Pose Estimation and Trajectory Derivation from Underwater Imagery

Anuj Sehgal*, Daniel Cernea†, Milena Makaveeva‡
*Computer Science, Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany
s.anuj@jacobs-university.de
†University of Kaiserslautern, Computer Graphics and HCI Group, 67653 Kaiserslautern, Germany
cernea@cs.uni-kl.de
‡Indian Underwater Robotics Society, E-118 Nar Vihar Part-I, Sector 34, 201301 Noida, India
milena@iurs.org

***Abstract*—Obtaining underwater imagery is normally a costly affair since expensive equipment such as multi-beam sonar scanners need to be utilized. Even though such scanners provide imagery in form of 3D point clouds, the tasks of locating accurate and dependable correspondences between point clouds and registration can be quite slow. Registered 3D point clouds can provide pose estimation and trajectory information vital to the navigation of a robot, however, the slow speed of point cloud registration normally means that maps are generated offline for later use. Furthermore, any algorithm must be robust against artifacts in 3D range data as sensor motion, reflection and refraction are commonplace. In our work we describe the use of the SIFT feature detector on scaled images based on point clouds captured by sonar in order to register them in real-time. This online registration approach is used to derive navigational information vital to underwater vehicles. The algorithm utilizes the known point correspondence registration algorithm in order to achieve real-time registration of point clouds, thereby generating 3D maps in real-time and providing 3D pose estimation and trajectory information.**

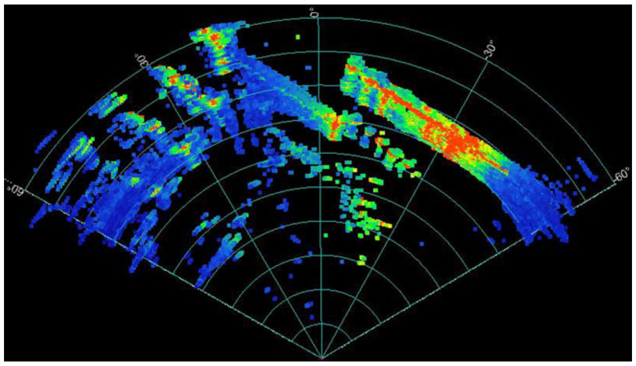

Figure 1. The point cloud representation returned from a Tritech Eclipse Multi-beam Sonar, as seen in the user interface [2]. Blue is weaker intensity, whereas red is strong. As can be seen, the data is pretty noisy due to the multiple acoustic reflections of the wall, albeit these are not strong in intensity.

I. Introduction

Underwater imaging and mapping is a complicated and expensive task due to the high accuracy sonar devices require. However, data acquired from such devices can not only benefit underwater robotics, since it can be used to generate navigational maps, but various fields such as archaeology and geology also stand to benefit from such data. Typically, in all these application domains it is rare to get a single representation of the data [1] and as such multiple frames of point clouds have to be obtained and registered with respect to each other in order to construct a composite map or scene. This composite data can be then further utilized for localization, analysis or visualization purposes. Such registered data can also be used to obtain pose-estimation and further derive a trajectory of a vehicle.

Full automation of the registration process of point clouds obtained from sonar imagery is a topic of active research and most systems still rely upon user input in order to determine the initial transformation. Additionally, the algorithms are highly processor intensive [3] making real-time registration of these point clouds a non-trivial effort. As such, when sonar devices are deployed for navigation tasks, they are mostly used to collect data and then register the point clouds offline for future use [4]. Furthermore, point clouds provide another challenge as compared to images in the form of noise that may be present within the returned data, causing false artifacts to appear in the point clouds or making the point cloud too sparse, with not enough usable information within it [5]. For example, point cloud data obtained from a multi-beam sonar such as the Tritech Eclipse normally suffers from a high degree of noise due to ambient conditions, reflections and refractions of the acoustic sonar beams. An example of such noisy data can be seen in in Figures 1 and 2, where a Tritech Eclipse multi-beam sonar was used to obtain point clouds of a wall. Such noise in point cloud data means that it is extremely important for the correspondence detection algorithm to be able to successfully function despite the existence of noise [6], and this makes most such algorithms processor intensive.

