Motion Tasks for Robot Manipulators on Embedded 2-D Manifolds

Xanthi Papageorgiou, Savvas G. Loizou and Kostas J. Kyriakopoulos

Abstract— **In this paper we present a methodology to drive the end effector of a robotic manipulator across the surface of an object in the workspace. Three typical tasks are considered, namely stabilization of the end effector over the object's surface, motion planning and eventually trajectory tracking of the end effector across the object's surface. The proposed controllers utilize navigation functions and are based on the belt zone vector fields concept. The derived dynamic controllers are realized using an integrator backstepping methodology. The derived feedback based controllers guarantee global convergence and collision avoidance. The closed form solution provides fast feedback rendering the methodology particularly suitable for implementation on real time systems. The properties of the proposed methodology are verified through non-trivial computer simulations.**

I. INTRODUCTION

Performing motion tasks across object surfaces constitutes a challenging problem of the robotics field with many applications including robotic surface painting, surface cleaning, surface inspection, etc. Our main motivation comes from the field of neuro-robotics. One of the main tasks of neuro-robotics is to make a robot execute a task by interfacing with the neural system (Fig. 1) e.g. by processing electromyographic activity, etc. In most of the cases, these signals are noisy and rather "incomprehensible" to directly control a robot particularly in clattered environments. In those cases we need a strategy to make the robot compliant with its environment and at the same time avoiding obstacles. The main difficulties of the above tasks arise when the considered surface is not planar. Moreover the non-planar surface might include "bad regions" that must be avoided.

Most of previous relevant research has focused on the problem of automotive painting of surfaces that are convex and have no holes, [1], [2], [3]. Also in [1], the authors decompose the coverage trajectory generation problem into three subproblems: selection of the start curve, selection of the speed profiles along each pass, and selection of the spacing between the passes.

In this work, we have used navigation functions, [4], to drive the manipulator safely to a surface. Once the end-effector is in the close proximity of the surface, a second controller takes over to stabilize the end-effector at a predefined distance from the surface (which partly depends on the surface curvature), while at the same time, depending on the application, performs a motion planning or trajectory tracking task across the surface, avoiding the "bad regions". Our basic idea is to use a navigation controller to drive the manipulator's end effector safely to the proximity area of the surface of interest and then according to the required task (motion planning or trajectory tracking), switch to a task specific controller. Each of those two task specific controllers can successfully carry out the task while keeping the manipulator's end effector at a predefined distance from the surface of interest. This is achieved through the use of appropriately constructed belt zone and task specific vector fields, introduced in [5], [6]. To handle the volume and the articulated nature of the robot manipulator we have used the methodology that was first introduced in [7]. The main contributions of this paper can be summarized as follows:

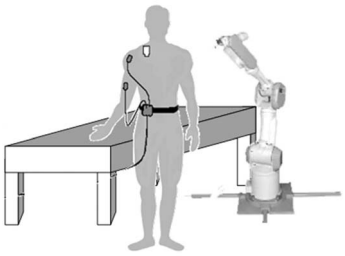

Fig. 1. The problem motivation.

- A novel theoretically guaranteed dynamic trajectory tracking controller, achieving obstacle avoidance and concurrent stabilization during tracking over 2-D manifolds embedded in 3-D workspaces, for articulated robot manipulators.
- A provably correct dynamic controller performing motion planning and concurrent stabilization tasks over 2-D manifolds embedded in 3-D workspaces, for articulated robot manipulators.

The rest of the paper is organized as follows: Section II formally states the considered problem, introducing preliminary definitions, notation and some technical Lemmas, required for further discussion. Section III describes the navigation of an articulated non-point robot while section IV presents the construction of the vector field that is used for robot navigation. Section V introduces the proposed control law. Section VI presents the simulation results and the paper concludes with section VII.

This work is partially supported by the European Commission through contract "FP6 - IST - 001917 - NEUROBOTICS: The fusion of Neuroscience and Robotics".

X. Papageorgiou is a PhD Student in the Mechanical Engineering Department, National Technical University of Athens, Athens, Greece, xpapag@mail.ntua.gr

S.G. Loizou is a Post Doctoral Researcher in the Grasp Laboratory, University of Pennsylvania, Philadelphia, PA 19104-6228, USA, sloizou@seas.upenn.edu

K.J. Kyriakopoulos is with the Faculty of Mechanical Engineering, National Technical University of Athens, Athens, Greece, kkyria@mail.ntua.gr

Proc. IEEE Int'l Symposium on Intelligent Control Munich (ISIC-2006), Germany, October 4-6, 2006. p.p 3047-3052

