A Tutorial on SLAM and Navigation Problem

By Sylvain Dindy-Bolongo

1.0 Introduction

SLAM (Simultaneous Localization and Mapping) is the process by which a mobile robot can built a map of an environment and at the same time use this map to determine its location[1]. At the beginning both the map of the environment and the robot position are not known, the vehicle has a known kinematic model, the environment within which the robot is moving is populated with artificial or natural landmark. The vehicle is equipped with sensors capable of taking measurement of the relative position between landmarks and the vehicle itself.

This tutorial is an overview of common techniques presently used to solve 3D SLAM.

2.0 SLAM Definition and Process

2.1 State Space Model

Let's consider the problem where a robot is moving in an environment. Initially both the map and the vehicle position are not known but the vehicle has known kinematic model and it is moving through the unknown environment which is populated with artificial or natural landmark.

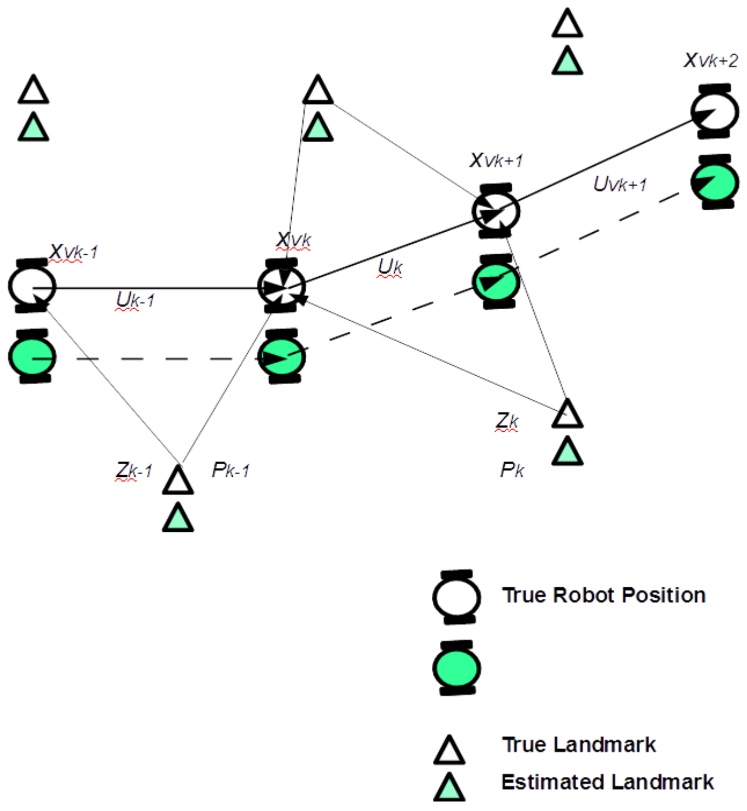

Figure 2.1 The essential SLAM Problem

The autonomous vehicle is moving as shown in Figure 2.1. At a time instant k , the following quantities are defined:

x_{v_k} : The state vector describing the location and orientation or pose of the vehicle

u_{v_k} : The control applied at the time $k-1$ to drive the vehicle to a state x_{v_k} at the time k

z_k : a measurement taken from the vehicle of the location of the ith landmark at the time k

The state of the system at a given moment k is modeled with the state of the vehicle x_{v_k} which contains the robot pose augmented by the 3D position of the landmark. Assuming that N landmarks have been validated and incorporated into the system, then the vector of the landmark is denoted by z_k see Figure 2.1.

$$x_{v_k} = [x_k \quad y_k \quad z_k \quad \psi_k \quad \theta_k \quad \phi_k]^T$$

$$p_k = [x_{1_k} \quad y_{1_k} \quad z_{1_k} \quad \dots \quad x_{N_k} \quad y_{N_k} \quad z_{N_k}]^T \tag{2.1}$$

$$x_k = [x_{v_k} \quad p_k]^T$$

Landmark position is assumed to be constant i.e all landmarks are considered stationary that means $p_k = p_{k-1}$ therefore the model for the evolution of the landmark does not contains any uncertainties[4].

