Appl Intell (2016) 45:443–457
DOI 10.1007/s10489-016-0762-6

Rapid building detection using machine learning

Joseph Paul Cohen[1] · **Wei Ding**[1] · **Caitlin Kuhlman**[1] · **Aijun Chen**[2] · **Liping Di**[2]

Published online: 19 March 2016
© Springer Science+Business Media New York 2016

Abstract This work describes algorithms for performing discrete object detection, specifically in the case of buildings, where usually only low quality RGB-only geospatial reflective imagery is available. We utilize new candidate search and feature extraction techniques to reduce the problem to a machine learning (ML) classification task. Here we can harness the complex patterns of contrast features contained in training data to establish a model of buildings. We avoid costly sliding windows to generate candidates; instead we innovatively stitch together well known image processing techniques to produce candidates for building detection that cover 80–85 % of buildings. Reducing the number of possible candidates is important due to the scale of the problem. Each candidate is subjected to classification which, although linear, costs time and prohibits large scale evaluation. We propose a candidate alignment algorithm to boost classification performance to 80–90 % precision with a linear time algorithm and show it has negligible cost. Also, we propose a new concept called a Permutable Haar Mesh (PHM) which we use to form and traverse a search space to recover candidate buildings which were lost in the initial preprocessing phase. All code and datasets from this paper are made available online (http://kdl.cs.umb.edu/w/datasets/ and https://github.com/caitlinkuhlman/ObjectDetectionCLUtility).

Keywords Building detection · Machine learning · Geospatial reflective imagery · Discrete object detection

✉ Joseph Paul Cohen
joecohen@cs.umb.edu

✉ Wei Ding
ding@cs.umb.edu

Caitlin Kuhlman
ckuhlman@cs.umb.edu

Aijun Chen
achen6@gmu.edu

Liping Di
ldi@gmu.edu

[1] Department of Computer Science, University of Massachusetts Boston, Boston, MA, USA

[2] Spatial Information Science and Systems, College of Science, George Mason University, Fairfax, VA, USA

1 Introduction

Rapid detection and classification of discrete objects such as buildings in geospatial imagery has many applications such as damage assessments by comparing before and after building detections [5, 10, 27]. Large scale change detection at an object level can enable computer assisted updating of maps by identifying new or removed objects between multiyear satellite imagery [3]. This could also allow for the next evolution of the USGS National Land Cover Database (NLCD) analysis [29]. Also, in a national security interest and in the funding motivation of this research, ontological analysis can be performed using the spatial arrangement of groups of buildings to identify large manufacturing, power generation, and weapons proliferation sites.

Problems restrict the usage of existing research which require camera alignment information (azimuth and zenith angles) and/or special equipment that captures near-infrared channels. Runtime is also a large factor which restricts the scale of deployment. In this work we present a combination of methods which have minimum imagery requirements (they work on common grayscale imagery) and provides

scale and rotation invariant detection with a relatively inexpensive computation.

The first contribution of this paper is our method does not depend on sliding windows to generate building candidates (Section 2.1). Building candidates are rectangles, identified by a center, height, width, and rotation, that likely contain a building. If these were generated using a brute force sliding window approach processing an image would be very expensive because the centers can be any pixel, the width and height can be any combination (non-overlapping), and the rotation can be between $0° - 180°$. We devise a linear time strategy utilizing building shadows as a major feature because they are high contrast straight 'L' shaped feature unique to man made objects [13, 15, 16].

The second contribution is how we align buildings in linear time to increase classification accuracy (Section 2.2). We utilize a summation of Gaussians each centered and scaled depending on the direction and magnitude of the vectors that form the contour around a building. We describe a linear time algorithm for computing this and show it has a negligible cost as well as a significant performance gain of up to 5 % accuracy.

The third contribution is our candidate Permutable Haar Mesh (PHM) search method that heuristically searches nearby candidate boxes to find buildings via a greedy graph search algorithm (Section 2.4). Because we utilize Haar contrast features [26] for their supreme performance; if our building candidate box does not properly cover the building it will not be considered a building because its feature distributions won't align to learned examples. The PHM approach is expensive and is not part of our rapid solution but can be employed to increase accuracy if it is really necessary.

2 Method

An overview of our method is shown in Fig. 1. First (in Fig. 1a) Canny edge detection is run using a range of threshold values. The Canny edge detection [6], a fast straightforward method, uses high and low thresholds to determine edges and using only one set of threshold values would not discover all buildings (Discussed in Section 2.1). Instead, all possible combinations of threshold values are used limited by a step size between the values. The resulting binary images are processed for contours (Fig. 1b) in linear time [7]. Each contour is considered a candidate. Some of the resulting contours are filtered out based on a minimum number of pixels that can be used for prediction and if they are redundant to other contours by only differing by less than 5 pixels.

These contours have their alignment (Fig. 1c) detected automatically (Note: in the figure a different building is used to illustrate this). Section 2.2 discusses the rotation method. The candidates are then automatically rotated to a standard alignment, scaled to a standard size, and converted to grayscale for Haar feature extraction (Fig. 1d). This rotation is so the Haar features will have more correlation when a model is built.

For every candidate, Haar features are extracted from fixed locations to capture contrast (Fig. 1e). Haar features have been successful and proved rapid and robust by [26]. To extract a Haar feature, a rectangle is first overlaid at a specific and consistent location on the image. The rectangle is split in half and the pixels inside each half are summed and subtracted from each other. The resulting value represents the contrast at that location in the image and can be compared to other images. Combinations of these features will be discriminative enough to build a model (Fig. 1f). This model can then be used to predictions when given unseen Haar feature values from a new test image.

To complement this method we present an optional step (due to computational cost) which is a novel candidate permutation method called a Permutable Haar Mesh (PHM) to increase recall of candidates via greedy graph search (Section 2.4). Recall is an evaluation metric representing how many buildings have not been missed, this metric is complementary to precision which represents how correct each prediction is. Candidates are surrounded by a bounding box and permuted by moving their top, bottom, left, and right boundaries in order to properly cover a candidate and capture buildings that would otherwise have been missed because the candidate didn't properly cover the building.

2.1 Candidate generation

We utilize building shadows as a major identifier of buildings because they are a high contrast feature which provides largely straight 'L' shaped contours unique to manmade objects [13, 15, 16]. Canny edge detection [6] is still the state of the art edge detection method that can capture these shadows well. The result of Canny edge detection is a binary image representing the edges of the input. Candidates are isolated by applying a linear time contour generation algorithm [7] which groups together edge pixels and returns a set of vectors that trace along these edges forming a contour. Each contour is considered to be a candidate building, we will also call the derived forms of this contour a candidate such as a bounding box around the contour and the image pixels within this bounding box.

Canny edge detection has two hyperparameters, a high and low threshold for hysteresis thresholding. Canny edge detection works by first computing the gradient of the image using the contrast between pixels (scaled between 0 and 1). Gradients below the low threshold are filtered out and will

