Service Oriented Soft Real-time implementation of SLAM capability for mobile robots

Clovis Peruchi Scotti * **Andrea Cesetti** ** **Gianluca di Buo** *** **Sauro Longhi** **

* *Santa Catarina Federal University (UFSC), Florianopolis, Brazil (e-mail: scotti@ieee.org).*
** *Universita Politecnica delle Marche, Ancona, Italy (e-mail: cesetti@diiga.univpm.it, sauro.longhi@univpm.it)*
*** *IDEA Scarl, Ancona, Italy (e-mail: gianluca@idea-on-line.it)*

Abstract: This paper focuses on solving practical challenges inherent from the use of state-of-art mobile robotics techniques in a resource hungry embedded mobile unit without inherent support for hard real-time operation. Such problems include real-time constraints, sensor acquisition independence from robot movement, multi-rate parallel data acquisition and memory limitations.

Keywords: soft real-time, SLAM, Microsoft Robotics Studio, Service-Oriented Architecture

1. INTRODUCTION

Although theoretical knowledge on the areas of mobile robots localization, map building and pose control is very well established (See, e.g. Thrun et al. (2006), Siegwart and Nourbakhsh (2004)), use of such knowledge in ready-to-market products is scarce, has been very implementation-specific and presents many unforeseen challenges. Some products associated with mobile robotics have reached the "market" stage and have succeeded in their niche but the gap between its technology and their research counterparts is very wide. The *Roomba*(`http://irobot.com`) for example uses behavioral robotics in a *brute force* floor sweeping task. *Kiva*(`http://kivasystems.com`) robots, on the other hand, make use of extensively mapped and controlled environment in order to perform complex warehouse automation tasks (See, e.g. D'Andrea and Wurman (2008)). The recently announced *Neato XV-11*(`http://neatorobotics.com`) claims to fill in this gap to some extent; its the first consumer robot to actually map its surroundings and perform its cleaning task in an *intelligent* way. This paper presents the results of the activity developed with an industrial partner for developing a mobile robot solution that, also aimed at filling the aforementioned gap, performs generic high-level tasks in indoor environments. The robot must be able to localize itself, map it's surroundings and navigate autonomously and consistently within such environment using state-of-art technology. Also it is crucial in achieving our goals that the localization and mapping systems performances are not degraded by robot random movement in reasonable speeds.

The proposed solution has been developed using the Microsoft Robotics Studio – MSRS – framework since it is becoming an industry standard. Software coordination and high-level interface (in which hypothetical users can request high-level tasks) were also developed.

This paper presents an alternative Service Oriented Architecture (SOA) for SLAM modules for mobile robots aiming at flexibility and scalability. This solution represents a different approach to the implementation of Extended Kalman Filter (EKF) localization and mapping (SLAM)(See e.g. Leonard and Durrant-Whyte (1991a) and Leonard and Durrant-Whyte (1991b)) that is independent of feature's nature and measurement model. A different approach to solving common timing problems due to, for example, communication overhead between different services, is proposed. Even though many works aim at *real-time* localization algorithms, they focus at low computing times in order to meet predefined deadlines. Our proposed solution enables the use of more complex SLAM algorithms in situations where normal deadline meeting is not possible by detecting missed deadlines and correcting its effects. Also, an overall description of a software-oriented system focusing on linking the associated theoretical background with its respective implementation is proposed.