The augmented state vector containing the state of the vehicle and the state of all landmarks (2.1) is then used to estimate a next state by means of the state transition model (2.2).

A state transition matrix F_k, vector of control input u_{v_k} and a vector of temporally uncorrelated process noise w_k with zero mean and with covariance Q_k are used to model the motion of the vehicle through the environment.

$$\begin{bmatrix} x_{v_{k+1}} \\ p_{1_{k+1}} \\ .. \\ p_{N_{k+1}} \end{bmatrix} = \begin{bmatrix} F_k & 0 & .. & 0 \\ 0 & I_{p_1} & .. & 0 \\ .. & .. & .. & 0 \\ 0 & 0 & 0 & I_{p_N} \end{bmatrix} \begin{bmatrix} x_{v_k} \\ p_{1_k} \\ .. \\ p_{N_k} \end{bmatrix} + \begin{bmatrix} u_{v_k} \\ o_{p_1} \\ .. \\ o_{p_N} \end{bmatrix} + \begin{bmatrix} w_{v_k} \\ o_{p_1} \\ .. \\ o_{p_N} \end{bmatrix} \tag{2.2}$$

Where I_{p_i} is an identity matrix of the size equal to the dimension of a landmark which is 3 for 3D landmark and o_{p_i} a 3D zero vector.

2.2 Measurement Model

The measurement model estimate the observation and is given by equation (2.3)

$$z_k = H_k x_k + v_k \tag{2.3}$$

Matrix H_k relates the output of a sensor to the state vector x_k, a vector v_k
for temporally uncorrelated observation, error with zero mean and covariance R_k is used.

3.1 Kalman Filtering (KF)

Kalman filtering is a technique for filtering and predicting linear systems. It is widely used in the control system community to solve stochastic control problem. In the framework of SLAM problem solution, this filter is comprised of three main steps: prediction, observation and update, which keep repeating as the time go by. The method can be described as follows:

a. In the prediction phase the vehicle kinematic model is used to provide an estimate of the state vector $\hat{x}_{k+1}$, the covariance matrix $\hat{P}_{k+1}$ and the measurement $\hat{z}_{k+1}$ at the time $k+1$ from the state vector x_k and the covariance matrix P_k
b. During the measurement phase the innovation vector v_k as the difference between the real and the estimated measurement and the associated covariance matrix Σ are computed
c. During the update phase the matrix Σ is used to compute the Kalman gain K. The gain K and the covariance matrix Σ are used in order to correct the state vector $\hat{x}_{k+1}$ and the covariance matrix $\hat{P}_{k+1}$ into x_{k+1} and P_{k+1}. The overall process can be summarized as follows.

Kalman filtering Algorithm

I. Prediction
   1. $\hat{x}_{k+1} = F_k x_k + u_k$
   2. $\hat{P}_{k+1} = F_k P_k F_k^T + Q_k$
   3. $\hat{z}_{k+1} = H_k \hat{x}_{k+1}$

II. Measurement Phase
   4. z_{k+1} is measured
   5. $v_{k+1} = z_{k+1} - \hat{z}_{k+1}$
   6. $\Sigma_{k+1} = H_k \hat{P}_{k+1} H_k^T + R_k$

III. Update Phase
   7. $K_{k+1} = \hat{P}_{k+1} H_k^T \Sigma_{k+1}^{-1}$
   8. $x_{k+1} = \hat{x}_{k+1} + K_{k+1} v_{k+1}$
   9. $P_{k+1} = (I - K_{k+1}\Sigma_{k+1})\hat{P}_{k+1}$
   10. Iterate with x_{k+1} and P_{k+1}

Where Q_k is the process covariance matrix and R_k the measurement covariance matrix.

3.2 Extended Kalman Filtering (EKF)

The assumption of linear state transition and linear measurement with added Gaussian noise are rarely fulfilled in practice. The motion and measurement equation are nonlinear in general.

