Docking Task for Nonholonomic Mobile Robots

Olivier Lefebvre, Florent Lamiraux
LAAS-CNRS
7 avenue du Colonel Roche
Toulouse, France
[olefebvr, florent]@laas.fr

***Abstract*— This paper presents a framework for precise parking for nonholonomic mobile robots: the docking task. It consists in following a planned trajectory and reaching a docking configuration, defined relatively to the environment. The trajectory is deformed in order to reach the docking configuration, to avoid obstacles and to keep the nonholonomic constraints satisfied. A generic framework to compute the docking configuration is presented. Then we give the principle of a nonholonomic path deformation method that was used to deform the planned trajectory towards the docking configuration. This framework has been tested on a real robot with a trailer in a realistic scenario.**

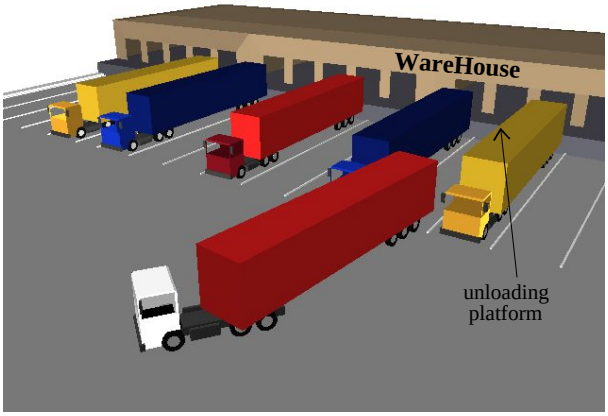

Fig. 1. A docking task for a truck: The final configuration is defined relatively to the unloading platform and to the white lines on the ground. The truck must move autonomously to this docking configuration.

I. Problem Statement and Related Work

The ability for a nonholonomic mobile robot to follow a planned trajectory while avoiding obstacles is of great interest since common vehicles are subjected to nonholonomic constraints. Research carried out on nonholonomic systems have potential applications in the Intelligent Transportation Systems area, such as automatic road, automatic parking or truck parking facilities.

In all these applications, a good localization of the system is essential if we want to follow the planned trajectory. Nevertheless it can happen that the trajectory needs to be adapted online. For instance if we want to avoid unexpected obstacles that were not in the map used to plan the trajectory. Moreover, the end of the trajectory may also need to be adapted at the parking stage, for several reasons:

- the parking process can require a precision that the localization is not able to provide,
- the map used for planning may be too imprecise to be employed to park in.
- the parking position may have changed.

All these elements converge towards the same idea: defining the parking configuration in the global frame does not allow for parking in practical applications. The parking configuration must be defined relatively to the environment. For instance, one can define a car parking lot as: “three white lines on the ground. One on each side of the car and one in front of it” (see figure 2). That is, a parking configuration can be defined indirectly through a set of landmarks to be perceived from this configuration. We call this set of landmarks a *docking pattern*.

In this paper we address the problem of precise motion of nonholonomic systems during the parking stage. Our approach takes advantage of a nonholonomic path deformation method [6] to reach a final configuration defined relatively to the environment.

The idea of defining a position as a desired sensor perception is the basis of sensor-based control. An instance of this approach is visual servoing [2], [5]. The objective is the positioning of a mobile camera with regard to the environment, with a task directly expressed as an error with respect to a goal image. The control is derived using the task function approach [8]. It has been extended to nonholonomic mobile robots in [11] by introducing additional degrees-of-freedom. The nonholonomic Camera-Space Manipulation (CSM) framework also addresses the issue of visual control of a nonholonomic mobile robot [10]. The extension to Mobile Camera-Space Manipulation (MCSM) [9] enables the cameras to be embedded on the robot. However it does not deal with systems with several nonholonomic constraints. A more generic framework for nonholonomic systems control has recently been proposed in [7]. Another advantage of our approach is that it can be coupled with a local obstacle avoidance method, as shown later.

To reach the final parking configuration, one could think about re-planning a trajectory using the current perception. This problem of reaching a constrained parking configuration is very similar to the extensively studied problem of part disassembly [3], [4]. However these works do not take into account nonholonomic systems. Moreover re-planning a trajectory using the current perception would be very time consuming. That is the reason why we will try to locally deform the reference trajectory rather than launch a global re-planning.

The paper is organized as follows: in section II we specify the concept of a docking task that allows to define the docking configuration with respect to a set of landmarks in the

environment (the *docking pattern*). In section III, we explain how to compute the docking configuration given a sensor perception and a *docking pattern*, using standard Kalman filtering techniques. In section IV we present the method used to deform the reference trajectory in order to avoid obstacles, to reach the docking configuration and to keep the nonholonomic constraints satisfied. Eventually, in section V, we present experimental results with a real robot towing a trailer.