However, fully automated online registration can be achieved in case 2D images are used. As such, our approach depends upon locating robust features, invariant to scale, rotation and point of view within 2D images derived from the point clouds by utilizing a square-root scaling approach to convert the euclidean distances to each individual point or using acoustic intensities associated with these points. These robust features can then be used with a high certainty to locate correspondences between the point clouds, by matching

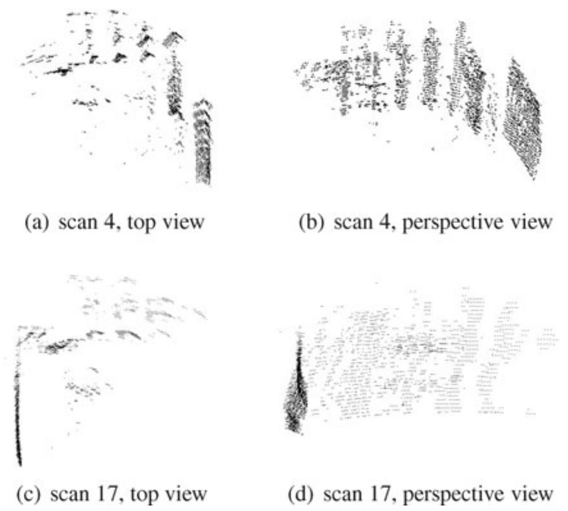

Figure 2. Multiple perspective views of the point cloud obtained from the data returned by the Tritech Eclipse [2], based on the data seen in Figure 1. It is quite clear that the data is pretty noisy.

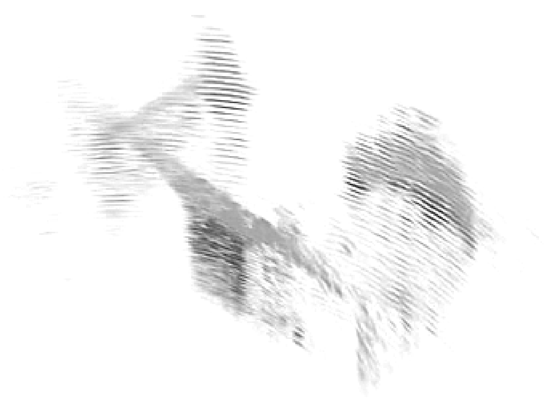

Figure 3. A 3D point cloud, representing the reflected acoustic intensity of each point in greyscale, of an underwater arch produced by the Tritech Eclipse Multi-beam Sonar.

these features within two sets of voxels. However, to reduce the effect of the possible noise in range data, it is necessary to select a feature descriptor that is very robust and, more importantly, invariant to scale and rotation changes.

To meet our goals of locating a high number of features with a high degree of certainty and repeatability from multiple poses, and in data with high noise, an algorithm utilizing features based upon the Scale Invariant Feature Transform (SIFT) descriptor model [7] was developed to find correspondences between the point clouds. The SIFT features are highly robust, in that they are orientation invariant and are applicable at multiple scales. The algorithm is able to generate a large number of localized features with a relatively low computational cost. It was applied to point clouds by utilizing a square-root scaling approach on the euclidean distance from origin to each point in the point cloud or by using acoustic intensities associated with these points [8]. The detected SIFT features in these intensity or range images can be matched to their corresponding voxels in the point clouds in a straightforward fashion.

Registration of the two point clouds is then carried out using a known point correspondences algorithm. The resulting rotation and translation matrices combined with the scaling factor provide for the pose estimation and can also be used to derive a trajectory based purely on sonar imagery.

The following sections of the paper present information related to the SIFT algorithm and then proceed to describe the approach used to find SIFT features in 3D point clouds. The correspondence detection algorithm and the registration method used are also discussed. Some results, test and performance data obtained using the approach are presented and discussed along with the conclusions that are drawn from the results.

II. The SIFT Feature Detector

Robust detection of features in a scene is necessary in order to find correspondences within a point cloud so as to expedite the registration process, which normally can be computationally expensive. The features provided by the SIFT algorithm are local and invariant to image scale and rotation, thereby making them quite robust [9]. The SIFT algorithm is implemented in four stages that provide a result in form of multiple feature descriptors that are represented as a 128-element vector to achieve scale and rotational invariance.

