Peer-Reviewed Technical Communication

Semisynthetic Versus Real-World Sonar Training Data for the Classification of Mine-Like Objects

Christopher Barngrover, Ryan Kastner, and Serge Belongie

***Abstract*—The detection of mine-like objects (MLOs) in sidescan sonar (SSS) imagery continues to be a challenging task. In practice, subject matter experts tediously analyze images searching for MLOs. In the literature, there are many attempts at automated target recognition (ATR) to detect the MLOs. This paper focuses on the classifiers that use computer vision and machine learning approaches. These techniques require large amounts of data, which is often prohibitive. For this reason, the use of synthetic and semisynthetic data sets for training and testing is commonplace. This paper shows how a simple semisynthetic data creation scheme can be used to pretest these data-hungry training algorithms to determine what features are of value. The paper provides real-world testing and training data sets in addition to the semisynthetic training and testing data sets. The paper considers the Haar-like and local binary pattern (LBP) features with boosting, showing improvements in performance with real classifiers over semisynthetic classifiers and improvements in performance as semisynthetic data set size increases.**

***Index Terms*—Haar-like feature, local binary pattern (LBP), mine-like object (MLO), object detection, sidescan sonar (SSS), synthetic.**

I. Introduction

PROCESSING of sidescan sonar (SSS) data has been a highly active field for decades, and specifically the task of detecting mine-like objects (MLOs) is very prominent. Traditionally, this was a manual process involving some post-processing to enhance the sonar image before a skilled operator tediously reviewed each image. Over the years, the process has shifted from this manual detection toward automatic target recognition (ATR).

Most of the early ATR research focuses on the shadows and highlights created through the obstruction of sound by any object protruding from the seafloor [1]–[3]. In much of the research, a model of the target is the basis for the feature, as with the matched filter approach introduced by Dobeck *et al.* [4]. This algorithm uses matched filters for various range regions and convolves the filter with the image to detect regions of interest. Some algorithms utilize machine-learning-type techniques for classification such as neural networks [5]–[7], k-nearest neighbor [6]–[8], and eigen-analysis [9].

The fusion of algorithms has also been considered in the research in various ways. One method processes high-frequency and low-frequency sonar images individually and then fuses the classification, either using the matched filter [8] or a wavelet decomposition [5] as the feature extraction method. Other fusion efforts combine coregistered sensor data as input to a classifier [10] or combine the output of known classifiers to determine a fused confidence [11]. One effort uses simple shadow attribute features with basic summing fusion of algorithms as well as the nonlinear Volterra feature with log-likelihood ratio test (LLRT)-based fusion rules [12].

The recent advances in autonomous underwater vehicles (AUVs) and sonar capabilities have caused an influx of more advanced computer vision features and machine learning techniques. Some researchers have moved away from the model-based approach and have started using local descriptors without target knowledge, such as the Haar-like feature [13]. More state-of-the-art machine learning approaches are being considered, such as boosting [13] and support vector machines (SVMs) [14].

The machine learning techniques of boosting and SVMs require a large training data set to learn the optimized combination of features. Creating a large data set can be difficult, time consuming, and expensive, which is why it is common to generate synthetic or semisynthetic data sets for training and testing [13], [15]–[17].

The goal of this paper is to consider the detection capabilities when using such synthetic data sets for training as compared to training on real-world examples. For this experiment, we use a very popular and simple training algorithm, which uses AdaBoost to select features and creates an optimized cascade of features for classifying windows as MLO or non-MLO [18]. This training algorithm requires a large amount of positive and negative training examples to allow the machine learning element to properly select the best features for the cascade. We run two versions of this experiment, one with the Haar-like feature and another with the local binary pattern (LBP) feature.

The three main contributions of this paper are as follows:

Manuscript received March 08, 2013; revised June 20, 2013 and October 03, 2013; accepted November 12, 2013. Date of publication January 17, 2014; date of current version January 09, 2015.

Associate Editor: A. Trucco.

C. Barngrover is with the Department of Computer Science, University of California San Diego, La Jolla, CA 92093 USA and also with the Space and Naval Warfare (SPAWAR) Systems Center Pacific, San Diego, CA 92110 USA (e-mail: cbarngrover@ucsd.edu; chris.barngrover@navy.mil).

R. Kastner and S. Belongie are with the Department of Computer Science, University of California San Diego, La Jolla, CA 92093 USA.

Digital Object Identifier 10.1109/JOE.2013.2291634

Fig. 1. Examples of the inert mines that are placed on the seafloor for various mine-related exercises.

TABLE I
DATA SET METRICS

	Training				Testing	
	Real	Synth	Synth Large	Synth Source	Real	Synth
Images	975	1,000	5,000	100	920	1,000
Type 1 Observations	426	480	2,531	50	401	486
Type 2 Observations	549	520	2,469	50	519	514
MLO	0	0	0	0	348	76
Positives	975	1,000	5,000	100	1,268	1,076

- evaluating the use of semisynthetic data sets for classifier generation in lieu of real-world data sets, specifically for classifying MLOs in SSS;
- quantifying the relative improvement when training on real-world data instead of semisynthetic;
- providing the data sets used in this research to the academic community. The real-world training and testing data sets are of particular significance.[1]

This paper is organized as follows. Section II describes each of the training and testing sets and the mechanisms for producing or collecting the data. Then, we explain in detail the training algorithm and the two features used for this experiment in Section III. Next, Section IV presents and explains the results from the experiments. Section V discusses the implications of the experimental results and describes the caveats on the findings. We finish with a conclusion in Section VI.

