution. Part of the popularity of AdaBoost is the many empirical demonstrations of this generalization property in the literature as well as the theoretical proof in Schapire et al. (1998), even as the complexity of the model increases.

AdaBoost maintains a probability distribution over all the training samples during each iteration of training to indicate how much weight each sample should be given during the training of the weak classifiers. This distribution is modified at each iteration with the newly selected weak classifier. We will denote this probability distribution by $D_t(x_i)$ at the start of the tth iteration of the AdaBoost algorithm. At the outset, that is, for $t = 0$, $D_0(x_i)$ is considered to be uniform.

At every iteration, $D_t(x_i)$ is updated so that the training samples that were misclassified at the end of the previous iteration have a higher weight. As to how $D_t(x_i)$ is updated, we first calculate the sample misclassification rate for current selected weak classifier h_t on all the training samples. The misclassification rate, denoted ϵ_t is calculated by the formula:

$$\epsilon_t = \frac{1}{2}\sum_{i=1}^{m} D_t(x_i)\cdot\left|h_t(x_i) - y_i\right| \tag{14}$$

These misclassification rates are translated into how much trust we can place in the weak classifier h_t by estimating the trust factor α_t as follows:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{15}$$

Finally, we update the probability distribution $D_t(x_i)$ for the next iteration by:

$$D_{t+1}(x_i) = \frac{D_t(x_i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} \tag{16}$$

where the role of Z_t is to serve as a normalizer, which implies

$$Z_t = \sum_{i=1}^{m} D_t(x_i)e^{-\alpha_t h_t(x_i) y_i} \tag{17}$$

Note that $D_{t+1}(x_i)$ decreases if the ith sample is correctly classified by the weak classifier h_t, implying that such a sample x_i will have a less weight in the next iteration

An important issue related to the implementation of a weak classifier is the method used for its training. For weak classifiers that consider only a single feature, as is the case for those based on the individual Haar-like and LBP features, we simply find a decision threshold on the feature value. We follow Viola and Jones (2004), which starts by constructing an ordered list of the feature values for all the training samples. You then step through all possible discrete values for the feature in question, from the lowest to the highest, and, considering each such value as a possible decision threshold θ, select the θ with the minimum misclassification rate. The reader is referred to Kak (2012) for a more thorough description of this algorithm that explains how the misclassification rate can be recursively calculated in an efficient implementation that requires only a single pass through all of the sorted feature values.

The computational complexity of this procedure is $O(LN\log N)$ with L as the number single-feature weak classifiers and N the number of samples. Although the number of training samples is in the tens of thousands and the number of weak classifiers is in the thousands, we can take advantage of the fact that, in the active learning based framework described in the next section, we update the classifier with only a small number of new samples. The old samples have already been sorted in the previous update; therefore we can sort the feature values from just the new samples and then merge the sorted new values with the sorted old values. Since the merging step on all of the weak classifiers requires only $O(LN)$ time, this procedure results in a significant reduction in the computational effort involved.

Regarding the weak classifiers corresponding to the 13-dimensional spectral signature vectors for each of the blocks in a scanning window, the best decision surface for each such vector is found by a linear SVM (Support Vector Machine) that models the relationship between the vector x and its classification $h(x)$ by

$$h(x) = \omega^T x + b \tag{18}$$

where ω is a normal to the decision hyperplane and b is a bias term. The parameters of the best separating hyperplane are found using stochastic gradient descent to solve

$$w^*, b^* = \arg\min_{\omega,b} \frac{1}{2}\lambda \|\omega\|_2^2 + \sum_{i=0}^{N-1} D_t(x_i)\cdot\text{hinge}(\omega^T x_i + b, y_i) \tag{19}$$

where λ is a pre-defined constant that serves as a relative weight for the regularization term and the hinge loss function, $\text{hinge}(h(x), y) = \max\{1 - h(x)y, 0\}$, penalizes any estimated classification in the margin.

5.1. Parallelized selection of the next best weak classifier

Each iteration of the AdaBoost algorithm requires that, from amongst all the potential weak classifiers, we find the one that gives the lowest misclassification rate. This computation at each iteration lends itself well to a parallel implementation that in our system is carried out with the help of multiple virtual machines (VM's) in a cloud-based computing platform.

Fig. 6 illustrates the method by which this parallelized processing is managed. We spawn multiple worker VM's and a head VM that coordinates everything.

At the initialization of the Active Learning Framework, the head VM sends each worker VM the initial set of samples and assigns each worker a fixed subset of the weak classifiers. Assigning each worker the same subset of the weak classifiers allows the worker to use information from prior classifier updates— for example the sort order needed by the method described earlier in Section 5 — to accelerate training. This is an important detail that can significantly reduce latencies in the human–computer interaction.

At the start of a classifier update, when the framework has collected new samples, the head sends each worker the new samples, and the workers assign all of the samples the same weight. At each iteration of AdaBoost, each worker trains its set of weak classifiers with the same current set of sample weights as the other workers, selects the one with

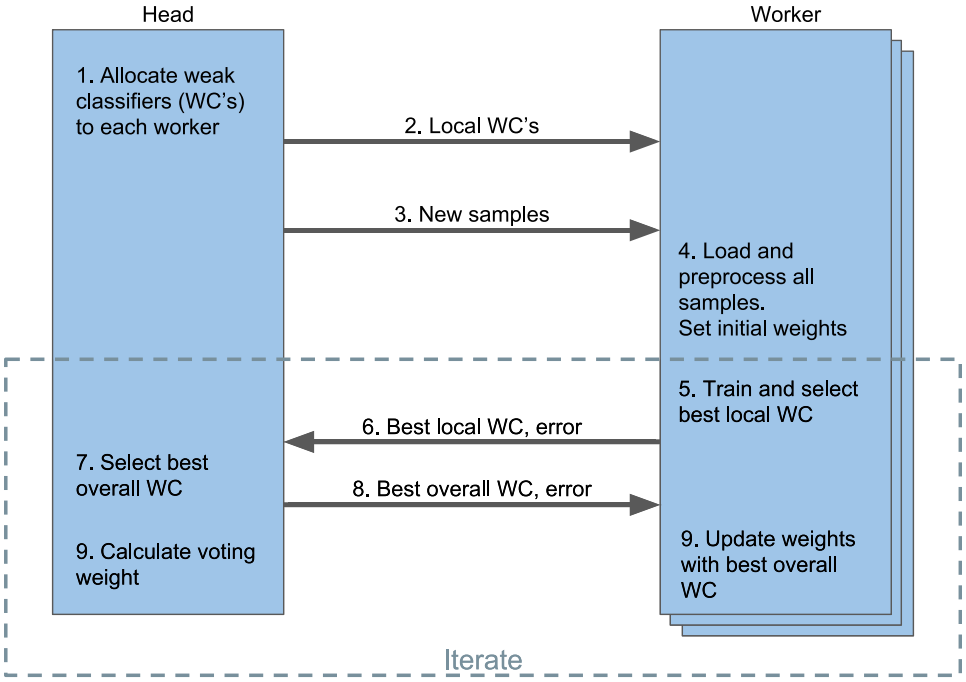


Fig. 6. Distributed boosted classifier training. WC stands for weak classifier.

