Shadow-Based Rooftop Segmentation in Visible Band Images

John Femiani, *Member, IEEE*, Er Li, *Member, IEEE*, Anshuman Razdan, *Member, IEEE*, and Peter Wonka, *Member, IEEE*

***Abstract*—This paper presents a method to extract rooftops from aerial images with only visible red, green, and blue bands of data. In particular, it does not require near-infrared data, lidar, or multiple viewpoints. The proposed method uses shadows in the image in order to detect buildings and to determine a set of constraints on which parts can or cannot be rooftops. We then use the grabcut algorithm to identify complete rooftop regions and a method to make corrections that simulate a user performing interactive image segmentation in order to improve the precision of our results. The precision, recall, and F-score of the proposed approach show significant improvement over two very recently published papers. On our test dataset, we observe an average F-score of 89% compared to scores of 68% and 33%.**

***Index Terms*—Buildings, rooftops detectors, shadows, urban areas.**

I. Introduction

ROOFTOPS are important features to extract from aerial images because there are many practical uses for information on rooftop locations, sizes, and shapes. Example applications include urban planning, where rooftops can be used to count the number of houses in an area to obtain an estimate of the population, the surface areas of rooftops could be useful for estimating the amount of energy needed for heating or cooling houses. Extracted rooftops can be especially important for simulation and training; realistic environments can be created by identifying features from aerial imagery and placing similar three-dimensional (3-D) models at the same locations.

One important application for rooftop extraction is to generate more realistic data for flight simulators. Aerial imagery is used in many flight simulation systems to capture the appearance of "clutter" on the ground; however, at low-altitude imagery alone does not exhibit the motion parallax effect, which is important for pilots to judge heading and altitude [1]. A system is proposed in Chladny *et al.* [2] to add vertical displacements to buildings and trees in order to render objects in a manner that is visually consistent with existing aerial images used as terrain skins in flight simulation systems, but the system would often need buildings to be identified from areal photographs with only visible red, green, and blue (RGB) imagery because of the large investment in existing texture maps used in flight simulations.

We posit that one of the most distinctive features of rooftops in photographic images is the shadows they cast; rooftops are often composed of straight edges or circular arcs, which cast shadows that are significant image features on one or more sides of a structure when the sun is not directly overhead. This work proposes a novel solution to exploit the shadows and color data in high-resolution imagery in order to identify rooftops. We first use simple methods to extract the shadow and vegetation from the aerial image and use them to generate initial foreground and background constraints on the image, then a grabcut segmentation is performed to segment the whole image into foreground and background regions, i.e., rooftops and nonrooftops as has previously been done by Ok *et al.* [3]. However, after the first segmentation, we propose a novel self-correction method to identify the mislabeled rooftops and remove those errors by running grabcut again with new constraints.

A key idea of this paper is that even simple methods to identify shadows can be used to detect buildings from color photographs. The approach is robust enough to be used in aerial orthophoto mosaics at resolutions of 1 pixel/m. The proposed method has the following benefits.

1) The method requires only reasonably high-resolution aerial image (e.g., one meter per pixel). Not required are multiple views, additional information such as near-infrared (NIR), lidar, or any elevation data.
2) The method is robust to variation in the image quality that would negatively affect many edge-based methods, and it makes few assumptions about the shape (rectangular, circular, and polygonal), or colors of buildings, with the notable exceptions that dark-green rooftops might be eliminated as trees, and the area of a rooftop is considered in a postprocessing step.
3) The method does not require a large amount of prelabeled data as would be needed by a supervised learning approach.

The problem of rooftop extraction has a long history, and detailed surveys can be found in [4] and [5]. However, the type of data available has evolved over time and now high-resolution color imagery is commonly available. While there are many papers that use only single band or color imagery to find rooftops, the majority of research on building extraction assumes that buildings are identified on source data with

Manuscript received June 23, 2014; revised September 27, 2014; accepted October 27, 2014.

J. Femiani, E. Li, and A. Razdan are with the Department of Engineering and Computing Systems, Arizona State University, Mesa, AZ 85212 USA (e-mail: john.femiani@asu.edu; erli2@asu.edu; razdan@asu.edu).

P. Wonka is with the Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 852871 USA.

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.2014.2369475

other information such as NIR, lidar, or multiple viewpoints. This paper makes the following contributions.

1) We present a novel method for extracting rooftops in imagery without exploiting additional bands such as NIR, so we can obtain accurate segmentations even when provided with only color images. The method employs a nearly[1] automatic way to identify corrections that simulate a user performing interactive segmentation.
2) We significantly improve over two very recent methods that can be applied to RGB data. In our test sets, we have aggregated precision, recall, and F-scores of 89% each, compared to an F-score of 33% for Cote and Saeedi [6] and 68% for Ok *et al*. [3].

The most significant requirement of the proposed method is a need for visible shadows with high contrast in the imagery. The proposed approach is best suited to images acquired around midday where shadows are thin, and the difference between shaded and unshaded regions is often quite significant. This requirement is often met in aerial orthophoto images acquired for flight simulation.

II. Prior Art

Many techniques detect buildings by exploiting high-resolution height information obtained by lidar or stereophotogrammetry [7]–[12]. However, the additional information changes the problem significantly. The ability of the proposed technique to identify rooftops without height data is important because archival data often do not include this information. For example, one significant application of the proposed work is to generate high-resolution displacement maps for use in flight simulators. Many existing databases have terrain and texture map data without the elevation needed for rendering high-resolution displacement maps, and the proposed approach can be used to identify rooftops as a first step to generate high-resolution displacement maps.

Shape analysis is one classical way of extracting rooftops. Based on an observation that most rooftops are rectangular or combinations of several rectangles, Cui and Reinartz [13] used the Hough transform to extract the structure of buildings and then constructed a graph from those region information. A cycle detection on the graph is utilized finally to extract the boundary of buildings. Benedek *et al.* [14] constructed a hierarchical framework to create various building appearance models from different elementary feature-based modules. The interaction between object extraction and local textural image-similarity information in their framework was exploited in a unified probabilistic model. However, those methods are more suitable for buildings with rectangle shapes.

Considering that rooftops usually have strong edges in aerial images, some methods try to extract the rooftops from these features. Sirmacek and Unsalan [15] proposed the use of the scale invariant feature transform (SIFT) and graph theoretical tools to detect buildings from urban area. One limitation of their method is that they need specific building templates for the subgraph matching. Cote and Saeedi [6] used corners and variational level set evolution (VLSE) to extract rooftops from nadir-looking aerial imagery. The corners are assessed using multiple color and color-invariant spaces, then rooftop outlines are obtained from selected corners through level-set curve evolution. Their method does not depend on the existence of shadows in aerial images; however, it is sensitive to the resolution of aerial images and their method cannot distinguish rooftops from other structures with salient boundaries. Section IV-B3 includes a more detailed comparison of the proposed method and their level set approach.

Several authors have used shadows for rooftop segmentation from aerial images; however, shadows are most-often used after an initial building detection step, for building hypothesis verification and height estimation [16]–[20]. One example approach was due to Sirmacek and Unsalan [21] who tried to verify the appearance of buildings using the shadow information; however, this work is limited to rectangular buildings. The proposed approach differs from these in that it actually uses shadows to generate and refine building hypothesis in addition to using them as a verification step. Some prior work does use shadows to generate a building hypotheses. Akcay and Aksoy [22] used a watershed segmentation and proposed shadow information and directional spatial constraints as a way to detect candidate building regions. Liow and Pavlidis [23] extract line segments adjacent to shadows in the image and employ a region growing algorithm to find other edges of the building. However, gable and hip roofs can have strong ridges [Fig. 1(b)], which can be confused with edges of the building. Our proposed approach does not depend on edge detection, and it can handle buildings that are nonrectangular [Fig. 1(a)] or, in many cases, rooftops that have internal edges such as gable ridges.

Recently, Ok *et al.* [3] used a method that employed grabcut and shadows to segment rooftops from high-resolution imagery. Similar to the proposed approach, shadows were first detected and foreground (rooftop), and background pixels were labeled adjacent to them based on light direction. This was followed by iterative graph custs (a modified version of the grabcut algorithm) on a region of interest (ROI) for each shadow. Unlike the approach we propose, they required an additional near-infrared (NIR) band of data in order to locate shadows in the imagery. They employed an ROI determined by dilation of the shadow component with flat kernels opposite to the direction of light. Foreground pixels are determined by double thresholding a fuzzy region extended from shadows opposite to light direction. Our method differs from their method in several important ways. We run grabcut on overlapping tiles from the original image and not on an ROI for each shadow. Thus, we are able to find buildings whose shadows are incomplete because of clutter or vegetation near the buildings. Unlike their approach, the proposed method can be used to identify shadows that are larger than the ROI, such as industrial buildings or warehouses in commercial areas [see Fig. 1(c)]. This is important for applications involving flight simulators because those types of buildings are common near airports where pilots can see the depth in those building during low-altitude flight. We further implement a self-correcting scheme that identifies

[1]The process requires two data-dependent parameters, the direction of light and a threshold for detecting shadows. These parameters can be set for large areas that cover many images; and can be derived from metadata such as the time and location that an image was acquired.