II. TRAINING AND TESTING DATA SETS

There are some common traits of all of the data in this paper, which are kept static to reduce the number of variables in the trials. All of the data sets use SSS data captured via REMUS AUVs equipped with two 900-kHz Marine Sonic sensors. The sonar images produced by this sensor are 1024 × 1000 pixels, as are all the images used in this paper. The vehicles are run in the same locations at a goal altitude of 4 m to collect 30-m slant-range data. The data collected for all data sets are limited to this scope; however, the framework we describe could be used to consider data collected under broader parameters. The actual mines included in the data are the type 1 and type 2 mines shown in Fig. 1. Note that there are other MLOs present, which are not necessarily designated mine shapes. Table I shows the metrics of the six different data sets for training and testing.

The Space and Naval Warfare Systems Center Pacific (SSC–PAC), San Diego, CA, USA, collected all of the data. There are ten different type 1 mines and seven different type 2 mines in the various fields used for collection. We present the differences in the creation of the semisynthetic data set versus the real-world data set in Sections II-A–II-C.

[1]http://kastner.ucsd.edu/datasets/barngrover/

A. Semisynthetic Training Data Sets

The most prominent problem with research on automatic target recognition in SSS is the lack of labeled data sets. Therefore, to train using machine learning algorithms requiring thousands of examples, it is common to create a synthetic or semisynthetic data set. In this section, we describe our technique, which attempts to maximize variability with a limited number of positive mine examples.

First, we collect positive examples of mine images from real-world SSS data. Our pool of positive examples includes 50 images containing type 1 mines and 50 images containing type 2 mines. These 100 images make up the data set we refer to in Table I as Synth Source, which is used as a baseline for comparison to our semisynthetic data sets. Then, we label the mine highlight and shadow individually by choosing points to represent a polygon around each. A trained operator, who is also an author on this paper, labels the positive targets manually. Next, we collect negative examples from real-world SSS data. These are images from the same environment that do not contain a mine. We have 1000 such negative images for semisynthetic data set creation.

The semisynthetic data set creation algorithm processes each negative image as a destination image by adding a single mine as well as three patches of background. Fig. 2 graphically shows the process. For each negative sonar image, one of the 100 positive mine images is selected at random and the polygons of the highlight and shadow are used to copy only the relevant pixels. The side of the negative sonar image on which to place the mine pixels is chosen at random. If the mine is placed on the opposite side of the negative image as compared to the side of positive mine image from which it came, then it is mirrored on the across-track axis to maintain the shadow's proper orientation. Finally, the along-track location for the mine is chosen at random, with the across-track range preserved because of its correlation to the size of the shadow. After all of the randomized parameters are set, the mine can be placed by copying the pixels within the labeled polygons of the positive mine image to the negative image.

In addition, we attempt to account for the potential artifacts introduced through this process by adding randomized nonmine pixels from the positive mine image to the negative, or destination, image. The logic is that any artifacts from the polygon copying process, such as mismatched average pixel intensity or polygon placement on a surface return, are present in these negative portions of our final image as well as in the positive regions. The learning, therefore, does not occur based on any copying artifacts alone. There are potential drawbacks to this which we leave to the discussion in Section V. We extract pixels from random locations in the positive sonar image and place them in the same location of the destination image. The same polygon shapes are used for this background pixel extraction as were used for the positive target pixel extraction to better mimic any artifacts created from the pixel extraction and placement process. We add three of the negative regions to have more negative than positive examples of any artifacts. The three small

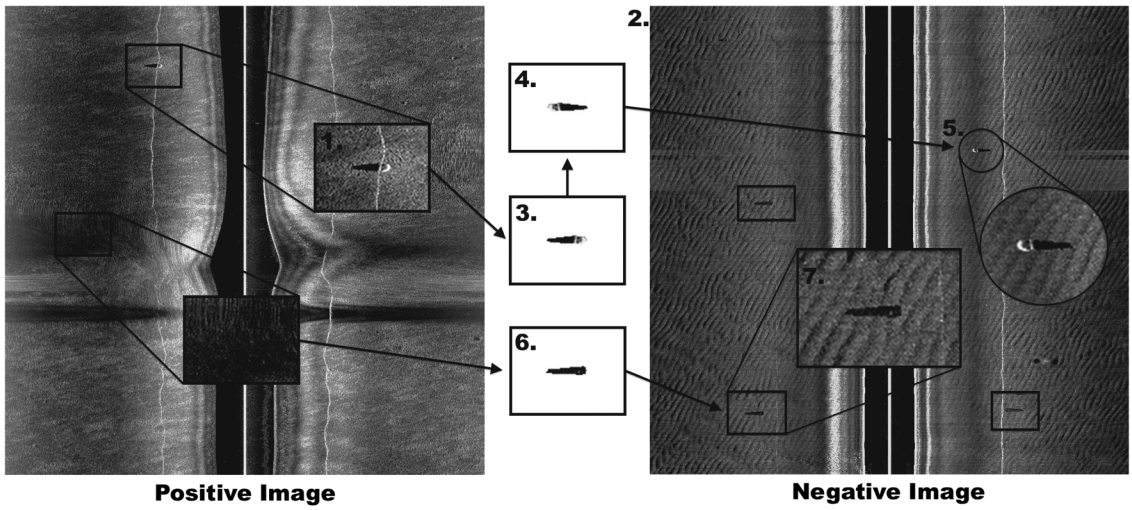


Fig. 2. The synthetic image creation algorithm. 1. Label the 100 positive examples. 2. Collect the 1000 negative examples. 3. For each negative image, choose a positive example and capture only the positive pixels. 4. Choose the side of the negative image and a y-location for placement. 5. Copy over the positive pixels. 6. Choose a random location in the positive image and capture negative pixels. 7. Copy the negative pixels to a random location.

rectangles in the "negative image" of Fig. 2 are the randomly placed negative pixels.

There are some flaws to this paradigm for creating a semisynthetic data set. For one, there is a small number of example positives that must represent a larger sample. The only change for a given mine is the side of the image and the y-location, which is not a lot of variety. The background in close proximity to the mine is another change between semisynthetic images. However, one flaw is the difference in average intensity from image to image. For example, if the positive sonar image is darker on average, then the positive mine stands out even more. In addition, the negative pixels also stand out as darker than the background. The example in Fig. 2 shows this intensity problem because of the darker background locations of the positive image, which are copied over to the lighter background of the negative image.

