Extended and Unscented Kalman Filters for mobile robot localization and environment reconstruction

Giuseppe Cotugno, Luigi D'Alfonso, Walter Lucia, Pietro Muraca, Paolo Pugliese

***Abstract*— In this work we compare the performance of two algorithms, respectively based on the Extended Kalman Filter and the Unscented Kalman Filter, for the mobile robot localization and environment reconstruction problem. The proposed algorithms do not require any assumption on the robot working space: they are driven only by the measurements taken using ultrasonic sensors located onboard the robot. We also devise a switching sensors activation policy, which allows energy saving still achieving accurate tracking and reliable mapping of the workspace. The results show that the two filters work comparably well, in spite of the superior theoretical properties of the Unscented Filter.**

I. INTRODUCTION

Mobile robot navigation in an unknown environment is a well-known problem that involves:

- building a (possibly partial) map of the workspace;
- localizing the robot within such map.

The two tasks are related to each other: on the one hand, to know where the robot is located, a map of the environment is needed; on the other hand, to reconstruct the environment, robot position and orientation must be estimated, because mapping is performed using such variables [1].

Many classical solutions to the navigation problem in a known environment are based on the use of the Extended Kalman Filter. When no information on the robot workspace is given, the filter must be complemented with a suitable model for the measurement devices: in the most common situation, mobile robots are equipped with onboard ultrasonic sensors, battery-powered, that measure the distance from the boundaries of the surroundings.

The problem has received relevant attention in the literature; among the others, a graph based technique has been proposed in [2], where the environment is scanned using a laser sensor and the measurements are clustered and assigned to a feature-type group (i.e. lines, points or edges). Then a graph is built from the features, that represents an observation of the environment. Other techniques for *simultaneous* localization and mapping (SLAM) can be found in [3]-[4].

In this work we assume that the robot workspace is totally unknown, and we use local information to give a good measurement model to complement the filter. The basic idea behind our algorithms is that we do not need to know the whole environment to estimate the current position and orientation of the robot, but only those portions that interact with the robot sensors. This leads to the *Neighbors Based Algorithm* (NBA) of Section III.

Another issue we have investigated is the opportunity to use a sensor switching policy. Mobile robots are equipped with several sensors, but although it is intuitive that the more of them you use, the better estimate you get, yet there are reasons for do not exceed in their use.

The most important is that sensors consume battery power, and an intensive activation reduces its autonomy. There are also situations where multiple sonar sensors cannot operate simultaneously, for example when they use the same frequency band [5]. Also bandwidth consumption and possible collisions when sending measurements from the sensors to the filter have to be considered. These considerations impose to find a trade-off between the number of sensors which are active at each instant and the accuracy of the estimate. In other words, we look for the best sequence of activation of a small fraction of the available sensors.

We have adapted the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) to include the *Neighbors Based Algorithm*, and have also included in the filters the proposed sensors switching policy. It has turned out that, in spite of the superior theoretical properties of the Unscented Filter, the Extended Filter performs comparably well on this specific problem, and the proposed rule to switch between sensors gives for both filters good results.

The paper is organized as follows: in Section II the model of the mobile robot is presented; in Section III we present our modified version of the EKF and of the UKF; in Section IV we propose the switching technique to improve the accuracy of the state estimate; in Section V we describe the algorithm used to reconstruct an environment estimation. Finally in Section VI and VII we present and discuss the results of the experiments and outline some possible extensions for future investigations.

II. ROBOT MODEL

We consider a battery-powered mobile robot with two independent driving wheels and one castor wheel, as shown in Fig. 1. As an example we refer to the Khepera III robot [6] used in our experiments and in many academic laboratories. An approximated, discrete-time model of such a robot, which neglects the dynamic of the motors and the frictions, is [7]:

$$\begin{cases} x^1_{k+1} = x^1_k + v_k\, T \cos(\theta_{k+1}) + w^x_k \\ x^2_{k+1} = x^2_k + v_k\, T \sin(\theta_{k+1}) + w^y_k \\ \theta_{k+1} = \theta_k + \Delta_k + w^\theta_k, \end{cases} \tag{1}$$

where the state of the system is:

- (x^1_k, x^2_k): the position of the robot center at time t_k,
- θ_k: the angle between the robot axle and the x-axis,

G. Cotugno is with King's College London, England. E-mail: *giuseppe.cotugno@kcl.ac.uk*

L. D'Alfonso, W.Lucia, P. Muraca, P. Pugliese are with DIMES, Università della Calabria, Rende, Italy E-mail: *ldalfonso,wlucia,muraca,pugliese@dimes.unical.it*

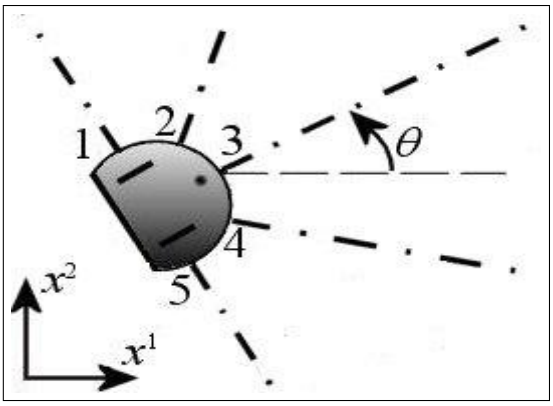


Fig. 1. Robot sketch.

