Navigation using 3D features from side-scan sonar data for a deep-sea AUV

Philipp Woock
Vision and Fusion Laboratory
Institute for Anthropomatics
Karlsruhe Institute of Technology (KIT), Germany
philipp.woock@kit.edu

Technical Report IES-2011-03

Abstract:
Simultaneous navigation and mapping (SLAM) of an autonomous underwater vehicle (AUV) based on side-scan sonar data has significant peculiarities that make standard SLAM techniques inapplicable. In particular, recognition of already visited places (loop closure) which is an important tool improving the navigation accuracy cannot be done with the raw sonar data. This and other navigation tasks can be more conveniently performed based on the 3D seafloor shape (elevation map). In this report we first present an extension of a well-known sonar data reconstruction method to account for nontrivial AUV motion, and then discuss various algorithms useful in the context of SLAM processing elevation data.

1 Introduction

Mankind is in need of more and more natural resources so that areas of easily obtainable raw materials on land are mostly explored. In the oceans and especially in the deep-sea huge amounts of raw materials (metals, gas hydrates) can be found. Unfortunately, knowledge about the deep sea is still scarce. Autonomous underwater vehicles (AUVs) are a way to acquire seafloor maps automatically with comparably low effort. However, The quality of the map is strongly dependent on the navigation capabilities of an AUV. The latter can be improved by employing Simultaneous Localization and Mapping (SLAM) methods.

Our goal is to establish whether sonar-based SLAM may serve as a reliable navigation tool for vehicles operating autonomously for several hours.
In all SLAM methods, re-visiting a location and detecting it as such (loop closure) is used to limit the positional error growth in the current motion and map estimates. Given that the data association at the loop closure is correct, the growth of motion and map errors is limited. The aiding sensor in this case is an imaging side-scan sonar and re-visits are to be found based on that sonar data.

The outline of a workflow how this can be achieved is given in Figure 1.1.

### 2 Side-scan sonar data processing

Side-scan sonar data is recorded as echo amplitude over time, which poses certain challenges interpreting the data spatially: For example, it leads to an effect called ‘sonar shadow’: When a certain area is ensonified by an AUV for a second time upon a re-visit, it is very likely that the traveling direction of the AUV is different from the first visit. That in turn leads to sonar shadows cast differently and simple image matching techniques are unlikely to succeed as the same surface may have a varied appearance when viewed from a different angle. For a detailed summary of side-scan sonar signal interpretation and the related navigation issues see [Woo10], [WF10], and [Woo11b].

In addition to that, side-scan sonar data is usually processed by stacking the data lines on top of each other to form a sonar image. For an AUV that does not move in a straight line, this creates distortions in the image accruing from the resulting irregular sampling of the seafloor. Those distortions make image matching even more difficult.

In addition to the inherent ambiguity of sonar data, the motion and the rotation of AUV lead to irregular seafloor sampling. Since our prototype vehicle built in
the joint Fraunhofer project TIETeK\footnote{http://www.tietek.de} is equipped with an inertial measurement unit (IMU), we incorporate the ego-motion information to compensate for these distortions.

This paper first recaps an existing side-scan sonar data inversion technique for stacked sonar data lines. Then, this method is extended to take non-straight non-uniform vehicle motion into account. In the following section various 3D-features are presented and their underlying principles are discussed.

2.1 Sonar inversion

The literature on 3D geometry reconstruction from side-scan sonar data is rather scarce. To the best of our knowledge, the method of Coiras et al. [CPL07] describes the most relevant method for 3D geometry reconstruction using side-scan sonar data. Using a 2D sonar image created from stacked sonar lines, they try to iteratively estimate the surface model parameters for each pixel individually that would yield the recorded sonar echo. The model parameters are restricted to surface shape \( Z(x, y) \), reflectivity (mostly depending on sediment type) and the sensors’ antenna beam form. Regularization constraints for the reflectivity and beam form are imposed by means of the full 2D image.

A strong sonar response may therefore originate from either the surface being inclined towards the sensor, the surface consisting of well-reflecting materials, or a strong ensonification by the sensor. The surface inclination with respect to the sensor is calculated for each surface patch by the angle between surface normal and the line of sight between the sensor and the given surface patch.