the least misclassification rate, and the reports it to the head. The head then selects the best weak classifier from the set of weak classifiers it has received from the workers. The head reports this best overall weak classifier to each the workers, and the workers use it to update the sample weights according to Eq. (16) and proceed to the next iteration of AdaBoost.

6. The active learning framework

The Active Learning Framework (ALF) begins with the human agent providing an initial set of labeled samples to the framework, which uses the samples to create an initial classifier. These initial samples would obviously be strongly positive and strongly negative exemplars of the object for which a detector is being designed. We refer to this step as *Detector Initialization*.

Subsequently, ALF examines a set of unlabeled training samples that it must partition into two subsets, one for those samples that ALF can label with confidence, and other for those samples that it is not so sure about. The samples in the latter category will generally be those that are too close to the decision threshold. This sampling process can be made to depend on the distance of a sample from current best decision threshold. We will refer to this action by ALF as *Query Selection*.

The training samples that the Query Selection step designates as requiring labels are presented to the human agent, and are then deposited in a buffer along with their newly-acquired labels. We refer to this ALF action as *Oracle Annotation*.

Subsequently, using the human supplied annotations, ALF must update the decision thresholds using the AdaBoost logic. We refer to this ALF action as *Classifier Update*.

After Detector Initialization is complete, the Query Selection, the Oracle Annotation, and the Classifier Update steps must be repeated until a satisfactorily accurate detector is created, as illustrated in Fig. 7.

The subsections that follow elaborate on each of these four ALF steps.

6.1. Detector initialization

As mentioned, ALF is initialized with a small number of strongly positive and strongly negative training samples supplied by a human. To find these initial samples, we created a plugin for QGIS, a popular open-source desktop geographic information system. Users can explore satellite images in QGIS, visually locate an object, and use the plugin to extract a patch that contains the object. The plugin presents this patch to the user and allows the user to specify the window around the object. To somewhat desensitize the training process to the precise placement of the object inside the window, in addition to extracting the user-specified scanning window as a positive sample, ALF also extracts eight more training samples that are one pixel off of the original user-specified sample.

For collecting negative training samples, when the user sees a portion of the image with no objects, a mouse click by the user delineates a patch in which all of scanning windows are deemed to be negative training samples.

6.2. Query selection

For the Query Selection step, ALF shows to the user not just the scanning window for which it cannot make a decision with confidence, but the entire image patch from which the scanning window was extracted.[8] Looking at the larger image patch while supplying a label for the training sample in question allows the user to take advantage of the context. It is widely known in computer vision that the ability of a human to discern objects at the threshold of detectability depends significantly on the image context surrounding those objects. Patches are extracted by one of two methods: grid-based and road-based.

In the grid-based method, a grid of possible patches is created that covers the satellite image, and the patches are extracted in random order. Fig. 8(a) illustrates a grid of candidate patches for random selection overlaid over a portion of a satellite image. Note that in this particular depiction the patches do not align with the row-column axes of the satellite image. This particular example was chosen to highlight the fact that, in general, patch orientations may depend on what logic is being used for the detection of the objects. For example, if the object in question is a tall structure and its shadow can be used for recognizing the object, the shadow angle may be derived from the satellite image

[8] See Section 4.1 for the distinction between image patches and scanning windows.

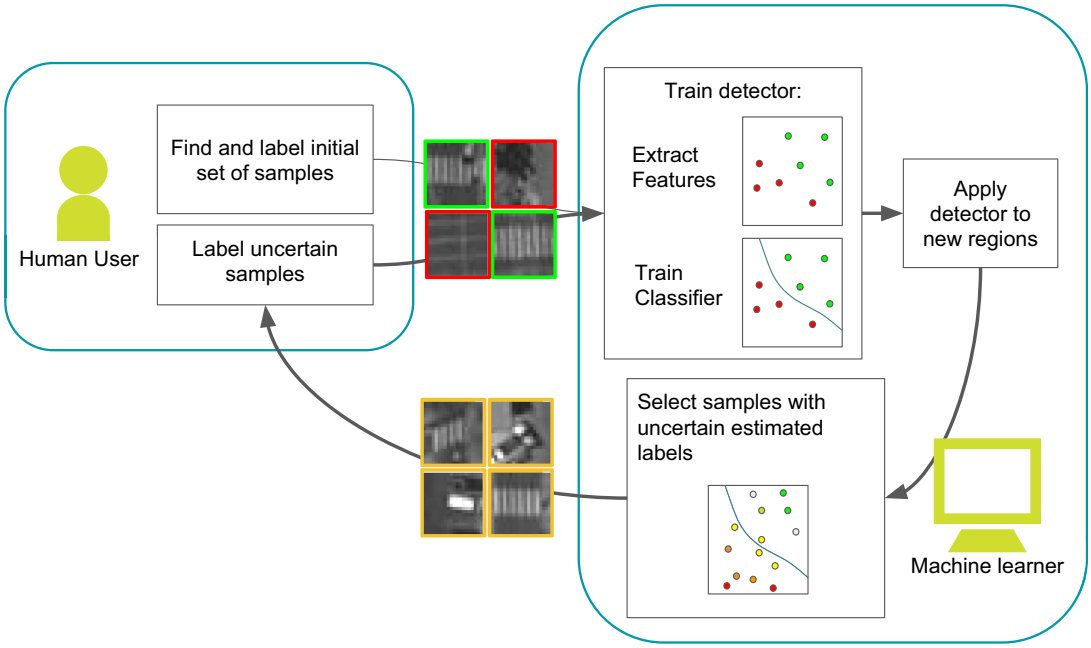
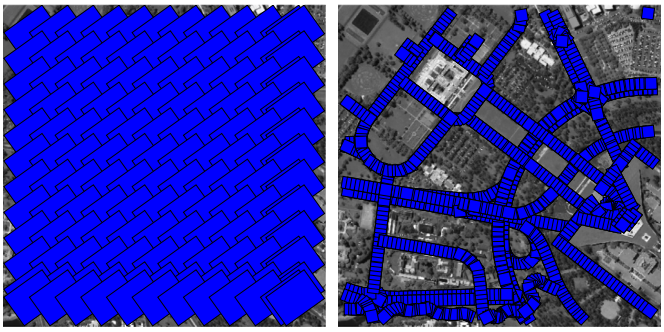


Fig. 7. Active learning framework.



