Implementation of a Simultaneous Localization and Mapping system using Growing Neural Gas

Authors:
Alex Goldhoorn, 1434284
Hans Stadman, 1403982
Herman Eldering, 1437135

December 2006

Supervisors:
Dr. B. de Boer
Drs. G. Kootstra

Artificial Intelligence
Rijks*universiteit* Groningen

RuG

Abstract

Simultaneous Localization and Mapping (SLAM) is concurrently gathering features of the environment and building a map accordingly, to determine your current position. We applied this technique on a Pioneer robot using Growing Neural Gas (GNG). GNG is an algorithm which is comparable to Kohonen, but it is capable of dynamically adding or removing nodes from the neural network.

We set out to investigate how well this system of SLAM and GNG will perform when information from camera images, sonar and odometry sensors is used. None of these sensors give precise or complete information, but our aim is that they might be able to fill each other's gaps.

We show that our system performs best when information from all the sensors is used, but also that when sonar is not used, the system still performs well. The system uses data that was gathered before learning occurred and is therefore not yet a true SLAM system. Future research can improve the system by collecting features from the data (using computer vision techniques to extract features from the camera images) and by implementing it as a true SLAM system.

1 Introduction

An autonomous robot is capable of performing its task without human intervention. When the tasks of the autonomous mobile robot require it to navigate through an environment, this environment should not be pre-mapped onto the robot, which would limit its autonomy (Fehrmann, 2002) and would also make human intervention necessary when the environment changes. This means that the robot has to make its own map of the environment and use that map to complete its tasks. This process is called Simultaneous Localization and Mapping (SLAM).

The problem with SLAM is stated as follows: an autonomous mobile robot is moving around collecting a set of sensor measurements of features of the unknown environment. This sensor data needs to be used by the robot to make an estimate of its current position, while concurrently building a map of the environment. The difficulty of this problem lies in the fact that the robot needs an accurate measurement of its trajectory in order to make a good map. This trajectory is calculated from the odometry measurements of the robot, which will become inaccurate over time unless they can be corrected using a precise map (Tard´os et al. 2005). Since the object was to make this map in the first place, we cannot rely on the odometry measurements as our only means of position determining.

The challenge lies in having a mobile robot learn its dynamic environment using only onboard sensors. The sensors that will be used in this experiment are sonar sensors for approximate distances, video images for landmark recognition and odometry measurements for approximate coordinates. Because none of these sensors give precise or complete information about the environment, they have to be combined in the learning process of the robot. In this paper we will present the learning approach we implemented on our robot, which uses an adapted version of the implementation of the Growing Neural Gas (GNG) algorithm by Van der Ploeg (2005; originally by Fritzke, 1995).

Our research question is as follows:

> *How can we use the GNG algorithm to create a robust SLAM system using a combination of different sensors (such as a camera image, sonar sensors and odometry)?*

To answer the above research question we will show how the GNG algorithm functions in the SLAM process. We will also show results of the performance and the robustness of the SLAM system.

2 Related work

Much work in the field of AI has been conducted on robot navigation. However, since all real world environments are dynamic, there is much more interest in the SLAM technique for the navigation of robots. Rudolf Fehrmann (2002) has done research very similar to ours, by implementing a navigational behaviour seen in insects. His research shows that a Self Organizing Map is capable of learning important landmarks in the environment, even during a live run under varying lighting conditions. The robots in his experiment were also able to make a local topological landmark map of the environment, which would theoretically enable the robot to plan a path between landmarks.

Van der Ploeg (2005) implemented an adaptive learning scheme which was based on GNG. A Kohonen network has a fixed number of nodes, but a GNG network does not. The GNG algorithm 'grows' (i.e. increases) the network by adding nodes to places where there is a high error. The high error occurs because the Euclidean distance to the input data at that part in the network is still relatively high. This algorithm keeps adding nodes until the overall squared distance between the nodes and the input features are considered small enough. The learning mechanism as implemented by Van der Ploeg (2005) can only be used off-line as it uses a distance matrix consisting of the (average) locations of landmarks which was constructed from all the locations stored during the recognition run of the robots. As input data, Van der Ploeg uses three snapshots of each landmark that is recorded, a frontal view, and views slightly to the left and right of the landmark.

Van der Ploeg dedicated most of his project to path planning, which enabled the robot to efficiently move between landmarks. His implementation however, only allowed for 'blind' travelling. Once the robot chose its next destination it would not keep track of the current goal while moving. This means that when the avoidance behaviour of the robot moved it from its current heading, the goal would be missed. Missing the goal can also occur because of errors in the odometry measurements. We therefore link our research mostly to that of Fehrmann (2002), but with the addition of implementing a GNG network. We will also make use of the same type of robot and work in the same environment.

In Wolf and Sukhatme (2004) a model is presented that also uses landmarks for SLAM. In contrast to the research of Fehrmann and Van der Ploeg a laser range finder is used instead of mainly relying on visual sensor data. In their paper Wolf and Sukhatme focus on the aspect of static and dynamic parts in the environment. While the model can create good maps from the environment the use of a range finder is not always possible and a visual system offers some benefits like its price and the number of ways and places it can be used.

An historic and technical overview of robotic mapping can be found in Thrun (2002). However this survey heavily focuses on the use of range data and associated techniques, such as the popular Kalman filters and expectation maximization.

3 Method

3.1 Experimental setup

Robot specifications

The robot used for this experiment is the ActivMedia Pioneer 2 DX robot, which moves on two wheels with a small turntable wheel at the back, enabling it to turn in tight corners. The robot has a length of 44 cm, a width of 33 cm, a total height of 27 cm and it weighs 9 kg.

The robot has an array of eight sonar sensors at the front and the back of the robot, which provides information about the distance of objects to the robot. Figure 3.1 shows a schematic overview of the sonar sensors on the robot. The sonar fires at 25 Hz and its sensitivity ranges from 10 cm to more than five meters. Dead-reckoning and speed sensing is done by a high-resolution optical quadratic shaft encoder that provides 19 ticks per millimeter. The robot also has a color CCD camera mounted on the top with an effective resolution of 440,000 pixels.

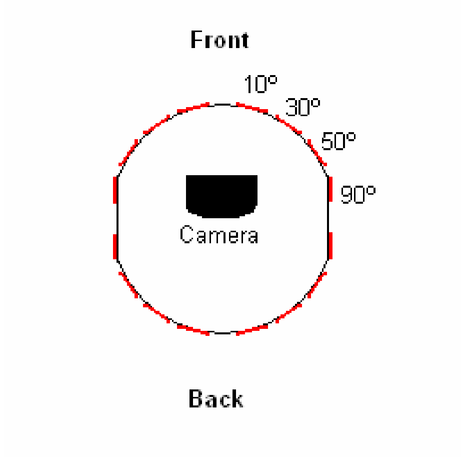


Figure 3.1: A schematic top view of the Pioneer 2 robot, showing the position of the camera and the sonar sensors. The robot has 16 sonar sensors which are indicated by the red lines.

