A Generic Road-Following Framework for Detecting Markings and Objects in Satellite Imagery

Tanmay Prakash, Bharath Comandur, Tommy Chang, Noha Elfiky, and Avinash Kak

***Abstract*—In order to construct algorithmic solutions to the problem of geolocalization of user-generated photographs and videos, one must first populate a geographic database with the different types of objects and markings that are likely to be seen in the photographs and videos. Toward that end, this paper presents a framework for detecting and labeling objects in satellite imagery. The objects that we are interested in are characterized by low-level features that exist mostly at or beyond the limits of spatial resolution in the satellite images. To deal with the challenges posed by the extraction of such features from satellite imagery, we confine our search to the vicinity of the OpenStreetMap (OSM) delineated roads in a geographic area. The OSM roads are projected into the satellite images through inverse orthorectification. As an illustration of the performance of the object detection framework presented in this paper, our system can detect pedestrian crosswalks in a 200 000 sq. km. region of Australia (that is covered by 222 satellite images) with a recall rate of 63% and a precision of 89% (evaluated on a 100 sq. km. subregion). All of the computer processing for this result takes a total of 6 h on our in-house cloud computing framework that consists of five nodes, each an off-the-shelf high-end PC class workstation.**

***Index Terms*—Geolocalization, object detection, volunteered geographic information (VGI).**

I. Introduction

THE object detection research we report in this paper is motivated primarily by the needs of algorithms that geolocalize user-generated visual data such as photographs and videos. Geolocalization implies matching—matching the “objects” and their spatial interrelationships as seen in a photograph with the same in a geographic database. Establishing correspondences between a small number of objects in a photograph and the same in a geographic database can significantly reduce the search space for solving the geolocalization problem.

For example, in an urban setting, a spatial configuration involving a pedestrian crosswalk, a stand of roadside trees, and some topological attribute[1] related to how the nearby buildings are laid out in relation to the crosswalk and the trees can significantly narrow down the search space for localizing the camera used for the photograph.

Obviously, then, before the problem of geolocalization of photographs and videos can be solved, we must populate a database with known objects and markings on the ground. It is finding a solution to this problem through satellite imagery that is the focus of this paper.

Objects and markings that need to be detected in satellite imagery are obviously characterized by certain spatial patterns. For these patterns to be recognized, we must look for the low-level features that characterize the patterns. For the sort of objects and markings that are typically needed for solving a geolocalization problem, these low-level features often exist at or beyond the spatial resolution limits in commercially available satellite imagery.

The above-mentioned fact related to spatial resolution implies that great care must be exercised in how one goes about extracting the low-level features associated with the objects and markings on the ground. In the work we report in this paper, this care translates into extracting the low-level features only along and near the roads—a processing step greatly facilitated by the easy availability of OpenStreetMap (OSM) road maps for many parts of the world.

A critical issue related to the use of OSM road maps for guiding the search for the low-level image features (that characterize the objects) is how to project them into the satellite images. One has only two choices: 1) either you first orthorectify the satellite images before projecting the roads into the images or 2) you first inverse-orthorectify the road maps and then project the road data directly into the raw satellite data. By inverse-orthorectification, we mean the process of taking a geodesic coordinate pair as input and computing the corresponding row and column pixel indices in the original raw image.

Orthorectifying individual satellite images arguably presents a large computational burden when dealing with geographic areas that are as large as 200 000 sq. km. and that may be covered by hundreds of images. Even if we discount the computational effort involved, a more serious issue with orthorectification is that it introduces aliasing and other artifacts into the images that significantly impair the extraction of low-level features at the limits of spatial resolution in the imagery, and this will be demonstrated further in the paper. Hence, we have chosen the second option. That is, we first inverse-orthorectify

Manuscript received June 26, 2015; revised October 16, 2015; accepted October 20, 2015. Date of publication November 17, 2015; date of current version January 08, 2016. This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory under 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. The views and conclusion 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.

The authors are with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: tprakash@purdue.edu; bcomandu@purdue.edu; chang177@purdue.edu; nelfiky@purdue.edu; kak@purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTARS.2015.2495142

[1]We are referring to attributes that can easily be inferred from photographs, such as “left-right,” “far-near,” “clockwise order,” and so on.

the OSM road data and then project them into the raw satellite images.

As a consequence, our overall problem of object detection in satellite images boils down to the following: 1) inverse-orthorectify the OSM roads into the satellite images; 2) scan the roads in the images with a search window that straddles the OSM roads, the search window being sufficiently wide to compensate for any registration errors associated with projecting the roads into the images; and 3) invoke the computer vision algorithms for extracting the low-level features that define the objects of interest.

This paper presents a generic solution to the problem as stated above. Our solution is generic in the sense that its front-end—which consists of inverse-orthorectification of the road maps, projecting these roads into the images, and scanning along the roads—remains the same for the different types of objects for which we want to create a detection algorithm. The only part that changes from one object type to another is the computer vision module that is applied to the pixels in the search window.

Our paper presents two applications of this generic road-following framework for object detection: detecting pedestrian crosswalks and linear trees in *large* geographic areas. When we say large, we mean geographic areas that may be as large as 200 000 sq. km. in area and that are covered by hundreds of satellite images.

Requiring our detection framework to work over large geographic areas implies that any decision thresholds we use in our computer vision algorithms cannot be fine-tuned to individual satellite images. *Therefore, any detection performance numbers we present must be seen against a background of this fact.*

To give a reader a brief preview of the sort of performance numbers we are talking about, our system can detect pedestrian crosswalks from the satellite images of a 200 000 sq. km. region of interest (ROI) in Australia with a recall rate of 63% and a precision of 89%.

This paper is organized as follows. Section II presents a brief review of the relevant literature. Section III then presents a brief description of the satellite data we have used in our experimental work. Section IV presents the road-following framework that underlies the pedestrian crosswalk and linear trees detectors we present subsequently in Sections V and VI. The experimental results are presented in Section VII. Finally, we conclude in Section VIII.

II. Related Literature

A great deal of work has been done in extracting objects from remote sensing imagery, but the objects targeted have rarely been useful to the task of geolocalizing user-generated visual data. Much of the object detection literature has focused on buildings [1]–[4], roads [3], [5]–[7], and vehicles [8]–[17].

The work presented in [18] is one of the few to address road marking detection. The overall goal is actually road extraction, but it does include a crosswalk detector as a means of improving the road extraction. Their crosswalk detector applies morphological operators to a segmented image in order to create blobs, and then accepts blobs as crosswalks based on size and shape. Unlike our approach, their work makes no use of the striped pattern of a crosswalk. Furthermore, their work presents no quantitative analysis of the detection results, making comparisons difficult.

There has been some work in the past on the detection of trees and forests in general [19], [20]. The contribution in [19] requires very close range photography as it is based on the assumption that inter-pixel distances are of the order of 30 cm at most. The contribution in [20] models the treetops as circular clusters of pixels and then estimates the presence of those clusters and their associated radii by scale-space analysis of the NDVI values. The authors have shown their results on two time-lapsed satellite images to demonstrate the effectiveness of their algorithm. We believe that scale-space analysis, while ideal for processing an entire satellite image all at once, would be inappropriate for a road-following-based tree detector in which only those pixels that are on or in the vicinity of roads are examined.

It should also be noted here that crosswalks and linear trees are included in OSM under the tags, "crossing" and "tree row." However, these tags show up very infrequently in all of geographic areas we have looked at. Additionally, the tag "crossing" may also be used in OSM to indicate a pedestrian walkway delineated by two parallel lines perpendicular to the road—an object that is not of any interest to us.

Since our paper deals with object detection on or in the vicinity of the roads, and since there have been past contributions on vehicle detection in satellite imagery, here we cite those contributions that are relevant to our work. In general, the steps involved in vehicle detection consist of two stages: 1) feature extraction and 2) machine learning [8]–[17]. The features that have been investigated include: 1) Haar-like features [9], [12]; 2) histogram of oriented gradients [8], [9]; and 3) local binary patterns [9]. The machine learning techniques used include: 1) adaptive boosting [9], [12]; 2) partial least squares for dimensionality reduction [8]; 3) support vector machines [8]; 4) linear and quadratic discriminant analysis [11]; and 5) morphological share-weight neural networks (MSNN) [10].