Consider the nonlinear system described by the following equations:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) + w_k \\ z_k &= h(x_k, u_k) + v_k \end{aligned} \tag{3.1}$$

Where $f(.)$ models the kinematics and w_k are additive zero mean uncorrelated Gaussian motion disturbances with covariance Q_k and where $h(.)$ describe the geometry of the observation and v_k white Gaussian noise with covariance R_k .

Equation (3.1) can be linearized along a given trajectory resulting in the Extended Kalman Filtering method.

Defined the Jacobians J_{f_k} and J_{h_k} as follows:

$$\begin{aligned} J_{f_k} &= \frac{\partial f(x_{k-1}, u_k)}{\partial x_{k-1}} \\ J_{h_k} &= \frac{\partial h(x_{k-1}, u_k)}{\partial x_{k-1}} \end{aligned} \tag{3.2}$$

A similar iterative process as described for the KF problem can be used and is summarized as follows:

Extended Kalman filtering Algorithm

```
I.    Prediction
      1. $\hat{x}_{k+1} = f(x_k, u_{k+1})$
      2. $\hat{P}_{k+1} = J_{f_k} P_k J_{f_k}^T + Q_k$
      3. $\hat{z}_{k+1} = h(\hat{x}_{k+1})$
II.   Measurement Phase
      4. $z_{k+1}$ is measured
      5. $v_{k+1} = z_{k+1} - \hat{z}_{k+1}$
      6. $\Sigma_{k+1} = J_{h_k} \hat{P}_{k+1} J_{h_k}^T + R_k$
III.  Update Phase
      7. $K_{k+1} = \hat{P}_{k+1} J_{h_k}^T \Sigma_{k+1}^{-1}$
      8. $x_{k+1} = \hat{x}_{k+1} + K_{k+1} v_{k+1}$
      9. $P_{k+1} = (I - K_{k+1}\Sigma_{k+1})\hat{P}_{k+1}$
      10. Iterate with $x_{k+1}$ and $P_{k+1}$
```

Where Q_k is the process covariance matrix and R_k the measurement covariance matrix.

3.3 Compressed Extended Kalman Filtering (CEKF)

The compressed Extended Kalman filtering is an implementation of the EKF where the computational effort is reduced[4]. It takes advantage of the fact that large sequences of prediction and observation step depend only on reduced set of states, because they are independent from most element in the state vector, making unnecessary to perform a full SLAM update in local area.

The CEKF take advantage of this fact by working only over local regions, performing local SLAM with those features that are in the vicinity of the vehicle independently of the size of the of the whole map then a global update is conducted. The Algorithm is described as follows[4].

The robot environment is divided into two regions. A global region denoted B and a local region denoted A and also called active region. This result in the partitioning of the state vector x_k and the covariance matrix P_k into the active part x_{a_k} and P_{aa_k} containing the vehicle state and landmark inside the active region, and the passive part x_{b_k}, P_{bb_k} and P_{ab_k} containing landmark outside the active region. Note that, at the very beginning, x_{a_0} will contain the vehicle state but no landmarks, and the x_{b_0} will be an empty entity. This partitioning is given as follows[4]:

$$x = \begin{bmatrix} x_a \\ x_b \end{bmatrix} \qquad P = \begin{bmatrix} P_{aa} & P_{ab} \\ P_{ba} & P_{bb} \end{bmatrix}$$

The essence of CEKF algorithm is to perform EKF algorithm on x_a and P_{aa}. This is possible thanks to the fact that the innovation matrix Σ_a and the Kalman gain K_a for the local region A are independent from the vector and matrices related to region B, i.e. x_b, P_{bb} and P_a. The final output of this local SLAM is then used to update the global x and P.