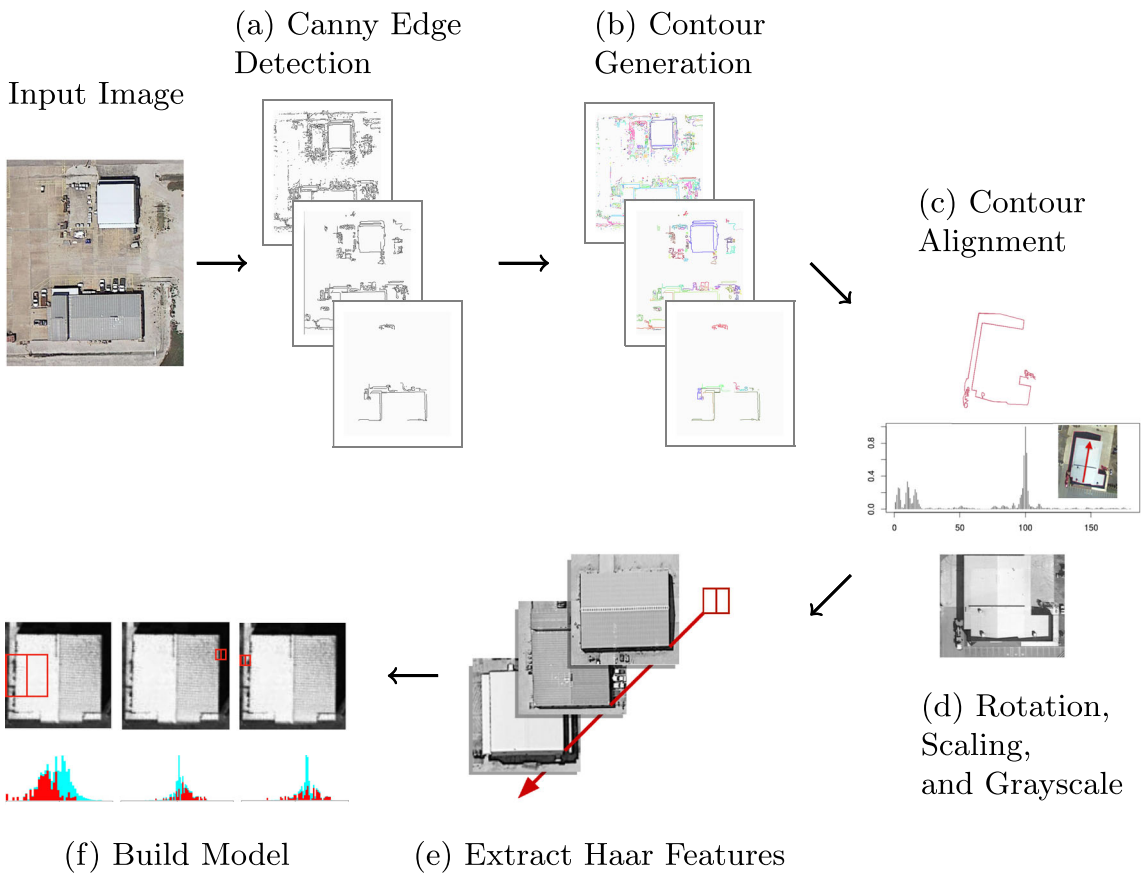

Fig. 1 An overview of the method is shown. First **a** Canny edge detection is run using a range of threshold values. The resulting binary images are processed for contours (**b**). Each contour is considered a candidate. These contours have their alignment detected (Note: a different building is used to illustrate this) (**c**). They are then rotated to standard alignment, scaled to a standard size, and converted to grayscale (**d**). For every candidate, Haar image masks are extracted from fixed locations to capture contrast (**e**). Next these contrast values are discriminative enough to build a model and make accurate predictions (**f**)

not be considered edges. Gradients above the high threshold are set as edges and any remaining gradients that are connected to edges are set as edges. One combination of parameters will likely not return correct candidates for all buildings in an image as shown in Fig. 2 because too high of a threshold can cause gradients of objects that neighbor buildings to become part of its contour while too low of a threshold may cause the gradients of a building not to be considered. These issues are almost always the case when buildings vary in size in the same image because gaps in high gradients along the side of a building require lower thresholds which will cause smaller buildings to be connected to neighboring objects.

In order to be scale invariant the union of the resulting contours from many different combinations of Canny threshold parameters are used to form the set of candidates. If the candidates generated in Fig. 2 from the three different pairs of threshold values are merged together then all buildings will be included in the candidate set. However, as more threshold values are included, more non-buildings are included as well and create a challenge to later steps. Threshold values are chosen from a grid which is parametrized by a step size which controls the density of the grid. As the step size is decreased, more threshold values are included which results in more candidates. Section 3.1 studies the trade-off when decreasing the step size in order to maximize precision and recall.

2.2 Building contour alignment

Contours resulting from Chang's contour detection [7] are represented by a set of vectors c and each component vector c_i. From these vectors we want to determine the aggregate direction of the object they represent. By rotating these candidates into alignment before extraction of the Haar features, the features become more discriminative and will result in an increase in accuracy of the trained classifier (explained in Section 2.3).

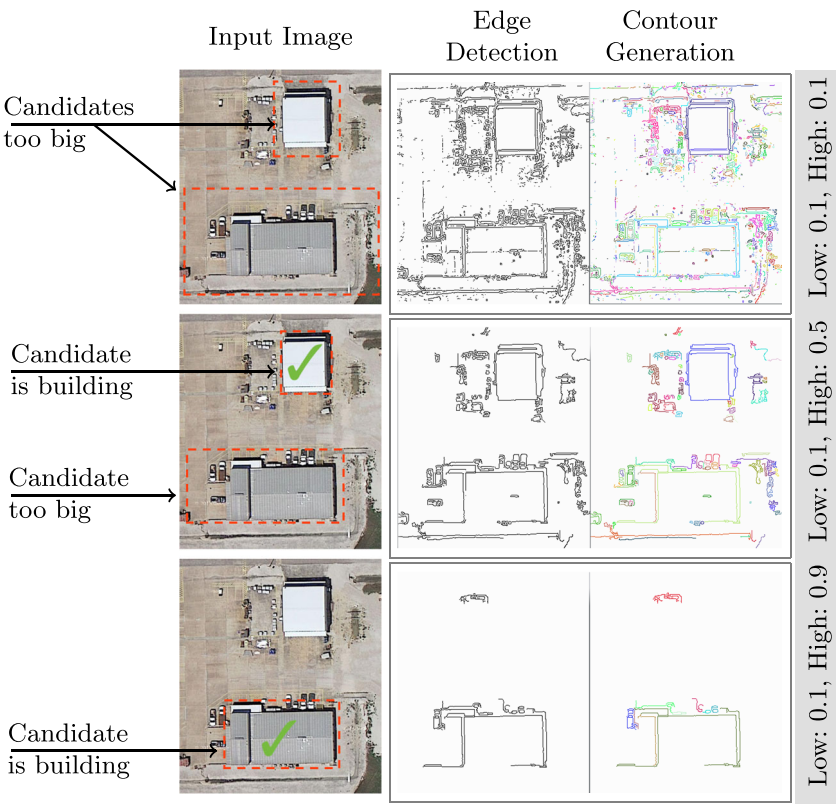

Fig. 2 This figure shows the application of Canny edge detection (*center*) and contour detection (*right*) at various threshold values to generate candidates. *Red dashed* boxes on the left show candidates that enclose buildings and *green check* marks are candidates that will be classified as buildings. As the high threshold parameter to the Canny edge detector is varied from 0.1 at the top to 0.9 at the bottom different contours are generated. There is no perfect parameters to generate correct candidates for both buildings

Determining the aggregate direction is difficult because buildings may not have their walls parallel to each other and the edge and contour detection methods may have introduced noise in the vector directions. Consider the simple example in Fig. 3; suppose we have a contour made up of four vectors with the following directions and magnitudes $(30°, 5)$, $(31°, 5)$, $(120°, 3)$, $(120°, 3)$ which would appear to be a rectangle with the longest side as the dominant edge. If the assumption is made that the majority of the walls length will point in the dominant direction of the building then we should be able to sum the vectors with the same angle to determine which angle the majority of the walls align with. However, taking the sum for each direction would not capture the similarity of $angle(c_i) = 30°$ and $31°$. They would be considered independent and (1) would result in $120°$ as the dominant direction of the building which is false.

$$argmax_\theta \sum_{c_i \in c} \{|c_i| : \theta = angle(c_i)\} \tag{1}$$

We need to tolerate this noisy data and take these situations into account because contours can be even more complex and misleading as seen in Fig. 4. To accomplish this we use a method similar to a kernel density estimation which utilizes a sum of Gaussian distributions, one for each vector's degree normalized by its magnitude, shown in (2).

$$argmax_\theta \sum_{c_i \in c} \left(\underbrace{\left(\frac{|c_i|}{\sum_{c_i \in c} |c_i|} \right)}_{\text{Normalized Magnitude}} \underbrace{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\theta - angle(c_i)}{\sigma}\right)^2}}_{\text{Gaussian on orientation}} \right) \tag{2}$$

To determine the alignment direction we evaluate the summation for a specific input degree from $0°-180°$. Algorithm 1 formalizes this method. For each contour segment c_i the angle α is determined using the arctangent. The Gaussians are normalized based on their magnitude to the sum of all magnitudes. The maximum θ is then found by iterating over 180 possible angles. Figure 4 shows this method not only handles the specific issue we discussed of non parallel walls but also tolerates noise in the contour data. Noise meaning jitter in the angle of the vectors as they wrap around the building. This can be due to pixelation error during capturing the image, contours containing vectors that don't overlap the building walls, or non-rectangular building shapes. This rotation method not only increases classification accuracy but does so with negligible increase in time (shown in Section 3.5).