Generating one semisynthetic image takes approximately 0.5 s, allowing for very large data set creation in a reasonable amount of time. All of the negative images are processed to create a semisynthetic training data set of 1000 images, which is referred to as Synth in Table I. The randomization creates a data set of 480 type 1 mines and 520 type 2 mines. One of the benefits of semisynthetic data sets is the ability to create large numbers of training images. Therefore, we also process each of the negative images five times to create a semisynthetic data set of 5000 images, which is referred to as Synth Large in Table I. This larger data set contains 2531 type 1 mines and 2469 type 2 mines. Because of the multiple tiers of randomization in the process, we are able to create a much larger data set of positive training examples that contain real-world mines in real-world backgrounds.

B. Real-World Training Data Set

The ideal training set for creating a classifier for a particular problem is a real-world training set. In some cases, this is not a difficult endeavor, but in many cases, such as with SSS, the data collection process is expensive and time consuming. The collection of such a large training data set was a huge undertaking in collaboration with SSC–PAC over the span of a year. This set of images is different from the 100 images used to create the semisynthetic data set.

The data collection effort involves various missions in different inert mine fields of San Diego Bay. For a certain mission, the REMUS AUV was programmed to run a reacquire-and-investigate (R&I)-type mission, which is a starlike pattern, over a specific geodetic location. Normally, the R&I mission is designed to collect 5–10-m range data, but in this case, it was altered to collect 30-m range data. This type of mission allows us to maximize the number of looks at each mine.

Each mission produces multiple hundreds of SSS images, which must be processed to find the specific mine. Then, the mine is labeled with a polygon for the highlight and a polygon for the shadow. This entire labeling process was performed by a trained operator, who is also an author on this paper, over the course of many months.

The result is a collection of labeled positive images, of which half are designated training and half are designated testing. The real-world training data set used for this experiment contains 975 SSS images, including 426 type 1 and 549 type 2.

C. Testing Data Sets

The real-world testing data set is the primary testing data set for this experiment. It comes from the same collection as described in Section II-B, but it is a completely separate set from the training. It is also different from the 100 images used to create the semisynthetic data set. The real-world testing data set contains 920 SSS images, including 401 type 1 and 519 type 2 as well as 348 other MLOs present in the field.

In addition to the real-world test, we have a semisynthetic testing data set for evaluating another interesting aspect of this experiment. The same algorithm is used to create this testing set as was used to create the two semisynthetic training data sets. This means that the same 100 positive images and 1000 negative background images were used. The only difference is the randomization inherent in the algorithm. The semisynthetic testing data set contains 1000 SSS images, including 486 type 1 and 514 type 2 as well as 76 other MLOs present in the base negative images. The synthetic source and the two semisynthetic

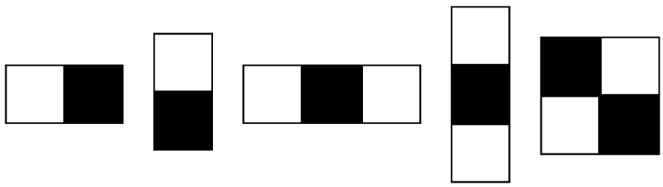


Fig. 3. *Basic* group of Haar-like features in the OpenCV library includes these five variations, which we use in this experiment. The sum of the pixels in the white rectangle is subtracted from the sum of the pixels in the black rectangle to calculate the Haar-like feature value.

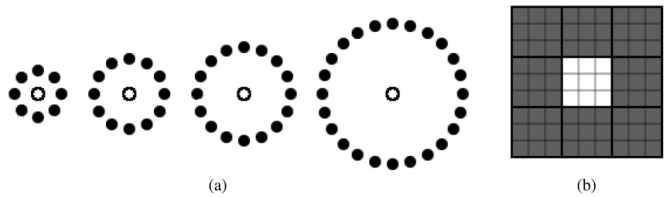


Fig. 4. Visualizations of the two extensions of LBP. (a) Circular (8, 1), (12, 1.5), (16, 2), and (24, 3) neighborhoods for LBP calculation. The white dot is the focus pixel and the black dots are neighborhood pattern pixels. (b) A 3 × 3 multiscale block LBP. The white square of nine pixels is the focus block and the gray squares are the neighborhood pattern blocks.

classifiers are tested on the real and semisynthetic testing data sets to analyze the capability of the semisynthetic training.

III. Boosted Cascade Feature Selection

To quantify the benefits of a semisynthetic training set compared to a real-world training set, we must utilize a training algorithm that traditionally requires a large amount of labeled data. We have chosen to use the technique proposed by Viola and Jones [18], which utilizes AdaBoost as a feature selection mechanism to form a cascade of weak learners in stages. Specifically, we have implemented a version of this algorithm under the OpenCV library.

The first step of the algorithm is creating a pool of features for selection purposes. Like the Viola–Jones paper, we use the Haar-like feature in the pool of features for one of our experiment algorithms. The Haar-like feature is based on the Haar wavelet, which does not use intensity directly like the original Haar basis functions [19]. Fig. 3 shows visualizations of the five types of Haar-like features included in our feature pool.

This feature captures the difference in pixel intensity between designated rectangles in a given location of the considered window. The feature extraction subtracts the sum of the pixels in the white rectangles from the sum of the pixels in the black rectangles, thereby representing the gradient with a single float value. The actual feature calculation is based on a combination of the size of the rectangles and the location of the upper left corner of the feature in the window. The pool of features includes every combination of rectangle and every location of feature possible in the designated window. For the fixed window in this paper, the result is a pool containing 2 543 145 Haar-like features.

