A Comparison of Feature Detectors for Underwater Sonar Imagery

Peter Tueller
Computer Science and Engineering
University of California San Diego
La Jolla, CA, United States
ptueller@ucsd.edu

Ryan Kastner
Computer Science and Engineering
University of California San Diego
La Jolla, CA, United States
rkastner@ucsd.edu

Roee Diamant
Dep. of Marine Technologies
University of Haifa
Haifa, Israel
roee.d@univ.haifa.ac.il

***Abstract*—In this work we compare the performance of seven popular feature detection algorithms on a synthetic sonar image dataset. The dataset consists of a single mine-like object (MLO) superimposed on three different backgrounds: grass, sand ripple, and sand. We explore the performance of Harris, Shi-Tomasi, SIFT, SURF, STAR, FAST, and ORB on each of these backgrounds, and all the backgrounds at once by training an SVM classifier. Performance is evaluated with ROC curves by comparing the number of correctly identified features belonging to objects (True Positives) and the number of incorrectly identified features belonging to background noise (False Positives).**

***Index Terms*—feature detection, sonar, visual odometry**

I. Introduction

Deriving information in an underwater environment is difficult; radio frequencies typically used for terrestrial communications do not propagate well, and cameras are severely hindered by a lack of visibility. Sonar imagery, on the other hand, can be a valuable tool for visualizing and analyzing the seafloor. These images can be used for a number of tasks, such as mine-like object (MLO) detection [3] [12], or odometry and Simultaneous Localization and Mapping (SLAM) [10] [8]. Sonar has a number of weaknesses, though: it can suffer from low resolution and high image acquisition time, as well as significant shadows and distortions in the images themselves.

Current analysis of sonar imagery is done through human observation or segmentation, where an image is divided into 'object', 'shadow', and 'background' classes. At the core of any autonomous approach such as segmentation and object classification is a need for quantizing the details in an image. This is typically done through a process called feature detection.

Feature detection is an image processing technique that identifies unique pixel regions that have a high probability of being associated with a real object, and as such will provide good initial estimates to begin the classification process 1. Feature detectors have historically been developed and tested on camera images. As opposed to light, sonar images are created by sound reflections from the sea floor and from submerged objects. Sonar images have a unique noise profile and intensities that are of different texture from optical images. In particular, sonar images are characterized by intensity inhomogeneous backgrounds that are hard to model.

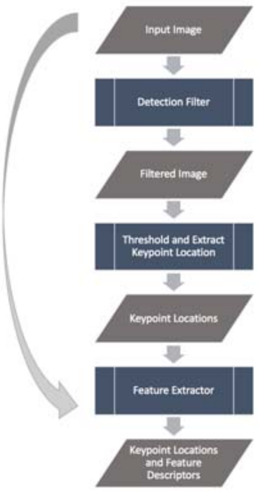

Input Image
Detection Filter
Filtered Image
Threshold and Extract Keypoint Location
Keypoint Locations
Feature Extractor
Keypoint Locations and Feature Descriptors

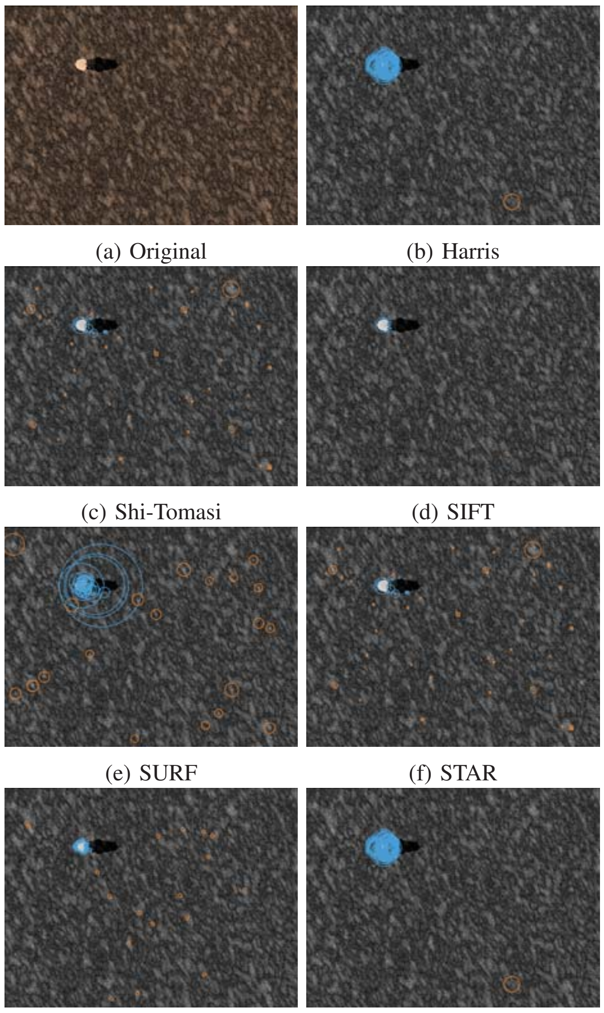
Fig. 1: General Feature Detection.