2. RELATED WORKS

Chen and Bai (2008) and Lang et al. (2008) address the use of SOA and MSRS for simple applications but do not focus on more complex issues (e.g SLAM, pose control) within the services context. The use of EKF to solve the SLAM problem has been around for many years and most of the recent academic works on SLAM have moved into newer, more sophisticated, alternatives which offer many critical improvements towards some of the pitfalls of that technique. Yan et al. (2009) provide a study on the drawbacks of such technique, which include poor performance when the measurement and motion models are very non linear (such that linearization ceases being

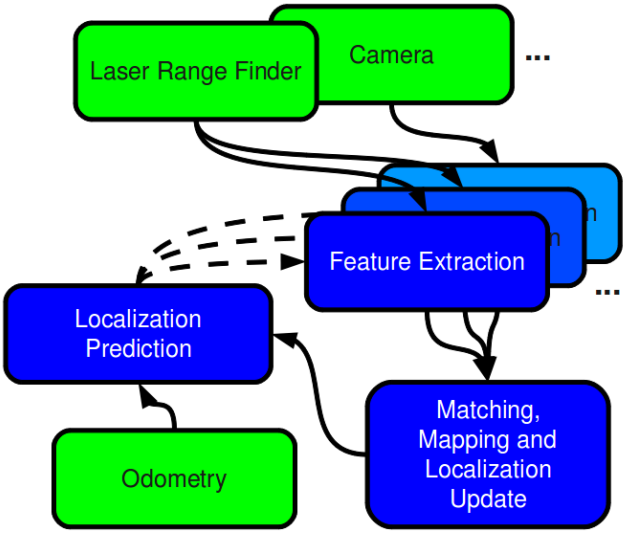


Fig. 1. SLAM Related Services. Green colored services deal with hardware specific data input and blue colored ones lie upon higher abstraction layers. Dashed lines represent optional dependencies.

reasonable), complete unawareness of negative information by the algorithm and no addressing to the *kidnapped robot problem*. More modern techniques include, among others, the Unscented Kalman Filter (UKF) (See, e.g. Holmes et al. (2009)) which differs from the EKF approach in the way linearization is carried but continues employing unimodal belief distributions and thus suffers substantially from the same drawbacks as EKF and the Factored Solution to SLAM (FastSLAM) (See, e.g. Fulgenzi et al. (2009), Grisetti et al. (2007) and Armesto et al. (2008)) which uses Rao Blackwellized Particle Filtering in a hybrid approach that represents robot's position belief by multi modal distributions (discrete particles sets) and map features by unimodal distributions. This last technique stands as the most promising solution since it was shown to be very scalable and won't suffer from the aforementioned drawbacks. Many works address *real-time* constraints for similar purposes with satisfactory results (See, e.g. Newman et al. (2002), Schleicher et al. (2009), Guivant and Nebot (2001), Davison et al. (2007)) but such approaches showed of no avail in our experiments.

3. SERVICES ARCHITECTURE

Figure 1 shows our proposed architecture focusing on services related to SLAM. As a hub for position belief, the *Localization Prediction* service holds the robot's position belief and integrates (i.e. Prediction) such belief upon *Odometry* data. The complementary *Mapping, Matching and Localization Update* service holds the map (i.e. environment) information that is independent of robot position, proceeds in matching observed features to its previously mapped counterparts and computes the posterior, updated, beliefs for map and robot position. The *Feature Extraction* services (note that these are indeed a subgroup of different specialized services) handle data from sensors (e.g. Camera, LRF) and produce abstract, higher level, information; this services manage the sensor specific interfaces generating, from sensor data, lists of generic detected features (e.g. landmarks, lines) expressed in the global reference frame (hence the necessary connections between the *Localization Prediction* and *Feature Extraction* services). Services like *Camera*, *Laser Range Finder* and *Odometry* are hardware specific and each provides a specific generic interface common for such class of sensor, this solution is constantly observed throughout MSRS applications since it facilitates the construction of higher level services independent on underlying hardware technology or simulated device; it depends only on such generic interface. When observing the *Localization Prediction* service, the reader familiar with SLAM implementations will note that two commonly coupled parts of SLAM algorithms are decoupled, such design decision is explained in Section 4.

