IET Radar, Sonar & Navigation

Research Article

IET Journals
The Institution of Engineering and Technology

Target detection using features for sonar images

ISSN 1751-8784
Received on 12th May 2020
Revised 24th July 2020
Accepted on 24th August 2020
doi: 10.1049/iet-rsn.2020.0224
www.ietdl.org

Peter Tueller[1] ✉, *Ryan Kastner*[1], *Roee Diamant*[2]

[1]*Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093, USA*
[2]*Department of Marine Technologies, University of Haifa, Haifa 3498838, Israel*
✉ *E-mail: ptueller@eng.ucsd.edu*

Abstract: Robust object detection in sonar images is an important task for underwater exploration, navigation and mapping. Current methods make assumptions about the shape, highlight or shadow of an object, which may be invalid for some environments or targets. We focus on the area of feature extraction-based detection, which does not rely on information about the shape of the target, towards a robust framework for target detection for a variety of seabed structures and target types. The proposed framework first estimates the seabed type from the spatial distribution of features to determine the set of optimal parameters, and then obtains a set of features which are filtered according to intensity and distribution to yield a detection decision. The proposed method also provides a means to determine the seabed type, and a machine-learning based methodology to choose the feature detectors' parameters to match the evaluated seabed type. We report the performance of a variety of feature detectors for a simulated environment and of one feature detector for real sonar images. Results show the importance of choosing the parameters of the feature extractors based on the current environmental conditions and the proposed method obtains a favourable tradeoff between detection and false alarm rates.

1 Introduction

Sonar imaging is a valuable tool for analysing the seafloor. Detecting the presence of sunken ships and archaeological sites on the seafloor, mapping structures and objects, or performing pipeline inspection are all extremely useful tasks that are difficult for divers to accomplish, particularly in areas of significant depth. Current synthetic aperture sonar (SAS) imaging systems provide cm-scale resolution at a range of100 m and can be equipped on autonomous underwater vehicles (AUVs) or vessels that remain on the surface, such as ships or unmanned surface vehicles. Yet, sonar images are heterogeneous by nature and the design of a fully automated system for object detection that is robust to various seafloor environments and that functions well for different types of targets is challenging. Fig. 1 shows a sonar image that contains different seabed structures where Poseidonea, sand and sand ripples exist in the same image. Each of these has different features and characteristics that make detection challenging.

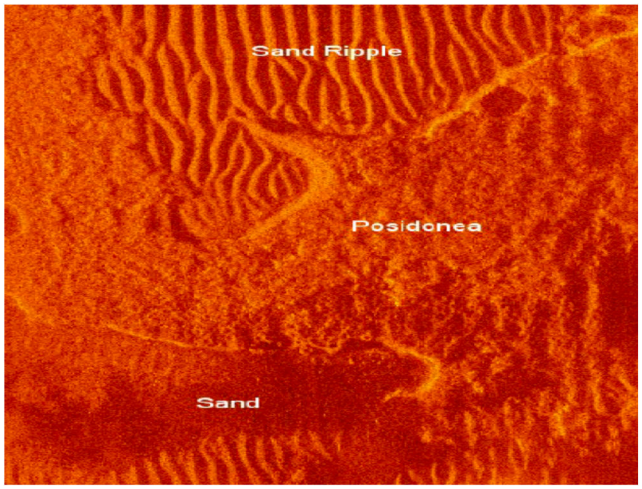


Fig. 1 *Sonar image with heterogenous seabed structures – sand, sand ripples and Posidonea plant life*

There are a number of approaches for the task of detection objects in a sonar image. One possibility is using a supervised learning approach that compares regions in the image to previously known patterns that correspond to objects. Barngrover [1] trains a Haar-like feature classifier on a large data set of objects for mine detection. These approaches are prone to false negatives when there is a mismatch between the template and the actual sonar image, e.g. due to changes with altitude and angle of the sonar system with respect to the target. Further, this supervised learning may only correspond to the case when the shape of the target is known a priori. Myers and Fawcett [2] suggest an alternative which segments a sonar image into five classes based on simple thresholding and compares them with previously established templates. These templates are generated from a database of targets at a variety of orientations. Recently, Abu *et al.* [3] abandons supervised learning altogether and proposes a statistical detection approach. Their classifier uses the sonar parameters and on the expected distribution of the highlight, shadow and background. The result is a detected region-of-interest based on segmentation of highlights and shadow, set by a likelihood ratio.

While the above approaches perform well for their application, they all inherently assume some knowledge about the target, e.g. the shape of the object in template matching or an assumed distribution for the highlight or shadow as in statistical detection methods. Instead, in this work, we focus on the case where such information is not available. Our method only relies on the assumption that the object is sufficiently different from its environment. That is we assume that there exist some features related only to the object to allow us to separate it from its environment. As shown in the example in Fig. 2, these objects can manifest as a bright circle against a sandy background or a shadow with complex geometry.

Feature detectors lie at the heart of all indirect image processing techniques. They do not require prior knowledge of the target; instead, they aim to identify geometry that is highly distinguishable from the background. As a result, feature detectors fulfil the robustness requirement for handling multiple sea environments and sonar images. Yet, although feature detectors have been successfully applied for the detection of objects in optical images or videos [4–7], there has been little done to explore their effectiveness in the sonar realm.

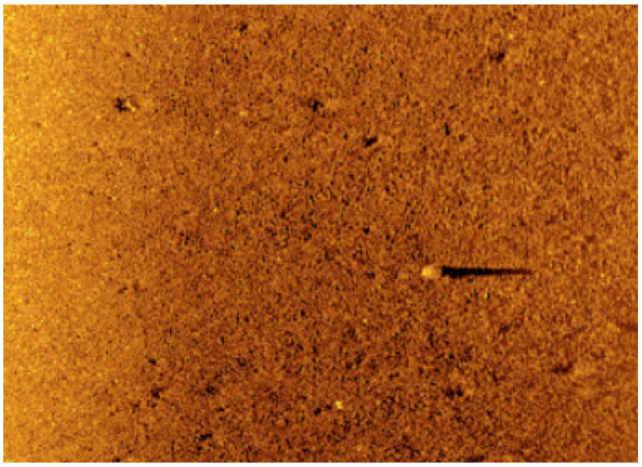


Fig. 2 *SAS image of the seafloor with an object appearing prominently in the centre right*

Table 1 Comparison of feature detectors

Feature detector	Description	Number of parameters, number of configurations	Average time per feature, μs	Pros	Cons
Harris and Stephens [9]	computes gradient with respect to each direction	5 parameters, 10,000 configurations	637.01	distinguishes corners and edges well	susceptible to scale variance
Shi *et al*. [10]	similar to Harris but uses a simpler thresholding method	3 parameters, 10,000 configurations	682.77	distinguishes corners and edges well	susceptible to scale variance
STAR [11]	multi-scale detector with no subsampling base on CenSure [12]	5 parameters, 1000 configurations	1257.68	robust to viewpoint changes	susceptible to brightness changes
FAST [13]	computes difference in brightness of neighbours in Bresenham circle	3 parameters, 500 configurations	16.42	efficient	poor noise robustness, susceptible to scale and illumination variance
SIFT [14]	computes oriented gradient histograms for patches around a point	4 parameters, 5000 configurations	793.94	rotation and scale invariant	computationally expensive, susceptible to blur
SURF [15]	a more efficient approximation of SIFT	3 parameters, 1000 configurations	73.35	faster than SIFT	susceptible to viewpoint and illumination change
ORB [16]	efficient replacement for SIFT that builds off of the FAST detector	6 parameters, 15,000 configurations	34.84	scale and rotation invariant, real-time, noise resilience	generally fewer features

The two main challenges of feature-based target detection are (i) separating the target from the background and (ii) setting up of the feature parameters. Regarding the former, the features extracted from the target and from target-like objects like rocks or ripples are hard to distinguish especially in a diverse seabed environment. Regarding the latter, the number of parameters range from 3 to 6 depending on the method used (see Table 1). Yet, the number of values for each parameter has thousands of possibilities. While the default parameters have been set to give good performance with simple images, these default parameters are far from being suitable in the general case [8] and we show that custom parameters are necessary for sonar images. Unfortunately, the brute force exploration of all possible configurations is not feasible. This is exacerbated by the fact that the performance of the feature extraction is highly affected by the unique patterns on the seafloor, such as sand, sand ripples or grass, as seen in Fig. 1.

