# VISION-BASED MOBILE ROBOT NAVIGATION

*Sid Ahmed Berrabah, Eric Colon*


## Abstract:

*This paper presents a vision-based navigation system for mobile robots. It enables the robot to build a map of its environment, localize efficiently itself without use of any artificial markers or other modifications, and navigate without colliding with obstacles.*

*The Simultaneous Localization And Mapping (SLAM) procedure builds a global representation of the environment based on several size limited local maps built using the approach introduced by Davison* et al. *[1]. Two methods for global map are presented; the first method consists in transforming each local map into a global frame before to start building a new local map. While in the second method, the global map consists only in a set of robot positions where new local maps are started (i.e. the base references of the local maps). In both methods, the base frame for the global map is the robot position at instant* $t_0$.

*Based on the estimated map and its global position, the robot can find a path and navigate without colliding with obstacles to reach a goal defined the user. The moving objects in the scene are detected and their motion is estimated using a combination of Gaussian Mixture Model (GMM) background subtraction approach and a Maximum a Posteriori Probability Markov Random Field (MAP-MRF) framework [2].*

*Experimental results in real scenes are presented to illustrate the effectiveness of the proposed method.*


***Keywords:*** *robot navigation, vision-based SLAM, local and global mapping, adaptive fuzzy control.*

## 1. Introduction

In recent years, an increasing amount of robotic research has focused on the problem of map building, navigation and executing task motion autonomously, i.e. without human guidance. Such ability is essential for robotic systems in tile emerging field of service robotics, which includes security guarding, waste management, cleaning, and others. To reach a reasonable degree of autonomy, two basic requirements are sensing and reasoning. Sensing is provided by an on board sensory system that gathers information about the robot itself and the surrounding environment. Reasoning is accomplished by developing algorithms that exploit this information in order to generate appropriate commands for the robot.

The reasoning system is the central unit in an autonomous robot. According to the environment state, it must allow the robot to localize itself in the environment and seek for free paths. To accomplish these two tasks, it bases its reasoning on a model or a description of the environment. The environment model is not always available and hence the robot should have the means to build such a model over time as it explores its environments. This latest problem is known as mapping problem.

*Mapping and localization are interlinked problems:* If the robot's pose (spatial position and orientation) is known all long, building a map would be quite simple. Conversely, if a map of the environment exists already, it will be very easy to determine accurately the robot's pose at any time. In combination, however, the problem is much harder. The literature refers to the mapping problem often in conjunction with the localization problem named as Simultaneous Localization and Mapping (SLAM) [3, 4] or Concurrent Mapping and Localization (CML) [5, 6, 7].

In this study, we focus on monocular vision-based SLAM where a single camera is moving through the scene. This is interesting because it offers a low-cost and real-time approach to SLAM in unprepared environments. The camera identifies natural features in the scene and uses these as landmarks in its map. The proposed approach is an extension to a real-time SLAM approach, proposed by Davison et al. [1], which recovers the 3D trajectory of a monocular camera, moving through previously unknown room-size indoor domains. The role of the map in [1] is primarily to permit real-time localization rather than serve as a complete scene description of the scene.

This method is however not suitable in large environments [1], and particularly in case of perceptual aliasing, which refers to situations where several places are perceptually similar enough to be confused by the robot. There-fore, currently observable features alone may not be sufficient to uniquely identify the robot's true location. To overcome these problems, we propose the use of a 'history memory', which accumulates sensory evidence over time to identify places with a stochastic model of the correlation between map features. This also will allow obtaining a complete description of the scene organized as global map composed of several small local maps.

An other problem for SLAM methods is change over time of the environment. Some changes may be relatively slow, such as the change of appearance of a tree across different seasons, or the structural changes that most office buildings are subjected to over time. Others are faster, such as the change of door status or the location of furniture items, such as chairs. Even faster may be the change of location of other agents in the environment, such as cars or people. To deal with this problem of dynamic scenes (with moving objects) we use an algorithm for motion segmentation [2] to remove the outliers features

which are associated with moving objects.

Based on the estimated global map, the robot can fined a path to reach a user defined goal and it uses a procedure based on fuzzy logic control to navigate and avoid unplanned obstacles in the SLAM procedure and the detected moving obstacles. To predict the position of the detected moving objects, Kalman filter tracking procedure is applied to the output object mask of the motion detection process. The fuzzy logic controller for obstacle avoidance is equipped with reinforcement learning algorithm, which consists of a scalar reinforcement signal as a performance feedback from the environment enabling the navigator to tune itself.

The rest of the paper is organized as follows: Section 2 describes the different parts of the proposed approach. Experimental results are shown in section 3, followed by conclusions in section 4.

## 2. Method overview

### 2.1. Vision-based simultaneous localization and mapping

The SLAM problem is tackled as a stochastic problem using an Extended Kalman Filtering to maintain the state vector, $\mathbf{x}$, consisting of the robot state, $\mathbf{x}_R$, and map feature states, $\mathbf{x}_{L_i}$. It also maintains a covariance matrix $\mathbf{P}$, which includes the uncertainties in the various states as well as correlations between the states.