Fig. 1. Examples of roof shapes that the proposed method can handle: (a) objects with circular or curved roofs; (b) rooftops with internal edges; (c) very large rooftops with obtuse angles; (d) 1-m resolution image with different roof shapes; (e)–(h) show the segmentation results of (a)–(d), respectively, using the proposed algorithm.

falsely labeled pixels by analyzing the contours of buildings identified by the first pass of grabcut. We rerun grabcut until the segmented results are consistent with shadows, whereas their method does not refine the initial segmentation. Section IV-B3 includes a more detailed discussion and comparison of the result of the proposed approach and their method.

III. Segmentation Algorithm

The proposed approach for segmenting rooftops is based on an interactive approach called grabcut [24]. The grabcut algorithm is initialized by a set of constraints, called a trimap, that is usually created interactively by a user. Users are provided with a sketch-based interface that allows them to mark certain pixels as foreground or background while leaving most of the pixels unknown. The grabcut algorithm then iterates between an expectation maximization step in order to fit Gaussian mixture models (GMMs) to the foreground and background pixel colors, and a conditional random field (CRF) optimization step in order to assign labels to the unconstrained pixels using a globally optimal graphcut method [25]. Grabcut is an effective approach to segmenting images, but it can mislabel large portions of the image when the colors are under-constrained by the initial trimap. In an interactive setting, a user would look at the result of grabcut and place marks on only a few pixels where the image is over segmented, or under segmented. The user would then repeat grabcut with new constraints, leading to highly accurate results with very little user interaction.

Grabcut is appealing for rooftop segmentation because it requires few prior assumptions regarding the colors or textures of rooftops, and it can segment objects that are not globally separable by color or texture features alone. The main idea of our approach is to provide an automatic process to replace the user interaction in grabcut. The flowchart of the entire process is shown in Fig. 2. We initialize the algorithm based on shadows as was recently done by Ok *et al.* [3] using a modified four-band (*RGB* and NIR) version of grabcut; however, unlike Ok, we also add corrections to the results wherever they are inconsistent with shadows in the image until grabcut converges to a segmentation that is consistent with the shadows. The additional precision obtained by adding iterative corrections to the grabcut segmentation allows us to achieve competitive rooftop extraction results using only three-band *RGB* input data. The key steps of the proposed process are as follows.

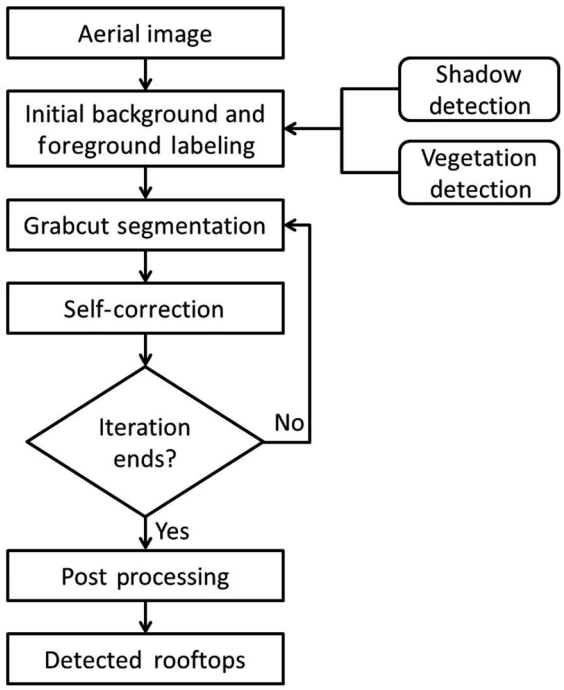


Fig. 2. Flowchart of the pipeline of the proposed approach.

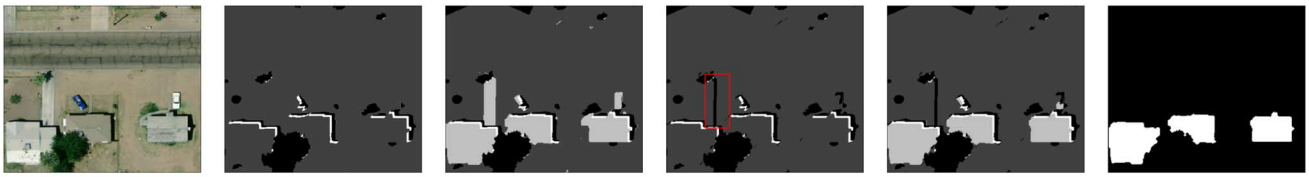


Fig. 3. Key steps of rooftop segmentation: (a) an aerial image; (b) initial labeling of foreground (white) and background (black) based on shadows, unclassified pixels are 50% gray; (c) result of running grabcut on (b) with pixels classified as foreground at 75% gray and background at 25%gray; (d) additional background constraints added for roof pixels that do not cast shadows (inside the red window); (e) corrected result after running grabcut on (d); and (f) rooftops are finally extracted from (e).

1) We generate an initial trimap [Fig. 3(b)] by analyzing the image to detect shadows and vegetation.
2) We use grabcut to determine an initial segmentation of the image [Fig. 3(c)].
3) We analyze the contours of the segmented image and generate new constraints where the contour edges do not cast shadows as expected [Fig. 3(d)].
4) Steps 2) and 3) are repeated. Fig. 3(e) shows result after repeating steps 2) and 3) once.
5) We refine the results by eliminating small foreground regions that are either too small or too thin to be rooftops.

A. Grabcut Segmentation

Grabcut is an interactive foreground/background segmentation algorithm introduced by Rother *et al*. [24]. Since, it is important for the proposed approach for rooftop extraction, this section will provide a brief explanation of the grabcut algorithm. Grabcut takes as input an image as an array of n pixels $\mathbf{z} = (z_1, z_2, \ldots, z_n)$, where each z_i is a triple in an *RGB* color space. It also takes as input an incomplete labeling (a trimap) in which each pixel is associated with a label in the set $T = \{T_B, T_F, T_U\}$. The labels T_B and T_F represent the background and foreground constraints assigned interactively by a user, and T_U denotes the pixels with unknown labels. Since the grabcut algorithm was originally designed as a matting tool, the segmentation of the images is expressed by an array of opacity (alpha channel) values $\underline{\boldsymbol{\alpha}} = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ where $\alpha_i = 0$ if z_i is a background pixel and $\alpha_i = 1$ if z_i is foreground pixel.

The grabcut algorithm models the colors in an image using two full covariance GMMs with K components each (typically $K = 5$). For reasons of efficiency, Rother *et al*. associate each pixel with single component of either the foreground (if $\alpha_i = 1$) or background ($\alpha_i = 0$) mixture. An array representing the GMM components for each pixel is defined as $\mathbf{k} = \{k_1, k_2, \ldots, k_n\}$ with $k_i \in \{1, \ldots, K\}$. The grabcut method segments an image by optimizing a Gibbs energy function of a CRF model defined on unary and pairwise cliques as

$$E(\underline{\boldsymbol{\alpha}}, \mathbf{k}, \underline{\boldsymbol{\theta}}, \mathbf{z}) = U(\underline{\boldsymbol{\alpha}}, \mathbf{k}, \underline{\boldsymbol{\theta}}, \mathbf{z}) + V(\underline{\boldsymbol{\alpha}}, \mathbf{z}) \tag{1}$$

and the unary term U is defined, using the GMM models, as

$$U(\underline{\boldsymbol{\alpha}}, \mathbf{k}, \underline{\boldsymbol{\theta}}, \mathbf{z}) = \sum_{i=1}^{n} D(\alpha_i, k_i, \underline{\boldsymbol{\theta}}, z_i) \tag{2}$$

where D is

$$D(\alpha_i, k_i, \underline{\boldsymbol{\theta}}, z_i) = -\log p(z_i | \alpha_i, k_i, \underline{\boldsymbol{\theta}}) - \log \pi(\alpha_i, k_i) \tag{3}$$

where $p(\cdot)$ is a multivariate Gaussian probability distribution, $\pi(\cdot)$ denotes the weighting coefficient of one of the GMM components, and $\underline{\theta}$ represents the parameters of a mixture model

$$\underline{\boldsymbol{\theta}} = \{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k) | \alpha \in \{0, 1\}, \;\; k \in \{1, \ldots, K\}\} \tag{4}$$

where weights π, means μ, and covariance matrices Σ are from the $2K$ Gaussian components.

The pairwise smoothness term V is defined over all pairs of neighboring pixels as

$$V(\underline{\boldsymbol{\alpha}}, \mathbf{z}) = \lambda \sum_{(i,j) \in C} [\alpha_j \neq \alpha_j] e^{-\beta \|z_i - z_j\|^2} \tag{5}$$

where C is a set of pairs of direct or diagonally adjacent pixels, and β and λ are constants controlling the degree of smoothness. The constant β is chosen to be

$$\beta = (2 \left\langle \|z_i - z_j\|^2 \right\rangle)^{-1} \tag{6}$$

where $\langle \cdot \rangle$ denotes expectation over an image sample [25], and the smoothing constant λ is set to 50 following the guidelines in [24].

The Gibbs energy is minimized by repeating the following three steps until convergence. Step 1) the energy is minimized over $\mathbf{k}$ by enumerating all possibilities. Step 2) minimizes with respect to $\underline{\theta}$ by calculating the sample means and covariance matrices of the pixels assigned to each GMM component. Step 3) is global optimization step that solves for $\underline{\alpha}$ using a minimum cut algorithm on a graph constructed as described in [25] so that the weight of a cut in the graph is proportional to the Gibbs energy. Rother *et al.* [24] repeat these three steps until the energy ceases to decrease significantly.