The estimation process starts with a flat surface and the original sonar image as reflectivity map as well as a homogeneous antenna beam form and then creates a virtual measurement given these parameters. The difference between the recorded measurement and the virtual measurement shows where the model does not yet match the environment. Parameters are adjusted accordingly and a new virtual measurement is generated. This is done until the virtual measurement is close enough to the recorded measurement (see Figure 2.1).

To obtain a reasonable parameter estimation from the model, it is necessary to restrict the freedom of the model by so-called regularization. These regularization constraints employ additional knowledge about the parameters and enforce them in the estimation process, e.g., it is unlikely that the sediment type changes extremely frequently on a small patch of ground. It can also be assumed that the beam form of the sensor changes only very slowly throughout the duration of a mission. An
Figure 2.1: Iterative estimation process of the seabed surface shape.

assumption also made in [CPL07] is that in shadowed areas the surface reflectivity is not different from the adjacent non-shadowed parts. To establish this constraint, a separate detection of shadowed areas is used. Additional constraints can also be imposed on the surface shape (e.g., regarding surface smoothness), however those constraints need to be carefully chosen to still allow the necessary variation to represent the environment correctly while at the same time the estimation of any implausible parameter sets is avoided, like for example an overly spiky seafloor shape.

The estimation of the parameters is done in a hierarchical fashion using parameter pyramids containing subsampled versions of the parameter sets. This helps to avoid local minima in the parameter estimation process while at the same time the subsampling employed to obtain the lower resolution stages also aids the convergence stability as noise is mainly smoothed out.

2.2 Related work

Further work of this group [CG09] was aimed mainly at SAS (synthetic aperture sonar) systems and due to its cylindrical coordinate system it was not that suitable for doing SLAM in a cartesian framework. However, the idea of treating the seafloor as deformable mesh is worth considering also in the cartesian case.

The method of Bikonis [BSM08] is using a shape-from-shading approach but is relying on proper shadow zone detection, which can be unreliable. Other research on SLAM with side-scan sonar sensors is treating the sonar data as an image with
3 Sonar inversion for a moving vehicle

The method described in 2.1 implicitly makes an assumption that is not necessarily true: By stacking the individual lines to an image, a neighborhood relationship in the resulting pixel grid is introduced among different sonar scan lines. Such a relationship is only true for an AUV that is going straight ahead at a constant speed. If the vehicle performs another movement, like a turn for example, the sonar data recorded in this turn does not stem from surface parts directly adjacent to the parts ensonified by the previous sonar measurement. This is the reason why distortion is introduced to a map by simple stacking of sonar lines. Using the neighboring pixels for a query in the stacked representation means that values are used which are not necessarily closest to the query pixel in reality. Additionally, the neighborhood order could be flipped upside down on a vehicle turn. Figures 3.1 and 3.2 illustrate this problem. The currently processed sample is colored yellow. The red sample, which is a top neighbor in the pixel grid, is a bottom neighbor in reality. Additionally, neither the red or the green sample are the closest neighbors like the pixel grid suggests. The magenta colored samples are the closest ones in reality.

3.1 Inversion in consideration of true spatial point neighborhoods

The inversion method therefore needs to be extended to treat non-straight non-uniform vehicle motion and the resulting irregular sampling of the seabed correctly. To accomplish this, the vehicle’s IMU (inertial measurement unit) data is used to correctly reference the sonar scan lines to each other. With that information true neighborhood information can be obtained.
Figure 3.2: Nearest neighbors on a curved trajectory. The nearest neighbors from the pixel grid are wrongly ordered in reality and are not necessarily nearest.

The extension is quite straightforward and does not change the basic working principle of the method: The original method uses surface gradients $\frac{\partial Z}{\partial x}$ and $\frac{\partial Z}{\partial y}$ in the estimation of echo intensity. The local coordinate frame for each scan line is $x$ in across track direction, $y$ is along track direction and $z$ is pointing up. Therefore, the gradient $\frac{\partial Z}{\partial x}$ is estimated within each scan line whereas the gradient $\frac{\partial Z}{\partial y}$ is taken across different scan lines in the pixel grid of the sonar image. This is where the extension has its main difference: The $\frac{\partial Z}{\partial y}$ gradient is estimated from surface points in the true seafloor surface neighborhood.