In [1], feature states $\mathbf{x}_{L_i}$ are the 3D position vectors of the locations of point features and the (robot) camera state vector $\mathbf{x}_R$ comprises a metric 3D position vector $\mathbf{r}^{\mathcal{W}}$, orientation quaternion $\mathbf{q}^{\mathcal{RW}}$, velocity vector $\nu^{\mathcal{W}}$, and angular velocity vector $\omega^{\mathcal{R}}$ relative to a fixed world frame $\mathcal{W}$ and 'robot' frame $\mathcal{R}$ carried by the camera (13 parameters):

$$\mathbf{x}_R = \begin{bmatrix} \mathbf{r}^{\mathcal{W}} \\ \mathbf{q}^{\mathcal{RW}} \\ \nu^{\mathcal{W}} \\ \omega^{\mathcal{R}} \end{bmatrix} \tag{1}$$

Features are image patches with a size of 11 × 11 pixels. The patches are supposed locally as planar surfaces perpendicular to the vector from the feature to the camera at initialization. When making measurements of a feature from new camera positions, each patch can be projected from 3D to the image plane to produce a template for matching with the real image. This template will be a warped version of the original square template captured when the feature was first detected.

To avoid using outlier features, the moving object mask detected by the motion segmentation procedure intro-duced in [2] is used. Subsequently, during map building, the detected features on the moving parts are excluded.

The coordinate frame is chosen at the starting position and the system is helped at the beginning with an amount of prior information about the scene. A shape of a known target is placed in front of the camera, which provides several features with known positions and known appearance. This will help to know the scaling of the estimated map and an initialization of the map as with only a single camera, the features cannot be initialized on the map based only on one measurement because of their unknown depth. The system starts with zero uncertainty.

A constant velocity model is considered and the robot (camera) state update is produced by:

$$\begin{aligned} \mathbf{x}^t &= \begin{bmatrix} \mathbf{r}^t \\ \mathbf{q}^t \\ \nu^t \\ \omega^t \end{bmatrix} = \mathbf{f}\left(\mathbf{x}^{t-1}\right) + \mathbf{v}^t = \\ &= \begin{bmatrix} \mathbf{r}^{t-1} + (\nu^{t-1} + \mathbf{V})\Delta t \\ \mathbf{q}^{t-1} + q((\omega^{t-1} + \mathbf{\Omega}^{t-1})\Delta t) \\ \nu^{t-1} + \mathbf{V}^{t-1} \\ \omega^{t-1} + \mathbf{\Omega}^{t-1} \end{bmatrix} \end{aligned} \tag{2}$$

where $\mathbf{f}$ is the transition function and $\mathbf{v}$ is a random vector describing the unmodelled aspects of the vehicle. $\mathbf{V}$ and $\mathbf{\Omega}$ are respectively the impulse of velocity and the angular velocity caused by unknown acceleration and angular acceleration processes of zero mean and Gaussian distribution. $q((\omega^t + \mathbf{\Omega}^t)\Delta t)$ denotes the quaternion trivially defined by the angle-axis rotation vector $(\omega^t + \mathbf{\Omega}^t)\Delta t$.

Feature matching is carried out using straightforward normalization cross-correlation search for the template patch projected into the current camera estimate. The image is scanned at each location until a match is found. This searching for match is computationally expensive.

Considering a perspective projection, the position $\mathbf{x}^t_{L_i} = (x^t_{L_i}, y^t_{L_i})$ at which the feature $L_i$ would be expected to be found in the image is found using the standard pinhole model:

$$\mathbf{x}^t_{L_i} = \begin{bmatrix} x^t_{L_i} \\ y^t_{L_i} \end{bmatrix} = \begin{bmatrix} x_0 - \kappa_x \frac{x^t_{L/\mathcal{R}}}{z^t_{L/\mathcal{R}}} \\ y_0 - \kappa_y \frac{y^t_{L/\mathcal{R}}}{z^t_{L/\mathcal{R}}} \end{bmatrix} \tag{3}$$

where $\kappa_x, \kappa_y, x_0$ and $y_0$ are the standard camera calibration parameters

$$\mathbf{x}^t_{L/\mathcal{R}} = (x^t_{L/\mathcal{R}}, y^t_{L/\mathcal{R}}, z^t_{L/\mathcal{R}})$$

is the 3D position of the feature relative to the camera.

The EKF consists in prediction and update steps. At each time step of the filter we obtain the predicted state $\mathbf{x}$ and covariance $\mathbf{P}$ using the state transition function. The Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, which appears as a result of the linearization process of the transition function $\mathbf{f}$, is used to transfer the map covariance, with the addition of the process noise covariance $\mathbf{Q}$.

$$\mathbf{x}^{t|t-1} = \mathbf{f}\left(\mathbf{x}^{t-1|t-1}, u^t\right) \tag{4}$$

$$\mathbf{P}^{t|t-1} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{t-1|t-1} \mathbf{P}^{t-1|t-1} \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^T\right|_{t-1|t-1} + \mathbf{Q}^{t-1} \tag{5}$$

The covariance of the state, $\mathbf{P}$, is therefore in block form, containing the vehicle covariance $P_{RR}$, the feature covariances, $P_{L_iL_i}$, the cross-covariances between features, $P_{L_iL_j}$, and the cross-covariances between the features and the vehicle, $P_{RL_i}$.

