

Performance Characterisation of a Feature-Based Gaussian Pose Tracker for Mobile Robots in Indoor Environments

David Austin

Robotic Systems Laboratory,
Australian National University
Australia.

d.austin@computer.org had@roboken.esys.tsukuba.ac.jp
robot.anu.edu.au

Yasushi Hada

Intelligent Robot Laboratory
University of Tsukuba
Japan.

Abstract— This paper introduces a redundant localisation system, combining a low-cost Gaussian pose tracker and a particle filter localiser which is more CPU intensive. Ideally, the Gaussian pose tracker can be used for most of the time and the particle filter localiser switched on only when the Gaussian pose tracker fails. The problem studied here is to determine when the Gaussian pose tracker has failed. A number of different measures are proposed which indicate the state of the pose tracker. Experimental results are presented which illustrate the relative performance of the proposed measures. The results demonstrate that a measure derived from the size of the covariance of the Gaussian pose tracker gives a good prediction of when the pose tracker has failed.

I. INTRODUCTION

For some time, both the Australian National University and the University of Tsukuba have been conducting long term robotics experiments[1], [2], [3]. One of the basic requirements for long-term operation of a mobile robot is that it can move from place to place to fulfil its tasks. Usually, motion is accomplished by splitting the problem into two parts. Firstly, the robot position is determined (localisation) and then a navigation module uses this information and free-space data from sensors to control the motion of the robot. Thus, localisation is a key technology in most mobile robot systems (e.g. [4], [5], [6]). For long-term, reliable operations it is therefore important to detect and recover from localisation failures.

The ANU mobile robot currently has two localisation methods, a Gaussian pose tracker which uses a line or wall map[7], [8] and a particle filter localiser which uses line and door features[9], [10]. Both detectors use data from a SICK laser scanner, though the particle filter has previously been used with sonar data as well[9]. The Gaussian pose tracker is inexpensive to run, consuming less than 5% of an 750MHz Pentium III CPU. On the other hand, the particle filter localiser is a heavy computational burden and consumes 80-90% of the same CPU. Therefore, it is desirable to use the Gaussian pose tracker as the primary localiser and switch to the particle filter localiser whenever the pose tracker fails.

This paper considers the problem of determining when

the Gaussian pose tracker has failed. It is desirable to have a simple measure for failure detection which relies on internal information. Clearly, this is a hard problem and no perfect solution can be found. This paper studies this problem as follows. Section II gives an overview of the system and the long term operation project at the ANU. Next, Section III presents an overview of the Gaussian pose tracker that we use, with details relevant to this study. Some measures for failure detection and proposed in Section IV and experimental results demonstrating the performance of each of the measures are given in Section V.

II. SYSTEM OVERVIEW

The system developed for our mobile robot is shown in Figure 1. The robot contains two on-board computers, both 750MHz Pentium CPUs. The lower CPU runs the navigation software, consisting of localisation, path planning and obstacle avoidance modules. In addition, the mapping process runs on the lower computer. The upper CPU has video capture cards and is used to process the incoming video data. At present, the vision information is used only for ongoing research in vision-based SLAM and visual-servoing for recharger docking. The robot operates on one floor of our building, mainly in the corridors. A map is shown in Figure 2.

The hardware and software systems required to build and run a mobile robot are extremely varied and complex. Clearly, we, the robotics community, must develop methods to manage this complexity or we will be unable to create reliable and scalable robot systems. Experimentation on real-world robot systems is required to gain the experience and insight to address the issue of complexity. Our robot runs DROS[11], an open source mobile robot “operating system” that we have developed. DROS consists of over 100,000 lines of code, with more in development. An explicit goal for DROS is to simplify the tasks of modular programming. Modular programming is typical in research environments, where students work on separate aspects of the system. In addition, the hope is that multiple modules can be developed for each im-

