Combining Visual Features and Growing Neural Gas Networks for Robotic 3D SLAM

Diego Viejo, Jose Garcia-Rodriguez, Miguel Cazorla

Instituto de Investigación en Informática.
University of Alicante. Alicante (Spain)
dviejo@dccia.ua.es, jgarcia@dtic.ua.es, miguel.cazorla@ua.es

Abstract

The use of 3D data in mobile robotics provides valuable information about the robot's environment. Traditionally, stereo cameras have been used as a low-cost 3D sensor. However, the lack of precision and texture for some surfaces suggests that the use of other 3D sensors could be more suitable. In this work, we examine the use of two sensors: an infrared SR4000 and a Kinect camera. We use a combination of 3D data obtained by these cameras, along with features obtained from 2D images acquired from these cameras, using a Growing Neural Gas (GNG) network applied to the 3D data. The goal is to obtain a robust egomotion technique. The GNG network is used to reduce the camera error. To calculate the egomotion, we test two methods for 3D registration. One is based on an iterative closest points algorithm, and the other employs random sample consensus. Finally, a simultaneous localization and mapping method is applied to the complete sequence to reduce the global error. The error from each sensor and the mapping results from the proposed method are examined.

1. Introduction

Robots and autonomous vehicles use maps to determine their location or pose within an environment, and to plan routes and trajectories [26], [40]. Thus, the problem of automatic map building is an important topic in mobile robotics [25]. To achieve automatic map building, the mobile vehicle has to be accurately located inside the map that is being built. Simultaneous Localization and Mapping (SLAM) techniques [13], [9] have been proposed to overcome this issue, which can be thought of as a chicken-and-egg problem: an unbiased map is necessary for localization, while an accurate pose estimate is needed to build that map. On the one hand, SLAM can be considered as a global rectification [34] of the mobile robot's poses along its route once the loop is closed, and when the robot observes again a previously seen object. On the other hand, estimating the movement performed by a mobile robot using the observations gathered between two consecutive poses is considered to give a local rectification. This topic is usually referred to as egomotion [29]. The methods related to this research are called pose registration, and can be used within SLAM. Our main goal is to perform six degrees-of-freedom (6DoF) SLAM in a semi-structured environment, i.e. a man-made indoor environment [14].

The disadvantages of stereo cameras when used in low-textured environments motivate the use of two new 3D sensors. First, we use an SR4000 infrared camera [2], which measures the distance to objects using the time that the emitted infrared light takes to hit the object and return to the sensor. The SR4000 provides a set of 3D points, a 2D intensity image, and a map of certainty. Second, we use a Kinect camera [3]. Such cameras (also called, generically, RGBD cameras), were originally designed for entertainment, but have become widely popular due to

their notable accuracy and low price. Furthermore, it is relatively easy to use them as a human interaction interface, as they easily detect human body gestures [31]. From this camera, we obtain a 3D dataset along with a traditional 2D image. In contrast to methods that obtain 3D shapes from sets of 2D images, such as [21], [19], and [20], we utilize both the 3D data and 2D images obtained directly from the SR4000 and Kinect sensors.

Obtaining the 6DoF transformation from the robot (egomotion) using only 3D information is not an easy task. One of the first methods used to achieve this was the Iterative Closest Points (ICP) algorithm [11], [7]. ICP is divided into two main steps. First, the relationship between the two sets of input data is obtained by searching for the closest points. This search gives a list of matched points. The second step involves computing the transformation that best aligns the two sets from the matches found in the previous step. This is used to transform the input sets before the first step of the next iteration. The algorithm is iterated until some convergence parameter is attained. Determining this transformation is equivalent to finding the movement performed by the robot between the poses at which data were captured. Some variants of the ICP algorithm have been proposed [5], [10], [4] and [42]. Nevertheless, ICP methods do not work correctly in the presence of outliers (features observed in one frame but not in the other) [35]. The larger the robot movement, the greater the number of outliers.

In contrast, RANdom SAmple Consensus (RANSAC) [16] is an iterative method for estimating the parameters of a mathematical model from a set of observed data that contains outliers. RANSAC can be used to estimate the transformation that best aligns two sets of matched features extracted from two consecutive poses.

In this paper, we propose the use of scale-invariant visual features (SIFT) [32]

from the 2D image, together with a 3D representation of the scene based on a Growing Neural Gas (GNG) network [18], [33], [17]. By means of competitive learning, GNG adapts the reference vectors of the neurons, as well as their interconnection network, to obtain a mapping that tries to preserve the topology of a scene. In addition, GNG networks are capable of a continuous re-adaptation process, even if new patterns are entered, with no need to restart the learning. Thus, GNG provides a fast and high-quality representation of a 3D space, obtaining an induced Delaunay triangulation of the input space that is very useful for determining features such as corners, edges, and so on. We modify the original GNG method to enable its application to sequences: the GNG is adapted sequentially, i.e. the result in a given frame is taken as the input for the next frame. Some 3D reconstruction applications of neural networks can be found in [12], [36], [28], [15], and [27]. However, none of these studies consider sequences of 3D data.

Once the GNG network has been obtained, the 2D visual features are hooked to the closest element on the Delaunay mesh. Then, an egomotion method is used to find the transformation between two consecutive frames. In this paper, we test the two egomotion methods of ICP and RANSAC to establish which offers the best results. We modify the original ICP method to incorporate visual features in the matching process. Correspondences are found that allow us to compare the descriptors of each feature. RANSAC may be more appropriate for this issue, as it is less sensitive to outliers. We empirically compare both approaches in section 5.

The rest of the paper is organized as follows: first, section 2 describes the cameras used in this paper; then, the GNG algorithm is explained in section 3. In section 4, we detail the complete method for solving the SLAM problem; the

experiments discussed in section 5 demonstrate our results, and we present our conclusions and ideas for future work in section 6.

2. SR4000 and Kinect

In recent years, stereo cameras have been replaced by those better able to provide 3D data. Stereo cameras have a very important problem: the lack of texture. In this situation, no 3D information is provided. Nowadays, two new camera models are available. We briefly describe both.

Figure 1: Left: SwissRanger SR4000 Time-of-Flight camera. Right: Kinect camera.

The first kind, known as Time-of-Flight (ToF) cameras, were developed to deliver range (distance) and amplitude maps using a modulated light source. Their main advantage with respect to other 3D devices is the possibility to acquire data at video frame rates, and to obtain 3D point clouds without scanning and from a single point of view.