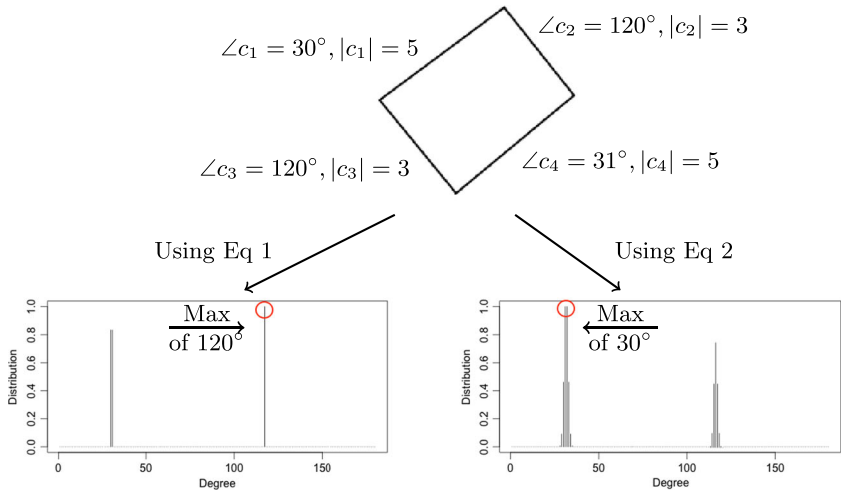

Fig. 3 Comparing the direction information obtained from the two discussed equations we can see a disagreement. The input contour contains four vectors ($c_i, 1 \leq i \leq 4$) Eq. 1 results in an aggregate angle of 120° while (2) results in a more expected direction of 30.5° because it is the mean of the angles. For our application rounding to 30° and 31° would both yield satisfactory features

Algorithm 1 Rotate Building Candidate

Input: Contour c
Output: Rotated Contour c_{rot}

$K \leftarrow \{\}$ // Set of Gaussians
for $c_i \in c$ **do**
  $xdist \leftarrow c_i.x - c_{(i+1) mod |c|}.x$
  $ydist \leftarrow c_i.y - c_{(i+1) mod |c|}.y$
  if $ydist < 0$ **then**
    $xdist \leftarrow -(xdist)$ // keep angle between 0° and 180°
    $ydist \leftarrow -(ydist)$
  $\alpha \leftarrow atan2(ydist, xdist)\frac{360}{2\pi}$ // angle of contour segment
  $\sigma \leftarrow 1$ // stdev of Gaussian distribution
  $\lambda \leftarrow \frac{|c_i|}{\sum_{c_i \in c} |c_i|}$ // normalization factor
  $f_{c_i, c_{(i+1) mod |c|}}(\theta) = \lambda \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\theta - angle(c_i)}{\sigma}\right)^2}$
  $K \leftarrow K \cup \{f_{c_i, c_{(i+1) mod |c|}}(\theta)\}$
$c_{angle} \leftarrow max_\theta(\Sigma_{f \in K} f(\theta))$
$c_{rot} \leftarrow rotate(c, c_{angle})$

2.3 Building candidate feature construction

To build a classification model that can filter candidates into building and non-building, we need features that can discriminate effectively and are efficiently computed. Haar features have been shown to quickly capture discriminative contrast patterns effectively [26]. They are generated by taking a rectangular image mask and dividing it into two rectangles, with a horizontal or vertical division. The sum of the pixel values in one rectangle are subtracted from the sum of the pixel values in the other. Haar features are discriminative in face and crater detection [8] because these domains have similar contrast at specific positions of the candidates. In this work each candidate is scaled to 200 × 200 pixels before Haar features are extracted. Horizontal and vertical Haar features are extracted in a sliding window fashion which extracts square regions from the image systematically from the top left to the bottom right. Square regions are extracted with pixel width 40, 80, and 100 and are applied with a step size of 10 pixels. Also, square regions are extracted with width 20 with a step size of 5 pixels in order to capture small details. This yields a total of 3592 features. Each feature represents the horizontal or vertical contrast in that region with a signed integer value. A value of 0 means no contrast where a positive or negative value represents contrast in the positive or negative direction. The sign of the number is dependent on the order of the subtraction during extraction and is only useful for comparison.

By aligning buildings and adding padding to expose its edges, which have high contrast, we are able to obtain contrast patterns between candidates. For example the Haar features being extracted in Fig. 5a will statistically expose higher contrast in candidates which contain buildings due to the edges of this building appearing in the same location across examples. Also, roof texture and the surrounding area texture may also be consistent enough to provide linear separable distributions of values with respect to a building and

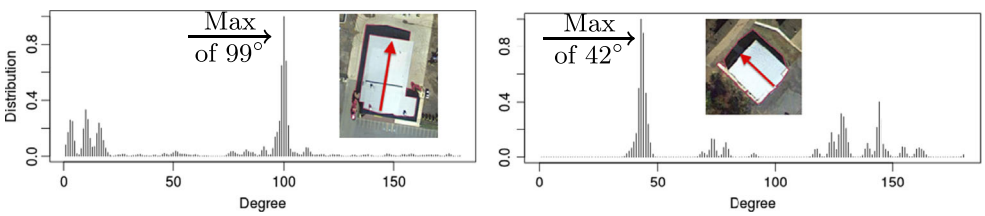

Fig. 4 Histograms of the Gaussian summations of contour components evaluated at specific angles when Algorithm 1 is applied to candidates A (*left*) and B (*right*). Our method correctly identified Candidate A at a 99 degree angle and candidate B at a 42 degree angle

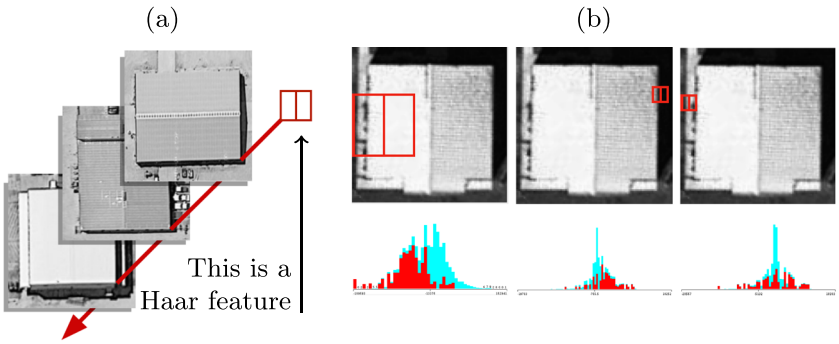

Fig. 5 **a** Example of a Haar feature being extracted from building candidates at the same position on multiple candidates in order capture the contrast at the edge of the building. **b** The three highest weighted Haar features of a Linear AdaBoost classifier in descending from left to right. The distribution of values extracted from Dataset A for each feature is shown at the bottom to show their linear separability

non-building. In order to gain more insight we analyze the top weighted Haar features in the Linear AdaBoost classifier in Fig. 5b where it can be seen that edges of buildings are very discriminative. We are able to conclude that the statements from previous work that find shadows a dominant feature are correct. Shadows will generally exist at the edges of buildings and provide strong contrast values at the edge of the roof where the shadow begins. Together, many of these features allow us to obtain a linear separable feature space to achieve accurate classification. One problem that arises from using these features is when buildings have black roofs the contrast between the roof and the shadow is very low and might appear to be very similar to a solid surface.

2.4 Candidate permutation search (PHM)

Some candidates are lost during the initial preprocessing step due to contours that cover part (or too much) of the building as shown in Fig. 6 This leads to a misalignment of Haar features.

