Real-time side scan image generation and registration framework for AUV route following

Peter King*, Andrew Vardy†‡, Peter Vandrish‡ and Benjamin Anstey‡

*Marine Environmental Research Lab for Intelligent Vehicles
†Department of Computer Science
‡Faculty of Engineering and Applied Science

Memorial University of Newfoundland
St. John's, Canada
peter.king@mun.ca

***Abstract*—Memorial University is in the development stages of a Qualitative Navigation System (QNS) to be deployed on the Memorial Explorer AUV. This system will allow localization and path following along a trained route without the necessity of a globally referenced position estimate. Previous QNS work has been on terrestrial robots using optical images. Our main challenge lies in utilization of side scan sonar as the imaging medium, as this type of sonar is prevalent on AUVs and provides much better range and coverage than optics in water. To achieve this, a sonar image processing and registration framework has been developed. To be useful such a framework should be fully-autonomous, robust, and operate in real-time, where real-time operation is defined as the ability to process, register and localize data at the rate it is collected, or faster.**

In this paper we describe our framework for processing sonar data, generating image tiles, extracting unique features and localizing against a reference set. We also present some results of this system based on raw sonar input data collected by the AUV.

I. Introduction

The framework described in this paper is a set of software modules which interface to an autonomous underwater vehicle (AUV) and an integrated side scan sonar. The task of the framework is to allow real-time collection of side scan data, generation of geographically corrected two dimensional image tiles, and localization of a given tile to a database of previously collected tiles. This framework has been developed to support Memorial University's ongoing effort to implement a qualitative navigation system (QNS) for its Explorer AUV. Qualitative navigation is a vision based navigation strategy, which side-steps the problems incurred by long term position and orientation drift. The QNS employs this framework to generate an estimate of the vehicle's progress along a previously trained route and its orientation and position with respect to the route.

A. Background

This framework supports an algorithm for the registration of side scan sonar images, which is at the core of our autonomous route following strategy [1]. The approach taken is to represent the route as a sequence of nodes, each with an associated side scan sonar image which was collected during training. This strategy is based on the notion of topological navigation pioneered in mobile robotics [2] which has also been described as *qualitative navigation* [3], [4]. The approach of qualitative navigation is to represent the environment as a set of connected places with mechanisms for travelling between places. It is quite explicit that these places are *not* represented in the same global coordinate system. These ideas have led to considerable success in recent years in allowing an outdoor mobile robot to autonomously follow a trained route, despite changes in illumination and variable terrain [4], [5]. To our knowledge, our work represents the first attempt to apply a qualitative navigation approach to the underwater domain.

II. Framework

The framework described here consists of a robust set of tools to be deployed on an AUV and run *in-situ*. To date this framework is still in development, but has reached sufficient maturity to have been successfully deployed on Memorial's Explorer AUV as of July 2012.

A. Constraints

Design of the framework was bounded by several constraints to ensure the goals of the QNS system were maintained. Most critically the framework assumes no dependence on the vehicle's current global position estimate. This is required to allow navigation in the presence of a degraded or lost position estimate. The goal of QNS is to provide its own relative position estimate along a trained route. Sonar data conversion, processing and localization aspects of the framework do not rely on any filtered or estimated AUV position in the global coordinate frame and only utilize instantaneous measurements from the AUVs sensors. Instantaneous measurements will have a static error not growing over time. For this work heading, altitude and speed were employed as the Explorer AUV can provide these values from direct measurement.

As this system is intended for autonomous operation the framework assumes no input from an outside user beyond initial configuration and tuning. Once deployed information is

exchanged with no entity other than the AUV and its sensor payload. This is especially important in the determination of best match of the current location tile to those of the trained route.The QNS system is solely responsible to use its best match to aid in updating the overall estimate of progress along the route.

The framework should provide a localization result in *real-time*. Once a tile of side scan data has been collected the image generation, feature extraction, tile matching, localization, and best estimate will occur in a time frame less than that of the collection of the next tile. To put it bluntly, processing should take less time than collection. This will ensure that the estimate is always for the most recent tile and that the delay does not grow as more tiles are collected.

B. Architecture

The design of the high-level architecture and division of modules is based on the required flow of data and the intermediate products that are necessary. There is a natural linear flow of data from one stage to another. The stages we have identified are:

Sonar collection
: Raw sonar data is collected and stored to the hard disk. This is handled by the AUVs payload system and is not considered part of this framework. It should be noted that this collection happens independently and in parallel to the QNS.