The work described in [8] creates a two-stage cascade of SVM classifiers to detect vehicles. The initial feature set consists of 70 000 dimensions, but using partial least squares for dimensionality reduction, and ordered predictors selection for feature selection, the descriptor is reduced to a much lower dimension before the SVM classifiers are trained. The work presented in [10] creates a classifier using MSNN, which is essentially a linear shared-weight neural network where the features are morphological. In the work described in [12], the authors detect individual vehicles using a classifier trained by adaptively boosting Haar-like features, as well as queues of vehicles using a line extraction technique. Additional individual vehicles can then be extracted from the vehicle queues. In the work described in [17], the authors use morphological operations to detect vehicles against light or dark backgrounds.

With regard to the relationship of the road-following aspect of our work to how road networks have been used in the past, many vehicle detection and traffic flow estimation algorithms make use of GIS road network data [12]–[16]. The road-following approach described in [13], meant for helicopter-based overhead imagery, is similar to ours in that their approach extracts windows along the road and rotates the windows to a canonical orientation. However, in addition to using high-quality images recorded at low altitudes, their work uses orthorectified images, rather than raw images. As we have mentioned, and as we will discuss further, orthorectification can introduce aliasing artifacts and a slight loss of resolution that can hinder algorithmic detection of objects whose features are at the limits of spatial resolution.

We should also mention that in contrast with how road networks have been used by other authors in the past, we focus not only on the road pixels per se, but also on the pixels in the vicinity of the roads. Some of the objects and the markings we are interested in are not on the roads themselves, but on the sides of the roads. This distinction is important, because it means our detectors must handle a variety of clutter that is greater than what is seen on just the roads. At the same time, using a processing window larger than the width of a road means an object detector is less likely to miss the targets because of any misalignment between the OSM roads and the images.

III. Satellite Data Used in our Evaluation of Object Detectors

For the purpose of this study, satellite images are drawn from two large ROIs, one covering a 200 000 sq. km. area in the southeast of Australia and the other covering a 10 000 sq. km. area in the west coast of Taiwan. The former are panchromatic images at a spatial resolution of $0.4 - 0.5$ m per pixel taken by the GeoEye-1 satellite, whereas the latter are pansharpened images at a similar spatial resolution obtained from Worldview-2 satellite images. The pansharpening was performed using the Orfeo Toolbox [21]. From the corresponding OSM layers, we extract only major roads and exclude smaller service roads. As demonstrated in later sections, our road guided framework makes it easy to incorporate intelligent search strategies to eliminate many possible sources of confusion to the object detectors as well as to handle the issues of scalability to big data. Working within this framework, our detectors take only a few hours to run across these large ROIs using just a five-node cluster of PC-class machines.

IV. Road-Following Framework

As mentioned in Section I, we are only interested in detecting objects along and in the vicinity of the roads in large geographic regions. The road maps we use in our detection framework are those made available by OSM.[2] OSM is vector data. By that, we mean that each road segment is represented by its endpoints in geodesic coordinates. Therefore, to utilize the two data sources together—OSM road maps and the satellite images—we first need to bring them to a common coordinate system.

The traditional approach to this coregistration of a satellite image with the OSM roads involves orthorectification of the satellite image using algorithms that are now fairly standardized [22]. These algorithms convert the image from an array defined with pixel coordinates to an array defined with lat/long coordinates. Subsequently, the OSM roads can be projected straightforwardly into the orthorectified images using the geodesic coordinates of the end points of the road segments. However, the interpolation and other approximations associated with converting an array of pixel coordinates into an array of lat/long coordinates lead, in general, to a slight loss of spatial resolution and possibly aliasing artifacts in the orthorectified images.[3] We will visually illustrate the loss of resolution and the aliasing artifacts on a portion of a satellite image later in this section. For most applications of satellite imagery, the artifacts and the loss of resolution are not noticeable. However, in our case, when the objects and the markings we are looking for possess low-level features that are at very low spatial resolutions (near 0.5 m), such artifacts can significantly impair detector performance.

It is important to note that the loss of information due to orthorectification cannot be prevented by merely upsampling or using different resampling techniques. As an illustration of the effects of orthorectification in Section VII-A, the recall and precision of the crosswalk detector both drop by nearly 20% when orthorectified images are used.

Hence, in this paper, we use the other approach to the co-registration of a satellite image with OSM—the approach based on inverse-orthorectification. Since inverse orthorectification does not involve directly manipulating the pixels in a satellite image, it does not result in the aliasing artifacts and the resolution loss. Inverse orthorectification is also computationally much faster since now there is *no* need to process *all* the pixels in *all* the satellite images for a given ROI. Inverse orthorectification requires us to only project the OSM vector endpoints into the raw satellite images—the images themselves need not be modified in any manner. As those familiar with orthorectification would guess, the inverse orthorectification process also uses rational polynomial coefficients (RPCs) and the DEM that correspond to a particular satellite image. The RPC equations shown below model the pixel coordinates as ratios of polynomials over the geodesic coordinates [22]

$$r_n = \frac{f^r_{Num}(\phi_n, \lambda_n, h_n)}{f^r_{Den}(\phi_n, \lambda_n, h_n)}, \quad c_n = \frac{f^c_{Num}(\phi_n, \lambda_n, h_n)}{f^c_{Den}(\phi_n, \lambda_n, h_n)} \tag{1}$$

where each f is a different cubic polynomial in ϕ, λ, and h with 20 coefficients and where

$$\phi_n = \frac{\phi - \phi_{offset}}{\phi_{scale}}, \quad \lambda_n = \frac{\lambda - \lambda_{offset}}{\lambda_{scale}}, \quad h_n = \frac{h - h_{offset}}{h_{scale}}$$
$$r_n = \frac{r - r_{offset}}{r_{scale}}, \quad c_n = \frac{c - c_{offset}}{c_{scale}}. \tag{2}$$

In the above equations, ϕ, λ, and h denote the latitude, longitude, and height coordinates for a given geolocation. The variables r and c denote the corresponding pixel row and column values. The offsets and the scale factors are provided

[2]In order to extract the OSM vectors relevant to a satellite image, for each satellite image that covers any portion of a geographic ROI, we use the latitude and longitude offsets associated with the image along with its pixel resolution and dimensions to segment out the corresponding part from OSM.

[3]The formulas that convert pixel coordinates of the raw satellite images into lat/long coordinates utilize a digital elevation map (DEM) model of the portion of the earth's surface that corresponds to the satellite image. These formulas are iterative and involve various approximations for warping the image data onto the earth's surface, in addition to the interpolation needed to create a uniformly sampled array in the lat/long coordinates. The errors in orthorectification computations are exacerbated by any errors in the DEM model. Typical DEM models are at 30 m resolution.

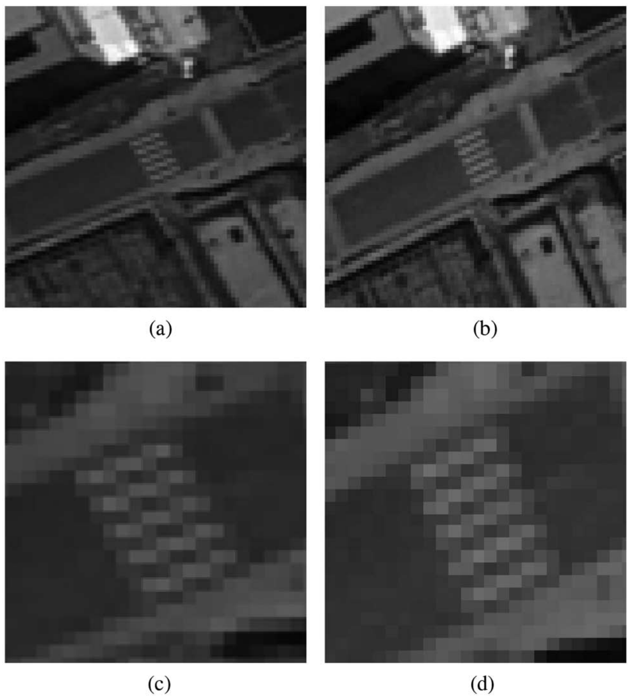

Fig. 1. Inverse-ortho means that we are directly using the raw image. Ortho means that the image is rectified into a lat/long array which entails a slight loss of resolution and introduces additional aliasing artifacts. (a) Ortho. (b) Inverse-ortho. (c) Ortho (crosswalk magnified). (d) Inverse-ortho (crosswalk magnified).

together with the RPC coefficients in the metadata associated with a satellite image. It is possible to use just the RPC-based equations without DEM by assuming an average elevation for the earth's surface covered by a satellite image. However, using height information from the DEM reduces the errors in projection.