In this paper we present a method for comparison of feature detector algorithms on sonar imagery.

II. Feature Detection

The particular feature detectors that we used are outlined in Table I and their performance on one of the images in our dataset can be seen in Figure 2. We chose these seven detectors because they are the state of the art general-purpose detectors that are readily available in the open source package OpenCV [5].

Previous work on sonar feature detection has typically used simple and fundamental methods of detection [3] [8] [17] [13] whereas our detectors build on these fundamentals and have been extensively used in other domains, such as visual SLAM and object tracking in videos. These methods generally use gradient filtering, as one would do for line detection, then thresholding to select points of high intensity (so far this is extremely similar to our Harris and Shi-Tomasi detectors), and then clustering to remove extraneous features. Our feature detectors examine a point in an image first, and then quantify the pixels around it to determine how "strong" of a point it is. For example, FAST analyzes a ring of 16 pixels around a point and if they are all a certain amount brighter or darker than the center, it qualifies as a point. Additionally there is a non-maximum suppression step where the only strongest FAST features in a local area are retained. SIFT, SURF, STAR, and ORB additionally add a scale component, where corners of different sizes can be detected in the same pass. It is for this flexibility and complexity that these feature detectors were chosen over simpler approaches that have been historically used.



(a) Original (b) Harris (c) Shi-Tomasi (d) SIFT (e) SURF (f) STAR (g) FAST (h) ORB

Fig. 2: Simulated Images with Feature Detector Response.

III. Object Classification

To perform classification, we train and execute an SVM model, which can generically be seen in Figure 3. First, the features across the entire dataset are randomly split four ways. One part is retained as the test set and the remaining three are retained as a single training set. That training set is then randomly split four ways, and one part is retained as the cross-validation set. The SVM model is trained on the remaining three parts using a set value for gamma and cost. The model's performance on the cross-validation set is examined and saved. The model is then retrained using a different gamma and cost and the performance is examined, and this is done a total of thirty times. The model that used the gamma-cost combination that resulted in the highest performance on the cross-validation set is then used to evaluate performance on the test set.

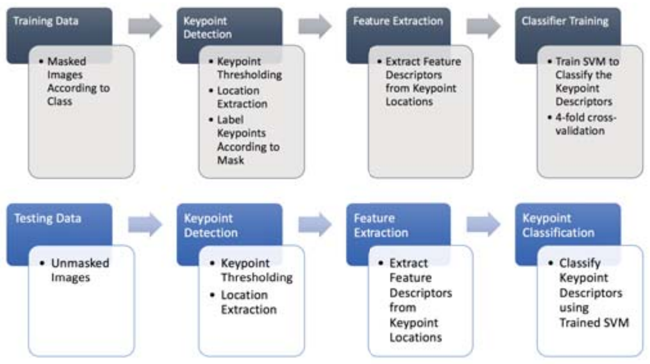

Training Data
• Masked Images According to Class
Keypoint Detection
• Keypoint Thresholding
• Location Extraction
• Label Keypoints According to Mask
Feature Extraction
• Extract Feature Descriptors from Keypoint Locations
Classifier Training
• Train SVM to Classify the Keypoint Descriptors
• 4-fold cross-validation
Testing Data
• Unmasked Images
Keypoint Detection
• Keypoint Thresholding
• Location Extraction
Feature Extraction
• Extract Feature Descriptors from Keypoint Locations
Keypoint Classification
• Classify Keypoint Descriptors using Trained SVM

Fig. 3: Feature Classification.