The state transition function $\mathbf{f}$ does not alter the feature states. A result of this is that its Jacobian is very simple:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{x}_R} & 0_{3\times3} & 0_{3\times3} & \cdots \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} & \cdots \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{6}$$

and the process noise

$$\mathbf{Q} = [\mathbf{Q}_R \quad 0 \quad \cdots]^T \tag{7}$$

The update to include a new measurement incorporates the innovation $\varepsilon$, which is the difference between the measurement and its prediction.

$$\mathbf{x}^{t|t} = \mathbf{x}^{t|t-1} + \mathbf{W}^t \varepsilon^t \tag{8}$$

$$\mathbf{P}^{t|t} = \mathbf{P}^{t|t-1} - \mathbf{W}^t \mathbf{S}^t \mathbf{W}^{tT} \tag{9}$$

Where

$$\mathbf{W}^t = \mathbf{P}^{t|t-1} \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^T\right|_{t-1|t-1} (\mathbf{S}^{-1})^t \tag{10}$$

$$\mathbf{S}^t = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{t-1|t-1} \mathbf{P}^{t|t-1} \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^T\right|_{t-1|t-1} + \mathbf{R}^t \tag{11}$$

$$\varepsilon = \mathbf{z}^t - \mathbf{h}(\mathbf{x}^{t|t-1}) \tag{12}$$

$\mathbf{Q}$ and $\mathbf{R}$ are block-diagonal matrices obtained empirically defining the error covariance matrices characterizing the noise in the model and the observations respectively.

A measurement of feature $\mathbf{x}_i$ is not related to the measurement of any other feature so

$$\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}} = [\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_R} \quad 0 \quad 0 \cdots \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_i} \quad 0 \quad \cdots] \tag{13}$$

where $\mathbf{h}_i$ is the measurement model for the $i$'th feature and then

$$\begin{aligned} \mathbf{S} = &\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_R} P_{RR} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_R}^T + \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{L_i}} P_{L_iR} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_R}^T + \\ &+ \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_R} P_{RL_i} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{L_i}}^T + \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{L_i}} P_{L_iL_i} \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_{L_i}}^T + \mathbf{R} \end{aligned} \tag{14}$$

which depends only on the measurements of the feature $i$ and the vehicle state.

After identification of the first measurement a new feature is to initialize a 3D line into the map along which the feature must lie. This is a semi-infinite line, starting at the estimated camera position and heading to infinity along the feature viewing direction. All possible 3D locations of the feature point lie somewhere along this line. To estimate the correct location (its depth), a set of discrete depth hypotheses is uniformly distributed along this line, which can be thought of as a one-dimensional probability density over depth represented by a 1D particle distribution or histogram. The observations of the new feature in the next few time-steps will provide information about its depth coordinate.

### 2.2. Global mapping for large environment

The SLAM algorithm presented in the previous sections has two important limitations when mapping large environments. First, the computational cost of updating the map grows with $O(N^2)$ ($N$ = number_of_features + size_of_robot_stat). Second, as the map grows, the estimates obtained by the EKF equations quickly become inconsistent due to linearization errors [8].

To overcome these limitations, we use the previous algorithm to build small independent local maps of limited size and join them in global one. For joining the local maps we propose two approaches, the first method consists in transforming each local map into a global frame before to start building a new local map. While in the second method, the global map consists only in a set of robot positions where new local maps are started (i.e. the base references of the local maps). In both methods, the base frame for the global map is the robot position at instant $t_0$, $\mathbf{x}_R^{t_0}$.

Each local map can be built as follows: at a given instant $t_k$, a new map is initialized using the current vehicle location, $\mathbf{x}_R^{t_k}$ as base reference $B_i = \mathbf{x}_R^{t_k}$, $i = 1, 2, ...$ is the local map order. Then, the vehicle performs a limited motion acquiring sensor information about the neighbouring environment features $L_i$. The EKF-based technique presented in the previous section is used to obtain a local map:

$$\mathfrak{M}_i = \mathfrak{M}_{L_i}^{B_i} = (\mathbf{x}_{L_i}^{B_i}, \mathbf{P}_{L_i}^{B_i}) \tag{15}$$

where $\mathbf{x}_{L_i}^{B_i}$ is the coordinate vector in the base reference $B_i$ of the $L_i$ detected features and $\mathbf{P}_{L_i}^{B_i}$ is their covariance matrix estimated in $B_i$.

The robot decides to start a new local map when the number of mapped features reaches a pre-defined threshold.

