51st IEEE Conference on Decision and Control
December 10-13, 2012. Maui, Hawaii, USA

Learning Potential Functions by Demonstration for Path Planning

Andrew Winn and Xuemei Gao and Sandipan Mishra and A. Agung Julius

Abstract— **Potential functions can be used to design efficient path planning schemes. However, it is often difficult to design appropriate potential functions to mimic desired behavior of the agent. Instead of using a pre-designed potential function for path planning, this paper presents an algorithm that learns the underlying potential function from a given sample trajectory generated by a "expert" (say, a human). This underlying potential function implicitly incorporates obstacle avoidance information that may be intuitive or experience-based. The potential function to be learned is parametrized and the parameter weights are obtained through minimization of a well-designed cost function via a gradient descent search algorithm. Once learned, this potential function can be used for path planning in case of alternative (and more complex) scenarios, such as those with multiple obstacles. The paper presents the theoretical foundation and numerical validation of the proposed algorithm.**

I. Introduction

Path planning plays a significant role in robotics. The path that is being planned is essentially a sequence of actions to determine how an agent or a manipulator transforms from one configuration to another. Currently existing path planning methods can be divided into sampling-based path planning and combinatorial path planning (as in [1]) based on whether the continuous configuration space is discretized. Combinatorial path planning is limited due to spatial degree restrictions. Sampling-based path planning of a variety of types have been developed, like rapidly exploring random trees (RRT) [2], path-length localized RRT-like searches [3] and probabilistic roadmaps (PRMs) [4]. Potential function is special in path planning since it can be used in both ways. For example, randomized potential field [5] was one of the first sampling-based planning algorithms, while navigation functions is built upon potential functions in a continuous configuration [6]. However, the key step of construction of potential functions [7] has been typically heuristic in nature.

To enable an agent to adapt to unforeseen circumstances, machine learning algorithms, such as structured machine learning algorithms [8] and more recently, deep learning algorithms [9] have been developed upon sampling-based path planning. These machine learning algorithms learn structured hypotheses from data, which includes structured inputs and outputs, as well as internal structure usually in the form of one or more statistical inference relations. Apprenticeship learning (also called learning from demonstration) [10] is an example of this class of methods. Most approaches in apprenticeship learning attempt to imitate the expert by learning a direct mapping from inputs to outputs with either an explicitly expressed supervised reward function, such as [11] and [12], or an unknown reward function, like apprenticeship learning via inverse reinforcement learning [10], [13]. Since machine learning research involved structured representations from the beginning [8], changes beyond structured environmental features may lead to a failure in the application of learning results.

Most of the work in the field of design and analysis of potential functions has focused on the benefits of different types of potential functions, and require manual tuning of functions to generate a desired behavior of the system. A few automated tuning methods have also been explored, such as adjustable navigation functions for unknown sphere world [14]. However, a systematic learning-based approach to design potential fields is generally lacking.

In this paper, we propose a learning method that determines a suitable potential function from a given sample trajectory for a "simple" navigation task. The learned potential function can then be employed to handle more complex navigation task. This idea is inspired by a human's learning process, i.e., starting to learn from simple cases and constructing complex environments using these structured blocks.

The proposed method learns from a trajectory demonstrated by an expert, such as a human. With a cost function, the proposed method takes a desired expert-generated trajectory and iteratively adjust a parameterized space to reshape the potential function to capture the essence of this path planning, such as obstacle avoidance and destination reaching. The benefit of learning potential functions is that it provides us more flexibility to apply the learned result from simple situations to do path planing in more complicated situations, like multiple obstacles.

II. Notation

$\eta, \tilde{\eta}$	Traj. generated by nominal/perturbed PFs, resp.
η^*	Expert-generated (desired) trajectory
$\boldsymbol{\eta}, \boldsymbol{\eta}^*, \tilde{\boldsymbol{\eta}}$	Lifted form of trajectories
ϕ, Φ	Component (obstacle, goal, etc.) and total PFs, resp.
F	Perturbation potential function
Δ_ϵ	Difference between perturbed and nominal traj.
T_s	Sample time
$J(\eta^*, \Phi)$	Cost functional
A, B, C	Linear mappings
$\mathbf{p}$	Potential function parameters
$\delta\mathbf{p}$	Update direction in parameter space
$C^\dagger$	Moore-Penrose Pseudoinverse of C
α, β	Tikhonov regularization constants