To solve this problem we present a Permutable Haar Mesh (PHM) algorithm which iteratively permutes the building candidate using a custom heuristic function to search the space shown in Fig. 7. We perform a multi objective greedy search (for speed) using the following function (for accuracy) based on the result of a classifier:

$$\mathbb{H}(can, \mathcal{L}) = \frac{2\overbrace{\mathcal{L}^{+}(can)}^{\text{P(bldg)}}(1-\overbrace{\mathcal{L}^{-}(can)}^{\text{P(not bldg)}})}{\mathcal{L}^{+}(can)+(1-\mathcal{L}^{-}(can))}$$

Here we take the harmonic mean of $\mathcal{L}^{+}(can)$, the probability that *can* is a building, and $(1-\mathcal{L}^{-}(can))$, the complement of the probability that *can* is not a building. Using a greedy search we evaluate each permutation and select the best increase in probability at each step of the iteration until we cannot improve the hypothesis probability. This method is outlined in Algorithm 2.

Algorithm 2 Greedy Candidate Permutation Search

Input: Candidate *can*
Permutation Rate r
Heuristic function $\mathbb{H}$
Output: Best Candidate *best*

$d = r \cdot \frac{(bottom-top)+(right-left)}{2}$
$P \leftarrow \{can + (d \cdot p) | p \in \{ \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} | x_i \in \{0, -1, 1\}\}$
$best \leftarrow argmax_p(\mathbb{H}(p))$ where $p \in P$
if $best \neq can$ **then**
  return $permute(best)$ *//if $\mathbb{H}$ improved continue search*
else
  return *best*

Fig. 6 Example of contours that overdetected a candidate. The *green* and *red lines* are the contour lines. The *bounding boxes* can be repositioned to detect these buildings

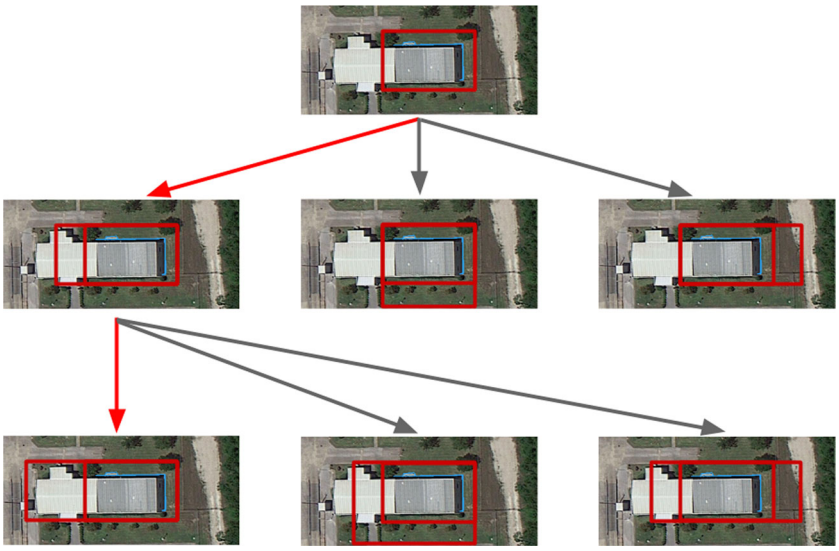

Fig. 7 An example of the PHM search space being traversed in a greedy manner. Each potential permutation becomes a link which represents a new frame that Haar features are extracted from. The *red lines* indicate the path taken during the search to cover candidates

2.5 Complexity

Our method is $O(n)$ for generating candidates which places the training complexity on the classifier used. Each candidate generated as a negative example adds to the complexity. This can be reduced by generating less negative examples but this may also generate a classifier with lower performance.

When utilizing the classifier our method is $O(n)$ in terms of pixels or candidates. In the worst case every pixel could be considered a candidate which would be determined in linear time using Canny edge detection and Chang's linear contour detection, we call this n. When sampling and merging using a specific step we incur a fixed cost dependent on the step size chosen. For 0.05 this is 400 leading to a potential $400n$ candidates to evaluate. Our rotation method is based on the number of vectors in the contour (c) of the candidate. The maximum number of contours would be the number of pixels in the candidate. Our approximation method solves this in $360|c|$. Each candidate then has a fixed number (4, 240) of Haar features extracted which is one initial cost of the candidates pixels for an integral image and then 4 additions per Haar feature. When using a linear classification model, such as Naive Bayes or AdaBoost on linear decision stump classifiers, each candidate can then be classified in linear time.

3 Experimental evaluation

In order to evaluate our method we looked for publicly available datasets that would allow us to study the errors when applying methods to the average residential buildings as well as unique industrial buildings. Mnih and Hinton [19] has generated a benchmark dataset using MassGIS which contains average residential buildings but industrial buildings such as coal and nuclear power plants are not released by MassGIS. Because of this we have built a dataset of nuclear power plant buildings that can be shared with the research community. We utilize these two datasets in order to showcase the robustness of our algorithm on imagery with various quality and content. All code and datasets from this paper are made available online (http://kdl.cs.umb.edu/w/datasets/ and https://github.com/caitlinkuhlman/ObjectDetectionCLUtility).

Dataset A (Fig. 8a) was constructed using images from Google Maps[1] with various resolution, size, illumination, geographic region, building size, and building purpose. There are 411 buildings in this dataset which are mostly non-residential including large industrial and power generation. These buildings can be very unique to a specific purpose and vary greatly in size.

Dataset B (Fig. 8b) is a labelled subset of the dataset used in [19].[2] We used a higher resolution (15 cm/pixel) version of the same imagery acquired from MassGIS (coq2008_15cm_jp2). All buildings have the same illumination. This dataset is of a contiguous area composed of mostly residential buildings. In total there are 1337 buildings.

We use these datasets to first evaluate the recall obtained by our method. Recall is an evaluation metric representing how many buildings have not been missed, this metric is complementary to precision which represents how correct each prediction is. After this we discuss how our positive and negative examples are constructed to train a classifier. This is followed by an analysis of candidate alignments effect on these examples on various classifiers. We then

[1] https://maps.google.com/

[2] http://www.cs.toronto.edu/~vmnih/data/

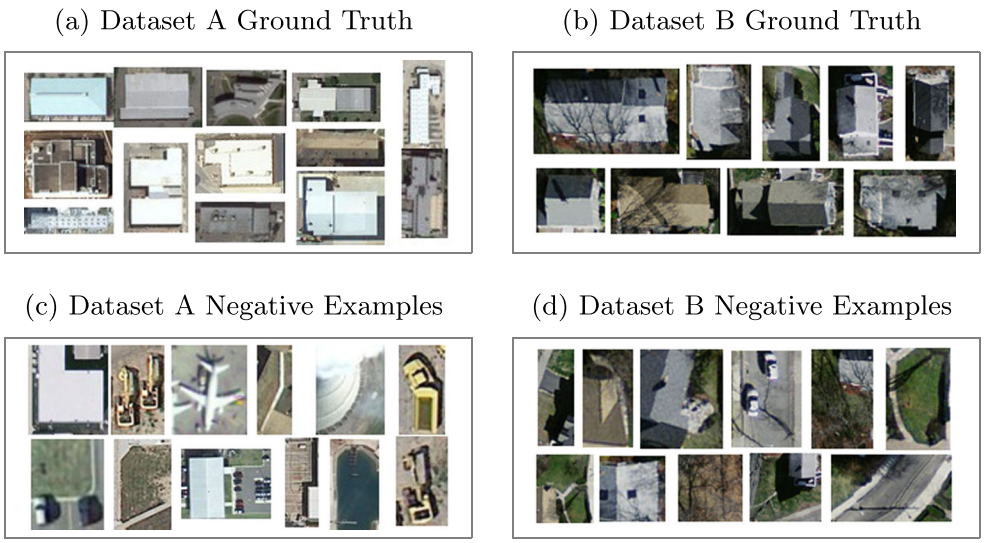

Fig. 8 Shown here are samples images of the two datasets used in analysis. All images are automatically cropped and rotated based on their contours. At the *top* we have ground truth buildings and at the *bottom* are negative examples

discuss how we can increase recall with our PHM method which can recover candidates and achieve better accuracy at the cost of a more computationally expensive method. Finally we evaluate the runtime of different components of our algorithm.

