Perceptual navigation around a sensori-motor trajectory

Cédric Pradalier and Pierre Bessière
Inria\textsuperscript{a} Rhône-Alpes & Gravir\textsuperscript{b}-CNRS\textsuperscript{c}
655 av. de l’Europe, Montbonnot, 38334 St Ismier Cedex, France
Tel. +33 476 61 54 36 — Fax. +33 476 61 52 10
cedric.pradalier@inrialpes.fr
http://www.inrialpes.fr/sharp

March 8, 2004
Published Version

Abstract — Autonomous navigation of a mobile robot along a predefined trajectory is a widely studied problem in the robotics community. We propose a Bayesian architecture that aims at being able to replay any sensori-motor trajectory – trajectory defined as a sequence of perceptions and actions – as long as the robot starts in its neighbourhood. In order to increase robustness, we also use this Bayesian framework to estimate system self-confidence while the robot is moving. This work has been validated both on a simulated robot and on a real robot: the CyCab.

Keywords — Mobile robot navigation, Sensor-based control, Bayesian programming.

\textsuperscript{a}Institut National de Recherche en Informatique et en Automatique.
\textsuperscript{b}Lab. Graphisme, Vision et Robotique.
\textsuperscript{c}Centre National de la Recherche Scientifique.
Perceptual navigation around a sensori-motor trajectory
Cédric Pradalier and Pierre Bessière
GRAVIR – CNRS – INRIA Rhône-Alpes – INPG, Grenoble - FRANCE

Abstract—Autonomous navigation of a mobile robot along a predefined trajectory is a widely studied problem in the robotics community. We propose a Bayesian architecture that aims at being able to replay any sensori-motor trajectory – trajectory defined as a sequence of perceptions and actions – as long as the robot starts in its neighbourhood. In order to increase robustness, we also use this Bayesian framework to estimate system self-confidence while the robot is moving. This work has been validated both on a simulated robot and on a real robot: the CyCab.

Index Terms—Mobile robot navigation, Sensor-based control, Bayesian programming.

I. INTRODUCTION

Autonomous navigation of a mobile robot has been a widely studied problem in the robotic community. Most robot designed for this task are equipped with onboard sensor(s) to perceive external world (sonars, laser telemeters, camera). Then, two main kinds of approach to autonomous navigation have been proposed: reactive navigation where the robot uses only current perception to move and explore without colliding ([1], [3]) and servo-ed navigation where the robot is given a pre-planned reference trajectory and use some closed-loop control law to follow it [13], [12]. Among servo-ed navigation, two classes of approaches can again be separated: state space tracking[8], [7] and perception space tracking[16], [4].

State space tracking implies two specificities: first to be given a reference trajectory in the state space, and second, to be able to localise the robot, also in the state space. Conversely, perception space tracking implies that trajectory is defined with respect to perception only, hence avoiding the need for global localisation. A specific application of perception tracking is visual servoing, classically implemented as the convergence of observed image to a fixed reference image.

In this paper, we are specifically interested in the case of perceptual tracking of a perceptual trajectory with a mobile robot. We assume that: i) reference trajectory is defined as a sequence of observations perceived by an onboard sensor along robot movement; ii) no localisation system (neither GPS nor landmark based) is available to perform tracking. This situation is interesting for at least three reasons: first, since trajectory is not defined with respect to a Cartesian frame we don’t need to deal with the complex task of global localisation, second this kind of trajectory can be naturally and easily learned from examples, and third, it can be seen has an hypothesis on how biologic entities memorise and represent paths.

This paper will be organised as follows: after some preliminary definitions and problem specification in sec. II and III, we deal separately with initialisation, tracking and failure diagnosis in sec. IV to VI. Finally sec. VII presents implementation details and results.

II. DEFINITIONS AND HYPOTHESIS

Let us start by introducing some useful notations: our robot is described by a configuration $C$ and commanded with a command $U$, our sensor gives observations $O$ and a difference of configuration is noted $\xi$. $O(t)$ is the set of past observations at time $t$: $\{O(u) \mid u \leq t\}$. Finally, previous instant $t - \Delta t$ with respect to instant $t$ will be noted $t'$.