The first stage of the algorithm is where all possible points of interest, known as key-points, are detected. In order to achieve this, the input data is successively convolved with Gaussian filters at different scales, and then the difference of successive Gaussian-blurred images are taken. The local extremum points that exist within the Difference of Gaussians (DoG), an approximation to the Laplacian, at multiple scales are then accepted as the key points. Once the initial set of candidate key points is obtained from the DoG images, they are analyzed within their own neighborhood and adjacent scales, to determine whether they are a local maxima or minima. Furthermore, the second step discards the key-point coordinates that are located in noisy space. This is achieved by eliminating candidates that lie in a region of low contrast or on the edges.

The third step achieves invariance to rotation by assigning each key-point one or more orientations. To compute the orientation of a point in a scale-invariant manner, the Gaussian-smoothed image corresponding to the scale from which the key-point was originally derived is taken and an orientation and gradient magnitude assigned to it. Magnitude and direction calculations for the key-points are performed for every pixel in the neighborhood and an orientation histogram is generated with 36 bins, each bin covering 10 degrees. Once the histogram is fully populated the orientations with the highest peaks and those that are within 80% of the highest peaks are assigned to the key point.

The final step in the SIFT algorithm actually computes the descriptor vector that can be used to identify and further match each key point. This step is extremely similar to the orientation assignment method. The feature descriptor is computed as a set of orientation histograms on a pixel neighborhood of

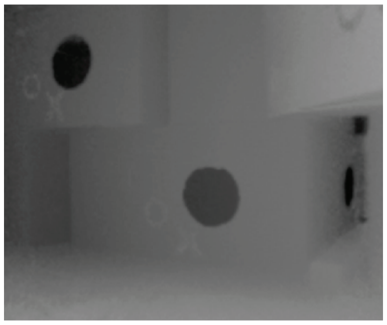

Figure 4. An example of a square-root scaled image of a point cloud as obtained from a terrestrial Swiss ranger. Acoustic intensity images were used in our experimentation due to the ease of filtering out noise from it.

size 4 times 4. The histograms are relative to the key point orientation and the orientation data is derived from the image that corresponds to the key point's scale. The representations now contain 8 bins, each leading to the derivation of a SIFT feature vector that contains 128 elements. This vector may be used to perform image matching or pattern recognition.

III. Registering 3D Sonar Imagery

The aim of the work presented is to be able to enable automatic real-time registration of the 3D point clouds for underwater imagery. Currently, the most popular method for registration is the Iterative Closest Point (ICP) algorithm or some derivative of the same [1]. The ICP algorithm and most of its derivatives are computationally expensive, giving rise to the necessity of being able to perform registration based upon pre-located correspondences from a fewer set of points, in order to speed up the overall performance of the registration process. However, this approach requires that the pre-computed correspondences between the point clouds be calculated quickly, while also ensuring their accuracy between frames that could have changing rotation, translation and scaling. In order to achieve this goal a three-step algorithm that uses the SIFT feature descriptor to describe key points in point clouds is designed. The three steps of the algorithm (data preprocessing; SIFT descriptor generation and feature matching to locate correspondences; and registration of point clouds) are discussed in the following subsections.

A. Data Preprocessing

The SIFT feature detector is designed to function only with 2D datasets and as such, the point clouds need to be converted to images before features can be located within them. Considering that most sonar devices provide range data along with acoustic intensity information, there are two options available in order to convert the point clouds to 2D images.

The most straightforward approach creates a greyscale image based on the acoustic intensities associated with each point in the point cloud. An advantage of using such an image is that it can be easier to filter some of the noise out. For example, low-intensities can be completely ignored in order to filter out the reflections that arrive at the receiver; this will work in most scenarios since reflections always travel longer leading to lower intensities. A point cloud based on this intensity data can be seen in Figure 3.