3.1 Candidate recall

It is important that we achieve high recall in order to not miss any potential buildings using our candidate generation method. Unfortunately there are some complications that we had to overcome. Using a single high and low Canny threshold value we are only able to achieve low recall values. In Fig. 9 we explore all possible configurations of low and high threshold values on dataset A. These results show a strange surface due to a trade off of capturing different sizes of the buildings. There does seem to be a peak but it is very low $\approx$ 60 %. Some buildings are only identified as candidates at specific threshold values so changing them misses some while finding others. The problem is that these values are not the same for every building in a dataset as shown in Fig. 2. This observation leads us to our solution, because some buildings are only captured by different threshold values.

To solve this problem we generate candidates by **sampling and merging** the results of candidate generation at many different threshold values. The question now is what Low/High threshold values to use. We experiment with various step sizes through the space (0,0) to (1,1) in Fig. 10. As the step size is reduced from 0.2 to 0.01 the recall increases at a diminishing rate. However, there is trade-off that must be made when choosing a small step size. In Fig. 11 the total number of candidates that must be evaluated is analyzed. As the step size is reduced the total number of candidates increases to numbers that are much larger than the number of buildings that exist in those images. This may not only increase running time but also decrease overall performance by increasing the chance that a classifier may misclassify.

To put more context on Fig. 10, in dataset A we start with 411 labeled buildings and our preprocessing step is able

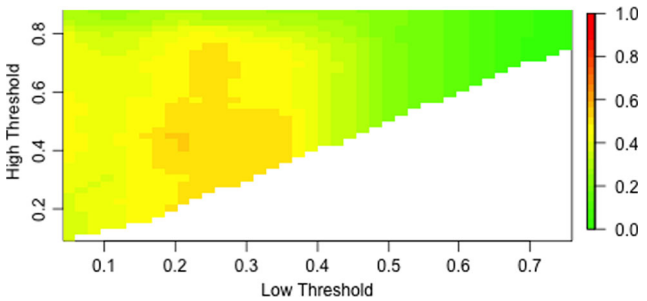

Fig. 9 Here all possible high and low threshold values from 0,0 to 1,1 for the Canny edge detector are evaluated on dataset A with step size 0.05. The recall value is plotted and we can observe a spike at 0.2,0.4. Further inspection reveals that different buildings are being captures at different combinations resulting is no one maximizing combination of threshold values

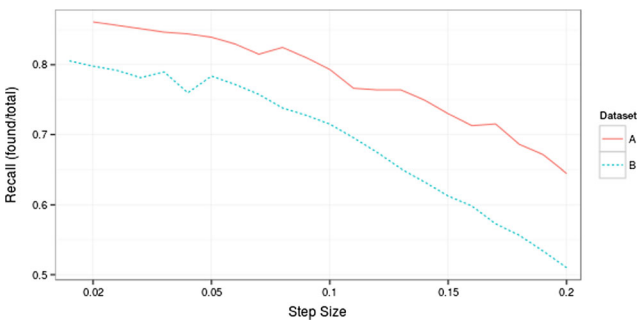

Fig. 10 We vary the step size used to generate candidates. As we decrease the step size, meaning more samples, the recall increases and we are able to capture more of the buildings

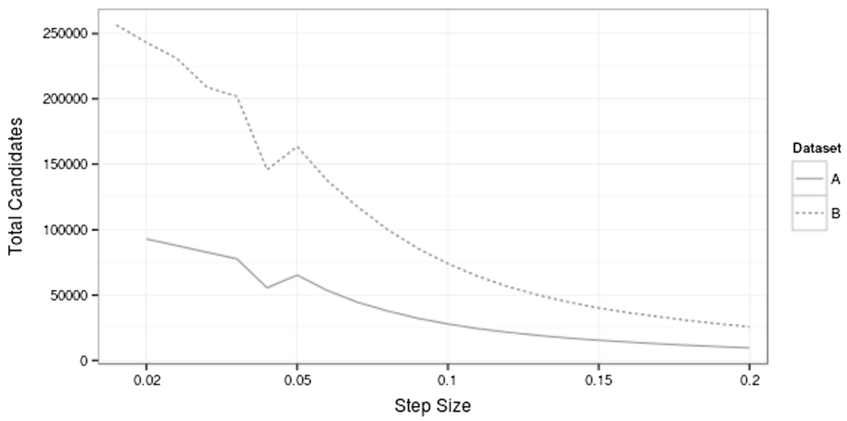

Fig. 11 We vary the step size used to generate candidates. As we decrease the step size in order to gain higher recall the number of candidates increases

to find 86 % when generating about 90,000 candidates. In dataset B we start with 1,337 labeled buildings and our preprocessing step is able to find 80 % when generating about 240,000 candidates. To put this in perspective, without this preprocessing step, because the centers can be any pixel, the width and height can be any combination (non-overlapping), and the rotation can be between $0° - 180°$, a small 1,000 x 1,000 image can easily generate over 1 billion candidates using a sliding window for just one image in order to achieve 100 % recall.

3.2 Training set construction

To learn an accurate classifier requires constructing a training set containing difficult realistic examples of what will be presented to the classifier during testing. We run the candidate generation process and subtract the positive examples. This process includes candidates that partially overlap the ground truth in order to train on examples that may be misclassified during testing. Our goal is to select strong representative examples that we expect to reside near the decision boundary of a classifier.

For all the evaluations following this section, 10-fold cross validation is used to calculate the F1-Score obtainable with a classifier. We define the F1-Score as follows:

$$F1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Dataset A has 383 positive and 4,912 negative examples. Dataset B has 992 positive and 11,488 negative examples. The number of positive examples is less than the total ground truth number because some candidates are excluded because the 5 % padding that is added goes out of the image bounds and is not included. The datasets are balanced in order for the classifiers to properly learn. This is done by randomly sampling with replacement to add duplicates to the positive examples.

All experiments are performed with the AdaBoost classifier unless otherwise noted. In the next section we compare many different classifiers. The Weka implementations of these algorithms are used with their default values.

- **AdaBoost** is an ensemble of weighted linear classifiers with one feature each. The classifier is trained for 10 epochs with a weight threshold of 100 to prune the weights [12].
- **Naive Bayes** assumes all variables are conditionally independent with respect to the class label. This classifier then simply uses Bayes' rule to determine the probability of a class attribute given feature values [14].
- **J48** constructs a decision tree by iteratively splitting each tree node if classification error is reduced when discriminated by an attribute. The Weka version is based on the C4.5 implementation by Quinlan and uses the default confidence value of 0.25 to estimate error [22].
- **Random Forest** constructs decision trees from subsets of features which are drawn uniformly with replacement from the global feature set. 100 trees are constructed. Each decision tree is constructed similar to J48. The resulting classification is a majority vote by all trees for a class label [4].
- **Support Vector Machine**: The Weka LibSVM implementation of C-SVC was used as described by [9]. A radial basis kernel was used with the parameters $\nu = 0.5, \gamma = 0, loss = 0.1, cost = 1$.

3.3 Rotation impact on classifiers

Analysis is performed to evaluate the effect of rotating candidates on the overall pipeline. To demonstrate the versatility of this step we evaluate many classifiers. In Fig. 12 it can be observed that rotating candidates increases the F1-Score of standard classification algorithms.