For each road segment, we use the equations shown above to project the geodesic coordinates of the OSM vector endpoints into the raw satellite images.

Now that we have made clear what we mean by inverse orthorectification, it is time to illustrate how the "quality" of a satellite image at the limits of its spatial resolution can deteriorate by orthorectification. Figs. 1(a) and (b) are the same portions of a satellite image; the one in (a) is orthorectified and the one in (b) corresponds to inverse orthorectification (meaning what is shown is the raw image patch itself). Figs. 1(c) and (d) show just the crosswalk portions of the image patches in (a) and (b), respectively. Notice how the crosswalk strips in the orthorectified images in (a) and (c) suffer from a strong staircase effect due to image aliasing, *vis-a-vis* how they show up in (b) and (d).[4]

Once the two data sources are combined through inverse orthorectification, we can use the road network to guide our detectors. This is described as follows.

[4]If the reader would indulge us by viewing the four images in Fig. 1 from a distance, he/she will notice a relatively strong semblance of crosswalk strips in the two images in the right column. However, the same strips in the two images in the left column have a strong staircase effect caused by image aliasing. This staircase effect can make it more difficult to detect periodicities when the contrast differences are low.

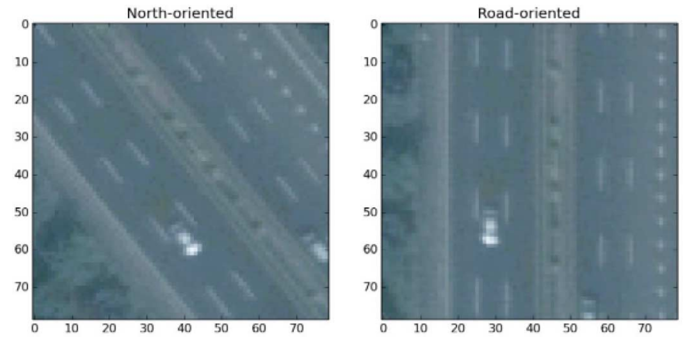

Fig. 2. Rotating the road to align with the vertical axis.

A. Road-Following Algorithm

Our object detectors use a moving window approach for extracting spatial and spectral features. A normal pixel-based approach to object detection on (and in the vicinity of) the roads would be to carry out a rectilinear scan of an image and center a search window at each road pixel. Instead, we use a more efficient approach in which the search window moves along the road segments, one segment at a time.[5] The road-following algorithm consists of the following steps.

1) Interpolate between the two endpoints of a road segment and select fixed points along the road as center points for the window.
2) The spacing between two successive points on a road segment (for placing the search window) depends on the object in question and the dimensions of the window. We ensure that there is sufficient overlap between successive placements of the search window so as not to miss the target object.
3) Since we know the actual geographic orientation of the road, we rotate the search window at each location so that, from the standpoint of computer processing, the road is aligned along the vertical axis. We use Lanczos interpolation [23], [24] to obtain a smooth rotation.

An important consequence of Step 3) is that it makes it easier to write the computer vision code for detecting objects in the search window. In general, how an object is oriented with respect to a road influences the logic of how an object is detected *vis-a-vis* the road. Orienting the road vertically eliminates at least one degree-of-freedom in the prescription of that logic. Fig. 2 shows an example of this step.

Obviously, any misregistration between the OSM data and the satellite imagery will propagate into our framework. However, choosing appropriate dimensions for the search window can mitigate the effects of small to moderate misregistration errors to a significant extent—as demonstrated by our results.

[5]More specifically, OSM is clipped according to the lat/long coordinates corresponding to a given satellite image. What one gets from such a clipped region is a list of straight road segments, with each segment defined by its two endpoints. Our road scanner scans through each item in this list.

B. *Cloud-Based Processing*

For fast processing speeds, the road-following framework, along with the logic of the detectors presented in Sections V and VI, runs in parallel on our in-house cloud computing cluster called the RVL-Cloud. This cluster consists of five high-performance physical nodes, each with up to 48 cores and 256 GB of RAM. A 36-TB network attached storage connected to the cloud contains all the satellite images for the datasets mentioned in Section III. In general, an ROI may be covered by hundreds of satellite images. Multiple virtual machines (VMs) are used to process all relevant satellite images, with each VM processing one image at a time. Since each VM is assigned dozens of cores, a given VM further parallelizes the task by chopping a large image into smaller sections, and distributing the sections among multiple processes, with each process running in a separate core.

We now present two object detectors that have been implemented by incorporating specially designed computer vision modules into the road-following framework described so far.

V. Pedestrian Crosswalk Detector

Typically, a pedestrian crosswalk appears in a satellite image as a rectangular portion of a road that contains pixels corresponding to the black and white stripes of the crosswalk. An example of a crosswalk is shown in Fig. 5. When such pixels are reliably sensed, the striped pattern exhibits a periodicity that is a fairly discriminative feature which can be leveraged by the crosswalk detector.

Since this detector is expected to work over large geographic areas that may be covered by hundreds of satellite images, there can be significant variation in how the pixels corresponding to a crosswalk appear in the images. Another complicating factor relates to the spatial resolution in the images. It is often the case that each stripe of a crosswalk is sampled by just one pixel.[6] This makes it difficult to estimate the periodicities associated with the pixels that fall on a crosswalk. Additionally, many crosswalks that have been subject to much traffic without repainting can appear faded. Occlusions from cars and trees can further confound detection. All of these are reasons for why an actual crosswalk on the ground may not be detectable as such in a satellite image. On the other side of the coin, we also have situations when noncrosswalk blobs of pixels may be detected as crosswalks. For example, rows of windows on buildings, lines of cars in parking lots, lane markings on wide roads, etc., all present high degrees of periodicity that may be mistaken for crosswalks if they appear inside a search window.

A. *Crosswalk Detection Algorithm*

The pedestrian crosswalk detector involves a cascade of three stages, with the first stage choosing *pixels of interest* from within a search window where one should test for periodicity, the second stage applying a periodicity detector to the image patches centered at the pixels of interest, and the third stage agglomerating those periodic pixels of interest into groups so that a crosswalk is marked with a single detection.

[6]When the sampling rate falls below one-pixel per stripe (as does occur in some regions of the world), the Nyquist limit is violated, which injects aliasing noise into the estimation of the periodicities.

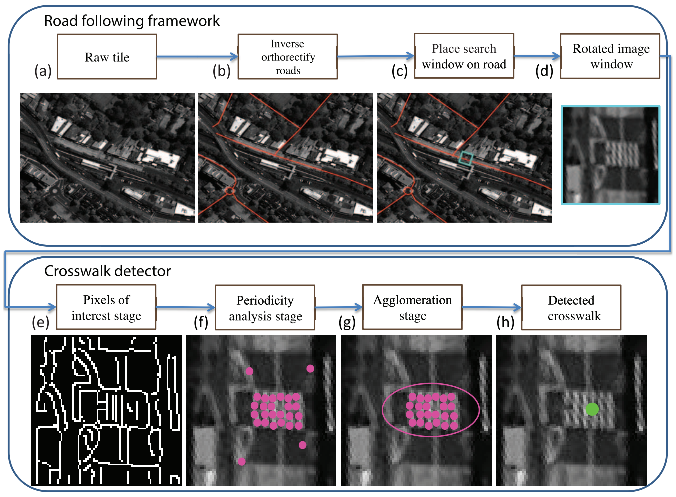

Fig. 3. Flowchart illustrating our road-following framework, as well as the downstream crosswalk detector. (e) Pixels of interest stage: Extract pixels that may fall on a crosswalk. (f) Periodicity analysis stage: Analyze pixels of interest for periodicity, culling those without significant periodicity. (g) Agglomeration stage: Agglomerate periodic pixels into groups, and use size and orientation to filter groups.

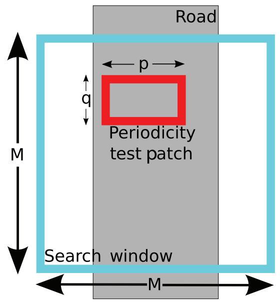

Fig. 4. Periodicity analysis stage. The search window placed on the road is shown in cyan. A patch of the image, shown in red, is extracted and analyzed for periodicity. The patches are only extracted around pixels of interest.

With regard to why only certain pixels should be chosen for testing for periodicities, note that, as mentioned earlier, a search window that is placed on a road point for crosswalk detection is larger than a typical crosswalk in order to compensate for residual misregistration errors between the projected roads and the satellite image. Therefore, it makes no sense to apply Stages 2 and 3 to every pixel in a search window. A pixel within a search window is considered to be a "pixel of interest" if it satisfies certain necessary (although by no means sufficient) conditions for a patch situated there to be a candidate for periodicity testing. These conditions are described in Section V-A1.