A. Winn and A. Julius are with the department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 {winna,juliua2}@rpi.edu

X. Gao and S. Misrha are with the department of Mechanical Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 {gaox4,mishrs2}@rpi.edu

978-1-4673-2066-5/12/$31.00 ©2012 IEEE

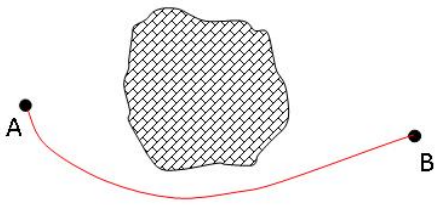


Fig. 1. Learning Scenario: Path Planning from Point A to Point B with One Obstacle In-between, and a Sample Trajectory is Provided by an Expert.

III. Problem Description

Potential functions can be used to generate a path between an initial point and a goal point for an agent in different ways [15], [16]. For this paper, we will consider a method in which the negative gradient of the potential function determines the velocity of the agent at that point.

Consider an agent traveling in a two-dimensional space shown in Fig. 1. There is one obstacle between the starting point A and the goal B. We have a desired trajectory, η^*, provided by an expert between point (A) and point (B). Starting with some initial potential field Φ composed of an obstacle potential function ϕ_o and a goal potential function ϕ_g, as well as the trajectory η that Φ generates, we seek a method to iteratively update the potential field to have η match the desired trajectory as closely as possible with respect to some cost function.

Let $\eta = [\mathbf{x}\ \mathbf{y}]^T$ be the trajectory that the agent traverses. This trajectory is generated by the potential functions in the discrete domain by

$$\frac{\eta[n+1]-\eta[n]}{T_s} = -\nabla\Phi(\eta[n]), \tag{1}$$

where $\Phi(\eta[n])$ is the sum of the obstacle and goal potentials functions at the point n on trajectory η (denoted as $\eta[n]$). Rearranging the terms yields the following state equations

$$\eta[n+1] = \eta[n] - \nabla\Phi(\eta[n])\cdot T_s. \tag{2}$$

We define the cost functional, $J(\eta^*,\Phi)$, to be

$$J(\eta^*,\Phi) \triangleq \sum_{n=1}^{N} \|\eta[n]-\eta^*[n]\|_2^2, \tag{3}$$

where N is the number of points in the desired trajectory η^*, and η is the trajectory generated by the potential field Φ.

The objective is to search for an optimal potential field, Φ, that minimizes the cost function $J(\eta^\star,\Phi)$. This potential field Φ can be used to do path planing in different and more complicated situations.

IV. Algorithm for Learning Potential Function

The goal of the proposed algorithm is to learn a potential function that captures the essence of an expert's approach to path planning. Sections IV-A and IV-B build up the theoretical foundation for a viable algorithm, and Section IV-C presents the resulting algorithm. Section IV-D provides solutions to overfitting and ill-conditioned problems possibly encountered in the process of potential function learning.

Note that in describing the algorithm, we use boldface to denote the lifted form of the given sequence, wherein the sequence is lifted into a column vector. For example,

$$\boldsymbol{\eta} \triangleq [\eta^T[1], \eta^T[2], \ldots, \eta^T[N]]^T.$$

A. Gradient Descent Search

To minimize the functional $J(\eta^\star,\Phi)$, we want to calculate the functional derivative (Gâteaux differential) of this functional with respect to the total potential function. We find

$$dJ(\eta^*,\Phi,F) \triangleq \lim_{\epsilon\to 0}\frac{J(\eta^*,\Phi+\epsilon F)-J(\eta^*,\Phi)}{\epsilon}. \tag{4}$$

is the differential of J in the direction of F. Our goal is to update the potential function with some F that makes this differential negative. To this end, let us define $\tilde{\eta}$ to be the perturbed trajectory generated by $\Phi+\epsilon F$, which by (2) we find to be

$$\tilde{\eta}[n+1] \triangleq \tilde{\eta}[n] - \nabla\left(\Phi(\tilde{\eta}[n]) + \epsilon F(\tilde{\eta}[n])\right)\cdot T_s \tag{5}$$

$$\tilde{\eta}[0] \triangleq \eta[0]. \tag{6}$$

We can define Δ_ϵ to be the difference between the perturbed and nominal trajectory, i.e.,

$$\Delta_\epsilon[n] = \tilde{\eta}[n] - \eta[n]. \tag{7}$$

Expanding this relationship with (2) and (5) substituting $\tilde{\eta}[n] = \eta[n] + \Delta_\epsilon[n]$ yields

$$\Delta_\epsilon[n+1] = \Delta_\epsilon[n] - T_s\nabla\Phi(\eta[n]+\Delta_\epsilon[n]) + \epsilon T_s\nabla F(\eta[n]+\Delta_\epsilon[n]) + \nabla\Phi(\eta[n]). \tag{8}$$