B. Shadows

The first and most critical step of the proposed method is to detect shadow regions and use the shadows to place both foreground (rooftop) and background constraints in the image. A number of methods to detect shadows in aerial images have been proposed; however, most are aimed at removing shadows from the image [16]–[19], [26]. Notably, Tsai [27] explored the

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

Fig. 4. Example of successful roof detection when the shadow has gaps (notice the trees obstructing the shadow of the building): (a) roof image; (b) initial foreground/background assignment; (c) foreground/background labels with additional vegetation constraint; and (d) result of grabcut on (c).

use of color invariants to detect shadows automatically in aerial images, and Chung *et al.* extend the approach using a multistage thresholding algorithm [28]. We found that the image resolution and noise in the images prevented the automatic threshold selection approach used by Tsai from correctly identifying shadows. The multithresholding approach of Chung is best suited for capturing the shape of large shadows; however, when the angle between the sun and the viewing direction of the aerial photo is small the shadows in aerial images are often very thin. In our system, we expect a threshold for relative shadow intensity as an input parameter to the system. Rayleigh scattering in the atmosphere causes a fraction of the sun's light to be scattered into shadows as ambient illumination, where the exact amount of scattered light varies depending on the position of the sun and the amount of dust or clouds in the air. We manually choose a threshold T_S for shadows between 15% and 25% of the luminous intensity of an image. We convert RGB to YUV color space, and then the set of shadow pixels is $\mathcal{S} = \{p_i | Y_i < T_S\}$, where p_i is a pixel of the original image and Y_i is the luminance channel of a color image. In Section IV-A2, the sensitivity of the algorithm to different values of T_S is evaluated.

It is important for any algorithm that is based on shadows to recognize that shadows are not always cast onto flat ground; bushes, trees, cars, or other elements within the region receiving the shadow often interrupt the shadow's contour. If any of those extend upward then they may be lit, and introduce gaps or other artifacts within the receiving region [Fig. 4(a)]. The proposed method does not require that the shadow contours be complete because gaps in the shadow may be filled in by the graph minimum-cut steps used in the grabcut algorithm. In Fig. 4, even with broken shadow, the rooftop segmentation result is accurate. However, the algorithm does not identify the rooftop area occluded by the tree as foreground.

Once shadows have been identified, we identify a set of pixels $\mathcal{F}$, shown as white pixels in Fig. 3(b), which we treat as likely elevated areas that cast the shadows. We require that the component of the light direction that lies within the image plane L is provided as a unit vector that points away from the light source. We construct the set $\mathcal{F}$ using an assumption that shadows are the result of light occluded by a raised structure opposite the direction of light and adjacent to the shadow region. Note that this assumption can be a source of error in the method that is discussed further in Section IV-B4. In order to construct $\mathcal{F}$, the pixels in $\mathcal{S}$ are shifted by a distance d_F opposite the direction of light by a morphological operator constructed as follows. Given a light direction L and a distance d_F, we define structuring element $v_{(-L,d_F)}$ as a line segment with one end at the origin of the structuring element and the other end a distance d_F opposite the direction L. Then, we can obtain the initial foreground sets $\mathcal{F}$ from the set of shadow pixels $\mathcal{S}$ as

$$\mathcal{F} = (\mathcal{S} \oplus v_{(-L,d_F)}) - \mathcal{S} \tag{7}$$

where $\oplus$ denotes the morphological dilation operator. The thickness d_F of the foreground regions needs to be far enough that some pixels in the image will be constrained as foreground. We have set d_F as 2 m in our implementation for image resolutions ranging from 0.15 to 1 m/pixel (so d_F is between 6 and 2 pixels), and Section IV-A4 explores the effect of choosing different values for d_F.

Fences or block walls are a particularly troublesome issue for methods, which use shadows to indicate the presence of rooftops because they cast shadows that are similarly shaped to those cast by buildings. Ok *et al.* [3] use a threshold on the thickness of shadows to distinguish fences from rooftop shadows. This works well on some images, but we find that it is difficult to choose an appropriate thickness threshold for several reasons. First, the shadows of buildings can become very thin when the light direction becomes nearly parallel to one of the sides of the rooftop, and the spatial resolution of the image is not sufficient to distinguish between shadows of buildings and shadows of fences. Second, fences are often adjacent to bushes or asphalt, which can darken their shadows and make them appear thicker. Finally, shadows of buildings are likely to be partially occluded by the buildings themselves due to a combination of overhang in roofs and an effect called "lean" caused by perspective distortion when the roof is not directly under the camera as an aerial image is acquired. Together these effects prevent a single threshold on shadow thickness from capturing the shadows of buildings without also including the shadows of many fences.

When the shadow of a fence is used to generate foreground constraints, the result will be a number of false foreground constraints in $\mathcal{F}$. However, our self-correction approach often removes many of the false positive labels. As a result, fences tend to generate very thin regions of false positives which we allow at this point in the algorithm as they will be removed in step 5).

C. Additional Constraints

The grabcut algorithm uses a trimap in order to both constrain the resulting labels and also to learn the distribution of colors in foreground and background regions. The trimaps shown in Figs. 3, 4(b) and (c) are examples of trimaps, which we use to incorporate prior knowledge into the proposed

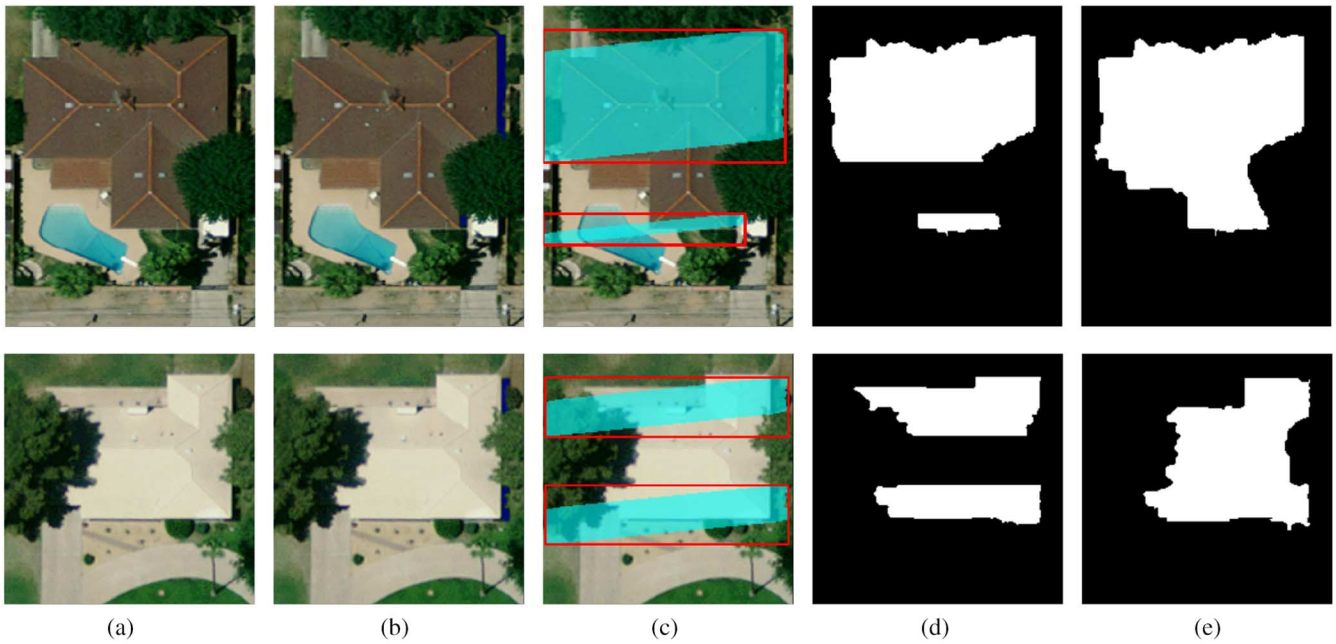


Fig. 5. Comparison of grabcut within ROI and the whole image: (a) roof image; (b) shadows (dark blue parts); (c) ROI (cyan part) with bounding box (red rectangles); (d) result of grabcut within ROI; and (e) result of grabcut applied to the whole image.

method. The grabcut algorithm will always label the white pixels as rooftop, and it will always label the black pixels as background.

We use trimaps in order to add prior knowledge about rooftops by adding pixels that are very unlikely to be rooftop to a background set $\mathcal{B}$ before running grabcut. The trimap is constructed with the set $\mathcal{F} - \mathcal{B}$ constrained to foreground (white), and the set $\mathcal{B}$ treated as background constraints (black). We consider our initial shadow-based foreground estimate $\mathcal{F}$ to have less confidence than the features (such as shadows themselves) that indicate that a pixel is not elevated. Forcing shadows to be considered background pixels has the potential to introduce errors whenever shadows are cast onto rooftops by adjacent structures, such as multilevel or complex rooftop surfaces or rooftops with adjacent trees that are taller than the building; however, we allow these errors in our method.