II. PROBLEM STATEMENT

Our analysis is demonstrated by considering the motion planning problem of a robotic manipulator in a workspace with obstacles under additional task constraints. Consider a manipulator of m-dof, which is described dynamically as:

$$B(q)\cdot\ddot{q}+C(q,\dot{q})+gr(q)=\tau \tag{1}$$

where $B(q)$ is the inertia matrix, $C(q,\dot{q})$ is the Coriolis term, $gr(q)$ is the Gravity term, $q=[q_1\ \dots\ q_m]^T\in\mathbb{R}^m$ is the vector of arm joint variables and u the joint torque inputs, [8]. Let the admissible and feasible configuration space (workspace) for the manipulator be $\mathscr{W}\subset\mathbb{R}^m$. The obstacle free subset of the workspace is denoted $\mathscr{W}_{free}\subseteq\mathscr{W}$, and $\varphi:\mathscr{W}\to\mathbb{R}$ is the potential (navigation) function which models the environment. Let $\mathscr{O}\in\mathscr{W}\setminus\mathscr{W}_{free}$ be the set of all obstacles in 3-D workspace. Define a vector valued C^2 function:

$$g(s_1,s_2):\mathbb{R}\times\mathbb{R}\to\mathscr{R}(g) \tag{2}$$

representing a closed surface. The range $\mathscr{R}(g)\subset\mathscr{W}_{free}$ of the function will serve as the boundary of the surface across which (in a $\delta>0$ distance) the surface processing task will take place. Let us define the following topology (Fig. 2):

1) The surface's internal, J^-.
2) The surface's boundary, ∂g.
3) The surface's external, J^+.

The tangent vectors on the surface w.r.t parameters s_1 and s_2 can be defined as:

$$g_{s_1}(s_1,s_2)=\frac{\partial g(s_1,s_2)}{\partial s_1}=\left[\frac{\partial g_x}{\partial s_1},\frac{\partial g_y}{\partial s_1},\frac{\partial g_z}{\partial s_1}\right]^T \tag{3}$$

$$g_{s_2}(s_1,s_2)=\frac{\partial g(s_1,s_2)}{\partial s_2}=\left[\frac{\partial g_x}{\partial s_2},\frac{\partial g_y}{\partial s_2},\frac{\partial g_z}{\partial s_2}\right]^T \tag{4}$$

where the g_x,g_y,g_z denote the coordinate functions of g across the respective dimension.

Due to the C^2 continuity of $g(s_1,s_2)$, we have that $(g_{s_1}\times g_{s_2})\neq 0$, $\forall s_1,s_2\in\mathbb{R}$, [9] , and the vectors g_{s_1}, g_{s_2} are linearly independent everywhere. Therefore, every tangent vector to the surface is a linear combination of the vectors g_{s_1} and g_{s_2}, (Fig. 2).

Also, we can define a normalized perpendicular vector to the surface as: $N=\frac{g_{s_1}\times g_{s_2}}{\|g_{s_1}\times g_{s_2}\|}$.

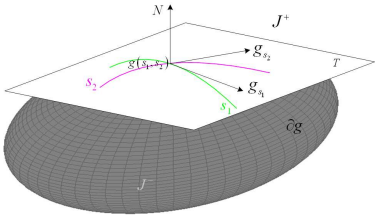

Fig. 2. Representation of tangent's and perpendicular's vectors, of a surface, variations w.r.t parameter's modification.

We now define the vector valued function

$$a(s_1,s_2)=g(s_1,s_2)+\rho\cdot N(s_1,s_2) \tag{5}$$

Following the same line of thought as the authors in [5], [6] have proved, we need the function $a(s_1,s_2)$, from (5) to be bijective.

This implies that locally, in order to satisfy the needed condition of the function $a(s_1,s_2)$, we must choose a $\rho>0$ such that, [6]:

$$\rho<\rho_m=\min_{s_1,s_2}|\frac{1}{\kappa_n}|=\frac{1}{\kappa^*} \tag{6}$$

This justifies the selection of the maximum curvature, so a non-negative ρ can always be defined.

The problem can now be stated as follows:
Given a robot manipulator, and a closed surface, find a feedback dynamic control law that stabilizes the end-effector of the manipulator over the given surface, while steering it across the surface

- *to navigate to any feasible surface point*
- *to track a predefined trajectory across the surface*

The environment is considered known, static and bounded.

III. NAVIGATION OF AN ARTICULATED NON-POINT ROBOT

A. A Potential Field for Non-Point Robots

For constructing a potential field it is necessary to represent mathematically the system and its environment. Most of previous works are based on the assumption that we can represent the system as a point in the workspace, [6]. To apply the methodology to an actual articulated robotic manipulator, though, we need to also consider the volume occupied by the manipulator. To this extend we have used the method proposed in [7] which extends the navigation function approach of [10] to articulated non-point robots using a series of diffeomorphic transformations.

According to [7], the shape of the robotic system R and the obstacles $\mathscr{O}$ in a 3-D workspace $\mathscr{W}\subset\mathbb{R}^3$ are considered as a unions of generalized n-ellipsoids: $\mathrm{R}=\cup_{j\in\mathscr{J}}\mathrm{R}_j$ and $\mathscr{O}=\cup_{i\in\mathscr{I}}\mathscr{O}_i$, where $\mathscr{J}=1,\dots,n_{\mathrm{R}}$ and $\mathscr{I}=1,\dots,n_{\mathscr{O}}$ the number of ellipsoids covering the volume of robot and obstacles, respectively.

