# Motion Design and Learning of Autonomous Robots Based on Primitives and Heuristic Cost-to-Go

Keyong Li Raffaello D'Andrea
likeyong@ieee.org rd28@cornell.edu
Dept. of Mechanical & Aerospace Engineering,
Cornell University,
Ithaca, NY 14853.

**Abstract**

The task of trajectory design of autonomous vehicles is typically two-fold. First, it needs to take into account the intrinsic dynamics of the vehicle, which are sometimes termed local constraints. Second, on a higher level, the designed trajectories must allow the vehicle to achieve some application-specific task. The specification of the task results in the so-called global constraints. Both of these two components of trajectory design are generally nontrivial problems, and very often, they are pursued as two parallel areas. When the results drawn from the two areas are applied in conjunction, the synthesis is usually somewhat arbitrary.

In this paper, we assume some optimal control strategy that addresses the vehicle dynamics is available as a set of motion primitives. The trajectories that achieve the task are determined solely through the primitives and do not reference the vehicle dynamics directly. For the higher level, we translate the task into a very special type of cost-to-go function, which is partially specified artificially, and partially determined by an admissibility condition imposed by the set of primitives. The optimality feature of the primitives is formally extended to the final trajectory design. We illustrate our result with the example of a mobile robot retrieving an object, which is an interesting problem of its own right. Both a direct design approach and a learning approach are presented.

## I. Introduction

Layering is necessary for efficient control design in most autonomous systems. It facilitates the sharing of behavior elements and accommodates the coexistence of multiple control paradigms. It also coincides with the understanding of our own intelligence as human beings. For an autonomous system to achieve a mission in the real world, its control design must take into account of both its own dynamics, sometimes referred to as the *local constraints*, and the environment, sometimes referred to as the *global constraints*. Two levels of control can then be distinguished naturally (see [18]) — the *motion primitive* level, which concentrates on the local constraints, and the *elementary move* level, which is to achieve simple tasks based on

Both authors gratefully acknowledge support from the Air Force under grant F49620-02-1-0388 and from the NSF under grant ECS-0329743.

the primitives. From a broader perspective, the elementary moves are only building blocks for higher level control, including strategic planning, cooperation, etc., hence the name. However, it is the higher one between the only two levels considered in this paper. So, this paper will use the term *move* or *task* instead of elementary move.

The control problems subject to either local constraints (primitive level) or global constraints (task level) alone are generally nontrivial. Results that concentrate on the local constraints of autonomous vehicles include, for example, [3], [5], [20], [10], [19], [22], [27], etc. Among these, the first three are on nonholonomic vehicles while the other four are on omnidirectional vehicles. Work that concentrate on global constraints include [14], [17], [24], [26], [28], etc. In [14], both the *roadmap* method and the *potential field* method are discussed. The other four examples listed here develop various techniques based on potential fields. Although techniques that solve the combined problem have also been proposed, it seems that they all have to trade-off between optimality and heavy computation. See [4], [7]-[9], [13]-[16], [23], etc. In particular, [16] discusses two different approaches — *dynamic programming iteration*, which finds the optimal solution but is computationally very expensive; and *rapidly-exploring random tree*, which involves less computation but does not ensure optimality. An excellent review of trajectory design in the presence of both local and global constraints is also provided by [16].

In this paper, we introduce a scheme for rigorously integrating the control designs of the primitive level and the task level, with emphases on preserving optimality, avoiding heavy on-line computation, and learning. We assume some optimal control strategy that addresses the vehicle dynamics is available as a set of motion primitives. For the higher level, we capture the characteristic of the desired move using a heuristic cost-to-go function. Based on this cost-to-go, we show that the trajectory design that preserves the optimality of the primitives (in a sense further explained in this paper) can be calculated with very little computation. A very interesting finding is: For the optimality feature of the primitives to be fully preserved, the heuristic cost-to-go function must satisfy certain conditions which depend on the low-level system dynamics through the set of primitives. These conditions help us narrow down the candidates.

In the particular case of an omnidirectional robot driven by acceleration, it turns out that these conditions may be combined with the existing artificial potential field-based motion planning techniques to produce the heuristic cost-to-go and the trajectory design. For studies of artificial potential fields, see [12], [17], [24], [26], etc. These motion planning techniques are computationally efficient. However, they concentrate mostly on the global constraints, and their synthesis with lower level control has been more or less arbitrary. Thus, in this particular case, our result provides a significant improvement for motion synthesis based on artificial potential fields. Moreover, we will present both a direct design approach and a learning approach. Their pros and cons will be discussed.

Section II formulates the problem of optimal trajectory design based on optimal motion primitives. Section III proves the main theorem of this paper, which shows how to compute the trajectory design based on a heuristic cost-to-go function, as well as the admissibility condition for the heuristic cost-to-go function. The next two sections apply the main theorem to trajectory design and learning of an omnidirectional robot driven by acceleration. In both sections, the general-task solutions of the omnidirectional robot are derived first; then they are illustrated with the more concrete task of the robot retrieving and object. Section IV takes a direct design approach. The heuristic cost-to-go is prescribed by a human expert for a subspace of the state space. Then the prescription is extended in closed form to the whole state space according to the admissibility condition. Section V discusses a learning approach. The robot is commanded to make attempts without a priori knowledge of how to achieve the task, but only knows what constitute failures. The robot then "learns" from the failures and construct the heuristic cost-to-go using a set of basis functions. Section VI concludes the paper and suggests future work.

## II. Problem formulation

Let the system model at the primitive level be

$$\dot{x} = g(x, u), \tag{1}$$

in which $x \in \mathbb{R}_x$ is the state, $u \in \mathbb{R}_u$ is the control, and $g(x, u)$ is Lipschitz in $x$ and differentiable to the second order with respect to $u$. Assume all the possible equilibria of this plant can be reached from arbitrary initial conditions.

Let the primitives be control laws of the form

$$u = f(x, d), \tag{2}$$

in which $f(x, d)$ is Lipschitz in $x$ and differentiable in $d$. The vector $d \in \mathbb{R}_d$ parameterizes the set of primitives. Assume the dimension of $\mathbb{R}_d$ equals that of $\mathbb{R}_u$, $f(x, d_1) \neq f(x, d_2)$ for any $x$ and $d_1 \neq d_2$, and $\partial f / \partial d$ is non-singular.

Further suppose each primitive is the solution of the optimization problem parameterized by $d$:

$$V(x(0), d) = \min_u \int_0^\infty [r(u, x) + q(x, d)]dt, \tag{3}$$

for arbitrary values of $x(0)$ and subject to (1) and (2). $V(x, d),\ x \in \mathbb{R}_x,\ d \in \mathbb{R}_d$, is the cost-to-go function for driving the system from $x$ to some desired state that depends on $d$. Assume $r(u, x), q(x, d) \geq 0$ for all $u, x$ and $d$, equalities are achieved only when $x$ is at the desired equilibrium state and $u$ renders no motion in $x$. Also assume $r(u, x)$ is differentiable to the second-order with respect to $u$. Given the functions $r(u, x)$ and $q(x, d)$, the optimal

control $u = f(x, d)$ and cost-to-go $V(x, d)$ satisfy the *principle of dynamic programming* (see [6]):

$$\begin{cases} f(x,d) = \underset{u}{\text{argmin}}\{J(x,u,d)\} \\ 0 = J(x, f(x,d), d), \end{cases} \tag{4}$$

in which

$$\begin{aligned} J(x,u,d) &= r(u,x) + q(x,d) + \tfrac{\partial V(x,d)}{\partial x}\ \dot{x} \\ &= r(u,x) + q(x,d) + \tfrac{\partial V(x,d)}{\partial x}\ g(x,u). \end{aligned}$$

By assuming the range of $u$ is the whole $\mathbb{R}_u$, it is implied that $u = f(x, d)$ is a regular optimal solution of (4), thus

$$\left.\frac{\partial J(x,u)}{\partial u}\right|_{u=f(x,d)} = (\frac{\partial r}{\partial u} + \frac{\partial V(x,d)}{\partial x}\ \frac{\partial g}{\partial u}) = 0. \tag{5}$$

We further assume for simplicity that

$$\left.\frac{\partial^2 J(x,u)}{\partial u^2}\right|_{u=f(x,d)} = \frac{\partial}{\partial u}(\frac{\partial r}{\partial u} + \frac{\partial V}{\partial x}\frac{\partial g}{\partial u}) \qquad \text{is positive definite.} \tag{6}$$

Note that this assumption is introduced only to simplify the exposition; the results in this paper do not depend on it.