II. Docking task

Autonomous motion for a mobile robot is generally addressed in two steps. First a collision-free trajectory is planned within a model of the environment. Then the robot follows this reference trajectory and adapts it locally in order to avoid unexpected obstacles.

As explained in the introduction, this technique does not allow precise parking, since it does not adapt the parking configuration to the environment. The concept of docking task addresses this issue.

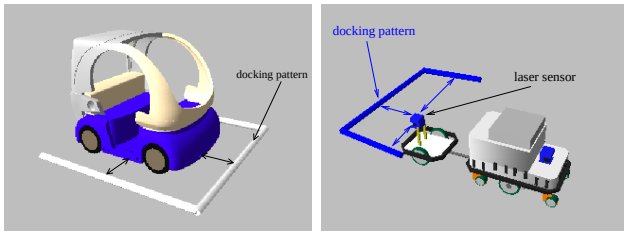

Fig. 2. *Docking pattern*. It consists in a set of landmarks defined relatively to a sensor. On the left image, the *docking pattern* defines a parking lot for a CyCab car. On the right image the *docking pattern* is defined relatively to the laser sensor mounted on the trailer of a robot.

A docking task is a mission given to a robot that consists in following a planned trajectory and reaching a docking configuration. The docking configuration is not defined beforehand as a known robot location. On the contrary it is specified as a set of sensor perceptions from this configuration. The set of landmarks to be perceived when the robot is at the docking configuration is called a *docking pattern*. Figure 2 presents such *docking patterns*. On each image, the docking configuration is represented relatively to the docking pattern.

Thus a docking task takes as input:

- a collision free trajectory planned within a model of the environment
- a set of landmarks relative to the docking configuration: the *docking patterns*.

Figure 3 illustrates the principle of a docking task. The robot is following the trajectory planned from q_{init} to q_{end}. Arriving at the end, it detects the docking pattern in the environment using its sensor. It must then:

- compute the docking configuration q_{dock} defined as the configuration where the *docking pattern* matches the sensor perception,
- deform the trajectory to reach the docking configuration while avoiding collisions and keeping the nonholonomic constraints satisfied.

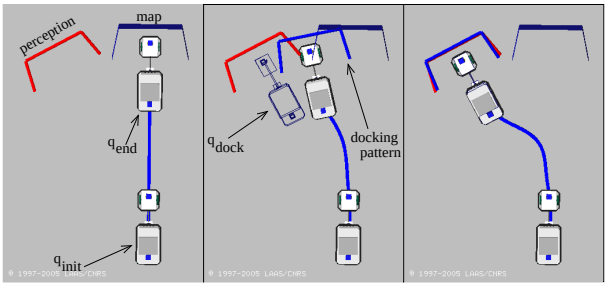

Fig. 3. A trajectory planned for a robot towing a trailer, with a false model of the environment. The *docking pattern* is a set of landmarks relative to the docking configuration. The trajectory is deformed in order to avoid obstacles, to keep nonholonomic constraints satisfied and to reach $\mathbf{q}_{dock}$: the configuration where the *docking pattern* matches sensor perception.

III. Docking configuration computation

In the absence of any additional information, the docking configuration is the last configuration of the planned trajectory. Otherwise, the comparison between *docking patterns* and sensors perceptions can be used to compute the docking configuration: i.e. the robot configuration where sensors perceptions best match *docking patterns*. We use a classical Extended Kalman filter approach with the observation step, the matching step and the update step, to integrate this information.

A. Notations

1) Configurations and positions: Let $\mathcal{C}$ be the configuration space of our system. A configuration of the robot is denoted by $\mathbf{q}$. Let $\mathbf{q}^{dock}$ be the docking configuration, the configuration we are computing in this section.

Let $\mathcal{W}$ represent the 3D workspace, with origin frame $\mathbf{O}$. The position and orientation of a frame $\mathbf{F}'$ expressed in a frame $\mathbf{F}$ can be represented by the homogeneous matrix $\mathbf{x}_{\mathbf{F}'/\mathbf{F}}$. A frame $\mathbf{F}$ expressed in the workspace is simply denoted by $\mathbf{x}_{\mathbf{F}}$, representing the transformation from frame $\mathbf{O}$ to frame $\mathbf{F}$. We consider a multi-body robot equipped with n sensors and we note $\mathbf{x}_i(\mathbf{q})$ the position of sensor i when robot is at configuration $\mathbf{q}$. From now on, we refer to the sensor position when robot is at docking configuration simply as $\mathbf{x}_i^{dock}$:

$$\mathbf{x}_i^{dock} = \mathbf{x}_i(\mathbf{q}^{dock})$$

2) Observation function of a sensor: We focus now on a single sensor. The current robot configuration is $\mathbf{q}$ and the current sensor position is $\mathbf{x}(\mathbf{q})$. We are interested in computing $\mathbf{x}^{dock}$.