The extension initializes the surface as flat lines, each line at the depth of the detected first bottom return (FBR) (see [WF10] for further reading about the FBR) and inserts the estimation scan lines according to the vehicle position and orientation into a point cloud. Depending on the vehicle pitch motion, the initial estimated line is placed not exactly beneath the vehicle but according to the lever arm consisting of the pitch angle and the altitude (see Figure 3.3). Vehicle roll motion is more difficult to handle as this affects the sonar propagation direction (see Figure 3.4). There may be areas right beneath the vehicle where multiple echoes are received (so-called layover). When this information is discarded and the roll angle increases, the angles necessary for calculation of ground-range correction ($\phi$ and $\phi'$) deviate more from 90°. More detailed reading about the changes to ground range correction can be found in [Woo11a].
Figure 3.3: Vehicle pitch angle.

Figure 3.4: Sonar sensor beams without roll motion. $r_s$ denotes slant range coordinates, $r_g$ denotes ground range coordinates, $\gamma$ denotes the mounting angle of the side-scan sonar sensor and $b$ denotes the first bottom return (FBR).
For each point the $x$-gradient is calculated just like in the unmodified case by finite differences within the same scan line. This is done because the nearest neighbors in the local $x$ direction almost always lie within the line independent of how the line is oriented in space. The $y$-gradient is calculated by estimating a surface normal out of the nearest $k$ neighbors that do not belong to the same line (see Figure 3.6). The reasoning behind that decision is that by using simply all the nearest neighbors, by far the most neighbors would stem from the same sonar line where the query point lies due to the sampling density within a sonar line being much higher than between lines. However, it may still happen that all $k$ nearest neighbors stem from only one of the other lines.

The second reason is that the immediate neighbors from the same line already contribute to the $x$-gradient and would be counted twice. Figure 3.7 shows that choosing a fixed radius instead of $k$-nearest neighbors is worse due to the varying sample density where $k$-nearest neighbors algorithm inherently is able to adapt to.

Depending on how many neighbors are used for the normal vector calculation a certain smoothing effect is observed. Using only relatively few neighbors, the estimated surface normal is relatively unstable as it is strongly influenced by outliers, whereas using too many neighboring points for the estimation may smooth out surface details.

The estimated surface normal vector needs then be projected onto a plane perpendicular to the query line in order to remove the gradient component in $x$ direction.
Figure 3.6: Choosing $k$ nearest neighbors can cope with different sampling densities. The yellow marked samples would indicate the nearest neighbors to the left query point if the line where the query point (red) originates was not omitted from the query surface.

Figure 3.7: Choosing neighbors in an area described by a fixed radius will not work as the sampling density of the area is highly variable. There may easily be zero points in a given distance that belong to a different line than the query point.
Figure 3.8: Projecting the surface normal vector into the local $yz$-plane to obtain the $y$-gradient.

(which is already covered by the in-line gradient calculation). From that vector, the sought-after surface gradient in $y$ direction is obtained (see Figure 3.8). The result of the estimation is a patch of the environment elevation map located at the current vehicle position estimate.

3.2 Sliding window approach

As computation power is limited, the estimation of the surface gradients is becoming increasingly lengthy as the mission duration grows. Therefore, a sliding window approach in combination with a fixed lag is proposed. That way, a fixed number of measurement lines is treated in each step which keeps the computational burden of the inversion step constant. As new measurements arrive, older lines are dropped from the sliding window and the estimation is executed again. Of course it is not necessary to perform a re-estimation for every single new line as the other measurements stays the same and therefore the normal estimation also for the most part remains the same. Only if a new measurement comes to lie in the neighborhood, the normal estimation changes.

Besides that, the fact that due to the fixed lag the query line does not lie on the
boundary avoids unstable surface normal estimation in most cases as the surface normal vector cannot be estimated well on boundary points. Yet, not all of the available information on a certain location is used. This is illustrated in the top image of Figure 3.9 where older lines are not considered even though they could contribute to the estimation. A specific selection of only those lines that lie in a certain area would highly increase the complexity as a search over all measurements of the mission would be necessary. At the moment, we have not implemented this feature due to high computational cost, but in further versions with non-line-based surface parameterization one could perhaps implement it efficiently.

