Markov techniques for object localization with force-controlled robots

Klaas Gadeyne, Herman Bruyninckx*
Dept. of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium
Klaas.Gadeyne@mech.kuleuven.ac.be

Abstract

This paper deals with object localization with force-controlled robots in the Bayesian framework [1]. It describes a method based on Markov Localization techniques with a Monte Carlo implementation applied for solving $3D$ (6 degrees of freedom) global localization problems with force-controlled robots. The approach was successfully applied to problems such as the recursive localization of a box by a robot manipulator.

1 Introduction

In order to be able to automate force-controlled production processes without the need for precise positioning of the workpieces, fast and accurate localization methods are required (Figure 1). This paper investigates what concepts can be reused from successful mobile robotics research (where Markov Localization methods have been proven to work, e.g., [2]), and what are the fundamental differences between mobile robotics and force/tactile controlled robots.

A broad range of techniques are used for pattern recognition and object localization (neural networks [3], fuzzy logic [4], possibility theory [5], Bayesian inference [6, 7], ...). Bayesian localization of 3D objects performs inference in the form of posterior probability distributions that incorporate information about i) the prior knowledge ("belief") about the problem, ii) sensor characteristics, and iii) the shape of the object to be recognized or localized:

$$P(\mathrm{l} \mid \mathrm{s}, H_0) \sim P(\mathrm{s} \mid \mathrm{l}, H_0) P(\mathrm{l} \mid H_0) \qquad (1)$$

where $P(\mathrm{l} \mid \mathrm{s}, H_0)$ denotes the conditional probability of a certain position/orientation l of the object, given a measurement vector s and background knowledge about the problem to be solved H_0 (e.g., the shape of the object). Information of both the sensor characteristics and the model of the object is represented by the *likelihood function* (a Probability Density Function (PDF)) $P(\mathrm{s} \mid \mathrm{l}, H_0)$. The likelihood corresponds to a *prediction*: given the position and the shape of the object and taking into account the sensor characteristics, $P(\mathrm{s} \mid \mathrm{l}, H_0)$ indicates how likely the sensor measurements are. Bayes' rule combines the *a priori* PDF $P(\mathrm{l} \mid H_0)$ about the position of the object, with the likelihood PDF $P(\mathrm{s} \mid \mathrm{l}, H_0)$ to obtain the a posterior PDF $P(\mathrm{l} \mid \mathrm{s}, H_0)$, that represents the information on the problem *after* having seen the measurements.

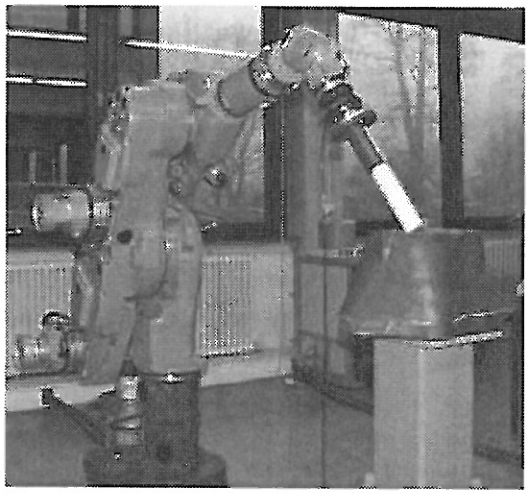

Figure 1: Localization of a casting piece by a robot manipulator equipped with a force-controlled "tactile" sensor.

Bayesian techniques are appropriate to incorporate *model-based* information, [8, 9], although it might be difficult to turn available models into an appropriate a priori PDF. *Computational issues* such as huge memory requirements and computation time often show up in practical cases; and therefore, most Bayesian models make simplifying assumptions about the probability functions (e.g.,

*Postdoctoral Fellow of the F.W.O–Vlaanderen. The financial support by the Belgian State—Prime Minister's Office—Science Policy Programme (IUAP), and by K.U.Leuven's Concerted Research Action GOA/99/04 are gratefully acknowledged.

Kalman Filters [10], Hidden Markov Models [11]), or rely on sampling (e.g., Monte Carlo Markov Localization [2]).

The following Section describes a Markov approach for object localization. Markov Localization (ML) is an iterative way of using Bayes' rule to update the available information each time new sensor data comes in. Then, Section 3 reviews a case study where a Monte Carlo implementation of the ML algorithm is successfully applied. Finally, some considerations about efficiency improvements are made.