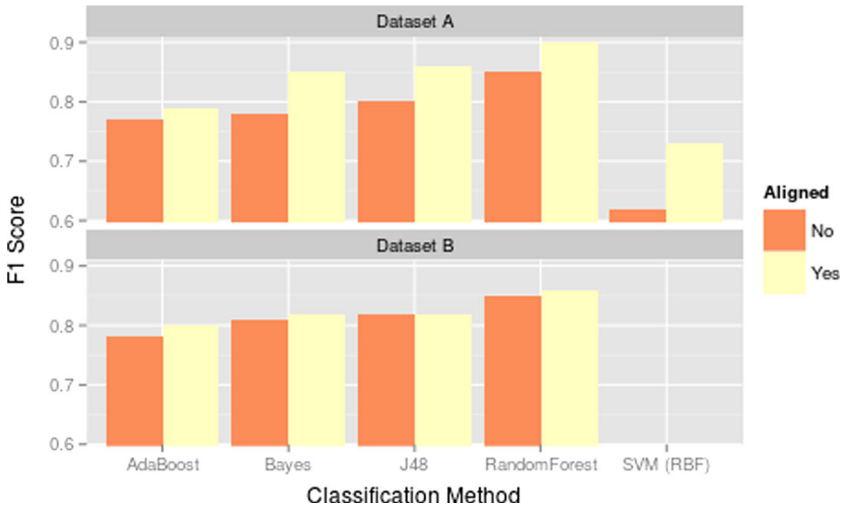

Fig. 12 We compare AdaBoost (with linear decision stump classifiers), Naive Bayes, J48 Decision Trees, Random Forest, and SVM (with a radial basis function kernel) classifiers applied to datasets A and B via their F1-Score with and without rotation of the candidates

To evaluate the following classification methods we generate candidates from each training set using the sample and merging method with step size 0.05 and form an isolated set of candidate images so that 10-fold cross validation can easily be performed. The results here are the metrics from these isolated sets and therefore don't reflect the impact of recall loss from the preprocessing method which is analyzed in Section 3.1.

We evaluate AdaBoost because it was used as part of the Viola and Jones face detection pipeline [26]. AdaBoost is expected to be well suited for this task because it performs feature selection on the many Haar features generated from the candidate in both situations. This is however not the case. AdaBoost ranks among the worst classifiers evaluated.

We evaluate Naive Bayes and J48 Decision Tree classification models as baselines which are quick to train that we expect the reader will be familiar with. A random classifier was used to confirm 50 % F1-Score indicating balanced training data. We also evaluate Random Forest and find it to outperform all other methods.

The previous classification models discussed can rapidly be trained and utilized in comparison to a Support Vector Machine (SVM) with a non-linear kernel. We were able to evaluate Dataset A using an SVM with a radial basis function kernel. However, due to the computational cost we are unable to evaluate Dataset B using an SVM because the algorithm did not terminate in 72 hours. It is interesting how poorly the SVM model performs. We can speculate that it may be caused by noisy or irrelevant Haar features. A large amount of features may cause the classifier to weight features inappropriately and skew classification. The increase in performance after candidate rotation may indicate this as it causes features to have a higher discriminative ability which can more easily be separated.

Overall, every classification method had its F1-Score increase after the alignment of candidates. The most significant increase was for an SVM classifier.

3.4 Best PHM permutation rate

The primary goal of our preprocessing method is to maintain high recall. If candidates are still missed we can use the PHM method to salvage over/underdetected candidates as outlined in Section 2.4. This method is analyzed in Fig. 13

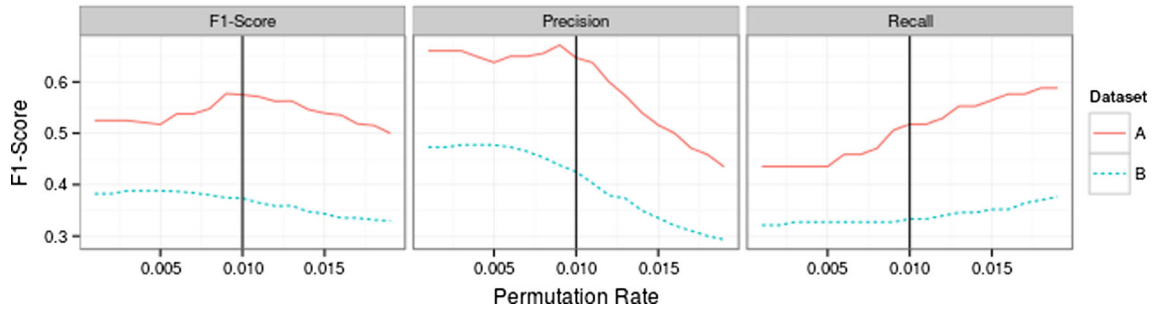

Fig. 13 Our pipeline using Canny threshold values of low:0.2/high:0.4 varying the permutation rate on both datasets. A permutation rate of 0.01 is able to increase recall while maintaining precision to yield a higher F1 value

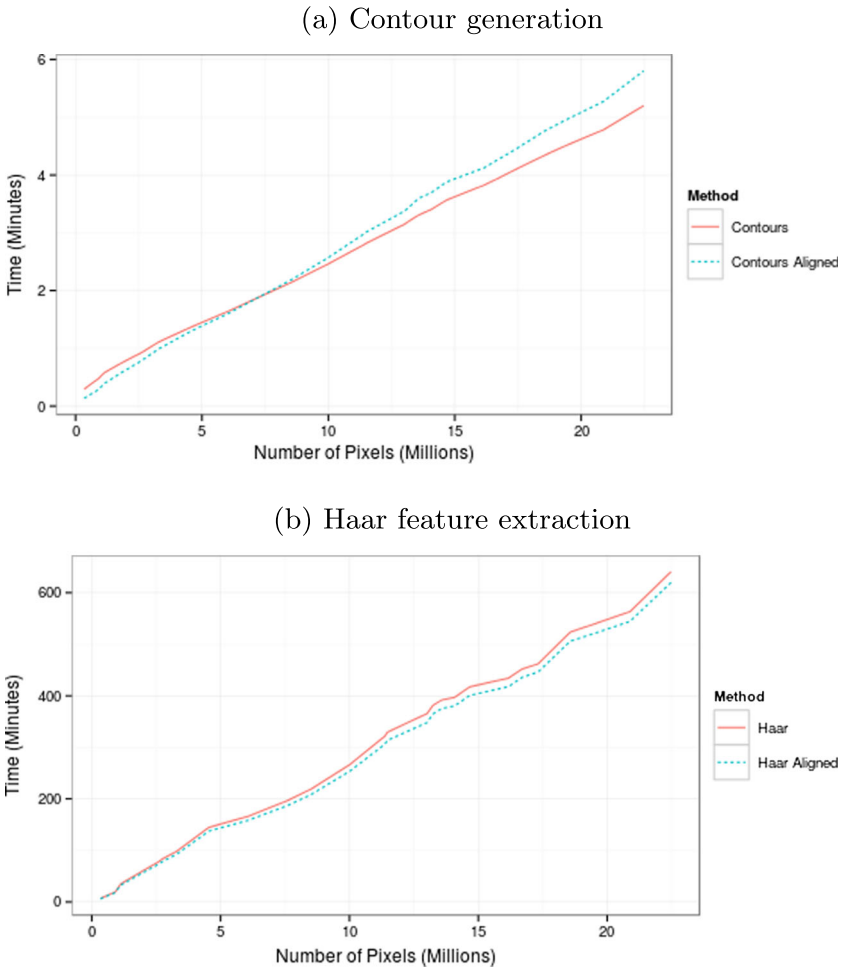

Fig. 14 Here we show the impact of rotation on runtime during the contour generation (**a**) and Haar feature extraction (**b**) parts of the process

to study how the F1-Score is impacted as the permutation rate is increased. For these experiments we used one combination of high and low Canny threshold values instead of merging many values together which yields lower recall values from the start.

In Fig. 13 as the rate of permutation increases so does the recall. However, similarly as the permutation rate increases the precision falls. The increase in precision error is due to more candidates being presented to the classifier which appear to be buildings as a result of the PHM process itself.