The core of our object classification process is using a Support Vector Machine (SVM) to label each feature as 'object' or 'non-object'. We found that using a simple threshold for labeling yielded poor performance, and we did not already have a model of our feature space, so we needed some machine learning method. Our dataset was relatively limited, so training an SVM model was the most apt method. The model takes the parameters returned by the detectors in OpenCV, as well as the following calculated parameters.

Objects in an image typically have multiple features associated with it, and as such we want the relative position information to serve as an input to our SVM model. We do not want the absolute position information to be an input, though, as an object may appear at any position in the image. So we calculate two parameters that correspond to feature density.

The first parameter describes any given feature's distance to the closest centroid. If there is a single object in an image, as our simulated dataset is set up to be, then one centroid would be associated more strongly with that object over the other centroid. Therefore, theoretically, features that have smaller distances to their nearest centroids would be more likely associated with an object. The pseudocode is outlined in Algorithm 1.

The second parameter is an implementation of Equation 21 in a paper presented by R. Diamant et al. [6]. This value quantifies the relative distance from a single point to all other points in the image: features that are near to many other features will have a lower value than features that are in sparse areas of an image. The MATLAB code is listed as CalcWeights in the appendix.

We quantify the performance of a feature detector by generating a Receiver Operating Characteristic (ROC) curve. The y-axis represents the total number of True Positives (features that were correctly labeled as 'object') and the x-axis represents the total number of False Positives (features that were incorrectly labeled as 'object'). A feature detector that performs well will

Feature Detector	Description	Pros	Cons
Harris [7]	Computes differential with respect to each direction. 5 parameters.	Distinguishes corners and edges well	Susceptible to scale variance
Shi-Tomasi [16]	Very similar to Harris but uses a simpler thresholding method for accepting or rejecting corners. 3 parameters.	Distinguishes corners and edges well	Susceptible to scale variance
STAR [11]	Based off of the CenSurE feature detector [1], multi-scale detector with no subsampling. 5 parameters.	Efficient, robust to viewpoint changes	Susceptible to brightness changes
FAST [14]	Computes difference in brightness based off neighbors in Bresenham circle. 3 parameters.	Efficient	Not robust to significant noise, susceptible to scale and illumination variance
SIFT [9]	Computes oriented gradient histograms for patches around a point. 4 parameters.	Rotation and scale invariant	Computationally expensive, susceptible to blur
SURF [4]	A more efficient approximation of SIFT. 3 parameters.	Faster than SIFT	Susceptible to viewpoint and illumination change
ORB [15]	Replacement for SIFT that builds off of the FAST detector. 6 parameters.	Scale and rotation invariant, good for real-time, resilient to noise	Generally fewer features

TABLE I: Comparison of feature detectors.

ALGORITHM 1
Distance to Nearest Centroid

```
$nClasses \leftarrow 2$
$kMeansIterations \leftarrow 15$
$centroids \leftarrow kMeansInitCentroids(features, nClasses)$
**for** $i = 1 : kMeansIterations$ **do**
    $idx \leftarrow findClosestCentroids(features, centroids)$
    $centroids \leftarrow computeCentroids(features, idx, nClasses)$
**end for**
**for** $p in features$ **do**
    $distCentroid1 \leftarrow norm(p - centroids(1))$
    $distCentroid2 \leftarrow norm(p - centroids(2))$
    **if** $distCentroid1 < distCentroid2$ **then**
        $distance(p) \leftarrow distCentroid1$
    **else**
        $distance(p) \leftarrow distCentroid2$
    **end if**
**end for**
```

generate an ROC curve that approaches near the upper-left corner (i.e. yields a high number of true positives and a low number of false positives).

IV. Analysis

Our dataset is comprised of 600 synthetic sonar images, which previous work has shown to be a good substitute when approaching feature and MLO detection [2]. It is split into three parts corresponding to the background texture used: sand, sand ripples, and grass, with 200 images in each. Within each of those three categories, 120 contain an MLO and 80 do not. Across all images, the intensities of the object, shadow, and background have been varied so as to mimic different levels of sonar quality. Each of the seven feature detectors were tested four times: once for each background category and once using all the images irrespective of background.

For each feature detector, four SVM models were trained on each dataset: Sand, Ripple, Grass, and All. The models were trained in accordance to the process described in the previous section, using a maximum of 200 features from each image, for an upper bound of 40,000 features as inputs. The detectors Harris, Shi-Tomasi, and FAST each had 3 parameters associated with each feature: the distance to nearest centroid and H value as described in the previous section, and intensity. The remaining detectors had 4 parameters in addition to those same 3: size, angle, octave, and ID, which are automatically computed by the detection step in OpenCV.