Let $\mathbf{l}$ be a landmark[1], represented by a n^l dimensional real vector. A landmark is defined relatively to the sensor position when the robot is in docking configuration. That is a landmark $\mathbf{l}$ is expressed in frame $\mathbf{x}^{dock}$:

$$\mathbf{l} \triangleq \mathbf{l}_{/\mathbf{x}^{dock}}$$

Then for each sensor, we define a *docking pattern* as a set of l landmarks $\mathcal{L} = \{\mathbf{l}^1, \mathbf{l}^2, \ldots, \mathbf{l}^l\}$.

Let $\mathbf{p}$ be a feature perceived[2] by the sensor, and represented by a n^p dimensional real vector. The perception is naturally

[1] $\mathbf{l}$ for *landmark*
[2] $\mathbf{p}$ for *perception*

acquired in sensor frame $\mathbf{x}(\mathbf{q})$. But for computation convenience we need it expressed in the workspace $\mathcal{W}$, that is in frame $\mathbf{O}$:

$$\mathbf{p} \triangleq \mathbf{p}_{/\mathbf{O}}$$

It is always possible to define a function TR that transforms a perception expressed in frame $\mathbf{x}(\mathbf{q})$ into a perception expressed in frame $\mathbf{O}$:

$$\mathbf{p}_{/\mathbf{O}} = TR(\mathbf{x}(\mathbf{q}), \mathbf{p}_{/\mathbf{x}(\mathbf{q})}) \tag{1}$$

If the perceived feature $\mathbf{p}$ is a point for instance, the transformation function TR is simply the homogeneous matrix $\mathbf{x}(\mathbf{q})$. Then we note $\mathcal{P} = \{\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^p\}$ the set of p features perceived by the sensor.

We define an observation function f that maps a sensor position $\mathbf{x}$ and a landmark from $\mathcal{L}$ with a perception from $\mathcal{P}$ as :

$$\begin{array}{rccc} f: & \mathbb{R}^{n^l} \times SE(3) & \rightarrow & \mathbb{R}^{n^p} \\ & (\mathbf{l}, \mathbf{x}) & \mapsto & \mathbf{p} = f(\mathbf{l}, \mathbf{x}) \end{array} \tag{2}$$

What is important to notice here is that the sensor position $\mathbf{x}^{dock}$ when the robot is at docking configuration is solution of equation (2). Then, given a sensor docking position $\mathbf{x}^{dock}$, the set of n^m couples $m = (\mathbf{l}^j, \mathbf{p}^k)$ that satisfy equation (2) is noted $\mathcal{M}$. It is the set of matches landmark-perception.

We can define a batch observation function F that maps all the elements of $\mathcal{M}$ from a given sensor position $\mathbf{x}$. Let $\mathbf{L}$ and $\mathbf{P}$ be such that:

$$\begin{array}{ccc} \mathbf{P} & = & F(\mathbf{L}, \mathbf{x}) \\ \begin{pmatrix} \mathbf{p}^1 \\ \vdots \\ \mathbf{p}^{n^m} \end{pmatrix} & = & \begin{pmatrix} f(\mathbf{l}^1, \mathbf{x}) \\ \vdots \\ f(\mathbf{l}^{n^m}, \mathbf{x}) \end{pmatrix} \end{array} \tag{3}$$

Figure 4 presents these notations in a docking task scene with two sensors. The robot being at a current configuration $\mathbf{q}$, it must compute the docking configuration $\mathbf{q}^{dock}$ (right image). At this docking configuration, the observation function (2) is satisfied for all matched couples landmark-perception:$(\mathbf{l}_i^j, \mathbf{p}_i^j)$, for each sensor i. That is:
$\forall i, j$

$$\mathbf{p}_i^j = f(\mathbf{x}_i^{dock}, \mathbf{l}_i^j)$$

3) Probabilistic framework: Because we do not measure the true values of any of the preceding variables, we model them as real random variables. Measure noises are assumed normally distributed with zero mean.

Then equation (2) becomes:

$$\mathbf{p} = f(\mathbf{l}, \mathbf{x}) + \mathbf{w} \tag{4}$$

Where $\mathbf{w}$ is the error on perception $\mathbf{p}$. It is composed of a part due to sensor noise and of a part due to sensor localization error through equation (1). We note $\mathbf{V_w}$ its variance.