(a) Grid-based patch extraction (b) Road-based patch extraction

Fig. 8. Shown at left are the candidate image patches for a grid-based approach to patch extraction. Patches are selected randomly from this grid for collecting the training data. The orientation of the patches may be dictated by considerations such as the shadow angle for an object. Shown at right are the patches that are extracted only along the OSM-delineated roads as projected into a satellite image. Note that the patches shown along the road are overlapping and the large patches are simply the last to be rendered.

meta-data, and the detector design for such a case may include extracting the information contained in the shadow of the object.[9]

The road-based approach to patch extraction is the same as was presented previously in Prakash et al. (2015). In this approach, the patches are extracted by following along a road specified by, say, the OpenStreetMap. As explained in Prakash et al. (2015), the roads are projected into the satellite images. Subsequently, image patches that align with the roads are extracted at regular intervals along the centerlines of the roads. Fig. 8(b) illustrates the extracted patches overlaid on a portion of a satellite image. Many of the objects of interest for geolocalization are located along roads, and the road-based approach allows ALF to significantly limit the search space. Furthermore, the objects of interest are often aligned with the road (which for the case of crosswalks means that the crosswalk stripes are likely to be parallel to the road centerline), so this approach also brings such objects into what may be referred to as a canonical orientation.

As was described earlier in Section 4.1, each randomly selected image patch is scanned with a moving scanning window and the detector, trained using AdaBoost on the labeled samples collected so far, is applied to each such window. The classification score for each such window is normalized to $[-1, 1]$, with scores close to -1 and $+1$ indicating that ALF is confident about those labels. On the other hand, scores closer to 0 indicate a lack of such confidence and such training samples are presented to the human agent for annotation.

If the logic described above is used without care, it can result in too many redundant samples. This is particularly an issue for negative training samples because it is so easy to generate a large number of them. To eliminate such negative training samples from further processing, only those samples that are *not* spatially adjacent in the dot representation of the scanning windows are retained, as illustrated by the example in Fig. 9.

6.2.1. Distributed query selection

As mentioned earlier, finding positive samples of infrequently occurring object over large swaths of the earth can result in unacceptable latencies in the human-computer interaction we have described so far.

As described in Section 5.1, we address this problem by distributing the task of query selection over multiple query-selectors, each on a different virtual machine on a cloud platform, and coordinating the effort via a head node. Each query-selector scans different areas of the satellite imagery and sends its batch of training samples for which it needs human help in labeling to the head node. The head node collects such training samples and presents them one-by-one to the human agent for annotation. The head node additionally updates the classifier with the annotated samples and sends the new classifier to each of the query-selectors.

6.3. Oracle annotation

If a patch contains query samples, ALF presents the patch to the human oracle with the query locations marked with dots (Fig. 10). At this point the human oracle may mark any location in the patch as a positive sample or a negative sample. Generally, the locations the oracle marks are queries, however allowing the human to choose any location can correct misalignments. In many cases, all of the presented

[9] As the reader will see in Section 7.1, this is an important part of the logic we used for designing a detector for electric transmission-line towers.

(a) Uncertain samples (b) Without redundancy

Fig. 9. Eliminating redundant negative training samples. (a) shows the positions of the scanning window currently under consideration. (b) shows the retained negative samples. The orange dots are the locations with positive scores and the yellow dots are the locations with negative scores. Note that the user had previously marked all of these samples as negative. The uncertain samples are shown with dots, rather than the windows to avoid cluttering the visualization.

Fig. 10. Interactive training user interface used for oracle annotation. The samples that ALF is presenting the human oracle for annotation are represented with dots, with the color indicating the confidence (yellow: weakly negative, orange: weakly positive, red: strongly positive). The red window is a human-oracle-supplied negative sample. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

candidate training samples are negative; in such cases, the human oracle may press a key to label all such samples as negative and move on, as done in Abramson and Freund (2005) to accelerate the human interaction. If the queried samples cannot be conclusively labeled, the human oracle may ignore them and ALF will not use them for updating the classifier.

6.4. Classifier update

Ordinarily, the Oracle Annotation step would involve only one training sample at a time and the decision boundaries of the detector would be updated with each annotation. Our experience with satellite images has shown that it is best to update the decision thresholds in the classifier in a batch mode as described below.

Samples that have been labeled by the human oracle are collected in a buffer. When the buffer is filled, the samples are added to the previously collected labeled samples and the classifier is trained on all of the collected samples. Therefore the decision boundary is not actually updated every time a new sample is labeled. Updating the decision boundary with just a small buffer is necessary to ensure that redundant unlabeled samples are not presented to the user. If we keep the buffer size small, it is unlikely that too many redundant samples will be encountered before the decision boundary is updated.

7. Incorporating object-specific information in detector design

Our detector design methodology presented so far in this paper has been generic — generic in the sense that it can be applied to any object type. We can expect a wide class of object types to yield discriminatory Haar-like, LBP, and spectral features that can be exploited in an active learning based framework for detector design.

However, we can also expect that each object type may provide us with special opportunities that could be leveraged to enhance the performance of a detector that was originally based entirely on generic considerations.

For example, we know that electric transmission-line towers are constructed, for the most part, along long straight lines on the ground. So why not take advantage of that fact to fine tune the decision thresholds for this object detector. That is, why not carry out an initial detection with generic logic and then use the Hough transform to group together the detections along linear paths on the ground. Subsequently, we can lower the detection threshold along such linear paths to capture a few more detections.