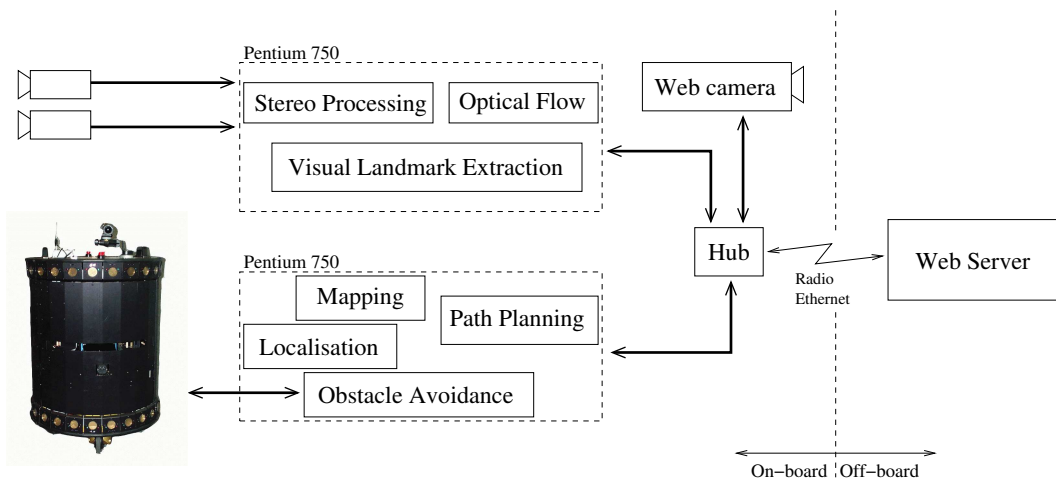


Fig. 1. Overview of the hardware and software components that are used for the mobile robot

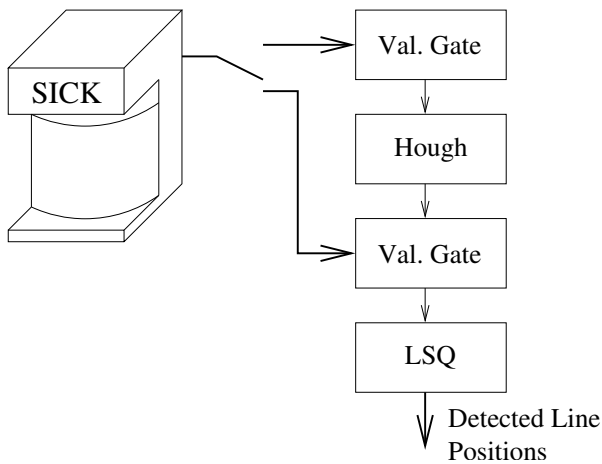


Fig. 3. The two-step validation of the laser pose tracker. When the covariance P is large, both validation gates are used to try to reject noisy data.

portant robotic task to promote error recovery. One way to increase the reliability of a complex system is to use redundant modules to implement critical functions.

The part of the system under study in this paper is the localisation system. Redundant localisation is used with two very different localisation modules currently available. To save CPU cycles, the simple Gaussian pose tracker is used most of the time and the heavy duty particle filter is used only when the robot becomes lost.

III. GAUSSIAN POSE TRACKER

For the redundant localisation scheme, it is highly desirable to use the fast Gaussian pose tracker as the primary localisation method in order to save CPU time. The Gaussian pose tracker was developed by Dr. Patric Jensfelt at the Royal Institute of Technology[7], [8]. The

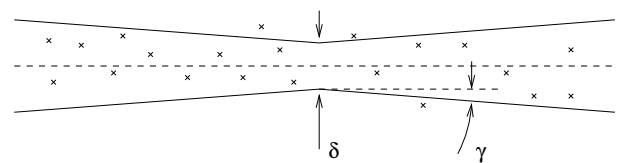


Fig. 4. Validation gate used by the laser pose tracker to search for lines in the laser scan data. The size of the gate, δ and γ , is governed by the covariance of the extended Kalman filter.

pose tracker uses an extended Kalman filter to maintain an estimate of the robot pose (state)

$$\mathbf{x} = (x^W, y^W, \theta) \quad (1)$$

with a 3×3 covariance matrix P .