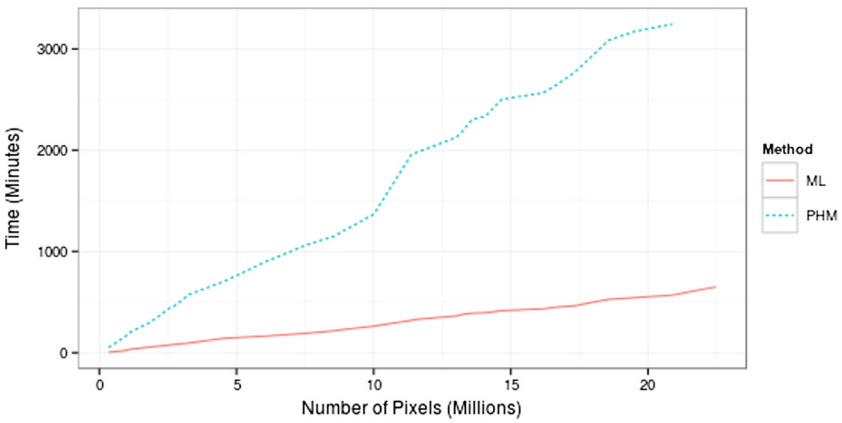

Fig. 15 Here the runtime is evaluated using the complete pipeline for our ML and PHM methods

A compromise is found at the peak of the F1-Score plot of 0.01.

3.5 Linear time feature extraction

Our machine learning pipeline runs in linear time as theoretically explained in Section 2.5. We empirically evaluate the runtime on a single 3.07 GHz Intel Xeon CPU. However many parts of the algorithm are easily made parallel to achieve major speed improvements.

The first way to empirically show this is during the initial contour extraction phase analyzed in Fig. 14a. Here images are processed one after another, the total number of pixels processed is plotted against the time taken. Here it is observed that aligning the contours only slightly increases the processing cost.

In Fig. 14b we perform the same evaluation but allow the process to continue to the step of extracting Haar features from every candidate. A strange result is that it takes less time when we add the rotation step. An answer for this may

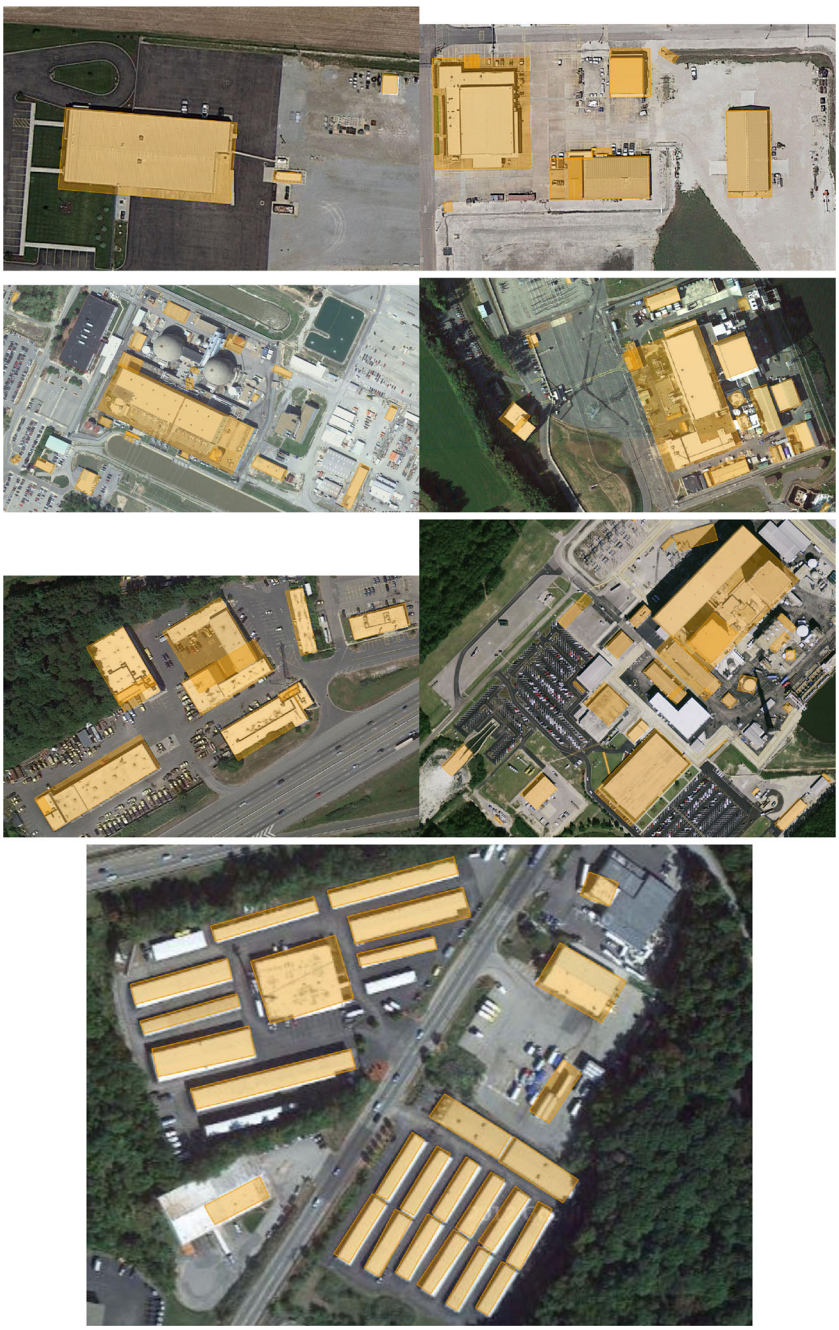

Fig. 16 Some images from Dataset A are analyzed with our machine learning method using an AdaBoost classifier. Predictions are highlighted in *yellow*. On these examples we detect over 90 % of the buildings except on heavily clustered buildings around nuclear power plants which present a difficult task because candidate building borders abut each other and prevent shadows

be that the scaling phase before Haar features are extracted is sped up because images contain less edges on diagonals.

In Fig. 15 we evaluate the entire pipeline and observe that our basic machine learning (ML) approach appears significantly faster than PHM. For every candidate encountered during the algorithm the PHM will search possibly 100's of surrounding candidates to find a better match. From our experience the machine learning approach appears to work in almost realtime on reasonably sized images.

4 Related work

Automated labeling of aerial images has been a motivating problem for researchers for a very long time [13]. The development of an automated system to identify discrete objects, such as buildings, has been a much sought after goal. Many techniques from the field of computer vision have been employed, as well and statistical machine learning approaches. A number of surveys including [1, 11, 18] indicate the depth of this field.

Unlike our method which relies only on RGB images, much work has been done using very high spatial resolution (VHR) multispectral data, [25] synthetic aperture radar (SAR) data [23] and light detection and ranging (LIDAR). This information has been used to filter out sections of images corresponding to non-building areas such as vegetation or water. Information such as azimuth and zenith angles has been used to calculate the shadow locations and near infrared to better determine building shadows from plant shadows [21].

Working only with images, other researchers have explored techniques using many different types of features that can capture texture information, color, shape, and contextual information. Simple features can be built using the color and intensity of pixels, and gradient based features have also been used. Local scale and rotation invariant features like Lowe's SIFT [17] and the sped up version SURF [2] have been evaluated [24, 30].

Shadows have been picked up as a powerful building indicator that can be identified by simple algorithms similar to ours [13, 28]. Machine learning has been employed extensively, with various systems using features to train classifiers such as Support Vector Machines [20]. Lately, deep learning techniques such as Convolutional Neural Networks have been used to good effect [19].

Our method stands out from these other approaches because of our focus on speed and applicability to all geospatial imagery because our method only needs pure RGB images and does not require a near-infrared channel or azimuth and zenith angles. Also, unlike other methods we provide an implementation of our method.

5 Conclusion

In this paper we describe algorithms for reducing discrete object detection in reflective geospatial imagery to machine learning, specifically in the case of buildings. Results from the application of this method are shown in Fig. 16. We have shown the complex patterns of a discrete object's contrast features can be learned using state of the art ML methods. The reduction requires non-trivial ML-aware preprocessing methods. We have shown that these methods consistently increase the performance of classification algorithms. We also present the concept of a PHM in order to recover candidates that fail to be classified correctly. This method generates a search space which has potential to greatly increase detection rates and requires further research to fully utilize beyond what is explored in this paper.