Fig. 3 illustrates the three-stage cascade, as well as how the detector fits into the road-following framework. The schematic in Fig. 4 illustrates placing an $M \times M$ search window at a point on the road segment and the placement of patches at the pixels of interest within the search window for the detection of periodicities.

We provide further details regarding the three stages of the cascade in the sections that follow.

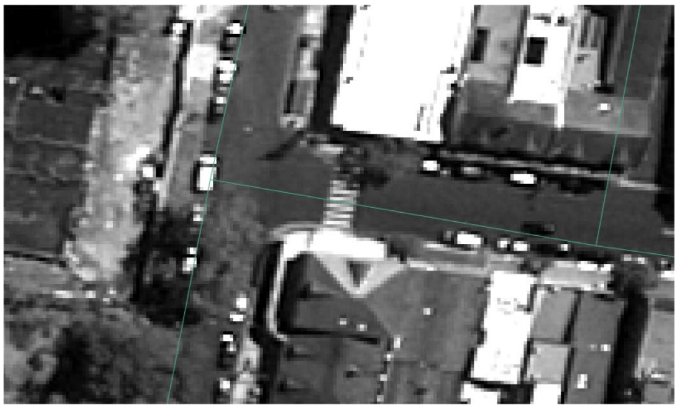

Fig. 5. Example pedestrian crosswalk in Australia from GeoEye panchromatic imagery. The green lines are the overlaid OSM roads.

1) Stage 1: Identifying Pixels of Interest Within a Search Window: The purpose of this first stage is to identity those pixels in a search window—as mentioned, we refer to these as *pixels of interest*—where an image patch may lend itself well to periodicity analysis. This stage is a first pass over the pixels in a search window in which we use simple heuristics to limit the number of locations that must be processed by the more computationally expensive periodicity analysis stage.

Depending on how close the frequency of stripes is to the Nyquist sampling criterion, a crosswalk may appear as strongly or weakly periodic. This stage uses the union of the two approaches described below to ensure that neither type is filtered out before the periodicity analysis stage.

Strongly periodic blobs are indicated by the presence of edge pixels. The edge pixels will correspond to the contrast differences between the white stripes and the background road portions of a crosswalk. So, if edges pixels are clearly discernible inside a candidate patch in a search window, those patches can subsequently be subject to the Stage 2 processing for the detection of periodicity. Pixels on strongly periodic crosswalks are detected using Canny edge detection. Plate (e) in Fig. 3 shows the edge pixels inside a search window. Subsequently, only the image patches centered at these edge pixels are analyzed for periodicity by Stage 2 that is described in Section V-A2.

Unfortunately, since crosswalks may be faded and/or because their periodicities may be beyond the Nyquist limit with regard to the image resolution, a detector that uses edge pixels for pre-filtering the candidate locations is likely to miss a significant number of crosswalks. In order to cope with this problem, our detector also uses a second approach that is geared specifically toward such cases. Before we describe the second approach in what follows, we want to add quickly that the two approaches are complementary—in the sense each works where the other fails.

About the second approach, when a crosswalk is faded and/or when its depiction in a satellite image is pushing the Nyuquist limit, it frequently appears as a gray band with relatively weak variations in the image gray levels. Fig. 6(c) shows an example of what we mean by a gray band with a weakly periodic

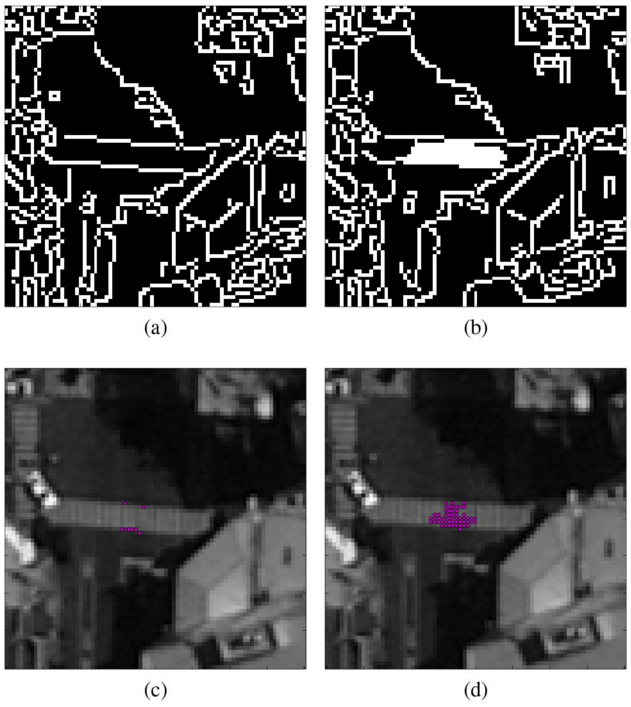

Fig. 6. Comparison of pixels of interest with just edge detection and with a combination of edge detection and integral image features. For the case when only edge detection is used, shown in (a) are the pixels of interest (white pixels) and in (c) the periodicity detections (pink dots). For the case when the pixels of interest are extracted with a combination of edge detection and integral image features, shown in (b) are the pixels of interest (white pixels) and in (d) the periodicity detections (pink dots). Note that only a few pixels of interest are included within the crosswalk when just the edges are used. The resulting periodicity detections are too few for the crosswalk to be detected reliably.

structure. If we were to apply an edge detector to such pixels, we may only be able to get some pixels on the border of the gray band, as shown by the white pixels in Fig. 6(a). Image patches centered at such pixels are unlikely to yield the periodicity information needed for identifying a crosswalk. In Fig. 6(c), the periodicity detections found at these edge pixels of interest are shown as magenta dots.

Therefore, in order to detect such weakly periodic crosswalk renderings, we must first search for appropriately oriented gray bands in a search window and then subject the pixels inside the gray bands to periodicity analysis as described in the Section V-A2. So, in this case, many of the pixels inside the gray band become our *pixels of interest*.

We detect the gray bands by applying appropriately shaped Haar-like derivative operators to the pixels inside a search window. The Haar features are convenient for ensuring that we look for a rectangular region for which the contrasts at the enclosing boundary correspond roughly to those present at the rectangular enclosure of a crosswalk and for which the contrast variations within the rectangular enclosure are within a certain bound. A threshold on the mean brightness within the regions above and below the candidate crosswalk location ensures that the values are low enough to be sections of road. A threshold on the variance within the candidate crosswalk ensures that the variance is low, as one would expect in a solid gray band. All of these features can be calculated quickly using an integral image representation of the pixels inside the search window [25].

Finally, from the pixels of interest derived from both methods—the edge-based pixels of interest and the Haar-based pixels of interest—we delete those where the brightness levels fall below a threshold. This ensures that they do not fall on areas that are too dark to be crosswalks, like asphalt.

2) Stage 2: Periodicity Analysis Stage: In this stage, we consider a patch around each pixel of interest returned by the first stage for periodicity analysis. The goal is to select just those pixels of interest where the periodicity is sufficiently strong.

Given a pixel of interest returned by Stage 1, the periodicity detector first extracts a $p \times q$ patch centered at that pixel. It then sums the pixel values along the vertical columns of the patch, creating a one-dimensional (1-D) p-sample signal. This integration along the road suppresses some of the noise, while maintaining the periodicity of the crosswalk stripes that are typically perpendicular to the road. The periodicity detector then calculates the discrete Fourier transform of the 1-D p-sample signal.

Two different thresholds are applied to the calculated 1-D Fourier transform: 1) one for the frequency and 2) the other for the magnitude of the transform itself at the selected frequencies. Through our prior knowledge about pedestrian crosswalks generally, and also our prior knowledge of spatial resolution in a satellite image, we know that, for valid patches, the peaks in the 1-D discrete Fourier transform must occur close to the highest frequency for which the DFT is calculated. We can, therefore, choose a frequency below which we can simply ignore the output of the Fourier transform calculator.

The second threshold is related to the magnitude of the peak in the frequency window that is let through by the first threshold. If the magnitude falls below this threshold, the peak is too weak for the patch to be considered periodic. This threshold is set relative to the maximum magnitude in the DFT, excluding the dc component. This allows the threshold to adapt to different illuminations (as caused by different sun angles, cloud cover, etc.). However, we do not allow the threshold to fall below a certain value; this suppresses those cases when the maximum magnitude is very low, as might happen when the variation in the patch is just noise.