We introduce a new methodology for feature-based target detection that addresses these two challenges. Our method accounts for the expected spatial separation of the target and the type of background in the sonar image. The method also efficiently explores the parameter space across the various environment backgrounds to find the optimal or near-optimal parameter configuration for a given sonar image background type. Our method does not guarantee optimal detection, but rather approaches optimality with low complexity.

Our method fits target identification applications like simultaneous localisation and mapping (SLAM) or identification of targets that stand out from their environment, e.g. pipeline exploration, archaeological surveys and mine detection. The algorithm works by first gathering features across a data set of sonar images and training a classifier to label each feature as object or non-object. We then modify the parameters of the feature detectors and find the set of best parameters to fit the evaluated seabed type using design space exploration. This solves the problem of needing to hand tune the parameters of the detector, and manually test tens of thousands of configurations to optimise performance. Finally, given a set of features generated by the optimal parameters, we perform target detection by prioritising detection of distinct features in space to identify the coordinates of single or multiple targets.

Contributions of the proposed work are as follows:

(i). A method for feature-based target detection on sonar imagery that takes into account the spatial identification of a target, as well as the type of background.
(ii). An implementation of an active learning design space exploration method to optimise feature detector parameters on sonar imagery.
(iii). A comparison of the relative performances of feature detectors on a simulated SAS data set.

We explore the performance of our algorithm via a designated simulator for sonar images, and over a set of sonar images were obtained using our own AUV platform. This SAS image simulator [17] has been used previously to train methods that have been experimentally verified on real SAS images, and thus we can

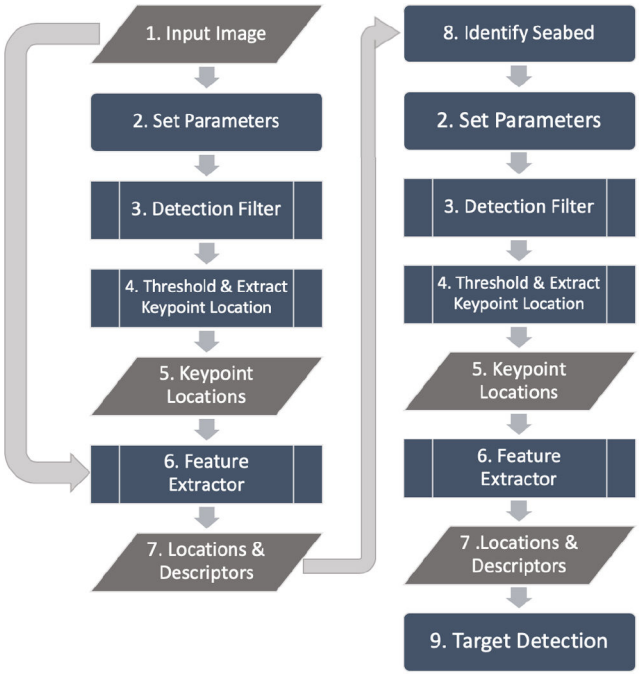


Fig. 3 *General framework for our feature-based detection algorithm*

expect similar reliability. The results show that identifying the background and utilising a feature detector with parameters optimised for that background yields the best result. Yet, we also observe that in the general case where the background is not known or estimated, utilising parameters trained for all backgrounds together produces the most reliable performance. Additionally, the design space exploration yields much higher performance than the default values for every feature detector. The optimised parameters for each background also yield different performances, dramatically so for some feature detectors.

This paper is organised as follows. In Section 2, we describe the state of the art in target detection on sonar images. Section 3 includes the details of our target detection method. In Section 4 we show how the feature exploration's parameters are set. Section 5 includes results from synthetic images and from real sonar images. We conclude in Section 6.

2 Related work

There has been a great deal of interest of non-model-based object detection in sonar imagery for segmentation and mapping unknown environments. However, the literature is sparse when it comes to analysing and optimising feature detection, which is the very first step of the process.

Feature-based detectors for sonar images have been mostly used for applications related to subsea navigation. In particular, the main approaches use feature extraction to identify similar objects within a set of sonar images. Williams *et al.* [18] perform two steps for such identification: first, they identify areas of constant depth as those may correspond to real objects. Then they classify pixels which are of consistent depth and not larger than a specified threshold as point features and feed them into subsequent feature matching for navigation. Similarly, the work by Johannsson *et al.* [19] utilises a simple computer vision pipeline of smoothing, gradient thresholding and clustering, which is similar to simple feature detectors that the work by the evaluates of Harris and Stephens [9]. They use these features to perform registration on overlapping frames, which in turn generates motion estimates between frames for SLAM.

The use of feature extraction for SLAM applications is common. Ribas *et al.* [20, 21] used mechanical forward-looking scanning (FLS) sonar that produces consecutive sonar images while the underwater vehicle is moving. The movement creates a warped image since different parts of the image will be observed from different locations. To remove these distortions, they use a motion estimate to re-align each pixel to a globally consistent frame. After this pre-processing step, they use a simple line extraction detector for target identification. Similarly, Fallon *et al.* [22] proposed a feature-based navigation system to incorporate features from high-resolution maps of the seafloor and relocate objects using a lower resolution FLS. To detect features from the FLS, they first adaptively thresholded each image based on altitude to remove background noise. Then, they segment the image and use a gradient-based feature detector with a threshold based on the average background image level to identify points of highlight and shadow. They then register these features back against the map and use them in a SLAM framework. Yet, the results show that performances are highly dependent on the seabed type.

While the above methods are tested for realistic sonar images, we identify some major gaps. In particular, the algorithms do not consider the spatial variation in sonar images. Even more importantly, the available methods optimise their parameters based on a small scale of cases, which may not fit the aim of robustness to various target types and different seabed structures. Specifically, these methods are missing an analysis of how to optimise the performance of these detectors in a broader set of environments. In this paper, we consider these two observations and offer a method for feature-based target detection that uses spatial information in target identification, as well as a rigorous methodology of how to set parameters to match a self-identified seabed type.

3 Feature-based target detection

3.1 System model and key idea

Our system model includes a single sonar image of any type, e.g. SAS, multibeam or FLS. The sonar image we consider is two dimensional (2D), although the 3D extension is straightforward. The sonar image may or may not include a target, but we assume that if a target is present, it is distinguishable from the seabed environment. That is it is either isolated compared to similar objects (e.g. a cluster of rocks) or it has a unique pattern. We do not assume knowledge about the shape of the target. Our goal is to identify the existence and location of a target in the given sonar image. In our scheme, we use knowledge of the seabed environment type: specifically sand ripples, grass and a mix of these backgrounds.