We would like to expand the gradients into Taylor series about $\eta[n]$ and separate the $o(\epsilon)$ terms. In the extended version of this article [17], we show that

$$\Delta_\epsilon[n+1] = -\epsilon T_s\nabla F(\eta[n]) - \epsilon T_s\sum_{k=0}^{n-1}\left(\prod_{i=k+1}^{n}(I - T_s H\Phi(\eta[i]))\right)\nabla F(\eta[k]) + \mathbf{o}(\epsilon). \tag{9}$$

Using lifted form, we can represent $\boldsymbol{\Delta}_\epsilon$ as

$$\boldsymbol{\Delta}_\epsilon = -\epsilon T_s A\boldsymbol{\nabla}\mathbf{F} + \mathbf{o}(\epsilon), \tag{10}$$

where A is defined to be

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ I & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ (I-T_sH\Phi(\eta[1])) & I & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} & \mathbf{0} \\ \prod_{i=1}^{N-1}(I-T_sH\Phi(\eta[i])) & \prod_{i=2}^{N-1}(I-T_sH\Phi(\eta[i])) & \cdots & I & \mathbf{0} \end{bmatrix}, \tag{11}$$

and $\boldsymbol{\nabla}\mathbf{F}$ is the lifted form of ∇F.

Using notations in lifted form transforms the ℓ^2 space to the Euclidean space. Examining (4), we are able to show (see [17]),

$$dJ(\boldsymbol{\eta}^*,\Phi,F) = -2T_s(\boldsymbol{\eta}-\boldsymbol{\eta}^*)^T A\boldsymbol{\nabla}\boldsymbol{F}. \tag{12}$$

Thus our goal is to find a potential function F such that $-2T_s(\boldsymbol{\eta}-\boldsymbol{\eta}^*)^T A\nabla F < 0$.

B. Parameter Update Law

F is a function of class C^2 in our system space. Without any constraint, our search space for F would be infinite dimensional and intractable; further, we would expect an arbitrary potential function to be able to produce the expert generated path exactly, which may not generalize well to other scenarios due to overfitting (e.g., see [18]). If, however, we define our potential function Φ to be a sum of obstacle and goal potential functions that are specified by a finite number of parameters, we constrain F to a set of functions that represent changes in those finite (and possibly bounded) parameters. Further, if the gradient of F at any point can be represented by a linear combination of these parameters, then F can be found using the tools of linear algebra. For the rest of this discussion we only consider potential functions of this form.

Since we are assuming the gradient of F can be determined by a linear combination of the changes in the potential function parameters, we can represent this as the following matrix equation,

$$\nabla F = B\delta\mathbf{p} \tag{13}$$

where $\delta\mathbf{p}$ is the corresponding changes in the parameters $\mathbf{p}$, both of which are lifted into column vectors. B is the linear mapping between $\delta\mathbf{p}$ and the gradient of F. Then in order to guarantee a negative cost differential, we propose the following update law,

$$\mathbf{p}_{j+1} = \mathbf{p}_j + \epsilon \cdot \delta\mathbf{p}_j \tag{14}$$

where j is the iteration index for the learning algorithm, ϵ is the iteration step size, and

$$\delta\mathbf{p} = C^{\dagger}(\boldsymbol{\eta} - \boldsymbol{\eta}^*), \qquad C \triangleq AB. \tag{15}$$

Any standard gradient descent algorithm can now be used to iteratively find the optimal potential function. In this paper we use a gradient descent algorithm with an adaptive step size.