The LBP feature is another local-type descriptor common to computer vision, which we use for our other experiment algorithm. The LBP feature considers a neighborhood around a center focus pixel and thresholds each pixel intensity compared to the focus pixel, producing a binary representation [20]. Originally, the neighborhood was a 3×3 region, but it has been extended to include multiple sizes [21]. The notation (P, R) represents a given LBP feature, where P is the number of sampling points equally spaced on the circle and R is the radius of the circle. Fig. 4(a) shows visualizations of a few possible circular neighborhoods.

This feature captures the texture for a region of the considered window. The feature extraction thresholds each of the pixel locations, represented by the black dots, against the focus pixel, represented by the white dot. Then, the resulting binary numbers are concatenated into a string by traveling the circumference of the circle. Finally, the binary pattern is converted to its decimal form, thereby representing the texture with a single integer.

The LBP feature was extended again to use multiscale blocks instead of individual pixels to obtain a more macroscopic representation of a local region [22]. In this version, the comparison between pixels and the focus pixel is replaced by average pixel intensity of blocks compared with average pixel intensity of a focus block, as visualized in Fig. 4(b). We use this multiscale block LBP feature of the OpenCV library for our second algorithm. The pool of features includes every combination of focus block location and size in the prescribed window. For the fixed window in this paper, the result is a pool containing 138 645 LBP features.

We use a fixed window due to the nature of the sonar images, specifically that a target of a certain size will produce a shadow of varying size based on the altitude of the vehicle and the range to target. This means that differences in the sizes of targets are primarily in the across-track plane and that the shadow part of the target is the only part changing. We are able to choose a fixed window because of the limited scope of the data, as described in Section II. An alternative to the fixed window is to use scaling, which is complicated since only the shadow scales and primarily in the across-track plane.

The window size is chosen based on the statistics of the mines labeled in the entire real-world training data set collected by SSC–PAC. We could have used the largest dimensions from the data or even the geometric maximum dimensions based on the known size of the two mines and the altitude of the vehicle for this data set. This would produce a larger window, which in turn means more features to consider. Since we are focusing on the transition area between mine highlight and the mine shadow, due to the unknown end of the shadow based on varying length, we do not need a window that includes all possible lengths of target. Therefore, we have chosen to use a standard deviation from the mean to make sure to cover the width of the highlight and a large portion of all shadows. The formulas used to calculate the width and height of the window are, respectively

$$w = \bar{w} + 3\sigma_w + 2m \tag{1}$$

$$h = \bar{h} + 3\sigma_h + 2m. \tag{2}$$

The average width $\bar{w}$ of the mines, including highlight and shadow, across the entire data library is 42 pixels. The standard deviation of the width σ_w is 11 pixels. We also include a margin m of two pixels on each side of the mine. The average height $\bar{h}$ of the mines is 19 pixels, while the standard deviation of the height σ_h is two pixels. The same margin m is used again for

Fig. 5. These positive MLO examples have varying shadow lengths, but an approximately consistent highlight location within the window.

the height. The outcome of the statistics analysis is a 79 × 29 pixel window.

The preprocessing of the training data set before AdaBoost optimization involves the creation of positive and negative data files based on the labeled location of the MLOs. Since both features include the location within the window, the positive examples must be aligned such that the highlight portion of the MLO is in the same location in the window. We alter all positive examples to appear as if on the left-hand side of the SSS image. We also right justify and vertically center the MLO in the window with at least a two-pixel buffer. Fig. 5 shows three examples of positive MLOs aligned in approximately the same location of the window.

The negative, or background, image that is provided during preprocessing includes the half of the image that does not contain the MLO. The optimization algorithm utilizes various windows from the background image for training purposes.

In each stage of training, the AdaBoost feature selection process iteratively chooses the next feature, or weak classifier in the jargon of boosting, that best separates the positives from the negatives. The boosting continues to select additional features until the stage is able to achieve certain thresholds for hit rate and false alarm rate. The hit rate is the number of targets correctly classified as positive out of the total number of positive windows considered. The false alarm rate is the number of targets incorrectly classified as positive out of the total number of negative windows considered. In our experimentation, the minimum hit rate is 0.995 and the maximum false alarm rate is 0.5. The boosting terminates when the designated number of stages is met or when the entire classifier passes the acceptance ratio.

The optimized classifier consists of a series of stages, each containing features of interest. When the classifier is used to process an image via the OpenCV library, a sliding window approach considers all possible windows of the prescribed fixed window size with an across-track step of one pixel and an along-track step of two pixels. Each stage of the classifier must be passed to label a window as positive. Due to the fine granularity of the sliding window, there are often multiple positive windows in a cluster. There is a minimum neighbors threshold that requires a specific number of windows in a positive cluster for an actual positive labeling. For instance, if the minimum neighbors is set to five, then a cluster of five or more positive windows results in one positive label, while four or fewer positive windows results in a negative label.

IV. Experimental Results

We train on four data sets using two different algorithms for this experiment. The first classifier we refer to as *real* since it was trained on the real-world SSS imagery data set. The second classifier we refer to as *synth* since it was trained on a semisynthetic data set. The *synth_large* classifier was trained on a semisynthetic data set five times larger than the *real* and *synth* classifiers. Finally, the *synth_source* data set contains the 100 positive examples, which are the source for the semisynthetic data set creation algorithm. The two algorithms described in Section III use the boosted cascade with the Haar-like feature and LBP feature, respectively.