#### *2.2.1. First method for global map building*

In this method the first local map is used as global map. Each finalized local map is transferred to the global map before starting a new one, by computing the state vectors and the covariance matrix. The goal of map joining is to obtain one full stochastic map:

$$\mathfrak{M}_G = (\mathbf{x}_{(L_0+L_1+L_2+...)}^{B_0}, \mathbf{P}_{(L_0+L_1+L_2+...)}^{B_0}) \tag{16}$$

where $\mathbf{x}_{(L_0+L_1+L_2+...)}^{B_0}$ is a concatenation of all features from local maps $0, 1, 2...$

$$\mathbf{x}_{(L_0+L_1+L_2+...)}^{B_0} = (\mathbf{x}_{L_0}^{B_0}, \mathbf{x}_{L_1}^{B_0}, \mathbf{x}_{L_2}^{B_0}, ...) \tag{17}$$

The location of feature $j$ from the local map $\mathfrak{M}_1$ is given in the frame $B_0$ as follows:

$$\begin{aligned} \mathbf{x}_{L_{j,1}}^{B_0} = &\begin{bmatrix} x_{L_{j,1}}^{B_1} \\ y_{L_{j,1}}^{B_1} \\ \theta_{L_{j,1}}^{B_1} \end{bmatrix} + \\ &+ \begin{bmatrix} x_R^{t_{k_1}/B_0} cos(\theta_R^{t_{k_1}/B_0}) - y_R^{t_{k_1}/B_0} sin(\theta_R^{t_{k_1}/B_0}) \\ x_R^{t_{k_1}/B_0} sin(\theta_R^{t_{k_1}/B_0}) + y_R^{t_{k_1}/B_0} cos(\theta_R^{t_{k_1}/B_0}) \\ \theta_R^{t_{k_1}/B_0} \end{bmatrix} \end{aligned} \tag{18}$$

where $\mathbf{x}_{L_{j,1}}^{B_1} = (x_{L_{j,1}}^{B_1}, y_{L_{j,1}}^{B_1}, \theta_{L_{j,1}}^{B_1})$ is the feature location

in the local map $\mathfrak{M}_1$and
$\mathbf{x}_R^{t_{k_1}/B_0} = (x_R^{t_{k_1}/B_0}, y_R^{t_{k_1}/B_0}, \theta_R^{t_{k_1}/B_0})$
is the $\mathfrak{M}_1$ frame reference in the global frame $B_0$.

The same way, the location of feature $j$ from local map $\mathfrak{M}_1$ is given in the frame $B_0$ as follows:

$$\mathbf{x}_{L_{j,i}}^{B_0} = \begin{bmatrix} x_{L_{j,i}}^{B_i} \\ y_{L_{j,i}}^{B_i} \\ \theta_{L_{j,i}}^{B_i} \end{bmatrix} + \begin{bmatrix} x_R^{t_{k_i}/B_0} cos(\theta_R^{t_{k_i}/B_0}) - y_R^{t_{k_i}/B_0} sin(\theta_R^{t_{k_i}/B_0}) \\ x_R^{t_{k_i}/B_0} sin(\theta_R^{t_{k_i}/B_0}) + y_R^{t_{k_i}/B_0} cos(\theta_R^{t_{k_i}/B_0}) \\ \theta_R^{t_{k_i}/B_0} \end{bmatrix} \tag{19}$$

where $\mathbf{x}_{L_{j,i}}^{B_i} = (x_{L_{j,i}}^{B_i}, y_{L_{j,i}}^{B_i}, \theta_{L_{j,i}}^{B_i})$

is the feature location in the local map $\mathfrak{M}_i$ and

$\mathbf{x}_R^{t_{k_i}/B_0} = (x_R^{t_{k_i}/B_0}, y_R^{t_{k_i}/B_0}, \theta_R^{t_{k_i}/B_0})$

is the $\mathfrak{M}_i$ frame reference in the global frame $B_0$:

$$\mathbf{x}_R^{t_{k_i}/B_0} = \begin{bmatrix} x_R^{t_{k_{i-1}}/B_{i-1}} \\ y_R^{t_{k_{i-1}}/B_{i-1}} \\ \theta_R^{t_{k_{i-1}}/B_{i-1}} \end{bmatrix} + \begin{bmatrix} x_R^{t_{k_{i-2}}/B_{i-2}} cos(\theta_R^{t_{k_{i-2}}/B_{i-2}}) - \\ x_R^{t_{k_{i-2}}/B_{i-2}} sin(\theta_R^{t_{k_{i-2}}/B_{i-2}}) + \\ \theta_R^{t_{k_{i-2}}/B_{i-2}} \end{bmatrix} + \begin{bmatrix} - y_R^{t_{k_{i-2}}/B_{i-2}} sin(\theta_R^{t_{k_{i-2}}/B_{i-2}}) \\ + y_R^{t_{k_{i-2}}/B_{i-2}} cos(\theta_R^{t_{k_{i-2}}/B_{i-2}}) \end{bmatrix} + \ldots + \begin{bmatrix} x_R^{t_{k_1}/B_0} cos(\theta_R^{t_{k_1}/B_0} - y_R^{t_{k_1}/B_0} sin(\theta_R^{t_{k_1}/B_0} \\ x_R^{t_{k_1}/B_0} sin(\theta_R^{t_{k_1}/B_0} + y_R^{t_{k_1}/B_0} cos(\theta_R^{t_{k_1}/B_0}) \\ \theta_R^{t_{k_1}/B_0} \end{bmatrix} \tag{20}$$