The basic principle of ToF cameras involves an amplitude-modulated infrared light source and a sensor field that measures the intensity of the backscattered infrared light. The infrared source is constantly emitting light that varies sinusoidally. Objects situated at different distances are reached by different parts of the sinusoidal wave. The reflected light is then compared to the original wave,

allowing the phase shift to be calculated by measuring the intensity of the incoming light (as the phase shift is proportional to the ToF of the light reflected by a distant object). A detailed description of the ToF principle can be found in [23]. The device used in this work is the SwissRanger SR4000 ToF camera, pictured in Figure 1.

In our tests, all the data were acquired directly from the camera, which delivers XYZ point coordinates, amplitude data of the scene, and a confidence map of the distance measurements. In particular, the confidence map is obtained using a combination of distance and amplitude measurements and their temporal variations. This map represents a measure of the probability that the distance measurement of each pixel is correct, so it can be used to select regions containing high-quality measurements or reject low-quality ones. The output data are arranged in 16-bit arrays, of length $176 \times 144 = 25344$. By default, the amplitude data is converted into a value that is independent of distance and position in the image array. The amplitude data has a range of 0-0x7FFF. In our experiments, the amplitude data has low contrast, so we have used an equalization method.

The Kinect device has been a great advance in the field of robotics. It is composed of two sensors: an IR (infrared) projector and IR CMOS camera, and an RGB camera. IR sensors provide depth information—the IR projector sends out a fixed pattern of light and dark speckles, and depth is calculated by triangulation against a known pattern from the projector. The pattern is memorized at a known depth, and then for each pixel, the correlation between the known pattern and current pattern is calculated, providing the current depth at this pixel. The RGB camera has a resolution of 640×480 (307200 pixels) and a working range of between 1 and 8 metres. The advantage of this camera against the SR4000

is that RGB information is obtained, whereas the SR4000 provides information in the infrared spectrum, which cannot return good results using current feature detectors.

3. GNG Algorithm

Under the GNG algorithm [18], a growth process starts from a network of minimal size, successively inserting new units using a particular type of vector quantization [30]. To determine where the new units should be inserted, local error measures are gathered during the adaptation process, and each new unit is inserted near the unit with the highest accumulated error. At each adaptation step, a connection between the winner and the second-nearest unit is created, as dictated by the competitive Hebbian learning algorithm. This process continues until an end condition is fulfilled, for example, the determination of the optimal network topology or some time deadline. In addition, the learning parameters in GNG networks are constant in time, in contrast to other methods where learning is based on decaying parameters. In the rest of this section, we describe the GNG algorithm and end condition used in this study. The network is specified as:

- A set N of nodes (neurons). Each neuron $c \in N$ has an associated reference vector $w_c \in R^d$. The reference vectors can be considered as positions in the input space of their corresponding neurons.

- A set of edges (connections) between pairs of neurons. These connections are not weighted, and their purpose is to define the topological structure. An edge aging scheme is used to remove connections that become invalid due to the motion of the neuron during the adaptation process.

The GNG learning algorithm mapping the network to the input manifold is as follows:

1. Start with two neurons a and b at random positions w_a and w_b in R^d.
2. Generate a random input pattern ξ according to the data distribution $P(\xi)$ of each input pattern.
3. Find the nearest neuron (winner neuron) s_1 and the second-nearest s_2.
4. Increase the age of all the edges emanating from s_1.
5. Add the squared distance between the input signal and the winner neuron to a counter error for s_1, such as:

$$\triangle error(s_1) = \|w_{s_1} - \xi\|^2. \tag{1}$$

6. Move the winner neuron s_1 and its topological neighbors (neurons connected to s_1) towards ξ by a learning step ϵ_w and ϵ_n, respectively, for the total distance:

$$\triangle w_{s_1} = \epsilon_w(\xi - w_{s_1}) \tag{2}$$

$$\triangle w_{s_n} = \epsilon_n(\xi - w_{s_n}) \tag{3}$$

   for all direct neighbors n of s_1.
7. If s_1 and s_2 are connected by an edge, set the age of this edge to 0. If the edge does not exist, create it.
8. Remove edges older than a_{max}. If this results in isolated neurons (without emanating edges), remove them as well.
9. For every λ input patterns generated, insert a new neuron as follows:
   - Determine the neuron q with the maximum accumulated error.

- Insert a new neuron r between q and its farthest neighbor f:

$$w_r = 0.5(w_q + w_f). \tag{4}$$

- Insert new edges connecting neuron r with neurons q and f, removing the old edge between q and f.

10. Decrease the error variables of neurons q and f by multiplying them by a constant α. Initialize the error variable of r with the new value of the error variable of q and f.
11. Decrease all error variables by multiplying them by a constant γ.
12. If the stopping criterion is not yet achieved (in our case the stopping criterion is the number of neurons), go to step 2.

Figure 2 shows a flowchart of the GNG learning algorithm. With regard to the processing of image sequences, we have introduced several improvements to the network to accelerate the representation and allow the architecture to work faster.

For the experiments, we used GNG parameters of $N = 2000$, $\lambda = 500$, $\epsilon_w = 0.1$, $\epsilon_n = 0.001$, $\alpha = 0.5$, $\gamma = 0.95$, and $\alpha_{max} = 250$. Figure 3 shows the result of applying this GNG algorithm to a 3D points set from a Kinect camera.

The GNG process takes almost 2 s to process the first pose, but then, thanks to our adaptive method, less than 50 ms is required to process subsequent poses. It has been shown [41] that applying GNG to 3D data reduces the error in 3D data alignment processes. We use the GNG process when (Project number 44173) from the camera is noisy. In fact, the SR4000 camera is much noisier than the Kinect, and thus we apply the GNG process to data from the SR4000.