In order to perform this global update, the vector x_{b_k} and the matrices P_{bb_k} and P_{ab_k} must be also updated. For this reason several auxiliary computations are added in the prediction and the update stages, without interfering the local SLAM procedure, cost and final result. These extra computations only generates auxiliary variables (ϕ, ψ and θ), that accumulate the effect of each single iteration during the local SLAM process. Define:

$$\phi_0 = I \qquad \phi_{k+1} = (I - \mu_k)\phi_k$$

$$\psi = \overline{0} \qquad \psi_{k+1} = \psi_k + \phi^T \beta_k \phi_k$$

$$\theta_0 = \overline{0} \qquad \theta_{k+1} = \theta_k + \phi_{k-1}^T J_{ha_k}^T \Sigma_{a_k} z_{a_k} \quad (3.4)$$

Where $\beta_k = J_{a_k}^T \Sigma_{a_k} J_{a_k}$, $\mu_k = P_{aa}\beta_k$ and $\psi = \overline{0}$ a $\dim(\psi)$ zero vector.

These variables are then used to update the value of x_{vb} and P_{bb} and P_{ab} to the global map.

$$P_{ab_{k+1}} = \phi_k P_{ab_k}$$

$$P_{bb_{k+1}} = P_{bb_k} - P_{ab_k} \psi_k P_{ab_k}$$

$$x_{vb_{k+1}} = x_{vb_k} - P_{ba_k} \theta_k$$

The CEKF algorithm is given as follows[4]:

CEKF Algorithm

```
1. Define actual active region
2. Define $x_{va_k}$, $P_{aa_k}$ and $x_{vb_k}$, $P_{ab_k}$ and $P_{bb_k}$
3. Initialize $P_{a_k}, J_{a_k}, Q_{a_k}, R_{a_k} u_{a_k}$ for the active region
4. $x_n = x_{va_k}$
5. $P_n = P_{a_k}$
6. $J_{f_n} = J_{fa_k}, J_n = J_{ha_k}, Q_n = Q_{a_k}, R_n = R_{a_k}, u_n = u_{a_k}$
7. $\phi_n = I, \psi_n = \overline{0}, \theta_n = \overline{0}$
8. Active Region Extended Kalman Filter
    I    Prediction
        1. $\hat{x}_{n+1} = f(x_n, u_{n+1})$
        2. $\hat{P}_{n+1} = J_{f_n} P_n J_{f_n}^T + Q_n$
        3. $\hat{z}_{n+1} = h(\hat{x}_{n+1})$
        4. $\phi_n = I, \psi_n = \overline{0}, \theta_n = \overline{0}$
    II.  Measurement Phase
        5. $z_{n+1}$ is measured
        6. $\nu_{n+1} = z_{n+1} - \hat{z}_{n+1}$
        7. $\Sigma_{a_{n+1}} = J_{ha_n} \hat{P}_{n+1} J_{ha_n}^T + R_n$
    III. Update Phase
        8. $K_{n+1} = \hat{P}_{n+1} J_{h_n}^T \Sigma_{n+1}^{-1}$
        9. $x_{n+1} = \hat{x}_{n+1} + K_{n+1} \nu_{n+1}$
        10. $P_{n+1} = (I - K_{n+1} \Sigma_{n+1}) \hat{P}_{n+1}$
        11. $\beta_k = J_{ha_k}^T \Sigma_{a_k}^{-1} J_{ha_k}$
```

12. $\mu_k = P_{aa_k} \beta_k$

13. $\phi_{k+1} = (I - \mu_k)\phi_k$

14. $\psi_{k+1} = \psi_k + \phi_k^T \beta_k \phi_k$

15. $\theta_{k+1} = \theta_k + \phi_{k-1} J_{ha_k}^T \Sigma_{a_k}^{-1} z_{a_k}$

Iterate with $x_{n+1}, P_{n+1}, \phi_{n+1}, \psi_{n+1}, \theta_{n+1}$

Global Update

9. $x_{va_{k+1}} = x_{n+1}$

10. $P_{aa_{k+1}} = P_{n+1}$

11. $x_{vb_{k+1}} = x_{vb_k} + P_{ab_k}^T \theta_{k+1}$

12. $P_{bb_{k+1}} = P_{bb_k} - P_{ab_k}^T \psi_{k+1}$

13. $P_{ab_{k+1}} = \phi_{k+1} P_{ab_k}$

14. $x_{k+1} = \begin{bmatrix} x_{va_k} & x_{vb_k} \end{bmatrix}^T$

15. $P_{k+1} = \begin{bmatrix} P_{aa_k} & P_{ab_k} \\ P_{ba_k} & P_{bb_k} \end{bmatrix}$

Iterate with x_{k+1} and P_{k+1}

3.4 Unscented Kalman Filtering (UKF)

The unscented Kalman filter is based on the idea of unscented transform.