Environment

The environment for our experiments is the Robocup soccer field in the old robotics lab of the Artificial Intelligence department of Rijksuniversiteit Groningen. This field is located in a closed room with windows on one side. The field consists of a green floor with white boarding around with a height of 40 cm. The surface area of the field is approximately 4 by 4.6 meters. There are two goals placed opposite of each other on the short sides of the field, one is blue and the other is yellow. A schematic of the field is given in figure 3.2, which was taken from Van der Ploeg (2005).

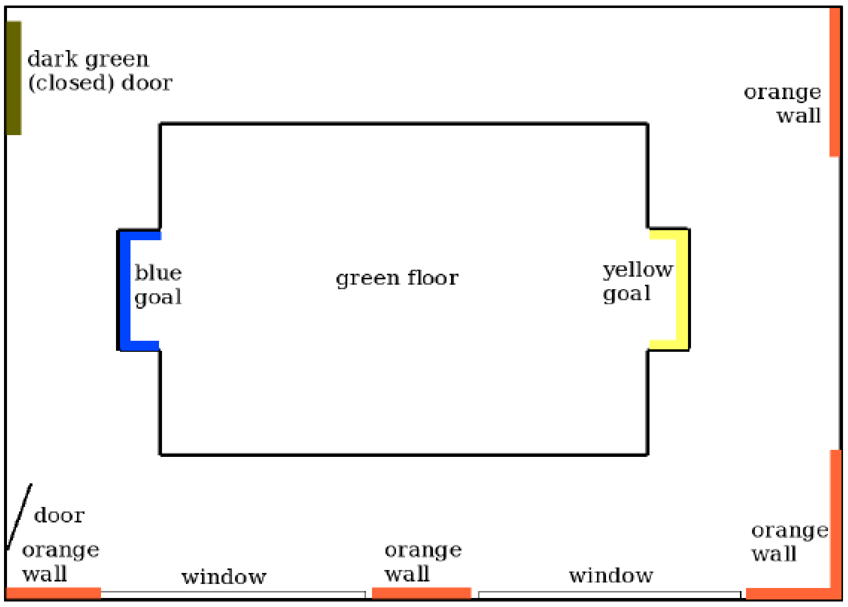


Figure 3.2: Schematic overview of the robotics lab, showing the environment of the robot. Taken from Van der Ploeg (2005).

3.2 Data Gathering

During the testing of our SLAM algorithm we used a fixed data set. This data was acquired by recording sensor values while the robot was driving on the field of the robotics lab. The driving algorithm for the robot consisted of only a few conditions. The robot would continuously drive forward until the collision-detection module indicated a nearby object, causing the robot to randomly change its heading. The collision detection was done by using a threshold for the sonar values. This threshold ranged from 1100 mm for the front sonars to 150 mm for the side sonars. To record distinctive snapshots, the sonar values were determined such that the robot would be close enough to the objects to record an accurate position, but still maintain enough distance to record an informative image of the object. To cope with the noisiness of the sonar data we used a running average of the last three time steps of the sonar values. Once the collision-detection module indicated a near collision, a snapshot was taken and stored into the data file. This snapshot contained an image from the camera, all 16 sonar values, the odometry position and angle and a manually recorded position.

For the manual position, we divided the surface area of the robotics lab into a matrix of 8 by 8 sub-areas. Each sub-area of the surface was uniquely labelled with a letter and a number. Whenever the robot took a snapshot, the true location (one of the sub-areas) was manually written down in a log file. With this procedure, each snapshot location could be verified once the experimental run was completed. A few manual positions (10 out of 1006) were incorrectly written down, which were later corrected by averaging the previous and next positions of the robot.

3.3 Growing Neural Gas

During our experiment a large amount of data was collected, because the robot takes numerous image snapshots and stores both the sonar and odometry values at various places in the environment. To semantically relate items in the data we use a form of data clustering. Clustering is a very convenient tool in reducing large amounts of raw data by categorizing into smaller sets of similar items, where the items in each set share similar features. In our case, each cluster should be a set of snapshots belonging to a certain object in the environment. All clusters combined give us a topological map of the environment. The most common clustering algorithm is the K-means clustering algorithm by (MacQueen, 1967), other algorithms include Kohonen's Self Organizing Map (SOM) (Kohonen, 1982) and the Neural Gas algorithm by (Martinetz and Schulten, 1991).

The clustering algorithms mentioned above are not suitable to use in our experiment, because the algorithms use a preset number of nodes and it is simply not possible to determine how many nodes our network will need upon creation. Instead, we prefer a network that is able to grow in the number of nodes and is still able to cluster the data. An algorithm that is capable of this is an incremental clustering algorithm called Growing Neural Gas (GNG) by (Fritzke, 1995). With this method, you initially start out with a small network which is extended by successively adding new nodes, thereby evaluating local statistical measures gathered from previous adaptation steps. While the algorithm is fed with input data, it tries to minimize the error between the nodes in the network, by moving the nodes closer and by inserting new nodes. The error is the squared value distance between the input data item and the existing node in the network with the current best fit to that item. Figure 3.3 gives an example of how the GNG algorithm is capable of fitting a signal distribution which has different dimensionalities in different areas of the input space (Fritzke, 1995).

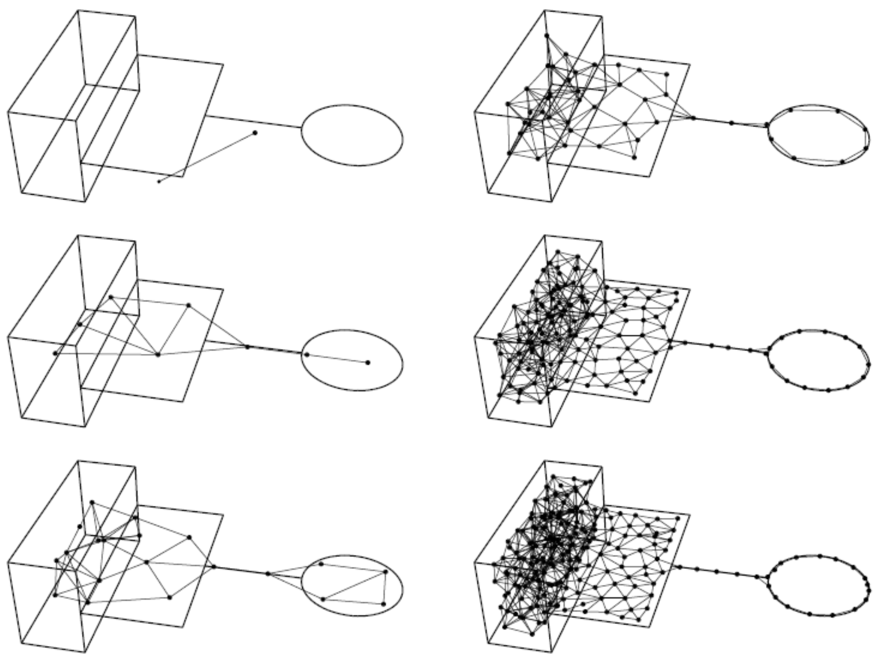


Figure 3.3: The Growing Neural Gas algorithm is able to adapt to a signal distribution with different dimensionalities in different areas of the input space by incrementing the number of nodes where the error between existing nodes and new input data is largest. This figure is taken from Fritzke (1995).