For the task level, the question is to choose a control law for $d$,

$$d = h(x),$$

such that $x$ goes to a desired state while avoiding the global constraints, and in an optimal manner in some sense.

In association with the primitives, one may formulate the question as the optimization:

$$U(x(0)) = \min_{d} \int_0^\infty [r(f(x,d),x) + l(x)]dt, \tag{7}$$

subject to both the dynamics of the system given by (1) and (2) and the global constraints, assuming a unique optimal solution does exist. Note that the cost term on control effort is the same as in the primitives. The function $l(x)$ is yet to be chosen.

One may specify the function $l(x)$ intuitively and solve the optimization problem (7) using numerical dynamic programming iterations (see [6] and [16]), which would explore the feasible state space either by simulation — if reliable model of the global constraints is available — or by physical experiments. Because such design is based on the intuitive choice of $l(x)$, tuning is usually needed. But $l(\cdot)$ is not a good tuning knob, because evaluating the effect of changing $l(\cdot)$ requires running dynamic programming iterations or similar algorithms, which is computationally costly.

Here, we do not specify $l(\cdot)$ directly. Instead, we let it inherit from the primitive design to take the form

$$l(x) = q(x, d^*), \tag{8}$$

in which $d^*$ equals the optimal solution of $d$. Note that given the desired move, $d^*$ is fixed although unknown.

Up to this point, the trajectory design still cannot be determined, because the specification of the task has yet to play its role. In this paper, we introduce the specifications of the tasks through cost functions instead of considering the global constraints explicitly. Since $l(x)$ does not serve as a good tuning knob, we try an alternative one: $U(x)$. Indeed, if we somehow know the optimal cost-to-go function $U(x),\ x \in \mathbb{R}_x$, then the optimal trajectory design can be determined with relatively little amount of computation. In general, the true cost-to-go is determined by the dynamics of the vehicle, the global constraints, and the nominal context of the task as well. Of course, we usually do not know the true cost-to-go easily. However, as will be shown in this paper, heuristic cost-to-go functions can be constructed, tuned, and used to produce desirable trajectories.

## III. Trajectory design based on motion primitives and heuristic cost-to-go

Continuing from the formulation of the last section, the following theorem provides both a way to calculate the optimal primitive-based control law, $d = h(x)$, and the admissibility condition for the heuristic cost-to-go, $U(x)$. Without loss of generality, we assume the destination is $x = 0$.

*Theorem 1:* Assume the optimization (7) subject to (1), (2) and the global constraints has a unique solution. If $U(x),\ x \in \mathbb{R}_x,\ U(0) = 0$, is the optimal cost-to-go subject to the robot dynamics and the global constraints, then the solution of $d$ of

$$\begin{cases} (\dfrac{\partial U(x)}{\partial x} - \dfrac{\partial V(x,d)}{\partial x})\ \dfrac{\partial g(x,u)}{\partial u}\Big|_{u=f(x,d)} & = 0 \\ (\dfrac{\partial U(x)}{\partial x} - \dfrac{\partial V(x,d)}{\partial x}) g(x, f(x,d)) & = 0 \end{cases} \tag{9}$$

exists. Further, if $d = h(x)$ is the solution of

$$\frac{\partial U(x)}{\partial x} = \frac{\partial V(x,d)}{\partial x}, \tag{10}$$

and the solution of (9) is unique (thus equals $h(x)$), then $d = h(x),\ x \in \mathbb{R}_x$ is the optimal control law.

*Proof:* For the first part of the theorem, if $U(x)$ is the cost-to-go function, with $d^*$ being the optimal solution, then from the principle of dynamic programming, $d^*$ and $U(x)$ satisfy

$$\begin{cases} d^* = \arg\min\limits_{d} L(x,d) \\ 0 = L(x, d^*), \end{cases} \tag{11}$$

in which

$$L(x,d) = r(f(x,d), x) + q(x, d^*) + \frac{\partial U}{\partial x} g(x, f(x,d)).$$

By the construction of the problem, $d^*$ is a regular solution. So

$$\frac{\partial L(x,d)}{\partial d} = \left[ (\frac{\partial r}{\partial u} + \frac{\partial U}{\partial x} \ \frac{\partial g}{\partial u}) \frac{\partial f}{\partial d} \right] \Big|_{u=f(x,d)} = 0. \tag{12}$$

Because $\partial f/\partial d$ is non-singular, (12) is equivalent to

$$(\frac{\partial r}{\partial u} + \frac{\partial U}{\partial x} \ \frac{\partial g}{\partial u}) \Big|_{u=f(x,d)} = 0. \tag{13}$$

Comparing (13) and (5) gives the first equation in (9). The second equation in (9) derives from comparing the second equation in (11) with the second equation in (4).

For the second part, we have a control law $d = h(x)$ and cost-to-go function $U(x)$ that satisfy (10) and $U(0) = 0$. What we wish to show is that $d^* = h(x)$ satisfies (11). The optimality of the control law then follows from the principle of dynamic programming.

To prove the first equation in (11), first notice that by satisfying (10), $d = h(x)$ is guaranteed a solution of (9). By the hypothesis in the theorem, it is the unique solution. Recalling equation (5) and that $\partial f/\partial d$ is non-singular, the first equation in (9) is equivalent to (12). Thus $d = h(x)$ is the unique solution of

$$\frac{\partial L(x,d)}{\partial d} = 0.$$

(We have followed the proof of the first part in the reverse direction.) It can also be verified using (6) that $\partial^2 L/\partial d^2$ is positive definite for $d = h(x)$. So, $d = h(x)$ minimizes $L(x,d)$ for any given $x$. The second equation of (11) follows directly from the second equation of (9) and the second equation of (4). The second part of the theorem is proved. ■

**Remarks**:

- A significant contribution of Theorem 1 is that it reveals how much freedom one may have in choosing the heuristic cost-to-go function. Given a set of primitives, Theorem 1 shows that certain choices of the function $U(x)$ may not be eligible for being the optimal cost-to-go, namely, those that fail to satisfy (9). Thus, the set of primitives imposes some admissibility conditions for the heuristic cost-to-go function.
- Despite the admissibility conditions, there may still be plenty of freedom in choosing the heuristic cost-to-go. This appears to be the case in the application discussed later in this paper. The second part of the theorem suggests a sufficient condition of admissibility that is relatively simple to verify. The corresponding optimal control law is also provided by the theorem. We find this sufficient condition particularly useful to the application that we are considering, because — as will be discussed in what follows — it is satisfied by heuristic cost-to-go functions that result in highly desirable motions.
- The trajectory design based on the primitives does not need to know the functions $g, f, r$, and $q$. It only needs to know the cost-to-go function $V$ of the primitive level. Thus, the low level system behavior is encapsulated by the primitives.