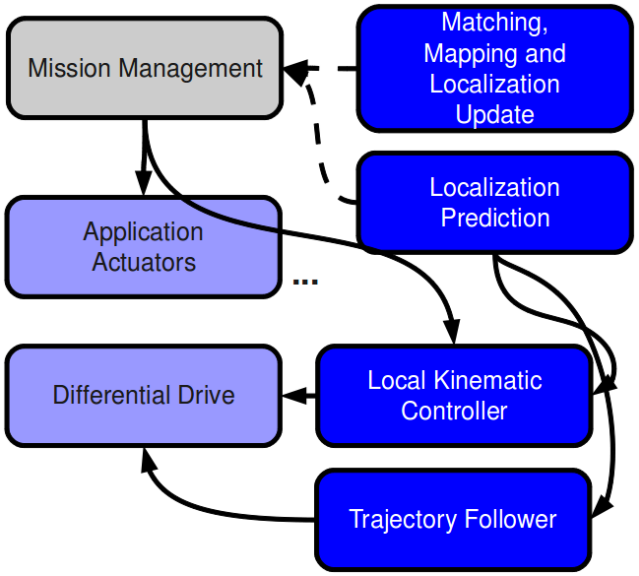


Fig. 2. General purpose services that rely on the localization infrastructure. Violet colored services deal with hardware specific output and the gray one deals with application specific coordination.

Other general purpose services are shown in Figure 2; note that the services that hold the system state appear in both diagrams since they act as an information hub between different systems. Here also, the *Differential Drive* and *Application Actuators* services represent hardware specific services that provide generic interfaces.

3.1 Local Kinematic Controller

Additionally, we implemented some essential control services necessary to truthful evaluation of our system. As an example, *Local Kinematic Controller* is an exponential pose controller used to perform maneuvers like parking, approaching an object, etc. It is primarily based on the controller found in Siegwart and Nourbakhsh (2004) but with some important improvements. In our implementation, the pose feedback signal is retrieved from the current belief of the concurrent SLAM service, this leads to some interesting stability issues since the pose feedback signal is not smooth (belief updates generate "jumps" when recovering from high pose uncertainty situations). In order to assure robust behavior, switching the adequate controller structure (forward or backwards driving) is made possible throughout the whole controller's operation cycle. It is

important to note that a hysteresis based protection mechanism is required in order to prevent oscillations around local minima.

3.2 Feature Extraction Services

Extracting abstract features from raw sensor data is very attractive since it greatly reduces memory consumption and decreases SLAM computational complexity; additionally this features are more human readable. Examples of this class of services are *Line Feature Extraction* and *Reflector Beacon Feature Extraction*. It can be shown that equation 1 gives the theoretical number N_{ideal} of reflecting echoes for a beacon at distance ρ with diameter L_{marker} detected by a Laser Range Finder with resolution R_{laser} expressed in n_{echo}/rad.

$$N_{ideal} = \frac{L_{marker}}{\rho R_{laser}} \tag{1}$$

The detection of the *Reflector Beacon* features consists in finding a cluster of adjacent N_{meas} reflexive echoes such that:

$$N_{meas} \cong N_{ideal} \tag{2}$$

For *Line* features, a *Split and Merge* approach similar to the one described in Borges and Aldon (2000) and Armesto and Tornero (2006) was used. One interesting difference proposed is to use the raw laser range data to infer which lines are adjacent instead of using the proximity of centers of gravity as previously proposed. A more detailed explanation of such algorithms is presented on Algorithm 1.

Algorithm 1. Split and Merge Line Extraction

```
1  LineExtraction(rawData)
2
3  //Split step
4  clusters = split_by_break_point(rawData)
5  lines = least_square_lines(clusters)
6   while max_line_dispersion(lines) >
        threshold
7    for candidate in lines
8     if dispersion(candidate) > threshold
9      line1,line2 = split(candidate)
10     lines.remove(candidate)
11     lines.add(line1)
12     lines.add(line2)
13   lines = least_square_lines(clusters)
14
15 //Merge step
16 lineAdded = 1
17 while lineAdded > 0
18  lineAdded = 0
19  for line1,line2 in adjacent(lines)
20   candidate = merge_lines(line1,line2)
21   if dispersion(candidate) < threshold
22    lines.remove(line1,line2)
23    lines.add(candidate)
24    lineAdded++
```