We assume to be given a sensor model $H$ and a kinematic model $K$. $H$ predicts observation $\hat{O}$ given a reference observation $O_{ref}$ and a difference of view point $\xi$: $\hat{O} = H(O_{ref}, \xi)$. $K$ is defined with $C(t) = K(C(t'), U(t'), \Delta t)$.

Finally, we write $G(\mu, \sigma)$ a Gaussian distribution centred on $\mu$ with covariance $\sigma$.

III. PROBLEM SPECIFICATION

A. Sensory-motor trajectories

Definition 1: We define a sensori-motor trajectory as a function of time with values in the Cartesian product of the robot command space $U$ and its observation space $O$. Formally:

$$T_{sm} : [0, t_1] \rightarrow O \times U$$

In the following, we will note $T_{sm}(t) = [T_{sm}(t), O, T_{sm}(t), U]$.

B. Objectives

In this paper, we want to be able to replay any sensori-motor trajectory, as long as the robot starts in its neighbourhood. Our understanding of replay will be explicated below. We divided our objective in three tasks:

- **Initialisation:** Given a sensori-motor trajectory $T_{sm}$, and given an observation $O \in O$, what is the robot temporal position $\tau(0)$? In other words, we want to find $\tau(0) \in [0, t_1]$ for which $O$ is closest from $T_{sm}(\tau(0))$. Note that this does not guarantee that $O$ is an observation on $T_{sm}$.
- **Tracking:** Given $T_{sm}$, $O$ and an estimate $\tau(t)$ of the robot temporal position at time $t$, we look for commands which lead to a replay of $T_{sm}$. This
implies that we also have to track τ during trajectory replay.

- **Self-diagnosis:** Given $T_{sm}$, $O$ and $\tau(t)$, we want the system to find out whether the hypothesis on which current behaviour is built are still valid. This means that the system should track an estimate of its self-confidence over time. Special care will be taken when defining self-confidence.

## C. Trajectory replay

In the context of control theory, tracking a trajectory can be grossly defined as guaranteeing that at any time $t$, robot configuration $C(t)$ is converging to the nominal configuration $C_{nom}(t)$, defined with $T_{sm}(t)$. Nevertheless, if we want to replay our trajectory while using obstacle avoidance, it may occurs that security constraints stop the robot for a while, or even introduce a momentary diversion from the nominal trajectory. So, in our case, as we rather want to navigate around nominal trajectory, we use a more flexible definition of trajectory tracking, i.e. guaranteeing that it exists a monotonic function $\tau$, with $\partial \tau / \partial t \in [0, 1]$, such that, at any time $t$, robot configuration $C(t)$ is as close as possible to $C_{nom}(\tau(t))$.

## D. Bayesian Software Architecture

Since we are convinced[14], [2]) that probabilities and Bayesian inference are perfect tools to cope with incompleteness and uncertainties inherent to the navigation of a mobile robot in a real environment, we want to achieve sensori-motor trajectory replay using a Bayesian software architecture.

Following sections will show how a set of distinct Bayesian modules can be integrated in a modular Bayesian architecture in order to fulfil robustly objectives presented in this section.

## IV. Initialisation

In this section, we present how system initialisation is performed. We assume that a sensori-motor trajectory $T_{sm}$ is known and that an observation $O_0$ has been done. We will use the Bayesian programming framework to compute a probability distribution over possible temporal position in the trajectory. Fig. 1 formalises the Bayesian program for initial localisation (see [14], [2] for details about this formalism).