Supposing estimated values are close to real values, we can linearize equation (4) around the estimated value. Then the estimated observation and its variance are:

$$\begin{array}{rcl} \hat{\mathbf{p}} & = & f(\hat{\mathbf{l}}, \hat{\mathbf{x}}) \\ \mathbf{V_p} & = & J_{\mathbf{x}} \mathbf{V_x} J_{\mathbf{x}}^T + J_{\mathbf{l}} \mathbf{V_l} J_{\mathbf{l}}^T + \mathbf{V_w} \end{array} \tag{5}$$

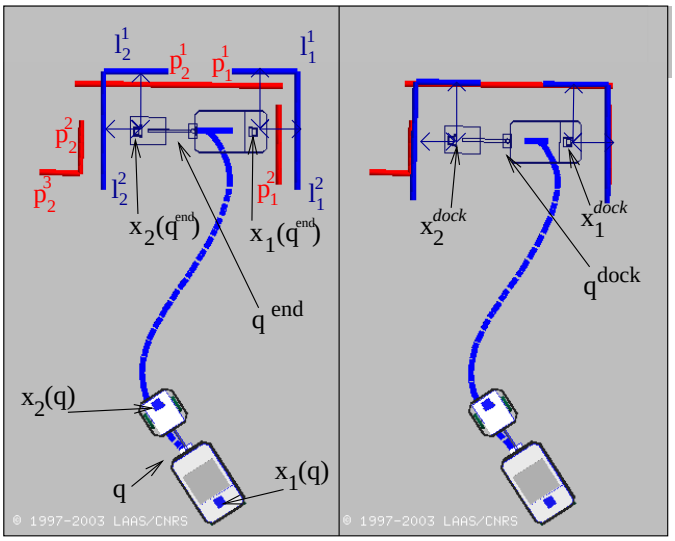

Fig. 4. Notation for the docking configuration computation in the case of two sensors. On the left image, the couples landmark-perception are represented. For each sensor i, landmark $\mathbf{l}_i^j$ and perception $\mathbf{p}_i^j$ match together. Landmarks are carrier line of segments. The current robot configuration is $\mathbf{q}$. Current sensors positions are respectively $\mathbf{x}_1(\mathbf{q})$ and $\mathbf{x}_2(\mathbf{q})$. The sensors positions at the end of the trajectory are $\mathbf{x}_1(\mathbf{q}^{end})$ and $\mathbf{x}_2(\mathbf{q}^{end})$ and they are not solutions of observation function (2) for any couple $(\mathbf{l}_i^j, \mathbf{p}_i^j)$. On the right image, sensors docking positions $\mathbf{x}_1^{dock}$ and $\mathbf{x}_2^{dock}$ are computed as solutions of observation function (2).

Where $J_{\mathbf{x}} = \left.\frac{df}{d\mathbf{x}}\right|_{\hat{\mathbf{x}}}$ is the Jacobian of f with respect to $\mathbf{x}$ evaluated at $\hat{\mathbf{x}}$. And $J_{\mathbf{l}} = \left.\frac{df}{d\mathbf{l}}\right|_{\hat{\mathbf{l}}}$ is the Jacobian of f with respect to $\mathbf{l}$ evaluated at $\hat{\mathbf{l}}$.
This is because $\mathbf{x}$, $\mathbf{l}$ and $\mathbf{w}$ are independent variables.

In a similar manner the batch function (3) becomes

$$\mathbf{P} = F(\mathbf{L}, \mathbf{x}) + \mathbf{W} \tag{6}$$

Where $\mathbf{W}$ is equal to $(\mathbf{w}^1, \ldots, \mathbf{w}^{n^m})^T$. The estimated observation vector $\hat{\mathbf{P}}$ and its covariance matrix $\mathbf{V_P}$ are computed similarly to equation (5).

One must notice that:

- $\mathbf{x}$ and $\mathbf{L}$ are independent variables since $(\forall \mathbf{l}^j \in \mathcal{L}) : \mathbf{x}$ and $\mathbf{l}^j$ are independent.
- the covariance matrix $\mathbf{V_W}$ is *not* diagonal since observation noises are correlated through equation (1).
- the covariance matrix $\mathbf{V_L}$ of the *docking pattern* expresses an *a priori* knowledge of the docking task.
- $\mathbf{x}$, $\mathbf{L}$ and $\mathbf{W}$ are independent variables.

The robot localization is known through a noisy process, as for its internal configuration variables. Thus $\hat{\mathbf{q}}$ represents an estimate of the current robot configuration, used in equation (1).

The docking configuration $\mathbf{q}^{dock}$, which is the variable of interest in this section, is modeled as a random variable. The *a priori* estimated value of $\mathbf{q}^{dock}$ is the last configuration of the planned trajectory. We note $\mathbf{q}^{\ominus}$ this *a priori* docking configuration:

$$\hat{\mathbf{q}}^{\ominus} = \mathbf{q}^{end} \tag{7}$$

Its variance is denoted by $\mathbf{V}_{\mathbf{q}^\ominus}$ and is a parameter of the docking task.

B. Matching

Arriving close to the *a priori* docking configuration $\hat{\mathbf{q}}^\ominus$, the robot must determine for each sensor i which elements perceived correspond to elements of the *docking pattern*. This matching step consists in finding for each sensor i the set $\mathcal{M}_i$ of couples $m_i = (\mathbf{l}_i^j, \mathbf{p}_i^k)$ landmark-perception that may verify equation (2).