The set $\mathcal{B}$ includes more than just shadows; any other pixel that is unlikely to be rooftop can also be included in the set. In particular, we consider vegetation as unlikely to be part of a rooftop. A vegetation mask covers the region occupied by trees, grass, or shrubs. One source of such a mask is a thematic map constructed using additional data such as NIR or normalized difference vegetation index (NDVI) data. If a vegetation mask is not present, we use a simple color-based thresholding technique to identify pixels that are unlikely to be rooftops. Most green-colored regions belong to vegetation, whereas green-colored roofs exist but are very rare in our test sites, so we adopt the method in [29] by Otsu thresholding [30] the color index

$$C_v = \frac{4}{\pi} \cdot \arctan\left(\frac{G-B}{G+B}\right). \tag{8}$$

Pixels with a color index larger than the threshold identified by Otsu's method are marked as vegetation. Vegetation often has soft or irregularly shaped contours, and trees or shrubbery can be partially transparent at their edges where the leaves are sparse. In addition, vegetation is often dark and can be misclassified as shadows and produce false foreground constraints in our method. In order to compensate for errors near the edges of vegetation, we dilate the vegetation mask using a circular structuring element with a radius of approximately 1 m to obtain the set $\mathcal{V}$ of pixels to be treated as vegetation, and hence not rooftop. If $\mathcal{V}$ is the only constraint, then the set $\mathcal{B}$ is set to $\mathcal{B} = \mathcal{S} \cup \mathcal{V}$, where $\mathcal{S}$ is the set of pixels in shadow. Fig. 4(c) shows how the addition of vegetation constraints is incorporated into the trimap construction so that grabcut does not mislabel vegetation as rooftop. The approach discussed here for adding constraints can also be used to guide rooftop extraction using other sources of information, e.g., Femiani and Li [31] discuss an approach to identify constraints from GIS vector data such as street maps, NIR, and lidar data if it is available.

D. Segmentation

1) Grabcut Segmentation on Overlapping Tiles: After labeling the foreground and background, the *RGB* components of the image are converted into the LUV color space and grabcut is used to label the remaining pixels in the image. In our implementation, grabcut is initialized with the set $\mathcal{F} - \mathcal{B}$ constrained to foreground (roof) and $\mathcal{B}$ constrained to the background. All other pixels are used to form an initial estimate of the distribution of background colors. The examples shown in this paper use an implementation of grabcut [32], which uses a mixture of five full-covariance normal distributions in order to model the foreground and background colors. Once a trimap is constructed, the application of grabcut to the image is straightforward; the key challenge is how to run grabcut on

large images, which may not fit into a system's available RAM, and to process the images with acceptable running times.

Aerial images in our datasets are often represented as $8k \times 8k$ or $16k \times 16k$ arrays of pixels. Running the grabcut algorithm on an image that large is not practical because it is $O(n^2)$ in the worst case, where n is the number of pixels. Even though performance is often better in practice, it is polynomially larger that $O(n)$. Ok *et al.* [3] address the run time by first detecting shadows and then limiting grabcut to small regions near shadows. However, since the size of buildings in the image may vary dramatically, it is hard to include all parts of each building just using one fixed size parameter. For example, a large parameter would be needed to cover the building in Fig. 1(c). Another challenge to using shadows to determine ROIs is that the shadow contours may not be complete, and one building may be broken into several parts. Fig. 5 illustrates the limitation of running grabcut within ROI. Due to the trees around the buildings, the shadow is incomplete and thus the grabcut only extracts part of the building.

In order to process large datasets, we divide the input into small tiles and then operate grabcut in each tile. However, a potential problem with running grabcut tile by tile is that one building can span two neighboring tiles and its shadow may be cast into only one of the tiles. For such a building, only the part within the tile where shadows are located would be detected since the tile with no shadows would receive no initial foreground constraints for the building. To overcome this problem, we use fixed-size overlapping tiles in the image, so that we can utilize prior information from neighboring tiles to avoid incomplete extraction of buildings that span multiple tiles.

Since buildings from one tile may cast shadows that reach another tile in the light direction, we always process tiles starting from the corner of the input farthest from to the light source. Tiles are processed in descending order based on the inner product of L and the tiles minimum coordinate. Each tile overlaps by T_p pixels with the previous tiles (up to three tiles). When processing a tile the trimap is initialized with the labels of any overlapping tile that has already been processed, and the previously calculated labels are treated as foreground and background constraints in the new tile. Fig. 6 illustrates the process. By choosing different tile sizes, it is possible to trade global accuracy for improving the running time over an entire image; which is linear on the number of tiles. In practice, we choose tile sizes of 512×512 pixels with an overlap of $T_p = 20$ pixels. The effect of other tile sizes on running time and accuracy is evaluated in Section IV-B1 and Table I.

2) Self-Correction: The grabcut algorithm will favor segmentations with fewer boundary edges, which can result in the merging of adjacent rooftop regions. In addition, the colors in the image are not always sufficiently separable to achieve an accurate segmentation of rooftops from other features. Fig. 3(c) shows such a case in which the foreground bled into the pavement next to the rooftop. This may have happened because colors in the pavement are more common in the adjacent rooftop than they are in rest of the background. In order to address this issue, Ok *et al.* [3] introduced a maximum building size and limited grabcut to run within that region.

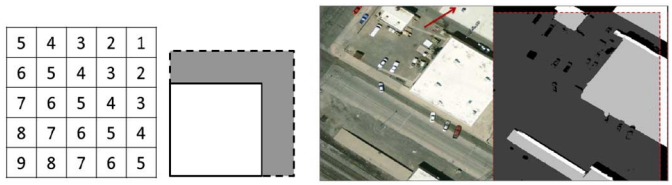


Fig. 6. Tile decomposition for distributing processing of large images. (a) The processing order of tiles, assuming the light direction is northeast, the tile at the right top corner will be processed first, then the other tiles will be handled in ascending order of the numbers. The tiles with same number can be processed in parallel to accelerate the whole process, each tile is initialized with constraints from overlapping tiles that have already been processed. In this case, the tiles above and to the right have been used to set constraints for the current tile. (b) is an example tile with the top and right portion constrained, the right image shows the constraints as black and white pixels and the labels assigned by grabcut are indicated by light and dark gray pixels. Notice how the building indicated by red arrow is recovered using overlapping tiles.

We do not assume a maximum region size. Instead, we run grabcut and then process the set of pixels, which are likely to be false positives based on the structure of shadows in the image. These pixels are then constrained to the background, and grabcut is repeated until a plausible set of labels is identified. We estimate that a pixel is a false positive if the pixel is within a distance, d_2 of a contour, and that contour is within a distance d_1 of the nearest shadow with distances measured in the direction of light. The parameter d_2 controls how aggressive, we are in marking corrections, and d_1 is chosen to be robust to errors in shadow detection. One caveat is that buildings in an aerial photograph exhibit an effect called "lean," which can shift the contour of the rooftop relative to its shadow by a small amount and result in some portion of the rooftop appearing to cast no shadow. The effect is most common near the corners of buildings. In order to compensate for lean, we use a dilated shadow mask $\mathcal{S}'$ that is the result of morphological dilation on $\mathcal{S}$ by a disk with radius r, where r is treated as an input to the system. In our examples, we set $r = 3$ pixels.

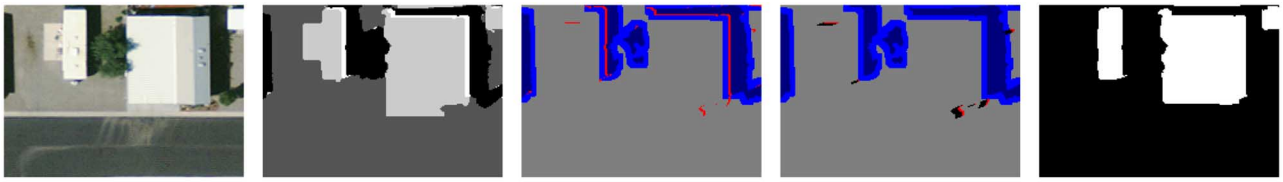
Our method to determine the corrections can be described using morphological operations. Let $\mathcal{F}^{(i)}$ denote pixels labeled as foreground *after* running the grabcut algorithm i times, so $i = 1$ after the first attempt to segment the image with grabcut. The set $\mathcal{B}^{(i)}$ is the pixels constrained to background in the trimap before the ith iteration, so $\mathcal{B}^{(0)} = \mathcal{B}$. Then, we can find the set of corrections $\mathcal{C}^{(i)}$ that are pixels, which are not consistent with shadows after i attempts as follows:

$$\mathcal{C}^{(i)} = (\mathcal{F}^{(i)} \oplus v_{(L,d_1)} - \mathcal{F}^{(i)} - \mathcal{S}') \oplus v_{(-L,d_2)} \cap \mathcal{F}^{(i)}. \quad (9)$$

Equation (9) should be understood as first identifying a set of pixels that should be shadows given $\mathcal{F}^{(i)}$ using a morphological dilation in the direction of light, and then subtracting out the existing shadows. The pixels which remain are pixels, which should be shadows if $\mathcal{F}^{(i)}$ were correct, and yet they do not appear to be shadows in the image. Another dilation opposite the direction of light determines the pixels in $\mathcal{F}^{(i)}$ which are not casting shadows, and therefore are likely false positives. During

TABLE I
TIME COST STATISTICS ON AN 8540 × 8540 AERIAL IMAGE WITH 955 BUILDINGS

Tile size	Number of tiles	Time per tile (s)	Total w/o parallelization (s)	Total with parallelization (s)	P(%), R(%), F_1(%)
128 × 128	4489	1.21	5445.42	804.29	82, 53, 65
256 × 256	1156	4.05	4683.60	703.02	81, 65, 72
512 × 512	289	14.00	4046.02	719.61	81, 74, 77
1024 × 1024	81	43.45	3519.74	840.75	81, 75, 77



(a) (b) (c) (d) (e)

Fig. 7. Illustration of self-correction; (a) a roof image; (b) the initial result of grabcut on (a); (c) the dark blue area represents $\mathcal{S}$, and the dark and light blue area together represent $\mathcal{S}'$, and the red area represents $\mathcal{F}^{(i)} \oplus v_{(L,d_1)} - \mathcal{F}^{(i)}$; (d) the red area represents $\mathcal{F}^{(i)} \oplus v_(L, d_1) - \mathcal{F}^{(i)} - \mathcal{S}'$ and the black area is the final $\mathcal{C}^{(i)}$; (e) the result of running grabcut again using the new constraints.