C. Algorithm

In light of the previous results, we propose the following algorithm:

Choose initial $\Phi(\mathbf{p}), \epsilon$
Cost $= J(\eta^*, \Phi(\mathbf{p}))$
while Cost $>$ Threshold **do**
  Calculate A from (11)
  Calculate B from (13)
  $\delta\mathbf{p} = (AB)^{\dagger}(\eta - \eta^*)$
  $\mathbf{p} = \mathbf{p} + \epsilon \cdot \delta\mathbf{p}$
  if $J(\eta^*, \Phi(\mathbf{p})) <$ Cost **then**
    $\epsilon = \beta\epsilon, \qquad \beta > 1$
  else
    $\epsilon = \alpha\epsilon, \qquad 0 < \alpha < 1$
  end if
  Cost $= J(\eta^*, \Phi(\mathbf{p}))$
end while

This learning process is shown in Fig. 2.

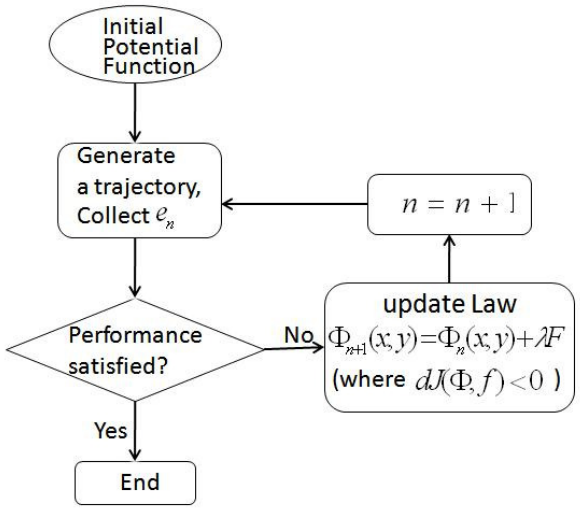


Fig. 2. Flow Chart of Learning Potential Functions from Demonstration

D. A Note on Regularization

If the class of potential functions is sufficiently large, then the potential function can start matching subtle undesired aspects of the expert-generated trajectory; this is analogous to overfitting. Another concern is that C^TC, whose inverse is required for (15) may be poorly conditioned. If this is the case, then large, sudden changes may occur in the $\delta\mathbf{p}$ vector, and the result may be erratic.

One method of handling ill-conditioned matrices is to use truncated singular value decomposition (TSVD) [19], [20], which takes the singular value decomposition of C^TC and sets the inversion to

$$(C^TC)^{-1} = V_t\Sigma_t^{-1}U_t^*$$

Where Σ_t is a $t \times t$ matrix consisting of the singular values for whose normalized value is greater than some tolerance. V_t and U_t consist of the first t columns in from the singular value decomposition matrices of C^TC. This prevents the code from breaking if C^TC is singular, and provides automatic regularization.

This issue of overfitting may be also handled using Tikhonov regularization [21]. In this case, the update law with regularization is given by

$$\begin{aligned}\mathbf{p}_{j+1} &= \mathbf{p}_j + \delta\mathbf{p}_j \\ \delta\mathbf{p} &= (C^TC + \Gamma^T\Gamma)^{-1}C^T(\boldsymbol{\eta} - \boldsymbol{\eta}^*)\end{aligned} \tag{16}$$

where Γ is some suitably chosen matrix, often chosen to be αI for $\alpha > 0$.

In the Sec. V, both methods are employed in conjunction.

V. Implementation Example

For our simulation, we aim to generate a potential function in two-dimensional space that starts at some initial point $\mathbf{x}_{\text{init}}$ and terminates at some goal point $\mathbf{x}_g$ while avoiding some number of circular obstacles centered at $\mathbf{x}_{o,i}$, where i is the

index of the obstacle. We represent our potential function by

$$\Phi(\eta[n]) = \phi_g(\eta[n]) + \sum_{i=1}^{n_{\text{obs}}} \phi_{o,i}(\eta[n]), \tag{17}$$