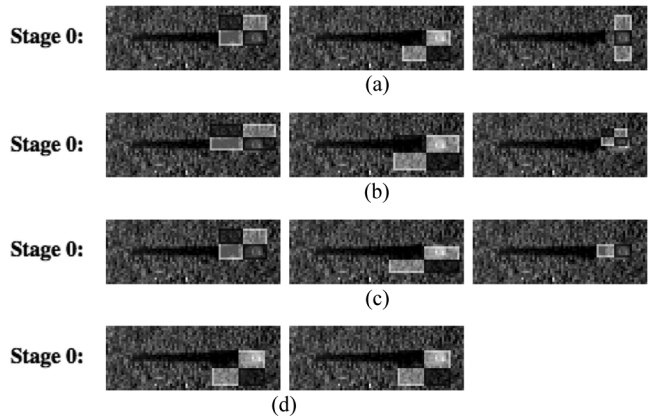


Fig. 6. Visualization of the first stage for each of the four Haar-like cascade classifiers. Each window shows the Haar-like feature overlaid on an example positive MLO window. The white and black rectangles represent the regions that are compared to calculate the feature value: (a) *real*; (b) *synth*; (c) *synth_large*; and (d) *synth_source*.

The first algorithm we train on our four data sets uses a pool of Haar-like features. The *real* Haar classifier contains seven stages and a total of 34 Haar-like features. The *synth* Haar classifier has eight stages with a total of 49 Haar-like features and the *synth_large* Haar classifier also contains eight stages, including a total of 73 Haar-like features. The *synth_source* Haar classifier has only four stages with a total of seven Haar-like features. Visualizations of the first stage for each of the four Haar classifiers are shown in Fig. 6. Each window in the figure shows one Haar-like feature from the pool of features presented in Fig. 3 overlaid on an example mine window. The choice of white or black rectangles does not necessarily correlate to the highlight or shadow in the window, since the feature can be negative or positive to represent the difference in intensity.

The second algorithm in this experiment uses the LBP feature in the pool for boosting. The *real* LBP classifier has 12 stages with a total of 97 LBP features. There are 13 stages for both the *synth* and *synth_large* LBP classifiers with 136 and 188 LBP features per classifier, respectively. The *synth_source* LBP classifier has 12 stages with only 44 LBP features. Visualizations of the first stage for each of these four classifiers are shown in Fig. 7. Like the Haar classifiers, each image is a selected LBP feature overlaid on the same example mine window. The white rectangle represents the focus block and the darker rectangles on the perimeter are the comparison blocks.

The efficiency of the algorithms is an important metric to understand for any image processing experiments. The training and processing for these experiments was performed on a 1.7-GHz Intel Core i5 processor with 4-GB 1333-MHz DDR3 RAM. Table II shows the time taken to train in hours and the processing time of one full sonar image in seconds for both Haar and LBP classifiers. Obviously the type of image will

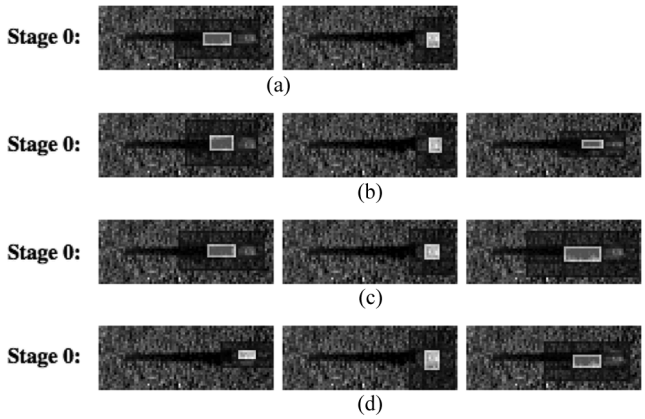


Fig. 7. Visualization of the first stage for each of the four LBP cascade classifiers. Each window shows the LBP feature overlaid on an example positive MLO window. The white rectangles represent the location of the focus block and the black rectangles represent the eight perimeter comparison blocks: (a) *real*; (b) *synth*; (c) *synth_large*; and (d) *synth_source*.

TABLE II
CLASSIFIER EFFICIENCIES

	Haar		LBP	
	Training (hours)	Processing (seconds)	Training (hours)	Processing (seconds)
Real	17.55	0.0473	01.60	0.0627
Synth	14.11	0.0499	03.18	0.0801
Synth Large	81.88	0.0508	22.74	0.0812
Synth Source	00.15	0.0415	00.08	0.0682

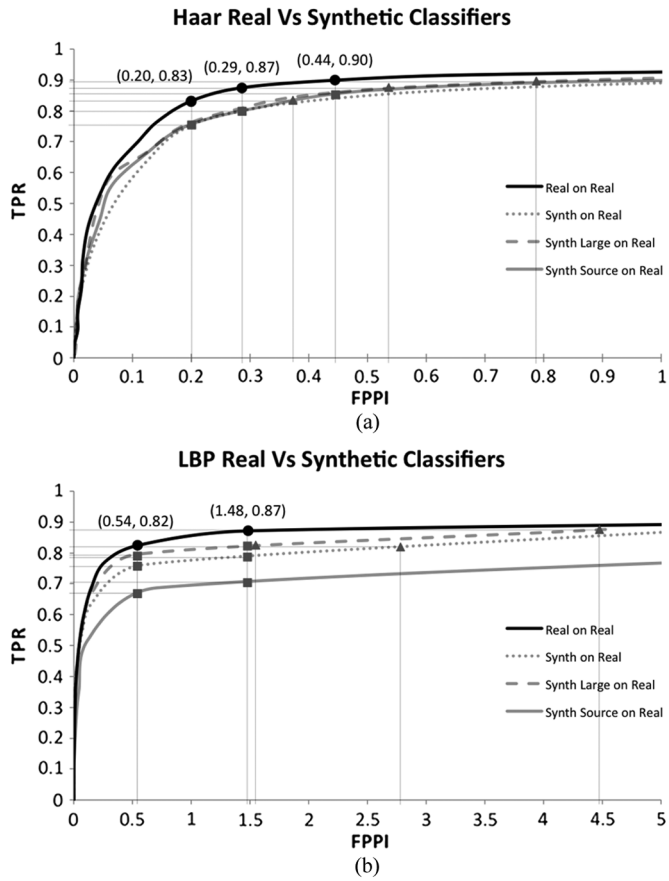


Fig. 8. ROC curve for the four classifiers when processing the real testing data set. The vertical axis is TPR and the horizontal axis is FPPI: (a) Haar classifiers; and (b) LBP classifiers.

determine how long it takes to process based on how many stages are passed for various windows of consideration.