Acknowledgments This work is partially funded by a grant from the National Nuclear Security Agency of the U.S. Department of Energy (grant number: DE-NA0001123) as well as by the National Science Foundation Graduate Research Fellowship Program (grant number: DGE-1356104). This work utilized the supercomputing facilities managed by the Research Computing Department at the University of Massachusetts Boston as well as the resources provided by the Open Science Grid, which is supported by the National Science Foundation and the U.S. Department of Energy's Office of Science.

References

1. Baltsavias EP (2004) Object extraction and revision by image analysis using existing geodata and knowledge: current status and steps towards operational systems. ISPRS J Photogramm Remote Sens 58(3):129–151
2. Bay H, Tuytelaars T, Van Gool L (2006) Surf: Speeded up robust features. In: Computer vision ECCV 2006, pp 404–417. Springer
3. Bonnefon R, Dherete P, Desachy J (2002) Geographic information system updating using remote sensing images. Pattern Recogn Lett 23(9):1073–1083
4. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
5. Brunner D, Lemoine G, Bruzzone L (2010) Earthquake damage assessment of buildings using VHR optical and SAR imagery. IEEE Trans Geosci Remote Sens 48(5):2403–2420
6. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 6:679–698
7. Chang F, Chen C-J, Lu C-J (2004) A linear-time component-labeling algorithm using contour tracing technique. Comput Vis Image Underst 93(2):206–220
8. Cohen JP, Ding W (2013) Crater detection via genetic search methods to reduce image features. Adv Space Res 53(12):1768–1782. doi:10.1016/j.asr.2013.05.010
9. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
10. Dong L, Shan J (2013) A comprehensive review of earthquake-induced building damage detection with remote sensing techniques. ISPRS J Photogramm Remote Sens 84:85–99
11. Druaguct L, Blaschke T (2006) Automated classification of landform elements using object-based image analysis. Geomorphology 81(3):330–344

12. Freund Y, Schapire RE (1996) Experiments with a New Boosting Algorithm
13. Irvin RB, McKeown DM (1989) Methods for exploiting the relationship between buildings and their shadows in aerial imagery. In: OE/LASE'89, 15–20 Jan., Los Angeles. CA, pp 156–164
14. John GH, Langley P (1995) Estimating continuous distributions in bayesian classifiers. In: Proceedings of the eleventh conference on uncertainty in artificial intelligence, UAI'95. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 338–345
15. Karantzalos K, Paragios N (2009) Recognition-driven two-dimensional competing priors toward automatic and accurate building detection. IEEE Trans Geosci Remote Sens 47(1):133–144
16. Lin C, Nevatia R (1998) Building detection and description from a single intensity image. Comput Vis Image Underst 72(2):101–121
17. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
18. Mayer H (1999) Automatic object extraction from aerial imagery: a survey focusing on buildings. Comput Vis Image Underst 74(2):138–149
19. Mnih V, Hinton GE (2010) Learning to detect roads in high-resolution aerial images. In: Computer visionECCV 2010, pp 210–223. Springer
20. Mountrakis G, Im J, Ogole C (2011) Support vector machines in remote sensing: a review. ISPRS J Photogramm Remote Sens 66(3):247–259
21. Ok AO (2013) Automated detection of buildings from single VHR multispectral images using shadow information and graph cuts. ISPRS J Photogramm Remote Sens 86:21–40
22. Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco
23. Simonetto E, Oriot H, Garello R (2005) Rectangular building extraction from stereoscopic airborne radar images. IEEE Trans Geosci Remote Sens 43(10):2386–2395
24. Sirmacek B, Unsalan C (2011) A probabilistic framework to detect buildings in aerial and satellite images. IEEE Trans Geosci Remote Sens 49(1):211–221
25. Sohn G, Dowman I (2007) Data fusion of high-resolution satellite imagery and liDAR data for automatic building extraction. ISPRS J Photogramm Remote Sens 62(1):43–63
26. Viola P, Jones MJ (2004) Robust real-time face detection. Int J Comput Vis 57(2):137–154
27. Voigt S, Kemper T, Riedlinger T, Kiefl R, Scholte K, Mehl H (2007) Satellite image analysis for disaster and crisis-management support. IEEE Trans Geosci Remote Sens 45(6):1520–1528
28. Wei Y, Zhao Z, Song J (2004) Urban building extraction from high-resolution satellite panchromatic image using clustering and edge detection. In: Geoscience and remote sensing symposium, 2004. IGARSS'04. Proceedings. 2004 IEEE international, vol 3, pp 2008–2010
29. Xian G, Homer C, Demitz J, Fry J, Hossain N, Wickham J (2011) Change of impervious surface area between 2001 and 2006 in the conterminous United States. Photogramm Eng Remote Sens 77(8):758–762
30. Yang Y, Newsam S (2013) Geographic image retrieval using local invariant features. IEEE Trans Geosci Remote Sens 51(2):818–832

Joseph Paul Cohen is a Ph.D Candidate in Computer Science at the University of Massachusetts Boston. His research interests include machine learning, computer vision, ad-hoc networking, and cyber security. Joseph received a U.S. National Science Foundation Graduate Fellowship in 2013 as well as COSPAR's Outstanding Paper Award for Young Scientists in the same year. Joseph is the creator of blucat; the cross-platform Bluetooth debugging tool. He is the founder of the Institute for Reproducible Research which produces Academic Torrents; a system designed to move large datasets and become the library of the future. He is also the creator of BlindTool; a mobile application providing a sense of vision to the blind by using an artificial neural network that speaks names of objects as they are identified. He has worked in industry for small startups, large corporations, government research labs, educational museums, as well as been involved in projects sponsored by NASA and the U.S Department of Energy.

Wei Ding received her Ph.D. degree in Computer Science from the University of Houston in 2008. She is an Associate Professor of Computer Science in the University of Massachusetts Boston. Her research interests include data mining, machine learning, artificial intelligence, computational semantics, and with applications to astronomy, geosciences, and environmental sciences. She has published more than 105 referred research papers, 1 book, and has 2 patents. She is an Associate Editor of Knowledge and Information Systems (KAIS) and an editorial board member of the Journal of Information System Education (JISE), the Journal of Big Data, and the Social Network Analysis and Mining Journal. She is the recipient of a Best Paper Award at the 2011 IEEE International Conference on Tools with Artificial Intelligence (ICTAI), a Best Paper Award at the 2010 IEEE International Conference on Cognitive Informatics (ICCI), a Best Poster Presentation award at the 2008 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPAITAL GIS), and a Best PhD Work Award between 2007 and 2010 from the University of Houston. Her research projects are currently sponsored by NASA and DOE. She is an IEEE senior member and an ACM senior member.

Caitlin Kuhlman is currently a PhD student in the Department of Computer Science at Worcester Polytechnic Institute. She is a member of the Database Systems Research Group. Her research interests include Machine Learning and Data Mining, as well as big data mangement and analytics.

Aijun Chen received his Ph.D. degree in cartography and geographic information system from Peking University,Beijing, China, in 2000. He joined the Center for Spatial Information Science and Systems, George Mason University, as a Postdoctoral Research Associate in May 2002 and was a Research Associate Professor since 2009. He has published more than 60 academic papers in journals, conferences proceedings and books. His research interests include standards-based remote sending data and information sharing, geospatial web services, geospatial Grid/Cloud computing, workflow-based geospatial modeling and simulation, and Cloud-based geospatial big data analysis.

Dr. Liping Di is a professor and the founding director of the Center for Spatial Information Science and Systems (CSISS) and a professor of the Department of Geography and Geoinformation Science, George Mason University, Fairfax, VA. He received his Ph.D. degree from the University of NebraskaLincoln in 1991. He has engaged in geoinformatics and remote sensing research for more than 30 years and has published over 350 publications. He has served as the principal investigator (PI) for more than 37 million research grants and as co-PI for more than 8 million research grants/contracts awarded by U.S. federal agencies and international organizations. His main research interests include remote sensing standards, web-based geospatial information and knowledge systems, and remote sensing applications. Dr. Di has actively participated in the activities of professional societies and international organizations, such as IEEE GRSS, ISPRS, CEOS, ISO TC 211, OGC, INCITS, and GEO. He currently chairs INCITS/L1, a U.S. national committee responsible for setting U.S. national standards on geographic information and representing U.S. at ISO TC 211.