where $\phi_g(\eta[n])$ is the potential function associated with the goal point, and $\phi_{o,i}(\eta[n])$ is the potential function associated with the i^{th} obstacle. We take our obstacle and goal potential functions to be radially symmetric about the corresponding obstacle or goal points. This is by no means necessary, but provides some convenient simplifications. In this case, the potential functions can be represented by functions of the radius, that is, $\phi_g(r_{g,n})$, and $\phi_{o,i}(r_{o,i,n})$, where $r_{g,n}$ is the distance from $\eta[n]$ to the goal, and $r_{o,i,n}$ is the distance from $\eta[n]$ to the i^{th} obstacle. Define $\mathbf{v}_{g,n} \triangleq \eta[n] - \mathbf{x}_g$ to be the vector from the goal point to $\eta[n]$ and $\mathbf{v}_{o,i,n} \triangleq \eta[n] - \mathbf{x}_{o,i}$ the vector from the center of the i^{th} obstacle to $\eta[n]$. With this we find that the gradient and Hessian of the goal potential function are given by

$$\nabla\phi_g(\eta[n]) = \frac{1}{r_{g,n}} \frac{d\phi_g}{dr}(r_{g,n}) \mathbf{v}_{g,n}, \tag{18}$$

$$H\phi_g(\eta[n]) = \left(\frac{d^2\phi_g}{dr^2}(r_{g,n}) - \frac{1}{r_{g,n}} \frac{d\phi_g}{dr}(r_{g,n}) \right) \mathbf{v}_{g,n}\mathbf{v}_{g,n}^T + \frac{1}{r_{g,n}} \frac{d\phi_g}{dr}(r_{g,n}) I, \tag{19}$$

with analogous results for the obstacle potential functions.

We represent $\phi_g(r_{g,n})$ by a cone parametrized by a steepness coefficient λ,

$$\phi_g(r_{g,n}) = \lambda r_{g,n}. \tag{20}$$

We represent the obstacle potential function by an arbitrary function of radial distance. There are several ways to do this. For each obstacle in our simulation we choose an *effective radius* $r_{\text{eff},i}$, outside of which the potential is zero everywhere. We then partition the range $R_i \leq r_i \leq r_{\text{eff},i}$ into intervals of size δr, where R_i is the radius of the circular obstacle. We then attempt to learn a step function for $\frac{d^2\phi_{o,i}}{dr^2}$ on this range with these intervals. In this case we calculate the potential and the gradient, $\phi_{o,i}$ and $\frac{d\phi_{o,i}}{dr}$, via numerical integration with the constraint that $\phi_{o,i}(r_{\text{eff},i}) = 0$ and $\frac{d\phi_{o,i}}{dr}(r_{\text{eff},i}) = 0$.

The only step left to perform is to determine the B matrix in (13) that maps changes in the parameters to changes in the gradient of F at each point $\eta[n]$. From (18) we see that for the goal potential function the effect on $\nabla F(\eta[n])$ by a change in the steepness coefficient $\delta\lambda$ is given by

$$\nabla F_g(\eta[n]) = \frac{\delta\lambda}{r_{g,n}} \mathbf{v}_{g,n}. \tag{21}$$

This implies that the matrix B_g which transforms $\delta\lambda$ into its corresponding change in the lifted ∇F is

$$B_g = \delta\lambda \begin{bmatrix} \frac{\mathbf{v}_{g,0}^T}{r_{g,0}} & \frac{\mathbf{v}_{g,1}^T}{r_{g,1}} & \cdots & \frac{\mathbf{v}_{g,n}^T}{r_{g,n}} \end{bmatrix}^T. \tag{22}$$

We determine the obstacle potential function $\nabla F_{o,i}(\eta[n])$ by $\frac{d\phi_{o,i}}{dr}(r_{o,i,n})$, which is integrated from our learned function $\frac{d^2\phi_{o,i}}{dr^2}$. With the boundary conditions that the potential function and its derivative must be zero at the effective radius, this is equivalent to summing $-\delta r \frac{d^2\phi_{o,i}}{dr^2}$ backwards from $r_{\text{eff},i}$ to the interval that contains $r_{o,i,n}$, call it the k^{th} interval, and adding $(r_{o,i,n} - (k+1)\delta r)\frac{d^2\phi_{o,i}}{dr^2}[k]$. Assume that the n^{th} point on the trajectory falls within k^{th} radial interval for the i^{th} obstacle. Thus for the column vector of the differential change in the second derivative $\delta\frac{d^2\phi_{o,i}}{dr^2}[k]$ we find

$$\nabla F_{o,i}(\eta[n]) = \cdots \underbrace{\frac{\mathbf{v}_{o,i,n}}{r_{o,i,n}} \begin{bmatrix} \underbrace{0 \cdots 0}_{k-1 \text{ times}} & r_{o,i,n} - (k+1)\delta r & -\delta r & \cdots & -\delta r \end{bmatrix}}_{B_{o,i}[n]} \delta\frac{d\phi_{o,i}}{dr}. \tag{23}$$