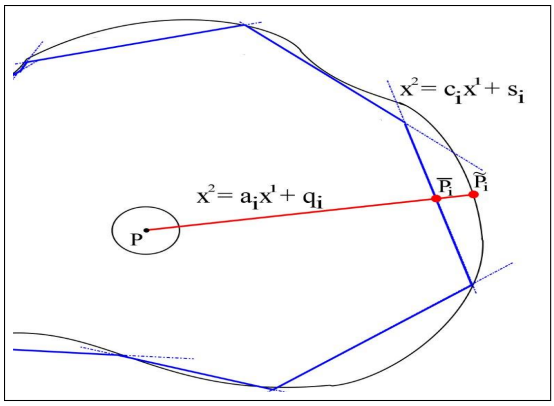


Fig. 2. Observation model.

and the other symbols denote:

- $v_k = R\,(\omega_k^l + \omega_k^r)/2$: the linear velocity of the robot,
- ω_k^l and ω_k^r: the angular velocities of wheels,
- w_k^x, w_k^y, w_k^θ: zero-mean uncorrelated Gaussian noises,
- R: the radius of the wheels,
- l: the length of the axes,
- $\Delta_k = R\,(\omega_k^l - \omega_k^r)T/l$: the rotation within $[t_{k-1}, t_k]$,
- $T = t_k - t_{k-1}$: the sampling period.

The input variables of the model are the angular velocities of the wheels, denoted by ω_k^l and ω_k^r, that have been precomputed to make the robot follow the desired trajectories. The Gaussian disturbances take into account for unmodeled dynamics, friction, wheels slipping and external disturbances.

Geometry and observation structure

We suppose that the robot is equipped with five ultrasonic sensors (this is the case of the Khepera III), located as shown in Fig. 1 and denoted by $S_i,\ i = 1, \ldots, 5$. Each sensor provides the distance between the center of the robot, denoted by $P = (x^1, x^2)$, and a point on the surrounding environment, denoted by $\tilde{P}_i = (\tilde{x}_i^1, \tilde{x}_i^2)$.

We assume that the robot is moving in an unknown workspace, that we represent by a set of segments, each of them intersecting at least one point of the boundary (see Fig. 2). The measurement provided by sensor S_i is approximated, as shown in that figure, by the distance between P and the intersection $\bar{P}_i = (\bar{x}_i^1, \bar{x}_i^2)$ between the axis of sensor S_i and one of the representative segments of the boundary.

Denoting the axis of sensor S_i by $x^2 = a_i x^1 + q_i$, and the axis of the representative segment by $x^2 = c_i x^1 + s_i$, the intersection $\bar{P}_i$ is

$$\bar{x}_i^1 = \frac{s_i - q_i}{a_i - c_i}, \qquad \bar{x}_i^2 = \frac{a_i s_i - c_i q_i}{a_i - c_i}, \tag{2}$$

therefore the distance between P and $\tilde{P}_i$ can be approximated by the distance between P and $\bar{P}_i$, which is given by

$$\eta_i = ((x^1 - \bar{x}_i^1)^2 + (x^2 - \bar{x}_i^2)^2)^{1/2}. \tag{3}$$

Now let us denote by α_i the orientation of each sensor S_i with respect to the robot axis (corresponding to the third sensor axis); the parameters of S_i axis are then given by:

$$a_i = \tan(\theta + \alpha_i), \quad q_i = x^2 - a_i x^1, \tag{4}$$

and using (2) and (4) within (3) we obtain a distance function η_i depending only on the robot state and segment (s_i, c_i)

$$\eta_i = h((x^1, x^2, \theta), (s_i, c_i)),\ i = 1, \ldots, 5.$$

These relationships allow to define the output equation of the model of the robot, and will be written in the compact form

$$y_k = h(x_k, (\bar{s}_k, \bar{c}_k)) + v_k, \tag{5}$$

where the dimension of vector y_k ranges from one to five, depending on how many sensors we use, $x_k = [x_k^1\ x_k^2\ \theta_k]'$ is the state of the robot at time k, the vector v_k collects the sensor noises, also assumed Gaussian and zero-mean, and uncorrelated with w_k; the vectors $\bar{s}_k$ and $\bar{c}_k$ contain the parameters (s, c) of the segments hit by the sensors at time k.

III. NEIGHBORS BASED EXTENDED AND UNSCENTED KALMAN FILTERS (NEKF, NUKF)

Nonlinear filtering is the problem of estimating the state of a nonlinear stochastic system from noisy measurements. For discrete-time systems the framework is given by the equations

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) + w_k \\ y_k &= h(x_k) + v_k, \end{aligned} \tag{6}$$

where x is the state to be estimated, y is the measured output, w and v are the system and measurements noises.