For another example, we know that pedestrian crosswalks can only occur on the roads. So why not limit the application of the generic object detection logic in this case to image patches that are along the roads.

We believe such “higher level” considerations apply to a large number of man-made object types on the ground. So it becomes a question of how to best integrate those considerations with the generic design considerations we have presented so far.

The next two subsections illustrate how these higher level considerations can be brought into play for two different object types that would generally be considered to be at the two opposite ends of the detection difficulty challenge.[10]

7.1. The case of electric transmission-line towers

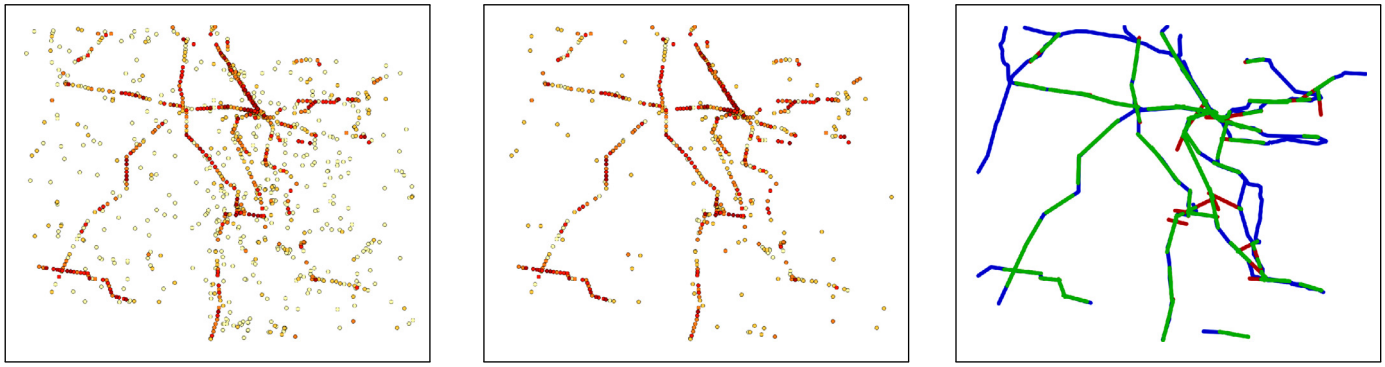
Electric transmission-line towers present a challenging test of the active learning framework because they can be difficult to detect in the relatively low-resolution satellite imagery even by humans. The towers are tall objects made from thin metal struts and their footprints on the ground are very small fractions of their height. For example in Fig. 12, the towers themselves appear often as small blurry gray blobs. Incorporating the shadow of the tower can increase the probability of detection, though, again referring to Fig. 12, we can see that the shadow may fall across different types of ground cover, which can complicate the role the shadow may play in the detection of the tower.

The shadows also fall in different directions depending on the time of day, but, fortunately, the metadata associated with the satellite images frequently contains the sun angle, from which we can easily derive the shadow angle. The challenge then becomes how to best incorporate the sun-angle information in the design of the detector.

We can think of the sun-angle (also referred to as the sun azimuth) as one of the higher-level object-specific consideration that ought to be integrated with the generic design considerations.

For a second higher-level consideration, we can exploit the fact electric transmission-line towers are placed mostly along long straight-line paths on the ground.

[10] Our choice of these two object types was dictated partially by the fact that one represents a tall 3D structure on the ground and the other a flat 2D pattern. The arguments we make for electric transmission-line towers could be applied to several classes of tall 3D structures on the ground. And the arguments we make for pedestrian crossings could be applied to several class of 2D man-made structures on the ground.



(a) Before TLA (b) After TLA (c) Detected Lines

Fig. 11. Transmission tower detections (a) before and (b) after transmission line augmentation. The color of the detection indicates the detection-score, with yellow being low and red being high. The linear arrangement of many sets of detections should be apparent in (a). Note that the detections shown here cover both training and testing regions. (c) shows the detected transmission lines (green for true positive, red for false positive) and the ground truth OSM transmission lines (blue) that were missed in the true positive detections. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

As it turns out, the logic needed for exploiting the sun azimuth is easy. All that needs to be done is to rotate the image patches so that they align with the direction of tower shadows. The rotation transform we use makes the shadow align length-wise with the horizontal axis on an image patch.

As the reader would expect, the logic need for invoking the second higher-level consideration — exploiting the fact that transmission-line towers occur mostly along straight-line paths — is slightly more complex and will be presented in the rest of this subsection.

The fact that transmission-line towers are constructed along long straight lines on the ground is evident from the example of the detections shown in Fig. 11(a). The figure also shows that we can certainly expect to find towers in what appear to be arbitrary relationship to the other nearby towers.[11] However, when a set of tower detections do form a straight line, we can exploit that fact to lower the detection threshold along that line on the ground to accept even more candidate detections that may otherwise be rejected. We refer to this approach to detecting and exploiting transmission lines as *transmission line augmentation (TLA)*. In what follows, we will present a two-stage algorithm for how this can be accomplished. The first stage finds an initial set of lines, while the second stage attempts to remove falsely detected lines.

7.1.1. Stage-1 detect an initial set of lines

We detect an initial set of lines using a modified version of the Hough Transform. To avoid finding lines that erroneously connect transmission tower detections over too great a distance we only look for lines within one small 4.5 × 4.5 km window at a time. This procedure is applied to a grid of overlapping windows that covers the entire ROI.

In the traditional Hough transform, the support for a line is determined by the number of points that fall on it, and the line is accepted if its support exceeds a threshold. In our modified approach, the support is instead defined as the sum of the confidence scores of the detections that fall on the line

$$lineScore(l) = \sum_{p \in P(l)} towerScore(p) \tag{20}$$

where the support set $P(l)$ is the set of tower detections located within a threshold distance of the line l.

This allows us to still consider lines that are supported by a large number of weak detections. The traditional Hough transform assumes that the points being fit exist with high confidence, so we should need to apply a threshold to discard weak detections before detecting lines.

[11] This may happen on account of the local topography or obstructions on the ground that may cause a transmission line to follow a locally zigzag path.

The Hough Transform outputs lines whose parameter values are quantized, so the lines are not necessarily the best fit to the detections that support the line. The points are within a threshold distance of the line, but the geometric error, i.e. the sum of the squared point-to-line distances, is not necessarily optimal. Therefore, we refine the line by using principal components analysis to minimize the geometric error.