2 Markov Techniques for object localization

Markov Localization [2] is often used in the mobile robotics world. The approach is capable of solving a *global* localization problem, i.e., given a map of its environment, a robot should be able to localise itself without knowing its initial position.

2.1 Theoretical derivation of ML for object localization

Let $Bel(\mathbf{L_k} = \mathrm{l}) = P(\mathbf{L_k} = \mathrm{l} \mid \mathrm{s})$ denote the robot's belief of the environment object's position at timestep k and $\mathrm{s_k}$ the sensor measurement vector at that moment (note that, in the general case $\mathbf{L_k}$ and $\mathrm{s_k}$ are not of the same dimension!). $\mathrm{s} = \mathrm{s_0}, \mathrm{s_1}, \ldots, \mathrm{s_k}$ represents all sensor measurements until step k. The algorithm finds an a posteriori PDF that reflects the robot's belief of the object's position, given all available sensor data:

$$P(\mathbf{L_k} = \mathrm{l} \mid \mathrm{s}) = P(\mathbf{L_k} = \mathrm{l} \mid \mathrm{s_0}, \mathrm{s_1}, \ldots, \mathrm{s_k}).$$

The robot's belief is recursively updated with each new sensor measurement according to Bayes' rule:

$$P(\mathbf{L_k} = \mathrm{l} \mid \mathrm{s_0}, \ldots, \mathrm{s_k}) = \frac{P(\mathrm{s_k} \mid \mathbf{L_k} = l, \mathrm{s_0}, \ldots, \mathrm{s_{k-1}}) P(\mathbf{L_k} = \mathrm{l} \mid \mathrm{s_0}, \ldots, \mathrm{s_{k-1}})}{P(\mathrm{s_k} \mid \mathrm{s_0}, \ldots, \mathrm{s_{k-1}})}. \quad (2)$$

A crucial factor for the efficiency of ML is the *Markov assumption*:

$$P(\mathrm{s_{k+1}}, \mathrm{s_{k+2}}, \ldots \mid \mathbf{L_k} = l, \mathrm{s_0}, \mathrm{s_1}, \ldots, \mathrm{s_k}) = P(\mathrm{s_{k+1}}, \mathrm{s_{k+2}}, \ldots \mid \mathbf{L_k} = l). \quad (3)$$

The Markov assumption says that old measurements $\mathrm{s_k}$ can be forgotten, as soon as they have been used to update $P(\mathbf{L_k} = \mathrm{l} \mid \mathrm{s_0}, \ldots, \mathrm{s_k})$. The Markov assumption reduces Eq. (2) to

$$Bel(\mathbf{L_k} = \mathrm{l}) = \alpha P(\mathrm{s_k} \mid \mathrm{l}) Bel(\mathbf{L_{k-1}} = \mathrm{l}), \quad (4)$$

where α is a normalization factor. $P(\mathrm{s_k} \mid \mathrm{l})$ is the likelihood function that predicts the measurements for each location l. Each new measurement allows the robot to improve its estimate based on a model of its sensors and the object.

The ML algorithm doesn't specify anything about *how* the robot's belief is represented; for localization problems, a Monte Carlo approach is an efficient implementation.

2.2 A Monte Carlo approach for Markov Localization

An often used implementation technique of the ML-algorithm is the Kalman-filter (KF). It minimizes the mean squared error estimate (MMSE), which corresponds to finding the mean value (first order moment) of $Bel(L_k) = \mathrm{l}$ *if* the system and measurement equations are linear. Major problems with KFs are that (i) first-order and second-order moments (i.e., the PDF is approximated by a Gaussian PDF) are not enough to describe multi-modal distributions, and (ii) the (Extendend) KF has problems dealing with non-linear system and measurement equations. But the KF remains the fastest means of localizing objects, provided that i) $Bel(\mathbf{L_k} = \mathrm{l})$ and the sensor characteristics can be modeled as a unimodal PDF, ii) both system and measurement equations are linear. However, for complex objects (such as the cast piece in Fig. 1) measurement equations are inherently non-linear. This limits the use of the Kalman filter mainly to *tracking* problems, i.e., the object has been localized already, but its position and orientation change (due to the object's proper motion, or motion of the robot that carries the sensors).