Beside $(\bar{s}_k, \bar{c}_k)$, model (1), (5) falls within this framework on defining

$$x_k = [x_k^1\ \ x_k^2\ \ \theta_k]',\ \ w_k = [w_k^x\ \ w_k^y\ \ w_k^\theta]',\ \ u_k = [\omega_k^l\ \ \omega_k^r]'.$$

From now on we will denote by W and V the covariance matrices of the noises w_k and v_k, respectively. These matrices will be assumed known. Moreover, we suppose we are given an initial estimate of the state $\hat{x}_{0|0}$ and its estimation error covariance matrix $P_{0|0}$.

The observation structure (5) is well-posed only if $(\bar{s}_k, \bar{c}_k)$ are known: the Neighbors Based Algorithm (NBA) has been devised to approximate them.

Neighbors Based Algorithm

The core idea behind the NBA is the *proximity* among acquired measurements; we define two points P_1, P_2 as *neighbors* if $||P_1 - P_2|| < r$, where r is a parameter of the algorithm. Moreover, for a given set of points $\mathcal{A}$ and a point $P \in \mathcal{A}$, we define the set-valued *closeness function* $\mathcal{N}$ by

$$\mathcal{B} = \mathcal{N}(P, \mathcal{A}),$$

where $\mathcal{B} = \{P_i \in \mathcal{A} : ||P_i - P|| < r\}$, i.e., the subset of points of $\mathcal{A}$ which are neighbors, in a radius r, of P.

Once a new measurement y_i has been acquired by sensor S_i, and given the actual estimate of the robot state, we can compute an approximation of the environment point $\tilde{P}_i$ hit by the sensor beam. This approximation, denoted by P_i^*, differs from the actual point because of both the estimation and the measurement errors.

The NBA then computes the closeness function of P_i^* on the set of the previously identified boundary points, denoted by $\mathcal{M}$. At this point we compute the parameters (s, c) of the Least Mean Square line approximating the points in this neighbors. This will be represented by the function

$$(s, c) = \text{LMS}(\mathcal{N}(P^*, \mathcal{M})).$$

Finally, the point $\bar{P}_i$ is computed as the intersection between the Least Mean Square line and the sensor beam. We summarize the NBA algorithm as follows:

Neighbors Based Algorithm

---

for each step k, given $\hat{x}_k^1, \hat{x}_k^2, \hat{\theta}_k, \mathcal{M}_k$, I_k, **do**

1) **for** $i \in \mathcal{I}_k$
   - acquire a measurement y_i from the sensor S_i
   - $P_{i,1}^* = \hat{x}_k^1 + y_i \cos(\hat{\theta}_k + \alpha_i)$
   - $P_{i,2}^* = \hat{x}_k^2 + y_i \sin(\hat{\theta}_k + \alpha_i)$
   - $P_i^* = (P_{i,1}^*, P_{i,2}^*)$

   end
2) $\mathcal{M}_{k+1} = \mathcal{M}_k \cup \{\cup_{i \in \mathcal{I}_k} \{P_i^*\}\}$
3) **for** $i \in \mathcal{I}_k$
   - $(\hat{s}_i, \hat{c}_i) = \text{LMS}(\mathcal{N}(P_i^*, \mathcal{M}_{k+1}))$

   end

end

- $\mathcal{I}_k$ is the set of sensor indexes to be used at step k
- $\hat{x}_k^1, \hat{x}_k^2, \hat{\theta}_k$ is the robot state estimation at step k
- $\mathcal{M}_k$ is the set of previously acquired environment points
- $(\hat{s}_i, \hat{c}_i)$ are the approximations, at step k, of the parameters of the segments intercepted by sensor S_i axis.

From now on, we will use the NBA algorithm as a function:

$$(\hat{s}_i, \hat{c}_i) = \text{NBA}(i, k),$$

where i is the sensor index and k is the time step, and we will use it to complement the EKF and UKF.

Neighbors based Extended Kalman Filter

The Extended Kalman Filter (EKF) has been used for many years to estimate the state of nonlinear systems from noisy measurements, and it has been probably the first concrete application of Kalman's work on filtering [8].

It is based on the linearization of the nonlinear maps (f, h) of (6) around the estimated trajectory, and on the assumption that the initial state and measurement noises are Gaussian and uncorrelated each other.

From the computational point of view, the EKF is simply a time-varying Kalman filter where the dynamic and output matrices are given by

$$A_k = \left.\frac{\partial f(x, u_k)}{\partial x}\right|_{x=\hat{x}_{k|k}}, \quad C_k = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}_{k|k-1}}, \quad (7)$$

and the output is a sequence of state estimates $\hat{x}_{k|k}$ and of matrices $P_{k|k}$. Including the NBA algorithm within a standard EKF [9], we have:

Neighbors based Extended Kalman Filter

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k)$$
$$P_{k+1|k} = A_k P_{k|k} A_k' + W$$
$$K_{k+1} = P_{k+1|k} C_{k+1}' (C_{k+1} P_{k+1|k} C_{k+1}' + V)^{-1}$$
$$(\hat{\bar{s}}_k, \hat{\bar{c}}_k) = \text{NBA}(\mathcal{I}_k, k)$$
$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(y_{k+1} - h(\hat{x}_{k+1|k}, (\hat{\bar{s}}_k, \hat{\bar{c}}_k)))$$
$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1} C_{k+1} P_{k+1|k}$$

There $\hat{x}_{k+1|k}$ represents the estimate of x_{k+1} *before* getting the observation y_{k+1}, $\hat{x}_{k+1|k+1}$ represents the estimate *after* getting that observation, and $(\hat{\bar{s}}_k, \hat{\bar{c}}_k)$ represent the output of NBA at time k and for each sensor $S_i, i \in \mathcal{I}_k$.