In $\mathscr{W}$, a sequence of smooth transformations is used to create spaces where the robot and all obstacles are represented by points. Therefore, we have a workspace $\mathscr{W}^*$ where the robot and obstacles are represented by points. The following represents a measure of proximity of the robot to the obstacles in the transformed workspace:

$$\beta_{\mathscr{O}}\triangleq\prod_{j\in\mathscr{J}}\prod_{i\in\mathscr{I}}\|h^*_{\mathrm{R}_j}-h^*_{\mathscr{O}_i}\|^2 \tag{7}$$

where $h^*_{\mathrm{R}_j}$ is the position of the transformed robot part j in $\mathscr{W}^*$ and $h^*_{\mathscr{O}_i}$ the position of the transformed obstacle i in $\mathscr{W}^*$. For details on the construction, the reader is referred to [7].

B. Singularities Avoidance

An introduction of a number of virtual obstacles can be used in order to avoid the manipulator's singularities. Singularity regions $\mathscr{S}=\cup_{k\in\mathscr{K}}\mathscr{S}_k$ are sets of measure zero within the configuration space but their shape and location

depends on the mechanical structure and cannot be generally described for an arbitrary mechanism. Singularities can be considered as solutions of the equation: $det(J^T J) = 0$, with J the Jacobian of the robot.

In the case when singularity regions can be decoupled to classes that depends on a subset of the configuration variables, it is always feasible to enclose the singularity regions inside ellipsoids $\mathscr{O}_{s_k}$, representing virtual obstacles affecting the motion of the robot end-effector. Following the approach of [7], each ellipsoid $\mathscr{O}_{s_k}$ is reduced into a point $h^*_{s_k}$. Therefore, the singularity avoidance is achieved by introducing virtual obstacles:

$$\beta_s \triangleq \prod_{j\in\mathscr{J}} \prod_{k\in\mathscr{K}} \|h^*_{\mathbf{R}_j} - h^*_{s_k}\|^2 \tag{8}$$

C. Navigation Vector Field

Let us now construct a navigation function $\varphi : \mathscr{W}_{ws} \to \mathbb{R}$ according to [4], with the following form:

$$\varphi(q) = \frac{\gamma_d(q)}{\left(\gamma_d^{\kappa}(q) + \beta_{ws}(q)\cdot\beta_{\mathscr{O}}(q)\cdot\beta_s(q)\right)^{\frac{1}{\kappa}}} \tag{9}$$

where

$$\gamma_d(q) = \|k(q) - p_{d_1}\|^2 \tag{10}$$

is the distance to the goal function, $\mathscr{W}_{ws} \subset \mathscr{W}_{free}$ is the set where the task takes place, $p = k(q)$ is the end effector position, with $k(q)$ the manipulator kinematics. Also, $p_{d_1} = \begin{bmatrix}\mathbf{x}_{d_1} & \theta_{d_1}\end{bmatrix}^T$ is the target configuration for the navigation function, where $\mathbf{x}_{d_1}$, and θ_{d_1} are the desired position and orientation of the robot's end-effector. The function

$$\beta_0(q) = -\|k(q) - p_0\|^2 + r_0^2 \tag{11}$$

provides the workspace potential for the navigation in J^+ (i.e. $\beta_{ws} := \beta_0, k(q) \in J^+$), $\beta_{\mathscr{O}}(q)$ and $\beta_s(q)$ are given from (7)-(8), respectively, and $\kappa > 0$ is a parameter. Thus, for the navigation task in J^+, the vector field is given by $\nabla\varphi_\tau = \nabla\varphi_{|\beta_{ws}:=\beta_0}$.

The surface described by the function $g(s_1,s_2)$, defined in (2) is modeled in the navigation function as an obstacle for the robot's links.

IV. TASK SPECIFIC VECTOR FIELDS

A. Belt Zones

As discussed in the previous section, a navigation function $\nabla\varphi_\tau = \nabla\varphi_{|\beta_{ws}:=\beta_0}$ is implemented to drive the end-effector towards $g(s_1,s_2)$. This is achieved by placing the destination configuration, p_{d_1} in J^-. Since our goal is to perform motion tasks across the boundary of $g(s_1,s_2)$, when the robot's end-effector reaches a certain distance from the boundary of $g(s_1,s_2)$, a switch to an appropriate task (navigation or trajectory tracking) specific controller occurs.

To construct such a vector field we are going to use the concept of belt zones, [5], [6], which is represented in Fig. 3. The "belt zone" is the region close to the ∂g and is thought to be composed of an "internal belt" and an "external belt" region. For the motion tasks considered in this paper the widths of the internal and external belt regions are considered to be fixed.