Approximation by a Gaussian might not be a good option; then, sampling the state space l could be an alternative. Such a *Monte Carlo* approach (i.e., any method using a *grid* on l), is a slower but more powerful method of implementing the ML algorithm. For every point of the grid, the robot updates its belief about the object's position at every time step. In this way $Bel(\mathbf{L_k} = \mathrm{l})$ can represent a PDF of any form.

Monte Carlo methods sample the state space in an "intelligent," problem-dependent way: the resolution of the grid of the state space is refined where $Bel(\mathbf{L_k} = \mathrm{l})$ attains sensible values, whereas the grid is not updated in places where the chance of finding the object is very small. Another way of improving performance is to do all prediction step calculations off line. In this way, computing the likelihood function reduces to looking up a value in a lookup table. However, compared to Kalman filtering, the implementation remains slower and more memory consuming.

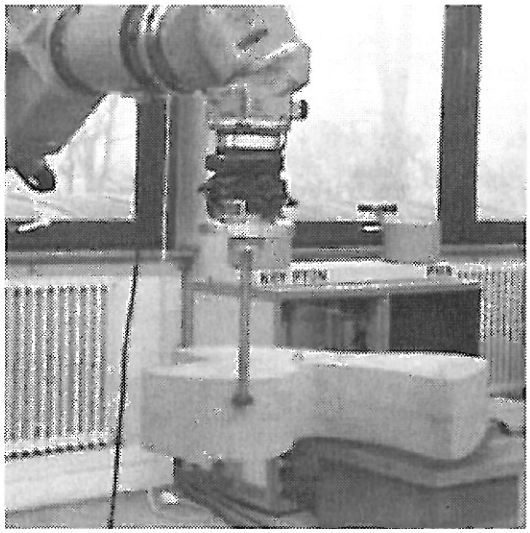

Figure 2: Following a contour with a force-controlled robot.

2.3 Differences between mobile robotics and force-controlled robots

As demonstrated before, due to its construction, the use of the KF is limited to tracking. For the global localization of objects, a Monte Carlo approach will yield better results. In the mobile robotics field, localization is executed in 3 degrees of freedom (DOFs)—2 for position and 1 for orientation—, whereas localization of a casting as in Figure 1 requires search in 6 DOFs (3 for position and another 3 for orientation).

However, following a contour of an object to get information about its shape and position (Figure 2) involves only a 3D space, *if* one knows it is fixed on a horizontal support. Localizing an object in an area of $40 \times 40 \times 40 cm^2$ with a precision of $5mm$ in translation and 2^o in rotation requires a state space grid 3×10^5 times bigger than the one of a mobile robot in a building of $100 \times 100 m^2$ with a grid of $20cm$ in x/y and 10^o in rotation.

Some sensors are *"tactile,"* and thus provide only *local* information; sensor such as laser scanners can provide *global* information about the environment or the position of the object. However, data processing becomes more complex as sensors become more powerful too. When tactile sensors are used, the time between two sensor measurements is rather high and a couple of seconds are available between the measurements for updating the belief. When information about the curvature is gathered by following the contour of an object with a force-controlled sensor as in Figure 2, the time delay between measurements is much higher, because the control problem requires higher rates. In this case, a *preprocessing* of the data (e.g., integrating the curvature along the trajectory) is possible. One can choose whether to use likelihood functions that process *raw* sensor data (point positions, forces or velocities), or *pre-processed* data (e.g., curvature and arc length) derived from the raw data.

A speed gain can also be obtained by an *active sensing* approach [12]. In this way, every sensor measurement gives a large increase in information. For example, when localising a cube, it is not useful to do 20 measurements on one side of the cube. The gathered sensor information should be as *complementary* as possible in order to reduce the second order moment (covariance) of the different peaks in $Bel(\mathbf{L_k} = \mathbf{l})$ as fast as possible.

The following section describes a case study where a Markov/Monte Carlo approach has been successfully applied to an object localization problem. The case study concerns a simple object geometry, and the state space localization problem contains only 3 degrees of freedom (position and orientation in a plane). (The same problem has been solved with a Kalman Filter approach by De Geeter et al., [13].)