The model for each detector-dataset pair was used to predict whether each feature was an 'object' or not. The features that were labeled as 'object' were sorted according to their intensity, and a final threshold was applied. The threshold first included the strongest point, and the number of true positives and false positives was calculated, then it included the next strongest point, and true positives and false positives were calculated, until eventually all features were included. At each threshold, then, the number of true positives and the number of false positives was calculated. These numbers have been plotted as ROC curves in Tables II and III.

From Tables II and III we can see that the Harris, Shi-Tomasi, and FAST detectors performed well in the model that included the dataset containing all the images. However, the simple gradient-based approaches Harris and Shi-Tomasi had very wide variation across each individual dataset, which indicates that the models that are trained in particular parts of the environment do not transfer well to other parts. FAST, on the other hand, seems to generally perform well in all datasets. This can be attributed to its simplicity in outright rejecting regions that do not have consistent definition (as one would see in a line or point), like Harris and Shi-Tomasi, while also integrating a nonmaximum suppression step that eliminates spurious features corresponding to noise.

There are some detectors that perform particularly well in certain scenarios. SURF, for example, has the best performance in the Sand dataset, but average in the others, while ORB performed quite well in the Ripple dataset, in contrast to Harris, Shi-Tomasi, SIFT, and STAR, all of which yielded very poor performance in that particular dataset.

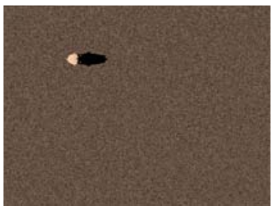
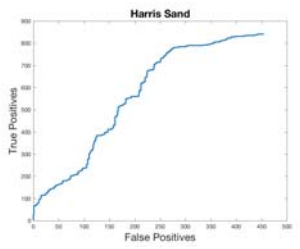
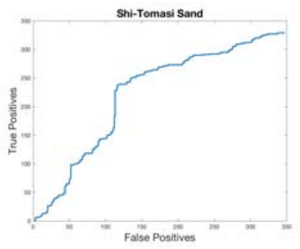
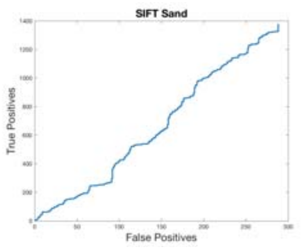
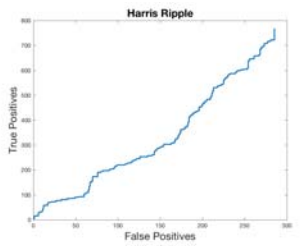
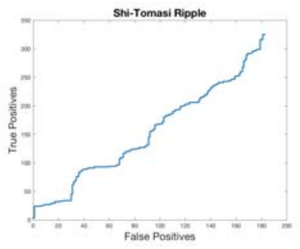
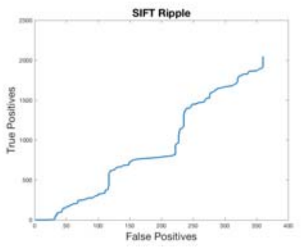
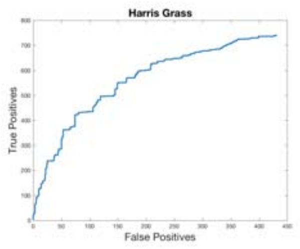
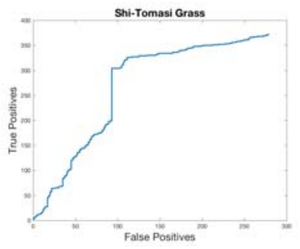
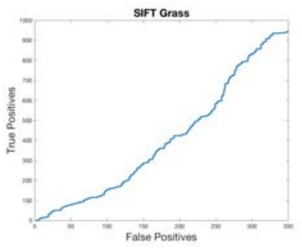
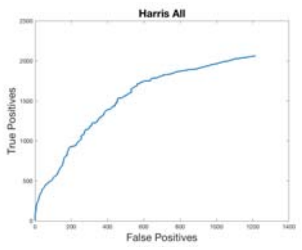
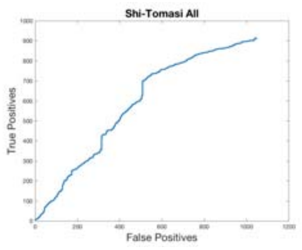
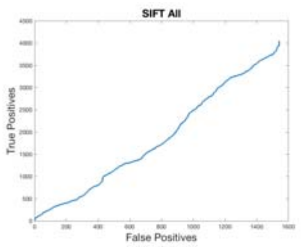
Dataset	Harris	Shi-Tomasi	SIFT
 Example Sand Image			
Example Sand Ripple Image			
Example Grass Image			
			