Because the real values of the variables are unknown we can use only the estimated values. Thus equation (2) is never exactly satisfied and we are bound to find the couples that "best" match. The criterion we use to evaluate the likelihood of a match is the Mahalanobis distance between the expected perception and the actual perception. The expected perception $\hat{\mathbf{p}}^j$ of the landmark $\hat{\mathbf{l}}^j$ from the sensor position $\hat{\mathbf{x}}$ is given by equation (5).

At this time, $\hat{\mathbf{x}} = \mathbf{x}(\hat{\mathbf{q}}^\ominus) = \mathbf{x}(\mathbf{q}^{end})$ is the sensor position at the last configuration on the planned trajectory (see figure 4). For a given perception $\mathbf{p}^k$, the Mahalanobis distance $(D^{jk})^2$ is then defined as:

$$(D^{jk})^2 = (\mathbf{p}^k - \hat{\mathbf{p}}^j)^T \mathbf{V}_{\mathbf{p}} (\mathbf{p}^k - \hat{\mathbf{p}}^j) \tag{8}$$

This distance follows a $\chi^2_{n^p}$ distribution, with n^p the dimension of the observation vector $\mathbf{p}$.

Algorithm 1 *Matching algorithm*

for each sensor i **do**
  $\chi^2_{95} \leftarrow p(\chi^2_{n^p} = 95\%)$
  $\hat{\mathbf{x}}_i \leftarrow \mathbf{x}_i(\hat{\mathbf{q}}^\ominus)$
  $\mathcal{M}_i \leftarrow \emptyset$
  for each observation $\mathbf{p}_i^k$ in $\mathcal{P}_i$ **do**
    $D^2 \leftarrow \infty$
    $\mathbf{l}^{best} \leftarrow \varnothing$
    for each landmark $\mathbf{l}_i^j$ in $\mathcal{L}_i$ **do**
      $\hat{\mathbf{p}}_i^j \leftarrow$ equation (2), $\mathbf{l}_i^j$ and $\hat{\mathbf{x}}_i$
      $(D^{jk})^2 \leftarrow$ equation (8), $\hat{\mathbf{p}}_i^j$ and $\mathbf{p}_i^k$
      if $((D^{jk})^2 < \chi^2_{95} \wedge (D^{jk})^2 < D^2)$ **then**
        $\mathbf{l}^{best} \leftarrow \mathbf{l}_i^j$
        $D^2 \leftarrow (D^{jk})^2$
      end if
    end for
    if $\mathbf{l}^{best} \neq \varnothing$ **then**
      insert $\{\mathbf{p}_i^k, \mathbf{l}^{best}\}$ in $\mathcal{M}_i$
    end if
  end for
end for

Algorithm 1 describes the matching. It returns for each sensor i a list $\mathcal{M}_i$ of couples landmark-perception that is used in the update step.

C. Update step

Let $\hat{\mathbf{x}}_i^\ominus = \mathbf{x}_i(\hat{\mathbf{q}}^\ominus)$ denote the sensor position at the *a priori* docking configuration.

The update then is done in two steps :

- Each sensor position at the docking configuration is updated using the matching step: $\hat{\mathbf{x}}_i^\ominus \rightarrow \hat{\mathbf{x}}_i^\oplus$.
- then the docking configuration is updated using the previous step: $\hat{\mathbf{q}}^\ominus \rightarrow \hat{\mathbf{q}}^\oplus$.

1) sensors docking positions update: The list of matching couples $(\mathbf{p}_i, \mathbf{l}_j)$ of each sensor i is used to update the prior estimated sensor docking position $\hat{\mathbf{x}}_i^\ominus$. The prior docking configuration $\hat{\mathbf{q}}^\ominus$ is the last configuration on the planned trajectory (equation (7)). We are looking for the *a posteriori* value $\mathbf{x}_i^\oplus$: $\mathbf{x}_i$ knowing the matching couples set $\mathcal{M}_i$.

We note $\mathbf{Z}_i$ the innovation vector of sensor i: $\mathbf{Z}_i = \mathbf{P}_i - \hat{\mathbf{P}}_i$. It is the difference between the actual and the expected perception. Using the notations of Kalman filter we have:

$$\begin{aligned} \hat{\mathbf{x}}_i^\oplus &= \hat{\mathbf{x}}_i^\ominus + K_{\mathbf{x}} \mathbf{Z}_i \\ \mathbf{V}_{\mathbf{x}_i}^\oplus &= (I - K_{\mathbf{x}} J_{\mathbf{x}}) \mathbf{V}_{\mathbf{x}_i^\ominus} \end{aligned}$$

with the Kalman gain:

$$K_{\mathbf{x}} = \mathbf{V}_{\mathbf{x}_i^\ominus} J_{\mathbf{x}}^T . (J_{\mathbf{x}} \mathbf{V}_{\mathbf{x}_i^\ominus} J_{\mathbf{x}}^T + J_{\mathbf{L}} \mathbf{V}_{\mathbf{L}_i} J_{\mathbf{L}}^T + \mathbf{V}_{\mathbf{W}})^{-1}$$