### A. Expression of $P(O \mid T \ T_{sm})$

$$P(O \mid T \ T_{sm})$$ should express which observations are expected around $T_{sm}(t)$. To this end, let us first define a probabilistic sensor model. Given a reference observation and a difference of viewpoint $\xi$, a sensor model expresses the probability of expected observation:

$$P_m(O \mid O_{ref} \xi) = \mathcal{G}(H(O_{ref}, \xi), \Sigma_m)$$

Using this model, we can define:

$$P(O \mid T \ T_{sm}) = \int_\xi P_m(O \mid O_{ref} = T_{sm}(t), O \xi) P(\xi) d\xi$$

Fig. 1. Bayesian program for initial localisation

To express the fact that we are interested by what is expected around the nominal trajectory, $P(\xi)$ is defined as zero-centred Gaussian, with a covariance formalising the “around $T_{sm}$” notion.

### B. Initialisation with $P(T \mid O \ T_{sm})$

Using Bayesian program shown in fig. 1, we can compute a numerical approximation of $P(T \mid O \ T_{sm})$, denoted as $\hat{P}$ in the following. Fig. 2 gives a typical result of this distribution.

Fig. 2. Example of initial $P(T \mid O \ T_{sm})$: Case of a planar robot observing landmarks (black squares) with a laser telemeter.

From this distribution, we have to extract a single value $t$ which will be our initial estimate $\tau(0)$. Depending on the shape of the distribution, the difficulty of this extraction can range from easy to hard or even impossible.

### C. Discussion

One point should be noted here. Even, when $\hat{P}$ is strongly peaked, there is no guarantee that we are indeed observing some part of $T_{sm}$. Nevertheless, without other information, we believe that we have to start moving as if we were confident with respect to our first estimation, while keeping in mind that future observations may infirm this estimation. This will be further discussed in section VI.

## V. Tracking

Once we know where the system start in the sensori-motor trajectory, we can start replaying it. Two variables
have to be tracked: temporal position and the error with respect to the nominal trajectory. From these variables, robot controls can be computed to track the trajectory.

A. Error tracking

Using current estimate \( \tau(t) \) of the temporal position, the sensor model defined in IV-A \( (P_m(O \mid O_{ref}\xi)) \) and Bayes rule, we can infer a distribution over difference of viewpoint \( \xi \) given current observation. To track this variable, we want to use a Bayesian filter:

\[
P(\xi(t) \mid O(t), T_{sm}(\tau(t)), O) \propto P_m(O(t) \mid \xi(t) T_{sm}(\tau(t)), O) \times 
\int_{\xi(t')} P(\xi(t) \mid \xi(t')) T_{sm}(t'), U) P(\xi(t') \mid O(t'))
\]

To this end, we define \( P(\xi(t) \mid \xi(t') T_{sm}) \) using the kinematic model \( K \) of our robot. Since we are only interested in the relative change of \( \xi \), we can express its variation as follows:

\[
\xi(t) = K(\xi(t'), U(t'), \Delta t) - K(0, U_{ref}(t'), \Delta t)
\]

And finally, \( P(\xi(t) \mid \xi(t') T_{sm}) \) will be defined as a Gaussian, centred on \( \xi(t) \), with a covariance matrix expressing our confidence on this model.

B. Temporal position tracking

Estimation of temporal position can be seen as another localisation process. Again we are tempted to use a Bayesian filter for this estimation:

\[
P(\tau(t) \mid O(t), T_{sm}) \propto P(O \mid \tau(t) T_{sm}) \times 
\int_{\tau(t')} P(\tau(t) \mid \tau(t')) P(\tau(t') \mid O(t'))
\]

where eq. 2 defines \( P(O \mid \tau(t) T_{sm}) \), and transition model \( P(\tau(t) \mid \tau(t')) \) is defined as follows:

\[
P(\tau(t) \mid \tau(t') \Delta t) = G(\tau(t') + \Delta t, \sigma_{\tau})
\]

This method works, but its computational complexity is prohibitive: we have to integrate \( P_m \) over the configuration space for every value of \( \tau \).

Heuristic: Due to the computational cost of direct application of the Bayesian filter, we chose to use a heuristic to keep track of temporal position: we assume that executed trajectory keep pace with respect to reference trajectory (i.e. \( \tau(t) = \tau(t') + \Delta t \) while \( |\xi(t)| \leq \xi_{lim} \). When error \( \xi(t) \) goes over a threshold, we keep \( \tau \) constant (i.e. \( \tau(t) = \tau(t') \)).