The pose tracker uses its current pose estimate to filter the current laser scan data. The robot uses its current pose estimate to predict where walls should lie, using the line map (see Figure 2). Using the estimated pose to search for data matching walls dramatically simplifies the problem and leads to the speed and simplicity of the pose tracker.

Figure 3 shows the steps used by the laser pose tracker to detect lines from the laser scan. Depending on the uncertainty about the robot position, a one or two step process is used. For uncertainties less than $0.05m$ in range (towards the line) and 0.5° in angle, only narrow validation gate is used, followed by least squares fitting of the line. For higher uncertainties, a wider initial validation gate is used, followed by a Hough transform to detect the strongest line candidate. The strongest line is then extracted using the narrow validation gate and least squares, as before. Figure 4 shows the parameters of the validation gates that are used. The width of the gate (the range uncertainty) is described by δ and the orientation uncertainty by γ . The key observation is that as

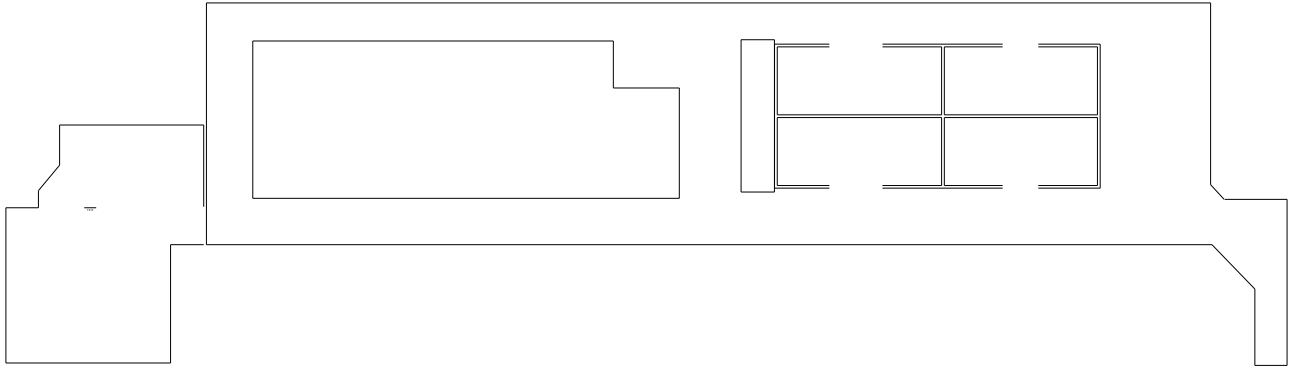


Fig. 2. Line map of the ANU building, as used by the Gaussian pose tracker. The mobile robotics lab is on the left, with the recharging station shown in the middle of the room.

the uncertainty grows, the validation gates become wider, increasing the risk that erroneous data will be used.

A. Practical Details

To avoid over-confidence in the covariance estimate, the diagonal elements of the covariance matrix are prevented from becoming too small. The limits used are[8]

$$P_{xx} \geq (30mm)^2 \quad (2)$$

$$P_{yy} \geq (30mm)^2 \quad (3)$$

$$P_{\theta\theta} \geq (0.065^\circ)^2 \quad (4)$$

The performance of any pose tracker depends strongly on the quality of the odometry of the robot. For the work presented in [7], [8], a Nomadic Technologies N200 robot was used, which has a three-wheel synchro-drive arrangement which delivered excellent odometric information, especially on the lino floors at the Royal Institute of Technology. At the Australian National University, we are using a Nomadic Technologies XR4000 robot, with 4 wheel independent steering. This robot has poor odometric performance, particularly on carpet, as at the ANU. As a result, the laser pose tracker gives worse performance on the ANU robot than for the long-term experiments reported in [7], [8], which provides additional motivation for this study.

IV. POSE TRACKER FAILURE DETECTION

Detection of failure of the pose tracker is difficult to achieve without using external information. In particular, in one failure mode the pose tracker makes a data association error and tracks a parallel structure instead of the wall. Structures parallel to the wall are quite common in office environments. Clearly it is very difficult to deal with all cases of data association errors because the Kalman filter will appear to be tracking properly. Furthermore, the amount of (translational) uncertainty allowed before data association errors become likely will depend heavily on the environment. For example, our mobile robotics lab has

a number of desks facing around the edges of the room. The modesty panels underneath the desks are therefore parallel to the walls at a distance of about 300mm. Given that there are a number of these desks and that the modesty panels prevent the “true” wall being seen, the risk of a data association error is quite significant.