The model in (Fritzke, 1995) is described as follows.

Given an n-dimensional input space $\mathbf{R}^n$, the network consists of:

- a set A of nodes (initially two). Each node $c \in A$ has an associated reference vector $w_c \in \mathbf{R}^n$. These reference vectors can be regarded as positions in the input space of the corresponding nodes.
- a set N of connections (or edges) between pairs of nodes. The purpose of these connections is to define the topological structure of the network.

There is also a (possibly infinite) number of *n*-dimensional input signals obeying some unknown probability density function $P(\xi)$.

The algorithm for Fritzke's model contains the following steps:

0. Start with two units *a* and *b* at random positions w_a and w_b in $\mathbf{R}^n$.
1. Generate an input signal ξ according to $P(\xi)$.
2. Find the nearest node s_1 and the second nearest node s_2.
3. Increment the age of all edges emanating from s_1.
4. Add the squared distance between the input signal and the nearest node in the input space to a local counter variable:

$$\Delta\text{error}(s_1) = \left\| w_{s_1} - \xi \right\|^2$$

5. Move s_1 and its direct topological neighbours towards ξ by fractions ε_b and ε_n, respectively, of the total distance:

$$\Delta w_{s_1} = \varepsilon_b \left(\xi - w_{s_1}\right)$$
$$\Delta w_n = \varepsilon_n \left(\xi - w_n\right) \text{ for all direct neighbors } n \text{ of } s_1.$$

6. If s_1 and s_2 are connected by an edge, set the age of this edge to zero. If such an edge does not exist, create it.
7. Remove edges with an age larger than a_{max}. If this results in points having no emanating edges, remove them as well.
8. If the number of input signals generated so far is an integer multiple of parameter λ, insert a new node as follows:
    - Determine the node q with the maximum accumulated error.
    - Insert a new node r halfway between node q and neighbour f with the largest error variable:

      $$w_r = \frac{w_q + w_f}{2}.$$

    - Insert edges connecting the new node r with nodes q and f, and remove the original edge between q and f.
    - Decrease the error variables of q and f by multiplying them with a constant α. Initialize the error variable of r with the new value of the error variable of q.
9. Decrease all error variables by multiplying them with a constant d.
10. If a stopping criterion (e.g. network size or some performance measure) is not yet fulfilled go to step 1.

In step 5 of the GNG algorithm the network is adapted by moving the nearest node and its topological neighbours towards the input signal. This helps to reduce the error, which is calculated in step 4. Each λ^{th} iteration a new node is inserted to reduce the error.

3.4 Simultaneous Localization and Mapping

The SLAM algorithm we implemented uses the GNG algorithm. The nodes of the neural network represent certain places on the map. One or more sensor values or features can be used in a node, for example the sonar values and an image of the camera.

As mentioned before, the learning was first done offline, but it should be rather easy to adapt the program for online learning. This could be done by creating a robot behaviour which uses the sensor data directly to create a map and another behaviour that uses the map for navigation.

Features

The sensors of the robot give information about its environment. This data has to be exploited when trying to learn the location based upon that information. A problem with the sensor data is that it might contain too much diversity of information and therefore it may be too complicated for learning. One solution is to reduce the dimension of the data. For images this can be done by reducing the size of the images for example. For our experiments we reduced the image size and experimented with two different sizes.

Using rgb (red green blue) values for the images causes problems, especially when the lighting is different, which could render completely different rgb values. Therefore using hsv (hue saturation and value) for the images is a better choice. To test the differences, we used both hsv and rgb images in the experiments.

Besides the images from the camera, two other sensors were used, odometry for absolute position and sonar for distance measurements to nearby objects. Both sensors are affected by noise. It should be noted that the odometry sensor suffers from an accumulated noise, which means that the error increases with the distance travelled. By combining data from all the sensors, the effect of this accumulated error is reduced.

Learning

The map of the environment is learned and adapted in an unsupervised way. The used learning mechanism is Growing Neural Gas as discussed in a previous section. The nodes in our networks consisted of a combination of several sensor values and/or feature values, which are mentioned in the previous section. In this section we will refer to the features and the (unmodified) sensor values as features. Per network all the nodes contain the same types of features.

The maps were created by 'feeding' each data record (containing the feature information) to the GNG algorithm, which was done three times each run.

Theoretically all combinations of the features can be used per network. To be able to control the influence of the different features used in a node, a weight was added to each feature. This weight is used for calculating the distance between two nodes, which is required by the nearest neighbour function for example. The following formula calculates the squared distance between node 1 and 2:

$$d^2 = \sum_{i=1}^{n} w_i (f_{node1,i} - f_{node2,i})^2 \qquad (3.1)$$

In this formula d is the distance, n the number of features per node, w_i the weight of feature i, and $f_{node1,i}$ and $f_{node2,i}$ are the i^{th} feature values of node 1 and node 2 respectively. The squared distance between two features is calculated like in formula 3.1: $(f_{node1,i} - f_{node2,i})^2$ in the cases of a one dimensional value. In the case of a matrix, like for example an image, the distance calculation is a Euclidean distance.

The distance in the GNG algorithm is used to find the nearest neighbour of an input node. The nearest neighbour in the map is shifted towards the input signal (step 5 of the GNG algorithm). When several input signals have been processed (i.e. the map has been adapted to accommodate these signals), the nearest neighbour of an input signal should represent the position of the input signal (the place where the robot recorded the input signal) on the map. The (learned) odometry position can be used to create a metric map of the environment for illustrative purpose and to be able to compare it to the manually recorded positions, as done in the scoring mechanism.

Score

To be able to find the best map, all maps have to be compared to each other. In the GNG algorithm of Fritzke (1995) the mean error of the nodes is used to compare the networks. For each input signal the error of the nearest neighbour in the network is increased. The error in the (nearest neighbour) node is increased with the squared distance to the input signal. This error value is used in the algorithm to decide where in the network more nodes are required. This error may be a good indication of how well the GNG network has converged towards a network which contains all frequently occurring data. In this experiment the required output is known roughly from the manually recorded data. Therefore, using this manually recorded data as reference is a better indication of how good a map is.

The scoring mechanism uses manually measured positions. The field was divided in an 8 by 8 matrix, which gives cells of about 50×58 cm^2. In the scoring mechanism, the cells have a size of 1×1. This is no problem because the learned positions are scaled to the manual position before comparison. The drawback of our scoring mechanism is the size of the cells and the discrete character of the measurements.