C. Trajectory tracking

Since we are replaying a sensori-motor trajectory, we can use temporal position estimate \( \tau(t) \) to extract reference observation \( O_{ref}(t) = T_{sm}(\tau(t)), O \) and reference controls \( U_{ref}(t) = T_{sm}(\tau(t)), U \). Then, from reference observation, we can compute error from current robot configuration to reference one. Finally, with reference controls and configuration error, we could apply a well-tuned control law given by control theory; and no doubt that robot would replay accurately its sensori-motor trajectory.

As we wanted to design a fully Bayesian application, we used inspiration from fuzzy logic control[10], [6] to build a probabilistic control law, expressed as a Bayesian data fusion problem: \( P(U \mid \xi U_{ref}) \) being expressed with \( P(U \mid U_{ref}) \) and \( P(U \mid \xi) \). Due to space limitation, this will not be developed in this paper. We refer interested reader to [18] for details about this implementation.

VI. SELF-DIAGNOSIS

As shown in section IV, our trajectory tracking system is based on a first estimation of initial temporal position. Due to such phenomena as perceptual aliasing, system cannot guarantee that this initial estimation is correct. In any case, the robot has to start moving, gathering information while following its sensori-motor trajectory. We use information gained on this movement to track a variable that expresses confidence of the system with respect to its previous assumptions and its current localisation.

Formally, variable \( \text{Conf}(t) \in \{0, 1\} \) will represent system self-confidence at time \( t \): When \( \text{Conf}(t) = 1 \) system is fully confident in its localisation (this should mean that it collected many evidences); conversely, \( \text{Conf}(t) = 0 \) express quasi-certainty that some failure has occurred: wrong initial localisation, environment change...

In many problem where state estimation is involved, model comparison is used to diagnose system state (see [17], [15] for instances). In the remaining of this section, we will show how model comparison can be used to track system self-confidence.

A. Model comparison

1) Principle: Let us assume that we work with a variable \( A \) that can be evaluated with two distinct models. We can build the following joint distribution:

\[
P(A \text{ Model} = P(\text{Model})P(A \mid \text{Model})
\]

\( P(\text{Model}) \) express our prior on which model is the best, and \( P(A \mid \text{Model}) \) evaluates probability distribution on \( A \) knowing which model is used. Then, we can use Bayes rule to compute \( P(\text{Model} \mid [A = a]) \), i.e. given a real observation \( a \) of \( A \), what is the model that best explains \( A = a \)?

2) Application: Self-confidence, as defined above, can be used as a switch between two models: one that expresses which observation can be expected given full confidence in temporal localisation, and one that expresses full distrust case.

Formally, we define \( P(O \mid \text{Conf}(t)) \). If \( \text{Conf}(t) = 0 \), since the system does know that it knows nothing on its localisation, \( P(O \mid [\text{Conf}(t) = 0]) \) is a “minimal knowledge” uniform distribution, otherwise, \( P(O \mid \text{Conf}(t) = 1) \) equals \( P(O \mid T_{sm}) \) as defined in eq. 2.

3) Tracking: Like previous trackings in this paper, self-confidence tracking will be implemented with a Markovian Bayesian filter. \( P(O \mid \text{Conf}(t)) \), as defined above, gives us an observation model, so we just have to define a transition model \( P(\text{Conf}(t) \mid \text{Conf}(t')) \) to obtain:

\[
P(\text{Conf}(t) \mid O) \propto P(O \mid \text{Conf}(t)) \times 
\sum_{\text{Conf}(t')} P(\text{Conf}(t) \mid \text{Conf}(t')) P(\text{Conf}(t'))
\]
Since confidence change without perceiving new information.

Increase when the system is able to predict a challenging and then design a probabilistic model which implement this behaviour. Mode using Bayes rule.