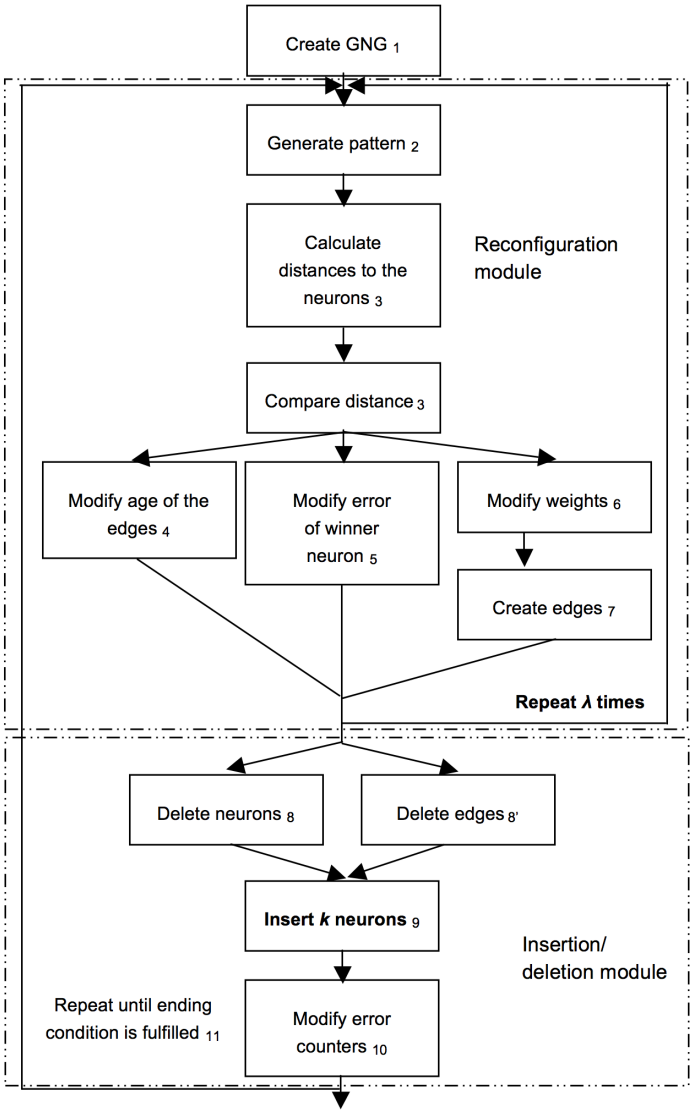


Figure 2: GNG flowchart.

3.1. Representation of Point Cloud Sequences

The GNG algorithm has been adapted to represent point cloud sequences using models learnt from previous data acquisitions. We have introduced several

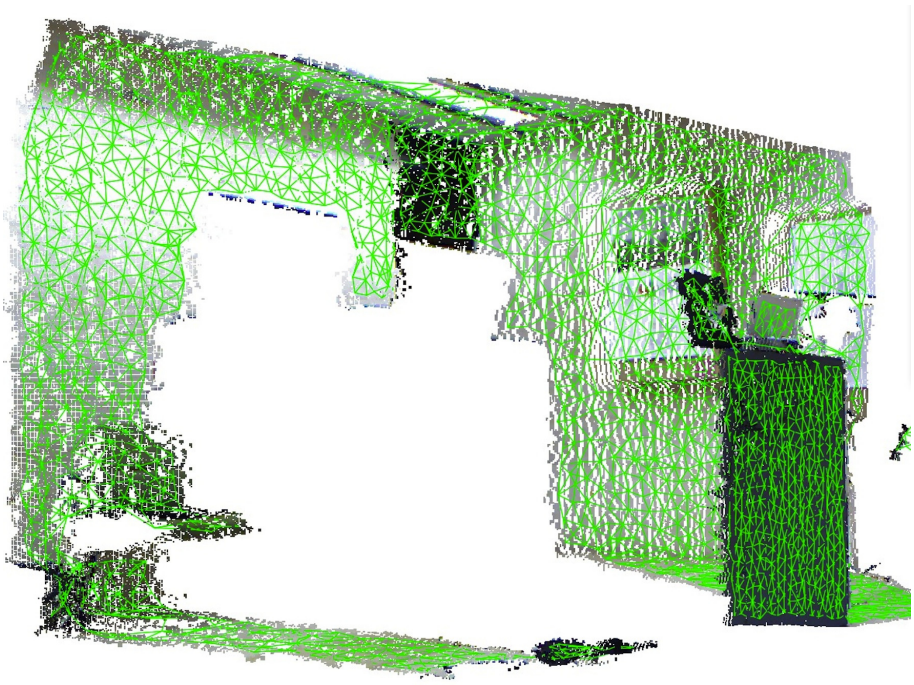


Figure 3: Applying GNG to the Kinect dataset. Green lines are the GNG structure.

improvements to the network that accelerate the representation and allow the architecture to work faster. The main difference when using the GNG algorithm is the omission of insertion–deletion actions (steps 9 to 11) after the first frame. Because no neurons are added or deleted, the system maintains correspondence during the whole sequence, intrinsically solving the correspondence problem by using neurons as fixed markers. At the initial moment t_0, the representation is obtained by performing a complete adaptation of a GNG. However, for the following frames, the previous network structure is employed. Hence, the new representation of the input space is obtained by iterating the internal loop of the learning algorithm of the GNG, relocating the neurons, and creating or removing the edges. This adaptive method can also handle real-time constraints, because the number

λ of times that the internal loop is performed can be chosen according to the time available between two successive frames, which depends on the acquisition rate. A more detailed description of this method can be found in [41]. The average time required to obtain a GNG network on a frame is less than 50 ms, using this adaptive method. GNG reduces the amount of input data, at the same time preserving its structure. This gives us two advantages. First, we have to process fewer points, so the next step of feature extraction is accelerated. Second, the number of outliers, which are one of the main sources of errors in this kind of application, is reduced.

3.2. Time Consumption per Stage

After presenting the different stages of the algorithm, it is necessary to compute the percentage of instructions executed at each step with respect to the total number. To achieve this, we use a profiler so that, depending on the values of the parameters with which we have adjusted the algorithm (number of neurons and number of input patterns), we obtain the percentage of instructions executed at each stage.

It can be seen that most of the execution time of the algorithm is consumed in the search for winning neurons, at which stage the Euclidean distances are also calculated.

Table 3.2 shows the percentage of instructions performed at each stage of the algorithm for different values of the number of neurons N and input patterns λ. The table also shows how stage 3 increases its percentage with respect to the total when N and λ are increased.

An accelerated version of the GNG algorithm has been developed and tested on a machine with an Intel Core i3 540 3.07 GHz processor. The multi-core

Neurons	Patterns	Stage 2	Stage 3	Stage 4,5,6,7	Stage 8	Stage 9
1000	500	1.8	**73.30**	15	1.2	0.8
5000	500	0.7	**88.80**	5.8	0.9	1
10000	500	0.4	**93.20**	3.3	0.6	0.8
20000	500	0.3	**97.60**	1.9	0.5	0..9
1000	1000	1.8	**69.60**	21.3	0.6	0.3
5000	1000	0.7	**90**	5.6	0.5	0.5
10000	1000	0.4	**94.3**	3.2	0.3	0.4
20000	1000	0.3	**96.5**	1.9	0.2	0.4

Table 1: Percentage of instructions executed at each stage of the GNG algorithm illustrated in figure 2

CPU implementation of the GNG algorithm was developed using Intel's Threading Building Blocks (TBB) library [1], taking advantage of the multi-core processor capabilities and avoiding the existing overhead [8]. The number of threads used in the multi-core CPU implementation is the maximum defined in the specifications of Intel's i3 540 processor.

3.3. CPU vs GPU Implementation

In Figure 4, it can be appreciated that the CPU version is faster during the first iterations, while the multi-core CPU implementation is slower because of the existing overhead caused by the management of threads and the subdivision of the problem. However, after a number of iterations, the performance of the multi-core CPU stabilizes and improves the CPU results.