This implies that the matrix $B_{o,i}$ which transforms $\delta\frac{d^2\phi_{o,i}}{dr^2}[k]$ into its corresponding change in the lifted ∇F is

$$B_{o,i} = \begin{bmatrix} B_{o,i}^T[0] & B_{o,i}^T[1] & \cdots & B_{o,i}^T[N] \end{bmatrix}^T. \tag{24}$$

Thus, if we define our lifted differential parameter vector from (13) to be

$$\delta\mathbf{p} = \begin{bmatrix} \delta\frac{d^2\phi_{o,1}}{dr^2} & \delta\frac{d^2\phi_{o,2}}{dr^2} & \cdots & \delta\frac{d^2\phi_{o,N}}{dr^2} & \delta\lambda. \end{bmatrix}^T. \tag{25}$$

The B matrix that satisfies the relationship in (13) is given by

$$B = \begin{bmatrix} B_{o,1} & B_{o,2} & \cdots & B_{o,N} & B_g \end{bmatrix}. \tag{26}$$

We now have the necessary components to implement our algorithm as described in Section IV.

For regularization we define two regularization constants, one that weighs all of the obstacle function's parameters equally, and the other that weighs the goal parameter separately, so that we can control the regulation on the obstacle and goal potential functions separately. If there are n parameters that define the obstacle potential function, then this yields a Tikhonov regularization matrix of

$$\Gamma = \begin{bmatrix} \alpha I_{n\times n} & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{1\times n} & \beta \end{bmatrix}. \tag{27}$$

We now have an appropriate framework and an adequate setup to perform validation of this theory in MATLAB. For the first set of simulations, we consider a two-dimensional task space with a single obstacle that occludes the goal point from the initial point. The obstacle is taken to have a bounding radius $R_o = 2$ and an effective radius $R_{\text{eff}} = 7$, outside which the potential field is zero. We set the TSVD tolerance defined in section IV-D to 10^{-10}, α to 1, β to 1 (from (16)), and our update step size ϵ to 0.01. These values are found heuristically to yield good results. In all cases we initialize our potential functions to

$$\frac{d^2\phi_{o,\text{init}}(r)}{dr^2} = \frac{-2}{(k\delta r)^2}, \qquad k = \left\lfloor \frac{r}{\delta r} \right\rfloor, \tag{28}$$

$$\phi_{g,\text{init}}(r) = 1.2r, \tag{29}$$

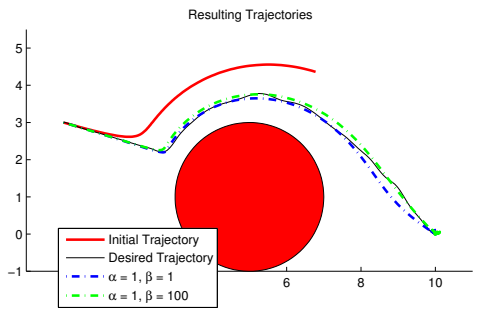


Fig. 3. Results for Human Trajectory

where $\lfloor \frac{r}{\delta r} \rfloor$ denotes the largest integer less than or equal to $\frac{r}{\delta r}$. Note that the obstacle's potential function is determined by integrating (28) with the constraints that its value and derivative at R_{eff} are both zero. We present it in this form because this is the form learned by the algorithm.

VI. Simulation Results

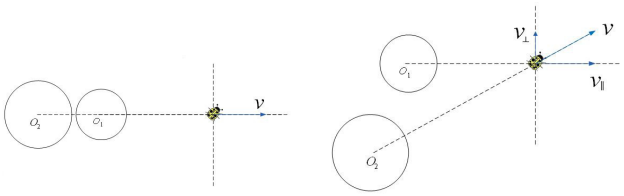
Our initial simulation tested our algorithm on trajectories generated from known potential functions; by doing this, we know the desired potential function *a priori* and can better gauge the quality of the results. We also examined the results with and without regularization which validate the use of them. These results can be found in the extended version of this paper [17]. The results presented here are the subsequent simulations that used trajectories generated by human users.