To understand the performance of the various classifiers, we provide in Fig. 8 the receiver operating characteristic (ROC) curve for each of the eight classifiers when applied to our real testing data set. Fig. 8(a) shows the curves for the Haar-like classifiers while Fig. 8(b) shows the curves for the LBP classifiers. The vertical is the true positive rate (TPR) as with most ROC curves, but on the horizontal, we consider false positives per image (FPPI), which is more telling than the false positive rate (FPR) for a sliding window classifier. An FPPI of one means that, on average, we have one false positive per processed sonar image, which is 1024×1000 pixels in these data sets. The minimum neighbors threshold for clustering positive windows is the variable changed from zero to 20 to create the points on the curve. This means that the actual points on the curves will not have round TPR or FPPI values, but rather have the particular TPR and FPPI for the certain minimum neighbors threshold.

Both figures have example points, shown as black circles, for comparison to the other curves. The triangle points on both figures show the change in FPPI necessary to achieve the same example TPR on other synthetic classifier curves. The square points show the reduction in TPR necessary to maintain the same FPPI on other synthetic classifier curves.

The secondary experiment considered in this paper is how the synthetic classifiers perform on a semisynthetic testing data set compared to on the real testing data set. Fig. 9 shows the ROC curves for both Haar in Fig. 9(a) and LBP in Fig. 9(b). The lighter curves show the performance of the semisynthetic classifiers on the real testing data set, which are the same curves as shown in Fig. 8, while the darker curves show the performance on the semisynthetic testing data set.

V. DISCUSSION

Let us start with the visualizations of the first stage for each of the four Haar classifier cascades. As expected, the training selects features that emphasize the contrast between highlight and shadow, highlight and background, as well as shadow and background. Many of the features chosen in these cascades are common between the classifiers. Specifically, the first two features of stage 0 are the same feature type in nearly the same location with nearly the same size parameters. The exception is the first feature of stage 0 for the *synth_source* cascade, which is the same as the second feature for all four cascades.

The first stage for each of the four LBP classifier cascades also emphasizes the highlights and shadows compared to the neighborhoods around them. The feature choices are not as consistent as in the first stage of the Haar cascades, but the second feature in all four is very consistent with a focus block right on the highlight of the example target. This same type of LBP feature is common throughout the stages.

The similarities of the early stages bolster the claim that the semisynthetic-based classifiers can be used as a proof of concept for the boosted cascade of both Haar-like and LBP features. While the early stages do drastically reduce the number of windows considered from a given image, the later stages do the fine-tuned analysis to determine the MLO classification.

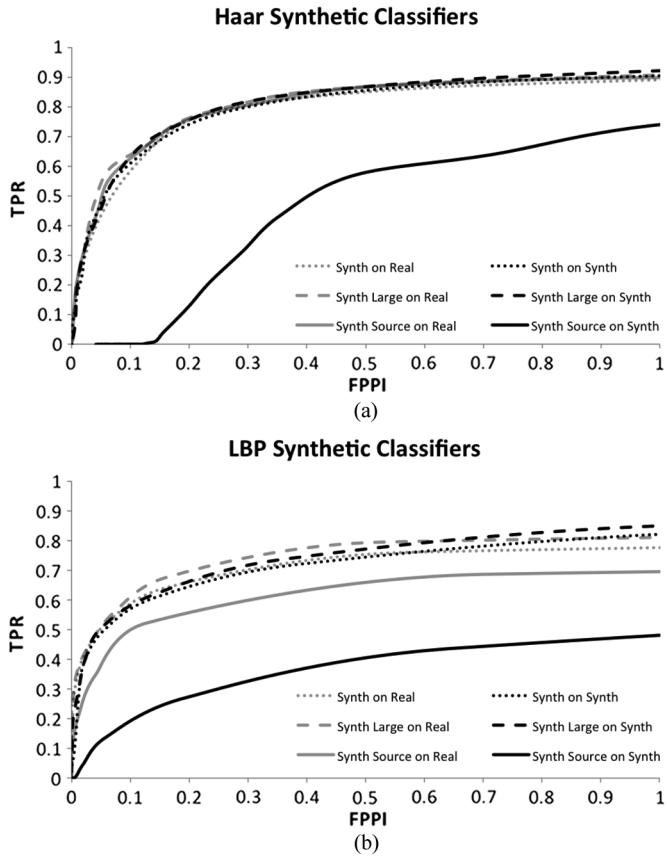


Fig. 9. ROC curve for the *synth*, *synth_large*, and *synth_source* classifiers applied to the real and semisynthetic testing data sets. The vertical axis is TPR and the horizontal axis is FPPI: (a) Haar classifiers; and (b) LBP classifiers.

It makes sense that the obvious differences between an MLO and a non-MLO captured in the early stages would be common among multiple classifiers, while the later stages would have more prominent differences.

The number of features needed to complete the classifier training is interesting. In general, more similarities between the negative and positive examples or more variation within the examples causes a larger feature count in the classifier. For these classifiers, there is an increase in feature count with the size of the training data set, not because of the increased data set size directly, but more likely because the additional data provide more variation of positive examples. Also, there is an increase from the real to synthetic classifiers, which implies that the negative and positive examples are more similar and harder to classify in the synthetic training set. This could be caused by some element of the synthetic creation process, such as a high variation within the negative background images used or complications with copying negative pixels to avoid artifact bias.

Analyzing the stages and features of each of the classifiers is one way to grasp the benefit of using semisynthetic versus real training data sets. Another more telling avenue is to consider the performance on a real testing data set.