7.1.2. Stage-2 score the detected lines without sharing tower detections

Among the transmission lines detected with the Hough transform, a common source of false positives is a single high-scoring tower detection in the support set of multiple transmission line detections. While the remainder of the support sets for such lines consists of low-scoring tower-detections, the single high-scoring tower detection is sufficient to push the *lineScore* above the threshold. Since it is rare for a transmission-line tower to be used for multiple non-parallel transmission lines, we drop such tower detections from the different support sets of the transmission lines. We calculate a new $lineScore^{(2)}$ for each detected line, this time imposing the constraint that no tower-detection may be in the support set of multiple line-detections. We approach this problem with a greedy algorithm:

(1) Sort the line-detections in a descending order according to the *lineScore* values.
(2) For each line-detection l in the sorted order, remove from the support set $P(l)$ any tower-detection that is in the support set of a higher-scored line-detection and calculate the new *lineScore* with this modified support set.

$$lineScore^{(2)}(l) = \sum_{p \in P^{(2)}(l)} towerScore(p) \tag{21}$$

$$P^{(2)}(l) = P(l) - \bigcup_{l' \in L_{>l}} P(l') \tag{22}$$

$$L_{>l} = \{l' | lineScore(l') > lineScore(l)\} \tag{23}$$

(3) Threshold the set of line-detections based on the new $lineScore^{(2)}$

Fig. 11(b) shows the transmission line detection results for the region in which the detector was evaluated.

7.2. The case of crosswalks

A crosswalk detector must contend with a number of challenges. At

Fig. 12. Example transmission tower detections (marked with red dots). The top row shows examples of true positive detections, the middle row false positive detections, and the bottom row false negative, i.e. missed, detections. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

0.5 m resolution, the striped pattern that generally distinguishes crosswalks is barely captured at the Nyquist limit; each stripe is often only one pixel wide. Crosswalks are often occluded, partially or fully, by cars, trees, and buildings. The ROIs over which the detectors in this paper are applied are arguably very large, so there is great variation in background clutter to which the detectors must become immune. In Prakash et al. (2015), we developed a road-following framework and demonstrated that it was possible to detect crosswalks with a human-engineered detector. In this work, we aim to demonstrate that the generic active learning framework presented in this paper can be used to achieve similar results.

We again utilize the road-following approach to create a crosswalk detector, this time using the approach for patch extraction in the query selection component of the active learning framework. This approach is powerful for the detection of object classes that occur on or near roads, e.g. crosswalks, because it not only reduces the search region, but it also often brings such object classes into a canonical orientation. As mentioned earlier, by canonical orientation for the case of crosswalks we mean when a crosswalk is located by following the centerline of a road, we can expect the stripes of the crosswalk to be parallel to the centerline.

8. Experimental results

We apply the trained detectors to a large database of satellite images by distributing the task over multiple virtual machines in a cloud platform in the same manner as described in Prakash et al. (2015). This approach has been tested both in our in-house cloud platform (RVL Cloud) as well as Amazon Web Services.

As is often the case with detection algorithms, a single object is often marked with multiple adjacent detections. Each detection has a classifier score, and, given a set of adjacent detections, we want to select only the detection with the maximum classifier score, i.e. we want to perform non-max suppression. Our non-max suppression algorithm simply removes any detection that is within a threshold distance of a higher-scoring detection.

The non-max suppression algorithm is applied to all detections found in the individual satellite images, with each detection location represented with geo-spatial coordinates rather than pixel coordinates. After non-max suppression is applied to each individual satellite image, the same procedure is applied to non-max-suppressed sets of detections in multiple overlapping satellite images to ensure that the objects are only detected once.

For crosswalk detection, we use a slightly modified variant of this non-max suppression. Rather than scoring a detection using the classifier score, we instead score the detection by the number of other detections within a threshold distance. As expected, a long crosswalk is often marked with a number of detections along its length. So we have found that the number of adjacent detections is a better indicator of the trustworthiness of a detection for the case of crosswalks.

8.1. Experimental results for electric transmission-line towers

As stated earlier, the transmission-line tower detector was used to

Table 2
Effect of transmission line augmentation (TLA). This table illustrates the transmission tower detector performance before and after transmission line augmentation.

	Precision (%)	Recall (%)
Before TLA	74	44
After TLA	80	50

generate the tower layer for a 150,000 sq. km area of Australia. For proprietary reasons, this is not the detector we present in this section. What we present here is an evaluation study of both the training procedures and detections achieved using a smaller number of images, to be specific four, over an area of size 1,000 sq. km These four images, which are spatially adjacent and were captured at about the same time, have been manually ground truthed by a human.

As described in Section 3, we used WorldView-2 8-band pan-sharpened, top-of-atmosphere-corrected satellite images with a spatial resolution of 0.4–0.5 meters per pixel. To initialize the active learning framework, the human user selected 25 image patches, each with a strongly positive sample of a tower, and 25 image patches with strongly negative samples (meaning samples that did not contain any towers). As was mentioned previously in Section 6.1, a patch containing one positive object sample actually becomes a source of nine positive samples since we also extract eight one-pixel shifted versions of the enclosing window as additional positive samples. Thus, the 25 initially selected image patches with positive samples end up supplying ALF with 225 positive samples for the object. Along the same lines, since *all* possible scanning windows in a negative patch are deemed to be negative samples of the object, ALF initialization ends up with 29,750 negative samples. The active learning framework was applied with the distributed query selection described in Section 6.2.1, with each query-selector drawing patches randomly from one of the four satellite images.

As mentioned earlier, we used a scanning window of size 36×120 for the transmission-line detector. From this window, 7,263 Haar, 700 LBP and 52 spectral features were extracted for a total of 8,015 features. During interactive training the classifier was trained with 300 iterations of the AdaBoost algorithm as described in Section 5. The AdaBoost algorithm was continued for another 100 iterations after the interactive session was over for stabilizing the calculated decision thresholds.

As is to be expected, interactive training (see Section 6.3) adds both positive and negative samples to the pool of samples collected during the initialization phase (See Section 6.1). Note that the process of converting each human labeled positive sample into 9 positive samples by eight one-pixel shifts of the sample actually labeled is also used during the interactive phase. For the particular evaluation study being reported here, we ended up with a total of 1,591 positive samples and 35,607 negative samples, including the initial samples. The interactive training phase took about 7 hours to complete.