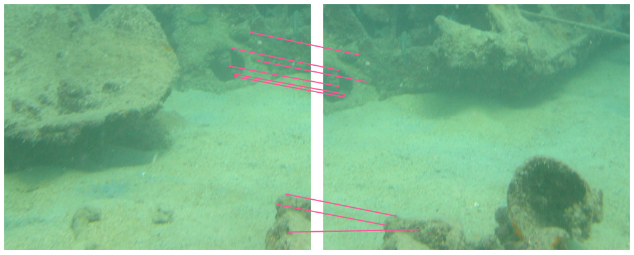

Figure 5. SIFT descriptor based matches for two frames from a set of underwater images captured by a waterproofed camera. The red lines show the SIFT feature correspondences discovered.

The second approach involves calculating the Euclidean distance to each individual (x, y, z) coordinates within the point cloud from the origin $(0, 0, 0)$ and scaling using square root scaling, i.e. taking the square root of this distance. An image representation of range data square root scaling can be seen in Figure 4. This data was derived from a Swiss IR Ranger mounted on a mobile robot. A sonar based range image is not pictured here since we utilized acoustic intensities, provided by the Tritech Eclipse, to construct images for the SIFT feature detector.

Upon obtaining the 2D image data, it is passed through a PNG converter in order to obtain images to which the SIFT operator is applied. The SIFT feature detector requires continuous points in the neighborhood of a pixel to function. Voxels in a 3D range point-cloud are not densely located, and as such the SIFT detector cannot be extended to 3D range point-clouds directly. This necessiates the preprocessing step to convert point clouds to images which can then be used with the SIFT feature detector.

B. SIFT Feature Detector and Matching

The SIFT feature detector is built using OpenCV [10] to follow closely the SIFT algorithm from [7], [9]. The SIFT algorithm takes as input a PNG image corresponding to the representation of the point cloud and computes the 128-element vectors for every identified feature key point.

Upon obtaining the SIFT feature descriptors from the square root scaled images for the two point clouds to be registered, correspondences between the (x, y, z) coordinates in the point clouds is obtained by searching for matching SIFT descriptors, using the RANSAC algorithm [11]. The RANSAC algorithm selects a set of feature pairs randomly and computes the set of all feature pairs conforming to the implied transformation. A support set is rejected if it results in a size that is below a certain threshold. Figure 5 shows matches found between two underwater images taken from a waterproofed camera, as a proof that the SIFT feature detector functions on underwater imagery.

Once the matches are found on the basis of the RANSAC algorithm, correspondences between the 3D point clouds are easily derivable since the corresponding location of each key point in the scaled image data is known within the point cloud as well. The set of resultant correspondences can now be further used with the chosen registration algorithm.

C. Point Cloud Registration

Registration is necessary in order to be able to compare or integrate the data from different measurements. This step provides the relative rotation, translation and scale of the two 3D point clouds being compared. The popular ICP algorithm is memory and processor intensive, thereby being unsuitable for real-time applications [12]. However, if a known points correspondence algorithm is used, this can considerably speed up the registration performance.

As such, for the purpose of speeding up the registration step and owing to the robustness of the SIFT features, the known points correspondence registration algorithm based on quaternions is utilized in our approach. Every corresponding point in the range point clouds can represent a quaternion with $c = 0$ and x, y and z coordinates given by the respective coordinates of the point in the 3D point cloud.

By comparing data from two consecutive point clouds, we are able to retrieve the quaternion of the rotation matrix $\breve{u}^*$ from the eigenvector corresponding to the maximum positive eigenvalue of the 4x4 matrix N shown below:

$$N = \sum_{i=1}^{n} \bar{\Gamma}(\breve{r}_{l,i})^T \Gamma(\breve{r}_{r,i}) \tag{1}$$

Further, the rotation matrix can be obtained from the rotation quaternion by,

$$R = \bar{\Gamma}(\breve{u}^*)^T \Gamma(\breve{u}^*) \tag{2}$$

Having obtained the rotation matrix R, the translation quaternion and scale factor are calculated using,

$$r^*_{l/r} \equiv \bar{r}_r + sR_{r/l}(\bar{r}_l) \tag{3}$$

$$s^* = \frac{\sum_{i=1}^{n} (r'_{r,i})^T R_{r/l}(r'_{l,i})}{\sum_{i=1}^{n} \left\| r'_{l,i} \right\|^2} \tag{4}$$

The matrix calculations required for the registration step were performed using the GSL library [13] and several extensions were written in order to calculate the eigenvectors, eigenvalues, vector normalization and etc.