And $J_{\mathbf{x}} = \frac{dF_i}{d\mathbf{x}}\big|_{\hat{\mathbf{x}}_i^\ominus}$ is the Jacobian of F_i with respect to $\mathbf{x}$ evaluated at $\hat{\mathbf{x}}_i^\ominus$. And $J_{\mathbf{L}} = \frac{dF_i}{d\mathbf{L}}\big|_{\hat{\mathbf{L}}_i}$ is the Jacobian of F_i with respect to $\mathbf{L}$ evaluated at $\hat{\mathbf{L}}_i$. As mentioned at the end of section III-A.2, this writing is possible since $\mathbf{L}$, $\mathbf{x}$ and $\mathbf{W}$ are independent.

2) docking configuration update: Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_n)^T$ be the column vector composed of all sensors docking positions. Since $\hat{\mathbf{x}}_i^\ominus = \mathbf{x}_i(\hat{\mathbf{q}}^\ominus)$, we note $\mathbf{X}^\ominus$ the *a priori* positions of sensors. The difference between $\mathbf{X}^\oplus$ and $\mathbf{X}^\ominus$ is used to update the docking configuration:

$$\hat{\mathbf{q}}^\oplus = \hat{\mathbf{q}}^\ominus + K_{\mathbf{q}} (\hat{\mathbf{X}}^\oplus - \hat{\mathbf{X}}^\ominus)$$

and the Kalman gain is:

$$K_{\mathbf{q}} = \mathbf{V}_{\mathbf{q}^\ominus} J_{\mathbf{q}}^T . (J_{\mathbf{q}} \mathbf{V}_{\mathbf{q}^\ominus} J_{\mathbf{q}}^T + \mathbf{V}_{\mathbf{x}_i}^\oplus)^{-1}$$

With $J_{\mathbf{q}} = \frac{d\mathbf{x}_i(\mathbf{q})}{d\mathbf{q}}\Big|_{\hat{\mathbf{q}}^\ominus}$ is the Jacobian of $\mathbf{x}_i(\mathbf{q})$ with respect to $\mathbf{q}$ evaluated at $\hat{\mathbf{q}}^\ominus$.

3) Batch update versus sequential update: We can remark that all updates are done in a batch way, that is all measures are concatenated in a single vector. The reason is that in our case, each measure vector (sensor perceptions $\mathbf{P}$ or sensor positions $\mathbf{X}$) is not independent element by element. However, for a large number of sensors and a large number of matching couples for each sensor, a sequential update would be preferable for computational complexity reasons. It can always be done by diagonalizing the covariance matrix of the measures as shown in [1].

D. Under-determined cases

A *docking pattern* $\mathcal{L}_i$ does not always fully determine a sensor position. For instance if the embedded sensor detects lines and the *docking pattern* is made of one line only, one degree of freedom is missing: the localization of the docking configuration with respect to the *docking pattern* is under-determined. It is important to notice that the working out presented above manages these cases since it uses the current final configuration of the trajectory as a prior estimation of the docking configuration.

IV. Nonholonomic Trajectory Deformation

Now that we have presented how to compute the desired docking configuration, we present how to deform the planned trajectory toward the docking configuration.

A method has been presented in [6] that reactively deforms a trajectory for a nonholonomic system in order to avoid obstacles detected by on-board sensors along the motion. The method is based on the minimization of a trajectory potential that increases when the trajectory gets closer to obstacles.

We present here the principle of this method and we show how it can be used to reach a desired goal configuration.

A. Principle

A nonholonomic system of dimension n is defined by $k < n$ control vector fields $\mathbf{X}_1$,...,$\mathbf{X}_k$ over the configuration space $\mathcal{C}$ of the system. An admissible trajectory $\mathbf{q}(s)$ is a mapping from an interval $[0, S]$ into the configuration space the derivative of which is a linear combination of the control vector fields and there exists a k-dimensional vector valued smooth mapping $\mathbf{u} = (u_1, ..., u_k)$ from $[0, S]$ into $\mathbb{R}$ such that:

$$\forall s \in [0, S], \quad \mathbf{q}'(s) = \sum_{i=1}^{k} u_i(s)\mathbf{X}_i(\mathbf{q}(s))$$

$'$ denotes the derivative w.r.t. s. The u_i's are the input functions relative to trajectory $\mathbf{q}$.