Sonar data to image conversion
: Raw side scan files need to be processed and converted to images from which key-points are extracted.

Localization of tiles
: Tiles are compared against those in the training set and a decision is made as to the most likely match. From this selected match the AUV can be localized along the training path.

Storage and reference set matching
: When training we need to build a database of tiles and key-points along the route. When route following we need to poll this data set and generate information on how the tiles match.

Starting this process is the raw sonar data which is continuously being collected by the AUV. As raw data files are created with a set number of pings the QNS system notifies the framework and the the processing begins. Figure 1 shows the data flow and intermediate data products. Where:

(1) Raw data from the sonar is written to the manufacturers specific file format (*e.g.* JSF for MUNs EdgeTech system)
(2) As each file is complete they are read into the sonar processor.
(3) Sonar data is converted to a geographically corrected image tile and can be optionally written to the disk.
(4) Path tiles containing the extracted key-points are passed to the localizer.

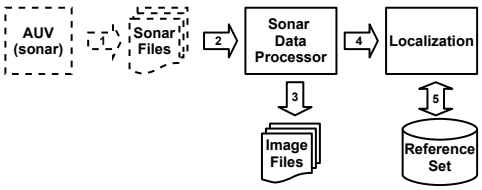


Fig. 1. Dataflow of the QNS framework

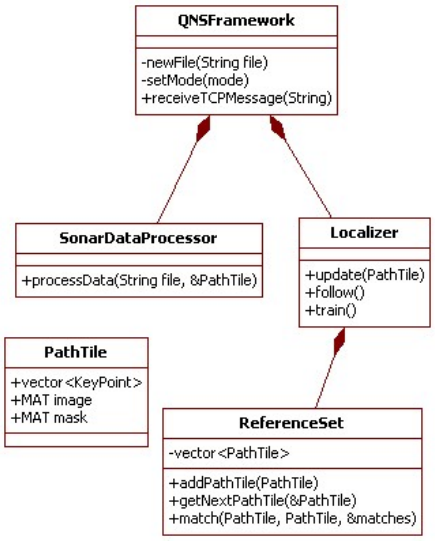


Fig. 2. High-level architecture

(5) Path tiles are sent to the reference set where they are either added to the training set or matched against the existing training set. Information about the matches are passed back to the localizer.

Figure 2 outlines the high-level architecture and the modules employed. This diagram in for illustrative purposes only and does not fully represent actual implementation details.

III. Modules

The following is a description of each of the software modules, their responsibilities, data products they accept and generate as well as any pertinent implementation details. Internal modules or structures are *italicized*.

A. SonarDataProcessor

The *SonarDataProcessor* module is responsible for reading each new raw side scan data file and producing a *PathTile*. A *PathTile* consists of: a two dimensional image representation of the sonar tile, a set of key-points that have been extracted from the image using a selected key-point extractor and an image mask which is used to ignore those regions of the image not displaying the sea bottom (*e.g.* the nadir).

Upon notification of a new sonar file the *SonarDataProcessor* reads each ping, or scan-line, of data and maps each sample to an image (x, y) coordinate. Using the AUVs heading

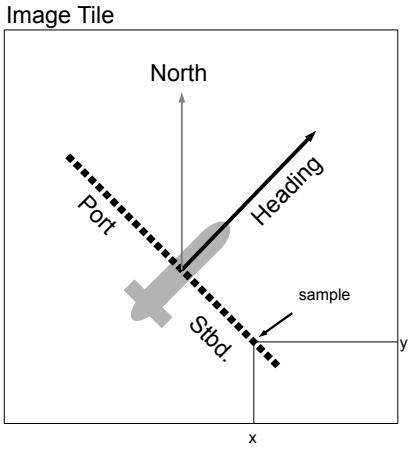


Fig. 3. Mapping of samples to image coordinates

and speed the centre coordinate of the scan-line is calculated as:

$$x_p = x_{p-1} * dt * s * \cos(h) \quad (1)$$

$$y_p = y_{p-1} * dt * s * \sin(h) \quad (2)$$

where x and y are the image coordinates, with subscript indicating current or previous ping, dt is the time step between pings in seconds, s is the vehicle speed in m/s and h is the heading in Radians from North. Sample positions are determined relative the ping's centre coordinate by:

$$x_{s_{port}} = \frac{x_p + \sqrt{r^2 - a^2} * \cos(h - \frac{\pi}{2})}{res} \quad (3)$$

$$y_{s_{port}} = \frac{y_p + \sqrt{r^2 - a^2} * \sin(h - \frac{\pi}{2})}{res} \quad (4)$$

$$x_{s_{stbd}} = \frac{x_p + \sqrt{r^2 - a^2} * \cos(h + \frac{\pi}{2})}{res} \quad (5)$$

$$y_{s_{stbd}} = \frac{y_p + \sqrt{r^2 - a^2} * \sin(h + \frac{\pi}{2})}{res} \quad (6)$$

where $x_{s_{port}}, y_{s_{port}}, x_{s_{stbd}}, y_{s_{stbd}}$ are the resultant sample coordinates for the port and starboard channels, r is the slant range in meters, a is the vehicle altitude, h is heading in degrees from North, and res is the image resolution in metres-per-pixel. Figure 3 shows how sample coordinates relate to the image coordinate system. These equations are derived with the assumption of a flat sea bottom and resolution is a system parameter set by the user. For debugging purposes the framework can write the produced image to the hard drive, though for space and performance this can be disabled.

The plotting of each sample relative to the vehicle position and to each ping creates an image in which samples are correctly located with respect to each other and the position on the seafloor from which they were extracted. The number of pings included in each tile is selected by the user. More pings in a tile can allow more opportunity for feature extraction, but will lead to tiles that cover larger areas of the seafloor, thus reducing the resolution of the entire QNS system. For most of this work we have utilized 1001 pings in each tile, where each tile represents 44 metres of travel. Without correction features in the tiles would appear with respect to the vantage point of the vehicle. Items nearer or further away would be skewed, hindering detection of similar areas in co-incident tiles taken from different views. Tiles are all oriented with respect to North; This removes rotation from tile matching.

To account for physical properties of the sonar two levels of gain are applied to the sample magnitude: a scalar value to account for weak or strong signals and a time-varying gain. As sonar signals spread out over distance their intensity at any given point along the wave front is reduced [7]. To correct for this a gain factor relative to the distance travelled is applied to ensure a smooth gain over the horizontal width of the scan-line. For our work this value is selected by analysing the training data and finding a best value, through visual assessment, for which intensity is uniform across the scanlines and the overall image is bright, but not blown out.

With a image representation of the tile key-point extraction begins. The *OpenCV 2D Feature framework* [6] offers a variety of feature extraction algorithms. The QNS framework supports any of the algorithms as they offer a common interface. The chosen algorithm is set through the user configuration. For most of this work SURF and SIFT were used. This decision is based on our previous work [1]. Although outside the scope of this paper a more in-depth discussion of these algorithms is provided by: [8], [9].

It should be noted is that it has been found in our research that the grainy nature of sonar data leads to a multitude of 'small' key-points based on the noisy aspects of the data. Through experimentation blurring the tile image has been found useful to allow the key-point extraction to focus on larger regional key-points. Due to the introduction of blurring there is no need to interpolate images to fill gaps in the samples as the blurring naturally fills these holes. Though the images are not visually accurate, they do perform better in terms of the key-point extraction methods employed in this work.

As side-scan sonars look outward laterally, there is an area of non-coverage directly below the sonar known as the nadir. This can be seen in Figure 6(a) as the dark center band. Also as we orient the samples in relation to the north axis there will be image areas with no coverage. In Figure 6(b) this is the black region extending around the perimeter and most notable in the corners. A mask is constructed by applying a binary threshold to the image to sharpen contours and to reduce bleeding into good and usable areas of the original image. *OpenCV* functions are then used to then generate contours of the image and to flood fill areas not considered part of the data. This area becomes the mask region we ignore and is included in the *PathTile* structure.

B. Localization

As *PathTiles* are generated they are passed to the *Localization* module. This module operates in two modes: training and following. In the training mode *PathTiles* are simply inserted into the *ReferenceSet*, which encapsulates a database

of *PathTiles* collected along the trained route. In following mode the localizer takes the *PathTile* and tries to register it to an existing tile in the *ReferenceSet*. Registration is the process of comparing one *PathTile* to a set of others to determine which, if any, matches best. For each pair the matches are analysed and a quality of match is determined. A best case scenario is that there is a strong match to a single reference tile, otherwise there may be candidate matches. The decision of the best match(s) is provided to the qualitative navigation engine for positioning and generation of a navigation suggestion. Qualitative navigation utilizes a Bayes filter to maintain and update and estimate of most likely position along the trained route and can inform the AUV of a correction to maintain traversal along the path. The operation of the complete QNS for navigation is explained further in [10].