The goal here is to develop a low cost measure which predicts failure. It is not required that the measure is 100% accurate. We would prefer that the measure is a bit too sensitive and predicts all failures with a number of false positives. In other words, it is permissible for the measure to predict that the Gaussian pose tracker has failed when it is still functioning correctly, because then we simply start the particle filter to verify this and waste some CPU cycles. On the other hand, it is much worse if the measure does not detect a significant number of failures.

There are a number of internal variables of the Gaussian pose tracker which should be considered for the failure prediction. Each of these is now discussed.

A. Length of Major Axis

The obvious measure of performance for a Gaussian pose tracker is the covariance of the Gaussian. The primary measure that we use here is the length of the major and minor axes of the covariance ellipse or the maximum and minimum standard deviation values, respectively. Only the translational degrees of freedom are considered. We denote these measures as Q_{major} and Q_{minor} .

We use these two primary measures and also form composite measures by combining them. Firstly, we can estimate the area of the Gaussian (or a proportion thereof) by computing the product of the lengths, denoted $Q_{major \times minor}$. Secondly, we propose simpler measures consisting of the sum and difference of the major and minor lengths, denoted $Q_{major+minor}$ and $Q_{major-minor}$. For these measures, we do not include the orientation degree of freedom for two reasons. First, it is difficult to appropriately scale the orientation with respect to the

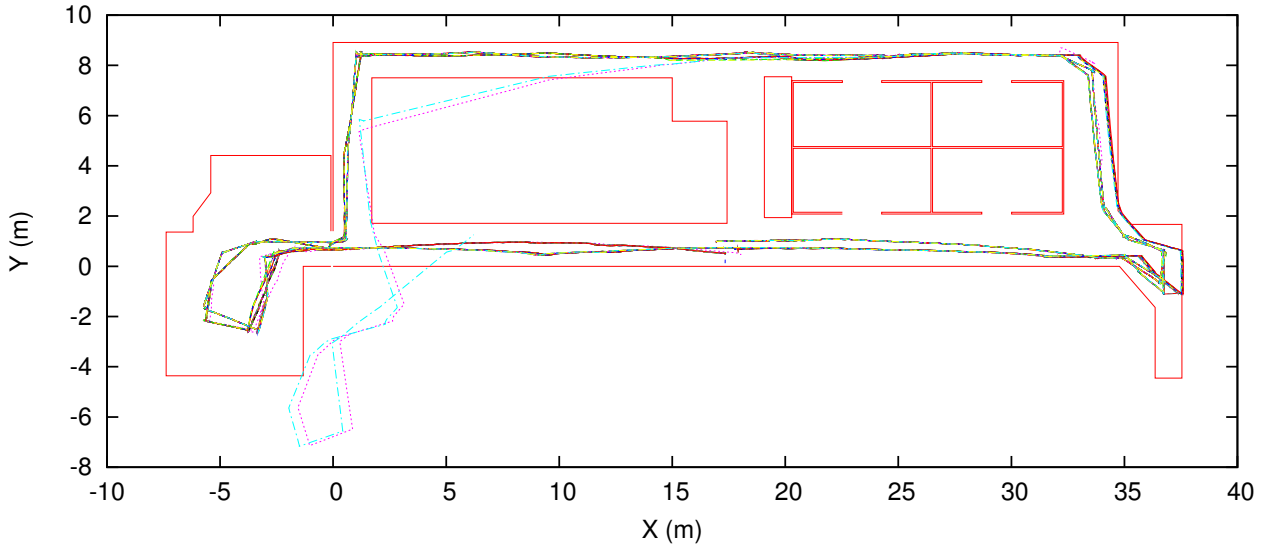


Fig. 5. Poses tracked by 50 different Gaussian pose trackers for a 212m path of two loops around the building.

translational dimensions. Secondly, orientation tracking is much less likely to fail in typical office environments. Generally, the robot can see enough of at least one wall to provide good orientation information.