(a) (b) (c) (d) (e)

Fig. 8. Example of self-correction: (a) a roof image; (b) the initial result of grabcut on (a); (c) additional background constraints added by the self-correction algorithm (black pixels in the red rectangles); (d) additional constraints made by a second iteration of the self-correction algorithm on the result of grabcut on (c); and (e) the final result, with small regions removed.

our experiments, we find $d_1 = 2$ and $d_2 = 5$ will give satisfactory results. Once $\mathcal{C}^{(i)}$ is determined a new set of background constraints $\mathcal{B}^{(i)}$ can be calculated as

$$\mathcal{B}^{(i)} = \mathcal{C}^{(i)} + \mathcal{B}^{(i-1)}. \tag{10}$$

Note that when trimaps are constructed for application $i + 1$ of grabcut, the foreground constraints are set to $\mathcal{F} - \mathcal{B}^{(i)}$ and the background constraints are set to $\mathcal{B}^{(i)}$. New foreground constraints are never added in the process. If $\mathcal{C}^{(i)}$ is empty then no further iterations of self-correction are performed. Fig. 7 gives a detailed illustration of the whole pipeline.

The process of self-correction is illustrated in Fig. 3(d) [compare with Fig. 3(b)]. Another example is shown in Fig. 8, where self-correction runs through two iterations to remove all of the pixels mislabeled as foreground.

3) Pruning Misclassified Contours by Size Constraints: The final segmentation may contain very small regions which may be cars and fences misclassified as rooftops. In order to prune them, we find the contour of each segmented region and if the length of the contour is less than 20 pixels, we mark the entire connected component as background. Fig. 9(c) shows the segmentation after pruning small contours from Fig. 9(b).

IV. EVALUATION

In order to evaluate our method, we manually labeled a number of regions and treated these labels as ground truth.

We use the common measures of precision (P), recall (R), and the F-score (F_1) to measure error [33], where

$$P = \frac{TP}{TP + FP} \tag{11}$$

$$R = \frac{TP}{TP + FN} \tag{12}$$

$$F_1 = \frac{2PR}{P + R}. \tag{13}$$

Here, TP stands for true positives and refers to the number of pixels assigned as rooftop in both ground truth and segmentation result. FP stands for false positives and refers to the number of pixels designated as rooftop in by the proposed approach but not in ground truth. FN stands for false negatives and refers to the number of pixels designated as rooftop in ground truth but not in the results. The F-score (F_1) captures both precision and recall as a single metric that gives each equal importance.

A. Parameters

The proposed process requires the following parameters to be specified by a user.

1) A threshold on the image luminance channel, used to identify shadows. Fortunately, shadows in the most aerial images seem to be separable based on intensity and this threshold is not difficult to identify if the user can preview

Fig. 9. Effect of pruning small contours: (a) an aerial image; (b) the result of rooftop segmentation before pruning; and (c) after pruning.

the thresholded image. On clear days a threshold between 15% and 20% of the image max intensity is a good choice.

2) The approximate direction of the light within the image plane (L). This can be estimated from an image by counting the number of pixels horizontally and vertically from one corner of a rooftop (or from a corner of the footprint if lean is an issue), to the corresponding corner of a buildings shadow. If streetlights are visible, their shadows are also good indicators for the direction of light. If information is available the date and time that a georeferenced image is acquired, it is also possible to calculate the direction of sunlight.
3) The shifting distance d_F which controls the range of initial foreground labels. This can often be set to 2 m.
4) The parameters in grabcut, including the maximum number graph cut iterations, the number K of components in the GMM, and the smoothing term λ.

1) Grabcut Parameters: During our experiments, we found that the most influential parameter of grabcut is the maximum number of iterations of grabcut. The majority of processing time for the proposed method is spent on graph cut iterations performed by grabcut. Fig. 10 shows the results after using 1, 5, 10, and 20 iterations. After a certain number, the results do not show a noticeable difference in the foreground [Fig. 10(d)]. Rather than iterating until convergence, we choose a number beyond which F_1 ceases to improve on training data. After 10 iterations, we do not perceive the difference in labels, and the F-score on the resulting pixel labels is not improved. If a higher number is chosen, such as 20 iterations in Fig. 10(e), then time is often wasted running iterations of graphcut that do not necessarily improve the overall accuracy. For the other parameters, the default settings ($K = 5, \lambda = 50$) suggested in [24] work well in our dataset.

2) Shadow Detection: The accuracy of shadow detection plays a very important role because initial foreground and background constraints are assigned on the basis of shadows. Several methods [27], [28] have been proposed to extract shadows from aerial images automatically; however, we find that these methods do not outperform the fixed threshold used by our method in our test dataset due to the presence of many dark roads, which are difficult to rule out using existing automatic methods as shown in Fig. 11. The aerial images in our dataset are acquired mostly around midday where shadows are quite clear and thin, and the intensity of shadows has small variance across the input tiles. Based on this observation, a fixed threshold on luminance works well. Fig. 12 shows the impact of using different threshold values in order to determine shadows. Fig. 12(b) shows initial foreground and background segmentation based on detected shadows and Fig. 12(c) shows the result of running grabcut. The threshold chosen is 25% of the max intensity in the image. Fig. 12(d) and (e) shows the same for a threshold of 0.15. Reducing the shadow threshold causes some building's shadows to be missed, which hurts recall. The effect of various values of T_S is plotted in Fig. 15(a).

3) Direction of Light: The direction of light is important to assign initial foreground and background pixel constraints once shadows are identified. If the light direction is not accurate, then it is possible that foreground and background are incorrectly assigned. When the light direction bisects, two edge normals of a rooftop's contour as in Fig. 13 then the method is very robust. In general, the sensitivity to L depends on the relationship between the rooftop orientations and the light direction; when the rooftop edges are parallel to the light direction then small perturbations in L can cause the shadows to switch from one side of the building to another. The degree of sensitivity to light direction is shown more quantitatively in Fig. 15(b), which plots R, P, and F_1 for various choices of L.

4) Foreground Constraints: Grabcut results improve if there are more accurate foreground pixels. Fig. 14 shows the effect of d_F for $d_F = 2$ or $d_F = 6$ for images with 0.32 m/pixel resolution. If d_F is too small then grabcut will not be able to infer the foreground distribution of colors properly because of aliasing or blurring artifacts in the imagery. If it is too large, then we will label pixels past the edges of the rooftop as foreground. In Fig. 15(c), we plot various choices of d_F, from which we conclude that 2 m is a good value to generate enough foreground for grabcut to detect buildings.

B. Experiments

We tested the proposed method on a set of aerial images with different sizes varying from $1K \times 1K$ to $8K \times 8K$. The test images are from urban or suburban regions of AZ, USA, and they include flat and gabled rooftops with complex shapes.

1) Performance: We implemented our algorithm in python and used the OpenCV implementation of grabcut. Our implementation was not optimized and took 14 s on average for each 512×512 pixels square tile using a PC (Intel Core i7 CPU 3.07 GHz with 6 GB RAM). The individual run times varied from 5 to 30 s based on the complexity of the tiles. We parallelized our algorithm to accelerate the process of large size aerial images. The running time on a 8540×8540 aerial images with 955 buildings is given in Table I. We tested four different tile sizes with an overlap of $T_p = 20$ pixels. There is no significant difference of the time cost when tile size varies from 128 to 1024, but our system ran out of RAM when we attempted to process an entire image as one tile. We also give the performance of different tile sizes. A small tile size resulted in a reduced recall score as shown in Table I. The recall is better at a tiles size of 512 and when the tile size increased from 512 to 1024 we observed no significant change. In all of our experiments, we chose 512 as the default tile size.

2) Qualitative and Quantitative Evaluations: A few tiles from one of the dataset are shown in Fig. 16. The images #1–7 have 0.32 m-per-pixel resolution and consist of a variety of

Fig. 10. Effect of different numbers of graphcut iterations: (a) an aerial image; (b)–(e) show the result of 1, 5, 10, and 20 iterations of graphcut, respectively, the F-scores are 69.7%, 85.0%, 87.1%, and 86.8%. Here, green, blue, and red colors represent TP, FP, and FN pixels, respectively. Note that in order to show the effects of different grabcut iterations, self-correction is not utilized.