The ROC curves in Fig. 8 show a couple of important facts. First, as expected, the *real* classifier outperforms the *synth* and *synth_large* classifiers for both Haar and LBP classifiers. This is expected because the semisynthetic data merely mimics the real data. It is important to note that the improvement with the real classifiers is only between 4% and 6% over the synthetic classifiers, suggesting the classifiers trained on the semisynthetic data sets provide a reasonable estimation of the resulting capability of the real classifier.

Furthermore, the *synth_large* classifier slightly outperforms the *synth* classifier. This means that a nominal improvement can be gained from the increase in semisynthetic training images. However, the *synth_large* classifier takes substantially longer to train, as shown in Table II, when compared to the *synth* classifier for both Haar and LBP. The additional time to train only results in a minor improvement of TPR, just over 3%, for the LBP and an even smaller improvement, less than 2%, for Haar. This is not a large return in accuracy based on the time cost, though the training is a onetime step that might be warranted for the improved capability in some circumstances.

Fig. 9 shows that the performance of the semisynthetic classifiers on the semisynthetic testing data set is nearly the same as the performance on the real testing data set for both Haar and LBP. For the Haar classifiers there is only about 1% difference in TPR between the real and synthetic test sets for *synth* and *synth_large*. There is a bit more separation in performance using the LBP feature, with about 3% difference in TPR. The similarity of performance for the synthetic classifiers on both testing data sets is important because it means that training and testing on semisynthetic data can appropriately estimate how these classifiers perform on a real testing data set.

The exception is the *synth_source* classifier, which performs much worse on the semisynthetic testing data set. This is a result of a specific bottom type containing sand ripples, which is present in the synthetic data but is less prevalent in the real testing data. This sand ripple bottom type is not in the *synth_source* training set, causing a large number of false positives when tested on the synthetic data compared to the real data. The *synth* classifier does not have the same problem when tested on the synthetic data because it is trained on images including the sand ripple bottom.

To verify that the sand ripples are causing the large number of false positives, we test on a subset of the synthetic testing set with some images removed. We remove 16 images with prominent sand ripple regions from the synthetic testing set and process this new testing set. There is a strong improvement just from removing a small number of sand ripple bottom images.

This exception emphasizes the need to have large comprehensive data sets for training when using such machine learning algorithms as boosting. The small number of training images is part of the problem, but this just highlights the poor performance resulting from a disproportionate variety in the training data compared to the testing data.

The major takeaway from these experimental results is that semisynthetic training and testing in the absence of real data can provide expectations for the performance when trained and tested on real data. Furthermore, it can lead toward decisions on the viability of one feature over another. For instance, we could correctly estimate from the semisynthetic classifiers on semisynthetic testing data that the Haar classifier is better for this data set than the LBP without needing the real data.

There are a few potential shortcomings of this research, and one example is the data labeling process. We use one trained operator to label all of the positive MLOs for synthetic data set creation as well as for the real training and testing data sets. This is a manageable approach for a small experiment, but an intraoperator and interoperator study would be an interesting experiment in the future.

The semisynthetic data set creation process has some shortcomings. There are only 50 unique mine examples of each mine type used to create the two different sizes of synthetic training data sets. This is a result of available data at the time, not a chosen parameter. For a given mine type, there is a limited amount of variation between observations when the data are captured at a constant altitude and frequency. Our 50 observations per mine type does not cover all of the variations, but given the size of the mine at this range, it covers a large amount of variation. We acknowledge that it would be worthwhile to create synthetic data sets with the larger number of unique positives now available.

Another problem with the semisynthetic data set creation process is that the copying of background pixels to the synthetic target image may create MLOs based on differences in average pixel intensity between images. One could consider normalizing the images so that they are similar in average pixel intensity to avoid this problem. Also, it would be interesting to remove from the creation process the step of copying the background pixels and then compare the results. This may improve the classifier, but it will be hard to quantify if this is a result of a copying artifact in the synthetic data versus removal of problem negative polygons.

There are limitations to this experiment based on the limited training algorithms considered and the singular semisynthetic data creation scheme. It would be interesting to consider more training algorithms and other synthetic and semisynthetic data in future experiments to more broadly quantify the use of automated data in the absence of real data.

VI. Conclusion

The experiments considered in this paper show the benefit of using a semisynthetic data set for the training and testing of data hungry machine learning algorithms, such as the boosted cascade using either the Haar-like feature or the LBP feature, when a real data set is not available. The experimental results show that semisynthetic classifiers perform within 1% and 3% for Haar and LBP, respectively, when tested on semisynthetic data versus real data. The results also show that the semisynthetic classifiers perform only 4%–6% worse than the real classifiers when tested on real data. Combined, the results from the two experiments mean that only using semisynthetic data for training and testing when real data are not available can provide insight into the capability of an algorithm.

Despite the limited scope of the experimentation in this paper, the results provide a foundation for comparison between synthetic and real training in terms of SSS imagery and mine-like object classification. In the context of this paper, it is clear that there is a benefit in using semisynthetic data sets for training in the absence of available real-world data.

References

[1] S. Reed, Y. Petillot, and J. Bell, "An automatic approach to the detection and extraction of mine features in sidescan sonar," *IEEE J. Ocean. Eng.*, vol. 28, no. 1, pp. 90–105, Jan. 2003.

[2] M. Mignotte, C. Collet, P. Perez, and P. Bouthemy, "Sonar image segmentation using an unsupervised hierarchical MRF model," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1216–1231, Jul. 1998.

[3] X. Wang, H. Wang, X. Ye, L. Zhao, and K. Wang, "A novel segmentation algorithm for side-scan sonar imagery with multi-object," in *Proc. Int. Conf. Robot. Biomimetics*, Dec. 2007, pp. 2110–2114.

[4] G. J. Dobeck, J. C. Hyland, and L. Smedley, "Automated detection and classification of sea mines in sonar imagery," *Proc. SPIE—Int. Soc. Opt. Eng.*, vol. 3079, pp. 90–110, 1997.