3 Case study: Recursive localization of a box by a robot manipulator

A KUKA robot with 6 degrees of freedom equipped with a 6 DOF Schunk force sensor is programmed to move until it touches a side of a cube in its environment (as in Figure 1, with the casting piece replaced), and stops when a threshold of $5N$ is reached. It then sends its position to a program that implements a Markov approach. The force/torque sensor measures forces in 6 dimensions (3 forces, 3 torques). Position errors are a consequence of i) errors on the (6) joint positions of the robot, ii) the proper compliance of the cube and the peg/force sensor (negligible in this case, due to their high stiffness), iii) measurement noise, and iv) discretization "noise" due to sampling. We assume a Gaussian characteristic for the error on the measured position.

Even for this simple object, the combination of the Gaussian sensor characteristics and the model of the box results in a mathematically complex function. For an object whose contour can be described as a curve S (figure 3), the likelihood function with a Gaussian sensor characteristic becomes

$$P(x,y) = \frac{1}{L} \int_{s=0}^{s=l} e^{f(x,y,s)} ds \qquad (5)$$

where L is the arclength of S, s indicates a parameter along the curve S and $f(x,y,s)$ is a function of x, y and s. For

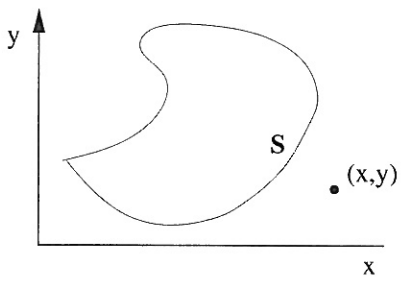

Figure 3: Construction of the likelihood function in 3 DOFs of an object with contour S.

one side of the cube, one obtains an expression of the form

$$P(x,y) = \frac{1}{L}\int_{s=0}^{s=L} e^{as^2+bs+c}ds \qquad (6)$$

where a, b and c depend on x and y, and L is the length of the side of the cube. The total likelihood function is the sum along 4 sides. A numerical integration library allows to construct a "likelihood look-up table"—a sampled version of $P(x,y)$— off-line. The result in the case of the cube where a sensor with Gaussian characteristics has been used is shown in Figure 4. The resolution of the grid determines the possible accuracy of the estimated location of the cube.

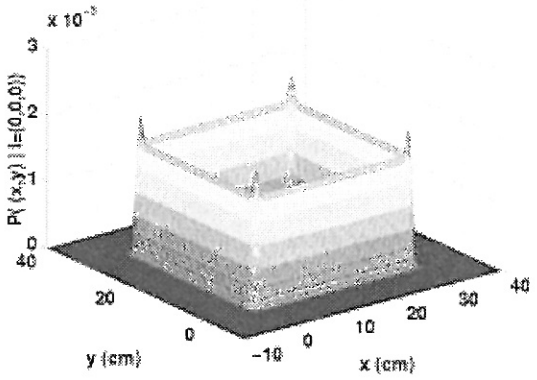

Figure 4: Visualisation of the sampled version of the likelihood function $P(x,y)$ for a cube with its corner located at (0,0). The look-up table has been constructed off-line with the octave numerical integration library.

The result of an approach with the look-up table from Figure 4 and a variable number of samples can be seen in Figure 5. Samples whose probability gets under a certain threshold are destroyed in order to be able to reduce the number of samples (and hence the required memory and the necessary time to update all sample probabilities) as fast as possible. For example, the number of samples in the previous experiment after 1 measurement is reduced with a factor 10! However, to be able to recover from bad measurements, such an approach is dangerous! Notice that