In the next two sections, we demonstrate two approaches (among other possible ones) with which the above theorem can be put to work. The problem considered have practical value of its own. In Section IV, the heuristic cost-to-to function is specified directly based on human expert intuition. The specification accommodates a few parameters, which can be tuned by trial-and-error, either automated or by hand. We will not use the word "learning" to describe this tuning process because the dimensionality of the parameter space is so low that little attention is needed in choosing a learning technique. Learning, however, is the subject of Section V, in which the heuristic function is not specified directly. Instead, the robot would make blind attempts, recognizing failures only after they have happened. Then, based on the list of recorded failure states, the robot would build the heuristic cost-to-go function using a set of basis functions that are not specific to any particular task.

## IV. The case of an omnidirectional vehicle driven by acceleration: direct specification of heuristic cost-to-go.

In what follows, superscript $^T$ indicates transpose. For a function $F(x),\ x \in \mathbb{R}^n$, we may alternate the notation between $F(x)$ and $F(x_1, x_2, \ldots, x_n)$.

Consider an omnidirectional robot described by

$$\dot{x}_1 = x_3,\ \dot{x}_2 = x_4,\ \dot{x}_3 = u_1,\ \dot{x}_4 = u_2, \tag{14}$$

in which $x_1,\ x_2$ are position coordinates in the global frame and $x_3,\ x_4$ are velocity components. Written in matrix form,

$$\dot{x} = g(x, u) = Ax + Bu, \tag{15}$$

in which

$$A = \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix}, \quad B = \begin{bmatrix} 0_{2\times2} \\ I_{2\times2} \end{bmatrix}. \tag{16}$$

Note that the actual dynamics of an omnidirectional robot is usually more complicated than this model. In our lab, we use a custom designed four-wheel omnidirectional robot. The drive system (including the motors, gears, tires, etc.) has its own dynamic states; the weight distribution of the robot may also shift during motion, causing certain wheels to lose traction. There is an additional lower hierarchy (an inner loop) of control, which incorporates wheel encoder and gyroscope feedback, that compensates for the aforementioned effects. It is because of this encapsulated control loop that the robot model appears to be the simple one described by (14). (See [10], [22], and [25].) This hierarchical approach is natural and essential for the design process. The design of the inner loop preserves almost the full capacity provided by the hardware. In addition to showing how this inner loop helps to reduce the dynamics of an omnidirectional robot to the simple linearized form, [10] and [22] respectively have created very high performing trajectory generation results using this inner loop in contexts that are

different from the present paper. Without the separation of hierarchies, those problems would have been hardly tractable.

Suppose the primitives based on the model (14) are LQR control laws for the robot to reach different destinations with zero terminal velocity. Let the parameter of the primitive be the destination, $d = [d_1,\ d_2]^T$. Denote $[d_1,\ d_2,\ 0,\ 0]^T$ by $\bar{d}$. The cost-to-go for each primitive is

$$\begin{aligned} & V(x(0), d) \\ & = \min_u \int_0^\infty \frac{1}{2}[u^T R u + (x - \bar{d})^T Q (x - \bar{d})] dt \\ & = \frac{1}{2}(x(0) - \bar{d})^T P (x(0) - \bar{d}), \end{aligned} \tag{17}$$

in which $R,\ Q$, and $P$ are symmetric matrices. $R$ and $P$ are positive definite and $Q$ is positive semidefinite. $P$ is determined by the algebraic Riccati equation

$$Q + PA + A^T P - PBR^{-1}B^T P = 0.$$

The corresponding optimal control law is

$$u = f(x, d) = -R^{-1}B^T P(x - \bar{d}).$$

Next, consider the higher-level trajectory design. Following the formulation introduced in the last section, the question is to find a function $h(x)$ such that $d = h(x)$ is the optimal solution of (denoting $[h^T(x) \quad 0_{1\times 2}]^T$ by $\bar{h}(x)$)

$$\begin{aligned} U(x(0)) = \min_d \int_0^\infty \frac{1}{2}[ & f^T(x, d) R f(x, d) \\ & + (x - \bar{h}(x))^T Q (x - \bar{h}(x))] dt, \end{aligned} \tag{18}$$

and $U(x)$ turns out to equal a heuristic function partially specified beforehand. We will solve this problem using Theorem 1.

First, it is straightforward to verify for this case that equation (9) in Theorem 1 has a unique solution of $d$ given an arbitrary choice of the function $U(x)$. So, the second part of Theorem 1 can be applied and we will consider the pairs of functions $U(x)$ and $h(x)$ that satisfies (10). Here, equation (10) has become

$$\frac{\partial U(x)}{\partial x} = \frac{\partial V(x, d)}{\partial x} = (x - \bar{d})^T P. \tag{19}$$

Let $\hat{U} = U - \frac{1}{2}x^T P x$, then

$$\frac{\partial \hat{U}}{\partial x} = -\bar{d}^T P \Longleftrightarrow -P^{-1}(\frac{\partial \hat{U}}{\partial x})^T = \bar{d}. \tag{20}$$

Write $P$ in block form,

$$P = \begin{bmatrix} P_A & P_B \\ P_C & P_D \end{bmatrix},$$

in which $P_A,\ P_B,\ P_C$, and $P_D$ are $2 \times 2$ matrices, and $P_B = P_C^T$. Then

$$P^{-1} = \begin{bmatrix} (P_A - P_B P_D^{-1} P_C)^{-1}[I_{2\times 2} & -P_B P_D^{-1}] \\ (P_D - P_C P_A^{-1} P_B)^{-1}[-P_C P_A^{-1} & I_{2\times 2}] \end{bmatrix}.$$

Equation (20) becomes

$$\begin{cases} (P_A - P_B P_D^{-1} P_C)^{-1}[-I_{2\times 2} \quad P_B P_D^{-1}](\dfrac{\partial \hat{U}}{\partial x})^T = d \\ (P_D - P_C P_A^{-1} P_B)^{-1}[-P_C P_A^{-1} \quad I_{2\times 2}](\dfrac{\partial \hat{U}}{\partial x})^T = 0 \end{cases} \tag{21}$$

Next, we introduce the artificial portion of the specification of $U$. Let

$$U(x_1, x_2, 0, 0) = U_0(x_1, x_2). \tag{22}$$

That is, $U(x)$ is specified for all zero-velocity states. From (21) and (22), $\hat{U}$ must satisfy

$$\begin{cases} \hat{U}(x_1, x_2, 0, 0) = U_0(x_1, x_2) - \dfrac{1}{2}[x_1,\ x_2] P_A [x_1,\ x_2]^T \\ [-P_C P_A^{-1} \qquad I_{2\times 2}](\dfrac{\partial \hat{U}}{\partial x})^T = 0. \end{cases} \tag{23}$$

This is a set of first-order linear PDEs in $\mathbb{R}^4$, with two equations and a boundary condition specified in a $\mathbb{R}^2$. It has a unique solution

$$\begin{aligned} \hat{U} = U_0([I_{2\times 2} \quad P_A^{-1} P_B]x) \\ -\frac{1}{2} x^T \begin{bmatrix} P_A & P_B \\ P_C & P_C P_A^{-1} P_B \end{bmatrix} x. \end{aligned} \tag{24}$$

See [21] for the procedure of solving first-order quasilinear PDEs. It then follows from the first equation in (21) with straightforward calculation that

$$\begin{aligned} d^* &= h(x) \\ &= [I_{2\times 2} \quad P_A^{-1} P_B]x - P_A^{-1}[\frac{\partial U_0(\xi_1, \xi_2)}{\partial(\xi_1, \xi_2)}]^T \end{aligned} \tag{25}$$

evaluated at

$$\begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = [I_{2\times 2} \quad P_A^{-1} P_B]x.$$

This is the optimal trajectory design based on the LQR primitives. The corresponding cost-to-go is

$$\begin{aligned} U(x) = U_0([I_{2\times 2} \quad P_A^{-1} P_B]x) \\ +\frac{1}{2}[x_3,\ x_4](P_D - P_C P_A^{-1} P_B)[x_3,\ x_4]^T. \end{aligned} \tag{26}$$

By specifying the function $U_0(x_1, x_2)$ differently, the above result can be applied to achieve a wide range of tasks. It is worth noting that there has been a considerable body of literature on motion planning of autonomous vehicles based on artificial potential field, which is

usually specified as a function of vehicle position. See for example [26]. Although some authors considered generalized potential field, which also depends on vehicle velocity, such dependency has been somewhat arbitrary. Combining with the result, the existing techniques on designing position-dependent artificial potential field may suggest good candidates for $U_0$, and the velocity dependency of the heuristic cost-to-go will be generated automatically. Moreover, the corresponding vehicle trajectories are optimal in a sense that accounts for both heuristic knowledge and the lower-level vehicle dynamics[1]. Thus, the solution to this example problem is itself a nice complement to existing techniques.