The main challenge of the *Localization* module is the systematic determination of quality-of-match between two tiles. As image tiles in this framework are oriented with respect to vehicle heading and are corrected for altitude the only transformation between them is lateral, *i.e.* there will be no variance in scale or rotation. Thus, matches can be statistically analysed based on variation in the slope between matched points and their absolute distance. A strong match being one where key-point pairs are offset by a consistent slope angle and by a consistent distance, or in other terms where there is general agreement in the vectors between coordinates of the key-point pairs. Figure 5 shows an example of two image tiles for which there is a strong match. As can be seen the vectors between key-points are parallel and are of similar length.

C. ReferenceSet

The *ReferenceSet* serves two roles: it encapsulates the trained route, allowing insertion and retrieval of tiles, and it generates matching pairs between a query *PathTile* and a reference *PathTile*. In its simplest form the trained route is a vector of *PathTiles* inserted in order of collection. When a query *PathTile* is presented the *ReferenceSet*'s matcher will generate a list of matching key-points for each possible pair of Tiles. As an estimate of position along the route is established the search space of the reference set can be reduced to an area of greater likelihood.

The matching is taken from the *OpenCV 2D feature framework*. This matcher selects likely matches between two sets of image key-points extracted from the method described in the Localization module. Currently OpenCV supports a FLANN based matcher, which selects nearest neighbour matches, and a BruteForce matcher, which as its name implies searches the entire key-point space for matching pairs [6].

IV. Results

The results presented here are taken from tests of the framework operating on data collected by the AUV during July 2012 field trials in Holyrood, Newfoundland and Labrador, Canada. During these trials the QNS framework was deployed on the vehicle and was run *in-situ* during data collection. Off-line tests were performed on an Intel E2140 dual core PC, operating at 1.6GHz, 4GB of RAM and running Ubuntu 10.04 On-line tests were run the AUVs payload computer, a Compact PCI Intel i7-2655LE computer operating at 2.2GHz, 4GB of RAM and running Ubuntu 12.04.

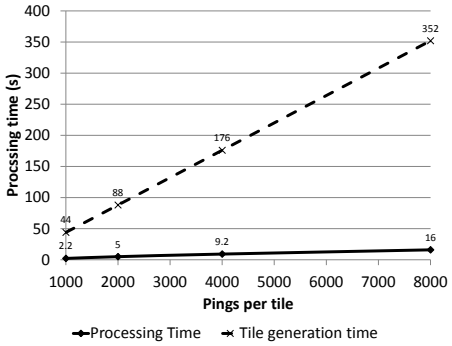


Fig. 4. Processing time versus number of pings in tile

A. Real-time performance

Real-time performance of this framework is paramount to ensure a successful implementation of qualitative navigation. As tiles are converted and matched a logging system records the beginning and completion of each phase. These time stamps are analysed to ensure the framework operates at a rate greater than the tile generation rate. Figure 4 plots processing and tile generation time for various tile sizes. Tile size in this case is defined as the number of pings a tile encapsulates. Since the ping rate is fixed the tile generation time grows linearly with size. As the figure shows, much more time is consumed by tile generation than by subsequent processing. The plotted results are from off-line testing where the processing time for a 1001 ping tile is 2.2 seconds.

Figure 5 plots average matching time and total running time over a varying size of *ReferenceSet*, where size is the number of tiles included in the search space. Calculation of the total running time is taken as the time to match a tile and the time to process the current query tile, in this case a tile size of 1001 is assumed with a static processing time of 2.2 seconds. To put the *ReferenceSet* search space size in context a tile of 1001 pings in our set-up would represent 44m of distance travelled; For our experiments the AUV travelled at $1m/s$. Searching 1000 tiles in the *ReferenceSet* would thus represent path length of 4.4km.

B. Tile generation

The framework is very reliable in converting raw sonar files into image tiles in all cases. Figure 6 shows the progression of a tile as it is processed.