The testing of the detector was carried out on the same four satellite images that were used for the training — but only after the training data was masked out in the manner described here. A human created a ground-truth set of transmission-line towers by visually scanning the four satellite images to locate the towers. This ground truth set was corroborated with the OpenStreetMap transmission-line tower layer to ensure accuracy. Subsequently, to separate out the training and the testing data, we created a training mask that marked out all pixels within 45 m of every positive sample used for initializing the active learning framework. This training mask also marked out all pixels within any of the patches scanned during training, regardless of whether the patch was shown to the human oracle. Because the patches were selected randomly during interactive training, this is equivalent to randomly partitioning the entire dataset into training and testing set as is often done in evaluating passive learning algorithms.

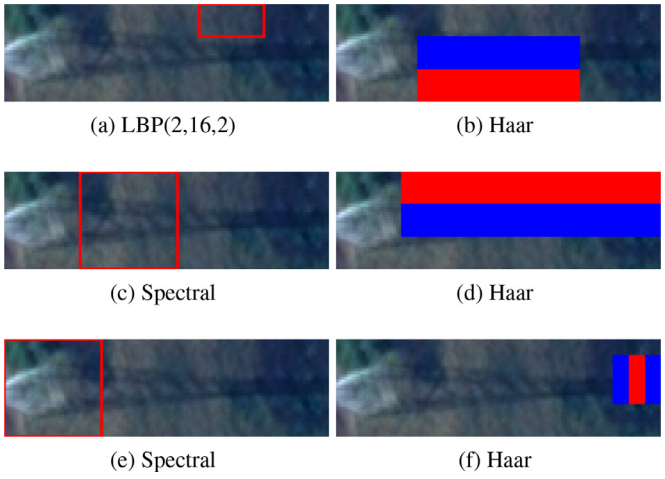


Fig. 13. Highest weighted selected features. For the selected LBP and spectral features, the block from which the features are calculated is highlighted with a red box. The caption of the LBP feature also shows the parameter values (LBP Label, *P, R*). The feature visualizations are overlaid on an example transmission tower. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The evaluation area covers 840.6 sq. km and contains 744 ground truth transmission towers. To evaluate the detector, we applied the detector to all four satellite images. Any detections or ground truth transmission towers that fell in the training mask were discarded. A detection was considered a true positive if there was a ground truth transmission tower within 45 m of the detection. Subsequently, we measured the recall and precision rates using the following definitions:

$$\text{Precision Rate} = \frac{\text{number of true detections}}{\text{total number of detections}} \tag{24}$$

$$\text{Recall Rate} = \frac{\text{number of true detections}}{\text{number of ground truth objects}} \tag{25}$$

When evaluating the transmission-line-augmented detections, we again applied the detector to the same four satellite images, discarded those detections that fell within the training mask, and then applied the transmission-line-augmentation to the remaining detections. The evaluation of transmission-line-augmentation presented unique challenges: Detections that fell within the training mask should not be considered for evaluation, but removing them before applying the transmission line detection could “disrupt” transmission lines that span regions both in and out of the training mask and thereby make it more difficult to detect such lines. Therefore the evaluation scheme described here provides a conservative estimate of the performance of transmission-line-augmentation.

Table 2 shows the improvement in detector performance provided by transmission-line-augmentation. Fig. 12 shows examples of true-positive, false-positive, and false negative detection. Fig. 13 provides insight into the detector itself, showing the highest weighted selected features.

8.2. Experimental results for crosswalks

As with the transmission-line tower detector, for evaluating the crosswalk detector we must use a relatively small manually ground-truthed dataset. “Relatively small” is in relation to the size of the region in Australia to which the detector was actually applied for creating a crosswalk object layer – a 180,000 sq. km region.

The evaluation dataset for the crosswalk detector consisted of randomly selected small sections from those portions of six satellite images that were rich in the occurrence of crosswalks. The six images were drawn from different regions of a 180,000 sq. km ROI in Australia. We