4. SLAM

We chose EKF for our system since it produces very good results despite its extreme simplicity and it serves for our purposes in showcasing the whole architecture. In our implementation we decoupled the two fundamental sections of the most common EKF SLAM algorithms focusing more on software advantages and flexibility than on mathematical clarity and algorithm correlation. Below, we delve into our implementation of EKF SLAM.

4.1 Splitting Prediction and Update Steps

This design choice enhances our system performance in numerous ways. It is easy to realize that even when no belief update is possible (absence of recognized features), the *prediction* itself is useful as it provides a reasonable position belief together with it's estimated uncertainty. Another key point resides in the heterogeneous execution rates and computation times that each step consumes. Other advantages of this approach include:

- Implicitly solves the problem of multi-rate sensor's data;
- Facilitates the use of different implementations of *Matching, Mapping and Update* algorithms even concurrently;
- Addition of Multiple Hypothesis Tracking (MHT) in the future is straight forward;
- Multiple Robots may share the same instance of the *Matching, Mapping and Update* service across the network (capability inherent from MSRS) in order to perform distributed SLAM;
- Robot's state estimation (which is used intensively by other services) is decoupled from intensive computation services. This provides great performance improvement when in multi-core platforms.

4.2 Prediction

This step involves integrating the state belief by moving the position belief accordingly to the applied control action (inferred from odometry increments) and by updating the belief uncertainty (covariance matrix) that will grow resulting from the motion uncertainty model. The implementation of this step followed the approaches found in Siegwart and Nourbakhsh (2004) and Thrun et al. (2006) without any important changes.

4.3 Data Association & Mapping

Two interesting aspects of our implementation are the extensive use of object oriented polymorphism and the support for "previously known" or "exact" map features. Polymorphism is used to enable the use of heterogeneous features in the same way. The support for static map features is interesting in industrial and domestic applications since it is very reasonable to assume that some features have their exact position known beforehand. For example, in a floor cleaning robot, its docking station may be used as the origin of the global reference frame since it is stationary and preferably exhibits good detectability; in a controlled industrial environment, a small number of

"flagship" landmarks can easily be employed to assure correct correlation between the dynamically generated map and the environment itself. Such landmarks differ from normally mapped ones from not having their position in the estimated state vector and for having very low, static position uncertainty.

4.4 Correction

The Correction step is carried by iterating through a list of pairs that contains already correlated map and measured features; like the prediction step, this implementation follows the literature without any major change. Noting that the Jacobian for the measurement model is selected depending on the feature's nature.

5. REAL-TIME OPERATION

5.1 Proposed Solution

It is quite obvious that real-time operation is crucial to the satisfactory performance of this kind of robots. Related works in this area state that, in order to achieve real-time operation, a complete iteration of the used SLAM algorithm must be completed before it's assigned deadline (which generally is the system's period itself) so that, at each time slot, the robot produces a new state estimation stemming from observations carried inside that time slot. Such approach poses great constraints to both, by the same reason, state estimation frequency and computational complexity of SLAM algorithm.

Our work in this area differs in the way that we assume that those deadlines *won't* be met; this assumption enables our *Prediction* algorithm to run in a much higher rate than the *Update* one. The problem that arises from this assumption is that newer state estimations are delayed by δ_t (which in our experiments could easily reach $1s$) and thus simply replacing the current state by them would generate brutal localization and mapping errors. To solve such problem we start by assuming that each sensor reading has an accurate *timestamp*, assigned as early as possible, and that either all the services share the same clock source (which was the studied case) or all clocks are properly synchronized. Possessing such information, we ensure that the *Prediction* step is computed in real time at frequency f_{odo}, which is reasonable due to the very low computational cost of its operations, and that a circular buffer retains the last K received odometric increments. In this way, when a new pose belief is generated, all odometry increments newer (in relation to its timestamp) than the new belief are used on N successive *simulated* Prediction Steps.