Let us define the vector functions which are used to describe the belt zones to which we have referred above.

$$\beta(s_1,s_2) = g(s_1,s_2) + \delta\cdot N \tag{12}$$

$$\gamma(s_1,s_2) = \beta(s_1,s_2) + \delta\cdot N \tag{13}$$

with $0 < 2\cdot\delta < \rho_m$. Both surface processing tasks require stabilization of the end-effector on the surface $\beta(s_1,s_2)$, defined by (12).

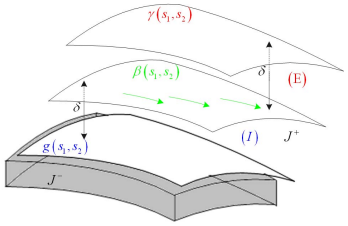

Fig. 3. Representation of Belt Zones, in a part of a surface.

Now we are in position to define the sets of "internal belt", $\mathscr{I}$, and the "external belt",$\mathscr{E}$ (Fig. 3).

$$\mathscr{I} = \{q : k(q) = (1-\lambda)\cdot\beta + \lambda\cdot g, \lambda \in [0,1]\} \tag{14}$$

$$\mathscr{E} = \{q : k(q) = (1-\lambda)\cdot\beta + \lambda\cdot\gamma, \lambda \in (0,1]\} \tag{15}$$

Since functions g,β,γ are bijective [6], [5], for every $k(q) \in \mathscr{E}\bigcup\mathscr{I}$ there is a unique couple (s_1,s_2).

B. Vector Field on a Surface

Now it is necessary to define the appropriate vector field, that will be used so that the robot end-effector navigates across the surface of interest to reach a specified destination point $p_{d_2} = \begin{bmatrix}\mathbf{x}_{d_2} & \theta_{d_2}\end{bmatrix}^T$.

To this extend we need to define a navigation function across the 2-D surface, that will provide the navigation vector field. Although theoretically a system that flows according to the tangent space of the 2-D, surface-wrapped navigation field, remains in that 2-D surface, various sources of uncertainty, like sensor noise, model uncertainties and numerical diffusion cause the system to deviate from this surface. To compensate for this problem, we designed an additional vector field perpendicular to the 2-D surface wrapped vector field, which attracts the system on the surface of interest. Such an attractive vector field is provided through an appropriate construction of the γ_d function and by introduction of an additional "perpendicular" workspace function that prohibits exiting the belt zone.

Assume that $h(x,y,z)$ is the distance from the surface $g(s_1,s_2)$ on the belt zones. For $h_0 = 0$ we have that the end-effector is on the surface defined by g (boundary of internal region), and for $h_{ext} = 2\cdot\delta$ we have that the end-effector is on the surface defined by γ (boundary of external region). Also, the desired distance from the surface $g(s_1,s_2)$ is at $h_d = \delta$,

that is, when the end-effector is in the surface $\beta(s_1,s_2)$. Thus, we can define the distance to the goal function as:

$$\gamma_d(q) = \left\| \begin{bmatrix} k(q) \\ h(k(q)) \end{bmatrix} - \begin{bmatrix} p_{d_2} \\ h_d \end{bmatrix} \right\|^2 \tag{16}$$

where the second term of the vectors is used to attract the end-effector to the surface β. Also the "perpendicular" workspace function is given from the equation:

$$\beta^{\perp}(q) = \frac{(h_{ext}-h_d)^2 - (h(q)-h_d)^2}{(h_{ext}-h_d)^2} \tag{17}$$

The function $\beta^{\perp}(q)$ in this case guarantees that the robot's end-effector cannot leave the belt zone, (Fig. 4). The workspace boundary for the navigation task is thus defined in both the 2-D workspace, where we usually place it to cover a "bad" region (incorporated in the $\beta_{\mathscr{O}}$ function) and in the "perpendicular" direction to prohibit exiting the belt zone. The vector field in this case is provided by $\nabla\varphi_{\tau} = \nabla\varphi_{|\beta_{ws}:=\beta^{\perp}}$ for the surface navigation task.

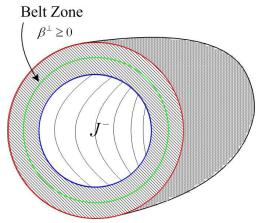

Fig. 4. Representation of the workspace obstacle function based on the Belt Zones construction.

V. CONTROL STRATEGY

We assume that we have a stationary environment and the robot manipulator can be described trivially by a fully actuated, second order dynamic model. In the workspace, the volume of the manipulator is represented by a point, using a series of transformations. The obstacles present in the environment are modeled by the navigation function. The goal is for the robot to be able to navigate using the navigation function constructed on a 3-D space, to move its end-effector across the surface $\beta(s_1,s_2)$ and perform a task on the surface, avoiding entering "bad" regions or colliding with obstacles.