### *A. An Omnidirectional robot retrieving an object*

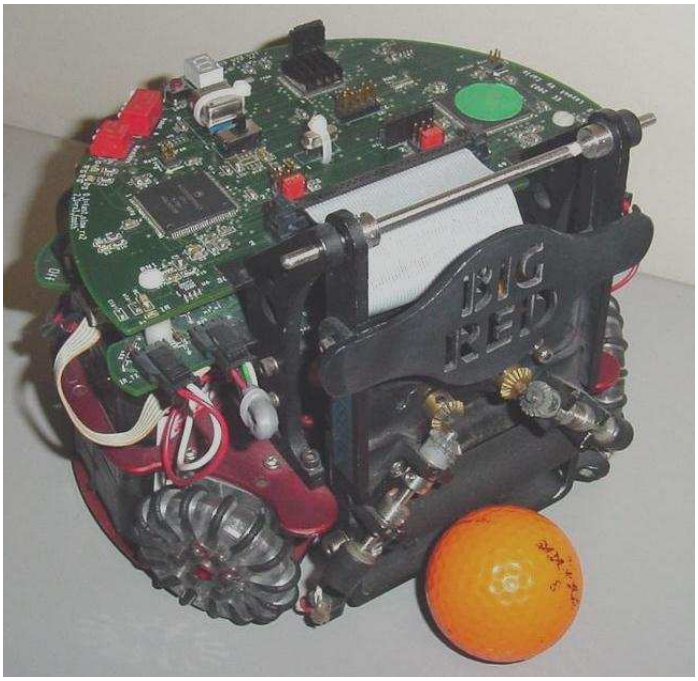


Fig. 1. An omnidirectional robot with a ball.

The task treated in this section is the following: Consider the omnidirectional mobile robot whose motion dynamics are described by (14). In addition, suppose the footprint of the robot is a circular disk of radius $r_0$, with $(x_1, x_2)$ denoting the center of the robot in the global frame. Without loss of generality, suppose a point object is placed at $(r_1, 0),\ r_1 > r_0$, and the task of the robot is to start from arbitrary initial position and reach the point $(0, 0)$ without touching the object — then the robot can push the object along the $x_1$-axis, which we assume leads to where the object is needed. Note that this task involves both going around the object (obstacle

[1]Although only second-order omnidirectional dynamics is considered here, extensions to other types of vehicle dynamics including nonholonomic cases also look promising.

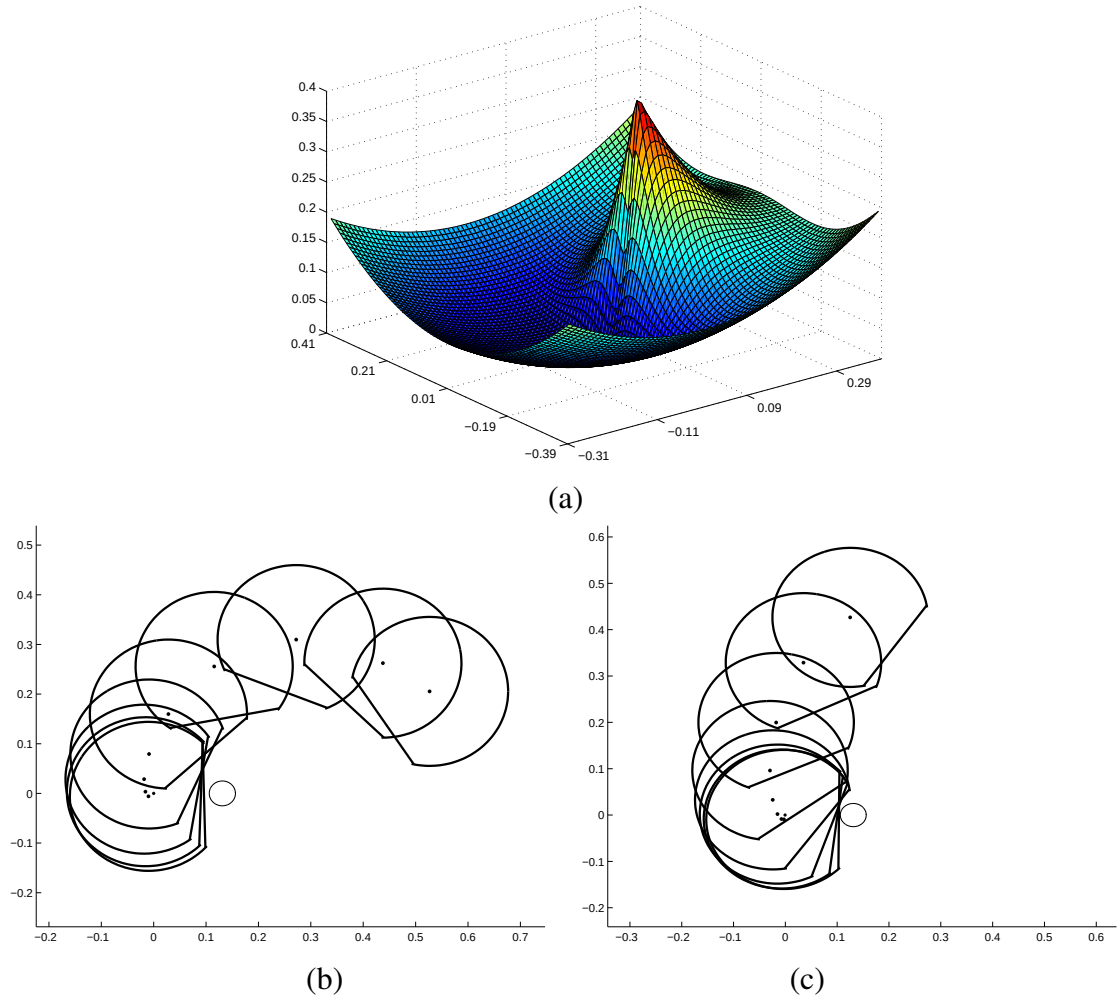


Fig. 2. (a) The function $U_0$. The peak is located at where the object is. (b) The designed robot trajectory from one initial position. (c) The designed robot trajectory from another initial position.

avoidance) and approaching it in a prescribed direction. We have applied our trajectory design technique to this problem in both simulation and a physical setting. Figure 1 shows one of our actual robots.