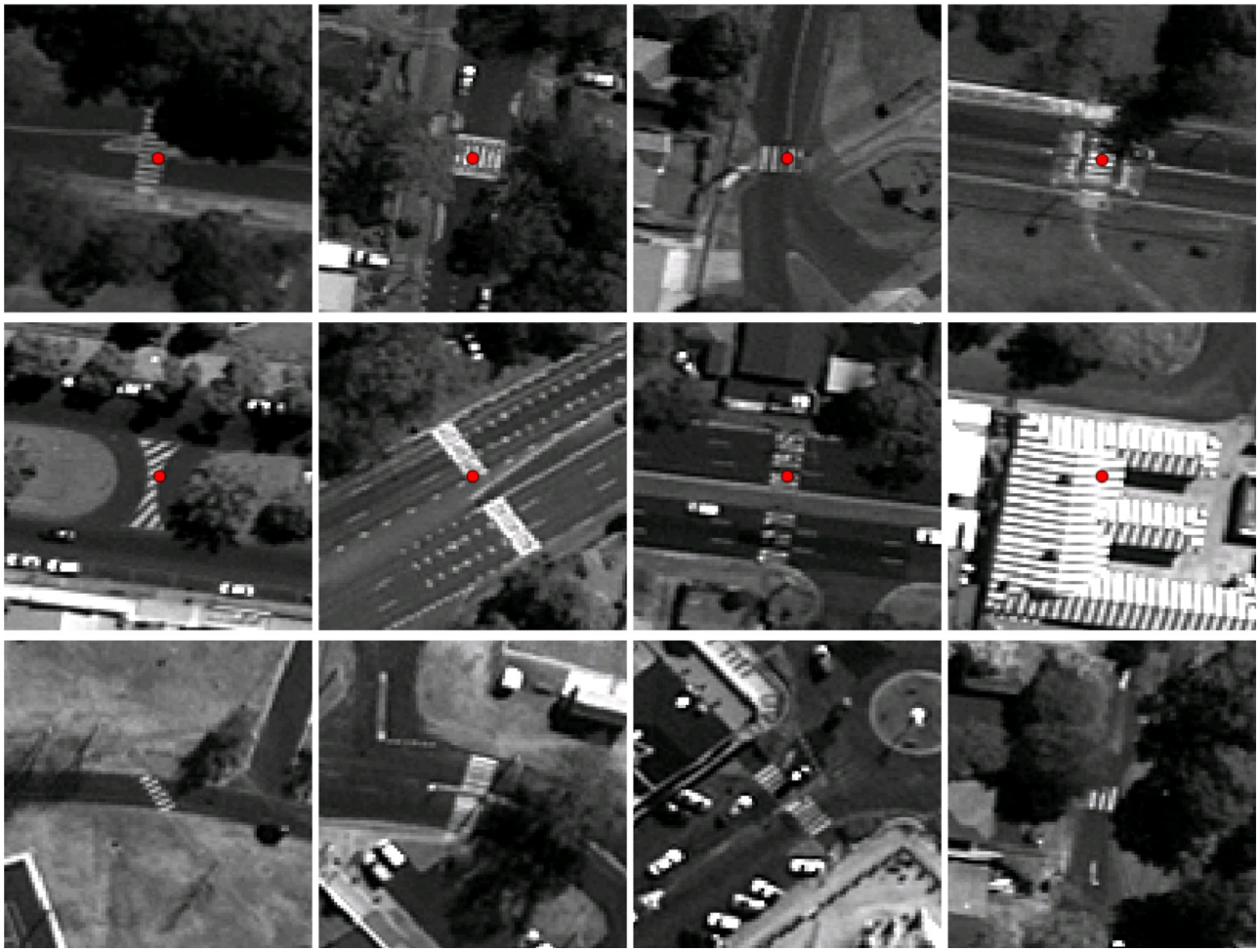


Fig. 14. Example crosswalk detections (marked with red dots). The top row shows examples of true positive detections, the middle row false positive detections, and the bottom row false negative, i.e. missed, detections. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Importance of different types of features for crosswalk detection. Crosswalk detectors were trained on only the Haar features, only the LBP features, and both the Haar and LBP features.

Features	Precision (%)	Recall (%)
Haar	87	12
LBP	90	48
Haar & LBP	92	72

used GeoEye-1 satellite panchromatic top-of-atmosphere-corrected imagery with a spatial resolution of 0.4–0.5 meters per pixel. For detector initialization (see Section 6.1), the human user located 25 positive and 25 negative example patches of crosswalks in a satellite image using the QGIS plugin as described earlier. The initial detector was trained on these samples. Then the human agent applied the interactive training using road-based patch extraction with the help of four query-selectors (see Section 6.2.1 for query selectors). This resulted in drawing patches randomly from the regions containing the most OSM roads in five of the six satellite images[12]. For this detector we used a scanning window of size 20 × 20, from which Haar and LBP features were extracted for a total of 5,977 features. The classifier was trained with 200 iterations of the AdaBoost algorithm.

At the end of the training, we had accumulated a total of 391 positive samples and 25,778 negative samples. Of these, 225 positive samples were extracted from the initially selected 25 patches for the 25 strongly positive samples. Recall from Section 6.1 that for a patch with a positive sample, we generate 8 additional positive samples through one-pixel shifts of the enclosing window to take care of the slightly different possible locations of the object within a positive scanning window. Thus a patch containing one positive sample becomes a source of 9 positive samples. The remaining 166 positive samples were collected during the interactive training with the shifting suppressed.[13] On the other hand, 24,025 negative samples were extracted from the initial negative patches, and the remaining 1 753 negative samples were extracted during the interactive training phase. As the reader will recall from Section 6.1 that when, during the initialization of ALF, the patch shown to the human oracle is negative, all of the scanning windows that can be extracted from the patch are added to the pool of negative samples. That is the reason for the large number of negative samples collected during the ALF initialization phase. The interactive training phase took about a 1.5 h to complete.

The crosswalk detector was evaluated on the same ground truth data as used in Prakash et al. (2015). The evaluation dataset spans

[12] We halted the interactive training before the framework could draw any patches from the sixth satellite image.

[13] In general, objects such as crosswalks that are too small in relation to the spatial resolution in a satellite image lend themselves well to the generation of additional positive samples from a human-labeled sample through small shifts only if they are strongly positive.

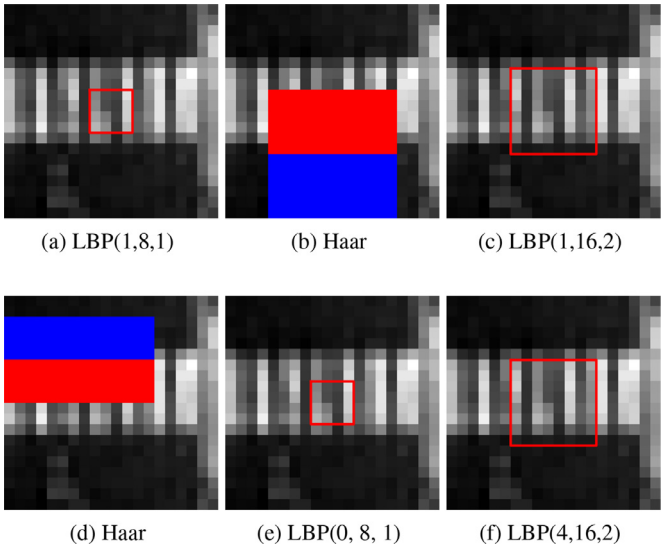


Fig. 15. The highest weighted features selected by the Active Learning Framework for the crosswalk detector. For the selected LBP features, the block from which the feature is calculated is highlighted with a red box and the captions show the associated parameter values (LBP Label, *P, R*). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Analysis of selected features for the crosswalk detector. Column 2 shows the number of AdaBoost iterations that selected each type of feature. The number in the parentheses shows the number of unique features used; a feature may be used in more than one iteration of the AbaBoost algorithm, though likely with a different decision threshold.

Features	Number of selected features (number unique)	Total normalized voting weight
Haar	119 (115)	0.567
LBP	81 (65)	0.433

104.1 sq. km and is comprised of eight separate test sections drawn from six different satellite images over the ROI. Each test section is a rectangular region with dimensions between 1 and 5 km. A human agent scanned the evaluation dataset exhaustively, marking any visible crosswalk that was in the vicinity of an OSM road. In total, the evaluation dataset contains 157 ground-truth crosswalks. A detection was considered to be a true detection if it was within 7.5 m of a ground-truth crosswalk. The crosswalk detector presented in this paper has a precision and recall of 92% and 72%, a significant improvement on the precision and recall of 89% and 63% in Prakash et al. (2015). See (Fig. 14) for examples of the crosswalk detection results.