A. Reaching a point on a surface

Assume that the robot's initial configuration is $q(0) \in \mathbb{R}^m$, with $p(0) = k(q(0)) \in J^+$, and we would like the end-effector move towards the surface, in order to reach a specific point on it (e.g. for the neuro-robotic set-up, a manipulator wants to place an object on a table, without breaking it).

We will consider the system as operating in two possible modes: mode Φ where $p \in \mathscr{W}_{ext}$, where $\mathscr{W}_{ext} = \{\mathscr{W}_{free} \bigcap J^+\} \setminus \{(\mathscr{E} - \delta\mathscr{E}) \bigcup \mathscr{I}\}$, with $p = k(q)$ and $\delta\mathscr{E} = \{q : k(q) = (1-\lambda)\cdot\beta + \lambda\cdot\gamma, \lambda \in (1-\varepsilon, 1]\}$, $\varepsilon > 0$ and mode $\mathscr{B}$ where $p \in \mathscr{E} \bigcup \mathscr{I}$. We define the following vector fields for each mode:

$$\begin{aligned} f_{\Phi}(q) &= -\nabla\varphi_{|\beta_{ws}:=\beta_0} \\ f_{\mathscr{B}}(q) &= -\nabla\varphi_{|\beta_{ws}:=\beta^{\perp}} \end{aligned} \tag{18}$$

The dynamic representation of the system is given from (1). Using the following equation

$$\tau = B(q)\cdot u + C(q,\dot{q}) + gr(q) \tag{19}$$

we have the linear representation of the system:

$$\ddot{q} = u \tag{20}$$

The convergence to the point on the surface is considered in a two step fashion: First a navigation controller brings the end effector in the belt zone and then a second controller takes over to navigate the system across the surface. We have the following:

***Proposition** 1:* Consider the system (20) and the control law:

$$u = -c\cdot(\dot{q} - f_i) + \frac{\partial f_i}{\partial q}\cdot\dot{q} + f_i \tag{21}$$

where c a positive tuning constant. For initial conditions in $\{W_{free} \bigcap J^+\} \setminus \{\mathscr{E} \bigcup \mathscr{I}\}$ and with the vector field as defined for $i = \Phi$, the system converges to the set $\{\mathscr{E} \bigcup \mathscr{I}\}$, a.e.[1]. When the system is in $\{\mathscr{E} \bigcup \mathscr{I}\}$, the vector field as defined for $i = \mathscr{B}$ is activated and the system converges globally asymptotically to the destination configuration, a.e.

Proof: The control law construction is inspired by the backstepping controller design proposed by [11].

Assume that the robot's initial conditions are in $\{W_{free} \bigcap J^+\} \setminus \{\mathscr{E} \bigcup \mathscr{I}\}$. Then the vector field is f_{Φ} from (18).

We form the Lyapunov function:

$$V(q,\dot{q}) = \varphi_{\Phi}(q) + \frac{1}{2}\cdot(\dot{q} - f_{\Phi}(q))^2 \tag{22}$$

where $\varphi_{\Phi} = \varphi_{|\beta_{ws}:=\beta_0}$ is given from (9). Taking the time derivative of the Lyapunov function:

$$\dot{V} = \nabla\varphi_{\Phi}\cdot\dot{q} + (\dot{q} - f_{\Phi})\cdot\left(u - \frac{\partial f_{\Phi}}{\partial q}\cdot\dot{q}\right)$$

Substituting the control law u as it is defined in (21), we have that:

$$\dot{V} = -\|\nabla\varphi_{\Phi}\|^2 - c\cdot(\dot{q} + \nabla\varphi_{\Phi})^2 \leq -\|\nabla\varphi_{\Phi}\|^2$$

Hence, LaSalle's Invariance Principle, [12] guarantees convergence of q to the largest invariant set contained in the set $\{q \mid \|\nabla\varphi_{\Phi}(q)\| = 0, \}$. The critical points of the navigation function are isolated, [4]. Thus the set of initial conditions that lead to saddle points are set of measure zero. Thus, the largest invariant set contained in the $\|\nabla\varphi_{\Phi}\| = 0$ consists of the target configuration p_{d_1} and the saddle points.

Since the robot's end-effector initial condition is $p_0 \in J^+$ and by construction it holds that $p_{d_1} \in J^-$, the solutions of (20), which are absolutely continuous, intersect the surface $g(s_1,s_2)$, using standard topological arguments. Therefore

[1]i.e. everywhere except a set of initial conditions of measure zero.

there exists finite time T for which the system enters the belt zones. When in the belt zone a mode switch occurs that activates mode $\mathscr{B}$ and the control law (21) changes to $u_{fi=f_{\mathscr{B}}}$. Once the robot end-effector enters the belt zone, it remain there as the boundaries of the belt zone are repulsive due to the construction of the workspace. Following the same procedure as for the mode Φ, by choosing a Lyapunov function of the form:

$$V(q,\dot{q}) = \varphi_{\mathscr{B}}(q) + \frac{1}{2}\cdot(\dot{q} - f_{\mathscr{B}}(q))^2$$

where $\varphi_{\mathscr{B}} = \varphi_{|\beta_{ws}:=\beta^{\perp}}$ it holds that the system converge to the target configuration p_{d_2} a.e. ■

B. Tracking

Now let us consider a tracking task across the surface $\beta(s_1,s_2)$. The task is described by a known trajectory $q_d(t)$ on it (e.g. for neuro-robotic set-up, a manipulator keep a pen and we would like it to write something on a paper). Let us now define the appropriate control law in order to track the predefined trajectory.