Note that at this point, a single crosswalk is likely to be marked by multiple periodic pixels. For convenience, we refer to those pixels of interest where periodicity is detected as periodic pixels.

3) Stage 3: Agglomeration Stage: This stage not only groups together those periodic pixels that mark the same crosswalk, but also filters out false positives. Unlike the other two stages, which process one search window at a time, the agglomeration stage processes the periodic pixels from all the search windows in a given satellite image all at once. This stage agglomerates the periodic pixels into clusters, which can be further filtered.

The agglomeration proceeds by first creating a binary image for the entire satellite image, with all the periodic pixels set to 1. Subsequently, this binary image is subject to a moving window aggregation with a disk-shaped window. By this, we simply mean that we place a disk-shaped window at every location in the image, find the number of pixels set to 1 within the window, and return the resulting total as the output value at that location. By applying a threshold N_{group} to this output, small spurious groups of periodic pixels are removed. We discuss how to choose the value of N_{group} in Section VII. The next step is to find the connected components of the thresholded image. Each connected component serves as a mask that contains all the periodic pixels in a cluster.

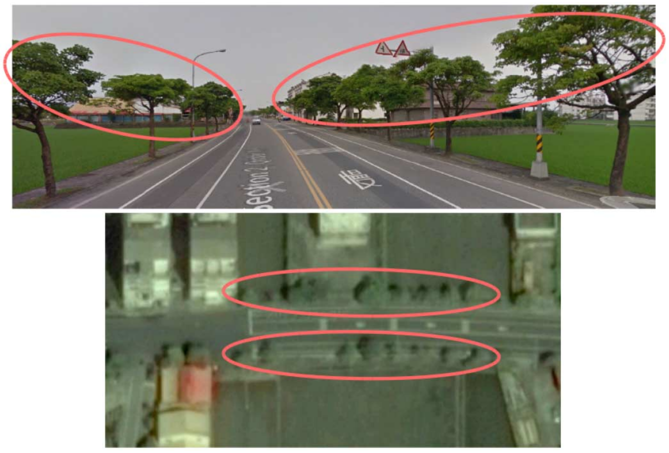

Fig. 7. Typical strips of roadside trees (circled in the red ovals) in Taiwan. The bottom image is the overhead view of the same trees shown in the top image. Image sources: Google Street View and Google Maps.

The detector finally filters the clusters based on their orientation with respect to the road. True crosswalks are, with a few exceptions, oriented at 90° to the road, where the angle is measured between the line perpendicular to the crosswalk stripes and the line along the center of the road. We find the orientation of the cluster by carrying out a principal components analysis (PCA) of the coordinates of the periodic pixels within the cluster. The angle of the cluster is calculated as that between the eigenvector with the largest eigenvalue, and the road through the cluster. If this angle falls below a threshold, the cluster is rejected. However, if the ratio of the eigenvalues is too close to 1, it implies that the angle measurement may not be accurate enough to warrant rejection. The centers of the accepted clusters are reported as the final crosswalk detections.

This marks the end of our presentation of the pedestrian crosswalk detector. We next describe the application of our generic road-following object detector to the case of detecting linear stands of trees along the roads.

VI. Linear Trees Detector

In this section, we describe roadside tree detection as another application of our road-based object detection framework. Specifically, we use the framework to detect roadside trees that are arranged in a linear fashion. Fig. 7 shows two typical strips of roadside trees. We refer to such formations of roadside trees as *linear trees* in the rest of the paper.

For the purpose of designing a computer vision algorithm for their detection, a *linear trees* object is defined as a collection of roadside vegetation pixels that satisfy the following two properties. 1) The pixels are bounded by a box that satisfies certain minimum constraints with regard to its length along the road and width perpendicular to it. Typically, we would want the bounding box for a linear trees object to possess a width that is between 2 and 4 m and a length greater than 20 m. 2) The pixel blob within the bounding box possesses a "blobular" shape. The shape property is necessary to separate a stand of trees from a uniform run of turf that is frequently found along major highways. We will detect the "blobular" property by applying erosion operators to the vegetation pixels to see if we end up with disconnected blobs. If a small number of erosions do not result in disconnected blobs, it is more likely that we are looking at linear turf along the road.

The computer vision algorithm that extracts such pixel formations consists of the following four steps.

1) Identify roadside vegetation pixels in the satellite image.
2) Sum the vegetation mass along the columns (meaning, in directions parallel to the road) and extract the width of the vegetation pixels perpendicular to the road.
3) Construct bounding boxes for the vegetation pixel blobs.
4) Apply erosion operators to the vegetation pixel blobs to identify the linear tree blobs.

We describe these steps in the next two sections.

A. *Identify Vegetation Pixels*

There does not exist sufficient information in the multispectral signatures in the satellite data that would allow for pixels to be directly identified as tree pixels. However, we can extract vegetation pixels along the roads fairly reliably. These vegetation pixels are subsequently subject to blob analysis for them to be labeled as tree pixels.

There exist the following two solutions to the identification of vegetation pixels.

1) Through the normalized difference vegetation index (NDVI) characterization of a pixel. NDVI is computed using the red and the near infrared (NIR) spectral bands and is thus effective at detecting vegetation. It has been shown that NDVI is directly related to the photosynthetic capacity of plant canopies [26]. The formula for NDVI is

$$\mathrm{NDVI} = \frac{\mathrm{NIR} - \mathrm{RED}}{\mathrm{NIR} + \mathrm{RED}}.$$

2) Through the visible atmospherically resistant index (VARI) characterization of a pixel. VARI is used to compute an approximation to vegetation fraction (VF) in the multispectral data [27]. Unlike NDVI, VARI does not need infrared and can be computed from the commonly available red, green, and blue visible spectral bands through the following formula:

$$\mathrm{VARI} = \frac{\mathrm{GREEN} - \mathrm{RED}}{\mathrm{GREEN} + \mathrm{RED} - \mathrm{BLUE}}.$$

In this paper, we say that a pixel is a vegetation pixel if its NDVI value exceeds some threshold V_{thres}. The resulting *vegetation segmentation* is a binary image in which the foreground consists of vegetation pixels. We discuss how to choose a value for V_{thres} in Section VII.

In general, NDVI values are high for all types of green vegetation including grasslands, shrubs, forests, and so on. Our blob and peak-value analysis described in Section VI-B eliminates many of these sources of high NDVI values. However, this analysis allows linear shrubs to be mistaken for linear tress. We consider those as examples of false positives.

B. *Identify Narrow Strips of Tree Pixels*

The logic for identifying narrow "blobular" strips of vegetation that would subsequently be labeled as *linear trees* is made particularly simple by the fact that, for the purpose of computer programming, our road-following framework always keeps the road vertical. This means that all we need to do is to examine the pixels within certain narrow columns on the two sides of a road and characterize what we find in those columns. No interpolation need be carried out for this column-based processing since all the pixels with the same column index belong to a single one-pixel wide strip parallel to the road. The question then becomes whether such column-based logic that makes inferences on the basis of the contents of multiple adjacent columns is feasible in the presence of residual registration errors between the OSM roads and the road pixels as they show up in the satellite images. Our results show that the trees can indeed be identified with usable levels of accuracy despite the registration errors.

To describe this column-based processing as applied to a search window, the portion of the image within the window is scanned one column at a time. For each column, we count the number of vegetation pixels in the column (See Fig. 14.). We then sort and visit the columns in decreasing order of these counts. For each column visited, we group the column with its neighboring columns on both sides when the *grouping criteria* described below are met.

However, before we describe the *grouping criteria*, we first define the following terms.

1) $mass(x)$: number of vegetation pixels in the image column x.
2) col: column index for a column with a high $mass$ value.
3) r: maximum expected radius of the trees.
4) $colL$: the column that is positioned r columns to the left of col.
5) $colR$: the column that is positioned r columns to the right of col.

The *grouping criteria* that have worked the best based on our experiments are

1) $mass(col) > M_{thres}$;
2) $mass(colL) < 0.5 \times mass(col)$;
3) $mass(colR) < 0.5 \times mass(col)$.

The purpose of Criteria 2 and 3 is to make sure that the boundary columns are near the edges of the vegetation strip if it is to be accepted as a tree strip. In other words, we do not expect many tree pixels outside the tree strip.

The group of columns from $colL$ to $colR$ will, in general, constitute a vegetation region. In order to label this region as a stand of trees, its shape along the road must be blobular—a shape characterization that can be determined by an iterative application of an erosion operator to the pixels. The number of applications of the erosion operator and the number of disconnected blobs one should expect for the *linear trees* label are experimental parameters that we will present in Section VII.