When a network is created (i.e. a map is learned), the scoring was done as follows:

1. The yellow and blue goals are located in the learned network by finding the nearest neighbour of the specific goals in the data file, which are manually indicated. When the blue and yellow goal have the same nearest neighbour, a score of 0 is given to the network, because the next steps in the scoring mechanism do not make any sense in that case. If the goals share the same nearest neighbour, the map is considered useless.
2. The angle between a vertical line in the centre and the line between the yellow and blue goal is calculated. This is to adjust the orientation of the collected data points to the orientation of the manual data points
3. The scaling factor is calculated by using the minimum and maximum x and y values.
4. A random 20% of the records of the original sensor data and features are selected to test the learned network.
    a. Per record the nearest neighbour in the learned network is located.
    b. The position of the nearest neighbour is compared with the manual position from the record and a score is calculated for that point.
    c. The score is 1.0 (maximum) when the learned position is in the square of the manual position (as indicated in paragraph 3.1). Otherwise the score is 0.5 times the reciprocal of the distance between the learned point and the centre of the manual square, see formula 3.2:

    $$s = \frac{0.5}{\sqrt{(x_{man} + 0.5 - x)^2 + (y_{man} + 0.5 - y)^2}} \tag{3.2}$$

    Where s is the score, (x_{man}, y_{man}) the manual position and (x, y) the rotated and scaled learned position. The shortest distance from the centre of a manual square to its border is 0.5, the nominator of (3.1), which limits the score to 1.0.
5. The total score of the network is the mean score of the scores calculated in step 4.

As the score of the network is used to judge its performance, it is important to put the score into perspective. We do this by comparing the score of the network to a score which was obtained with random data as input. The score function was tested by generating ten million uniform distributed random positions to calculate the score. For the manual x and y integer values from [0, 7] and for the 'learned' x and y floating point values from [0, 8) were generated.

Figure 4.1 shows the distribution of the results, where the black line is the distribution of the scores from random input. The mean score is 0.18, the median score 0.12 and the standard deviation 0.16. The distribution has a peak at score 1.0, because the score is set to 1.0 when the learned position is inside the square of the manual position. The percentage of values having a score of 1.0 is 1.56%. There is no score of 0, because the maximum distance is $\sqrt{2 \times (7 + 0.5 - 0)^2} = \sqrt{112.5} \approx 10.61$ and therefore the minimum score is $0.5 / \sqrt{112.5} \approx 0.047$.

To conclude, the scoring mechanism, as described in this section, is a way to evaluate the learned maps by comparing them to the real world map (created from the manually recorded data). In the results section the score will be used to compare the learned maps. In general the scores of the learned maps can be compared to the scores when the input to the score function is random. This last point is important to find out whether the mapping algorithm works better than a mapping 'algorithm' which guesses the position.

Finding good parameter values

The quality of the learned networks depends, among others, on the learning parameters. The learning parameters for the GNG network are mentioned in the section discussing GNG. These parameters need to be set to an optimal value. To find this optimal value one can simple try every possible value within a certain range and also in combination with values of other parameters. This method will most likely result in very good values (if possible for the data), but it is very time consuming. The time to try all parameter values and combinations grows exponential with the number of parameters to estimate.

A more efficient way to search for the best parameter values for learning is by using a maximum likelihood estimator. During this experiment, a method, based on Simulated Annealing (Kirkpatrick et al., 1983), is used to find the optimal values for the learning parameters of the GNG network and the weights. Simulated Annealing is based on thermodynamics and it uses energy and temperature to describe the state of the search space.

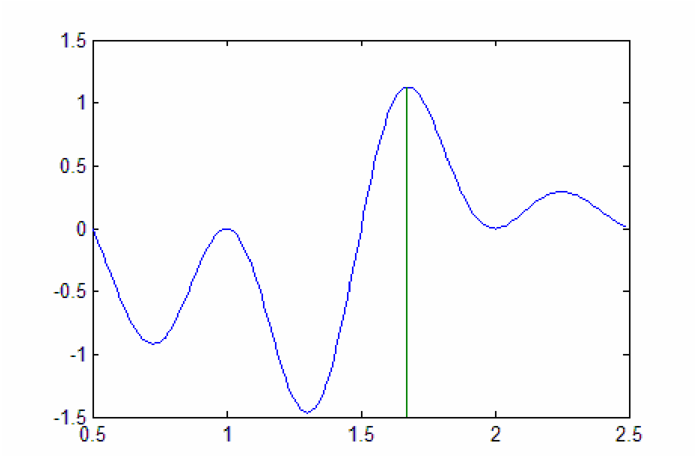


Figure 3.4: Explains optimal parameter search by Simulated Annealing. In this case, the optimal parameter value is at $x = 1.68$, but a suboptimal value could be found at $x = 1$.

Our algorithm can be described by the following steps:

1. Initialize all parameter values to a random value within 100% of their range.
2. Create a network (i.e. map) using the randomized parameters and calculate the score of the network.
3. Now decrease the range from 100% to less than 1% in *n* steps. Where a higher *n* will make the way to the optimum values more smooth and therefore the chance of reaching a suboptimum will be smaller (see figure 3.4).
    a. Create a network and calculate the score using the current parameter values.
    b. Compare the scores of the current network and the previous created network. Use the parameter values of the network with the highest score for the next step.
    c. Change the parameter values to a random value within a smaller range around the parameter values of the best network (from step 3b).

Table 3.1 shows the parameter values found by the Simulated Annealing algorithm and the range that was used by the algorithm.

Parameter	Optimal range	Search space
a_{max}	95 – 110	1 – 1000
λ	450 – 700	1 – 1000
α	0.4 – 0.6	0 – 0.9
d	0.57 – 0.63	0.1 – 0.9
n_{max}	1250 – 3000	5 – 10,000
ε_b	0.64 – 0.68	0.001 – 0.9
ε_n	0.045 – 0.047	0.001 – 0.9
w_{image}	0.62 – 0.64	0 – 1
w_{odo}	0.3 – 0.355	0 – 1
w_{sonar}	0 – 0.5	0 – 1

Table 3.1: The optimal values and the search space for the learning parameters found by our Simulated Annealing algorithm. The parameter n_{max} is the maximum number of nodes, w_{image}, w_{odo} and w_{sonar} are the weights of the image, odometry and sonar data respectively. The other parameters are from the GNG algorithm.

4 Results

In this section some results will be discussed using the techniques described in the previous sections. Since there is a huge amount of possible parameter values and combinations, a simulated annealing method was used to find good (preferably optimal) values. All experiments were done off-line to find the optimum values for the learning parameters and to find out which features can be used best. For the experiments, one data run was used which consisted of 1006 records. These records contained information from all sensors and the manual position. Due to human error, this set contained (at least) ten incorrect manual positions. These errors should not have a big effect on the score, because per learning session about 1000 to 8000 maps were created by training, each time with slightly different parameters.