In our application, we used maps with 2000 neurons, but in the case of larger maps, we should accelerate the process using a multi-core CPU or even multi-core

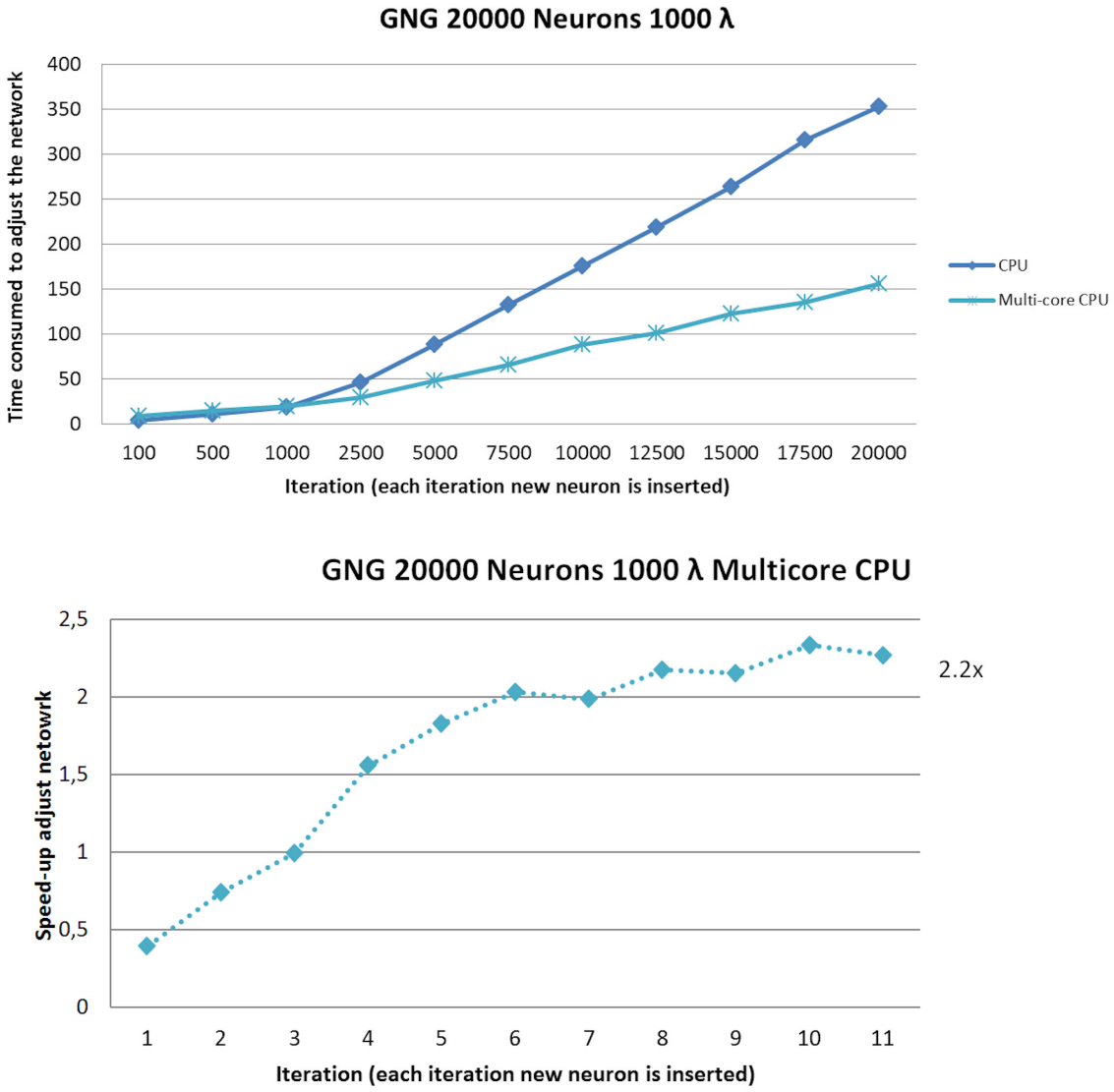


Figure 4: Example of CPU and multi-core CPU GNG runtime (ms) and speed-up.

GPUs.

3.4. Rate of Adjustments per Second

We have also performed experiments that show how the multi-threaded CPU implementation of the GNG algorithm is not only capable of learning faster than a single-threaded CPU, but can also obtain more adjustments per second. For instance, after learning a network of 20000 neurons, we can perform 8 adjustments per second using the multi-core CPU, whereas the single-core CPU achieves only 2.8 adjustments. This number grows from 22 to more than 30 for a network of 5000 neurons (our application). This means that the multi-core CPU implementation can obtain a better topological representation under time constraints. Figure 5 shows the adjustment rate per second performed by both implementations. The figure also shows that, when the number of neurons increases, the CPU implementation cannot attain a high rate of adjustments.

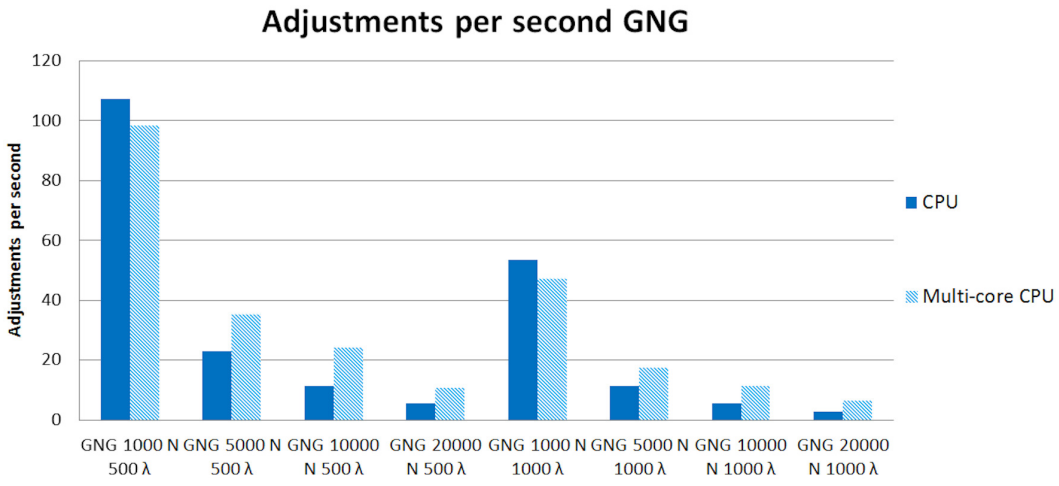


Figure 5: Rate of adjustments performed by different GPU devices and CPU.