A. One Obstacle with a Human-Generated Trajectory

For this simulation, we consider learning on a human-generated trajectory, one that may not be realizable by the choice of potential functions. To this end, we provide a user interface that is initialized with a straight-line trajectory from the initial point to the goal point with 30 evenly spaced points that can be moved along this line. The user is able to use these 30 points to arbitrarily shape the trajectory; cubic splines are then used to generate the desired resolution.

The results on a demonstration trajectory using two different sets of regularization parameters, $\alpha = 1, \beta = 1$ and $\alpha = 1, \beta = 100$ are given in Fig. 3. Clearly the results for the first set of parameters are worse. This is because the parameter on the goal potential function learns too fast compared to the parameters on the obstacle potential function. It seems possible that the goal potential function changes too much initially to minimize the error that it pulls the trajectory into a local minimum that prevents the obstacle potential function from continuing to change. When β was drastically increased to 100, the final trajectory matched the initial trajectory quite closely, as also shown in Fig. 3.

This puts us in a good position to examine the effect of α on the resulting trajectory. Looking at Fig. 3, one can see that the final trajectory starts to match some of the non-smoothness of the demonstration trajectory. This may not generalize to other scenarios well and thus may not be desirable. We have found that increasing α has a smoothing effect on the final trajectory, whereas decreasing α allows the human's trajectory to be followed more closely. Figures demonstrating this result are included in the extended version of this paper [17].



(a) Two Aligned Obstacles (b) Two Obstacles

Fig. 4. Modified Superposition Method

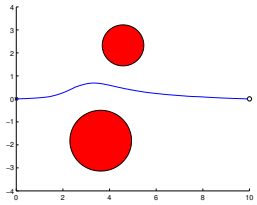
B. Handling Multiple Obstacle Scenarios

The potential function learned from the human-generated trajectory can be used to handle a situation with multiple obstacles of the same type, which is often an important skill for unmanned vehicles. Since we learned the potential function for an obstacle individually, we have more control on how to construct the potential function for multiple obstacles.

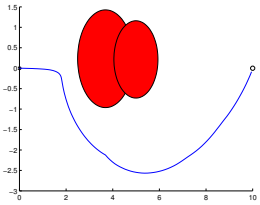
In the multiple obstacle case, summing their potential functions together is the most straightforward approach, however, it also causes several problems. If several obstacles are very close to each other, the potential function in some points are likely to be over-emphasized. For example, when two obstacles are aligned as in Fig. 4(a), the potential function becomes unnecessarily large at the origin, where the agent is. In the direction of the line between O_1 and the agent, the agent is unable to hit obstacle O_2 if it does not first hit the closest obstacle, obstacle O_1 in this case. By intuition ignoring the action of the further obstacle, O_2, can simplify this multiple-obstacle case.

From this perspective, we decompose the action of the further obstacle O_2 on the agent v into two components: the parallel component $v_\parallel$ and the perpendicular component $v_\perp$, as shown in Fig. 4(b). The parallel component is parallel to the line connecting the agent with the closest obstacle. Next only the largest of the parallel component $v_\parallel$ is kept. Since the multiple-obstacle case discussed in this paper only involves obstacles of the same kind, the obstacle with the largest parallel component is the one closest to the agent in that direction. That is to say, in the direction of the line between the closest obstacle and the agent, the resultant gradient of the combined potential function is the maximum one in that direction.

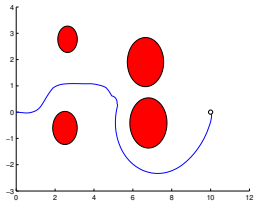
Finally, we implement the modified superposition method on multiple obstacles. Take two obstacles for example; we randomly generate two obstacles somewhere in-between the starting point and the destination. Fig. 5(a) and Fig. 5(b) show that our method is able to find a feasible trajectory either in the space on one side of two obstacles or through



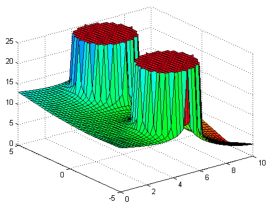
(a) A Trajectory Passing In-between Two Obstacles



(b) A Trajectory Passing by Two Obstacles



(c) A Trajectory Generated in the Situation of Four Obstacles



(d) Produced Potential Field for Two Obstacles

Fig. 5. Trajectories around Multiple Obstacles

the space between two obstacles. Furthermore, our method extends to situations with more than two obstacles. Fig. 5(d) shows the trajectory generated in the presence of 4 obstacles. When there is enough safe space between two obstacles for the agent to pass through, the proposed method generates a path through them; while if two obstacles are too close to each other, the proposed method finds a path around the obstacles.