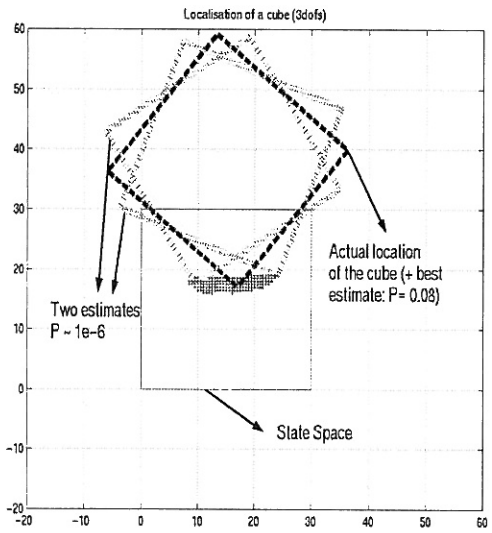

Figure 5: Result of the localisation of a cube in 3 dofs. The considered state space is a "square" with corners (0,0,0°) and (30,30,90°). The grid resolution is 0.2 cm (linear) and 1° (angular) respectively. This means an initial number of 2.000.000 samples. With this number of samples, the localisation can be performed in "real-time" (i.e. less than a second on a AMD-K6 475MHz, 64 Mb RAM, running the Linux OS, for all measurements). The "cloud" represents the remaining samples after 6 sensor measurements. Each sample represents the position of a corner (left under) of the cube (different orientations at a certain position are not visible on the plot!). The "exact" position of the cube is (17,17,50°) and corresponds with the best estimate. All coordinates are expressed in cm.

the (in this case simple) shape of the object has nothing to do with the (online) calculation time. Localising the complex shaped casting piece of Figure 1 requires more off-line calculation time, but once the lookup table has been calculated, the localisation is as fast as that of the cube. The calculation effort is mainly important in the first steps where a lot of samples have to be updated.

In 6 dofs, a stepwise approach where the granularity of the grid is adapted online, will certainly be necessary to be able to update all samples in "real-time". For high

precision applications, one could extend the Monte Carlo approach and really use different grids for different precisions. A first grid could e.g., be used for localization up to a precision of $5cm$, a second one for a precision up to $1cm$ and so on. For this purpose, different sensors (with different resolutions) could be used (or combined). The probability of each sensor measurement can be found in a likelihood function that takes into account the appropriate sensor characteristics.

One can take into account the symmetry of the object to reduce memory/time requirements. However, the time gap of a couple of seconds between the different touches allows to verify that the approach is well suited for multi-modal distributions (generated by the symmetries of the cube): the a posteriori distribution always contains four peaks. Figure 6 represents the remaining samples after taking into account 6 measurements.

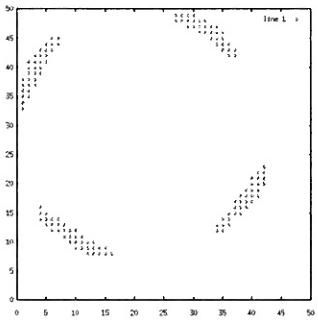

Figure 6: Localising a cube without taking into account the symmetry of the object: The figure shows the remaining samples after 4 measurements. The resulting PDF is clearly multi-modal.

Comparison to (recursive) least-squares. As long as one supposes that the sensor measurements are distributed normally, a recursive weighted least squares approach would yield the same results as the implementation of the Markov localization algorithm. The major difference is that the least squares method is only valid if the sensor characteristics can be written as a Gaussian distribution. When the sensor characteristic is differently distributed, only a Markov approach is possible.
Another difference is that the complexity of the least squares method will increase with the "complexity" of the object, unlike with the Markov approach where the lookup table contains a sampled version of the likelihood.

4 Efficiency improvements

Complexity. Due to the complexity of the likelihood functions and the large number of states, a Monte Carlo sampling approach will always be necessary in practice. Therefore it would be useful to be able to perform the localization algorithm in a lower-dimensional *recognition* space (Figure 7). In this way, it is possible to obtain a likelihood function with fewer degrees of freedom (and thus faster to evaluate).

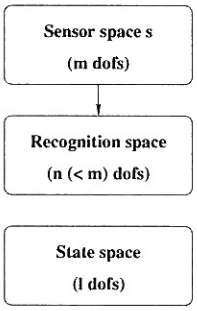

Figure 7: Localization in recognition space.

Another possibility is automating the construction of a *sampled* version of the likelihood function starting from a probability function expressing the sensor characteristics and a CAD-model of the object.

Pattern recognition The robot should be able to make a distinction between different objects to be manipulated/machined. This is a pattern recognition problem and can also be solved by a model-based approach. The continuous state variable l is replaced by a discrete number of classes and thus less complex than the localization problem. A localization problem can be seen as a pattern recognition problem where the number of patterns to be recognized tends towards infinity.