4. SLAM from Features

Once we are able to model 3D data, we obtain 2D features and place them in 3D, using the 3D information provided by the cameras. In the case of the SR4000, we extract features from the amplitude of the infrared image, which is similar to a grey-scale image but in the range of the infrared spectrum. For the Kinect, we directly use the 2D image provided. In this work, we will use SIFT features [32]. Nevertheless, the method can be applied to any other 2D feature detector.

The SIFT method is widely used in computer vision systems to detect and describe features in an image. It performs a local pixel appearance analysis at different scales, and obtains a descriptor for each feature that can be used for different tasks, such as object recognition. The SIFT features are designed to be invariant to image scale and rotation. Currently, there are a number of feature detectors and descriptors, like SURF [6], but it is beyond the scope of this work to determine the most efficient. For a good study of different features, see [22] and [39].

Once the features have been detected, they are hooked to the GNG or directly to the corresponding 3D point. For the case of GNG, the 2D coordinates of a SIFT feature are projected to 3D using the Kinect library. The SIFT feature is then attached to the closest node (using the Euclidean distance between the 3D SIFT coordinates and the 3D node coordinates) of the GNG structure. The SR4000 camera has the advantage of a confidence value, which can be used to remove those features that cannot be trusted. This represents an improvement over stereo systems and the Kinect camera, as it enhances accuracy by removing erroneous points.

To calculate the robot egomotion, we need a method to find the 3D transfor-

mation between two consecutive poses. We present two different methods that obtain the egomotion performed by the robot. Both are based on the matching information provided by the features. Feature descriptors are used to determine the matches from two consecutive poses. We now briefly describe the two methods.

The first is based on the RANSAC algorithm [16]. It is an iterative method that estimates the parameters of a mathematical model from a set of observed data which contains outliers. In our case, we look for a 3D Euclidean transformation (our model) that best explains the data (matches between 3D features). At each iteration of the algorithm, a subset of data elements (matches) is randomly selected. These elements are considered as inliers, and are used to compute a model (3D Euclidean transformation). All other data are then tested against the computed model, and included as inliers if their error is below a threshold. If the estimated model is reasonably good (i.e. its error is low enough and it has enough matches), it is considered to be a good solution. This process is repeated a number of times, before the best solution is returned.

The second method is based on the ICP algorithm [7], [37], [43]. ICP is used to match two 3D point sets, but it cannot find a good alignment in the presence of outliers. A survey on ICP-based methods can be found in [38]. ICP does not give good results for long-time movements, because these produce a lot of outliers. Using features like SIFT, along with additional information, i.e. descriptors which are robust to brightness and point-of-view changes, is sufficient for this task. Hence, we use descriptors to find matches, instead of using the Euclidean distance as in the original ICP. We have decided to select features close to the camera, because greater distances result in greater 3D errors. Thus, only features with a Z distance below a threshold are considered for matching between two con-

secutive sets. The movement between two consecutive frames is examined so that sets of features can be selected which intersect and have enough features to enable matching. If the movement is limited to, for example, 1 meter, we select features from 1 to 2 meters in the first frame, and from 0 to 1 in the second, and similarly for the angles. If there are not enough matches, we expand the limits from 1 to 3 meters and from 0 to 2, and so on, to find the minimal number of matches or to reach a long distance (8 or 10 meters, depending on the camera).

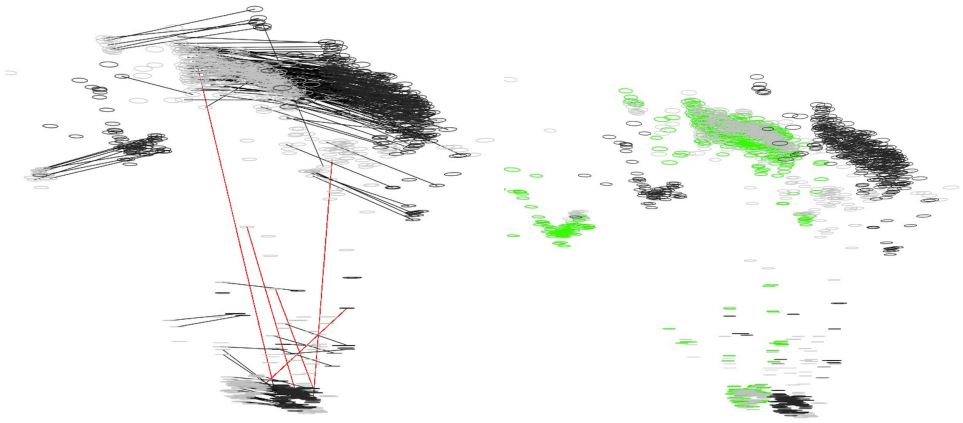


Figure 6: An example of the matching method. Left: initial. Black lines indicate correct matches. The lines in red represent incorrect matches. Right: final registration. The grey ellipses are from one pose, black from the other pose. Finally, the green ones are the black ones following transformation.

In Figure 6, we show an example of initial and final matching. On the left, the initial matching (lines) between features is shown in both frames (here, a frame has 3D points). On the right, the final pose is shown, once the 3D transformation has been determined and applied. On the left-hand side of Figure 7, two consecutive datasets are shown in the same coordinate frame. The right-hand side shows

Figure 7: An example of matching two consecutive poses. Left: before registration. Right: after registration.

the two frames after registration.

We apply one of the proposed methods to a sequence of robot poses. Thus, we have an estimation of the egomotion performed by the robot, without the use of odometry information. As the results will contain some errors, we apply a SLAM algorithm designed for 3D mapping using a graph representation, the Toro method [24], which adjusts to our results. This method uses a tree structure to define and efficiently update local regions in each iteration by applying a variant of stochastic gradient descent.

5. Results

We have used two indoor sequences to test the reliability of our method. We have selected indoor sequences because solar light has a negative influence on both the SR4000 and the Kinect. Besides, the cameras' range is limited to 8–10 meters. Both sequences were obtained by rotating the camera over its center, until a complete turn had been performed. The method can be applied to non-circular sequences (translational sequences), but a circular sequence ensures that the loop

closure is detected. Furthermore, in egomotion, rotation is the main source of error. The first sequence was taken by the SR4000 camera, and the second by the Kinect. The methods described in the preceding section were applied to both, but first, we applied the classical ICP [7] to the sequences in order to compare our results with a state-of-the-art method. As can be seen in Figure 8, the results given by the ICP method were not good. The ICP method was unable to determine a registration that was close to the actual one. We have tried to apply the TORO method to the ICP result, but TORO was also unable to minimize the error.

We then applied RANSAC and the ICP-like method explained in the previous section. Figures 9 to 12 show the reconstruction obtained after applying both methods. The red points in the middle of the images indicate the egomotion performed between poses. Note that after applying SLAM, the results are improved.