It is well known that the EKF is prone to diverge, mainly for bad initial estimates and high noises [10], but to the best of our knowledge no testable convergence conditions are known.

It is worth to remind that, differently from the linear case, the trace of $P_{k|k}$ only is an approximation (because of the linearization) of the MS estimation error $E\{\|x_k - \hat{x}_{k|k}\|^2\}$.

Neighbors based Unscented Kalman Filter

The Unscented Kalman Filter (UKF) has been developed in recent years to overcome two main problems of the EKF, i.e., the poor fitting properties of the first order approximation and the requirement for the noises to be Gaussian [11],[12].

The basic idea behind the UKF is that of finding a transformation that allows to approximate the mean and covariance of a random vector of length n when it is transformed by a nonlinear map. This is done by computing a set of $2\,n + 1$ points, called σ-points, on the basis of the mean and variance of the original vector, transforming these points by the nonlinear map and then approximating the mean and variance of the transformed vector from the transformed σ-points.

As for approximating properties of the filter, it has been shown [12] that, while the EKF estimate of the state is

accurate to the *first* order, the same UKF estimate is accurate to the *third* order in the case of Gaussian noises.
Moreover, the covariance estimate is accurate to the *first* order for the EKF, and to the *second* order for the UKF.

We report here the so-called NonAugmented version of that filter [13], which is appropriate for additive noises, and it is easy to implement. The description follows partially the one given in [14], with some modifications that make it more compact and suitable for a MATLAB [15] implementation, and also convenient to compute the trace of the covariance estimate. The values of the *weights* of that filter

$$R^m = [R_1^m \cdots R_{2n+1}^m]', \quad R^c = \mathrm{diag}\{R_1^c, \ldots, R_{2n+1}^c\}$$

and of the parameter λ are given in Section VI.

Neighbors based Unscented Kalman Filter

---

for each step k, starting from $\hat{x}_{k|k}$ and $P_{k|k}$, **do**

1) Compute $B_{k|k} = \sqrt{(n+\lambda)P_{k|k}}$, i.e., the scaled square root of matrix $P_{k|k}$
2) Compute the σ-points matrix

$$\chi_{k|k} = [\hat{x}_{k|k} \quad \hat{x}_{k|k}+B_{k|k} \quad \hat{x}_{k|k}-B_{k|k}] \in \mathbb{R}^{n\times(2n+1)}$$

There (and in the sequel) the sum of a vector plus a matrix is intended as summing the vector to all the column of the matrix (*à la* MATLAB)

3) Transform the σ-points matrix (columnwise)

$$\chi^*_{k+1|k} = f(\chi_{k|k}, u_k)$$

4) Compute the a-priori statistics

$$\hat{x}_{k+1|k} = \chi^*_{k+1|k}\, R^m$$
$$P_{k+1|k} = (\chi^*_{k+1|k} - \hat{x}_{k+1|k})R^c(\chi^*_{k+1|k} - \hat{x}_{k+1|k})' + W$$

5) Update B and compute the new σ-points

$$B_{k+1|k} = \sqrt{(n+\lambda)P_{k+1|k}}$$
$$\chi_{k+1|k} = [\hat{x}_{k+1|k} \quad \hat{x}_{k+1|k}+B_{k+1|k} \quad \hat{x}_{k+1|k}-B_{k+1|k}]$$

6) Compute NBA

$$(\hat{\bar{s}}_k, \hat{\bar{c}}_k) = \mathrm{NBA}(\mathcal{I}_k, k)$$

7) Compute the predicted output

$$\Gamma_{k+1|k} = h(\chi_{k+1|k}, (\hat{\bar{s}}_k, \hat{\bar{c}}_k))$$
$$\hat{y}_{k+1|k} = \Gamma_{k+1|k}\, R^m$$

8) Compute the Kalman gain

$$P_{yy} = (\Gamma_{k+1|k} - \hat{y}_{k+1|k})R^c(\Gamma_{k+1|k} - \hat{y}_{k+1|k})' + V$$
$$P_{xy} = (\chi_{k+1|k} - \hat{x}_{k+1|k})R^c(\Gamma_{k+1|k} - \hat{y}_{k+1|k})'$$
$$K_{k+1} = P_{xy}P_{yy}^{-1}$$

9) Compute the a-posteriori statistics

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(y_{k+1} - \hat{y}_{k+1|k})$$
$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1}P_{yy}K'_{k+1}$$

end

IV. THE SENSORS SWITCHING POLICY

Selecting at each time instant a small fraction of the available sensors to be activated is an old problem; the original motivation, three decades ago, was the limited capacity of the transmission medium and the low computational power of the processors that made the computation.