$$N = \delta_t f_{odo} \tag{3}$$

This newer pose belief is then used to replace the current one. This operation is carried if $N < K$, otherwise the update is discarded for being "exceedingly old". Figure 3 shows a conceptual diagram of our solution. It is also important to note that circular buffers are used in each *Feature Extraction* service to hold the last pose beliefs and sensor's readings, each with its timestamp, so that when the newly extracted features are transfered to the

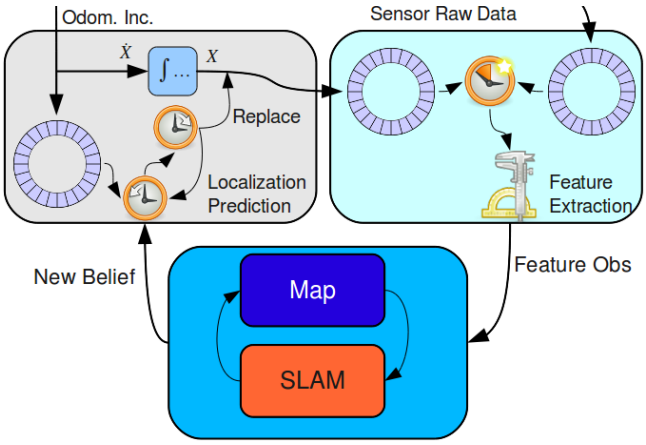


Fig. 3. Deadline extension mechanism where all exchanged data carries timestamps. Data is processed only when there is a timestamp match between the circular buffers referring to each input stream.

global reference frame, the correct belief is used. Similar approaches are already being used as in Lu et al. (2005) and Pornsarayouth and Wongsaisuwan (2009).

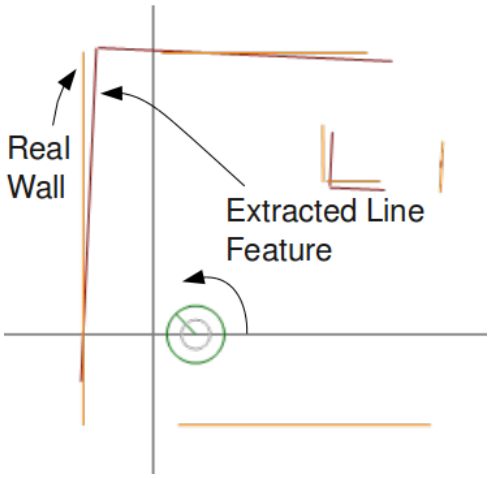
From a control point of view, the stability of the localization algorithm depends on the ratio of uncertainty growth (due to successive predictions) to uncertainty shrinking (due to updates) which must assure a bounded uncertainty for permanent regime states (constant or null robot velocity). In the case where no movement is carried, the uncertainty tends to decrease beyond reasonable values, to avoid this an artificial lower bound is used. Note that such lower bound is imposed to the covariance matrix itself after the update step and thus not affecting eventual accuracy increase. As for constant velocity movement, the uncertainty tends to grow until a stable limit depending on velocity itself as long as N is sufficient.