Fig. 3 outlines our general approach. We start with an image (block 1) and setting the parameters (block 2) to a feature detection algorithm to detect an object in the presence of any typical seafloor pattern. These default parameters will be the ones that give the best results across a mix of all backgrounds. The detection algorithm then returns key points (block 3), which are the pixel coordinates and calculated intensities associated with a feature. We rank the intensity of the key points and apply a threshold (block 4) (Note the threshold is determined as a system parameter and is not user defined.) to return the best key points and their coordinates. Once we identify the subset of features and locations (block 5), we extract the additional feature descriptors (blocks 6, 7) and we make a decision about the seabed type between four different clusters (block 8): sand ripple, grass, sand and mix. Then we execute the feature extraction algorithm again for the set of parameters that best fits the evaluated seabed environment (blocks 2–7). Finally, we perform target detection (block 9) based on the joint spatial spread of the detected features to identify a single or a set of targets that stand out from their environment either in terms of their pattern or by being spatially separate.

3.2 Feature extraction

We base our detector on an underlying feature extraction algorithm. In this work, we do not develop our own feature extracting algorithm. Rather, we explore a variety of such filters and provide the framework to include these algorithms for target detection in sonar images. Non-model-based feature detectors are all fundamentally based on analysing intensity gradients in an image, but differ in determining uniqueness based on patterns at a point (i.e. corner) or within a region surrounding a point (i.e. blob) or some combination of the two.

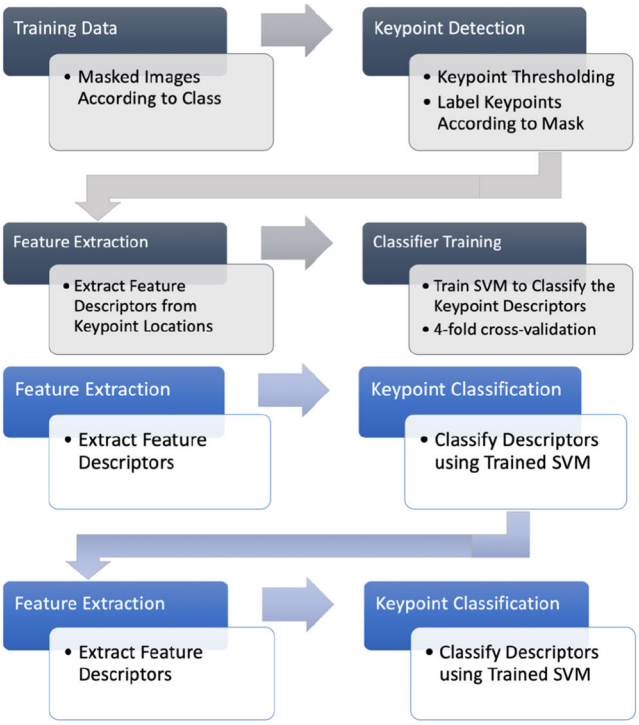


Fig. 4 *Feature classification. The first flow chart represents training an SVM model, and the second flow chart represents performing classification using that model*

$nClasses \leftarrow 2$
$kMeansIterations \leftarrow 15$
$c \leftarrow k\ Means\ Init\ Centroids(features, nClasses)$
for $i = 1 : kMeansIterations$ **do**
  $idx \leftarrow find\ Closest\ Centroids(features, c)$
  $c \leftarrow compute\ Centroids(features, idx, nClasses)$
end for
for $p \in features$ **do**
  $distCentroid1 \leftarrow norm(p - c(1))$
  $distCentroid2 \leftarrow norm(p - c(2))$
  if $distCentroid1 < distCentroid2$ **then**
    $distance(p) \leftarrow distCentroid1$
  else
    $distance(p) \leftarrow distCentroid2$
  end if
end for

Fig. 5 *Algorithm 1: distance to the nearest centroid*

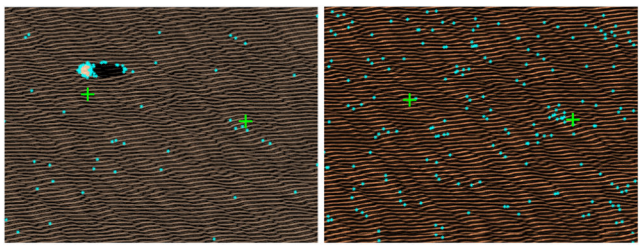


Fig. 6 *Placement of centroids on simulated sonar images. Features are displayed as cyan points, and the two centroids are the green plus symbols. Features associated with an object (left) will on average be closer to a centroid than features not associated with an object (right)*

We compare seven feature extraction algorithms: Harris and Stephens [9], Shi-Tomasi [10], STAR [11], FAST [13], SIFT [14], SURF [15] and ORB [16]. All of these are widely used, are considered for general purpose and are similar to detectors used by state-of-the-art feature-based navigation methods [19, 22]. Further, these algorithms are all available in the open-source library OpenCV [23]. These algorithms act as the detection filter block in Fig. 3. Given a set of parameters, the feature detector gives coordinates for a feature along with a single intensity value. Some detectors, namely STAR, SIFT, SURF and ORB, return additional information beyond the single intensity value, such as scale and orientation of each feature.

The key idea, pros and cons, average running time per feature and the number of parameters to tune for each of these detectors are summarised in Table 1. The average running time per feature is computed using each image in our data set (explained in Section 5.1) and every possible parameter configuration on a 3.2 GHz Intel Core i5 processor with 16-GB 2400-MHz DDR4 RAM. The parameters used by each of these detectors affect the performance in terms of feature detection. For example, SURF's three parameters affect the threshold at which points are accepted or rejected and the size and number of octave layers, which are fundamental components to its operation. These, in turn, affect the size of features that SURF detects and returns.

3.3 Feature classification

The output of the feature extraction process is a set of pixel coordinates with associated intensity values, i.e. a set of features. To decide on the important features out of those that passed the detection filter by means of classification, we label each feature with '1' if it is related to a target and '0' otherwise. As illustrated in Fig. 4, for classification we use this set of labelled features to train a support vector machine (SVM) model via a four-fold validation process over a radial basis function kernel. SVM was chosen due to its ability to perform well also for small sets of training data. The model takes the detectors' outputs (feature intensity, and, if relevant, scale and orientation), as well as the location of the identified features in the image.

Recall we assume that features corresponding to an object in a sonar image are isolated from other features. Thus, to classify an object, we use the relative position information to serve as an input to our SVM model. We avoid setting the absolute position information as an input, though, as an object may appear at any position in the image. Instead, we calculate two characteristics that relate to the feature's density. The first parameter describes any given feature's distance to the closest of two centroids, which we calculate given the total distribution of features in a single image. In images with a single object, numerous features around the object will create a centroid that is very close to the position of the object. Therefore, theoretically, features that have smaller distances to their nearest centroids would be more likely associated with an object. The pseudocode to set up the distance characteristic is outlined in Algorithm 1 (see Fig. 5) and a visual representation is depicted in Fig. 6.

The second parameter is a density index, as related by (1), that quantifies the relative distance from a single point to all other points in the image. This way, features that are close to many other features will have a lower value than features that are isolated in the image.

These two feature density parameters and the parameters inherently calculated by the feature detector all make up the 'key point descriptors' in Fig. 4 that are input to the SVM model. After training, the SVM, for each key point descriptor, will output either a '1' for 'object' or '0' for 'non-object'. By this method, we perform feature classification.