The set of LQR primitives has been designed in advance. All we need to know is the matrix $P$ from the primitive design and the function $U_0(x_1, x_2)$, which specifies the artificial portion of the heuristic cost-to-go. We chose $U_0$ as the following:

$$U_0(x_1, x_2) = \begin{cases} k(1 - \dfrac{\gamma(x_1, x_2)}{\sigma(x_1, x_2)})^3 + \dfrac{1}{2}[x_1,\ x_2]P_A[x_1,\ x_2]^T \\ \qquad\qquad \text{if} \quad \gamma(x_1, x_2) < \sigma(x_1, x_2) \\ \dfrac{1}{2}[x_1,\ x_2]P_A[x_1,\ x_2]^T \quad : \quad \text{otherwise}, \end{cases} \tag{27}$$

in which

$$\gamma(x_1, x_2) = \sqrt{(x_1 - r_1)^2 + x_2^2},$$

and

$$\sigma(x_1, x_2) = (1 + \frac{x_1 - r_1}{\gamma(x_1, x_2)})\frac{r_2}{2} + r_0.$$

The matrix $P_A$ is the first $2\times2$ block in $P$. Recalling that $r_0$ and $r_1$ are given in the description of the task, the two additional scalars $r_2 > r_1$ (with unit of length) and $k > 0$ (without unit) are tunable parameters. See Figure 2, (a) for the shape of the function $U_0(x_1, x_2)$. The function $U_0$ picked here has a singular point at $(r_1, 0)$, where the object itself is located. But since the pair $(\xi_1,\ \xi_2)$ in equation (25) never reaches that point, this does not cause any real problem. $U_0(x_1, x_2)$ is continuously differentiable elsewhere, including when $\gamma(x_1, x_2) = \sigma(x_1, x_2)$. Because the task involves both avoiding the object and approaching it in a prescribed direction, $U_0$ is set such that given the same distance to the object, the heuristic cost-to-go varies with the direction from the robot to the object. The function $\gamma(x_1, x_2)$ is the scaling factor for this purpose.

In our actual experiment, the matrices used in the LQR design of primitives are

$$Q = \text{diag}\{1.0,\ 1.0,\ 0.4\ 0.4\}, \quad R = I_{2\times2}.$$

Consequently,

$$P = \begin{bmatrix} 1.55 & 0 & 1.0 & 0 \\ 0 & 1.55 & 0 & 1.0 \\ 1.0 & 0 & 1.55 & 0 \\ 0 & 1.0 & 0 & 1.55 \end{bmatrix}.$$

All lengths are in meters. The effective radius[2] of the robot is $r_0 = 0.09$, and $r_1$ is set to 0.1. The tunable parameters are set to $k = 0.4$ and $r_2 = 0.8$.

Since both $P$ and $U_0(x_1, x_2)$ are available, the primitive-based trajectory design can then be computed from (25). (The heuristic cost-to-go $U(x)$ does not need to be computed explicitly.) See Figure 2, (b) and (c) for this result in action. Video clips of the simulation and the demonstration with an actual robot have been posted on the web [1]. In what the videos show, the orientation of the robot is controlled by a separate loop such that the robot always approximately face the ball. The main reason is to keep the partial heuristic cost-to-go specification in 2D, in another word, to keep the domain of $U_0$ in $\mathbb{R}^2$. Note that the mathematical procedure described in this section can be easily extended to systems with state spaces of higher dimensionality. However, the dimensionality of the domain of $U_0$ rises in proportion to the dimensionality of the full state space, which makes it harder (but not impossible) for human intuition to handle.

[2]The actual shape of our robot is mostly cylindrical, with radius of 0.15, but with a flat facet 0.09 from the center. For the experiment discussed in this section, the orientation of the robot is controlled separately such that the robot always face the object with its flat facet.

## V. LEARNING FROM FAILURES USING BASIS FUNCTIONS

We still consider omnidirectional robots in this section, but with the control of orientation integrated with the control of position. Although the rotation and translation of an omnidirectional robot are decoupled, the desirable trajectory of the robot position and that of the orientation may still be coupled because of the possible anisotropical shape of the robot combined with the presence of obstacles. The robot used in the demonstration is mostly cylindrical but has a flat "front". So, integrating the control of orientation does make sense. Later in this section, we will discuss the result of integrating orientation control as compared with the robot trajectories presented in the previous section.

The construct of the problem at hand is very similar to that of the last section, but we iterate the formulation briefly for clarity. The system considered is

$$\dot{x}_1 = x_4,\ \dot{x}_2 = x_5,\ \dot{x}_3 = x_6,\ \dot{x}_4 = u_1,\ \dot{x}_5 = u_2,\ \dot{x}_6 = u_3, \tag{28}$$

in which $x_1$ and $x_2$ are position coordinates and $x_3$ is the orientation, all in the global frame. Written in matrix form,

$$\dot{x} = Ax + Bu, \tag{29}$$

in which

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix}, \quad B = \begin{bmatrix} 0_{3\times3} \\ I_{3\times3} \end{bmatrix}. \tag{30}$$

The primitives are still LQR control laws, parameterized by $d = [d_1,\ d_2,\ d_3]^T$. Denote $[d_1,\ d_2,\ d_3,\ 0,\ 0,\ 0]^T$ by $\bar{d}$. The cost-to-go for a primitive is

$$\begin{aligned} &V(x(0), d) \\ &= \min_u \int_0^\infty \frac{1}{2}[u^T R u + (x - \bar{d})^T Q (x - \bar{d})]dt \\ &= \frac{1}{2}(x(0) - \bar{d})^T P (x(0) - \bar{d}). \end{aligned} \tag{31}$$

In block form,

$$P = \begin{bmatrix} P_A & P_B \\ P_C & P_D \end{bmatrix},$$

in which $P_A,\ P_B,\ P_C$, and $P_D$ are $3 \times 3$ matrices, and $P_B = P_C^T$. Again, by an application of Theorem 1, we consider pairs of $U(x)$ and $d = h(x),\ x \in \mathbb{R}^3$, that satisfies (10), which has become

$$\frac{\partial U(x)}{\partial x} = (x - \bar{d})^T P. \tag{32}$$

We seek to construct $U(x)$ such that the robot achieves the task in a desirable manner under the task-level control law $d = h(x)$. In this section, $U$ is constructed through learning with a set of basis functions that are not specific to particular tasks.

*A. The learning procedure*

Suppose no prior knowledge as to how to achieve the task is available. Let the initial heuristic cost-to-go be $U^{(0)}(x) = V(x, \mathbf{0}),\ x \in \mathbb{R}_x$. The task-level control law is then $d \equiv \mathbf{0}$. I.e., the robot will head for its terminal destination as if no task-specific constraint (such as an obstacle) is present. Surely, the attempts are likely to fail in the beginning. Assume that the robot can recognize a failure when it has occurred. For example, for navigating in a field with obstacles, colliding with an obstacle would be a failure; in the object-retrieval case, touching the object before settling into the terminal position means failure. It is important to note that the robot may detect such failures without knowing the geometry of the obstacle or the object involved, neither needs it to know its own shape. When colliding with an obstacle, the force of interaction will change the course that the robot is moving; when touching an object to be retrieved, the course of the object will change. When failure occurs, the robot records the corresponding state value, then take a randomized step away from the failure and start another attempt. Sometimes, the robot may reach the terminal destination simply by the randomness of initial condition. The robot will then go to a random position and start the next attempt. Denote the recorded failure states by $\eta^{(i)},\ i = 1, \ldots, N$. The heuristic cost-to-go is then updated as a function of the failure record such that the corresponding control law will drive the robot to achieve the task. Learning ends when the robot executes the task with a prescribed statistical success rate. Note that successfully executing a task with absolute certainty is almost impossible in reality because of noise and disturbance.