The covariance $\mathbf{P}_{(L_0+L_1+L_2+\ldots)}^{B_0}$ of the joint map is obtained from the linearization of the state transition function $\mathbf{f}$. As the local maps are independent, the Jacobian (from linearization) is then applied separately to the local map covariance:

$$\mathbf{P}_{(L_0+L_1+L_2+\ldots)}^{B_0} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}_{L_0}} P_0 \frac{\partial \mathbf{f}^T}{\partial \mathbf{x}_{L_0}} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}_{L_1}} P_1 \frac{\partial \mathbf{f}^T}{\partial \mathbf{x}_{L_1}} + \ldots + \frac{\partial \mathbf{f}}{\partial \mathbf{x}_{L_i}} P_i \frac{\partial \mathbf{f}^T}{\partial \mathbf{x}_{L_i}} + \ldots \tag{21}$$

*2.2.2. Second method for global map building*

In this method, the global map is limited to the set of the coordinates of the local maps frame origins.

$$\mathfrak{M}_G^B = (\mathbf{x}_R^{t_{k_0}/B_0}, \mathbf{x}_R^{t_{k_1}/B_0}, \mathbf{x}_R^{t_{k_2}/B_0}, \ldots) \tag{22}$$

where $t_{k_0}, t_{k_1}, t_{k_2}, \ldots$ are the instants when the robot decides to build a new local map.

$t_{k_0} = 0$ and $\mathbf{x}_R^{t_{k_0}/B_0} = (0,0,0)$.

If at instant $t_{k_i}$, the robot decides to build a new local map $\mathfrak{M}_i$ and its location in the base frame $B_{i-1}$ of the local map $\mathfrak{M}_{i-1}$, is

$\mathbf{x}_R^{t_{k_i}/B_{i-1}} = (x_R^{t_{k_i}/B_{i-1}}, y_R^{t_{k_i}/B_{i-1}}, \theta_R^{t_{k_i}/B_{i-1}})$

then:

$$\mathbf{x}_R^{t_{k_i}/B_0} = \begin{bmatrix} x_R^{t_{k_i}/B_{i-1}} \\ y_R^{t_{k_i}/B_{i-1}} \\ \theta_R^{t_{k_i}/B_{i-1}} \end{bmatrix} + \begin{bmatrix} x_R^{t_{k_{i-1}}/B_0} cos(\theta_R^{t_{k_{i-1}}/B_0}) - y_R^{t_{k_{i-1}}/B_0} sin(\theta_R^{t_{k_{i-1}}/B_0}) \\ x_R^{t_{k_{i-1}}/B_0} sin(\theta_R^{t_{k_{i-1}}/B_0}) + y_R^{t_{k_{i-1}}/B_0} cos(\theta_R^{t_{k_{i-1}}/B_0}) \\ \theta_{k_{i-2}}^{B_0} \end{bmatrix} \tag{23}$$

In this case, for feature matching at instant $t$, the robot uses the local map with closest base frame to its current location:

$$\min_i(\| \mathbf{x}_R^{t_{k_i}/B_0} - \mathbf{x}_R^{t/B_0} \|) \tag{24}$$

### 2.3. Path planning

A path planning system is used then by the robot to navigate in its environment and avoid unplanned obstacles by the SLAM procedure and the detected moving obstacles. The position of the detected moving objects is predicted by a Kalman filter tracking procedure applied to the output object mask from the motion detection process [2] and the used sensors for static obstacles detection are infrared and ultrasound sensors.

The local path planning procedure is based on two fuzzy logic controllers: a goal seeking controller and an obstacle avoidance controller. The goal seeking controller tries to find the optimal path to the intermediate goals (defined by the global path planning), while the obstacle avoidance controller has the mission to avoid obstacles. A command fusion scheme based on a conditioned activation for each controller arbitrates between the two behaviours.

Simple fuzzy control for the obstacle avoidance behaviour is suitable to design autonomous mobile robots. However, it is difficult to maintain the correctness, consistency, and completeness of the fuzzy rule base for obstacle avoidance constructed and tuned by a human expert. Therefore, a fuzzy system able to evolve and automatically improve its performance is recommended. Several learning algorithms have been proposed to construct fuzzy systems automatically, such as back-propagation algorithm [10], table-lookup scheme [10], evolutionary algorithm [13], and reinforcement learning [11, 12]. In our application we used reinforcement learning as it is an unsupervised method which does not need training data. This learning method requires only a scalar reinforcement signal as a performance feedback from the environment enabling the navigator to tune itself [12].