We introduce a navigation function of the form:

$$\varphi_{tr}(q,t) = \frac{\gamma_d(q,t)}{\left(\gamma_d^{\kappa}(q,t) + \beta^{\perp}(q)\cdot\beta_{\mathscr{O}}(q)\cdot\beta_s(q)\right)^{1/\kappa}} \qquad (23)$$

where γ_d is similar to (16), and represents the distance to the trajectory.

We consider convergence of the system to a small ball of radius $\varepsilon > 0$ containing the target.

To define the tracking controller we will use the P_1 region introduced in [13]. This set is used to identify sets of points that contain measure zero sets whose positive limit sets are saddle points:

$$P_1 \triangleq \left\{p : (\lambda_{\min} < 0) \wedge (\lambda_{\max} > 0) \wedge \left(\left|\hat{v}_{\lambda_{\min}}\cdot\nabla\varphi\right| \le \varepsilon_1\right)\right\}$$

with $\varepsilon_1 < \min_{C=\{p:\|p-p_d\|=\varepsilon\}} (\|\nabla\varphi(C)\|)$, where $\lambda_{\min}$, λ_{max} are the minimum and the maximum eigenvalues of the Hessian $\nabla^2\varphi$, and $\hat{v}_{\lambda_{\min}}$ the unit eigenvector corresponding to the minimum eigenvalue of the Hessian. If $\left|\hat{v}_{\lambda_{\min}}\cdot\nabla V\right| = 0$ then the set P_1 consists of the measure zero set of initial conditions that lead to saddle point, [13], [6]. In view of this, ε_1 can be chosen to be arbitrarily small so the sets defined by P_1 eventually consist of thin sets containing sets of initial conditions that lead to saddle points.

Now consider that the system is operating in tracking mode. Then the task specific vector field is

$$f^{tr} = -\nabla\varphi_{tr} - \frac{\partial\varphi_{tr}}{\partial t}\cdot\frac{\nabla\varphi_{tr}}{sw\left(\|\nabla\varphi_{tr}\|^2,\varepsilon^2\right) - \varepsilon^2\cdot\sigma\left(\frac{\partial\varphi_{tr}}{\partial t}\right)\cdot\sigma\left(\|\nabla\varphi_{tr}\|^2\right)}$$

with $\sigma(x) = \frac{x}{1+|x|}$, $sw(x,e) = \begin{cases} x, & x \ge e \\ e, & x < e \end{cases}$, and φ_{tr} is given from (23).

***Proposition** 2:* Consider the system (20) with initial conditions in $\{\mathscr{E}\bigcup\mathscr{I}\}\backslash P_1$. Then the control law

$$u = -c\cdot(\dot{q} - f^{tr}) + \left(\frac{\partial f^{tr}}{\partial q}\cdot\dot{q} + \frac{\partial f^{tr}}{\partial t}\right) - \nabla\varphi_{tr} \qquad (24)$$

converges to the set $\mathscr{T} = \{q : \|k(q) - p_d\| < \varepsilon\}$, a.e.

Proof: We form the Lyapunov function:

$$V(q,\dot{q}) = \varphi_{tr}(q) + \frac{1}{2}\cdot\left(\dot{q} - f^{tr}(q)\right)^2 \qquad (25)$$

and take its time derivative:

$$\dot{V} = \left(\frac{\partial\varphi_{tr}}{\partial t} + \nabla\varphi_{tr}\cdot\dot{q}\right) + (\dot{q} - f^{tr})\cdot\left(u - \frac{\partial f^{tr}}{\partial q}\cdot\dot{q} - \frac{\partial f^{tr}}{\partial t}\right)$$

Substituting the control law u as it is defined in (24), we have that:

$$\dot{V} = \left(\frac{\partial\varphi_{tr}}{\partial t} + \nabla\varphi_{tr}\cdot f^{tr}\right) - c\cdot(\dot{q} - f^{tr})^2 \Rightarrow$$
$$\dot{V} \le \frac{\partial\varphi_{tr}}{\partial t} + \nabla\varphi_{tr}\cdot f^{tr}$$