Nowadays, the above difficulties are vanished, thanks to the huge development in the transmission and computational devices; however, a new and important problem is that of saving the lifetime of the batteries that power the sensors.

For that sake, a switching policy to use a small fraction q of the p available sensors is worthwhile, although it is evident that such a policy will have an impact on the estimate.

To formalize the problem, on activating at each instant q out of p sensors, we get a q-valued output equation $h(\cdot)$ and a $q \times n$ output matrix C_k by Eq. (5) and (7). Different choices of the activation sequence return different estimates of x_k for both the EKF and the UKF, thus we are interested in finding the best, in some statistical sense, estimate.

The switching policy we propose may be described by:

> *At each instant, among the p sensors, choose q of them in such a way that the effect of the current observation on the estimate is maximized.*

In a linear estimation problem, a meaningful quality measurement of the estimate is the trace of the estimation error covariance matrix $P_{k|k}$, i.e., the MS estimation error, and a simple way to evaluate the effect of the current observation y_k is to take the trace of the difference between $P_{k|k-1}$ and $P_{k|k}$, i.e., the *a-priori* and *a-posteriori* estimation error.

This is correct under the additional assumption that all the state variables are coherent, i.e., they share the same measurement units. If that is not the case, as in our problem, a *weighted trace* has to be used, where the weights both make the sum coherent and also can be used to emphasize some eigenvalues of $P_{k|k}$ respect to the others.

Although in a nonlinear filtering problem the matrix $P_{k|k}$ only is an approximation of the estimation error covariance, its weighted trace has been chosen as our criterion, by analogy with the linear case, and because it is simple to be computed, as shown in the following. Our criterion will be

> *At each instant, among the p sensors, choose q of them in such a way that the trace of the difference between $P_{k|k-1}$ and $P_{k|k}$ is maximized.*

The approach of maximizing at each step that trace has been introduced for the first time in [16] for the state estimation of linear systems using a Kalman filter, but here it is used for the first time in conjunction with the Unscented Kalman Filter. It is worth noticing that the trace criterion becomes more meaningful when the UKF is used because the approximation of $P_{k|k}$ is correct to the second order.

Although there is no guarantee that this choice may take to the optimal switching sequence (finding it would be a problem of combinatorial complexity), we look at this criterion as a heuristic method to improve the convergence properties of the EKF and the UKF.

Computing the trace criterion for the EKF

As for the EKF, the difference $J_k = P_{k|k-1} - P_{k|k}$, whose trace has to be maximized by the choice of C_k, is simply

$$J_k = P_{k|k-1}C_k'(C_k P_{k|k-1} C_k' + V)^{-1} C_k P_{k|k-1}.$$

When a small fraction q of sensors is used, the computational cost of this formula is low, because of many common terms and symmetries, and it is suitable to be done online. The hypothesis that q is small is intrinsic in our problem (we want a small number of sensors active at each instant), hence the best trace is easily computed by enumeration of the $\binom{p}{q}$ possible values.

Computing the trace criterion for the UKF

For this filter the difference $J_k = P_{k|k-1} - P_{k|k}$ amounts to

$$J_k = K_k P_{yy} K_k'.$$

Here the computation is a little more complex; let Q_k be a $q \times p$ matrix where the j-th entry of each row is one if the j-th sensor is active, and zero otherwise, and define

$$\phi_k = (\chi_{k|k-1} - \hat{x}_{k|k-1})R^c, \quad \psi_k = Q_k(\Gamma_{k|k-1} - \hat{y}_{k|k-1}).$$

Then the value of the matrix J_k can be written as

$$J_k = \phi_k \psi_k' \, (\psi_k \, R^c \psi_k' + Q_k V Q_k')^{-1} \psi_k \phi_k',$$

and again, being R^c diagonal and Q_k sparse, and because of the symmetry, the computational cost of the trace index is low.

V. Environment Reconstruction

Environment reconstruction is obtained by clustering the set of points P_i^* generated by the algorithm, and then mapping each cluster to a graph, which is representative of a section of the workspace. Once all the graphs have been built, they are refined to get a more precise representation of the workspace boundaries by finding the shortest paths and selecting the most convenient path for approximating the environment. A sketch of the reconstruction algorithm is shown in Figure 3. Points clustering is performed using the Fuzzy C-Means algorithm (FCM) [17]. Clusters of points can be used as a rough estimate of sections of the environment. The need for clustering is to group the environment into large areas; for each area (cluster) a graph is associated. A graph is just a model of its associated area; the advantage of using graph models is the wide variety of well established algorithms that can be used for later elaboration. A graph is built as follows: each point in a cluster is a vertex of the associated graph, and an edge among two vertexes P_1^* and P_2^* there exists if and only if $\|P_1^* - P_2^*\| \leq \delta$, where δ is a small constant (in our experiments we use $\delta = 0.2$).