C. Tile registration

The main task of this framework is to reliably and autonomously match one sonar image to another within a set. As described in Section III-B this involves a process of extracting a set of image key-points, deriving matches from this set to

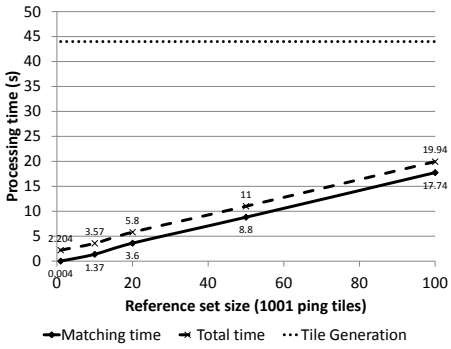
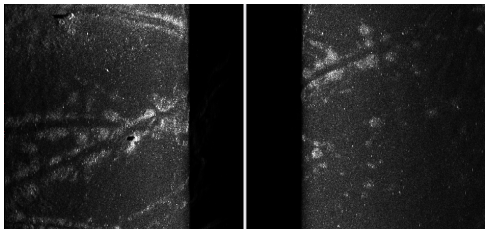
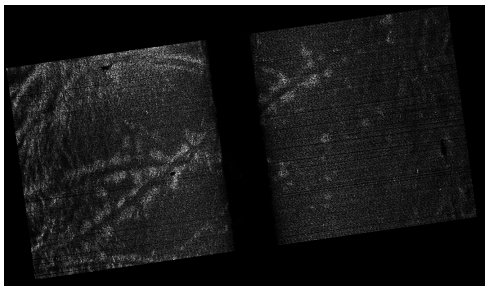


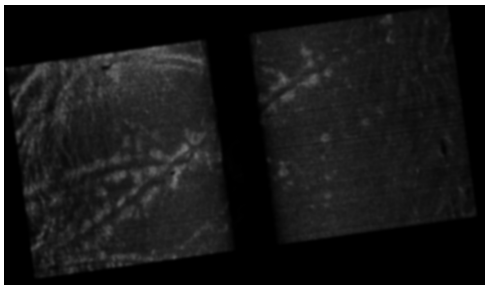
Fig. 5. Matching time versus number of tiles in ReferenceSet



(a) Uncorrected tile of sonar data



(b) Tile adjusted with speed, heading, altitude and gain



(c) Tile blurred to improve key-point extraction

Fig. 6. Progression of tile generation

TABLE I
MATCH RESULTS FOR TEST RUN

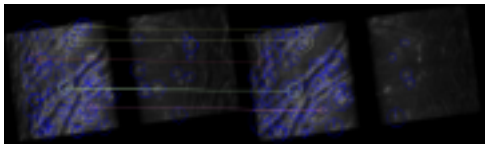
Query tile	Reference tile	Slope variance	Distance variance	Localized	Actual
1	1	0.84	80.57	y	y
2	2	0.005	1932	n	y
3	2	18.9	9049	n	n
3	3	21.8	6695	n	y
3	5	2.63	1843	n	n
4	2	7.8	6099	n	n
4	4	0.002	4.6	y	y
5	3	1.74	33482	n	n
5	5	0.0023	2.4	y	y
6	6	2.26	16043	y	y

a set of previously collected key-point sets, and making a decision as to which of the collected sets match best. The following images are taken from a section of two offset lines, with one serving as the reference set and the other the query set. Tile size is 1001 pings-per-tile and match decisions are based on the variance of the slope angle between matched key-point coordinates as well as the variance in vector distance between them. Matches are first filtered by aspects of the matching key-points. In this case SURF key-points are used which have a related angle and size; Matches are filtered to ensure variations between these quantities are below thresholds of 10^o for the angle and 30 pixels for size.

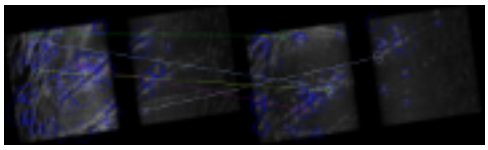
Figure 7(a) shows an example of a very strong match from this set. The displayed lines connect the matched key-points from one image to another. These lines are approximately parallel and of similar length. The framework calculated a variance of 0.847 in the match angles and 80.57 in distance. For comparison, two tiles that do not match well, like those shown in Figure 7(b), have variances of 18.89 and 9048 for angle and distance respectfully. Over the complete run of this short section of 7 match tiles and 7 reference tiles and decision thresholds of 5 for angle variance and 100 for distance threshold the framework successfully matches 4 of 7 tiles, with two false negatives and no false positives. Table I outlines the results, where the pairs of query and reference tiles are those pairs for which matches were found, the localized column indicates those matches selected by the *Localization* module and the actual column indicating the true matching based on actual position.