According to requirements in present robotic software architectures, being object-oriented component

based reusable software patterns, we developed such a framework, named Controlling Robots with CORBA [14] (CoRoBa), written in C++ and based on CORBA, a standardized and popular middleware. CoRoBa relies on well-known Design Patterns [15] and provides a component based development model. Concerning the tuning and simulation of navigation algorithms, a Java based multi mobile robot simulator (MoRoS3D) exists, and integrates seamlessly in the framework as described in detail in [16].

*2.3.1. Goal seeking controller*

Given the polar coordinates of the target in the robot frame, a Sugeno type fuzzy controller calculates the turn angle allowing the robot to reach the target.

The transfer function of the goal seeking controller representing the dependency of the turn angle on the goal distance and the goal angle (polar coordinates) is shown in Fig. 1. We can see that the turn angle to apply to the robot to reach the target is directly proportional to the goal angle and inversely proportional to the goal distance with more importance to the goal coordinates.

The goal distance and goal angle are fuzzified into 4 and 11 Gaussian fuzzy sets, respectively. The turn angle can take 11 crisp value in the interval [-180°,180°].

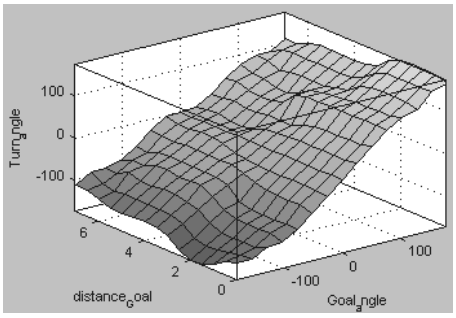


*Fig. 1. The transfer function of the goal seeking controller.*

*2.3.2. Obstacle avoidance controller*

The 0-order Takagi-Sugeno fuzzy controller used for obstacle avoidance is based on the distances to the obstacle detected by the sonar and infrared sensors as well as the estimated distance to the detected moving obstacles.

The front ultrasonic and infrared sensors of the robot are grouped in 7 groups as depicted in Fig. 2. The sensor data from groups G1 to G5 and the goal angle are used as inputs to the obstacle avoidance controller. Where as the data from the sensors of the groups G6 and G7 are used only in the mediation process. The output of the controller is the turn angle, which ranges between [-180°,180°].

The distance $d_i$ measured by the $i^{th}$ sensor group is expressed as:

For $i = 1,..,5,$

$$d_i = min\{d_{imax}, min_j\{\{d_{ij}\}\}; \quad j = 1..2 \tag{25}$$

where $d_{ij}$ is the distance measured by the $j^{th}$ sensor of the sensor group $i$. The distance measured by each sensor group $G_i$ is fuzzified into three Gaussian fuzzy sets.

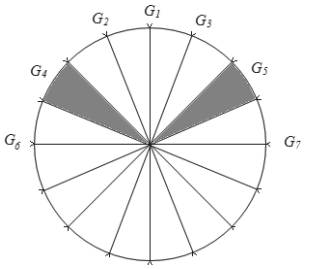


*Fig. 2. Diagram of the arrangement of the robot sensors.*

*2.3.3. Reinforcement learning for the obstacle avoidance controller*

Reinforcement learning is a control strategy in which the robot embedded in an environment attempts to maximize total reward in pursuit of a goal [12]. At any time step, the robot should take a decision to avoid the obstacles or to reach it goal. When the decision is taken and the action performed, the decision process receives a feedback in the form of a reward from the environment which indicates if a good or bad decision to take in attempting to achieve the goal.

As the learning concerns only the obstacle avoidance controller, the action corresponds to the possible scalar outputs of the controller and the states correspond to the different combinations of the input fuzzy variables of the controller.

The $Q$ learning algorithm used in our application is as follows:

1. Initialize $Q(s, a)$ to 0 for all state $s$ and action $a$
2. Perceive current state $s$
3. Choose an action $a$ according to action value function
4. Carry out action $a$ in the environment. Let the next state be $s'$and the reward be $r$.
5. Update action value function from $s, a, s'$, and $r$:

$$Q_{t+1}(s,a) = (1-\alpha_t)Q_t(s,a) + \alpha_t(r + \gamma \max_{a' \in A} Q_t(s', a'))$$

where $\alpha_t$is a learning rate parameter and $\gamma$ is a fixed discounting factor between 0 and 1.

6. Return to 2.

## 3. Experimental results

Figure 3 shows an example of SLAM using the proposed algorithm with the second method for global map building. Black squares describe the positions where the feature matching has passed from a local frame to an other. The ellipses around the features on the original frame represent the estimated covariance. The ellipses are drowning in straight line around non matched feature and in dots for matched features.

The simulation of the robot navigational behaviour is done with simulator MoRoS3D. A simulator as such increases safety when developing and testing algorithms. In MoRoS3D a robot can be placed in a 3D environment and interact with that environment in a manner similar to that of the robot in the real physical situation. Although MoRoS3D visualizes the entire surroundings of the robot, the robot software only "sees" the information it collects through its sensors, just like with a physical robot. The MoRoS3D simulator provides simple interaction with the user and offers different virtual cameras including on-

board and tracking ones. Simple distance sensors, such as Laser, US and IR, are simulated.