1) Direction of deformation: The nonholonomic trajectory deformation method is based on the perturbation of the input functions of the current trajectory $\mathbf{q}$. Let the input perturbation be defined as a k-dimensional vector valued smooth mapping $\mathbf{v} = (v_1, ..., v_k)$ from $[0, S]$ into $\mathbb{R}$ so that replacing each u_i by $u_i + \tau v_i$, where τ is a small positive real number, yields a new admissible trajectory:

$$\mathbf{u} \leftarrow \mathbf{u} + \tau\mathbf{v} \tag{9}$$

$$\mathbf{q}(s) \leftarrow \mathbf{q}(s) + \tau\eta(s) \tag{10}$$

$\eta(s)$ is called the *direction of deformation* and verifies:

$$\eta'(s) = A(s)\eta(s) + B(s)\mathbf{v}(s) \tag{11}$$

where $A(s)$ and $B(s)$ are the following $n \times n$ matrices:

$$A(s) = \sum_{i=1}^{k} u_i(s)\frac{\partial X_i}{\partial \mathbf{q}}(\mathbf{q}(s)) \quad B(s) = (X_1(\mathbf{q}(s)), \ldots, X_k(\mathbf{q}(s)))$$

2) Choice of input perturbation: The input perturbation $\mathbf{v}$ is restricted over a finite-dimensional subset of functions. For that, we define $\mathbf{e}_1, ..., \mathbf{e}_p$ ($p > n$), a set of smooth linearly independent vector-valued functions [3] of dimension k, defined over $[0, S]$ and we let the input perturbation be a linear combination of these input functions:

$$\mathbf{v}(s) = \sum_{i=1}^{p} \lambda_i \mathbf{e}_i(s) \tag{12}$$

For each of these functions, let $\mathbf{E}_i(s)$ be the solution of system (11) with initial condition $\eta_0 = 0$ and with $\mathbf{e}_i(s)$ as input. Since system (11) is linear in $\mathbf{v}$, the direction of deformation η corresponding to $\mathbf{v}$ is the linear combination of solutions $\mathbf{E}_i$

$$\eta(s) = \sum_{i=1}^{p} \lambda_i \mathbf{E}_i(s) \tag{13}$$

The *direction of deformation* is thus completely determined by vector λ. Now we present how to choose vector λ so that the deformed trajectory moves away from obstacles.

3) Trajectory Potential: We define a potential field U over the configuration space, decreasing when the distance between the robot and the obstacles increases. From this potential field, we define a potential field over the space of trajectories by integration of the configuration space potential value:

$$V = \int_0^S U(\gamma(s))ds$$

The variation of the potential induced by the input perturbation is given by:

$$\begin{aligned} \Delta V &= \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s))^T \eta(s) ds \\ &= \sum_{i=1}^{p} \lambda_i \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s))^T \mathbf{E}_i(s) ds \end{aligned}$$

The choice of λ that makes the variation of the potential negative is consequently:

$$\lambda_i = -\int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s))^T \mathbf{E}_i(s) ds \tag{14}$$

B. Boundary conditions

In the context of obstacle avoidance, we generally deform a portion of the trajectory only, on which a collision has been found. In order to keep the feasibility of the whole trajectory, two boundary conditions are imposed:

$$\eta(0) = 0 \tag{15}$$

$$\eta(S) = 0 \tag{16}$$

The first constraint is always satisfied since each $\mathbf{E}_i$ satisfies (15).

The second constraint (16), imposing the last configuration is unchanged, is in fact a linear constraint over vector λ:

$$L\lambda = 0 \tag{17}$$

Where $L = (\mathbf{E}_1(S), \cdots, \mathbf{E}_p(S))$ the $n \times p$-matrix ($p > n$) the columns of which are the $\mathbf{E}_i(S)$'s.

In the context of docking, we want the deformed trajectory to reach configuration $\mathbf{q}^{dock}$ that has been computed previously (section III). We note $\delta^{dock} \in \mathcal{C}$ the difference between the docking configuration and the last configuration of the actual trajectory ($\delta^{dock} = \mathbf{q}^{dock} - \mathbf{q}(S)$). The boundary condition in the context of docking is then:

$$L\lambda = \delta^{dock} \tag{18}$$

We project the vector λ computed from equation (14) over this subspace. We note L^+ the pseudo-inverse of L. It is the matrix verifying $LL^+L = I_p$. Then we have:

$$\bar{\lambda} = L^+\delta^{dock} + (I_p - L^+L)\lambda$$

the closest vector to λ that verifies equation (18).

[3] truncated Fourier series are used

V. Experimental results

A common scenario for a truck with a trailer is to park its trailer along an unloading platform. That is the final position of the trailer is defined relatively to the unloading platform. We have reproduced this scenario with a robot towing a trailer.

The trailer is endowed with a laser range sensor. Following the notation of section III-A.2, we define the *docking pattern* as a set of landmarks $\mathcal{L}$ relative to this sensor. In this experiment the landmarks are segments. The *docking pattern* $\mathcal{L}$ can be composed of any number of segments $\mathbf{l}_i$. In order to be robust to occlusions, the matching algorithm of section III-B treat segments as straight lines. In this experiment, the *docking pattern* represents the shape of the unloading platform as perceived by the sensor when the trailer is parked. It is represented in figure 2.