5.2 Results

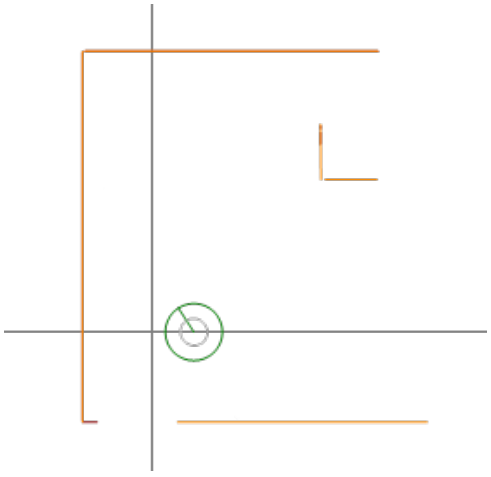
Figure 4 shows the performance of the *Line Feature Extraction* service when the robot is spinning with constant velocity $(+1rad/s)$. Figure 4(a) uses the current (old) localization belief to transform the detected line from the robot's reference frame to the global one. Figure 4(b) shows the same situation but using a localization belief with matching timestamp. The consequences of using an old position belief for the feature extraction can be noted on Figure 4(a) where the extracted features are drifted from real feature. The same behavior is observed in punctual landmarks (e.g. Reflector Beacons).

Table 1. Temporal Aspects

Sensor	Period(ms)	Latency(ms)	Computation Time(ms)
Odometry	50 ± 15	< 2	< 5
Line Features (LRF)	500 ± 50	< 25	150
Reflector Beacon Features (LRF)	500 ± 50	< 25	10
EKF Prediction	-	< 1	< 5
EKF Update	-	< 1	90



(a) Erroneous Transformation



(b) Correct Transformation

Fig. 4. Line Feature Extraction while spinning at $+1rad/s$; orange lines represent real walls.

Table 1 shows many timing aspects of the system. Tasks related to data acquisition and pre-processing have well known periods, latency and computation time. Localization related tasks (Prediction & Update), on the other hand, are triggered by notifications from the *Feature Extraction* services. Note that due to the higher rate of *Odometry* readings, in contrast to *Line Features* ones, an average of five prediction steps need to be re-applied (using the mechanism described in Section 5.1) after a Localization Update steming from *Line Features*.

6. EXPERIMENTS & FIELD APPLICATION

Our experimental setup consists of a *Robulab 80* robot from Robusoft®with an embedded PC-104 running *Windows XP Embedded*®. Since this platform does not support hard real-time operation, the adoption of the proposed solution was necessary in order to accomplish satisfactory performance. This robot uses standard differential drive with a Castor wheel, has precision encoders in each of the two driving wheels and uses a Laser Range Finder able to detect reflecting surfaces.

In a related project, we deployed our system into a simple but demanding application: an automated system for Washing Machine inspection in a reliability test lab (See Cesetti et al. (2010)). In such application, the robot must navigate in narrow corridors on its way to a specific washing machine and then approach it to take specific measurements or perform some operations. Even though the environment is controlled in such application, mapping capabilities are required to cope with moving washing machines, due to its constant vibration, or even with missing ones. Also, another *Feature Extraction* service was developed on top of an existing *Line Feature Extraction* service to detect washing machine patterns showing that those services can be easily stacked and rearranged to achieve diverse features extraction. In this application, which is in its final deployment stages, a supervisory system sends high level task requests (encoded in XML) which are dynamically executed in a FIFO manner.

Videos of these experiments and demonstrations can be found in `www.das.ufsc.br/~scotti/iva/`.

7. CONCLUSION

This paper focused on the deployment of state-of-art robotics theory into a final industrial or domestic solution. In pursuing our goal we developed some interesting mechanisms that may be of great avail for similar works. Our proposed use of SOA seems very promising and it already found its way into an industrial application, we believe that similar schemes will eventually become an industry standard for robotics applications using this kind of architecture. Another important contribution of this paper is our simple mechanism to, in a way, extend real-time imposed deadlines that has showed itself imperative to the successful operation of our system.

REFERENCES

Armesto, L., Ippoliti, G., Longhi, S., and Tornero, J. (2008). Probabilistic self-localization and mapping - an asynchronous multirate approach. *Robotics & Automation Magazine, IEEE*, 15(2), 77–88. doi:10.1109/M-RA.2007.907355.