Figure 4 shows the training of the path planning system in a realistic environment and Figure 5 shows some results of the developed path planning system in some realistic cases.

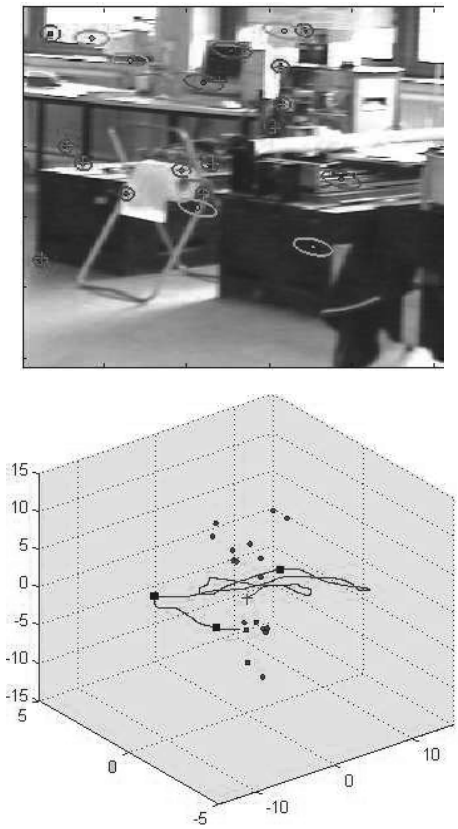


*Fig. 3. Example of Mapping and localization in a real scene.*

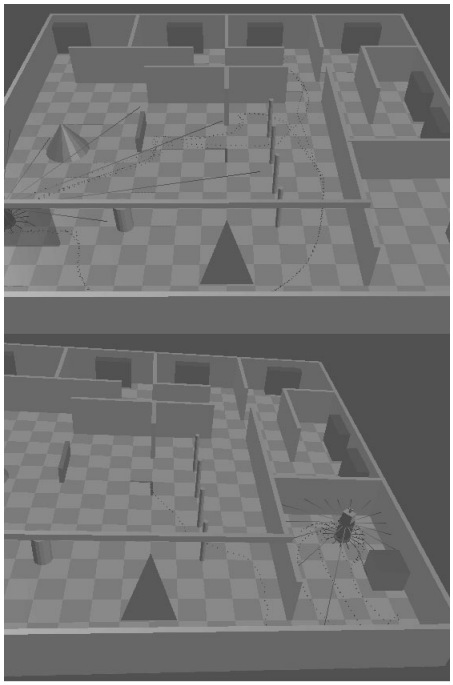

*Fig. 4. Diagram of the arrangement of the robot sensors.*

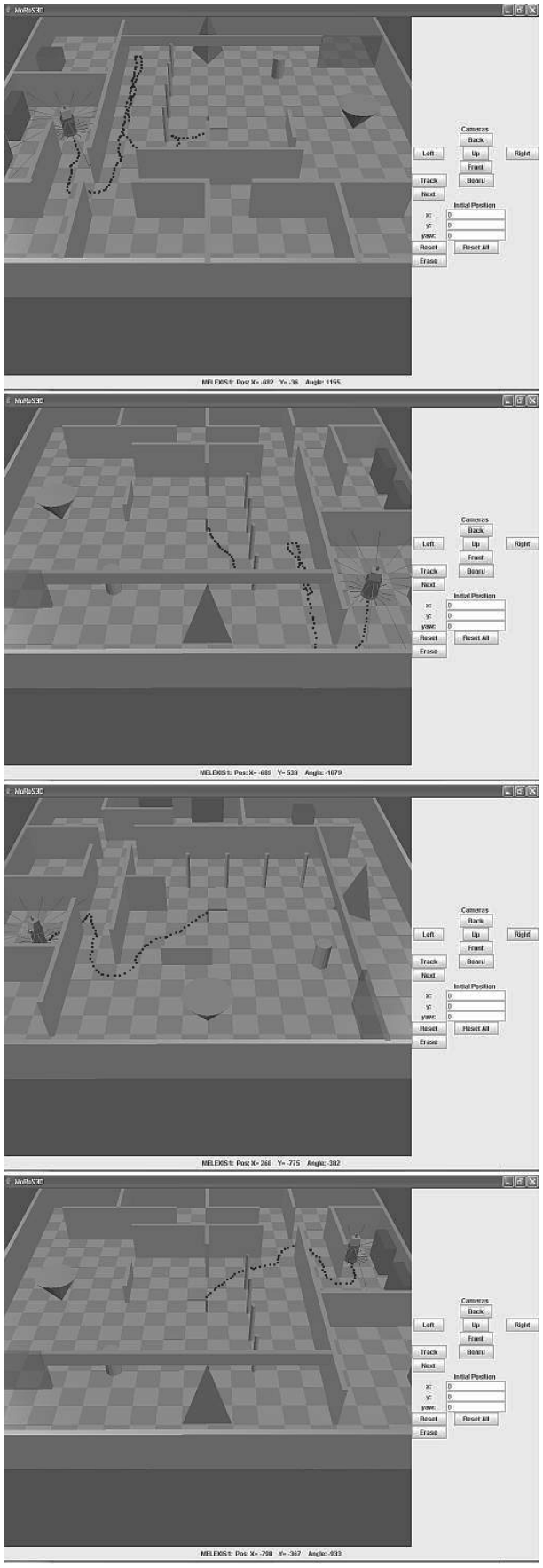


*Fig. 5. Results of the navigation strategy, presented in the MoRoS3D multi robot simulator.*

## 4. Conclusion

We presented a monocular-based EKF SLAM method. The proposed approach for SLAM is an extension to the method introduced by Davison *et al.* [1] to the cases of large environments with possible perceptual aliasing. For this, the proposed approach uses a history memory which accumulates sensory evidence over time to identify places with a stochastic model of the correlation between map features. This has allowed also building a complete description of the scene as a global map composed of several size limited local maps.

Based on the estimated map and its global position, the robot can fined a path and navigate without colliding with obstacles to reach a user defined goal using a control system based on adaptive fuzzy logic.

**AUTHORS**

**Sid Ahmed Berrabah*, Eric Colon** - Department of Mechanics, Royal Military Academy, Avenue de la Renaissance 30, 1000 Brussels, Belgium. E-mail: sidahmed.berrabah@rma.ac.be.

* Corresponding author

### References

[1] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, issue 6, June 2007, pp.10521067.

[2] S.A. Berrabah, G. De Cubber, V. Enescu, H. Sahli, "MRF-Based Foreground Detection in Image Sequences from a Moving Camera". In: *IEEE International Conference on Image Processing (ICIP 2006)*, Atlanta, GA, USA, Oct. 2006, pp.1125-1128.

[3] A. J. Davison, Y. G. Cid, N. Kita, "Real-time 3D SLAM with wide-angle vision". In: *Intelligent Autonomous Vehicles*, Lisboa-Portugal, Jul. 2004.

[4] J. Folkesson, P. Jensfelt, H. Christensen, "Graphical SLAM using vision and the measurement subspace". In: *IEEE/JRS -Intl Conf. on Intelligent Robotics and Systems (IROS)*, Edmonton-Canada, Aug 2005.