Once the environment reconstruction problem has been mapped into a graph theoretic problem, it is possible to apply graph theory techniques to refine the graph and approximate the environment. As graphs' structure could be very complex, featuring many edges an cycles, the core idea is to simplify the structure in order to extract a linear structure from a connected one. All the edges of such simplified structure approximate a small portion of the area modelled by the associated graph. In other words, an environment's border is approximated by an ordered set of segments, each segment is a graph's edge.

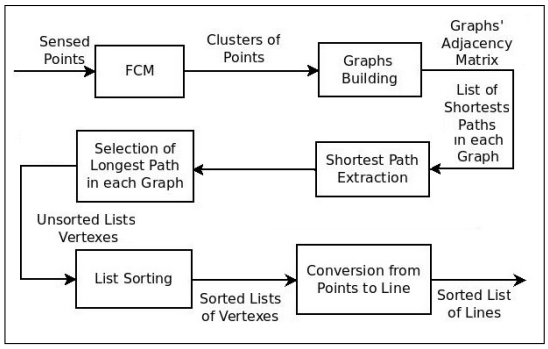


Fig. 3. A summary of the reconstruction algorithm.

In this context, the best way to highlight approximating sections of the graphs is to use Dijkstra's shortest path algorithm and then select the longest path from all the ones found by the algorithm. Dijkstra's algorithm is used to simplify the graph's structure into an acyclic linear sequence of segments; by selecting the longest path among all, most of the cluster's extension can be covered. Since the graphs are built from real measurements, there is a mapping between a real point and a vertex in the graph model; hence the correspondence between an edge and a boundary in the real world.

As this step is performed on every graph, and all the associated clusters cover the full extension of the environment, the final outcome is a collection of lists of vertexes, one for each area of the environment. The lists are then sorted, having the last vertex of a list as close as possible to the first vertex of another list. After connecting each last vertex to a first vertex, a continuous structure can be built by calculating the linear equations associated to each edge.

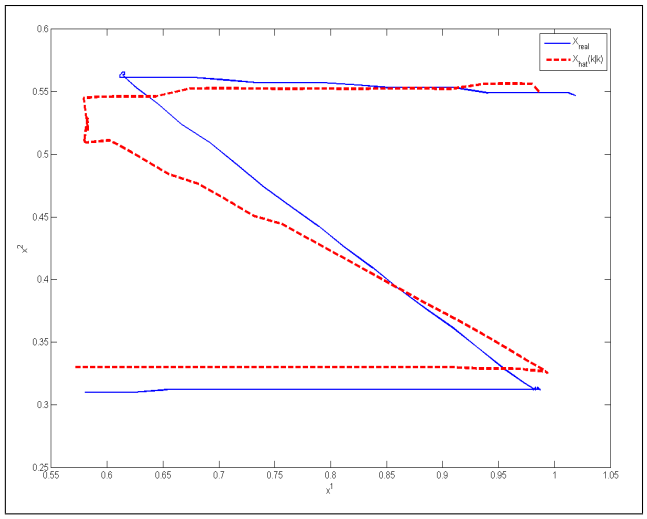
VI. Experimental Results

To evaluate the performances of the proposed filters we have performed three series of experiments, each series referring to a trajectory followed by the Khepera III robot in a rectangular workspace of 1.5×1.0 m.

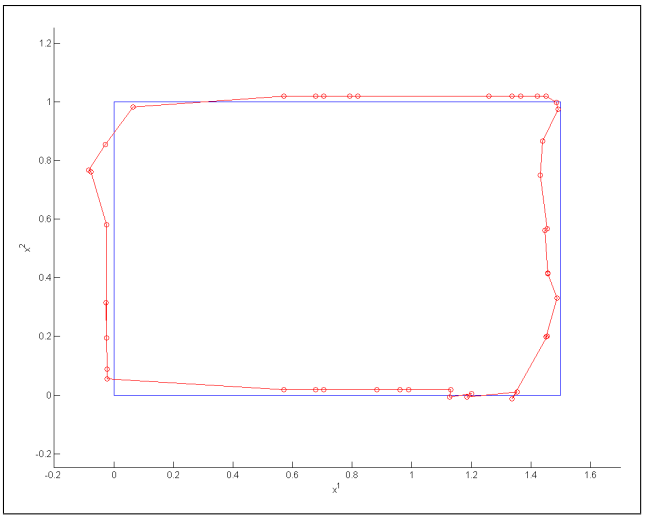
The first trajectory is an open Z-shaped path starting from the point $(0.6, 0.3)$ and passing by $(1.0, 0.3)$, $(0.6, 0.5)$, and finally $(1.0, 0.5)$. The second one is a circle of radius 0.2 m, centered in $(0.6, 0.5)$, starting from $(0.6, 0.3)$. The third trajectory is the same circle repeated three times.

The robot executed each trajectory 20 times; one run corresponds to one experiment. With a sampling time $T = 1$ sec, the first and the second trajectory have been completed in $k_f = 80$ sec, the third one in $k_f = 200$ sec. For each trajectory, a trapezoidal profile to the angular velocities of the wheels has been imposed.

In each simulation we have tested both filters using three different settings. In the first one a single sensor s_j is used,



(a) Z-shaped trajectory estimation.



(b) Workspace reconstruction.