It is expected that most of these measures will be good predictors of failure (with the exception of the minor axis length) because the probability of false matching (false data through the validation gates) rises as the uncertainty and, hence, the gate size, rises.

B. Number of Lines Tracked

Another measure of performance of the Gaussian pose tracker is the number of features that it is tracking for the current estimate. Since we use a line based pose tracker, this is the number of lines (or walls) that the robot is tracking and is called $Q_{\#lines}$.

A refinement of this measure is to realise that many of the tracked walls are parallel and therefore are not contributing strongly to the quality of the tracking. Therefore, we measure the number of different (all non-parallel) lines that are being tracked, denoted $Q_{\#\Delta lines}$.

C. Other Measures

Another possible measure of failure is when the pose tracker is not tracking a feature that it should be able to track, given its current pose estimate and the map. This requires computation of “visibility” of each of the features and so is a little more CPU intensive than the above measures. However, the biggest reason for not using this measure is that it assumes that the map completely describes the environment. In real-world situations, there are a number of obstacles not described in the map (e.g. people). If the robot is surrounded by people, then it will

have difficulty tracking the walls, even though it may be perfectly localised. Also, the simple line map does not describe a large number of stationary objects in our building which will prevent the robot from tracking walls in many situations. For these reasons, we do not consider this measure here.

Another possible indicator for failure would be the innovation matrix of the Kalman filter. Clearly, the innovation can detect gross failures (e.g. when no measurements fall within the validation gates). However, in the event of a data association error, the Kalman filter will appear to be tracking perfectly and, hence, there can be no information available from the innovation to help detect this failure mode. For this reason, we do not consider innovation-based measures.

V. EXPERIMENTAL RESULTS

The robot was driven a distance of 212m twice around our building, as shown in Figure 5. While the robot was driven, 50 laser pose trackers were run simultaneously. Each pose tracker was initialised with the true initial pose of the robot. To differentiate the pose trackers and so ensure that their performance is not identical, a small random perturbation is added to the initial pose estimate. The perturbation is uniform in the range $[-2.5cm, 2.5cm]$ added to the x and y estimate and $[-1^\circ, 1^\circ]$ added to the orientation. This perturbation is too small to affect the pose tracker performance (initially) but ensures that there will remain small differences between the pose trackers. Essentially, this just adds noise to ensure that the pose trackers do not produce identical results.

After the experiment, the logged data was processed to extract the mode (assumed to be the true pose of the robot)

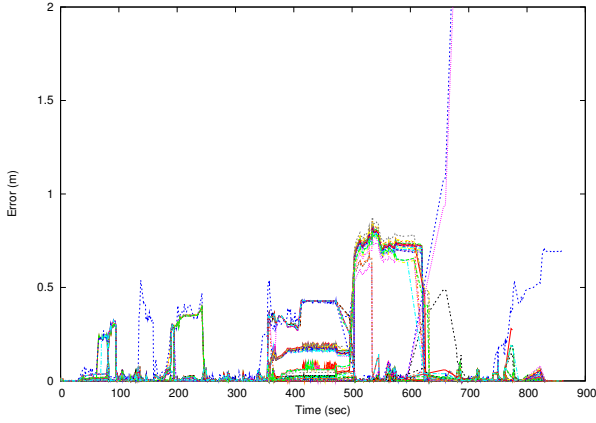


Fig. 6. Pose errors of 50 different Gaussian pose trackers for path of Figure 5.

from the data. Figure 6 shows the errors between the pose trackers and the true pose. It can be seen that many of the pose trackers failed. Two failed completely and begin to diverge after about 600s. The rest of the failures are temporary in nature and result from mis-localisation. In these cases, the pose tracker is correct in the y direction but wrong in the x direction (along the long axis of the building). These failures typically occur in the two long corridors between $x = 0m$ and $x = 15m$ where there are no features to permit determination of the x position.