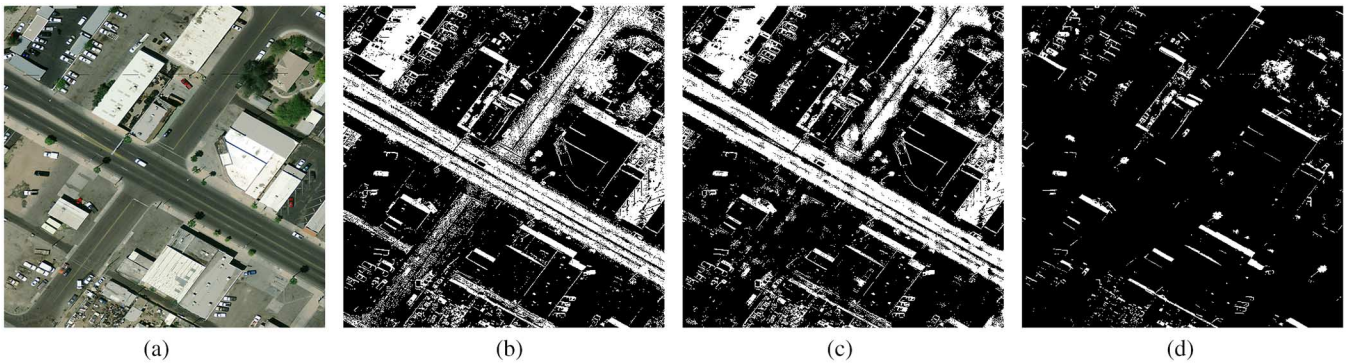


Fig. 11. Comparison of different shadow detection method: (a) is original aerial image; (b) is shadows extracted using method in [27]; (c) is shadows extracted using method in [28]; and (d) is shadows using $Y_i < 0.15$. White area denotes shadows.

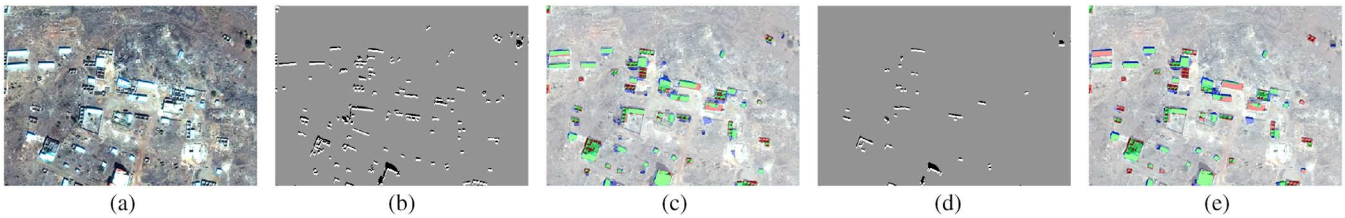


Fig. 12. Effect of accuracy of shadow detection: (a) an aerial image; (b) and (d) show initial foreground and background labeling using luminance threshold of 0.25 and 0.15, respectively; (c) and (e) show the result of grabcut on (b) and (d), respectively. True positives are shown in green, false negatives are shown in red, and false positives in blue.

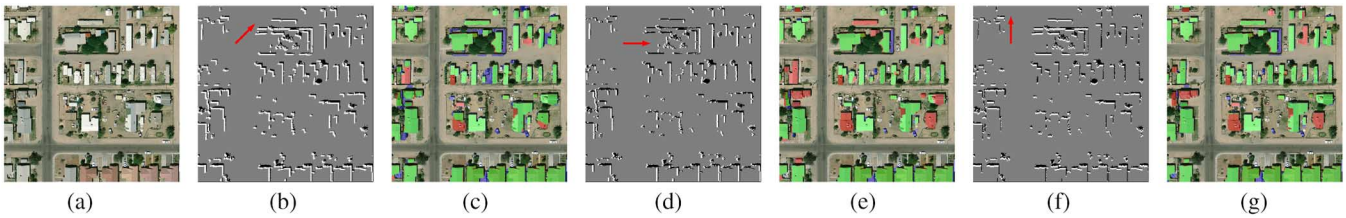


Fig. 13. Effect of accuracy of light direction, the red arrow indicates the light direction: (a) an aerial image; (b) using correct light direction for initial foreground and background labeling; (c) result of grabcut on (b); (d) initial labeling using light rotated by 45° clockwise; (e) result of grabcut on (d); (f) initial labeling using light rotated by 45° anticlockwise; (g) result of grabcut on (f). True positives are tinted green, false negatives are tinted red, and false positives are tinted blue.

rooftop shapes. Image #3 also has a building with curved roof. Image #8 has a resolution of 1 m/pixel. The figures show the TP, FP, and FN pixels in different colors. Table II shows precision, recall, and F-scores for all the test images shown based on manually entered ground truth. The precision and recall average 88% and 91%, respectively. We compare against the method of Cote and Saeedi [6] and a variant of the method in [3], but without using NIR data (since, we aim to process *RGB* imagery). We call this a *modified* Ok method. A comparison to method in [3] is also presented in section IV-B3. The proposed approach improves over the modified Ok method, which shows a precision of 87%, and recall 56% and over the method of Cote and Saeedi [6] which shows a precision of 27%, and recall 42% on these images.

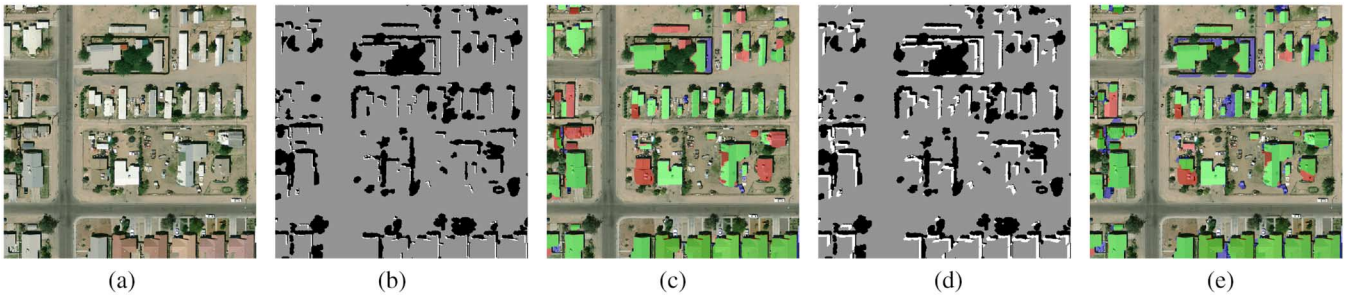


Fig. 14. Effect of choosing different d_F: (a) an aerial image; (b) and (d) show initial labeling with $d_F = 2'$ and $d_F = 6'$, respectively; (c) and (e) show the result of grabcut on (b) and (d), respectively. True positives are tinted green, false negatives are tinted red, and false positives are tinted blue.

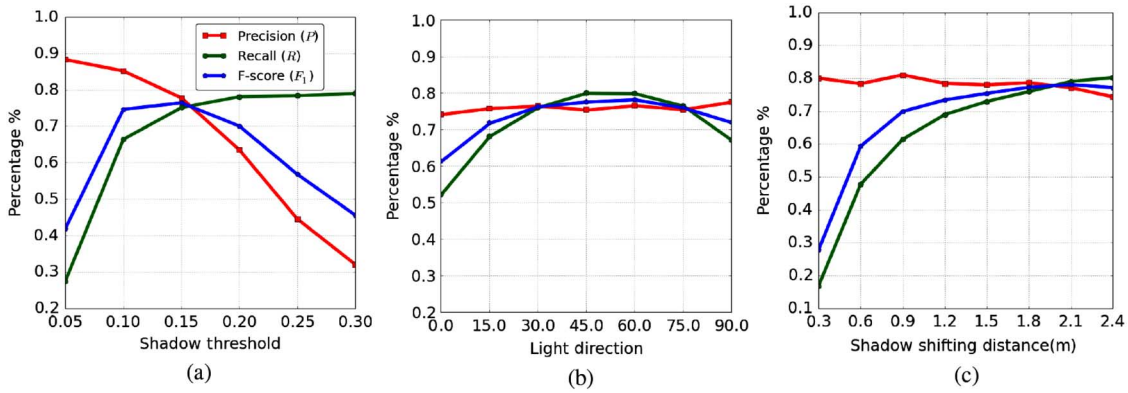


Fig. 15. Performance curves of each parameter: (a) effects of shadow threshold parameter; (b) effects of light direction starting from East (0°) CCW to North (90°); and (c) effects of shadow shifting distance. In each plot, the nonvarying coefficients are kept at their optimal settings: shadow threshold = 0.15, light direction = 45, shadow shifting distance = 2 m.

Fig. 16. (Odd column) test images #1–9; (even column) segmentation results for test images #1–9. True positives are tinted green, false negatives are tinted red, and false positives are tinted blue.