Fig. 4. Trajectories used to validate the filters

TABLE I
NEKF - ε

Trajectory	s_1	s_2	s_3	s_4	s_5	*Switching*	*All*
zeta	0.036	0.050	0.036	0.041	0.032	0.041	0.039
circle	0.058	0.053	0.046	0.043	0.032	0.044	0.034
$3 \times circle$	0.105	0.087	0.073	0.072	0.058	0.095	0.023

TABLE II
NEKF - μ*100

Trajectory	s_1	s_2	s_3	s_4	s_5	*Switching*	*All*
zeta	1.102	1.055	1.036	1.138	1.113	0.479	0.370
circle	1.131	1.131	1.03	1.134	1.100	0.473	0.371
$3 \times circle$	0.975	0.947	0.857	0.726	0.960	0.308	0.205

TABLE III
NEKF - ρ

Trajectory	s_1	s_2	s_3	s_4	s_5	*Switching*	*All*
zeta	0.031	0.034	0.030	0.033	0.030	0.033	0.0310
circle	0.033	0.030	0.031	0.033	0.043	0.043	0.027
$3 \times circle$	0.129	0.120	0.106	0.115	0.097	0.088	0.025

TABLE IV
NUKF - ε

Trajectory	s_1	s_2	s_3	s_4	s_5	*Switching*	*All*
zeta	0.040	0.047	0.038	0.034	0.028	0.037	0.036
circle	0.065	0.056	0.057	0.049	0.036	0.033	0.033
$3 \times circle$	0.122	0.094	0.089	0.086	0.058	0.071	0.021

TABLE V
NUKF - μ*100

Trajectory	s_1	s_2	s_3	s_4	s_5	*Switching*	*All*
zeta	1.151	1.106	1.129	1.194	1.242	0.664	0.375
circle	1.182	1.145	1.059	1.146	1.126	0.668	0.376
$3 \times circle$	1.038	0.979	0.882	0.870	0.992	0.535	0.212

TABLE VI
NUKF - ρ

Trajectory	s_1	s_2	s_3	s_4	s_5	*Switching*	*All*
zeta	0.033	0.031	0.029	0.029	0.028	0.032	0.030
circle	0.036	0.034	0.037	0.035	0.051	0.036	0.026
$3 \times circle$	0.133	0.138	0.130	0.120	0.096	0.073	0.023

keeping it fixed along the path, i.e. $I_k = \{j\}, k = 0, \ldots, k_f$; this has been repeated for all five sensors. In the second setting we use a single sensor ($q = 1$), switching between the five sensors available with the policy discussed in Section IV. Finally, in the third setting we use all five sensors together to provide a base for comparison.

The following values for the covariance parameters and the initial conditions were common to all simulations:

- $W = 10^{-4}\text{diag}\{4, 4, 3.046\}$: a standard deviation of 0.02 m on x^1 and x^2 and of 1° on θ.
- $V = 9 \times 10^{-4} I$: a standard deviation of 0.03 m for each sensor, supposed sharing the same statistical properties.
- $\hat{x}_{0|0} = [0.57, 0.33, 0]'$; the true initial state of the robot

is $x_0 = [0.6, 0.3, 0]'$ for each trajectory.

- $P_{0|0} = 10^{-2}\text{diag}\{0.25, 0.25, 0.27\}$: a standard deviation of 0.05 m on robot position and 3° on orientation.

We adopt as UKF weights the values $\alpha = 0.001$, $\beta = 2$, $\kappa = 3 - n$ and $\lambda = \alpha^2(n + \kappa) - n$. According to [11], the choice $\beta = 2$ minimizes the error in the fourth-order moment of the a-posteriori covariance when the noises are Gaussian. Furthermore:

$$R_1^m = \frac{\lambda}{n+\lambda},\ R_1^c = \frac{\lambda}{n+\lambda} + 1 + \beta - \alpha^2$$

$$R_j^m = R_j^c = \frac{\lambda}{2\,(n+\lambda)},\ j = 2, \ldots, 2n+1.$$

The closeness function radius has been set to $r = 0.1$ m. As for the initialization of the set $\mathcal{M}$, we allow 15 initial steps at the beginning of each experiment, during which we only acquire measurements and points of the environment; these points will form the initial condition for $\mathcal{M}$.

To evaluate the performance of the filters we define:

$$\varepsilon = \frac{1}{k_f+1}\sum_{k=0}^{k_f}\left((x_k^1 - \hat{x}_{k|k}^1)^2 + (x_k^2 - \hat{x}_{k|k}^2)^2\right)^{1/2},$$

$$\mu = \frac{1}{k_f+1}\sum_{k=0}^{k_f}\text{tr}\{P_{k|k}\},$$

where the first index is a measurement of the localization error, and the second is a measurement of the quality of the estimate. We have also defined a threshold $\varepsilon_d = 0.15$ m on the obtained estimates, to identify divergences of the filters: if the ε index of an experiment exceeds the threshold that experiment is discarded. To evaluate the quality of the environment reconstruction a further index ρ has been defined as follows. The edge E_i between two vertices v_i and v_{i+1} in the graph represents the segment between the two vertices. We consider the points $P_{ij}, j = 1, \ldots, n_i$, equally spaced by an amount equal to 0.01 m along the segment. For each of these points, the distance d_{ij} between it and the nearest portion of the real environment is computed, and the following index ρ has been defined

$$\rho = \frac{1}{n_e}\sum_{i=1}^{n_e}\left(\frac{1}{n_i}\sum_{j=1}^{n_i} d_{ij}\right),$$

where n_e is the number of edges of the graph.