[5] D. Yao, M. R. Azimi-Sadjadi, A. A. Jamshidi, and G. J. Dobeck, "A study of effects of sonar bandwidth for underwater target classification," *IEEE J. Ocean. Eng.*, vol. 27, no. 3, pp. 619–627, Jul. 2002.

[6] D. Li, M. R. Azimi-Sadjadi, and M. Robinson, "Comparison of different classification algorithms for underwater target discrimination," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 189–194, Jan. 2004.

[7] G. J. Dobeck, "The k-nearest neighbor attractor-based neural network and the optimal linear discriminatory filter classifier," in *Proc. OCEANS Conf.*, 2006, DOI: 10.1109/OCEANS.2006.307017.

[8] G. J. Dobeck, "Fusing sonar images for mine detection and classification," *Proc. SPIE—Int. Soc. Opt. Eng.*, vol. 3710, pp. 602–614, 1999.

[9] P. Saisan and S. Kadambe, "Shape normalized subspace analysis for underwater mine detection," in *Proc. 15th IEEE Int. Conf. Image Process.*, Oct. 2008, pp. 1892–1895.

[10] N. Klausner, M. Azimi-Sadjadi, and J. Tucker, "Underwater target detection from multi-platform sonar imagery using multi-channel coherence analysis," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, Oct. 2009, pp. 2728–2733.

[11] C. Ciany, W. Zurawski, G. Dobeck, and D. Weilert, "Real-time performance of fusion algorithms for computer aided detection and classification of bottom mines in the littoral environment," in *Proc. OCEANS Conf.*, Sep. 2003, vol. 2, pp. 1119–1125.

[12] T. Aridgides and M. Fernández, "Image-based ATR utilizing adaptive clutter filter detection, LLRT classification, and Volterra fusion with application to side-looking sonar," *Proc. SPIE—Int. Soc. Opt. Eng.*, vol. 7664, 2010, DOI:10.1117/12.846596.

[13] J. Sawas, Y. Petillot, and Y. Pailhas, "Cascade of boosted classifiers for rapid detection of underwater objects," in *Proc. Eur. Conf. Underwater Acoust.*, 2010, vol. 10.

[14] P. Hollesen, W. A. Connors, and T. Trappenberg, "Comparison of learned versus engineered features for classification of mine like objects from raw sonar images," in *Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2011, vol. 6657, pp. 174–185.

[15] M. Gendron, M. Lohrenz, and J. Dubberley, "Automated change detection using synthetic aperture sonar imagery," in *Proc. MTS/IEEE OCEANS Biloxi—Mar. Technol. Our Future: Global and Local Challenges*, Oct. 2009, pp. 1–4.

[16] E. Coiras, P.-Y. Mignotte, Y. Petillot, J. Bell, and K. Lebart, "Supervised target detection and classification by training on augmented reality data," *IET Radar Sonar Navig.*, vol. 1, no. 1, pp. 83–90, Feb. 2007.

[17] M. An, J. Tory Cobb, B. Shenefelt, and R. Tolimieri, "Advances in group filter applications to sea mine detection," in *Proc. OCEANS Conf.*, Sep. 2006, DOI: 10.1109/OCEANS.2006.306951.

[18] P. Viola and M. Jones, "Robust real-time object detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2001.

[19] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proc. 6th Int. Conf. Comput. Vis.*, Jan. 1998, pp. 555–562.

[20] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognit.*, vol. 29, no. 1, pp. 51–59, 1996.

[21] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

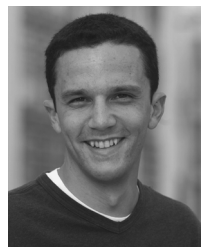
[22] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li, "Learning multi-scale block local binary patterns for face recognition," in *Advances in Biometrics*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2007, vol. 4642, pp. 828–837.

Christopher Barngrover received the B.S. degree in computer science and the B.S. degree in mathematics from Purdue University, West Lafayette, IN, USA, in 2005 and the M.S. degree in computer science from the University of California San Diego, La Jolla, CA, USA, in 2010, where he is currently working toward the Ph.D. degree with a target completion in early 2014.

He is currently a Software Engineer for the Space and Naval Warfare (SPAWAR) Systems Center Pacific, located on Point Loma, San Diego, CA, USA. He has spent his time following his B.S. degree working in the Unmanned Systems Branch of SPAWAR Systems Center, working on control software for unmanned systems and force protection sensors.

Ryan Kastner received the B.S. degree in electrical engineering and the B.S. degree in computer engineering, in 1999, and the M.S. degree in engineering, in 2000, from Northwestern University, Evanston, IL, USA and the Ph.D. degree in computer science from the University of Los Angeles (UCLA), Los Angeles, CA, USA, in 2002.

He is currently a Professor in the Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA, USA. He is the Co-Director of the Wireless Embedded Systems Master of Advanced Studies Program. He also codirects the Engineers for Exploration Program. His current research interests reside in the realm of embedded system design, in particular, the use of reconfigurable computing devices for digital signal processing.



Serge Belongie was born in Sacramento, CA, USA. He received the B.S. degree (with honors) in electrical engineering from the California Institute of Technology, Pasadena, CA, USA, in 1995 and the M.S. and Ph.D. degrees in electrical engineering and computer sciences (EECS) from the University of California Berkeley, Berkeley, CA, USA, in 1997 and 2000, respectively.

While at Berkeley, his research was supported by a National Science Foundation (NSF) Graduate Research Fellowship. He is also a Co-Founder of Digital Persona, Inc. and the Principal Architect of the Digital Persona fingerprint recognition algorithm. He is currently a Professor in the Computer Science and Engineering Department, University of California San Diego, La Jolla, CA, USA. His research interests include computer vision and pattern recognition.

Prof. Belongie is a recipient of the NSF CAREER Award and the Alfred P. Sloan Research Fellowship. In 2004, *MIT Technology Review* named him to the list of the 100 top young technology innovators in the world (TR100).