3) Comparisons: We first compare our method with the VLSE method proposed in [6]. We used source code provided by the author and all of the recommended parameters in order to produce the result of VLSE. We applied our method on the same data originally used in [6] and show the comparisons in the first row of Fig. 17. The proposed approach gives a highly competitive result with precision 82.2%, recall 95.0%, and F-score 88.1% compared with precision 83.7%, recall 88.7%, and

TABLE II
EVALUATION OF ROOFTOP SEGMENTATION

Test image	Modified Ok method (%)			Cote and Saeedi's method (%)			Proposed method (%)		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
#1	91	63	74	30	43	35	91	95	93
#2	76	46	57	8	98	15	88	98	93
#3	82	48	61	74	36	48	85	87	86
#4	90	53	67	13	25	18	96	86	91
#5	91	57	71	99	40	58	90	90	90
#6	91	60	72	96	50	66	87	94	90
#7	95	86	90	49	66	56	97	93	95
#8	73	21	33	1	5	1	83	74	78
#9	60	31	41	28	12	17	85	84	84
Avg by pixel	87	56	68	27	42	33	88	91	89

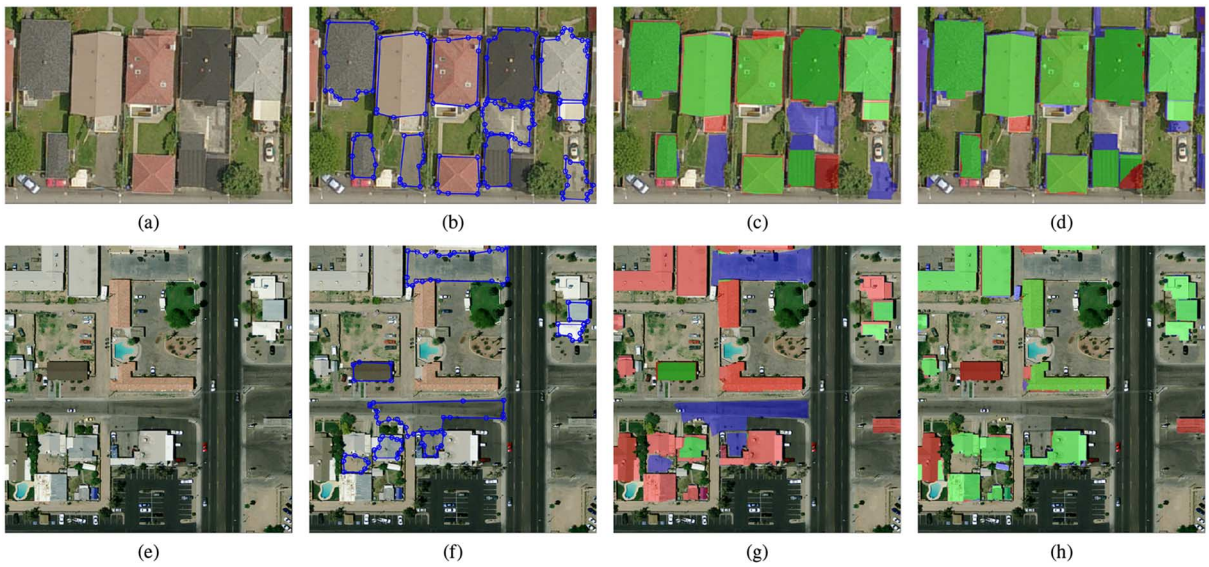


Fig. 17. Comparisons with method in [6]: (a) an aerial image; (b) the contours extracted by method in [6]; (c) the result of [6] compared with ground truth; (d) our result compared with ground truth; (e)–(h) show the same comparison on 1 m/pixel image. True positives are tinted green, false negatives are tinted red, and false positives are tinted blue.

F-score 86.1% by VLSE. In particular, the proposed approach produced fewer false positives than VLSE. We then applied both methods on data at a lower resolution of 0.32 m/pixel as shown in the second row of Fig. 17. VLSE performs poorly on lower resolution imagery with precision 22.2%, recall 12.8%, and F-score 16.2%; however, our method still gives high-quality extraction results with precision 95.7%, recall 80.4%, and F-score 87.3%. VLSE has the advantage of not relying on the shadows, but when the shadow information is available in the aerial image our method performs better. We also compare our method with the automated rooftop detection method proposed by Ok in [3] as shown in Fig. 18. The 512×512 image has four bands: red, green, blue, and NIR, which allows the original (four-band) version of Ok's method to be compared. The resolution is 0.6 m/pixel. Fig. 18(b) shows the result of Ok's method with precision 57.9%, recall 12.5%, and an F-score of 20.5%. We first apply our method on the image using the shadows provided by Ok *et al*. As shown in Fig. 18(c), our method achieves better results with precision 68.5%, recall 23.2%, and F-score 34.6%. We then apply the whole pipeline of our algorithm on the image and obtain a much better result with precision 56.9%, recall 46.1%, and F-score 51.0% as shown in Fig. 18(d). Their method is more sensitive to the shadows so they only use the most reliable shadows to improve the precision at the cost of recall. Our method keeps high score on both precision and recall with the help of the proposed self-correction steps.

4) Limitations and Failure Cases: Our approach is designed based on the assumption that the aerial image is acquired around midday, if this assumption is wrong then our approach may fail to capture the correct rooftop. Under oblique lighting, a gabled rooftop may exhibit significantly different intensities on the sloped portions of the roof, so the grabcut step may only capture one side of the rooftop as is shown in Fig. 19(a) and (b). In an extreme case, an entire face of the rooftop may be in shadow. Even if the image is acquired around noon, if the rooftop contains several components with different colors then our method will fail to obtain the entire rooftops as shown in Fig. 19(d), where the rooftops in the red rectangle area are missing.

Another limitation is caused by inaccurate shadow extracted by our method. Although we have shown in Fig. 11 that

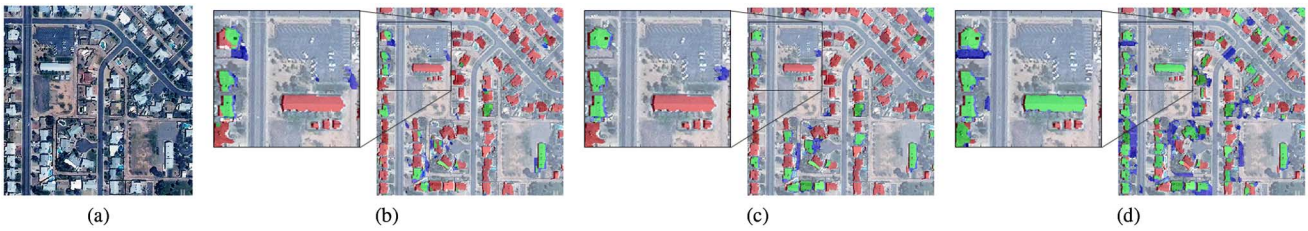


Fig. 18. Comparisons with method in [3]: (a) an aerial image; (b) the result of [3]; (c) the result of our approach using shadows detection method in [3]; and (d) results using the whole pipeline of our approach. True positives are tinted green, false negatives are tinted red, and false positives are tinted blue.

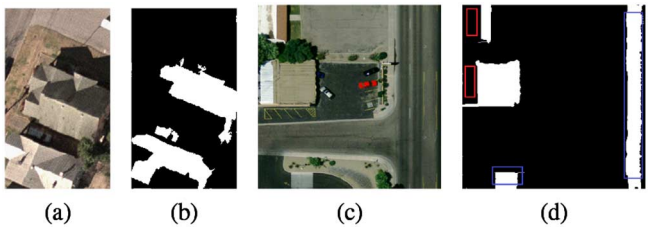


Fig. 19. Failure cases. If two sides of a gable rooftop have different illumination intensities (a) or the rooftops are composed of segments with different colors (c), our method can only capture part of the buildings [(b), red rectangle area in (d)]; very dark roads also produce false positives [blue rectangle areas in (d)]. White area denotes buildings extracted by our method.

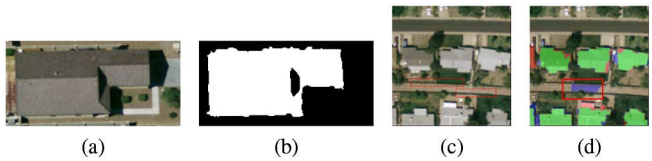


Fig. 20. Errors caused by shadows that do not fit our simple model: (a) a rooftop that casts a shadow onto itself; (b) the result of our method showing that the shadow becomes a false negative region; (c) fences that cast shadows; and (d) false positives for a region with one edge parallel to the direction of light, our self-correction method fails to completely remove this region. True positives are tinted green, false negatives are tinted red, and false positives are tinted blue.

our fixed threshold can rule out most of the dark roads, the classes are not globally separable by thresholding luminance and false positives occur as shown in the blue rectangle area in Fig. 19(d).

Our approach to labeling pixels by shadows makes the simplifying assumption that a scene has two levels of depth—elevated part (rooftops) and nonelevated parts (ground). In fact, rooftops vary in height and one rooftop may cast shadows onto another. Many buildings have multiple levels and one portion of the roof may cast a shadow onto another portion of the roof. Rooftops with multiple gables will contain wedge-shaped shadows as indicated in Fig. 20. In these cases, the shadow constraints discussed in this section will introduce errors, but these errors are rare and confined to small regions within a rooftop. Our approach also suffers from the shadows cast by walls, fences, or other man-made objects, as shown in Fig. 20(c) and (d). Self-correction can reduce the size of the false-positive regions caused by the shadows of walls, but if the contours of the false region are parallel to the direction of light then self-correction will fail. However, walls in our test imagery often occur as complete loops that enclose an area, and cast an inner shadow that forces the proposed method to label the enclosed area as ground. Our approach uses color to identify pixels as vegetation and constrain them to the background (non-rooftop), which may fail if there are green rooftops, which will be mislabeled as background.

V. Conclusion and Future Work

We have presented a novel approach for segmenting rooftops only using aerial images. The approach does not require any elevation or other additional data. It can incorporate constraints or corrections because it is built on top of the interactive grabcut method. The algorithm is sensitive to certain parameters such as shadow intensity threshold which need to be set accurately for best detection. But these can be easily chosen by looking at a small portion of the image. However, there are certain areas where we see an opportunity for further improvement.