3.4 Identifying seabed structure

Once we extract the set of features, $\mathcal{F}$, we make a decision on the type of seabed. Information on the seabed type allows us to better determine the feature extraction algorithm's parameters. We take a set of images of an unknown seabed and compute the number of features, the distribution of features within images, and feature descriptors. These criteria are compared to averages of the same criteria computed from a representative data set of three environments. If the criteria from $\mathcal{F}$ matches one of these environments within a predetermined threshold, we make a decision on the type of seabed and apply the optimal parameters for that seabed. In Fig. 7, we distinguish between seagrass, sand and sand ripples. We use mix and its corresponding optimal parameters

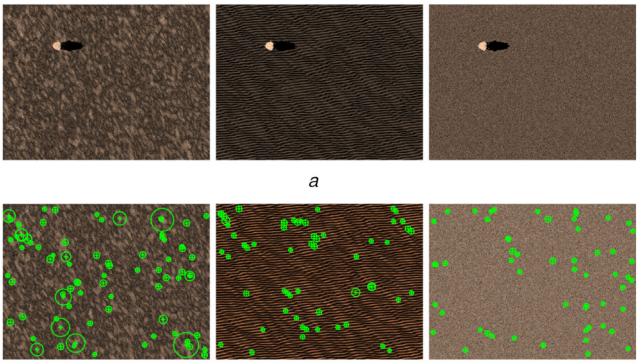


Fig. 7 *Simulated sonar images with grass (left), ripple (middle), and sand (right) seabeds*

(a) Example simulated sonar images, ***(b)*** Distribution of SURF features on the simulated seabeds. The location of a feature is represented by a green plus symbol and the scale associated with that feature is represented by a green circle around it

for a seabed environment that is not well distinguished by the above criteria. Then, once we properly set these parameters, we execute the feature extraction algorithm again so that we can glean the most accurate set of features possible.

We base our decision regarding the type of seabed on fixed thresholds on the distribution, the number of the features extracted and descriptors from the feature detector. We learn these thresholds from a simulator built by the definitions from Abu and Diamant [17] for the different three seabeds in Fig. 7, while normalising by the number of pixels in the sonar image.

To quantify the spatial spread of the features, we project the 2D location space of each feature f_i into a closeness index. Let $\boldsymbol{r}_i$ be the 2D location of the ith feature and $d_{i,j}$ be the Euclidean distance between features i and j. We choose the closeness index to be

$$h_i = \sum_{k,k\neq i} \sum_{j,j\neq i,k} \left(1 - \frac{(\boldsymbol{r}_i - \boldsymbol{r}_k)(\boldsymbol{r}_j - \boldsymbol{r}_k)}{\| d_{i,k} \| \| d_{i,j} \|}\right), \tag{1}$$

which for a point i iterates through each feature k and compares it to every pair of feature i, j. If i is far from most other features, and those features are relatively close to each other compared to i, then the equation will evaluate to a relatively large value. Thus, h_i measures how isolated feature i is relative to the other features, such that the higher h_i is, the more isolated feature i is. Then, we compare the mean of $h_i \forall i$ together with the number of extracted features by the above logic to determine if the current seabed is composed of a few/large number of isolated/dense features, and then identify the seabed environment.

3.5 Target detection

Once we identify the seabed, the feature classification step is run with the detector parameters tuned for that specific seabed type. This yields a set of features that we can now analyse to identify the coordinates of a single or a set number of targets within an image. We use three methods for identifying a target. The first method sorts the features by the intensity and applies a simple threshold to label the n features with the highest intensities as n targets in the image. This is the method that is typically used, as by design a higher intensity indicates a 'stronger' feature.

The second method calculates weights for each feature using (1). The weights are then normalised by calculating

$$\hat{w}_i = \frac{h_i}{\sum_j h_j}, \tag{2}$$

and then the weight $\hat{w}_i$ of each feature is multiplied by its intensity v_i

$$\bar{v}_i = \hat{w}_i v_i . \tag{3}$$

The intuition here is that intensity is not the only quality of a feature indicating it can correspond to an object. The spatial relationship between one feature and all other features can also hint at the existence of an object. Combining this information will theoretically lead to a more robust target detection.

The third method utilises the same weights $\hat{w}$ and multiplies by the intensity of a feature normalised with a fixed window:

$$\tilde{v}_i = \hat{w}_i \frac{v_i}{(1/\| S_j \|)\sum_{j\in S_j} v_j}, \tag{4}$$

where S_j is the set of features within a 5×5 pixel window surrounding feature j. Features corresponding to objects will theoretically be more different than their immediately surrounding features in comparison to the difference between features corresponding to noise or backgrounds. However, this method relies on the window size being larger than the objects in the image so that the intensity differences are more significant.

Finally, similar to the first method, a threshold is set for the sorted values of $\bar{v}$ and $\tilde{v}$ to identify the n targets within the image.

4 Design space exploration

One of the main challenges in our feature-based target detection is how to determine the optimal set of parameters for each feature detection algorithm to obtain a favourable trade-off between the true positive rate and the false positive rate. While each of the algorithms we consider may only have four or five parameters, the number of possible combinations quickly builds up as the granularity of the sampling for each parameter increases.

To illustrate this, consider an algorithm with only five parameters each having only four possible values. Then, the total number of designs is 1024. Now consider an evaluation process for a single design that lasts for 10 min. The result would be approximately 1 week of continuous computing to fully explore the design space. Naturally, this duration compounds when several seabed environments and target types are explored. Clearly, a method to reduce the computing time while still achieving a good approximation of the set of optimal designs is required. While for optic images the default parameters supplied with the feature extractors are designed to be suitable, to the best of our knowledge, how to set the parameters for sonar images has not been explored.

4.1 Exploring the space of parameters

We propose a parameter-tuning methodology based on active learning. While there exists several options for parameter exploration via active learning, e.g. [24–26], we adopt previous design space exploration work known as ATNE [27] that we have modified for our application, which works by sampling design based on whether it guesses that the results will either further explore the design space or get closer to the optimal edge. Instead of brute-force exploring the whole sample domain, ATNE covers a subset sufficient to estimate the pareto-optimal front for even very complicated non-linear design spaces. It marks an improvement on previous active learning papers by requiring fewer initial random samples: 6 compared to 3000 [26]. Additionally, it can handle complex design spaces of a variety of sizes and retain similar prediction quality. By running ATNE on each detector for each data set of seabed environments (200 images for grass, ripple and sand and all 600 images for the mix data set), we can generate a set of feature detector parameters. The result is decreased by 70% in the exploration time of the best set of parameters. Each design can take up to 8 min to evaluate on an Intel Core i5-6500 Skylake Quad 3.2 GHz with 16 GB of memory and a Sapphire Radeon Nitro R7 370 4 GB GPU. Thus, with a feature detector with 500 possible designs, it saves about 47 h of processing time.

To demonstrate the operation of ATNE, Fig. 8 shows that a 30% sampling of the design space with the predicted best configurations in black, compared to the true optimal (brute forced) pareto front in red. Additionally, the sampled designs are shown with a magenta dot, and the full-space of designs is shown simply as blue circles. It is clear that ATNE, though sampling only a portion of the designs,

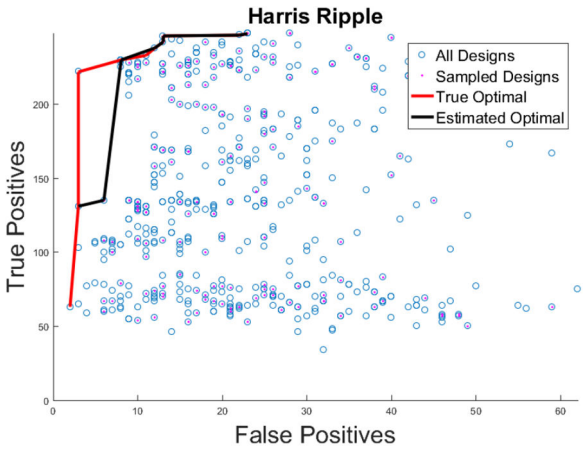


Fig. 8 *Example of a design space exploration. The red line shows the true set of optimal designs, and the black line shows what was estimated utilising only the magenta points*