The learning algorithm was tested with different input sets, which were extracted from the collected data set. The sets differ in image size, image format (hsv or rgb) and the use of sonar. These sets are shown in table 4.1. The learning parameter ranges used are listed in table 3.1

Set	Features				Score		
Number	*Image size*	*Image hsv/rgb*	*Odometry*	*Sonar*	*Mean*	*Confidence interval (α=0.001)*	*Sample count*
1	40×30	hsv	Yes	No	0.230	2.537×10^{-5}	7991
2	40×30	hsv	Yes	Yes	0.227	2.564×10^{-5}	7991
3	40×30	rgb	Yes	No	0.195	3.717×10^{-5}	7126
4	40×30	rgb	Yes	Yes	0.171	3.679×10^{-5}	7911
5	80×60	hsv	Yes	No	0.227	2.517×10^{-5}	7991
6	80×60	hsv	Yes	Yes	0.224	2.481×10^{-5}	7991
random	-	-	-	-	0.175	2.259×10^{-5}	10^7

Table 4.1: The contents of the data sets and the scores of the created maps. The mean and confidence interval of the scores are calculated after bootstrapping, the results are plotted in figure 4.2. The sample count is the number of scores in the set. All the sets did contain 7991 scores, but the 0 scores were filtered out, which only occurred in sets 3 and 4.

Score analysis

From the results the 0 scores were filtered out, these are given to maps where the blue and yellow goal are on the same position (node) on the created map. From these maps a score cannot be calculated and therefore they cannot be compared to other maps, which have a score based on the distance to the manual position.

Figure 4.1 shows the scores of the baseline and of the sets that were described above. The figure shows that the scores are not normally distributed. To be able to compare the results of the different sets with the baseline and with each other, we used the Wilcoxon Rank-Sum Test statistic and bootstrapping.

At first we want to know if the learned map is better than randomly guessing the position. All sets were significantly different (*p* << 0.001) from the random set according to the Rank-Sum Test. The results can be seen in table 4.1 and figure 4.1. Those results also show that all the sets have a better mean score than the scores from the random set.

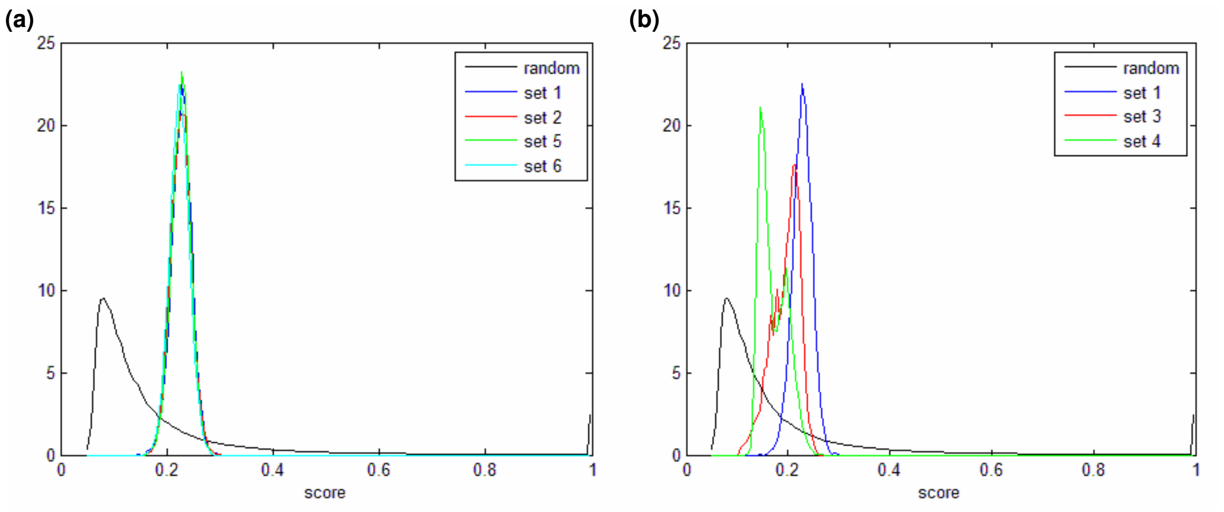


Figure 4.1: The distributions of the scores of the used data sets and the baseline (random score. The plotted distributions are histograms with 150 bins. a) Compares the sets that performed best with the baseline. b) Compares the sets with the worst performance to set 1 (best performance) and to the baseline.

Bootstrapping is a method to provide a confidence level for the means that were extracted from the data in figure 4.1. This method randomly selects a number of data points (10,000 for the sets and 1,000,000 for the baseline) and calculates the mean over those points. By also performing this operation a number of times (1000 for our data), the result is a set of means. This set of means is always normally distributed, if the number of selected data points is large enough. The mean and confidence interval (with α=0.001) of these bootstrapped sets are shown in table 4.1 and figure 4.2. Only set 4 is significantly worse than the random set (bar 0 in figure 4.2).