For a very large dataset, it is possible that light directions are not consistent throughout the region due to the process of stitching together images to form an orthophoto mosaic from images acquired at different times of the day. The direction of light as well as the lean of buildings will vary slightly and it is possible that this could introduce errors in the method. Future work could estimate which side of a shadow is elevated without requiring the light direction parameter, perhaps by analyzing the shape of a shadow region or the relationship between shadow colors and the adjacent material colors.

The proposed method uses a very simple shadow detection method employing a global threshold to extract shadows. Adaptive threshold methods, or methods that filter the shadow regions in order to distinguish shadows cast from buildings (e.g., by edges quality or shadow thickness) from other features could be used to improve the proposed method as future work.

A key contribution of this paper was a framework for identifying and correcting errors by analyzing the shadows of regions after using grabcut. We limited corrections to removing false positives; future work could also attempt to use additional scene knowledge to generate corrections in the same framework. For example, the curvatures of the region contours or the variation of colors within detected regions may carry additional information that could be used to make corrections.

ACKNOWLEDGMENT

The authors would like to thank A. Ok and M. Cote for graciously sharing code and results of their methods for comparison with the proposed method. In addition the authors would thank the Decision Center for a Desert City for providing the imagery used in Fig. 18.

REFERENCES

[1] H. M. McIntyre and M. E. C. Roberts, "Simulated visual scenes-which are the critical cues?" in *Thomson Training and Simulation Ltd, Flight Simulation: Where are the Challenges? p(SEE N 97-10546 01-09)*, pp. 36–42, 1996.

[2] B. Chladny, J. Femiani, and B. Graniela, "Realistic geo-specific feature densities in real-time synthetic environments," in *Proc. Interserv./Ind. Train. Simul. Edu. Conf. (I/ITSEC)*, 2013, vol. 2013, no. 1.

[3] A. Ok, C. Senaras, and B. Yuksel, "Automated detection of arbitrarily shaped buildings in complex environments from monocular VHR optical satellite imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 3, pp. 1701–1717, Mar. 2013.

[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] E. Baltsavias, "Object extraction and revision by image analysis using existing geodata and knowledge: Current status and steps towards operational systems," *ISPRS J. Photogramm. Remote Sens.*, vol. 58, no. 3–4, pp. 129–151, 2004.

[6] M. Cote and P. Saeedi, "Automatic rooftop extraction in nadir aerial imagery of suburban regions using corners and variational level set evolution," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 313–328, Jan. 2013.

[7] C. Vestri and F. Devernay, "Using robust methods for automatic extraction of buildings," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR'01)*, 2001, vol. 1, pp. I–133.

[8] F. Lafarge, X. Descombes, J. Zerubia, and M. Pierrot-Deseilligny, "Automatic building extraction from DEMS using an object approach and application to the 3d-city modeling," *ISPRS J. Photogramm. Remote Sens.*, vol. 63, no. 3, pp. 365–381, 2008.

[9] Q.-Y. Zhou and U. Neumann, "2.5d dual contouring: A robust approach to creating building models from aerial Lidar point clouds," in *Proc. Comput. Vis. (ECCV'10)*, 2010, pp. 115–128.

[10] C. Poullis and S. You, "Automatic reconstruction of cities from remote sensor data," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR'09)*, 2009, pp. 2775–2782.

[11] V. Verma, R. Kumar, and S. Hsu, "3d building detection and modeling from aerial Lidar data," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2006, vol. 2, pp. 2213–2220.

[12] B. C. Matei, H. S. Sawhney, S. Samarasekera, J. Kim, and R. Kumar, "Building segmentation for densely built urban regions using aerial lidar data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR'08)*, 2008, pp. 1–8.

[13] S. Cui, Q. Yan, and P. Reinartz, "Graph search and its application in building extraction from high resolution remote sensing imagery," in *Search Algorithms and Applications*, N. Mansour, Ed. Shanghai, China: InTech, 2011.

[14] C. Benedek, X. Descombes, and J. Zerubia, "Building development monitoring in multitemporal remotely sensed image pairs with stochastic birth-death dynamics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 33–50, May 2011.

[15] B. Sirmacek and C. Unsalan, "Urban-area and building detection using sift keypoints and graph theory," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 4, pp. 1156–1167, Mar. 2009.

[16] R. B. Irvin and D. M. McKeown, Jr., "Methods for exploiting the relationship between buildings and their shadows in aerial imagery," *IEEE Trans. Syst. Man Cybern.*, vol. 19, no. 6, pp. 1564–1575, Aug. 2002.

[17] C. Lin, A. Huertas, and R. Nevatia, "Detection of buildings using perceptual grouping and shadows," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR'94)*, 1994, pp. 62–69.

[18] C. Lin and R. Nevatia, "Building detection and description from a single intensity image," *Comput. Vis. Image Understand.*, vol. 72, no. 2, pp. 101–121, 1998.

[19] T. Kim, T. Javzandulam, and T.-Y. Lee, "Semiautomatic reconstruction of building height and footprints from single satellite images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS'07)*, 2007, pp. 4737–4740.

[20] T. Kim and J.-P. Muller, "Development of a graph-based approach for building detection," *Image Vis. Comput.*, vol. 17, no. 1, pp. 3–14, 1999.

[21] B. Sirmacek and C. Unsalan, "Building detection from aerial images using invariant color features and shadow information," in *Proc. 23rd Int. Symp. Comput. Inf. Sci. (ISCIS'08)*, 2008, pp. 1–5.

[22] H. Akcay and S. Aksoy, "Building detection using directional spatial constraints," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, 2010, pp. 1932–1935.

[23] Y.-T. Liow and T. Pavlidis, "Use of shadows for extracting buildings in aerial images," *Comput. Vis. Graph. Image Process.*, vol. 49, no. 2, pp. 242–277, 1990.

[24] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph. (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.

[25] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV'01)*, 2001, vol. 1, pp. 105–112.

[26] J. A. Shufelt, "Exploiting photogrammetric methods for building extraction in aerial images," *Int. Arch. Photogramm. Remote Sens.*, vol. 31, p. B6, 1996.

[27] V. J. Tsai, "A comparative study on shadow compensation of color aerial images in invariant color models," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 6, pp. 1661–1671, Jun. 2006.

[28] K.-L. Chung, Y.-R. Lin, and Y.-H. Huang, "Efficient shadow detection of color aerial images based on successive thresholding scheme," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 2, pp. 671–682, Dec. 2008.

[29] N. Shorter and T. Kasparis, "Automatic vegetation identification and building detection from a single nadir aerial image," *Remote Sens.*, vol. 1, no. 4, pp. 731–757, 2009.

[30] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285–296, pp. 23–27, 1975.

[31] J. Femiani and E. Li, "Graph cuts to combine multiple sources for feature extraction," in *Proc. IMAGE'14*, 2014.

[32] G. Bradskic, "The OpenCV Library," *Dr. Dobb's J. Softw. Tools*, 2000 [Online]. Available: http://code.opencv.org/projects/opencv/wiki/CiteOpenCV

[33] B. Özdemir, S. Aksoy, S. Eckert, M. Pesaresi, and D. Ehrlich, "Performance measures for object detection evaluation," *Pattern Recognit. Lett.*, vol. 31, no. 10, pp. 1128–1137, 2010.

John Femiani (M'07) received the Ph.D. degree in computer science from Arizona State University, Phoenix, AZ, USA, in 2009.

Currently, he is an Assistant Professor with the Department of Engineering, Arizona State University Polytechnic, Mesa, AZ, USA. His research interests include topics in computer graphics, visualization, computer vision, remote sensing, and image processing.

Er Li (M'13) received the B.S. degree in automation from Wuhan University, Wuhan, China, in 2007, and the Ph.D. degree in computer science from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2012.

From 2012 to 2013, he was a Postdoctoral Researcher with the Institute of Software, Chinese Academy of Sciences. He is currently a Postdoctoral Researcher with the Department of Engineering and Computing Systems, Arizona State University, Phoenix, AZ, USA. His research interests include image analysis, computer vision, and compute graphics.

Anshuman Razdan (M'01) received the B.S. degree in mechanical engineering from the Regional Engineering College, Kurukshetra, India, in 1986, the M.S. and Ph.D. degrees in mechanical engineering and computer science, in 1988 and 1995, respectively, from Arizona State University (ASU), Tempe, AZ, USA.

Currently, he is a Professor with the Division of Computing Studies and the Director of the Advanced Technology Innovation Collaboratory and the I3DEA Lab, ASU's Polytechnic Campus, Mesa, AZ, USA. His research interests include geometric design, computer graphics, document exploitation, and geospatial visualization and analysis. Dr. Razdan is a member of the IEEE Computer Society.

Peter Wonka received the Ph.D. degree in computer science and, in addition, the M.S. degree in urban planning from the Technical University of Vienna, Vienna, Austria, in 2001 and 2002, respectively.

Currently, he is a Professor with CEMSE Division, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia, and also an Associate Professor with Arizona State University (ASU), Phoenix, AZ, USA. He was a Postdoctorate Researcher with the Georgia Institute of Technology for 2 years. His research interests include topics in computer graphics, visualization, computer vision, remote sensing, image processing, and machine learning.