We also further extended our work in order to utilize registration to derive the roll, pitch and yaw between the consecutive frames. This can be extremely useful in robotics since it provides a method to derive odometry by using only range sensor data. After obtaining the rotation matrix, calculation of the respective roll, pitch and yaw is a straightforward task of selecting the appropriate row/column pairs from the rotation matrix. The change of attitude between the two point clouds can be computed from the least-square rotation matrix, $R = \{r_{i,j}\}$ for $i = 1, 2, 3$ and $j = 1, 2, 3$ by:

$$\begin{aligned} \psi &= \operatorname{atan2}(r_{22}, -r_{11}) \\ \theta &= \operatorname{atan2}(r_{32}, -r_{12}\sin\psi + r_{22}\cos\psi) \\ \phi &= \operatorname{atan2}(r_{13}\cos\psi + r_{23}\sin\psi, r_{11}\cos\psi + r_{21}\sin\psi) \end{aligned} \tag{5}$$

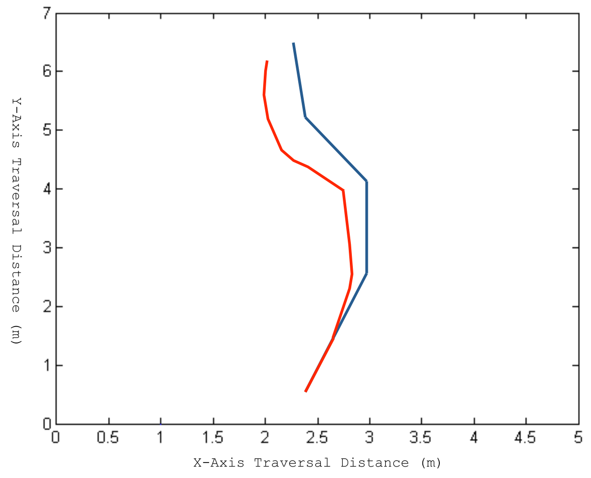

Figure 6. Motion of the robot as predicted by the odometry derived from the SIFT based registration procedure vs. robot motion as obtained from navigational sensors (IMU, gyroscope and compass). Red line is from point clouds matching, while blue is the recorded robot odometry.

Here, we use the Euler angles (roll ϕ, pitch θ, yaw ψ) to describe the attitude. Having obtained translation and the yaw, pitch and roll the robot odometry is available. This can be used in localization and mapping tasks commonly performed by robots. It can also be used to estimate the pose of the robot, given that the original pose is known. This is so, because a registration based approach such as this can only be used to obtain the change in attitude and position, rather than absolute position or attitude. As such, as long as the original position is known, the changed pose and trajectory, in relation to the original, can be obtained.

IV. Testing and Results

The dataset consists of 120 frames of point clouds obtained by moving a Tritech Eclipse mounted on a frame within a pool. The (x, y, z) location within the point cloud corresponds to the measured distance in meters. This data is retrieved frame by frame and supplied to the software running on a Linux platform in order to register the two point clouds. The test system used was an Ubuntu Linux installation on a platform with 1 GB RAM and a 2 GHz Intel Core 2 Duo CPU.

The robustness of the SIFT features and their ability to have more than a single orientation at a particular point can cause the RANSAC algorithm to successfully find more than one correspondence of a feature in the adjacent frame. This is especially useful since the approach requires that at least 3 correspondences be found between adjacent frames in order to successfully register them.

We ran an experiment to derive odometry based on the point cloud set. In order to do so, each consecutive point cloud was registered with the previous one and a rotation, translation and scaling were derived, which also provided us with the yaw, pitch and roll. The obtained yaw, pitch, roll, and translation values were compared with those provided by IMU, gyroscope and compass sensors attached to the rig by plotting a route map for the robot as predicted by both data sources and also plotting the yaw, pitch and roll in a similar fashion. The odometry derived from the point clouds closely matched that provided by the other sensors, however, a drift error was noticed. We also noticed that there were certain frames where not enough features would be found and this led to sudden jumps in the data, which further increased the drift error.