When processing the data, a pose tracker was said to have failed if its estimate diverged from the true pose by more than $0.2m$. From the experiment, there are 70 cases where one of the pose trackers fails. The data from the true pose was compared with the data from each performance measure. Table I presents the results for each of the performance measures, with a range of relevant thresholds for each. The true positives column list the number of times that each measure correctly predicts failure. The false positives column lists the number of false alarms (which we are willing to accept for this application) and the false negatives column shows the number of times each measure fails to predict that the pose tracker has failed. As stated earlier, the goal is to detect as many failures as possible, without generating too many false alarms (false positives). As expected, the length of the major axis of the uncertainty ellipse is a strong indicator of failure (see Table I). It seems that a failure detector could be built from the major axis data, with careful choice of the threshold to balance false negatives against false positives.

The surprising result from this data is the quality of the $Q_{major+minor}$ measure. Here it correctly predicts 95% of failures. This is a very good result for failure prediction without using external information. Also, as shown in Table I, the performance of the $Q_{major+minor}$ measure is not strongly dependent on the threshold value used. For a

Measure	Number of true positives	Number of false positives	Number of false negatives	Av. distance to detect (m)
$Q_{major>0.2m}$	27	66	43	0.6
$Q_{major>0.15m}$	37	122	33	2.9
$Q_{major>0.1m}$	44	124	26	1.9
$Q_{major>0.08m}$	66	246	4	1.0
$Q_{major>0.05m}$	66	423	4	0.2
$Q_{minor>0.07m}$	26	26	44	2.3
$Q_{minor>0.05m}$	26	97	44	2.0
$Q_{minor>0.04m}$	57	225	13	4.7
$Q_{minor>0.03m}$	70	2458	0	4.2
$Q_{\#lines<1}$	48	8	22	1.5
$Q_{\#lines<2}$	66	1050	4	3.6
$Q_{\#\Delta lines<1}$	48	8	22	1.5
$Q_{\#\Delta lines<2}$	70	1127	0	0.0
$Q_{major+minor>0.15m}$	40	120	30	3.8
$Q_{major+minor>0.12m}$	45	120	25	2.9
$Q_{major+minor>0.1m}$	67	121	3	1.5
$Q_{major+minor>0.08m}$	67	297	3	0.8
$Q_{major+minor>0.05m}$	67	650	3	0.1
$Q_{major-minor>0.15m}$	31	84	39	1.7
$Q_{major-minor>0.1m}$	44	123	26	4.1
$Q_{major-minor>0.05m}$	66	246	4	1.1
$Q_{major\times minor>0.01m^2}$	25	1	45	1.8
$Q_{major\times minor>0.005m^2}$	36	120	34	2.7
$Q_{major\times minor>0.002m^2}$	67	373	3	0.3

TABLE I
EXPERIMENTAL PERFORMANCE OF FAILURE DETECTORS.

small range, the only variation is that the number of false alarms increases as the detector is made more sensitive.

Another interesting characteristic for the failure detector is how long it takes to detect failure. Here we characterise the delay in terms of distance that the robot moves after it becomes lost (i.e. the pose error is more than $0.2m$). The last column of Table I gives the average detection distance for each of the measures. In addition, Figure 7 shows the histogram of distances between pose tracker failure and prediction of failure by the $Q_{major+minor>0.1m}$ measure. The vast majority of failures are detected within $2m$ of travel. The failures detected after $9m$ of travel are interesting and will be studied further. We suspect that the pose tracker is tracking incorrect features for some time.

There are three types of failure for the pose tracker:

- it can become completely lost and diverges from the true pose quite rapidly
- it can become mis-localised in one direction (usually the x direction) but keep tracking the other direction, or
- it can become confused by structures in the environment parallel to the walls that it is trying to track (a data association error).