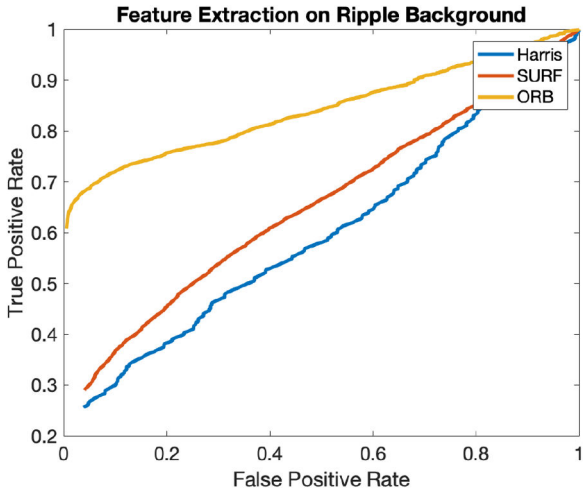


Fig. 9 *Three detectors' ROCs for a single design*

essentially reconstructs the space and estimates a pareto-optimal set of designs that are close to the ground truth optimal set. We measure the closeness to ground truth using the average minimum distance from reference set (ADRS) [27, 28] and ATNE. ADRS is defined as

$$\mathrm{ADRS}(P_{\mathrm{gt}}, P_{\mathrm{pred}}) = \frac{1}{|P_{\mathrm{gt}}|} \sum_{\boldsymbol{p}_{\mathrm{gt}} \in P_{\mathrm{gt}}} \min_{\boldsymbol{p}_{\mathrm{pred}} \in P_{\mathrm{pred}}} d(\boldsymbol{p}_{\mathrm{gt}}, \boldsymbol{p}_{\mathrm{pred}}), \quad (5)$$

where P_{pred} and P_{gt} are the sets of points in the predicted pareto front and the ground truth pareto front, and the function $d(\boldsymbol{p}_{\mathrm{gt}}, \boldsymbol{p}_{\mathrm{pred}})$ represents the distance between a point in the ground truth set and a point in the predicted set.

4.2 Utilisation function for parameter exploration

As any reinforcement learning technique, ATNE requires an objective function or space to converge to. Since our aim is target detection, we choose the receiver operator characteristic (ROC) which compares the number of correctly classified object features against the number of incorrectly classified non-object features. One alternative is the precision–recall curve, which is used in cases where there is an abundance of one class over another. Although there is an imbalance in our case (there are far fewer object pixels than background pixels), the ROC is a better choice because it is more important to us that the object as a whole is identified robustly in comparison with the background, rather than every single object pixel be identified. The ROC is a common way of choosing parameters for detections; yet, ROCs can also be used to compare detectors. In that context, we recall that a desired detector would yield a better false positive–true positive trade off in most of the cases compared to other options. If we use a graphical explanation (see for example Fig. 9), the best detector would be the one obtaining the far left curve of the ROC. Utilising this observation, as a utility function, we feed ATNE with approximations of the ROC by sampling three true positive–false positive pairs on the curve.

Our procedure works as follows. For each feature detector design and parameter configuration, we execute our detection and classification workflow (see Section 3) to obtain a list of features labelled as object. This list is then compared with ground truth information to yield a precision–recall pair for each parameter configuration. Once ATNE tests one design and configuration, it decides which design to sample next, based on its updated estimate of the design space. After sampling 30% of the space, we can then reasonably say that ATNE has recreated the pareto-optimal front of the design space. The pareto-optimal front will consist of the points that are on the edge of the scatter plot with the fewest false positives and most true positives.

5 Results

In this section, we explore the performance of our feature-based target detection scheme. Our data set comprises 600 simulated sonar images to determine the parameter space of each feature extraction algorithm. In addition, we explore the performance for some genuine sonar images that we gathered in multiple sea experiments. This data set includes sonar images from a SAS mounted on our Eca-robotics Alister A18 AUV and we gather multibeam images using a surface ship-mounted multibeam. We start by exploring the set of parameters based on our simulator and then provide sea trial results for target detection.

5.1 Simulations

The simulated SAS images depict a single target on one of three backgrounds: grass, sand ripple or sand. An example of all three environments is shown in Fig. 7. The backgrounds have equal representation in the data set, such that there are 200 grass-, 200 ripple- and 200 sand-based sonar images. We obtain this data set from a distribution function tailored to each of these seabed environments (see [17] for further details).

Within each background, there are many different target, shadow and background intensities, so that there is sufficient variety represented in the data set. As the data set is split into three backgrounds, there are four experiments that we can run: one for each background and one for a mix of all backgrounds. This allows us to test the suitability of various feature detectors for particular backgrounds in comparison to each other. This is done by measuring the number of true positives and false positives that each detector generates with a specific design, where the number of true positives corresponds to all the features that are within the bounding box of the object, and the number of false positives corresponds to all other features which are outside that bounding box. We relate these with (6) and (7) for true positives and false

positives, respectively. We measure the true positive performance by dividing the number of true positives, N_{tp}, by the number of pixels that should have been detected but were not (i.e. false negatives), N_{fn}, plus the number of true positives. This denominator corresponds to the number of pixels that correspond to the object in the image. We measure the false positive performance by dividing the number of false positives, N_{fp}, by the number of pixels that were correctly not identified (i.e. true negatives), N_{tn}, plus the number of false positives. This denominator corresponds to the number of pixels that correspond to the background of the image:

$$P_{tp} = \frac{N_{tp}}{N_{tp} + N_{fn}}, \tag{6}$$

$$P_{fp} = \frac{N_{\neg fp}}{N_{fp} + N_{tn}}, \tag{7}$$

5.2 Classification results

Table 2 illustrates the performance of each of the seven feature detectors on each of the four data sets.