Figure 8: Result of mapping using the classical ICP method for both cameras. Left: SR4000. Right: Kinect.

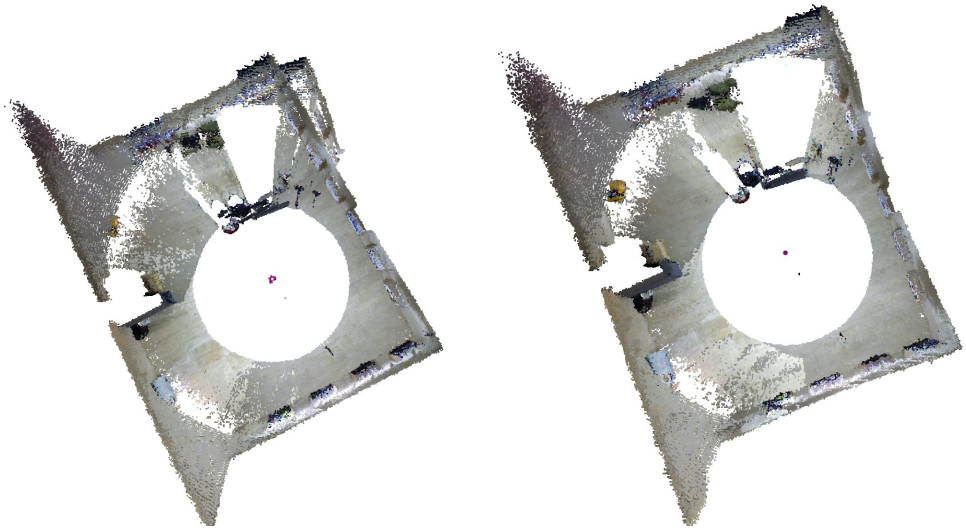


Figure 9: Result of mapping using the RANSAC method and the Kinect camera. Left: before applying Toro. Right: after Toro has been applied.

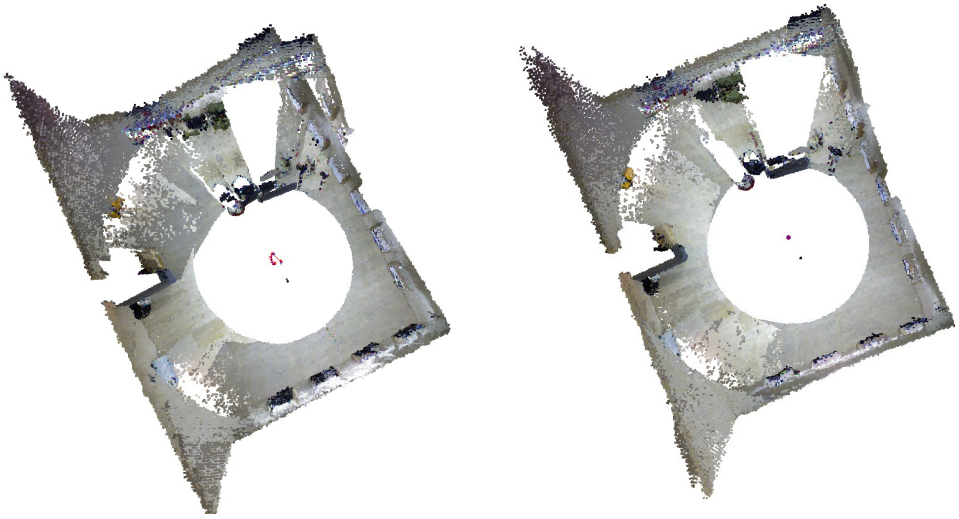


Figure 10: Result of mapping using the ICP-like method and the Kinect camera. Left: before applying Toro. Right: after Toro has been applied.

Figure 11: Result of mapping using the RANSAC method and the SR4000 camera. Left: before applying Toro. Right: after Toro has been applied. In this case, the SLAM result is worse than the others, as the bottom-right part of the environment has an error (it is a straight wall).

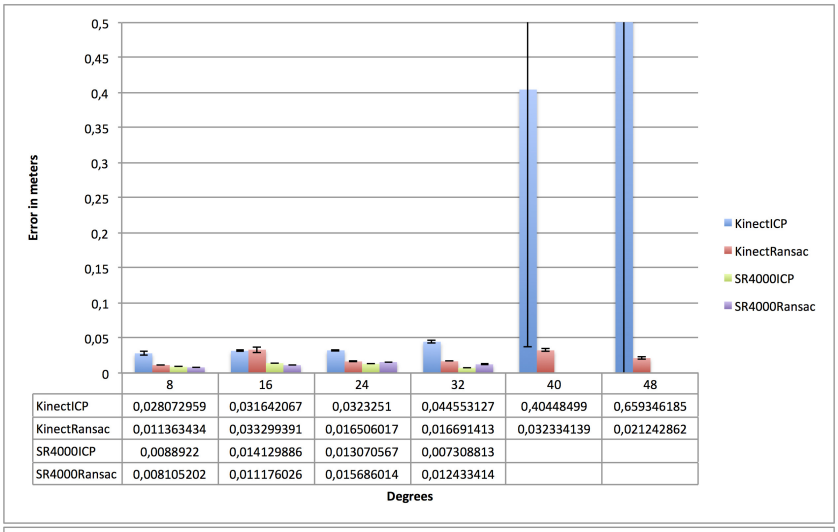
The RANSAC method takes an average of 45 ms, while the ICP-like one takes 30 ms with the Kinect camera. With the SR4000, both algorithms take an average of 10 ms. This is because the SR4000 camera has lower resolution, and the SIFT algorithm provides fewer features. For the same reason, the SIFT algorithm takes less time to process SR4000 images. For SR4000 images, SIFT is calculated in less than 100 ms, whereas for the Kinect camera, it takes more than 1.4 s. This is a huge processing time difference. Thus, the GNG algorithm can be applied to the SR4000 to reduce the associated noise (which is greater than that of the Kinect).

In Figure 13, we show the results of a study on the translational and angular error of both cameras and algorithms. We have calculated these errors for different angles, from $8 - -48^{\circ}$ at intervals of 8°. The top image shows the translational

Figure 12: Result of mapping using the ICP-like method and the SR4000 camera. Left: before applying Toro. Right: after Toro has been applied.