Figure 6 shows the path taken by our test rig. While the shape of the positional odometry derived from the 3D point cloud data is similar to the actual path, there is quite a lot of deviation between the two. Besides a lack of correspondences between some frames, this deviation can also be explained by the fact that in some frame-pairs, certain points had multiple correspondences; for simplicity, these were not considered in the final result since the multiple locations of the correspondences led to errors. These results could further be improved by applying some filtering, however, they clearly demonstrate the effectiveness of using point cloud data for obtaining odometry information as well. The obtained results can at least be used as a basis for understanding robot trajectory or assisting a vehicle in localization while information from other sensors like the IMU or USBL might be unavailable.

Lastly, the other important performance criterion is the run-time performance of the algorithm in finding the correspondences between point clouds and then performing the corresponding registration. In our tests, the software was able to achieve an average frame rate of 12.58 fps. We are confident that this could further be improved by adding optimizations methods, which were omitted in this version for the sake of simplicity and testing. For example, each step in the processing pipeline is current a separate program that writes files to a disk and then launches another program to continue the processing. It is safe to say that this adds considerable overhead. However, the obtained frame rate is within the range for real-time performance since most sonar devices themselves take a few moments to return data.

V. Conclusion

In our work we proposed a method to obtain trajectory and pose information using point clouds obtained from underwater sonar data. Our approach applies the SIFT feature detector to point clouds by scaling the data sets into images, which the SIFT descriptor can work with. The point clouds are then registered based upon the correspondences of the feature points. This implementation makes it clear that the SIFT descriptor retains its robustness even when utilized with range data. Moreover, even in highly noisy sonar range data, multiple correspondences are successfully found.

The algorithm performs quite well in locating correspondences between point clouds and registering them with near real time performance. Furthermore, the preliminary experimental results suggest that even odometry data derived from calculating the relative translation, rotation and scaling between successive point clouds is close to being accurate, however, it may need further filtering to have the error component removed and the drift error minimized.

References

[1] G. Bendels, P. Degener, R. Wahl, M. Koertgen, and R. Klein, "Image-based registration of 3d-range data using feature surface elements," in *Proceedings of The 5th International Symposium of Virtual Reality, Archaeology and Cultural Heritage (VAST 2004)*, 2004.

[2] H. Bülow and A. Birk, "Spectral registration of noisy sonar data for underwater 3d mapping," *Autonomous Robots*, vol. 30, no. 3, pp. 307–331, Apr. 2011.

[3] M. Callieri, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno, "Vclab's tools for 3d range data processing," in *Proceedings of the 1st EURO-GRAPHICS Workshop on Graphics and Cultural Heritage*, Brighton, UK, November 2003.

[4] N. Gracias and J. Santos-victor, "Trajectory reconstruction with uncertainty estimation using mosaic registration," *Robotics and Autonomous Systems*, vol. 35, pp. 163–177, 2001.

[5] D. Cernea, "Graphical methods for online surface fitting on 3d range sensor point clouds," Master's thesis, Jacobs University Bremen, Germany, August 2009.

[6] O. Schall, A. Belyaev, and H.-P. Seidel, "Robust filtering of noisy scattered point data," in *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings*, June 2005, pp. 71–144.

[7] D. Lowe, "Distinctive image features from scale- invariant keypoints," *International Journal of Computer Vision (Springer Netherlands)*, vol. 60, no. 2, November 2004.

[8] A. Sehgal, D. Cernea, and M. Makaveeva, "Real-time scale invariant 3d range point cloud registration," in *Image Analysis and Recognition*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6111, pp. 220–229.

[9] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of International Conference on Computer Vision*, Corfu, Greece, September 1999.

[10] V. Pisarevsky and et al, "Opencv, the open computer vision library," 2008, http://mloss.org/software/view/68/.

[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[12] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *Int. J. Comput. Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[13] B. Gough, Ed., *GNU Scientific Library Reference Manual - Second Edition*. Network Theory Ltd., 2003.