If \( P(\text{Conf}(t) \mid \text{Conf}(t') \text{,} Q_1 \geq 1) \), current observation was really hard to predict knowing only previous observation. So the bigger \( Q_2 - Q_1 \), the more important should be confidence increase.

To implement this behaviour, we use the Bayesian program defined in figure 3. This program is based on a definition of \( P(\text{Conf}(t) \mid \text{Conf}(t') \text{,} Q_1 \geq 1 \text{,} Q_2) \) similar to equation 7 except that doubling rate \( \delta \) and trusting rate \( \lambda \) are now functions of \( Q_1 \) and \( Q_2 \):

\[
\delta(Q_1, Q_2) = \begin{cases} 
0 & \text{if } Q_1 \geq 1 \\
\min(1, k_\delta(1 - Q_2)) & \text{otherwise} 
\end{cases} \tag{8}
\]

\[
\lambda(Q_1, Q_2) = \begin{cases} 
0 & \text{if } Q_1 \geq Q_2 \\
\min(1, k_\lambda(Q_2 - Q_1)) & \text{otherwise} 
\end{cases} \tag{9}
\]

Our experimental platform is a robotic golf-cab called CyCab, prototyped at INRIA and commercialised by French company RoboSoft. Its main specificity resides in its ability to steer both axles with different angles. This

---

### Table 1: Confidence Evolution Model

<table>
<thead>
<tr>
<th>( \text{Conf}(t) )</th>
<th>( \text{Conf}(t') )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( 1 - \lambda )</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( 1 - \lambda )</td>
<td>1 - 2( \delta )</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( P(\text{Conf}(t) \mid \text{Conf}(t')) \)

---

### B. Using innovation

Self-confidence tracking implemented as described above works fine, except an unwanted behaviour that appears when robot does not move: if current observation is well supported by the confident model, self-confidence rapidly converge to 1. This behaviour is not satisfying since confidence change without perceiving new information.

Since we are convinced that self-confidence should only increase when the system is able to predict a challenging observation, we designed a slightly different model, able to implement desired behaviour.

1) Observation model: Instead of using confidence as model switch, we introduce a new switch variable Mode \( \in \{0, 1, 2\} \). From this variable, we build an observation model \( P(O \mid \text{Mode}) \) such that:

- \( P(O(t) \mid \text{Mode} = 0) \) is a uniform distribution (equiv. to \( P(O \mid \text{Conf}(t) = 0) \) in previous model).
- \( P(O(t) \mid \text{Mode} = 1) = P(O(t) \mid O(t') U(t')) \) expresses expected observation knowing only last observation and displacement.
- \( P(O(t) \mid \text{Mode} = 2) = P_m(O(t) \mid T_{sm}(\tau(t))) \) expresses expected observation knowing maximum information: sensori-motor trajectory and temporal position.

From this model and a uniform prior \( P(\text{Mode}) \) we define a joint distribution: \( P(O(t) \text{, Mode}) = P(\text{Mode})P(O(t) \mid \text{Mode}) \) from which we can compute \( P(\text{Mode} \mid O(t)) \) using Bayes rule.

2) Confidence evolution model: For \( i \in \{0, 1, 2\} \), let us call \( M_i = P(\text{Mode} = i) \mid O(t(t)) \), and \( Q_i = \frac{M_i}{M_0} \). Knowing \( Q_1 \) and \( Q_2 \) we can predict how confidence should evolve, and then design a probabilistic model which implement this behaviour:

- If \( Q_2 < 1 \), observation prediction knowing sensori-motor trajectory and temporal position is worse than without prior knowledge. As in section VI-A, confidence should decrease in this case: the smaller \( Q_2 \) compared to 1, the bigger the decrease.
- If \( Q_1 \geq Q_2 \geq 1 \), observation prediction is better knowing only last observation than with maximum knowledge. This means that current observation does not reflect any innovation with respect to previous one. As there is no reason to change confidence without new evidences or counter-evidences, confidence should stay constant.