Assuming that

$$\dot{V}_{tr} = \frac{\partial\varphi_{tr}}{\partial t} + \nabla\varphi_{tr}\cdot f^{tr}$$

after substituting f^{tr} and since we pursue convergence in the set $\mathscr{T}$, we get:

$$\dot{V}_{tr} = -\|\nabla\varphi_{tr}\|^2 + \frac{\partial\varphi_{tr}}{\partial t}\cdot\left(1 - \frac{\|\nabla\varphi_{tr}\|^2}{\|\nabla\varphi_{tr}\|^2 - \varepsilon^2\cdot\sigma\left(\frac{\partial\varphi_{tr}}{\partial t}\right)\cdot\sigma\left(\|\nabla\varphi_{tr}\|^2\right)}\right)$$

We can now discriminate the following cases:

- $\frac{\partial\varphi_{tr}}{\partial t} = 0 \Rightarrow \dot{V}_{tr} = -\|\nabla\varphi_{tr}\|^2 \le 0$
- $\frac{\partial\varphi_{tr}}{\partial t} > 0 \Rightarrow 0 < \sigma(\frac{\partial\varphi_{tr}}{\partial t}) < 1 \Rightarrow \|\nabla\varphi_{tr}\|^2 - \varepsilon^2 < \|\nabla\varphi_{tr}\|^2 - \varepsilon^2\cdot\sigma\left(\frac{\partial\varphi_{tr}}{\partial t}\right)\cdot\sigma\left(\|\nabla\varphi_{tr}\|^2\right) < \|\nabla\varphi_{tr}\|^2 \Rightarrow \dot{V}_{tr} \le 0$
- $\frac{\partial\varphi_{tr}}{\partial t} < 0 \Rightarrow -1 < \sigma(\frac{\partial\varphi_{tr}}{\partial t}) < 0 \Rightarrow \|\nabla\varphi_{tr}\|^2 < \|\nabla\varphi_{tr}\|^2 - \varepsilon^2\cdot\sigma\left(\frac{\partial\varphi_{tr}}{\partial t}\right)\cdot\sigma\left(\|\nabla\varphi_{tr}\|^2\right) < \|\nabla\varphi_{tr}\|^2 + \varepsilon^2 \Rightarrow \dot{V}_{tr} \le 0$

Hence, LaSalle's Invariance Principle, guarantees convergence of q to the largest invariant set contained in the set $\{q \mid \|\nabla\varphi_{tr}\| \le \varepsilon_1\}$. We have assumed that the system's initial conditions are in the set $\{\mathscr{E}\bigcup\mathscr{I}\}\backslash P_1$. Since the set P_1 is repulsive, it holds that $\dot{V} \overset{a.e.}{<} 0$. ■

Remark 1: In practice we can choose an ε_1, such that $\varepsilon_1 < \min\{\varepsilon_0, \|\nabla\varphi_{tr}(q_0,t_0)\|\}$, so we can be sure that the system's initial conditions are not in $\mathscr{P}$.

VI. SIMULATION RESULTS

Computer simulations have been carried out to verify the feasibility and efficacy of the proposed methodology. The robot manipulator that we have used for the implementation of the simulations, is the model of PUMA-560 Unimate, with $m = 6$ d.o.f. The surface of interest $g(s_1,s_2)$, is an ellipsoid with $s_1 \in [0,2\pi)$ and $s_2 \in [\phi_0,\pi]$, [6], centered at the point $(0.0,0.0,0.0)$, with $(0.44,0.24,0.14)$ the lengths of the three semi-axes, respectively. The scenario of the simulation contains two 3-D (ellipsoid) obstacles centered at $\mathscr{O}_1 : (-0.4,0.5,0.2)$ and $\mathscr{O}_2 : (0.25,-0.5,-0.7)$. The "bad" region's obstacles are centered at $\mathscr{O}_{g1} : (-0.25,0.15,0.14)$, $\mathscr{O}_{g2} : (0.25,0.15,0.14)$ and $\mathscr{O}_{g3} : (-0.25,-0.15,-0.14)$. The robot manipulator's initial configuration was $p(0) = (0.26,-0.46,0.19,0.2,0.2,0)$, the first target was set at $p_{d_1} = (0,0,0,0,0,0)$ and the second target was set at $p_{d_2} = (0.25,-0.15,-0.14,1.07,0.44,-1.07)$. After the end-effector of the robot manipulator reaches its destination point

(p_{d_2}), it starts a tracking task, to track a predefined trajectory which is the yellow colored line in Fig. 5-8. Also, each of those figures depict the simulation results, from a different point of view, since there is a 3-D view simulation. Our algorithm successfully converges to the goal configuration and track the predefined trajectory avoiding obstacles. Also, in Fig. 7-8 is depicted clearly, that when the trajectory is passing through an obstacle, the robot can avoid it by leaving the trajectory until this is out of the obstacle again.