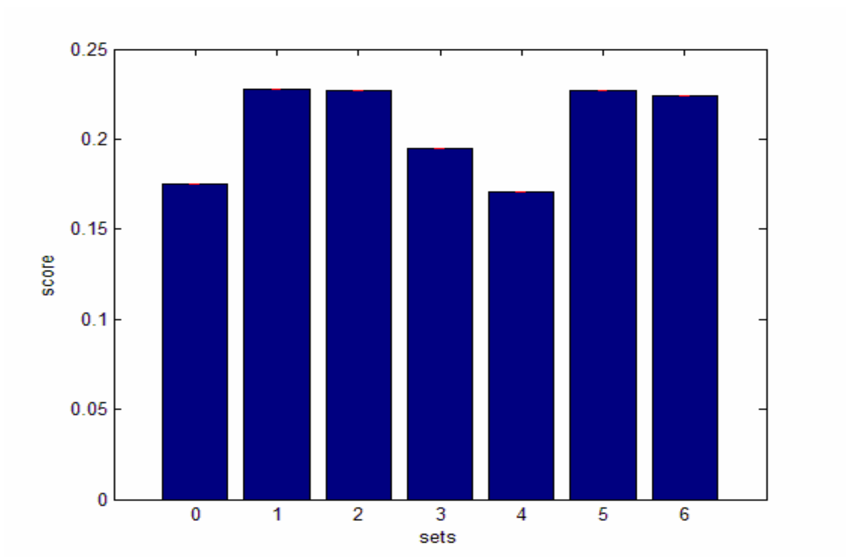


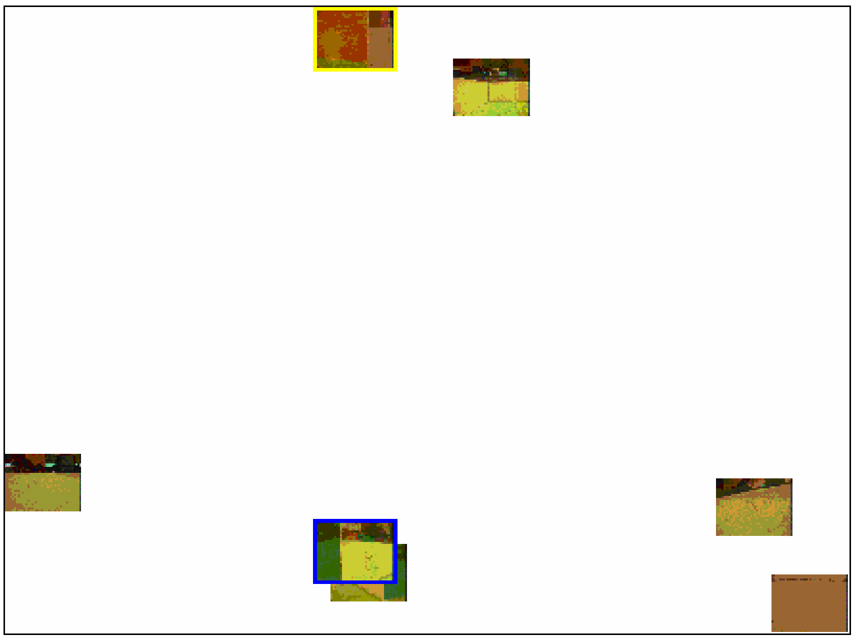
Figure 4.2: The mean scores of the sets with 0 as the baseline after bootstrapping. The confidence interval is also shown ($\alpha = 0.001$) in red, but these values are so small that they are hardly visible. Table 4.1 shows the values of the mean and the confidence interval.

Besides comparing the sets to the baseline, we can also make a comparison between the sets. Only the results of set 2 are not significantly different from set 5 ($p \approx 0.22$). The rest of the sets are significantly different from each other ($p << 0.001$) according to the Rank-Sum Test. According to the bootstrapping method, all the sets differ significantly. The best mean scores (after bootstrapping) are from high to low, set 1, 5, 2, 6, 3 and bellow the baseline, set 4.

The maps

Figure 4.3 shows maps created with parameter settings from set 1 and 5. The maps only contain seven and six nodes respectively. This low number of nodes is a result of the Simulated Annealing which found the 'optimum' number of nodes to be approximately six nodes. This amount depends on the parameter λ, which specifies after how many input signals a new node has to be added. The λ was set between 450 en 700 and the amount of records used as input was $3 \times 1006 = 3018$ records, therefore the number of nodes should be between four and seven. Appendix A shows a map with more nodes because of a lower λ.

The images on the maps in figure 4.3 are quite sharp, this is due the high ε_b. Step 5 of the GNG algorithm moves the input signal towards the nearest neighbour in the network with a fraction of ε_b. Appendix A shows a map with a lower ε_b, and where can be seen that the images are less sharp.


(a) A map created by using the parameter settings from set 1.

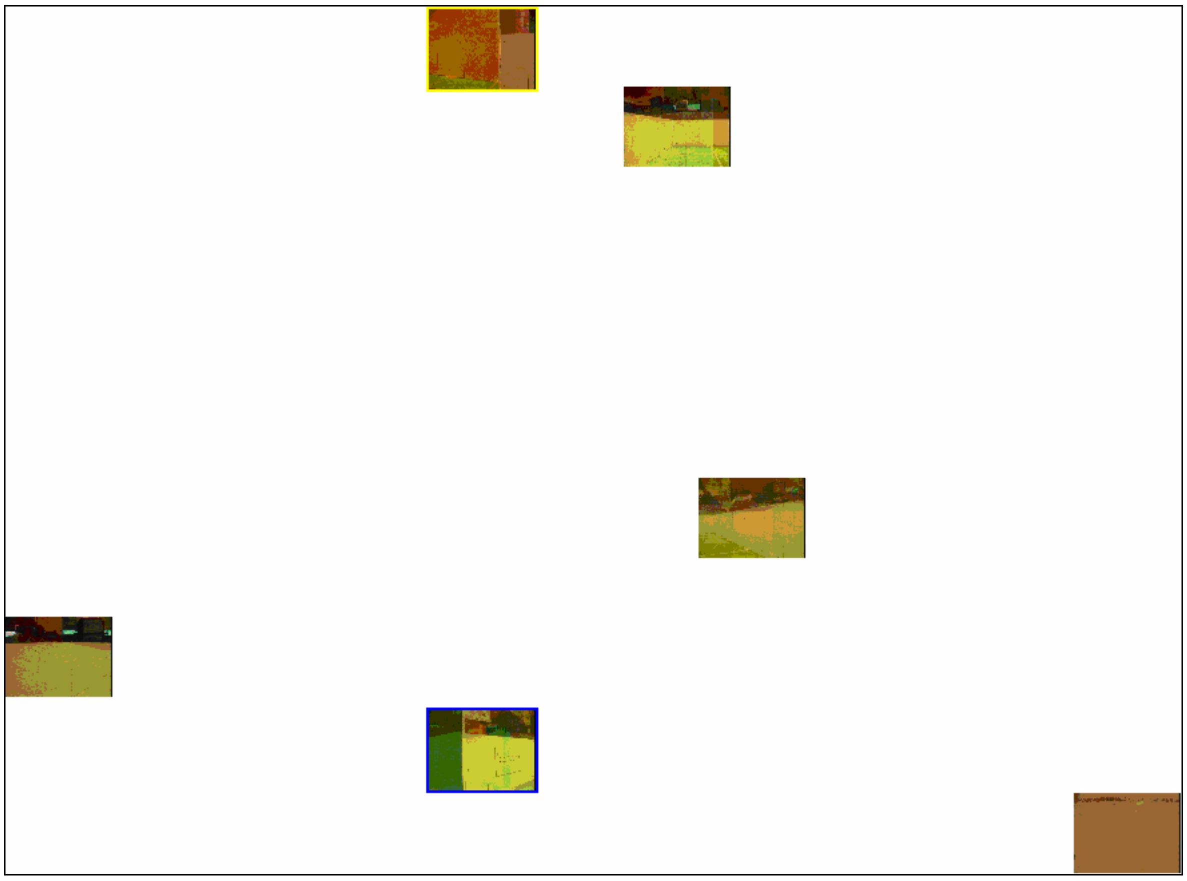

(b) A map created by using the parameter settings from set 6.

Figure 4.3: The maps show the images and their relative position after rotation and scaling, as discussed in the score section (steps 1 – 3). The yellow goal (shown by a yellow rectangle) is straight above the blue goal (shown by a blue rectangle) due to the rotation. The images were converted from hsv to rgb to make them better recognisable. The pictures are scaled, so the distances between nodes cannot be compared between the two maps and the images were painted on the map with the left bottom top as location of the node.

After analysing the maps we found out that they need to be flipped horizontally, because the images at the left show partly opened windows, but these should be at the right side of the map. An inspection into the processing of the images did not indicate any horizontal flipping, so we were not able to solve this problem. The output of the GNG algorithm should thus be flipped horizontally in order to obtain a map which is true to the real environment.

5 Discussion

In the previous section we have shown that our implementation of a SLAM system using a GNG algorithm is capable of learning its unknown (though controlled) environment. However, the system is not yet a true SLAM system since the learning was done after the data had been collected by a robot. Our system could become a true SLAM system by feeding each collected data sample directly to the learning algorithm instead of saving it in a data file first.