TABLE II: ROC curves

V. Conclusion

This paper presented a method by which to compare feature detectors on sonar imagery. There remain additional opportunities for comparison beyond the scope of this paper, however. In particular, this paper only presented an analysis on a simulated dataset, but, with care, a dataset comprised of real sonar images could be curated to develop additional dimensions of comparison, especially with regards to detecting multiple objects in a scene.

Also, these feature detectors were operated using fixed parameters. A method for parameter exploration could be developed to optimize the classification performance of each feature detector. In particular we may expect ORB to perform better with more optimal parameters, because it is fundamentally based on FAST, which yielded the highest performance on these datasets.

References

[1] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer, 2008.

[2] Christopher Barngrover, Ryan Kastner, and Serge Belongie. Semisynthetic versus real-world sonar training data for the classification of mine-like objects. *IEEE Journal of Oceanic Engineering*, 40(1):48–56, 2015.

[3] Christopher M Barngrover. *Automated Detection of Mine-Like Objects in Side Scan Sonar Imagery*. PhD thesis, UC San Diego, 2014.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Roee Diamant, Lars Michael Wolff, and Lutz Lampe. Location tracking of ocean-current-related underwater drifting nodes using doppler shift measurements. *IEEE Journal of Oceanic Engineering*, 40(4):887–902, 2015.

[7] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

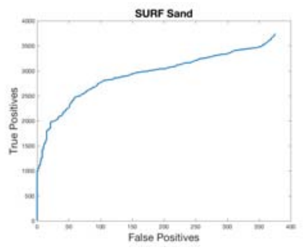
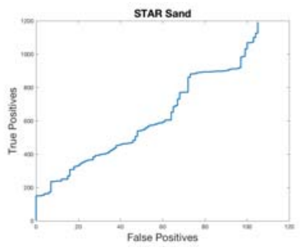
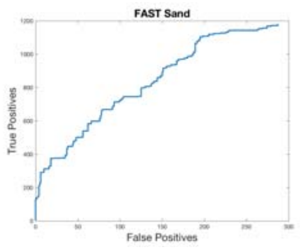
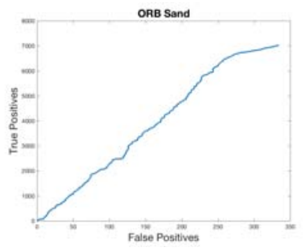
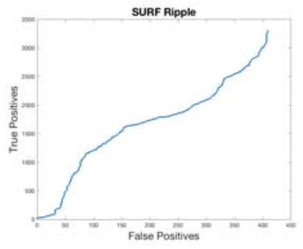
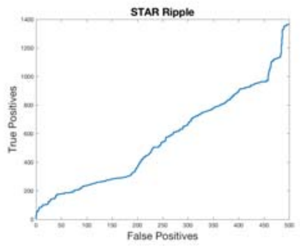
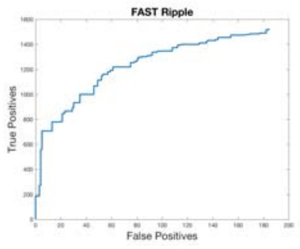
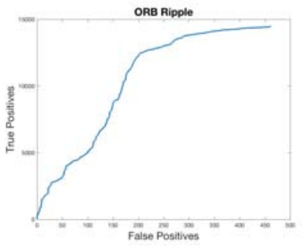
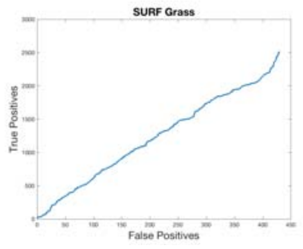
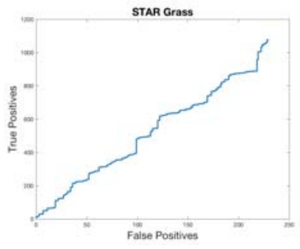
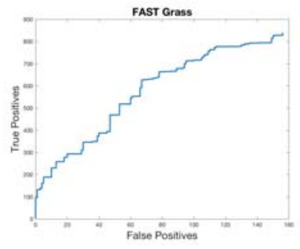
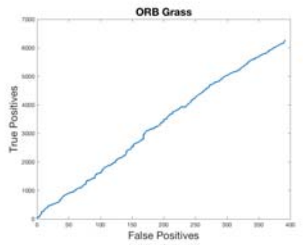
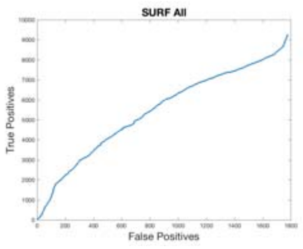
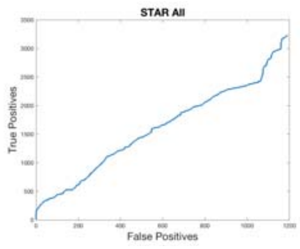
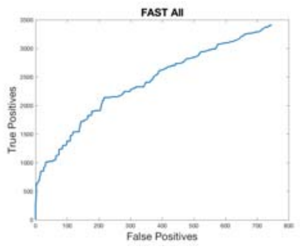
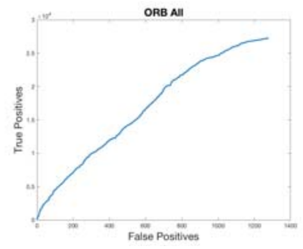
SURF	STAR	FAST	ORB
			