The clustering procedure requires the number of clusters used in the FCM step to be given: this has been set to $c = 5$ at the beginning of the computation, and to $c = 25$ when the computation reaches the steady state. A typical result of the algorithm can be viewed in Fig. 4(b) for the NUKF filter, while Fig. 4(a) reports the corresponding estimated trajectory. We report in Tables I, II and III the values of the indexes ε, μ and ρ, for the NEKF and all trajectories; in tables IV, V and VI the values of the same indexes, there using NUKF. All the values are averaged over the number of experiments that do not exceed the divergence threshold (they were very few). In all the tables:

- s_j refers to the cases where the j-th sensor is kept fixed along all the path
- *Switching* refers to the proposed switching policy
- *All* refers to the case where all sensors are used together
- *zeta*, *circle*, $3 \times circle$ refer to the first, the second and the third trajectory respectively.

VII. Conclusions and Future Works

The experiments performed have shown that the two filters work comparably well in giving both a reliable estimate of the robot pose and an accurate reconstruction of the working environment. In spite of the better theoretical approximating properties of the UKF over the EKF, the NUKF is only slightly better on the ε and ρ indices, while the NEKF is marginally better on the μ.

The surprising equivalence of the two filters is probably due to the fact that the nonlinearities in the model are not bad enough to highlight any substantial difference.

For both filters, the proposed switching rule always gives a value of the μ index which is much better than the one given by any fixed sensor, and close to the reference one obtained using all sensors together.

On considering workspace reconstruction, Tables III and VI show that, especially when robot performs long paths (i.e., $3 \times circle$), relying on one switching sensor is more effective than relying on one fixed sensor, whichever it is. Experiments with more complex workspaces are in progress.

Future research will investigate the effect of loss of information on the transmission channel from the sensors to the filters. Furthermore, at the moment the world reconstruction algorithm runs offline, because it is computationally expensive. We are working on a much faster version to be used simultaneously with the neighbors based filters.

References

[1] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, MIT Press, 2005

[2] T. Bailey, E. Nebot, Localisation in large-scale environment, *Robot. Auton. Syst.*, 37:261–281, 2001

[3] J.J. Leonard, H.F. Durrant-Whyte, Mobile robot localization by tracking geometric beacons, *IEEE Trans. Robot. Autom.*, 7(3):376–382, 1991

[4] H. Durrant-Whyte, T. Bailey, Simultaneous localisation and mapping (SLAM): part I – The essential algorithms, *IEEE Robot. Autom. Mag.*, 13(2):99–110, 2006

[5] V. Gupta, T. Chung, B. Hassibi, R.M. Murray, On a stochastic sensor selection algorithm with applications in sensor scheduling and dynamic sensor coverage, *Automatica*, 42(2):251–260, 2006

[6] K-TEAM Corporation [Online]. Available: http://www.k-team.com

[7] E. Ivanjko, I. Petrović, Extended Kalman filter based mobile robot pose tracking using occupancy grid maps, *Proc. of the IEEE MELECON*, Dubrovnik, Croatia, May 2004, pp. 311–314

[8] M.S. Grewal, A.P. Andrews, *Kalman Filtering: Theory And Practice Using MATLAB*, 3rd ed., Wiley, 2008

[9] B.D.O. Anderson, J.B. Moore, *Optimal Filtering*, Prentice-Hall, 1979

[10] K. Reif, S. Günther, E. Yaz, R. Unbehauen, Stochastic stability of the discrete-time Extended Kalman Filter, *IEEE Trans. Autom. Control*, 44(4):714–729, 1999

[11] S.J. Julier, J.K. Uhlmann, Unscented filtering and nonlinear estimation, *Proc. of the IEEE*, 92(3):401–422, 2004

[12] E. Wan, R. van der Merwe, The Unscented Kalman Filter, in *Kalman Filtering and Neural Networks*, S. Haykin, Ed., Wiley, 2001

[13] Y. Wu, D. Hu, M. Wu, X. Hu, Unscented Kalman filtering for additive noise case: augmented versus nonaugmented, *IEEE Signal Process. Lett.*, 12(5):357–360, 2005

[14] Yi Cao, Learning the Unscented Kalman Filter [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/18217

[15] The Mathworks, Inc., *MATLAB User's Guide*, Natick, MA, 1996

[16] L. Carotenuto, P. Muraca, G. Raiconi, On the optimal design of the output transformation for discrete-time linear systems, *J. Optim. Theory Appl.*, 68(1):1–18, 1991

[17] J.C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, *J. Cybernet.*, 3(3):32–57, 1973

[18] E.W. Dijkstra, A note on two problem in connexion with graphs, *Numer. Math.*, 1:269–271, 1959