That brings to an end our presentation of the essential elements of the detector for linear trees objects. Section VII presents results on the performance of the crosswalk and the linear trees detectors on large ROIs.

VII. Experiments

As mentioned in Section I, we started with the goal of creating object detectors that could be applied to possibly hundreds of satellite images covering large geographic ROI's. Our underlying motivation was to be able to detect objects that could subsequently be used for the geolocalization of photographs and videos. Since the objects we are interested in possess features at the limits of spatial resolution in the available imagery, we decided to create road-based detectors (since the photographs and videos are highly likely to be recorded from locations on or in the vicinity of the roads anyway).

As our experimental results in this section demonstrate, detectors based on road-following logic possess usable performance despite the image-to-image variability in the satellite data and despite the fact that the low-level features representing the objects of interest (say, the crosswalks) are at the limits of spatial resolution in the satellite data.

In what follows, we will first describe the performance of the crosswalk detector and then that of the linear trees detector.

A. Detecting Crosswalks

As a part of IARPA's geolocalization project, the crosswalk detector described in Section V was applied to a 200 000 sq. km. ROI in Australia and the detected crosswalks supplied as an object layer to our prime contractor Applied Research Associates where it is being used in the development of matchers for geolocalization of photographs and videos.

The various parameters used for the crosswalk are chosen based on observations of the satellite data. For instance, the patch size used for the periodicity analysis stage is set to $10\text{ m} \times 5\text{ m}$ based on the observation that most crosswalks are about 5 m wide along the road and at least 10 m wide across it. The threshold on frequency is set to 0.33 cycles per pixel based on the observation that the imaged stripe frequency varies near the Nyquist limit of 0.5 cycles per pixel. Where intuitive definitions are not possible, we test various values and choose that value that provides desirable detector performance. Fig. 8 shows how the detector performance varies as the threshold on group size in the agglomeration stage is varied over $[0, 100]$ detections, leading to the choice of 15 detections as the threshold for the baseline detector. Here, we define false positives per sq. km. (FPPSK) as

$$\text{FPPSK} = \frac{\text{Number of false positive detections}}{\text{Ground truth area in sq. km.}}. \tag{3}$$

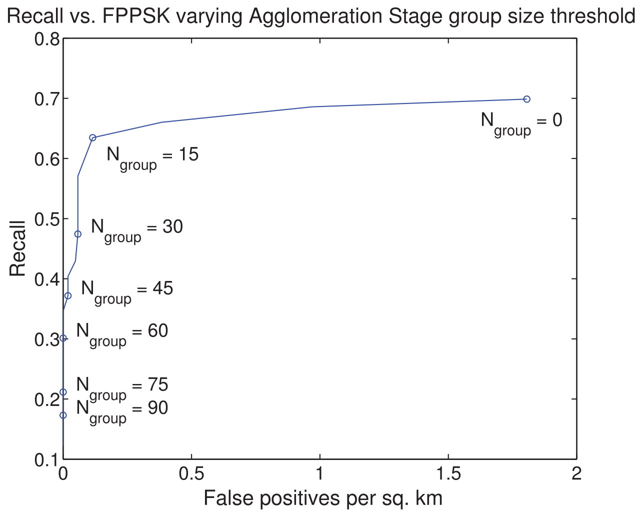

Fig. 8. Recall versus false positives per sq. km. for the crosswalk detector as N_{group} is varied. N_{group} is the threshold on the group size in the agglomeration stage.

To evaluate the crosswalk detector, we manually created ground truth for eight test sections in six satellite images over the ROI.[7] Each test section is rectangular with height and width between 1 and 5 km. The total area of all the test sections is 104.1 sq. km. A human operator generated the ground truth by visually scanning each test section in its entirety and accepting a crosswalk as a part of the ground truth if its location is in the vicinity of an OSM road. This check against the OSM roads—important since our road-following framework works off the OSM roads—is carried out by using the QGIS open-source tool to overlay the OSM roads on the satellite images. In total, 142 crosswalks were found in the ground truth sections.

For measuring the performance of the detector, the number of true positives is calculated as the number of detections within the test sections for which there is a ground truth crosswalk within a threshold distance. The recall and precision rates are defined as

$$\text{Precision Rate} = \frac{\text{Number of true detections}}{\text{Total number of detections}} \tag{4}$$

$$\text{Recall Rate} = \frac{\text{Number of true detections}}{\text{Number of ground truth crosswalks}}. \tag{5}$$

Overall, the recall and precision in the test sections were 63% and 89%, respectively. Examples of true detections are shown in Fig. 15. To better understand the effect of each stage, we tested the detector performance when each stage was removed and the others kept. When the pixels of interest stage (Stage 1) is removed, we instead simply apply the periodicity analysis to all running placements of 20×10 patches with their centers four pixels apart. When the periodicity analysis stage (Stage 2) is removed, we simply pass the pixels of interest directly to

[7]The only constraint on the test sections was that they be in urban and semiurban areas. This constraint was necessitated by the fact that crosswalks are relatively rare outside the urban areas. Note that crosswalk detection was applied to all of the satellite data covering 200 000 sq. km. It is only for the evaluation process that we limited the sections for manual groundtruthing to urban and semiurban areas.

TABLE I
PEDESTRIAN CROSSWALK DETECTOR PERFORMANCE

	Recall	Precision
Without Stage 1	52	89
Without Stage 2	33	0.3
Limited Stage 3	79	12
With all stages	63	89

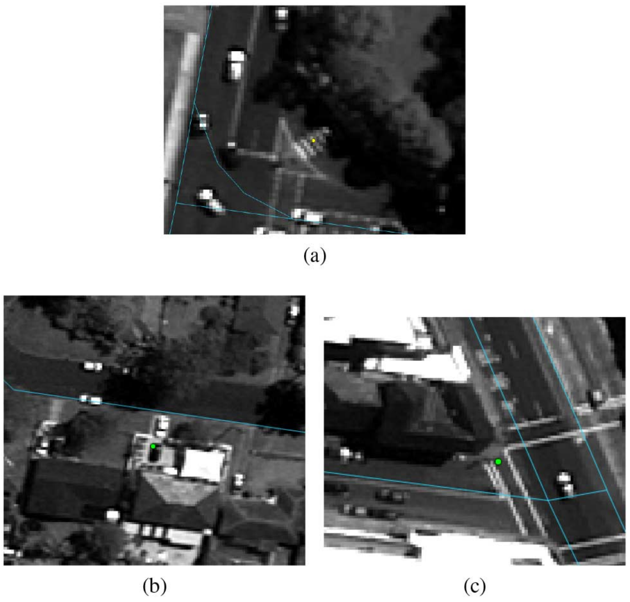

Fig. 9. (a) Missed detections and (b,c) false positives in Australia. The green dots are the detections, the yellow dot the ground truth, and the blue lines the OSM roads.

the agglomeration stage. Finally, rather than completely removing the agglomeration stage, we allowed it to form the groups of pixels, but we removed the step that culls groups based on size and orientation. Table I summarizes the results of these experiments. As the table shows, each step is important to the performance, especially the periodicity analysis stage.

As a test of the effects of orthorectification, the detector was also tested on the same regions, with exactly the same parameters, but using orthorectified satellite images rather than the raw images in our inverse orthorectification-based framework. With orthorectified images, the recall and the precision were 42% and 65%, respectively. These performance numbers speak for themselves with regard to how the loss of spatial resolution and the aliasing artifacts introduced into the images by orthorectification affect the detection of features at the limit of the available spatial resolution.

The detector misses those crosswalks that are not aligned perpendicularly to OSM road. This can happen when the crosswalk is painted diagonally across a road, and when the OSM road is not correctly aligned with the road in the satellite image. One example of the latter case is when the crosswalk is in a curved turn lane as in Fig. 9(a).

B. Detecting Linear Trees

As a part of the IARPA's geolocalization project, the linear trees detector described in Section VI was applied to a

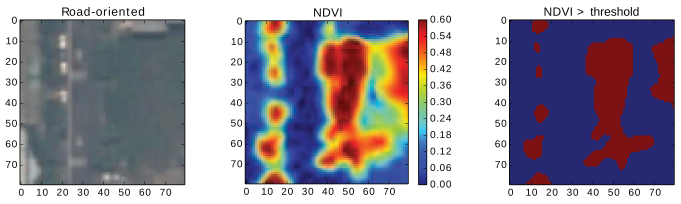

Fig. 10. Vegetation segmentation using NDVI. Left: 40 m × 40 m road-oriented search window; Center: NDVI feature map; Right: Vegetation pixels.