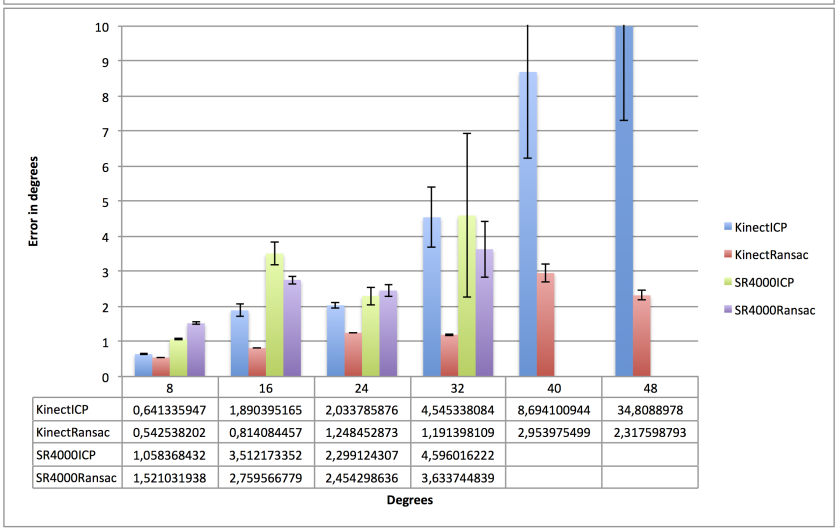
error. This is below 3 cm until an angle of $40°$. The SR4000 camera is not able to find enough correspondences from this angle, so it is impossible to obtain its error. This is due to the higher number of features detected by the Kinect camera, as well as its wider field of view. The angular error (bottom image) is less than $0.5°$. For the reason mentioned above, the SR4000 camera is not able to find sufficient correspondences. The Kinect camera with the RANSAC algorithm has the lowest error, both translational and angular, even for $48°$ of rotation.

As a qualitative observation, we can conclude that both cameras give similar results. The SR4000 camera with the RANSAC method has an error, after SLAM, which is bigger than that given by the other combinations, as can be seen in Figure 11. This comes from the fact that, in two consecutive frames of that sequence, the registration error was too high to be minimized by the TORO method. If the error between two consecutive frames is high enough, SLAM techniques may be



	8	16	24	32	40	48
KinectICP	0,028072959	0,031642067	0,0323251	0,044553127	0,40448499	0,659346185
KinectRansac	0,011363434	0,033299391	0,016506017	0,016691413	0,032334139	0,021242862
SR4000ICP	0,0088922	0,014129886	0,013070567	0,007308813		
SR4000Ransac	0,008105202	0,011176026	0,015686014	0,012433414		

Degrees



	8	16	24	32	40	48
KinectICP	0,641335947	1,890395165	2,033785876	4,545338084	8,694100944	34,8088978
KinectRansac	0,542538202	0,814084457	1,248452873	1,191398109	2,953975499	2,317598793
SR4000ICP	1,058368432	3,512173352	2,299124307	4,596016222		
SR4000Ransac	1,521031938	2,759566779	2,454298636	3,633744839		

Degrees

Figure 13: Translational (top) and angular (bottom) errors of both cameras and algorithms.

unable to find the correct solution. SLAM techniques minimize (propagating) the error of individual registrations. In the RANSAC solution, the registration in one frame had a very high error, and this was the source of error in the TORO method.

6. Conclusions and Future Work

In this paper, we have presented a study of two 3D cameras, and compared them using two new and complete SLAM methods. We also presented a new method for error reduction, based on GNG, which can be applied when the subjacent camera error is high. Both cameras provide 3D information together with 2D images. We combined this information, extracting 2D SIFT features and attaching them in 3D, both with the GNG information and directly using the 3D data provided by the camera. GNG enables a reduction of the input data, while preserving its structure. This gives us two advantages. First, we have to process fewer points, speeding up the next step of feature extraction. Second, the number of outliers is reduced. Outliers are one of the main sources of error in this kind of application.

With these features, we have presented two egomotion methods, one based on RANSAC and the other on ICP. We compared both cameras in a real scenario, calculating the egomotion of a sequence of movements performed by the robot. The results show the validity of both cameras as a SLAM tool. We measured the errors produced at different angular transformations. In these experiments, the best camera was the Kinect, and the best algorithm was RANSAC.

Unlike the classical ICP method, our methods were able to find correct solutions. In fact, ICP was unable to find even a close solution, so the TORO method was also unable to find the correct solution. Our methods not only found solutions, but also provided combinations which can be speeded up. GNG allows 3D error minimization, and also provides an easy parallelization, presenting a significant advantage over other methods.

In future work, we plan to extract and test other visual features (SURF, MSER, etc.) to determine which are optimal for each camera. We also plan to accelerate

the calculation time for the GNG algorithm, using GPU or other methods.

Acknowledgments

This work has been supported by grant DPI2009-07144 from Ministerio de Ciencia e Innovacion of the Spanish Government, University of Alicante projects GRE09-16 and GRE10-35, and Valencian Government project GV/2011/034.

References

[1] INTEL THREADING BUILDING BLOCKS 4.0 OPEN SOURCE. http://threadingbuildingblocks.org/., Intel, 2012.

[2] Mesa imaging. sr4000 camera., 2012.

[3] Microsoft corp. redmond wa. kinect for xbox 360., 2012.

[4] A. Almhdie, C. Léger, M. Deriche, R. Lédée, 3d registration using a new implementation of the icp algorithm based on a comprehensive lookup matrix: Application to medical imaging, Pattern Recogn. Lett. 28 (2007) 1523–1533. doi:10.1016/j.patrec.2007.03.005.

[5] L. Armesto, J. Minguez, L. Montesano, A generalization of the metric-based iterative closest point technique for 3d scan matching, in: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 1367–1372. doi:10.1109/ROBOT.2010.5509371.

[6] H. Bay, A. Ess, T. Tuytelaars, L.V. Gool, Surf: Speeded up robust features, Computer Vision and Image Understanding 110 (2008) 346–359.

[7] P. Besl, N. McKay, A method for registration of 3-d shapes, IEEE Trans. on Pattern Analysis and Machine Intelligence 14 (1992) 239–256. doi:10.1109/34.121791.

[8] A. Bhattacharjee, G. Contreras, M. Martonosi, Parallelization libraries: Characterizing and reducing overheads, ACM Trans. Archit. Code Optim. 8 (2011) 1–29. doi:10.1145/1952998.1953003.

[9] J.M. Carranza, A. Calway, W. Mayol-Cuevas, Enhancing 6d visual relocalisation with depth cameras, in: International Conference on Intelligent Robots and Systems IROS, IEEE, 2013.

[10] A. Censi, An icp variant using a point-to-line metric, in: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pp. 19–25. doi:10.1109/ROBOT.2008.4543181.

[11] Y. Chen, G. Medioni, Object modeling by registration of multiple range images, in: G. Medioni (Ed.), 1991 Proceedings., IEEE International Conference on Robotics and Automation, 1991., pp. 2724–2729 vol.3.