3.4.1 Unscented Transform

The unscented transform is a method for calculating the statistics of a random variable which undergoes a nonlinear transformation . Consider transforming a random variable x (with dimension M) using a nonlinear function $y = f(x)$. Assume that x has a mean $\bar{x}$ and covariance P_x [9].

In order to compute the statistic of y, we build a matrix χ of $2M+1$ sigma vector χ_i according to the following equations:

$$\chi_0 = x$$

$$\chi_i = x + \left(\sqrt{(M+\lambda)P_x}\right)i \qquad i = 1.....M \tag{3.4}$$

$$\chi_i = x - \left(\sqrt{(M+\lambda)P_x}\right)i - M \qquad i = M+1....2M$$

Where $\lambda = \alpha^2(M - \kappa) - M$ is a scaling parameter. The constant α determine the spread of the sigma points around x and is usually set to a small positive value. The constant κ is a secondary scaling parameter which is usually set to 0, and β is used to incorporate prior knowledge of the distribution of

x (for Gaussian distribution $\beta = 2$) . $\left(\sqrt{(M+\lambda)P_x}\right)i$ should be computed using Cholesky factorization. Once sigma vector have been calculated they are propagated through the nonlinear function

$$y_i = f(\chi_i) \qquad i = 0,....2M \tag{3.5}$$

And the weighted mean and covariance are computed as follows

$$y = \sum_{i=0}^{2M} W_i^{(m)} y_i \tag{3.6}$$

$$P_y = \sum_{i=0}^{2M} W_i^{(c)} \{y_i - y\}\{y_i - y\}^T \tag{3.7}$$

$$W_0^m = \frac{\lambda}{M+\lambda} \tag{3.8}$$

$$W_0^c = \frac{\lambda}{M+\lambda} + (1 - \alpha^2 + \beta)$$

$$W_i^m = W_i^c = \frac{\lambda}{M+\lambda} \qquad i = 1.....2M$$

3.4.2 Unscented Kalman Filtering Algorithm

The UKF algorithm has two major phases the prediction phase and the updated phases. First the weighting variable and the sigma are computed, then, the prediction phase begin, by predicting sigma χ_{k+1} for the step which are used in the estimation of the state vector $\hat{x}_{k+1}$ and the covariance matrix $\hat{P}_{k+1}$. The observation estimate corresponding to the predicted sigma ζ_{k+1} are used to predicted the measurement $\hat{z}_{k+1}$. At this point the update begin with the computation of the covariance P_{xx} and the cross covariance P_{xz} that are used to compute the Kalman gain K_{k+1} . The last two steps correct the state vector and the covariance matrix which the will be input for the next UKF iteration.

UKF Algorithm

1. Compute the weight using (3.8)

Compute the sigma points

2. $\gamma = \sqrt{M+\lambda}$

3. $\chi_k = \left[x_k \quad x_k + \gamma\sqrt{P_k} \quad x_k - \gamma\sqrt{P_k}\right]$

I. Prediction Phase

4. $\chi_{k+1} = f(\chi_k, u_k)$

5. $\hat{x}_{k+1} = \sum_{i=0}^{2M} W_i^{(m)} \chi_{k+1}$

6. $\hat{P}_{k+1} = \sum_{i=0}^{2M} W_i^c [\chi_{k+1} - \hat{x}_{k+1}][\chi_{k+1} - \hat{x}_{k+1}]^T + Q_k$