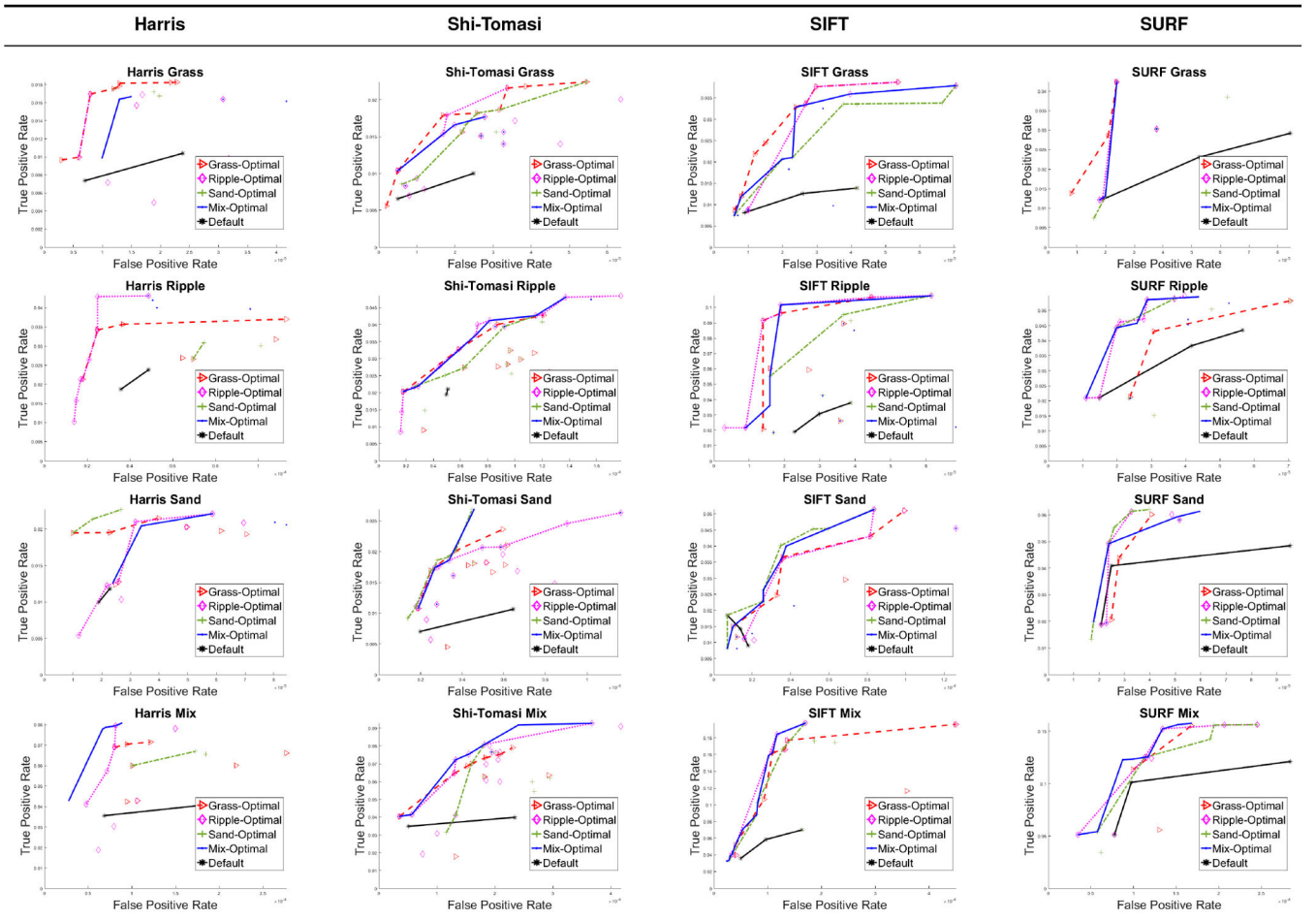
We plot the optimal set of designs for each data set for a given feature detector for comparison, with the true positive rate on the y-axis calculated by (6) and the false positive rate on the x-axis calculated by (7). With Fig. 10 we show the performance of the grass, ripple, sand and mix optimal designs for each of the four data sets. In each plot, there is a line of best fit for each of the optimal design sets to obtain an estimate of the pareto-optimal curve. For example, in the Harris grass plot, there are five sets of designs that we test on the grass data set. As expected, the designs that are optimised for the grass data set perform the best, as the red dotted line is the closest to the top left corner. The designs optimised for the other data sets are shown as well: ripple in magenta, sand in green, mix in blue and the default designs with no optimisations in black. In this particular instance, the sand designs appear closest to the most optimal set of designs.

We calculate the distance between each curve on a single plot using ADRS as defined in (5). Table 3 shows the ADRS values for each plot. ADRS gives a measure of how far the results are from the optimal results, so lower is better. An ADRS of 0 indicates that the sets exactly match. We calculate ADRS based off of the number of true positives and false positives for each point, rather than the performance metrics in (6) and (7). Since all the images in the data set contain the same number of pixels, this will represent the same relative performance as the plots in Fig. 10. For example, the Harris grass plot in Fig. 10 corresponds to the optimal grass row in the Harris sub-table in Table 3. In this particular case, as we already assumed from the plot, designs trained on the sand data set and tested on the grass data set are the closest to the grass optimal front, with an ADRS value of 1.47. This means that if we substitute the parameters in Harris plots which have been optimised for the sand data set, we will get a similar performance as if we used grass parameters, which are optimised for the actual data set that is being tested.

In all cases, as the plots illustrate, all optimised designs (even the ones trained and tested on mismatched data sets) perform better than the designs with default parameters. Additionally, in most of the combinations, using the designs optimised for the particular background that is being viewed will dramatically improve performance compared to using a mismatched training data set.

We would expect that there be a consensus of which set of optimal designs would be best to use without any knowledge of the background that is being tested. In other words, that a single column in each of the subtables in Table 3 will contain a majority of bold values, such as is the case for the Shi-Tomasi table, where

Table 2 ROC curves



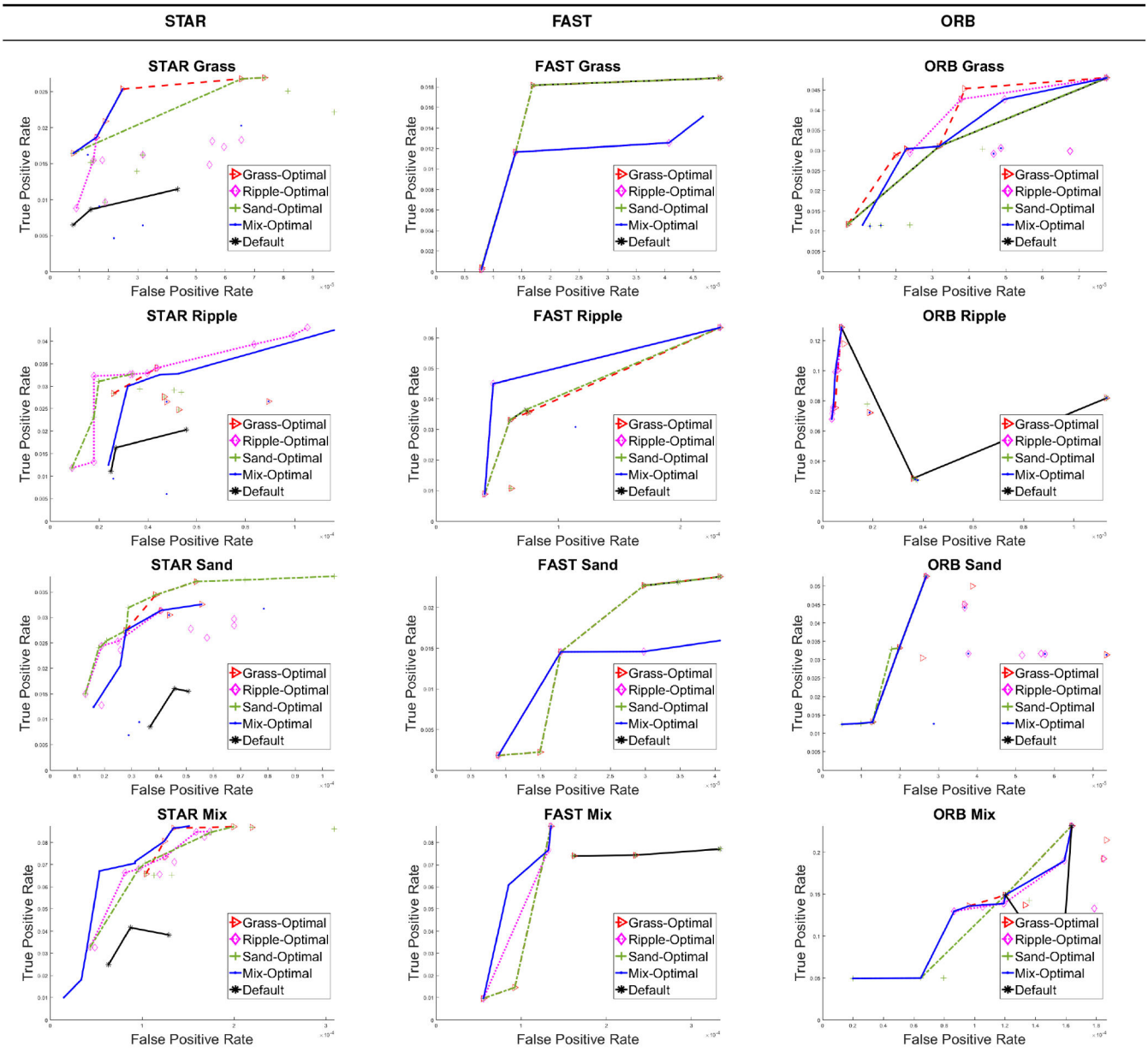


Table 3 ADRS values for each feature detector

Harris	Grass	Ripple	Sand	Mix	Shi-Tomasi	Grass	Ripple	Sand	Mix
optimal grass	0	9.14	**1.47**	5.12	optimal grass	0	9.01	**5.12**	7.78
optimal ripple	**9.24**	0	24.15	16.19	optimal ripple	5.02	0	7.44	**0.50**
optimal sand	13.35	34.23	0	**12.41**	optimal sand	7.37	6.24	0	**2.36**
optimal mix	86.16	**56.76**	142.80	0	optimal mix	28.17	**19.31**	28.59	0

SIFT	Grass	Ripple	Sand	Mix	SURF	Grass	Ripple	Sand	Mix
optimal grass	0	**2.30**	6.94	6.62	optimal grass	0	**10.06**	20.50	10.17
optimal ripple	79.96	0	93.25	**50.41**	optimal ripple	5.23	0	17.32	**3.91**
optimal sand	28.65	19.50	0	**18.04**	optimal sand	18.22	9.45	0	**7.79**
optimal mix	22.32	**11.90**	25.16	0	optimal mix	32.59	**1.82**	39.08	0

the designs optimised for the mix data set are always second best in each of the individual data sets.