*B. Constructing heuristic cost-to-go with basis functions*

Given a list of failure states $\eta_i,\ i = 1, \ldots, N$, consider a heuristic cost-to-go of the form

$$U^{(N)}(x) = \frac{K(N)}{N} \sum_{i=1}^{N} \psi(x, \eta_i) + V(x, \mathbf{0}), \tag{33}$$

in which $K(N)$ is a scaling factor that increases with $N$ but remains bounded for $N \to \infty$; $\psi(\cdot, \eta) \geq 0$ is the basis function corresponding to the failure state $\eta$. The reason why $K(N)$ must be bounded is practical. Due to noise and disturbance in reality, rarely can any control law completely avoid failure. If $K(N)$ does not remain bounded, $U^{(N)}$ may explode even after a good control law has been found.

Recalling that $x = \mathbf{0}$ is assumed to be the desired terminal state, $U^{(N)}(\mathbf{0}) = V(\mathbf{0}, \mathbf{0}) = 0$ must always hold. So, the basis function must satisfy

$$\psi(\mathbf{0}, \eta) \equiv 0, \quad \eta \in \mathbb{R}_x. \tag{34}$$

In addition, the basis function must be chosen such that the equation (32) has solution of $d$. With these in mind, we propose the following basis function. Let

$$W = \text{diag}\{I_{3\times 3}, 0_{3\times 3}\},$$
$$c(\eta) = \sqrt{\eta^T P W P \eta}, \quad \eta \in \mathbb{R}_x,$$

and

$$\sigma(\eta) = \frac{2\hat{\sigma}}{\pi} \arctan\left[\frac{\pi}{2\hat{\sigma}} c(\eta)\right], \quad \eta \in \mathbb{R}_x,$$

in which $\hat{\sigma}$ is a tunable constant. We then let

$$\psi(x,\eta) = \begin{cases} \frac{1}{2}[\cos\left[\frac{\pi}{\sigma\eta} c(x-\eta)\right] + 1] & : \ c(x-\eta) < \sigma(\eta) \\ 0 & : \ \text{otherwise.} \end{cases} \tag{35}$$

In addition, we choose

$$K(N) = \frac{2K_a}{\pi} \arctan\left[\frac{K_b \pi}{2K_a} N\right],$$

in which the constant $K_a$ determines the asymptotic value of $K(N),\ N \to \infty$, and $K_b$ determines the slope of $K(N)$ at $N = 0$.

The functions $\sigma(\eta),\ \psi(x,\eta)$, and $K(N)$ are plotted in Figure 3. Note that the first two functions are plotted against the scalar functions $c(\eta)$ and $c(x-\eta)/\sigma(\eta)$ respectively. The constant $\hat{\sigma}$, bearing the unit of length, plays the role of adjusting the range of the state space in which each failure state affects the heuristic cost-to-go. Increasing $\hat{\sigma}$ enlarges the range affected by each entry of the failure record. This consequently expedites the learning process and produces more conservative trajectories. The value of $\hat{\sigma}$ has little effect for the failure entries near the desired terminal state though.

We next verify that the basis functions so constructed indeed satisfy the desired conditions. First, for $x = \mathbf{0}$, we have $c(x-\eta) = c(\eta) > \sigma(\eta)$, thus $\psi(\mathbf{0},\eta) \equiv 0$ for all possible values of $\eta$. Thus (34) is satisfied.

Second, we verify the existence of solution of (32) by solving it. Here, we have

$$\begin{aligned} \frac{\partial U}{\partial x} &= \frac{K(N)}{N} \sum_{i=1}^{N} \frac{\partial \psi(x,\eta_i)}{\partial x} + \frac{\partial V(x,\mathbf{0})}{\partial x} \\ &= \frac{K(N)}{N} \sum_{\substack{i=1, \\ c(x-\eta_i) < \sigma(\eta_i)}}^{N} -\frac{1}{2} \sin\left[\frac{\pi}{\sigma(\eta_i)} c(x-\eta_i)\right] \frac{\partial c(x-\eta_i)}{\partial x} + \frac{\partial V(x,\mathbf{0})}{\partial x} \\ &= \frac{K(N)}{N} \sum_{\substack{i=1, \\ c(x-\eta_i) < \sigma(\eta_i)}}^{N} -\sin\left[\frac{\pi}{\sigma(\eta_i)} c(x-\eta_i)\right] (x-\eta_i)^T P W P + x^T P. \end{aligned} \tag{36}$$

The matrix $P$ is nonsingular. Thus

$$\bar{d}^T = -\frac{K(N)}{N} \sum_{\substack{i=1, \\ c(x-\eta_i) < \sigma(\eta_i)}}^{N} \sin\left[\frac{\pi}{\sigma(\eta_i)} c(x-\eta_i)\right] (x-\eta_i)^T P M \tag{37}$$

must hold. Recalling the block form representation of $P$,

$$PM = \begin{bmatrix} P_A & 0_{3\times 3} \\ P_C & 0_{3\times 3} \end{bmatrix}.$$

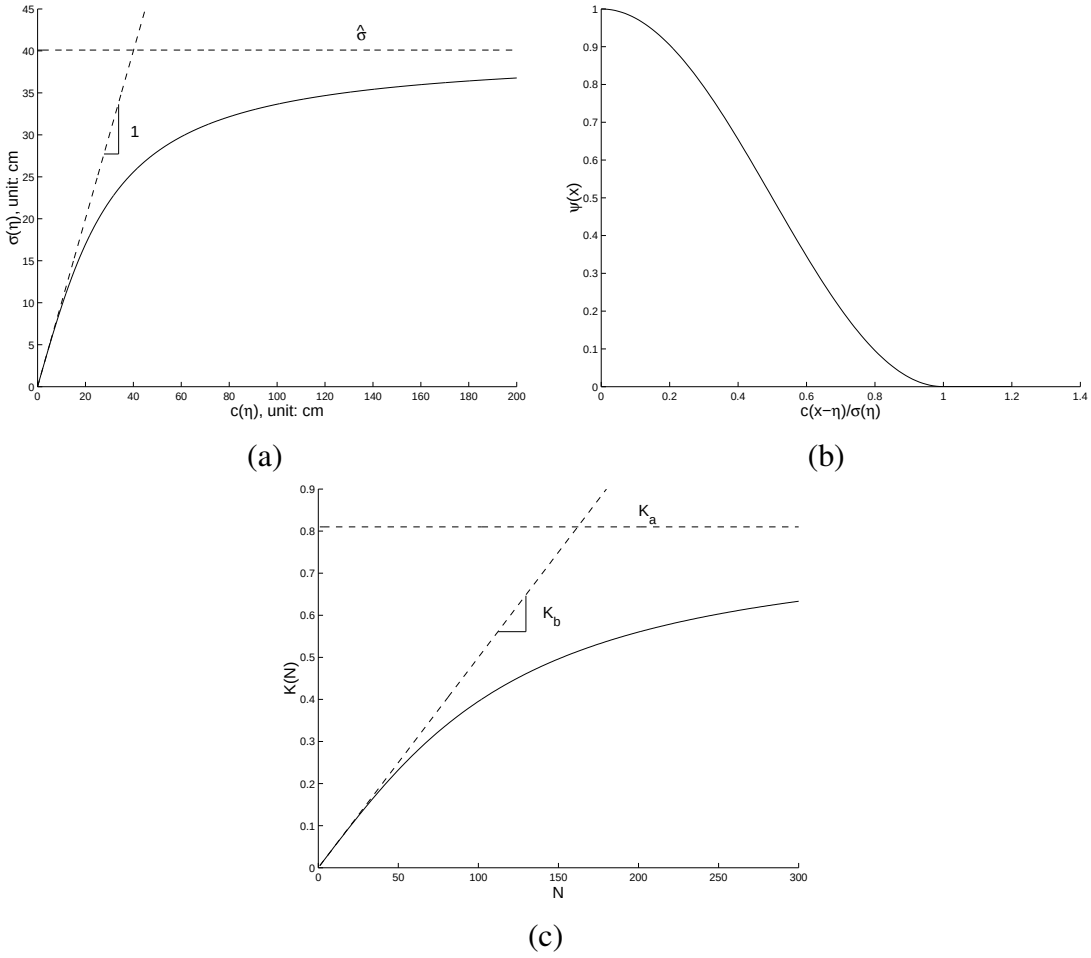


Fig. 3. (a) The function $\sigma(\eta)$ plotted against $c(\eta)$. (b) The basis function $\psi(x,\eta)$ plotted against $\frac{c(x-\eta)}{\sigma(\eta)}$. (c) The gain $K(N)$, a function of the length of the failure record.