- If \( Q_2 > Q_1 \geq 1 \), current observation is not only better predicted by maximum knowledge observation model, but it also contains innovation w.r.t. previous observation. This ability to predict innovative observation should increase self-confidence. Furthermore, a bigger difference between \( Q_2 \) and \( Q_1 \) means that current observation was really hard to predict knowing only previous observation. So the bigger \( Q_2 - Q_1 \), the more important should be confidence increase.

---

### VII. IMPLEMENTATION

#### A. Experimental platform

Our experimental platform is a robotic golf-cab called CyCab, prototyped at INRIA and commercialised by French company RoboSoft. Its main specificity resides in its ability to steer both axles with different angles. This

---

### Fig. 3. Bayesian program for self confidence tracking

### Program

1. Relevant Variables:
   - \( \text{Conf}(t) \) : Confidence at time \( t \)
   - \( \text{Conf}(t') \) : Confidence at time \( t' \)
   - \( Q_1, Q_2 \) : Observations, as def. in VI-B.2

2. Decomposition:
   - \( P(\text{Conf}(t) \mid \text{Conf}(t') Q_1 Q_2) = P(Q_1 Q_2) P(\text{Conf}(t') \mid Q_1 Q_2) \)
   - \( P(\text{Conf}(t) \mid \text{Conf}(t') Q_1 Q_2) = P(Q_1 Q_2) P(\text{Conf}(t') \mid Q_1 Q_2) \)
   - See sec. VI-B.2

3. Parametric Forms:
   - \( P(Q_1 Q_2) : \text{Undefined} \)
   - \( P(\text{Conf}(t')) : \text{From previous iteration} \)
   - \( P(\text{Conf}(t) \mid \text{Conf}(t') Q_1 Q_2) : \)

4. Identification: \( k_\delta \) and \( k_\lambda \) are adjusted empirically.

### Question:
- \( P(\text{Conf}(t) \mid Q_1 Q_2) \)

---

### Fig. 4. The CyCab robot
ability will not be fully used in this paper: we will steer both axles, but with equal angles.

For exteroceptive perception, our mini-car is equipped with a Sick laser range finder with an efficient range of 30 meters and an uncertainty of about 5 centimetres. Cylinders covered with reflector sheets act as landmarks in our environment (see fig. 4). We developed a preprocessing algorithm which detects out landmarks in the laser scans. By this way, we can consider that our robot is equipped with a landmark detector, and forget about the complete laser scan for the remaining of this paper. It should be noted that there is no distinctive marks on our landmarks, so they all return same sensor output.

In the context of this robot, a configuration is the position of the middle of its rear axle $(x, y)$ and its orientation $\theta$. An observation is a set of $n \in \mathbb{N}$ positions $(O_1, x_1, y_1, \theta_1)$ of observed landmark in sensor frame (note that $n$ is not constant). In this case, $H$ is a composition of translation and rotation ($R_{-\Delta \theta}$):

$$H[(O_1, \ldots, O_n), \xi] = [h(O_1, \xi), \ldots, h(O_n, \xi)]$$

$$h(O_i, [\Delta x, \Delta y, \Delta \theta]) = R_{-\Delta \theta}(O_i - [\Delta x, \Delta y]^T)$$

B. Observation model with implicit matching

Using our landmark detector, definition of $P_m(O \mid O_{ref})$ is not as simple as briefly described in eq. 1. Since our landmarks are indistinguishable, an observation of $O_{ref}$ can be any permutation of $H(O_{ref}, \xi)$. Furthermore, due to limited field of view of the sensor and eventually to moving objects, some reference landmarks may not appear among observed landmarks. Finally, due to incomplete modelling and also to moving objects, some observed landmarks may not be observation of reference landmarks.