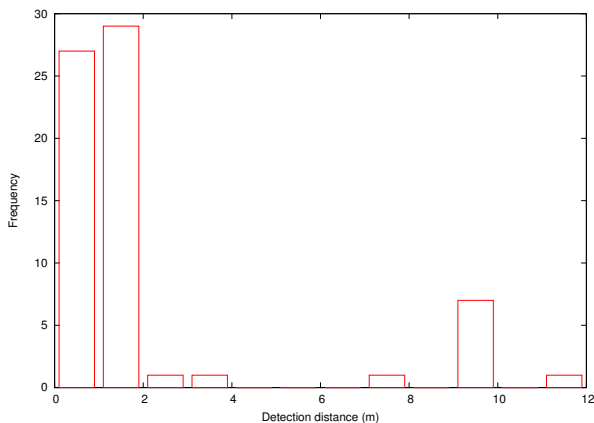


Fig. 7. Detection distance histogram for $Q_{major+minor} > 0.1m$.

Clearly, detection of the first case, “completely lost”, is quite simple as the number of features tracked is low and the uncertainty of the pose tracker grows rapidly. The second case, “mis-localised”, is harder to detect as the robot is behaving in a very similar fashion to areas where there are no features in one direction. For example, in our building, there are two long corridors where no features are found to help localisation along the corridor. Mis-localisation failures give exactly the same appearance as passing through these corridors. Therefore, the measure developed here will tend to predict failure in these regions, even if the pose tracker is functioning correctly. The third case, “data association error”, is very difficult to detect if it occurs because the Kalman filter appears to be performing perfectly (unless we use external information to detect this failure). Therefore, the best measure here predicts failure prior to the point where there is a significant risk of data association error.

As discussed above and in Section IV, the prediction of failures, particularly, data association errors, depends upon the environment. For data association errors, the key factors are the number of parallel structures and their distances from the walls in the map. In our environment, there are significant numbers of structures parallel to the walls with offsets of about $300mm$. We believe that this is related to the optimal choice of $0.1m$ found in Table I.

VI. CONCLUSION

The redundant localisation system used for the mobile robot at the ANU needs a simple way to determine when to switch on the CPU intensive particle filter (i.e. a need to detect when the lightweight Gaussian pose tracker has failed). This paper has studied this problem and proposed a number of measures for failure detection for the pose tracker. A simple threshold derived from the size of the pose tracker uncertainty delivers 95% detection of failure in 70 different test cases. The only shortcoming of this

measure is that it sometimes predicts failure when the pose tracker is working properly. However, false prediction of failure simply results in running the particle filter unnecessarily, which is not a significant problem. Therefore, the failure detector developed is a valuable addition to the redundant localisation scheme for the ANU robot.

VII. REFERENCES

- [1] S. Yuta and Y. Hada, “Long term activity of the autonomous robot - proposal of a bench-mark problem for the autonomy,” in *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’98)*, pp. 1871–1878, Oct. 1998.
- [2] Y. Hada and S. Yuta, “Robust navigation and battery re-charging system for long term activity of autonomous mobile robot,” in *The 9th International Conference on Advanced Robotics(’99 ICAR)*, pp. 297–302, Oct. 1999.
- [3] D. J. Austin, L. Fletcher, and A. Zelinsky, “Mobile robotics in the long term,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2001.
- [4] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, “Minerva: A second generation mobile tour-guide robot,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.
- [5] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, J. O’Sullivan, and M. M. Veloso, “Xavier: Experience with a layered robot architecture,” in *Agents ’97*, 1997.
- [6] J. Borenstein, H. R. Everett, and L. Feng, *Navigating Mobile Robots*. A K Peters, 1996.
- [7] P. Jensfelt and H. I. Christensen, “Pose tracking using laser scanning and minimalistic environmental models,” *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 138–147, Apr. 2001.
- [8] P. Jensfelt, *Approaches to Mobile Robot Localization in Indoor Environments*. PhD thesis, Royal Institute of Technology, Stockholm, 2001.
- [9] P. Jensfelt, D. J. Austin, O. Wijk, and M. Andersson, “Feature based condensation for mobile robot localization,” in *IEEE Intl. Conf. on Robotics and Automation*, 2000.
- [10] P. Jensfelt, O. Wijk, D. J. Austin, and M. Andersson, “Experiments on augmenting condensation for mobile robot localization,” in *IEEE Intl. Conf. on Robotics and Automation*, 2000.
- [11] D. Austin, “Dave’s robotic operating system,” 2002. <http://www.dros.com/>.