[5] F. Andrade-Cetto, A. Sanfelin, "Concurrent Map Building and Localization with landmark validation". In: *16th International Conference on Pattern Recognition ICPR/02*, 2002, vol.2.

[6] J. W. Fenwick, P. M. Newman, J. J. Leonard, "Cooperative Concurrent Mapping and Localization". In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation, May 2002, Washington, USA, pp. 1810-1817.

[7] S. Thrun, D. Fox, W. Burgard, "A probabilistic approach to concurrent Mapping and Localization for Mobile Robots", *Machine Learning*, vol.31, no. 1-3, 1998, pp. 29-53.

[8] J.D. Trados, J. Neira, P. Newman, J. Leonard, "Robust mapping and localization in indoor environments using sonar data", *International Journal of Robotics Research*, 2002, N. 21, pp. 311-330.

[9] M. W. M. Gamini Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, M. Csorba, "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem", *IEEE Transactions on Robotic and Automation*, 2001, vol. 17, no. 3, pp. 229-241.

[10] L.X. Wang, *Adaptive Fuzzy Systems and Control*, PTR Prentice Hall, 1994.

[11] A. Bonarini, "Delayed reinforcement, fuzzy Q-learning and fuzzy logic controllers". In: *Genetic Algorithms and Soft Computing*, Herrera, F. and Verdegay, J.L. (Eds.), Physica-Verlag, 1996.

[12] M.S. Kim, S.G. Hong, J.J. Lee, "Self-Learning Fuzzy Logic Controller using Q-Learning", *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol.4, no. 5, 2000, pp. 349-354.

[13] A. Gonzalez, R. Perez, *A Learning System of Fuzzy Control Rules Based on Genetic Algorithms, Genetic Algorithms and Soft Computing, Studies in Fuzziness and Soft Computing*, vol. 8, Physica-Verlag, September 1996, pp. 202-225.

[14] M. Henning, S. Vinoski, *Advanced corba programming with C++*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[15] E. Colon, H. Sahli, and Y. Baudoin, "Coroba, a multi mobile robot control and simulation framework", Special Issue on "Software Development and Integration in Robotics" *of the International Journal on Advanced Robotics*, vol. 3, 2006, no. 1, pp. 73-78.

[16] Thomas Geerinck, Eric Colon, Sid Ahmed Berrabah, and Kenny Cauwerts, "Tele-robot with shared autonomy: Distributed navigation development framework", *Integrated Computer-Aided Engineering (ICAE)*, vol. 13, no. 4, 2006, pp. 329-346.