7. $\zeta_{k+1} = h(\chi_{k+1})$

8. $\hat{z}_{k+1} = \sum_{i=0}^{2M} W_i^m \zeta_{k+1}$

II. Update (Correction)

9. $P_{zz} = \sum_{i=0}^{2M} W_i^c [\zeta_{k+1} - \hat{x}_{k+1}][\zeta_{k+1} - \hat{x}_{k+1}]^T + R_k$

10. $P_{xz} = \sum_{i=0}^{2M} W_i^c [\chi_{k+1} - \hat{x}_{k+1}][\chi_{k+1} - \hat{x}_{k+1}]^T$

11. $\kappa_{k+!} = P_{zz} P_{xz}^{-1}$

12. $x_{k+1} = \hat{x}_{k+1} + \kappa_{k+1}(z_{k+1} - \hat{z}_{k+1})$

13. $P_{k+1} = \hat{P}_{k+1} - \kappa_{k+1} P_{zz} \kappa_{k+1}^T$

Iterate with x_{k+1} and P_{k+1}

The unscented Kalman Filter (UKF) has gained popularity because it does not have the linearization step and resulting errors of the EKF[4],[9]. The UKF employs a deterministic sampling strategy to establish the minimum set of points around the mean. This set of point capture the true mean and covariance completely, then these points are propagated through nonlinear functions and the covariance of the estimation can be recuperated. Another advantage of the UKF is its ability to be employed in parallel implementation.

4.0 Particle Filters

Particle Filters are recursive implementation of the sequential Monte Carlo method[8]. This method builds the posterior density using several random samples called particles. Particles are propagated over time with a combination of sampling and re-sampling steps. At each iteration, the sampling step is used in order to discard some particles, increasing the relevance of regions with a higher posterior probability. In the filtering process, several particles of the same state variable are employed, and each particle has an associated weight that indicates the quality of the particle, therefore the estimation is the result of the weighted sum of all particles. The standard particle algorithm has two phases

1. The predicting phase and
2. The updating phase.

In the predicting phase, each particle is modified according to the existing model and account for the sum of random noise to simulate noise effect. Then in the updating phase, the weight of each particle is reevaluated using the last available sensor observation, and particles with lower weight are removed. The general particle filter algorithm comprises the following steps:

1. *Initialization of the particles*

a. Let N be the number of particles

b. $x^{(i)}(1)$ for $i = 1, ...N$

2. *Prediction Step*

a. For each particle i where $i = 1, ...N$ compute $\hat{x}_{k+1|k} = f(\hat{x}_k^{(i)}, u_k) + w_k$ where $f(.)$ is the process dynamic and w_k noise with Cauchy distribution.

3. *Evaluate particle weight*

a. Compute the predicted observation state of the systems using the current predicted state

$z^{(i)}{}_{k+1|k} = h(\hat{x}_{k+1|k}^{(i)}, u_k) + v_{k+1}$ where v_{k+1} has Gaussian distribution.

b. Compute the likelihood (weights) according to the given distribution. Consider

$$likelihood^{(i)} = N(z_{k+1|k}^{(i)}; z_{k+1}^{(i)}, \mathrm{var});$$

c. Normalized the weight as follows

$$\tilde{w}^{(i)} = \frac{likelihood^{(i)}}{\sum_{j=1}^{N} likelihood^{(i)}}$$

4. Resampling/Selection: multiply particles with higher weights and remove those with lower weights. The current state must be adjusted using the computed weights of the new particles.

a. Compute the cumulative weights as follows

$$cumWt^{(i)} = \sum_{j=1}^{i} \tilde{w}^{(j)}$$

b. Generate uniform distributed random variables from $U^{(i)} \sim W(0,1)$ with the number of steps equal to the number of particles.

c. Determine which particles should be multiplied and which ones removed

5. Propagation Phase:

a. Incorporate the new values of the state after the re-sampling of instant k to calculate the value at instant $k+1$. As follows

$$\hat{x}_{k+1|k+1}^{(1:N)} = \hat{x}_{k+1|k}$$

b. Compute the posterior mean as follows

$$\hat{x}_{k+1} = \frac{\sum_{i=1}^{N} x_{k+1|k+1}^{(i)}}{N}$$

c. repeat steps 2 to 5 for each time instant.