10 000 sq. km. ROI in Taiwan. The detected linear trees were supplied as an object layer to our prime contractor Applied Research Associates to support the development of matchers for geolocalization of photographs and videos.

For the rest of this section, we first mention how we determine the various parameters (V_{thres}, M_{thres}, and r) used in our linear trees detector. We then describe the ground truth collection and the evaluation protocols. Finally, we summarize the detector performance.

Recall from Section VI-A that V_{thres} is used to identify vegetation pixels from the portion of a satellite image inside the search window. We pick a lower bound threshold such that most of the vegetation pixels exceed that threshold. We have found this value to be around 0.4. Fig. 10 shows a typical 40 m × 40 m search window, its NDVI map, and the resulting vegetation segmentation.

To detect linear trees objects consisting of trees of diameter less than 7 m, we need to consider strips of width 3.5 m on both sides of the column col. (As defined in Section VI-A, col is a candidate column with high number of vegetation pixel counts.) Since r is the number of columns on either side of col and each pixel corresponds to 0.5 m, we set $r = 3.5/0.5 = 7$.

Finally, we must account for the fact that, when vegetation pixels form linear trees, there may or may not be any gaps between such pixel blobs along the road. To allow for both possibilities, we require at least 3/8 of the total pixels in col to be vegetation pixels. Since the search window height is 80 pixels in a 40 m × 40 m search window, we set $M_{thres} = 3/8 \times 80 = 30$.

Figs. 11 and 12 demonstrate the column grouping criteria using the parameter values mentioned. That is, $V_{thres} = 0.4$, $M_{thres} = 30$, and $r = 7$. Once a linear trees object is detected, we represent that detection by a geographic line consisting of two lat/long endpoints corresponding to the topmost and the bottommost tree pixels in col.

With regard to ground truth data, we manually annotate a 5 km × 5 km region from a typical satellite image. The annotation process involves manually placing 5 m × 40 m nonoverlapping rectangles on top of all true linear trees. Fig. 13 shows the ground truth rectangles and all the detections in a small area.

For performance metrics, we compute precision and recall rates as follows:

$$\text{Precision rate} = \frac{\text{Number of true detections}}{\text{Total number of detections}} \tag{6}$$

$$\text{Recall rate} = \frac{\text{Number of true detections}}{\text{Number of ground truth rectangles}}. \tag{7}$$

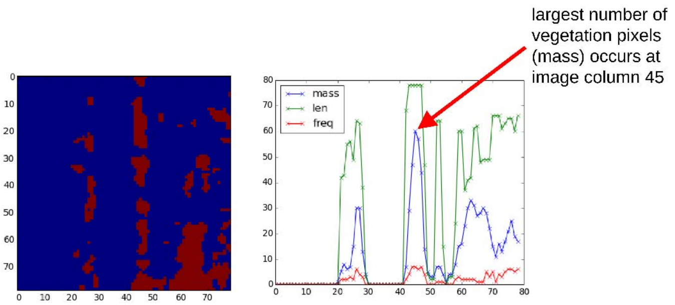

Fig. 11. Left: vegetation pixels. Right: plots of vegetation mass, length, and frequency against image columns. Here, length is the number of pixels between the topmost and the bottommost vegetation pixels. Frequency is the number of connected components.

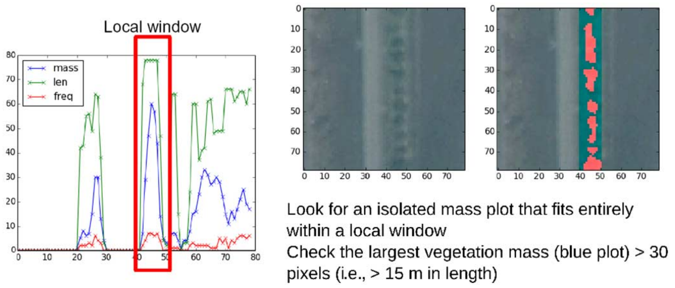

Fig. 12. Left: red rectangle shows a grouping of columns that satisfies the *grouping criteria*, defined in Section VI-B. Right: detected linear trees object is highlighted in red.

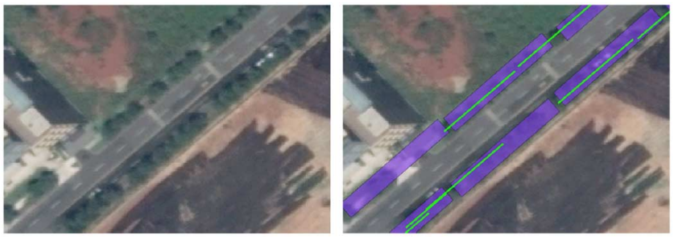

Fig. 13. Left: reference region with linear trees. Right: round truth rectangles are shown in purple and detections are shown as green lines.

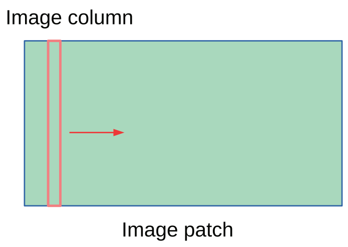

Fig. 14. Scan the image column-wise.

In order to compute precision and recall rates correctly, we need to account for the fact that a detection, as reported by the linear trees detector, may intersect multiple ground truth rectangles. When a single detection, which is a line object, intersects multiple ground truth rectangles, that is an example of multiple true detections and are counted as such. On the other hand, if a single ground-truth rectangle is intersected by multiple detections, those detections are counted as a single true detection.

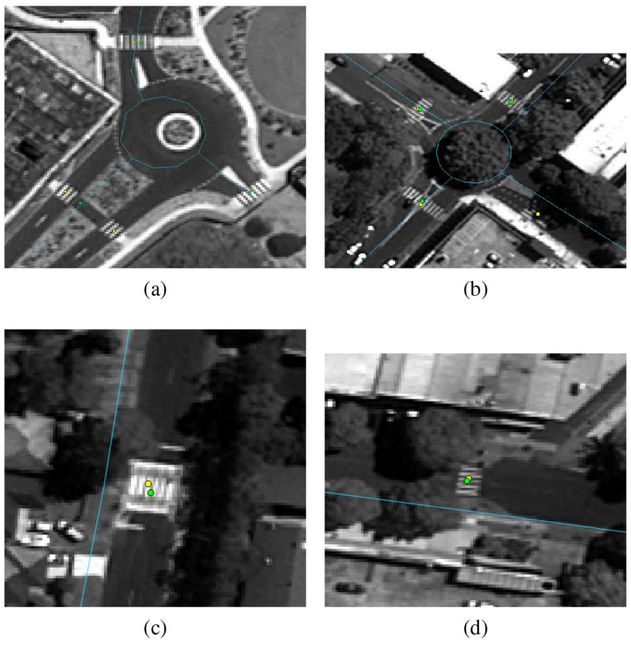

Fig. 15. True positive detections in Australia. The green dots are the detections, the yellow dots the ground truth, and the blue lines the OSM roads.

TABLE II
LINEAR TREES DETECTOR PERFORMANCE

Max # of erosions	Recall	Precision
0	60	57
1	47	69
2	51	67
3	53	67
4	54	67

Table II presents the precision/recall numbers for different degrees of erosion. As the reader will recall from Section VI-B, we apply the erosion operator to the vegetation pixels detected on the sides of the roads to discriminate between linear turf and linear trees. Roadside vegetation pixels that correspond to a linear stand of trees frequently have "blobular" appearance in overhead imagery. When a region of pixels is blobular, it is likely to become disconnected with iterative applications of the erosion operator. The first column of Table II shows the maximum number of erosion iterations allowed. To detect the presence of a *linear trees* object, we count the number of blobs at the end of each erosion iteration. If three or more blobs are present, we then label the object *linear trees*.

VIII. Discussion and Conclusion

The matching step in the algorithms for geolocalizing photographs and videos must of necessity be based on objects and markings on the ground. There is hence a need to create object layers for large geographic areas of the earth using, e.g., satellite imagery. Our paper demonstrates how such object layers can be created for areas that span hundreds of thousands of sq. km. and that are covered by hundreds of satellite images.