In order to manage all these events, we introduce two new variables for each observation $O_i$: $F_i$, boolean variable which indicates whether $O_i$ is a False-positive and $W_i \in [1 \ldots n]$, which indicates Which landmark was detected as $O_i$. Then we can refine $P_m(O \mid O_{ref})$ using following equations:

$$P_m'(O_i \mid O_{ref} \xi W_i F_i) =$$

$$\begin{cases} 
\text{Uniform if } F_i \text{ is true} \\
G(h(O_{ref}^{W_i}, \xi), \Sigma) \text{otherwise}
\end{cases}$$

(10)

$$P_m'(O \mid O_{ref} \xi) = \prod_{i=1}^{n} P_m'(O_i \mid O_{ref} \xi)$$

(11)

$$= \prod_{i=1}^{n} \left[ P(F_i) \sum_{W_i} P(W_i) P_m'(O_i \mid O_{ref} \xi W_i F_i) \right]$$

Even if computational cost of our sensor model is rather heavy, we chose this implementation for its general robustness, and specially for its low sensibility to landmarks occlusion.

C. Obstacle avoidance

Since our trajectory replay was designed to work in a moderately dynamic environment, we use controls computed for trajectory replay as inputs in our obstacle avoidance module.

This module was presented in previous articles ([11], [18]). Its principle is similar to such methods as Dynamic Window[5] or Ego-Kinematic Space[9]. Its specificity is mainly its expression as a Bayesian inference problem, making it particularly well suited for integration in this paper’s framework.

Basically, it takes as inputs data from proximity sensors and desired commands decided by an upper-level module (path planner, trajectory replay...) and uses them to compute commands really applied to the robot. These commands are built so as to follow desired commands as much as possible while granting security. We refer readers to our previous works for more details on this module.

VIII. RESULTS

A. Results on simulated platform

Results presented on figures 5 and 6 are computed on a simulated CyCab, equipped with a simulated sensor. This latter tries to mimic as closely as possible the behaviour of real sensor, particularly its limited field of view (180 degrees).

Both replay are made with respect to the same sensori-motor trajectory in the same environment (fig. 5 & 6, upper left). The only difference is initial position, or more precisely, initial orientation. Actually, initial position is a point located a 1.5 meter from the middle of the trajectory, with orientations that differs by 180 degrees (filled triangle in fig. 5 & 6, lower left).

From initial perception, an initial temporal localisation is computed (figs. 5 & 6, upper right). This estimation is quasi-certain in case of fig 5, not so certain in fig. 6. In this latter, most probable $\tau(0)$ is the one whose matching with observation is the least unlikely.

In both case, replay is started from initial temporal localisation, with inhibited obstacle avoidance. In fig. 5, initial position was quite close to nominal trajectory (1.5 meter, 0.05 radian), so trajectory tracking brings rapidly the robot on its nominal trajectory (lower left, and confidence rises to 0.9: quasi full confidence that localisation was good and replay successful. Conversely, in fig. 5, initial position was far from reference ($\approx \pi$ radians), so initial temporal localisation was wrong. In this case, trajectory tracking tries to follow hypothesised trajectory, but evidences against current hypothesis accumulates and confidence rapidly decreases toward zero.

Note that in previous experiments, confidence decrease has no influences on applied controls. One could suggest to make maximum speed decrease with confidence, so as to make the robot all the more cautious since its self-confidence is low.

B. Results on real platform

Figure 7 illustrates results of replayed trajectory on our real car-like robot. Our landmark detector is used as sensory input, and obstacle avoidance is used to check
whether proposed controls are safe. Trajectory replay is executed quite accurately (even if not so smoothly) at 2 m/s, moving at a few tenth of centimetres from parked cars.

IX. CONCLUSIONS

This paper presented how behavioural replay of a sensori-motor trajectory can be expressed as a fully Bayesian application: temporal and spatial localisation, control generation, obstacle avoidance and failure diagnosis were successfully implemented and integrated on a simulated robot and on a car-like autonomous vehicle on the car park area of our institute.

Using this Bayesian modular framework, we developed a set of tools that have been used not only in this work, but also in a more classical robotic application: trajectory planning and execution in a moderately dynamic environment[18].

Acknowledgements: This work was partially supported by the European project IST-2001-32115 “Bio-Inspired Brain & Artefacts” (http://www-biba.inrialpes.fr).

X. REFERENCES