4 Three-dimensional features

On the estimated elevation map environmental features can be extracted that describe salient points or regions. Points or regions where the feature description closely matches earlier occurrences build a so-called loop closure. However, it is of utmost importance that the match stems from an actual re-visit of the same
area. Unfortunately, it may also happen that the feature descriptor of two different regions is very similar and it would be fatal to introduce a loop closure there as no re-visit has happened. This is a case where a SLAM approach will no longer be able to create accurate map or localization estimates. Finding robust feature descriptors that are unique for a certain location is therefore key to successfully running a SLAM system. A more detailed study on loop closure in an AUV context can be found in [LHW+10].

A robust matching mechanism does not try to match single features but several features at once. Hence, using a sliding window approach lends itself well to a SLAM method which uses feature-based submaps. While the association of single features is rather unstable, matching a set of features of a submap to a set of features of another submap provides a much more robust SLAM estimation.

4.1 Short survey on terrain identification features

There is a multitude of three-dimensional features that may be suited to be used in a SLAM context. An overview and a description of several features found in literature is given in this section, however they have not yet been investigated for their robustness.

It can be argued that techniques to identify human fingerprints may also be well suited for recognition of places in submarine terrain. However, in fingerprint
research the identification task is mostly treated as a 2D problem where common image processing techniques are employed. In spite of this, using a 3D terrain map with elevation values interpreted as intensity values, fingerprint matching techniques may indeed be applied to the recognition problem.

In [ZYZ05] a descriptor is presented that is invariant regarding translation and rotation. Given an underwater vehicle moving relative to the seafloor these should be the transformations experienced most often in real data. The presented matching technique does not only use single minutiae of the fingerprint to perform the matching but rather a whole set that is matched at a time. This leads to a more robust estimation of the matching pairs.

Rusu et al. presented point histogram features (PFH) [RMBB08] and later a slimmed version called fast point histogram features (FPFH) [RBB09]. They generalize the mean curvature over $k$ neighbors and in this way obtain a discriminative point descriptor. The FPFH descriptor is slightly different from the PFH descriptor but can be computed much faster and is easily parallelizable. The faster computation is traded for worse predictability of the descriptor’s coverage area though. The descriptor can be calculated for every point whereas for matching purposes only points with a unique descriptor should be used.

In [CHH99] the focus is on large data sets. Their method is based on Spin Images [Joh97]. They describe that on differently meshed versions of a terrain the spin images at the mesh points change considerably. They propose to interpolate the meshed surface and create the spin images on the interpolated surface. That way the spin images resemble each other much more. The downside is that the surface interpolation is quite costly from a computing perspective. One has to keep in mind that as an unmeshed point cloud is given there should be quite some computing overhead.

Sun and Abidi [SA01] present another descriptor for points: They use geodesic circles around an interest point and project them onto the tangential plane in that point. This yields a kind of fingerprint for this point. Salient points have a great radius variation in their fingerprint. They show that for different views of the same point the feature descriptor is very similar. In order to use this descriptor on point cloud data, a geodesic measure needs to be obtained first. This in turn calls for a triangulation of the point cloud which is computationally expensive.

Point Signatures [CJ96] is a well-established descriptor that intersects a ball around an interest point on a surface to obtain a space curve. Subsequently, a plane is fitted through the space curve and the plane is moved along its normal vector until the interest point lies on the plane. After projecting the space curve to the shifted plane the projected curve is sampled clockwise at the point farthest away from the interest point. This is called signature of the point and it can deal with discontinuities while
being invariant regarding translation and rotation.

### 4.2 Terrain estimation

In [HBHH10] a method is presented to learn a continuous surface function by kernel functions. The method was originally created for 3D laser scanner data and has the advantage of not only estimating a surface but also calculating uncertainty bounds for the estimated surface. As the method is especially suited for variable point density, a kernel-based surface representation could also be suitable for the surface reconstructed from side-scan sonar data.

### 5 Conclusion

In this paper it is shown how an autonomous underwater vehicle can navigate based on side-scan sonar data. The importance of correct spatial referencing of sonar data and the implications for the sonar data inversion have been illustrated. It has also been described how a suitable SLAM framework may be designed and an overview of different surface features to employ in the SLAM context has been given.

The next step consists of finding 3D features that are robust against different sampling patterns of the surface.

### Bibliography


Navigation using 3D features from side-scan sonar data for a deep-sea AUV