[12] A.M. Cretu, E. Petriu, P. Payeur, Evaluation of growing neural gas networks for selective 3d scanning, in: Robotic and Sensors Environments, 2008. ROSE 2008. International Workshop on, pp. 108 –113. doi:10.1109/ROSE.2008.4669190.

[13] M. Dissanayake, M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, M. Csorba, A solution to the simultaneous localization and map building (slam) problem, Robotics and Automation, IEEE Transactions on 17 (2001) 229–241.

[14] J. Elseberg, D. Borrmann, A. Nuchter, 6dof semi-rigid slam for mobile scanning, in: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 1865–1870. doi:10.1109/IROS.2012.6385509.

[15] M.I. Fanany, I. Kumazawa, A neural network for recovering 3d shape from erroneous and few depth maps of shaded images, Pattern Recogn. Lett. 25 (2004) 377–389. doi:10.1016/j.patrec.2003.11.001.

[16] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (1981) 381–395. doi:http://doi.acm.org/10.1145/358669.358692.

[17] D. Fiser, J. Faigl, M. Kulich, Growing neural gas efficiently, Neurocomputing 104 (2013) 72 – 82. doi:http://dx.doi.org/10.1016/j.neucom.2012.10.004.

[18] B. Fritzke, A growing neural gas network learns topologies, A Growing Neural Gas Network Learns Topologies, volume 7, MIT Press, 1995, pp. 625–632.

[19] Y. Gao, J. Tang, R. Hong, S. Yan, Q. Dai, N. Zhang, T.S. Chua, Camera constraint-free view-based 3-d object retrieval, IEEE Transactions on Image Processing (2012) 2269–2281.

[20] Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai, 3-d object retrieval and recognition with hypergraph analysis, Image Processing, IEEE Transactions on 21 (2012) 4290 – 4303. doi:10.1109/TIP.2012.2199502.

[21] Y. Gao, M. Wang, Z. Zheng-Jun, Q. Tian, Q. Dai, N. Zhang, Less is more: Efficient 3-d object retrieval with query view selection, Multimedia, IEEE Transactions on 13 (2011) 1007 – 1018. doi:10.1109/TMM.2011.2160619.

[22] A. Gil, O.M. Mozos, M. Ballesta, O. Reinoso, A comparative evaluation of interest point detectors and local descriptors for visual slam, Mach. Vision Appl. 21 (2010) 905–920.

[23] S.B. Gokturk, H. Yalcin, C. Bamji, A time-of-flight depth sensor - system description, issues and solutions, in: CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 3, IEEE Computer Society, Washington, DC, USA, 2004, p. 35.

[24] G. Grisetti, C. Stachniss, W. Burgard, Non-linear constraint network optimization for efficient map learning, IEEE Transactions on Intelligent Transportation Systems 10 (2009) 428–439.

[25] S. Guadarrama, A. Ruiz-Mayor, Approximate robotic mapping from sonar data by modeling perceptions with antonyms., Information Sciences 180 (2010) 4164–4188.

[26] J. He, H. Gu, Z. Wang, Multi-instance multi-label learning based on gaussian process with application to visual mobile robot navigation, Information Sciences 190 (2012) 162 – 177. doi:http://dx.doi.org/10.1016/j.ins.2011.12.015.

[27] Y. Holdstein, A. Fischer, Three-dimensional surface reconstruction using meshing growing neural gas (mgng), Vis. Comput. 24 (2008) 295–302. doi:10.1007/s00371-007-0202-z.

[28] I. Ivrissimtzis, W.K. Jeong, H.P. Seidel, Using growing cell structures for surface reconstruction, in: Shape Modeling International, 2003, pp. 78 – 86. doi:10.1109/SMI.2003.1199604.

[29] O. Koch, S. Teller, Wide-area egomotion estimation from known 3d structure, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition CVPR '07, pp. 1–8. doi:10.1109/CVPR.2007.383027.

[30] T. Kohonen, Self-Organising Maps, Springer-Verlag, 2001.

[31] W. Liu, Y. Zhang, S. Tang, J. Tang, R. Hong, J. Li, Accurate estimation of human body orientation from rgb-d sensors, Cybernetics, IEEE Transactions on 43 (2013) 1442–1452. doi:10.1109/TCYB.2013.2272636.

[32] D.G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2004) 91–110.

[33] R. Mendona Ernesto Rego, A.F.R. Araujo, F. de Lima Neto, Growing self-reconstruction maps, Neural Networks, IEEE Transactions on 21 (2010) 211–223. doi:10.1109/TNN.2009.2035312.

[34] S. Park, S. Kim, M. Park, S.K. Park, Vision-based global localization for mobile robots with hybrid maps of objects and spatial layouts, Information Sciences 179 (2009) 4174 – 4198. doi:http://dx.doi.org/10.1016/j.ins.2009.06.030.

[35] K. Pulli, Multiview registration for large data sets, in: Proc. Second International Conference on 3-D Digital Imaging and Modeling, pp. 160–168. doi:10.1109/IM.1999.805346.

[36] R. do Rego, A. Araujo, F. de Lima Neto, Growing self-organizing maps for surface reconstruction from unstructured point clouds, in: Neural Networks, 2007. IJCNN 2007. International Joint Conference on, pp. 1900 – 1905. doi:10.1109/IJCNN.2007.4371248.

[37] S. Rusinkiewicz, M. Levoy, Efficient variants of the icp algorithm, in: Proc. Third International Conference on 3-D Digital Imaging and Modeling, pp. 145–152. doi:10.1109/IM.2001.924423.

[38] J. Salvi, C. Matabosch, D. Fofi, J. Forest, A review of recent range image registration methods with accuracy evaluation, Image Vision Comput. 25 (2007) 578–596.

[39] Z. Shi, Z. Liu, X. Wu, W. Xu, Feature selection for reliable data association in visual slam, Machine Vision and Applications 24 (2013) 667–682.

[40] G. Trivino, L. Mengual, A. van der Heide, Towards an architecture for semi-autonomous robot telecontrol systems, Information Sciences 179 (2009) 3973–3984. doi:10.1016/j.ins.2009.08.007.

[41] D. Viejo, J. Garcia, M. Cazorla, D. Gil, M. Johnsson, Using {GNG} to improve 3d feature extractionapplication to 6dof egomotion, Neural Networks 32 (2012) 138 – 146. doi:http://dx.doi.org/10.1016/j.neunet.2012.02.014.

[42] D. Viejo, C. Miguel, A robust and fast method for 6dof motion estimation from generalized 3d data, Autonomous Robots (2014). doi:10.1007/s10514-013-9354-z, accepted for publication, 2013.

[43] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, International Journal of Computer Vision 13 (1994) 119–152.