One main advantage of the road-based object detection framework we have presented in this paper is that it can easily be customized for the detection of any object (that is likely to be seen on or in the vicinity of the roads) by simply plugging into it a computer vision module that is appropriate for the object. We demonstrated this ease of customizability by showing two different types of detectors—a crosswalk detector and a linear trees detector—that are based on the same road-following framework.

A great benefit of detecting objects by just following the roads is that it significantly reduces the search space, besides reducing the occurrence of false positives. This is borne out by the performance numbers for the detectors presented in this paper. As shown in Section VII, our statistical evaluation indicates that our crosswalk detector works with a recall of 63% and precision of 89%.

References

[1] M. A. Maloof, P. Langley, T. O. Binford, R. Nevatia, and S. Sage, "Improved rooftop detection in aerial images with machine learning," *Mach. Learn.*, vol. 53, no. 1–2, pp. 157–191, 2003.

[2] O. O. Karadag, C. Senaras, and F. T. Y. Vural, "Segmentation fusion for building detection using domain-specific information," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 7, pp. 3305–3315, Jul. 2015.

[3] C. Ünsalan and K. L. Boyer, "A system to detect houses and residential street networks in multispectral satellite images," *Comput. Vis. Image Understand.*, vol. 98, no. 3, pp. 423–461, 2005.

[4] H. Mayer, "Automatic object extraction from aerial imagery—A survey focusing on buildings," *Comput. Vis. Image Understand/*, vol. 74, no. 2, pp. 138–149, 1999.

[5] A. Grote, M. Butenuth, and C. Heipke, "Road extraction in suburban areas based on normalized cuts," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 36, pp. 51–56, 2007.

[6] A. Grote, C. Heipke, F. Rottensteiner, and H. Meyer, "Road extraction in suburban areas by region-based road subgraph extraction and evaluation," in *Proc. Joint Urban Remote Sens. Event*, 2009, pp. 1–6.

[7] J. Yuan, D. Wang, B. Wu, L. Yan, and R. Li, "LEGION-based automatic road extraction from satellite imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 11, pp. 4528–4538, Nov. 2011.

[8] A. Kembhavi, D. Harwood, and L. S. Davis, "Vehicle detection using partial least squares," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 6, pp. 1250–1265, Jun. 2011.

[9] H. Grabner, T. T. Nguyen, B. Gruber, and H. Bischof, "On-line boosting-based car detection from aerial images," *ISPRS J. Photogramm. Remote Sens.*, vol. 63, no. 3, pp. 382–396, 2008.

[10] X. Jin and C. H. Davis, "Vehicle detection from high-resolution satellite imagery using morphological shared-weight neural networks," *Image Vis. Comput.*, vol. 25, no. 9, pp. 1422–1431, 2007.

[11] L. Eikvil, L. Aurdal, and H. Koren, "Classification-based vehicle detection in high-resolution satellite images," *ISPRS J. Photogramm. Remote Sens.*, vol. 64, no. 1, pp. 65–72, 2009.

[12] J. Leitloff, S. Hinz, and U. Stilla, "Vehicle detection in very high resolution satellite images of city areas," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 7, pp. 2795–2806, Jul. 2010.

[13] J. Leitloff, D. Rosenbaum, F. Kurz, O. Meynberg, and P. Reinartz, "An operational system for estimating road traffic information from aerial images," *Remote Sens.*, vol. 6, no. 11, pp. 11315–11341, 2014.

[14] C. K. Toth and D. Grejner-Brzezinska, "Extracting dynamic spatial data from airborne imaging sensors to support traffic flow estimation," *ISPRS J. Photogramm. Remote Sens.*, vol. 61, no. 3–4, pp. 137–148, 2006.

[15] X. Jin and C. Davis, "Vector-guided vehicle detection from high-resolution satellite imagery," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS'04)*, 2004, vol. 2, pp. 1095–1098.

[16] S. Hinz, J. Leitloff, and U. Stilla, "Context-supported vehicle detection in optical satellite images of urban areas," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS'05)*, 2005, vol. 4, pp. 2937–2941.

[17] Z. Zheng *et al.*, "A novel vehicle detection method with high resolution highway aerial image," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 6, pp. 2338–2343, Dec. 2013.

[18] C. Zhang, E. Baltsavias, and A. Gruen, "Knowledge-based image analysis for 3D road reconstruction," *Asian J. Geoinformat.*, vol. 1, no. 4, pp. 3–14, 2001.

[19] D. A. Pouliot, D. J. King, F. W. Bell, and D. G. Pitt, "Automated tree crown detection and delineation in high-resolution digital camera imagery of coniferous forest regeneration," *Remote Sens. Environ.*, vol. 82, no. 2–3, pp. 322–334, 2002.

[20] A. N. Skurikhin, S. R. Garrity, N. G. McDowell, and D. M. Cai, "Automated tree crown detection and size estimation using multi-scale analysis of high-resolution satellite imagery," *Remote Sens. Lett.*, vol. 4, no. 5, pp. 465–474, 2013.

[21] J. Inglada and E. Christophe, "The Orfeo toolbox remote sensing image processing software," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2009, vol. 4, pp. 733–736.

[22] GDAL Development Team, *GDAL—Geospatial Data Abstraction Library, Version 1.10.1*. Open Source Geospatial Foundation, 2013 [Online]. Available: http://www.gdal.org

[23] C. Lanczos, "Trigonometric interpolation of empirical and analytical functions," *J. Math. Phys.*, vol. 17, pp. 123–199, 1938.

[24] K. Turkowski, "Filters for common resampling tasks," in *Graphics Gems*. San Diego, CA: Academic Press Professional, 1990, pp. 147–165.

[25] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[26] R. Myneni, F. Hall, P. Sellers, and A. Marshak, "The interpretation of spectral vegetation indexes," *IEEE Trans. Geosci. Remote Sens.*, vol. 33, no. 2, pp. 481–486, Apr. 1995.

[27] A. A. Gitelson, Y. J. Kaufman, R. Stark, and D. Rundquist, "Novel algorithms for remote estimation of vegetation fraction," *Remote Sens. Environ.*, vol. 80, pp. 76–87, 2002.

Tanmay Prakash received the Bachelor of Science degree in electrical engineering from Duke University, Durham, NC, USA, in 2011. Since 2011, he has been pursuing the Ph.D. degree at the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA.

He is affiliated with the Robot Vision Laboratory, School of Electrical and Computer Engineering, Purdue University. His research interests include computer vision, machine learning, image processing, and robotics.

Bharath Comandur received the Bachelor of Science degree in electrical engineering from Indian Institute of Technology Madras, Chennai, India, in 2012. Since 2012, he has been pursuing the Ph.D. degree at the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA.

He is affiliated with the Robot Vision Laboratory, School of Electrical and Computer Engineering, Purdue University. His research interests include computer vision, image processing, sensor networks, and robotics.

Tommy Chang received the B.S. degree in electrical engineering and the B.S. degree in computer science from the University of Maryland, College Park, MD, USA, in 1997 and 1998, respectively, and the M.S. degree in computer science from Johns Hopkins University, Baltimore, MD, USA, in 2008. Since 2011, he has been pursuing the Ph.D. degree at the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA.

From 1997 until 2011, he was a member of the Intelligent Systems Division, National Institute of Standards and Technology, Gaithersburg, MD, USA. His research interests include computer vision, cloud computing, and robotics.

Noha Elfiky received the M.Sc. degree in computer science from the Computer Vision Center (CVC), Universitat Autonoma de Barcelona (UAB), Barcelona, Spain, in 2009, and the Ph.D. degree (*summa cum laude*) in computer vision and pattern recognition from UAB, in 2012.

Currently, she is a Research Scientist with the Department of Electrical and Computer Engineering (ECE), Purdue University, West Lafayette, IN, USA. Her research interests include computer vision, machine learning, and geolocalization.

Avinash Kak received the Ph.D degree in electrical engineering from the Indian Institute of Technology, Delhi, India, in 1970.

He is a Professor of Electrical and Computer Engineering with Purdue University, West Lafayette, IN, USA. His coauthored book *Principles of Computerized Tomographic Imaging* (Society of Industrial and Applied Mathematics) was republished as a classic in applied mathematics. His other coauthored book *Digital Picture Processing* (Academic Press, 1982) is also considered by many to be a classic in computer vision and image processing. His more recent books were written for his “Objects Trilogy” project. The first, *Programming With Objects* (Wiley, 2003), the second, *Scripting With Objects* (Wiley, 2008), and the last, *Designing With Objects* (Wiley, 2015). His research interests include algorithms, languages, and systems related to wired and wireless camera networks, robotics, and computer vision.