So,

$$(x-\eta_i)^T PM = \left[(x-\eta_i)^T \begin{bmatrix} P_A \\ P_C \end{bmatrix}, \quad 0_{6\times 3}\right].$$

The solution of (32) is thus

$$d = -\frac{K(N)}{N} \sum_{\substack{i=1, \\ c(x-\eta_i)<\sigma(\eta_i)}}^{N} \sin\left[\frac{\pi}{\sigma(\eta_i)} c(x-\eta_i)\right][P_A,\ P_B](x-\eta_i). \tag{38}$$

Of course, the construction of the basis function is not unique. However, the prototype that we propose is almost the simplest possible. It is also effective, as demonstrated by the experiment described next.

### C. Experiment results

Again, we use the object retrieval task as an example. Video clips of the physical experiment as well simulation result are posted on the web [2]. Also see Figure 4. Note that in Figure 4, (a), the cross section of the learned heuristic cost-to-go function is taken at $x_3,\ x_4,\ x_5,\ x_6$ all set to zero; and the initial conditions in Figure 4, (b) and (c) are the same as those in Figure 2.

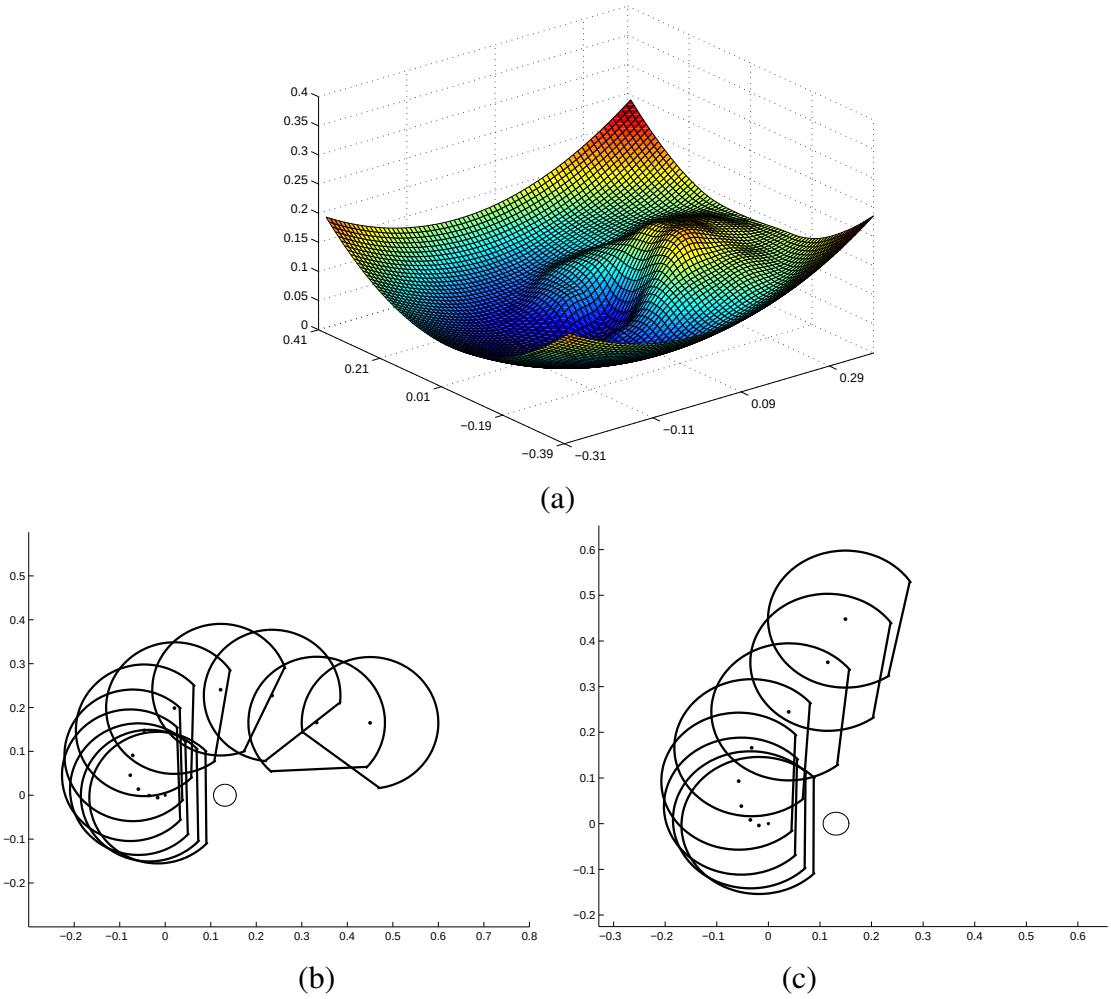


Fig. 4. (a) A cross section of the learned heuristic cost-to-go function at $x_3,\ x_4,\ x_5,\ x_6$ all set to zero.(b) One learned trajectory in simulation. (c) Another learned trajectory in simulation. The initial conditions in (b) and (c) are the same as those in Figure 2.

The matrices used in the LQR design of primitives, with the control of orientation integrated, are

$$Q = \text{diag}\{1.0,\ 1.0,\ 0.6,\ 0.4\ 0.4,\ 0.4\}, \quad R = I_{3\times 3}.$$

Consequently,

$$P = \begin{bmatrix} 1.55 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 1.55 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 1.08 & 0 & 0 & 0.77 \\ 1.0 & 0 & 0 & 1.55 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 1.55 & 0 \\ 0 & 0 & 0.77 & 0 & 0 & 1.40 \end{bmatrix}.$$

Again, all lengths are in meters. The radius of the robot is 0.15, the distance from the center of the robot to its flat facet is 0.09. Different from the previous section, we do not assume, neither do we prefer a priori, that the robot always face the object with its flat facet. The parameters of the basis function are set to $\hat{\sigma} = 0.4$, $K_a$ =0.8, $K_b$ =0.005. In the simulation, we terminate the learning process when the robot successfully reach the desired pose for 500 consecutive times. In the physical experiment, we consider the learning completed when failures seem to have stopped happening. Note that it is impractical to observe as many trials in the physical experiment as one can in the simulation. With these criteria, it takes about 60 to 100 failures for the robot to learn the task in both simulation and physical experiment.