To further reduce the computational effort of the pattern recognition problem with complex objects, the use of local/global *invariants* [14] as used in the object recognition algorithms for images, has to be considered. An invariant is a specific mathematical function that combines measurements in such a way that the result does not change under a specific group of transformations. For example, consider the corner of the cube of section 3. When measuring 2 points on both sides of the cube (in 1 horizontal plane) and calculating the angle between the lines through these points, one will allways obtain 90^o, no matter what the location of the box is. In this way, a complex model of that corner can be replaced by a single value.

Active sensing An active sensing approach based on the entropy of $Bel(\mathbf{L_k} = \mathbf{l})$ allows i) to filter out "bad" measurements, and ii) to take the most appropiate next sensor action. The entropy of the robot's belief is written as

$$E(Bel(\mathbf{L_k})) = \int_{L_k} Bel(\mathbf{L_k} = \mathbf{l}) \log(Bel(\mathbf{L_k} = \mathbf{l}))d\mathbf{L_k}.$$

It is maximal if the robot's belief is a uniform PDF.

When using the entropy of the distribution for filtering out bad measurements, one uses the assumption that a good measurement will improve the robot's position estimate of the object and thus reduce the entropy of $Bel(\mathbf{L_k})$. So

$$E(Bel(\mathbf{L_{k+1}})) \leq E(Bel(\mathbf{L_k})).$$

This means that filtering out bad measurements already requires a certain idea about the location of the object, and will not work well during the first measurements. One can only reject measurements if one has already an idea of what they should be!

The same goes for the choosing the most appropriate next sensor action. That is the measurement that produces maximal information and hence maximally reduces the entropy of the robot's belief. So $\mathbf{s_{k+1}}$ should be chosen in order to minimize $E(Bel(\mathbf{L_{k+1}}))$.

5 Conclusions

Although the Bayesian framework in general, and Markov Localization in particular, are well suited to solve global object localization in 6 degrees of freedom, computational requirements for representing the multi-modal non-parametrical PDFs are high. Therefore, several implementation issues are considered. A combination of a Monte Carlo sampling of the state space l, combined with an off-line evaluation of the prediction step of Bayes' rule and the use of invariants allow the ML to be executed in real-time. Construction of the likelihood function from available CAD-models and solving a pattern recognition problem would allow production processes to be more automated in a flexible way. Active sensing methods can help to eliminate erroneous sensor measurements and speed up the localization process by choosing the most appropriate next sensor measurement.

References

[1] H. Bruyninckx, J. De Schutter, and T. Lefebvre, "Autonomous compliant motion: The Bayesian approach," 2000.

[2] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, 1999.

[3] A. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks, a tutorial," *IEEE computer*, vol. 29, pp. 31–44, maart 1996.

[4] G. J. Klir and T. A. Folger, *Fuzzy sets, uncertainty, and information*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[5] D. Dubois and H. Prade, "A synthetic view of belief revision with uncertain inputs in the framework of possibility theory," *Int. J. of Approximate Reasoning*, vol. 17, no. 2/3, pp. 295–324, 1997.

[6] E. T. Jaynes, "Bayesian methods: general background," in *MaxEnt'84: Maximum Entropy and Bayesian Methods in Geophysical Inverse Problems* (J. H. Justice, ed.), pp. 1–25, Cambridge University Press, 1986.

[7] E. T. Jaynes, "Highly informative priors," in *Bayesian statistics* (J. M. B. et al., ed.), pp. 329–360, Elsevier Science Publishers, 1985.

[8] D. J. C. MacKay, "Probable networks and plausible predictions—A review of practical Bayesian methods for supervised neural networks," in *Network*, 1995.

[9] D. J. C. MacKay, "Bayesian methods for supervised neural networks," 1995.

[10] R. E. Kalman, "A new approach to linear filtering and prediction problems," vol. 82, pp. 34–45, 1960.

[11] L. R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[12] S. Abrams, P. K. Allen, and K. A. Tarabanis, "Dynamic sensor planning," pp. 605–610.

[13] J. De Geeter, J. De Schutter, H. Bruyninckx, H. Van Brussel, and M. Decréton, "Task-directed sensing with robust sensors for the local geometric modelling of a nuclear installation," in *Proc. BNES Conference on Remote Techniques for Hazardous Environments*, (London, England), 1999.

[14] T. Tuytelaars, *Local, invariant features for registration and recognition*. PhD thesis, K.U. Leuven, Department of electrical engineering, div. PSI, Kard. Mercierlaan 94, B-3001 Heverlee, Belgium, december 2000.