Armesto, L. and Tornero, J. (2006). Robust and efficient mobile robot self-localization using laser scanner and geometrical maps. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 3080–3085. doi:10.1109/IROS.2006.282325.

Borges, G. and Aldon, M.J. (2000). A split-and-merge segmentation algorithm for line extraction in 2d range images. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, 441–444 vol.1. doi:10.1109/ICPR.2000.905371.

Cesetti, A., Scotti, C.P., Buo, G.D., Babini, M., Donnini, R., Angione, G., Lattanzi, L., Cristalli, C., and Longhi, S. (2010). Field robot supporting the activities of a household appliances laboratory. *IAV 2010, 7th IFAC Symposium on Intelligent Autonomous Vehicles*.

Chen, Y. and Bai, X. (2008). On robotics applications in service-oriented architecture.

D'Andrea, R. and Wurman, P. (2008). Future challenges of coordinating hundreds of autonomous vehicles in distribution facilities. In *Technologies for Practical Robot Applications, 2008. TePRA 2008. IEEE International Conference on*, 80–83. doi:10.1109/TEPRA.2008.4686677.

Davison, A.J., Reid, I.D., Molton, N.D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 6*.

Fulgenzi, C., Ippoliti, G., and Longhi, S. (2009). Experimental validation of fastslam algorithm integrated with a linear features based map. *IFAC Mechatronics 19, 609616*.

Grisetti, G., Tipaldi, G.D., Stachniss, C., Burgard, W., and Nardi, D. (2007). Fast and accurate slam with rao-blackwellized particle filters. *Robot. Auton. Syst.*, 55(1), 30–38. doi:http://dx.doi.org/10.1016/j.robot.2006.06.007.

Guivant, J.E. and Nebot, E.M. (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *Transactions on Robotics and Automation, Vol. 17, No. 3*.

Holmes, S., Klein, G., and Murray, D. (2009). An o(n) square root unscented kalman filter for visual simultaneous localization and mapping. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(7), 1251–1263. doi:10.1109/TPAMI.2008.189.

Lang, H., Wang, Y., and de Silva, C.W. (2008). Mobile robot localization and object pose estimation using optical encoder, vision and laser sensors.

Leonard, J.J. and Durrant-Whyte, H.F. (1991a). Mobile robot localization by tracking geometric beacons. *Transactions on Robotics and Automation*.

Leonard, J. and Durrant-Whyte, H. (1991b). Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, 1442–1447 vol.3. doi:10.1109/IROS.1991.174711.

Lu, X., Zhang, H., Wang, W., and Teo, K.L. (2005). Kalman filtering for multiple time-delay systems. *Automatica*, 41(8), 1455 – 1461.

Newman, P., Leonard, J., Tardos, J., and Neira, J. (2002). Explore and return: experimental validation of real-time concurrent mapping and localization. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, 1802–1809 vol.2. doi:10.1109/ROBOT.2002.1014803.

Pornsarayouth, S. and Wongsaisuwan, M. (2009). Sensor fusion of delay and non-delay signal using kalman filter with moving covariance. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, 2045 –2049.

Schleicher, D., Bergasa, L.M., Ocana, M., Barea, R., and Lopez, M.E. (2009). Real-time hierarchical outdoor slam based on stereovision and gps fusion. *Transactions on Intelligent Transportation Systems, Vol. 10, No. 3*.

Siegwart, R. and Nourbakhsh, I. (2004). *Introduction to Autonomous Mobile Robots*. MIT Press, 2nd edition.

Thrun, S., Burgard, W., and Fox, D. (2006). *Probabilistic Robotics*. MIT Press, 2nd edition.

Yan, J., Guorong, L., Shenghua, L., and Lian, Z. (2009). A review on localization and mapping algorithm based on extended kalman filtering. In *Information Technology and Applications, 2009. IFITA '09. International Forum on*, volume 2, 435–440. doi:10.1109/IFITA.2009.284.