From a comparison between Figure 4 and Figure 2, there are differences as well as similarities. First, recall that the direct design in the previous section use a separate loop to control the robot orientation such that the robot is always approximately facing the object, whereas the learning approach in this section integrates orientation control with position control. Indeed, on the learned trajectories, the robot does not insist on facing the object. The difference is especially clear between Figure 2, (b) and Figure 4, (b). In the latter, the robot undergoes much less rotation than in the former. The learned trajectory is thus superior in this aspect. On the other hand, by a visual examination of the cross section of the learned heuristic cost-to-go function, one may notice that it is a little "bumpy". This is due to the fact that this cost-to-go function is the sum of a finite number of basis functions. An analogous phenomenon is seen in a signal reconstructed by a finite number of its frequency components. The bumpiness of learned heuristic cost-to-go can be reduced at the expense of slower learning. For instance, one can make the gain $K(N)$ increase slower with $N$, allowing the robot to accumulate more experience from failures before applying the experience to actions in a significant way. When the "bumps" and "dimples" are mild enough, they are dominated by the desired slope of the cost-to-go function and does not cause problems at most places. However, if the cost-to-go function is supposed to have an unstable equilibrium, then a dimple may turn that equilibrium into a stable one. It was shown in [11] that a potential field in a 2D space with $M$ obstacles must have $M$ saddle points. In our case, the cross section of the heuristic cost-to-go function shown in Figure 4, (a) has the same property. There is supposed to be a saddle point near the object but on the opposing side of the desired terminal position. Sometimes in our experiments, this supposed saddle point

turns into a stable local equilibrium in the learned heuristic cost-to-go. Consequently, the robot may get stuck in that area. Although only affecting a small area, this is certainly a disadvantage. One possible remedy is to postprocess the learned cost-to-go function (by hand or by algorithm) and eliminate the undesired stable local equilibria.

## VI. Conclusion and future work

In this paper, we have introduced a scheme for integrating the lower level plant-dynamics-oriented control design and the higher level task-oriented motion design. The distinctive feature of our scheme is that the optimality of the lower level design is extended to the higher level motion design in a rigorous fashion. At the same time, we identify a good tuning knob of the task-level design — the heuristic cost-to-go function, from which the control law can be produced for evaluation with little computation. The admissibility condition of such heuristic cost-to-go function is stated by the main theorem of this paper. Further, both a direct design approach and a learning approach resulted from the proposed scheme are discussed. The trajectory design problem of an omnidirectional robot is considered. On one hand, it illustrates the application of the proposed techniques. On the other hand, the problem itself is of importance, and our solution of the problem complement existing techniques nicely.

Applying our approach to the trajectory design of an omnidirectional robot, we have shown that the optimal trajectory design can be determined by the conditions for preserving optimality together with a task-oriented heuristic cost-to-go specified only in robot positions (whereas the robot state includes both position and velocity). In fact, the optimal design is solved in closed form. Further, the existing results on motion planning based on artificial potential fields may suggest good choices for the heuristic cost-to-go. Thus, our result connects optimal control design, which concentrates on dealing with the vehicle dynamics, and artificial potential-based motion planning techniques, which is computationally efficient but usually does not take into account of the vehicle dynamics rigorously.

There appear to be many directions in which our results can be extended. These include trajectory design for a wider range of tasks and autonomous systems, including nonholonomic vehicles, vehicles in 3-D, and non-vehicle robotics systems, etc. On the other hand, we have assumed that the primitives are regular solutions of an optimal control problem. However, in some cases the primitive are singular optimal solutions (e.g., [22]). We will also consider such scenarios in our future work.

## VII. REFERENCES


[1] Trajectory Design Based on Motion Primitives: Direct Design. Cornell University. [Online] http://www.mae.cornell.edu/keyong_li/TDMP.htm

[2] Trajectory Design Based on Motion Primitives: Learning. Cornell University. [Online] http://www.mae.cornell.edu/Keyong_li/TDMP_Learning.htm

[3] J. Baillieul & A. Suri. Information patterns and hedging Brockett's theorem in controlling vehicle formation, in *Proceedings of the 42nd IEEE Conference on Decision and Control*, Dec. 2003.

[4] J. Barraquand, B. Langlois, & J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles, in *Algorithmica*, 10:121-155, 1993.

[5] F. Bullo. Series expansions for the evolution of mechanical control systems. *SIAM J. Control and Optimization*, 40(1):166-190, 2001.

[6] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, Massachusetts, 2000.

[7] P. Cheng & S. M. LaValle. Resolution complete rapidly-exploring random trees, in *Proc. IEEE Int'l Conf. Robotics and Automation*, pp. 267-272, 2002.

[8] E. Frazzoli. Quasi-random algorithms for real-time spacecraft motion planning and coordination, *Acta Astronautica*, 53(4–10), pp. 485-489, 2003.

[9] E. Frazzoli, M. A. Dahleh, & E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal on Guidance, Control and Dynamics*, 25(1):116-129, 2002.

[10] T. Kalmar-Nagy, R. D'Andrea, & P. Gauguly. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle, *Robotics and Autonomous Systems*, 46:47-64, 2004.

[11] D. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations, in *IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, Mar. 1987, pp. 1-6.

[12] B. Krogh. A generalized potential field approach to obstacle avoidance control, in *Robotics Research: The Next Five Years and Beyond*, Bethlehem, PA, USA, Aug. 14C16, 1984, pp. 1C15.

[13] F. Lamiraux, D. Bonnafous, & C. V. Geem. Nonholonomic Path Optimization, in *Control Problems in Robotics*, A. Bicchi, H.I. Christensen, D. Prattichizzo (Eds.), Springer-Verlag Berlin Heidelberg, STAR 4, pp. 1-18, 2003.

[14] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[15] J. P. Laumond, S. Sekhavat, & F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots, in *Robot Motion Planning and Control*, J. P. Laumond, editor, pp. 1-53. Springer-Verlag, Berlin, 1998.

[16] S. M. LaValle. From dynamic programming to RRTs: algorithmic design of feasible trajectories, in *Control Problems in Robotics*, A. Bicchi, H.I. Christensen, D. Prattichizzo (Eds.), Springer-Verlag Berlin Heidelberg, STAR 4, pp. 19-37, 2003.

[17] S.A. Masoud & A.A. Masoud. Constrained Motion Control Using Vector Potential

Fields. *IEEE Transactions on systems, man, and cybernetics – Part A: Systems and Humans*, 30(3):251-272, May, 2000.

[18] A. M. Meystel & J. S. Albus. *Intelligent Systems*, Wiley-Interscience Publication, 2002.

[19] K. L. Moore, & N. S. Flann. A six-wheeled omnidirectional autonomous mobile robot, *Control Systems Magazine, IEEE*, 20(6):53-66, Dec. 2000.

[20] R. M. Murray & S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Trans. Automatic Control*, 38(5):700-716, 1993.

[21] Y. Pinchover & J. Rubinstein. *An Introduction to Partial Differential Equations*. Cambridge University Press, UK, 2005.

[22] O. Purwin & R. D'Andrea. Trajectory generation and control for four wheeled omnidirectional vehicles, *Robotics and Autonomous Systems*, 54(1):13-22, Jan 31, 2006.

[23] A. Richards, T. Schouwenaars, J. P. How, & E. Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming, *AIAA Journal on Guidance, Control and Dynamics*, 25(4), 2002.

[24] E. Rimon & D. Koditschek. Exact robot navigation using artificial potential functions, *IEEE Trans. Robot. Automat.*, 8:501C518, Oct. 1992.

[25] M. Sherback, O. Purwin, & R. D'Andrea. Real-time motion planning and control in the 2005 Cornell RoboCup system. In *Lecture Notes in Control and Information Sciences*. Springer Verlag. To appear.

[26] R. Volpe, Techniques for collision prevention, impact stability, and force control by space manipulators, in *Teleoperation and Robotics in Space*, S. Skaar and C. Ruoff, Eds. Washington, DC: AAAI, 1994, pp. 175C208.

[27] K. Watanabe, Y. Shiraishi, S.G. Tzafestas, J. Tang, & T. Fukuda. Feedback control of an omnidirectional autonomous platform for mobile service robots, *Journal of Intelligent and Robotics Systems*, v.22, pp. 315-330, 1998.

[28] S. Waydo & R.M. Murray. Vehicle motion planning using stream functions, in *2003 IEEE Int. Conf. Robotics Automation*, Sep. 2003.