Thus the inputs of the docking task are:

- a planned trajectory for the robot towards a goal configuration
- the *docking pattern* $\mathcal{L}$.

A. The Unloading platform has been moved

Figure 5 illustrates the case where the unloading platform has been moved and the map has not been updated. Moreover, the shape of the unloading platform has changed: it is larger than the *docking pattern*. The matching between the perception and the *docking pattern* is robust to these perturbations and the docking configuration is still defined relatively to the unloading platform.

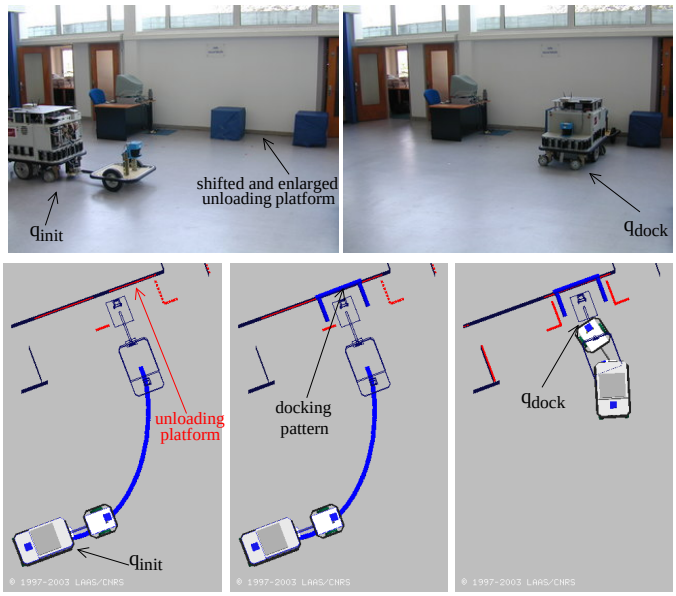

Fig. 5. The position and the shape of the unloading platform have been changed compared to the map of the environment. The unloading platform has been shifted to the right and it has been enlarged by 0.2 meters. The docking configuration is computed as the configuration where the *docking pattern* best fits the unloading platform.

Quantitative results are very good in these experiments. The error between the theoretical trailer position at the unloading platform and the experimental position is about 5 centimeters. The error is principally transversal to the robot and is mainly due to the robot motion control law that converges slowly in the transversal direction. The longitudinal error is less than 1 centimeter.

VI. Conclusion

We have presented a framework for sensor-based maneuvers for nonholonomic mobile robots: the docking task. It consists in defining a desired goal configuration of the robot relatively to the environment using a *docking pattern*. Given a planned trajectory, the docking task consists in following the trajectory while avoiding obstacles and in reaching the docking configuration: the configuration where sensor perception best matches the *docking pattern*. We use a nonholonomic path deformation method to make the planned trajectory reach the docking configuration.

This framework is generic for any nonholonomic mobile robot. Any number of sensors can be used, and *docking patterns* can possibly not fully determine the docking configuration.

It has been tested on a real robot with a trailer in a realistic scenario using a laser range finder to detect the docking pattern. An extension of this work would be to use a camera as a sensor in order to dock with respect to an image pattern.

References

[1] Y. Bar-Shalom and X.R. Li. *Estimation and Tracking: Principles, Techniques, and Software*. Artech House, Incorporated, 1993.

[2] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.

[3] E. Ferre and J.P. Laumond. An iterative diffusion algorithm for part disassembly. In *ICRA04*, New Orleans, April 2004. IEEE.

[4] D. Hsu, L.E. Kavraki, J.C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In P.K. Agarwal et al., editors, *Workshop on the Algorithmic Foundations of Robotics*, pages 141–154. A. K. Peters, 1998.

[5] S. A. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Trans. Robotics and Automation*, 12(5):651–670, October 1996.

[6] F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for non-holonomic mobile robots. *IEEE Transactions on Robotics*, 20(6):967–977, December 2004.

[7] P. Morin and C. Samson. Practical stabilization of driftless systems on lie groups: the transverse function approach. *IEEE Trans. on Automatic Control*, 48(9):1496–1508, September 2003.

[8] C. Samson, M. Leborgne, and B. Espiau, editors. *Robot Control: The Task-function Approach*. Oxford University Press, 1991.

[9] M. Seelinger, J.-D. Yoder, E.T. Baumgartner, and S.B. Skaar. High precision visual control of mobile manipulators. *IEEE Transactions on Robotics and Automation*, 18(6):957–965, Dec 2003.

[10] S.B. Skaar, I. Yalda-Mooshabad, and W.H. Brockman. Nonholonomic camera-space manipulation. *IEEE Transactions on Robotics and Automation*, 8:464–479, August 1992.

[11] D. Tsakiris, P. Rives, and C. Samson. Applying visual servoing techniques to control nonholonomic mobile robots. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, Grenoble, France, September 1997.