VII. Conclusion

This paper investigated the applicability of a gradient-search based learning from demonstration algorithm for path planning. The algorithm aimed to identify an unknown potential function that generates a given (prescribed) trajectory planned by a expert (a human, e.g.). In other words, the goal was to determine what potential field was subconsciously motivating the expert trajectory generation. Since the space of potential functions is very large (class C^2), the structure (or approximate parametrization) of the potential function was assumed to be known, with unknown parameters. The proposed method thus learned potential functions from a given trajectory which had embedded goal-reaching and obstacle-avoidance information. The learning by demonstration was carried out on simple obstacles and then these identified potential functions were subsequently used for more complex obstacle scenarios. As a result, this method avoided the tedious process of designing potential functions manually for desired trajectories. Using the learned results, path planning in a more complicated environment of obstacles of the same type was achieved from simple building blocks.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, May 2006.

[2] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 1478 –1483.

[3] N. Wedge and M. Branicky, "Using path-length localized rrt-like search to solve challenging planning problems," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 3713 –3718.

[4] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566 –580, aug 1996.

[5] J. Barraquand and J.-C. Latombe, "A monte-carlo algorithm for path planning with many degrees of freedom," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, may 1990, pp. 1712 –1717 vol.3.

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

[7] J.-C. Latombe, *Robot Motion Planning*, 1st ed., ser. The Springer International Series in Engineering and Computer Science. Springer, Dec. 1991. [Online]. Available: http://www.worldcat.org/isbn/0792391292

[8] T. G. Dietterich, P. Domingos, L. Getoor, S. Muggleton, and P. Tadepalli, "Structured machine learning: the next ten years," *Machine Learning*, vol. 73, no. 1, pp. 3–23, Oct. 2008. [Online]. Available: http://dx.doi.org/10.1007/s10994-008-5079-1

[9] Y. Bengio, "Learning deep architectures for AI," Dept. IRO, Universite de Montreal, Tech. Rep., 2007. [Online]. Available: http://www.iro.umontreal.ca/~lisa/pointeurs/TR1312.pdf

[10] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *In Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press, 2004.

[11] S. S. Christopher. G. Atkeson, "Robot learning from demonstration," in *Machine Learning: Proceedings of the Fourteenth International Conference (ICML '97)*.

[12] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, "Learning to fly," in *In Proceedings of the Ninth International Conference on Machine Learning*. Morgan Kaufmann, 1992, pp. 385–393.

[13] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 663–670.

[14] I. Filippidis and K. J. Kyriakopoulos, "Adjustable navigation functions for unknown sphere worlds," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, dec. 2011, pp. 4276 –4281.

[15] C. Warren, "Global path planning using artificial potential fields," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, may 1989, pp. 316 –321 vol.1.

[16] J.-O. Kim and P. Khosla, "Real-time obstacle avoidance using harmonic potential functions," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, apr 1991, pp. 790 –796 vol.1.

[17] A. Winn, X. Gao, S. Mishra, and A. Julius, "Learning potential functions by demonstration for path planning," "http://www.rpi.edu/~winna/papers/learning_pfs.pdf".

[18] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. AMLBook, 2012.

[19] P. C. Hansen, "The truncated SVD as a method for regularization," *BIT Numerical Mathematics*, vol. 27, no. 4, pp. 534–553, Dec. 1987. [Online]. Available: http://dx.doi.org/10.1007/BF01937276

[20] D. Gough, "The success story of the transfer and development of methods from geophysics to helioseismology," in *Inverse Methods*, ser. Lecture Notes in Earth Sciences, B. Jacobsen, K. Mosegaard, and P. Sibani, Eds. Springer Berlin / Heidelberg, 1996, vol. 63, pp. 1–31, 10.1007/BFb0011758. [Online]. Available: http://dx.doi.org/10.1007/BFb0011758

[21] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*. John Wiley & Sons, New York,, 1977.