5.3 Comparing feature extraction algorithms

While is not our main aim, our method provides an efficient way to compare the performance of the seven feature extraction algorithms used. We base our comparison on the ROC performance of the seven methods in Fig. 10 and Table 3.

STAR	Grass	Ripple	Sand	Mix	FAST	Grass	Ripple	Sand	Mix
optimal grass	0	20.00	**13.36**	33.35	optimal grass	0	2.95	**0**	9.77
optimal ripple	45.27	0	35.73	**25.60**	optimal ripple	46.33	0	52.87	**35.41**
optimal sand	**4.56**	7.60	0	25.35	optimal sand	**0**	0.24	0	3.68
optimal mix	**7.11**	10.77	9.46	0	optimal mix	20.22	**0**	17.49	0

ORB	Grass	Ripple	Sand	Mix
optimal grass	0	3.65	2.13	**1.71**
optimal ripple	**60.22**	0	289.55	140.56
optimal sand	19.24	26.60	0	**12.20**
optimal mix	25.52	9.58	**5.70**	0

The data set with the designs closest to each optimal front is in bold. This is a numerical representation of the plots in Fig. 10 and support the conclusions drawn from visual analysis.

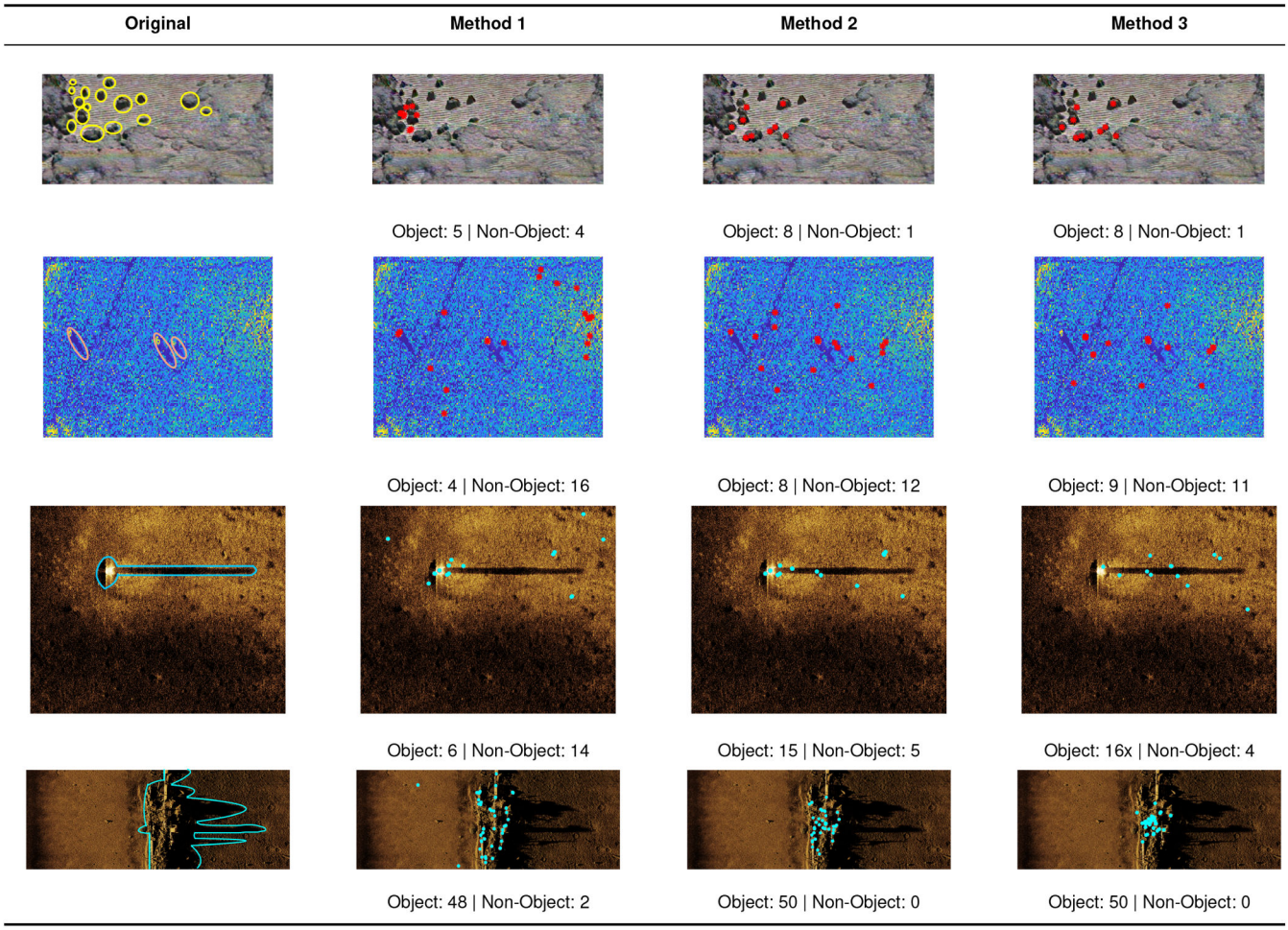


Fig. 10 *Target detection methods on real sonar images with the SURF detector*

From the plots, and verifying with the ADRS values, the Shi-Tomasi, SURF and ORB designs optimised for the data set containing a mix of all backgrounds performs consistently well in comparison to the other feature detectors in all categories. These are the most robust of the seven-feature detectors that were tested and would function better than the others in environments with a variety of seabeds if there was no ability to detect the type of seabed.

Harris is interesting in that each set of designs trained on the individual data sets perform very close to optimally when tested on the data sets all together, but the designs trained on the data sets all together only perform averagely when tested on individual data sets. FAST is also unique in that it has a smaller design space and a number of the designs have identical performance. This yields a situation where the designs trained on grass and sand are identically optimal when testing on grass and sand data sets, but have different ADRS values when testing on the ripple or mix data sets.

5.4 Real sonar images

Fig. 10 shows the performance of the target detection step on real sonar images. The first two images are from a surface ship-mounted multibeam sonar. The first image contains multiple prominent rocks on a non-homogeneous seafloor. The second image contains three targets, deliberately placed on the seafloor but is not as prominent against the background. The third and fourth images are from our AUV mounted SAS, and each contains a single target: a simple pipe and a complex shipwreck, respectively. For the first two images, which contain multiple targets, we expect that Method 1, a simple threshold filter, will identify the single most prominent target. We also expect that Methods 2 and 3, which

take the distribution of features into account, will identify other targets in the image as well. Additionally, for all images we expect that features with high intensities that correspond to the seafloor will be eliminated by Methods 2 and 3.