			
			
			

TABLE III: ROC curves continued

[8] Hordur Johannsson, Michael Kaess, Brendan Englot, Franz Hover, and John Leonard. Imaging sonar-aided navigation for autonomous underwater harbor surveillance. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4396–4403. IEEE, 2010.

[9] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[10] Paul Newman, John J Leonard, and Richard Rikoski. Towards constant-time slam on an autonomous underwater vehicle using synthetic aperture sonar. In *Proceedings of the eleventh international symposium on robotics research, Sienna, Italy*, 2003.

[11] OpenCV. STAR Class Description. https://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_feature_detectors.html#StarFeatureDetector%20:%20public%20FeatureDetector. [Online; accessed 18-April-2018].

[12] Scott Reed, Yvan Petillot, and Judith Bell. An automatic approach to the detection and extraction of mine features in sidescan sonar. *IEEE journal of oceanic engineering*, 28(1):90–105, 2003.

[13] David Ribas, Pere Ridao, Jose Neira, and Juan D Tardos. Slam using an imaging sonar for partially structured underwater environments. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5040–5045. IEEE, 2006.

[14] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[15] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[16] Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[17] Stefan Williams, Gamini Dissanayake, and Hugh Durrant-Whyte. Towards terrain-aided navigation for underwater robotics. *Advanced Robotics*, 15(5):533–549, 2001.

APPENDIX

```
function h_n = CalcWeights(NodeLoc, x_bar)

%NodeLoc is every single location of the features
%x_bar a list containing the Euclidean distance from every feature to a
%consistent location in the image.

DistanceMeasure = 1./x_bar;

num_trusted_nodes = length(x_bar);

h_n = zeros(1, num_trusted_nodes);
for nRx = 1 : num_trusted_nodes
    h_nk = zeros(1, num_trusted_nodes);
    for kRx = 1 : num_trusted_nodes
        if kRx ~= nRx
            SUMMER1 = 0;
            SUMMER2 = 0;
            for jRx = 1 : num_trusted_nodes
                if (jRx ~= nRx) && (jRx ~= kRx)
                    Temp = 1/DistanceMeasure(jRx)* ...
                        (1-(NodeLoc(kRx,:)-NodeLoc(nRx,:))...
                        * (NodeLoc(jRx,:)-NodeLoc(nRx,:))'/...
                        (DistanceMeasure(nRx)*DistanceMeasure(kRx)));
                    SUMMER1 = SUMMER1 + Temp;
                    SUMMER2 = SUMMER2 + DistanceMeasure(jRx);
                end
            end
            h_nk(kRx) = SUMMER1 / SUMMER2;
        end
    end
    h_n(nRx) = sum(h_nk);
end
```