8.2.1. Feature comparison

To understand how well Haar and LBP features can describe the objects by themselves, we trained two different crosswalk detectors, one based on just the Haar features and the other based on just the LBP features. Table 3 compares the performance of these detectors on the evaluation ground truth data. Note that the samples used were those collected while using the active learning framework to train the detector with Haar and LBP features. When creating the detector with only Haar features, the LBP features for each sample were discarded before training. A similar process was used to create the LBP-only detector. It is interesting to note that LBP features have a relatively high detector performance, which might be expected considering the distinct striped texture of a crosswalk. Though, of course, texture alone is not enough to characterize a crosswalk, as can be seen by the significantly improved performance of the detector utilizing both sets of features. Fig. 15, which shows the six highest weighted features, also illustrates how both types of features contribute to creating a strong detector, while Table 4 quantifies the contribution of each feature type to the overall detector.

9. Conclusion

In this paper, we have demonstrated how active learning can be used to create detectors for infrequently occurring objects over wide swaths of the earth covered by hundreds of satellite images. We demonstrated that active learning can significantly reduce the human annotation burden since a human's help is sought for only those candidate samples whose class label the system is not certain about. Since scanning satellite images for the presence/absence of objects is fundamentally a serial operation, even an active-learning based framework may result in unacceptable latencies in the human–computer interaction needed for eliciting the annotations. Our paper describes how we solved this problem through a cloud-based implementation of the framework in which several virtual machines simultaneously work on the image data over the different portions of a region of interest in order to more quickly generate the samples that require human annotation.

We have demonstrated the power of our approach on two very different types of objects — pedestrian crossings and electric power transmission-line towers. Considering how different these two object types are, we have reason to believe that our framework is quite generic with regard to the types of objects it can be used for. Obviously, each different object type is likely to require some customization of the overall framework, which may take the form of some preprocessing logic and/or some post-processing tuneup. For the two object types we addressed in this paper, the customization for pedestrian crossings consisted of limiting the search to roads, and, for the case of transmission-line towers, the customization consisted of exploiting properties of the spatial layout of such towers on the ground.

Acknowledgment

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory, contract FA8650-12-C-7214. The U.S Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

References

Abe, N., Mamitsuka, H., 1998. Query learning strategies using boosting and bagging. Proceedings of the International Conference on Machine Learning. 5. pp. 1–9. arXiv:1011.1669v3.
Abramson, Y., Freund, Y., 2005. Semi-automatic visual learning (seville): a tutorial on active learning for visual object recognition. International conference on computer vision and patternrecognition. (CVPR'05), San Diego.
Ahonen, T., Hadid, A., Pietikäinen, M., 2006. Face description with local binary patterns: application to face recognition. IEEE Trans. Pattern Anal. Mach. Intell. 28 (12), 2037–2041.
Balcan, M.F., Beygelzimer, A., Langford, J., 2006. Agnostic active learning. Proceedings of the International Conference on Machine Learning. 23. pp. 65–72.
Beygelzimer, A., Hsu, D., Karampatziakis, N., Langford, J., Zhang, T., 2011. Efficient active learning. Proceedings of the International Conference on Machine Learning.
Cohn, D., Atlas, L., Ladner, R., 1994. Improving generalization with active learning. Mach. Learn. 15 (2), 201–221.
Crawford, M.M., Tuia, D., Yang, H.L., 2013. Active learning: any value for classification of remotely sensed data? Proc. IEEE 101 (3), 593–608.
Dasgupta, S., Hsu, D., Monteleoni, C., 2007. A general agnostic active learning algorithm. NIPS 1–8.
Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-Line learning and an application to boosting. J. Comput. Syst. Sci. 55 (1), 119–139.
Gao, B.C., 1996. NDWI - A Normalized difference water index for remote sensing of vegetation liquid water from space. Remote Sens. Environ. 58 (3), 257–266.
Grabner, H., Nguyen, T.T., Gruber, B., Bischof, H., 2008. On-line boosting-based car

detection from aerial images. ISPRS J. Photogramm. Remote Sens. 63 (3), 382–396.
Hanneke, S., 2014. Theory of disagreement-Based active learning. Found. Trends Mach. Learn. 7 (2–3), 131–309.
Kak, A. C., 2012. AdaBoost for learning binary and multiclass discriminations. URL:https://engineering.purdue.edu/kak/Tutorials/AdaBoost.pdf.
Kak, A. C., 2016. Measuring texture and color in images. URL:https://engineering.purdue.edu/kak/Tutorials/TextureAndColor.pdf.
Ma, H., Qin, Q., Shen, X., 2008. Shadow segmentation and compensation in high resolution satellite images. Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS). 2. pp. 1036–1039.
Myneni, R.B., Hall, F.G., Sellers, P.J., Marshak, A.L., 1995. The interpretation of spectral vegetation. IEEE Trans. Geosci. Remote Sens. 33, 481–486.
Ojala, T., Pietikäinen, M., Mäenpää, T., 2002. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. 24 (7), 971–987.
OpenStreetMap, 2017. Planet dump retrieved from URL:https://planet.osm.org. URL:https://www.openstreetmap.org.
Prakash, T., Comandur, B., Chang, T., Elfiky, N., Kak, A., 2015. A generic road-Following framework for detecting markings and objects in satellite imagery. J. Sel. Top. Appl. Earth Observ. Remote Sens. 1–13.
QGIS geographic information system URL:http://qgis.osgeo.org.
Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S., 1998. Boosting the margin: a new explanation for the effectiveness of voting methods. Ann. Stat. 26 (5), 1651–1686.
Settles, B., 2012. Active learning. 6 Morgan&Claypool.
Smith, S., 1997. The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing.
Tuia, D., Volpi, M., Copa, L., Kanevski, M., Munoz-Mari, J., 2011. A survey of active learning algorithms for supervised remote sensing image classification. IEEE J. Sel. Top. Signal Process. 5 (3), 606–617.
Viola, P., Jones, M., 2004. Robust real-time face detection. Int. J. Comput. Vis. 57 (2), 137–154.
Wolf, A.F., 2012. Using WorldView-2 Vis-NIR multispectral imagery to support land mapping and feature extraction using normalized difference index ratios. Proceedings of the SPIE Defense, Security, and Sensing. International Society for Optics and Photonics.