Although the environment is learned fairly well, we believe it can be greatly improved by using better input data for the learning mechanisms of the system. For instance, since the data obtained from the sonar sensors is very noisy, a running average should be used for learning instead of the actual current value from the sensor. Learning could also be improved by handling the images in a different way. Currently, learning from the input images consists of comparing the pixel values of the input image to the images of the nodes already in the network. This comparison would most probably be much better if some pre-processing occurred on the images that allows features to be extracted from them. Images can be considered equivalent by the learning mechanism if they contain (roughly) the same features. There are many methods in the field of computer vision that can be applied to extract features from images, such as line or edge detection and texture extraction. Implementing some of these methods in the system would probably yield much better results.

There were also some design choices that reduced the complexity of the system, but restricted its potential as a result. For instance, the scoring algorithm calculates the distance to the centre of a manual square when it is not located within that square. If the distance would be calculated to the border of the square instead, a more accurate distance would be obtained, since points near the corner of the manual square now have a greater distance than points near the middle of a border of the square. This calculation would be a bit more complex though.

Initially, the score of the network was calculated, similar as in the GNG algorithm section, by evaluating the mean error, which was based on the distance between the nodes in the network and the new input samples. This technique was adopted from the research work by Van der Ploeg (2005). The downside of evaluating the network like this is that there is very little feedback on whether the algorithm performs properly on the data samples. As an improvement, we performed an experiment in which we stored both the data samples from the robot and manual data samples. This means that for each data sample, we could now verify where that sample should be placed on the map and where the scoring algorithm actually placed it (which node). However, the results we obtained from training the system on these data samples are biased since we used only one data set. The performance of the system could be more accurately tested by training it on one data set and testing it with a different data set.

6 Conclusion

The first step in creating the SLAM system was to pre-process the sensor values so that they could be used as input to the GNG algorithm. First of all the camera images were reduced in size and converted to hsv. The next step was to combine the sensor data in samples. These samples can then be used as input to the GNG algorithm. Weights were added to each sensor type to be able to let some types be more important than others. The GNG algorithm (as shown in section 3.3) uses the weights when an input sample is merged with an existing node in the network. The more input the network has had, the more the network converges.

The GNG algorithm has a number of parameters that determine its learning behaviour. We used a Simulated Annealing algorithm to find the best settings for these parameters. We conducted several experiments on the data that we had collected, which consisted of 1006 samples. The results show that our system is better than the baseline, in which the position is randomly determined. The only situation in which the system performed worse than the baseline was when it contained the scaled rgb images and sonar (set 4). The best (using bootstrapping to compare the results) was set 1, which uses hsv images of a small (40×30) size and no sonar. Although the sonar data did not improve the results, this can be a result of the noisiness of the sonar sensor. The learning might be improved if not the current value of the sonar sensor would be used, but a running average instead.

Although the results already show a better performance than the baseline, some major improvements can still be made. One of the most important improvements would be to use the relative distance to the previous point in order to overcome the accumulating error in the odometry. Its current (x,y) position could in that case be calculated based on the coordinates in the learned map and the distance to its previous point. This (x,y) position makes the map a more metric map, while it still is a topological map. To test a real topological map, it would be better to try to use it on a robot and see if it is able to find a path while interacting with its environment. This does however require a path planning behaviour. A second great improvement could be to extract features from the camera images instead of comparing images pixel wise. The only pre-processing we performed on the images was scaling and conversion from rgb to hsv.

In our experiments we used only one data set for both learning and testing, which was due to the extensive amount of work to create the manual data set. In a future experiment more data sets have to be generated, so that there are separate sets for both learning and testing. This would give a more accurate performance rating of the learning capabilities of the system.

Through our research, we have shown that the GNG algorithm is a very good learning tool to incorporate in a SLAM system. We are confident that with the proposed adjustments, this system could perform even better.

Appendix A: More maps

This appendix shows two more maps from learned networks. These maps did not have the highest score, but they are added to show the influence of some parameters.

Figure A.1 shows a map with more nodes than the maps with the best scores (figure 4.3). This was reached by decreasing λ, as can be seen in step 8 from the GNG algorithm (section 3.3).