We show results for the SURF feature extraction algorithm, and note that the results are similar in trends for the other feature extraction algorithms. The images each have at least one target, outlined in the first column. The methods we test are outlined in Section 3.5 here Method 1 corresponds to the simple feature intensity thresholding, Method 2 corresponds to (3) and Method 3 corresponds to (4).

We observe that Methods 2 and 3 present a clear improvement over Method 1. In particular, separate targets are identified more strongly in the first and second images, and in the first and third images the vast majority of false detections on the background are eliminated. The fourth image depicts a single, large, complex feature, and as such it is easily identifiable by all methods.

6 Conclusion

In this paper, we presented a feature extraction-based method to detect targets in sonar imagery. Our method does not require prior knowledge of the shape or distribution of the target, and is capable of target detection for multiple seabed environments and sonar types. We described the framework of our detector, and showed how it exploits the spatial diversity of the extracted features. Further, we described our methodology for exploring the parameter space across the various environment backgrounds to determine the parameters of the feature extraction algorithms. We demonstrated the applicability of our approach over a simulated data set and examined it on genuine sonar imagery of different sonar types. The results showed that optimised parameters yield much higher performances than default parameters. Further work will combine the positives of all feature extraction algorithms to yield a better and more robust detection rate. Additionally, the use of neural networks may illuminate additional optimisations that the SVMs we used could not, and presents an interesting opportunity for future research. This could be paired with additional research into a more robust sonar image simulator. The one used in this paper has limited dimensionality, and as such generating additional images to use a neural network runs the risk of overfitting [29].

7 Acknowledgments

This work was sponsored in part by the NATO Science for Peace and Security Programme under grant G5293

8 References

[1] Barngrover, C.M.: ‘Automated detection of mine-like objects in side scan sonar imagery’. UC San Diego, 2014

[2] Myers, V., Fawcett, J.: ‘A template matching procedure for automatic target recognition in synthetic aperture sonar imagery’, *IEEE Signal Process. Lett.*, 2010, **17**, (7), pp. 683–686

[3] Abu, A., Diamant, R.: ‘A statistically-based method for the detection of underwater objects in sonar imagery’, *IEEE Sens. J.*, 2019, **19**, (16), pp. 6858–6871

[4] Hu, W.C., Chen, C.H., Chen, T.Y., *et al.*: ‘Moving object detection and tracking from video captured by moving camera’, *J. Vis. Commun. Image Represent.*, 2015, **30**, pp. 164–180

[5] Li, J., Wang, T., Zhang, Y.: ‘Face detection using surf cascade’. 2011 IEEE int. Conf. on Computer Vision Workshops (ICCV workshops), Barcelona, Spain, 2011, pp. 2183–2190

[6] Dollár, P., Appel, R., Belongie, S., *et al.*: ‘Fast feature pyramids for object detection’, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2014, **36**, (8), pp. 1532–1545

[7] Zhao, W.L., Ngo, C.W.: ‘Flip-invariant sift for copy and object detection’, *IEEE Trans. Image Process.*, 2013, **22**, (3), pp. 980–991

[8] May, M., Turner, M.J., Morris, T.: ‘Scale invariant feature transform: a graphical parameter analysis’. Proc. of the British Machine Vision Conf. (BMVC) 2010 UK Postgraduate Workshop, Aberystwyth, Wales, UK, 2010, pp. 1–11

[9] Harris, C., Stephens, M.: ‘A combined corner and edge detector’. Alvey vision Conf., Manchester, England, UK, 1988, vol. 15, pp. 10–5244

[10] Shi, J.: ‘Good features to track’. 1994 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 1994. Proc. CVPR'94, Seattle, WA, USA, 1994, pp. 593–600

[11] OpenCV: ‘STAR class description’, 2018. Accessed 18 April 2018. Available at https://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_feature_detectors.htmlStarFeatureDetector%20:%20public%20FeatureDetector

[12] Agrawal, M., Konolige, K., Blas, M.R.: ‘Censure: center surround extremas for realtime feature detection and matching’. European Conf. on Computer Vision, Marseille, France, 2008, pp. 102–115

[13] Rosten, E., Drummond, T.: ‘Machine learning for high-speed corner detection’. European Conf. on Computer Vision, Graz, Austria, 2006, pp. 430–443

[14] Lowe, D.G.: ‘Distinctive image features from scale-invariant keypoints’, *Int. J. Comput. Vis.*, 2004, **60**, (2), pp. 91–110

[15] Bay, H., Tuytelaars, T., Van-Gool, L.: ‘Surf: speeded up robust features’. European Conf. on Computer Vision, Graz, Austria, 2006, pp. 404–417

[16] Rublee, E., Rabaud, V., Konolige, K., *et al.*: ‘Orb: an efficient alternative to sift or surf’. 2011 IEEE int. Conf. on Computer Vision (ICCV), Barcelona, Spain, 2011, pp. 2564–2571

[17] Abu, A., Diamant, R.: ‘Unsupervised local spatial mixture segmentation of underwater objects in sonar images’, *IEEE J. Ocean. Eng.*, 2018, **44**, (4), pp. 1179–1197

[18] Williams, S., Dissanayake, G., Durrant-Whyte, H.: ‘Towards terrain-aided navigation for underwater robotics’, *Adv. Robot.*, 2001, **15**, (5), pp. 533–549

[19] Johannsson, H., Kaess, M., Englot, B., *et al.*: ‘Imaging sonar-aided navigation for autonomous underwater harbor surveillance’. 2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 2010, pp. 4396–4403

[20] Ribas, D., Ridao, P., Neira, J., *et al.*: ‘Slam using an imaging sonar for partially structured underwater environments’. 2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Beijing, China, 2006, pp. 5040–5045

[21] Ribas, D., Ridao, P., Tardós, J.D., *et al.*: ‘Underwater slam in man-made structured environments’, *J. Field Robot.*, 2008, **25**, (11–12), pp. 898–921

[22] Fallon, M.F., Folkesson, J., McClelland, H., *et al.*: ‘Relocating underwater features autonomously using sonar-based slam’, *IEEE J. Ocean. Eng.*, 2013, **38**, (3), pp. 500–513

[23] Bradski, G.: ‘The OpenCV library’, *Dr Dobb's J. Softw. Tools*, 2000, **25**, pp. 120–125

[24] Liu, H.Y., Carloni, L.P.: ‘On learning-based methods for design-space exploration with high-level synthesis’. Proc. 50th annual design automation Conf., Austin, TX, USA, 2013, p. 50

[25] Ipek, E., McKee, S.A., Singh, K., *et al.*: ‘Efficient architectural design space exploration via predictive modeling’, *ACM Trans. Archit. Code Opt. (TACO)*, 2008, **4**, (4), p. 1

[26] Bodin, B., Nardi, L., Zia, M.Z., *et al.*: ‘Integrating algorithmic parameters into benchmarking and design space exploration in 3d scene understanding’. 2016 Int. Conf. on Parallel Architecture and Compilation Techniques (PACT), Haifa, Israel, 2016, pp. 57–69

[27] Meng, P., Althoff, A., Gautier, Q., *et al.*: ‘Adaptive threshold non-pareto elimination: Re-thinking machine learning for system level design space exploration on fpgas’. Proc. 2016 Conf. on Design, Automation & Test in Europe, Dresden, Germany, 2016, pp. 918–923

[28] Palermo, G., Silvano, C., Zaccaria, V.: ‘Respir: a response surface-based pareto iterative refinement for application-specific design space exploration’, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2009, **28**, (12), pp. 1816–1829

[29] Tueller, P., Kastner, R., Diamant, R.: ‘A comparison of feature detectors for underwater sonar imagery’. OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 2018, pp. 1–6