Particle filters are more flexible than Kalman Filters and can cope with nonlinear dependencies and non-Gaussian densities in the dynamic model and in the noise error, but they have some disadvantage: a large number of particles are required to obtain a small variance[8].

5.0 Data Association

Data association has always been a critical issues for practical SLAM implementation. Data association arises when landmark cannot be uniquely identified[1],[2],[4]. Data association can simply be presented as feature correspondence problem, which identifies two features observed in different positions and different point in time as being from the same physical object in the world. The two common use of such data association are:

a. Matching two successive scene
b. Closing a loop of a long trajectory when a robot goes back to the starting or previously visited point of the trajectory.

In order to successfully solve the correspondence problem, selection of robust features are necessary under weak lightning position or different points of view.

6.0 Feature Extraction

Feature detection, tracking and 3D reconstruction are important step in the SLAM process since they feed the measurement into the SLAM process. Feature detection consist in estimating the locations of features in image sequences using detectors such as Harris Corners, Random Sample Consensus (RANSAC), Scale Invariant Feature Transform (SIFT) or Speed Up Robust Feature (SURF)[7],[3]. 3D reconstruction is the problem of obtaining the 3D coordinates and the camera pose using two or more 2D images using epipolar geometry and fundamental matrix for example.

7.0 Conclusion

In this tutorial we covered several algorithms that are commonly used to solve 3D SLAM problem including:

- Kalman Filters
- Extended Kalman Filters
- Unscented Kalman Filters
- Particle Filters

A discussion of some of the robust feature extraction techniques in addition to 3D reconstruction problem and data association problem was presented.

7.0 Reference

[1] Durrant-Whyte Hugh, Tim Bailey "Simultaneous Localization and Mapping: Part I"
IEEE Robotics & Automation Magazine, June 2006
[2] Bailey Tim, Hugh Durrant-Whyte "Simultaneous Localization and Mapping: Part II"
IEEE Robotics & Automation Magazine, September 2006
[3] Newman Paul Michael "EKF Based Navigation and SLAM"
SLAM Summer School 2006, Oxford
[4] Aulinas Josep "3D Visual SLAM applied to Large Scale Underwater Scenarios"
Msc Thesis: Institute of Informatic and Application, University of Girons, Girons Spains, 2008
[5] Siegwert Roland, R. Nourbakash "Introduction to Autonomous Mobile Robots"
A Bradford Book, The MIT Press, Cambridge Mass, 2004
[6] Kelly Alonzo "3D State Space Formulation of a Navigation Kalman filter for Autonomous Vehicle"
The Robotics Institute Carnegie Mellon University, CMU-RI-TR-94-19-REV 2.0 May 2006
[7] Bradski Gary, Adran Kaeller "Learning OpenCV"
O'Reilly, September 2008: First Edition
[8] Castenedo Frederico "A review of Data Fusion Techniques"
Hidawi Publishing Corporation, The Scientific World Journal,
Volume 2013, Article ID 704504, 19 Pages.
http://dx.doi.org/10.11551/2013/704504
[9] Chu Fang-I "From Baysian to Particle Filter
MA Thesis, San Francisco State University, 2009

About the author:

Sylvain Dindy-Bolongo is currently Principal at VizCortex, prior to that, he worked as Software Engineer, Embedded Software Engineer, Control System Engineer, Control Architect for several companies in the San Francisco Bay Area, including HP, Agilent, Schneider Electric,
Tech-Mahindra, as well as a few startups. He also worked as Part-Time Instructor for the Art Institute of California at San Francisco, University of California at Bekeley Extension, Sonoma State University (Master of Engineering Program).
His interests include Robotics (SLAM Navigation, Manipulator Control), Machine Learning, Deep Learning, Computer Vision, Control Systems (Robust, Intelligent, Embedded).
He earned his BEE, MSEE from Ecole Polytechnique de Montreal (University of Montreal, Canada) and a PhD in Control Systems and Robotics from Wichita State University, Kansas.