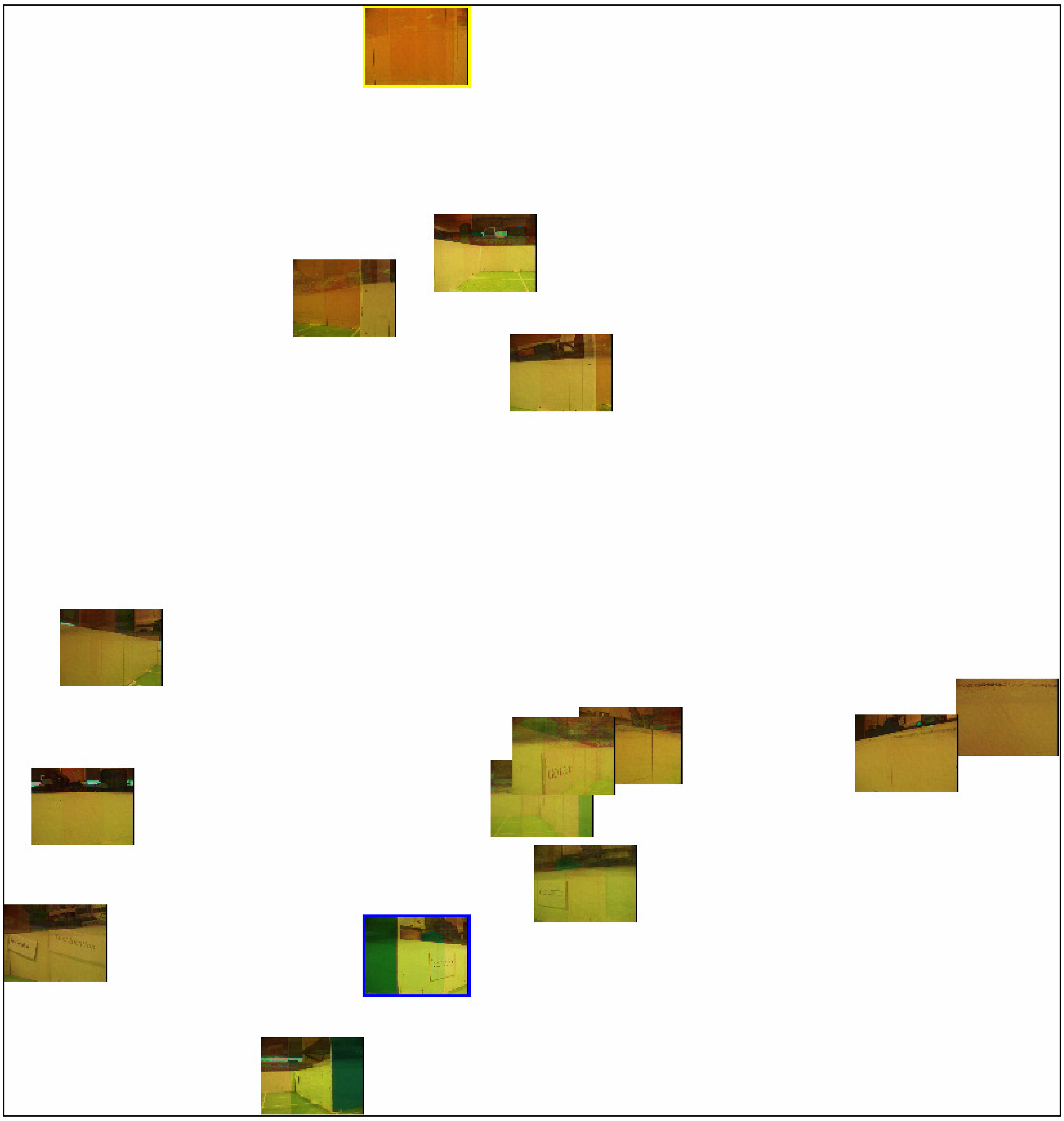


Figure A.1: A map created with parameters from set 6, except for λ which was set between 200 and 250. The map contains more nodes because of the lower λ (see the step 8 of the GNG algorithm).

Figure A.2 shows a map which was learned with a lower ε_b, this parameter is used in step 4 of the GNG algorithm and represents the learning speed. As can be seen from figure A.2, the images are less sharp, because they are a combination of more input images, due to the low learning speed.

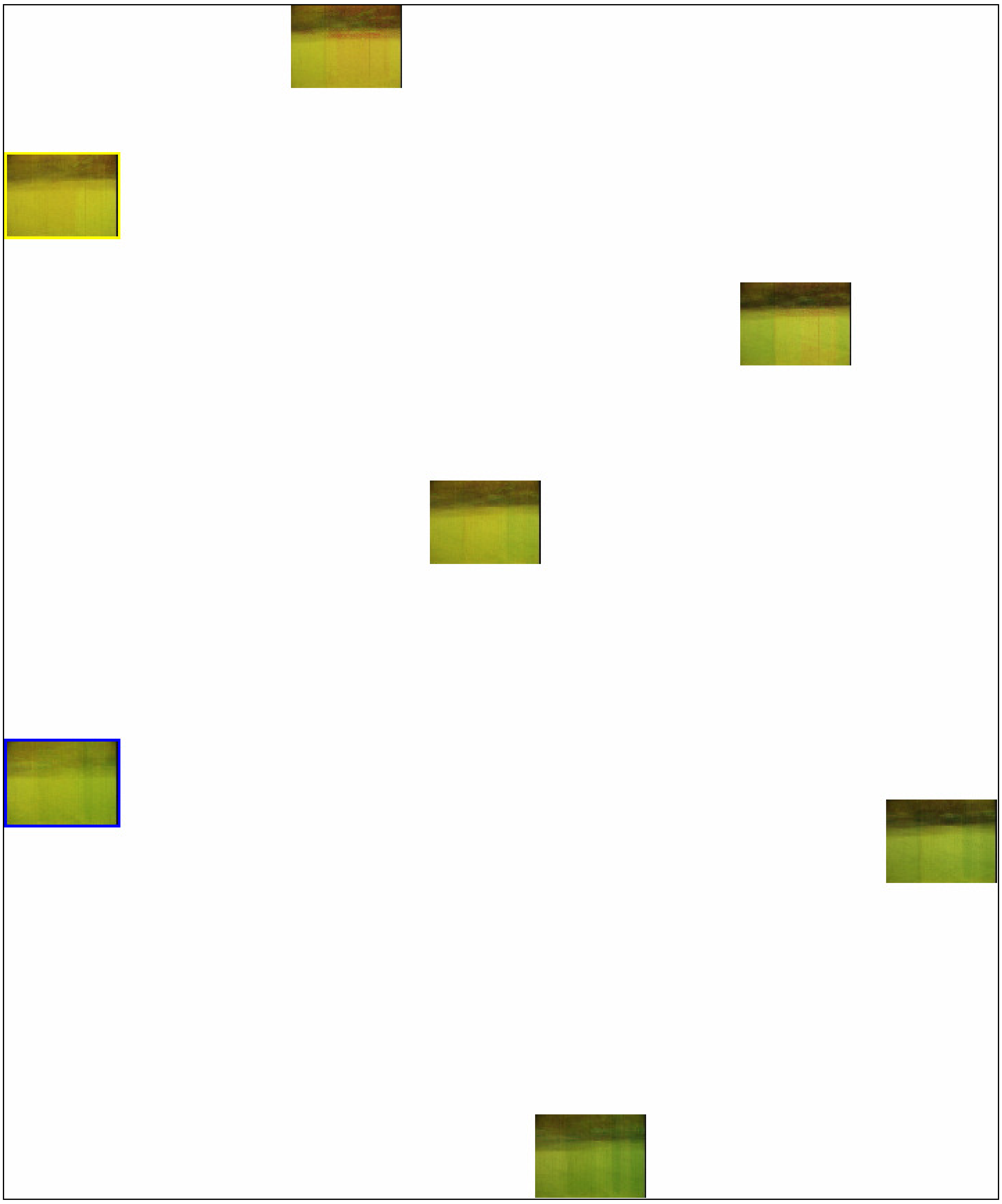


Figure A.2: A map created with the parameters from set 6, except for ε_b which was set to 0.052, this makes the learning slower and forgetting slower too.

References

Fehrmann, R.S.N. (2002) *Cognitive navigation modeling in robots*, graduation paper for Artificial Intelligence dept. of Rijks*universiteit* Groningen.

Fritzke, B. (1995) A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems*, **7**, Cambridge MA: MIT Press, 625-632.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) Optimization by Simulated Annealing, *Science*, **220**(4598), 671-680.

Kohonen, T. (1982) Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, **43,** 59-69.

MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations. In LeCam, L. and Neyman, J., editors, *Proceedings of the Fifth Berkeley symposium on Mathematical statistics and probability*, **1**, Berkeley: University of California Press, 281-297.

Martinetz, T. M. and Schulten, K. J. (1991) A `neural gas' network learns topologies. In *Proceedings of the International Conference on Artificial Neural Networks, Helsinki*, Amsterdam: Elsevier, 397-402.

Tard´os, J.D., Neira, J., Newman, P.P. and Leonard, J.J. (2002) Robust Mapping and Localization in Indoor Environments Using Sonar Data, *The International Journal of Robotics Research*, **21**(4), 311-330.

Thrun, S. (2002) Robotic mapping: A servey. In *Exploring Artificial Intelligence in the New Millenium, Morgan Kauffman*. On-line paper: citeseer.ist.psu.edu/thrun02robotic.html.

Van der Ploeg, H. (2005) *Behavior-based Perception for Autonomous Robot Navigation*, paper for Artificial Intelligence dept. of Rijks*universiteit* Groningen.

Wolf, D.F. and Sukhatme, G.S. (2004) Online Simultaneous Localization and Mapping in Dynamic Environments, In *IEEE International Conference on Robotics and Automation*, 1301-1306.