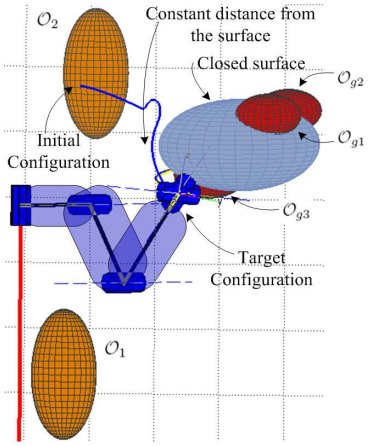

Fig. 5. Simulation results: reaching a point on the surface.

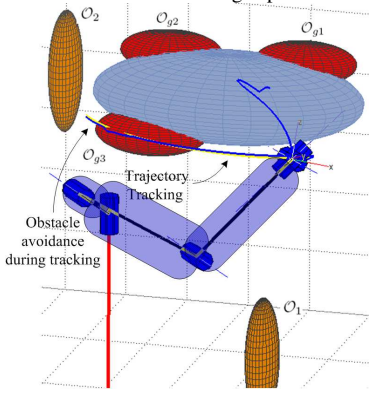

Fig. 6. Simulation results: reaching a point on the surface and tracking.

VII. CONCLUSIONS AND FUTURE WORKS

We presented a methodology for performing navigation and tracking tasks over a 2-dimensional manifold embedded in a 3-dimensional workspace applicable to articulated robotic manipulators. After safely navigating the manipulator's end-effector to the 2-D manifold, task specific vector fields direct the end-effector towards accomplishing a navigation or a trajectory tracking task across the 2-D manifold. The methodology has theoretically guaranteed global convergence and collision avoidance properties. Due to the closed form of the dynamic feedback controller, the methodology is particularly suitable for implementation on real time systems with limited computation capability.

Further research includes considering surface properties in the construction of the belt zone vector fields and implementing the methodology to real neuro-robotic systems taking into account their dynamics and kinematic constraints.

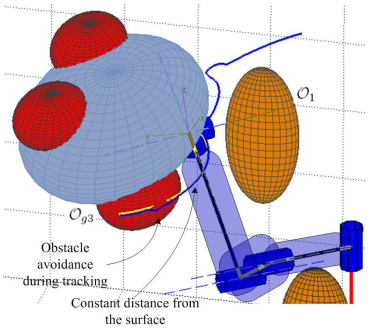

Fig. 7. Simulation results: collision avoidance during tracking.

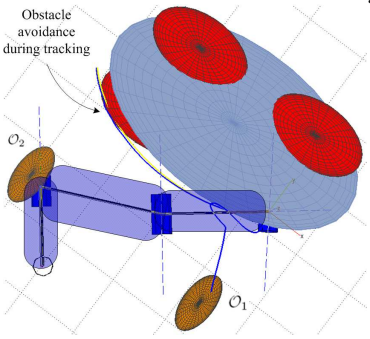

Fig. 8. Simulation results: top view, collision avoidance during tracking.

REFERENCES

[1] P. Atkar, D. Conner, A. Greenfield, H. Choset, and A. Rizzi, "Uniform coverage of simple surfaces embedded in $\mathbb{R}^3$ for auto-body painting." Carnegie Mellon University, 2004.

[2] P. Atkar, H. Choset, and A. Rizzi, "Towards optimal coverage of curve," *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, 2003.

[3] D. Conner, P. Atkar, A. Rizzi, and H. Choset, "Deposition modeling for paint application on surfaces embedded in $\mathbb{R}^3$," Carnegie Mellon University," Tech. Report, 2002.

[4] D. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances Appl. Math.*, vol. 11, pp. 412–442, 1990.

[5] S. Loizou, H. Tanner, V. Kumar, and K. Kyriakopoulos, "Closed loop motion planning and control for mobile robots in uncertain environmentss," *Proceedings of the 42nd IEEE Conference on Decision and Control*, 2003.

[6] X. Papageorgiou, S. Loizou, and K. Kyriakopoulos, "Motion planning and trajectory tracking on 2-D manifolds embedded in 3-D workspaces," *2005 IEEE International Conference on Robotics and Automation*, pp. 501–506, 2005.

[7] H. Tanner, S. Loizou, and K. Kyriakopoulos, "Nonholonomic navigation and control of cooperating mobile manipulators," *IEEE Trans. on Robotics and Automation*, vol. 19, no. 1, pp. 53–64, 2003.

[8] L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*. McGraw-Hill, 1996.

[9] W. M. Boothby, *An introduction to differentiable manifolds and Riemannian geometry*. Academic Press, 1986.

[10] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[11] M. Krstić, I. Kanellakopoulos, and P. Kokotović, *Nonlinear and Adaptive Control Design*. Wiley-Interscience, 1995.

[12] H. Khalil, *Nonlinear Systems*. Prentice-Hall, 1996.

[13] S. Loizou, D. Dimarogonas, and K. Kyriakopoulos, "Decentralized feedback stabilization of multiple nonholonomic agents," *2004 IEEE International Conference on Robotics and Automation*, 2004.