D. AUV deployment

During July 2012 trials a version of the QNS framework was deployed on the AUV. This test was to verify the framework , respond to new files produced by the AUV, perform processing and matching steps and switch between modes of operation. For these aspects the tests were a complete success. It should be noted; however, that due to a lack of collected data the framework was not tuned for the environment and though was able to produce image tiles and extract key-points, it did not make successful matches. There were hopes to tune the



(a) A strong tile match.



(b) A weak tile match.

Fig. 7. Comparison of good and bad matches

system in the field, but days were lost to various hardware issues and tuning did not occur. We feel the off-line tests more than validate this framework as the computational aspects are identical whether off-line or on-line. The operational aspects that were tested add to this confidence.

V. Conclusion

Though in its initial version the real-time sonar image generation and registration framework developed by Memorial University has proved to be a useful developmental exercise. Confidence is high that future performance will be sufficient to support the successful development of fully operational qualitative navigation system that will provide autonomous control of the Explorer AUV and allow long term path following over a trained route. Many lessons have been learned in this development; most importantly the need for a large data set to tune the system to a particular environment. Parameters such as sample gains, image blur levels, key-point extraction parameters and key-point match filtering thresholds can greatly affect the operation of the framework. Optimal values may only become apparent though off-line trials on collected data. The system is quite good at moving from sonar to image tile and generating many interesting key-points; the main challenge at this point is in the selection of relevant key-point pairs and the determination of overall match. From the data we have collected thus far we have shown that cases exist for which this framework can provide acceptable levels of matching.

A. Future considerations

Several considerations have become apparent that will help drive subsequent versions of the framework. These changes will allow the framework to become more efficient and robust. Currently the image data is retained and stored in the *ReferenceSet* along with key-point data, this may be unnecessary. In its current form once a tile has been added only the key-points are used in matching, requiring no further data from the image. Removal of the image data would vastly reduce the memory footprint of framework. For cases where the *ReferenceSet* does grow beyond the capacity of memory there should be an addition of a more elegant database back-end to allow efficient access of the *ReferenceSet* from the hard-disk. Currently all entries are stored in RAM, which is fine for smaller training sets, but will outgrow memory capacity as longer paths are attempted. To allow more versatility and robustness in changing environments, or paths that span multiple bottom types, some level of dynamic tuning of parameters should be investigated. This would involve auto-adjusting gains, blur levels and key-point filtering based on feedback from the bottom images or raw data. The selection of good matches and discarding of bad match pairs will need continued refinement. The current method of a threshold on the general agreement of slope angle and vector distance through the variance does work in some cases. What is problematic is outlier matches which skew this agreement. A more sophisticated method of determining match agreement is needed.

Acknowledgment

The work presented in this paper was funded by the Atlantic Canada Opportunities Agency, through an Atlantic Innovation Fund grant, under the Responsive AUV Localization and Mapping project (REALM), and Research and Development Corporation of Newfoundland and Labrador.

References

[1] P. Vandrish, A. Vardy, D. Walker, and O. Dobre, "Side-scan sonar image registration for AUV navigation," in *IEEE Symposium on Underwater Technology*, 2011.

[2] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Journal of Robotics and Autonomous Systems*, vol. 8, pp. 47–63, 1991.

[3] D. Dai and D. Lawton, "Range-free qualitative navigation," in *IEEE ICRA*, 1993.

[4] Z. Chen and S. Birchfield, "Qualitative vision-based path following," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 749–754, 2009.

[5] A. Zhang and L. Kleeman, "Robust appearance based visual route following for navigation in large-scale outdoor environments," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 331–356, 2009.

[6] G. Bradski. (2012, July) Opencv wiki. [Online]. Available: http://opencv.willowgarage.com

[7] L. Brekhovskikh and Y. Lysanov, *Fundamentals of Ocean Acoustics*. Springer-Verlag, 1982.

[8] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[9] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[10] P. Vandrish, A. Vardy, and P.King, "Towards auv route following using qualitative navigation," in *Conference on Computer and Robot Vision*, 2012.