The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Complementarity-based Dynamic Simulation for Kinodynamic Motion Planning

Nilanjan Chakraborty
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
nilanjan@cs.cmu.edu

Srinivas Akella
Department of Computer Science
University of North Carolina
Charlotte, NC 28223
sakella@uncc.edu

Jeff Trinkle
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York 12180
trink@cs.rpi.edu

***Abstract*— In this paper, we present the use of complementarity-based dynamic simulation algorithms for kinodynamic motion planning. Dynamic simulation algorithms are used as local planning methods in sampling-based motion planning algorithms to find inputs that ensure the resulting trajectory satisfies the dynamics constraints. However, the inputs are not guaranteed to give collision-free path segments. The inputs, chosen either by random sampling or from a discretization of the available inputs, are rejected if the path segment is not collision free. In cluttered environments, finding a feasible input is difficult and sensitive to the duration Δt of application of the input, and to the discretization resolution of the input set. When the collision constraints (or any inequality constraints on the state of the robot) are modeled as a set of complementarity constraints, the dynamic simulation algorithm gives a path segment that touches the obstacles and a set of *contact forces* whenever the robot makes contact with the obstacles. The sum of the chosen input forces and the contact forces transformed to the input space gives a control input that guarantees a collision-free path segment (provided it is within the actuator bounds). Thus in cluttered environments, using a complementarity-based dynamic simulation algorithm, we can find a feasible input that is relatively insensitive to the choice of Δt and the discretization resolution of the input set. We present simple simulation examples showing the advantages of our algorithm in cluttered environments.**

I. Introduction

The motion planning problem for a single robot subject to kinematics, dynamics, and collision constraints can be formulated as an optimal control problem [11], [6]. However, in practice it is possible to solve this problem only for very simple cases. Finding an exact time-optimal trajectory for a point mass (with bounded velocity and acceleration) moving among polyhedral obstacles in $\mathbb{R}^3$ has been proven to be NP-hard [5]. Therefore, sampling-based randomized techniques [7], [12], [13], [11] that try to provide inputs such that the robot's state satisfies differential constraints (kinematic and dynamic constraints) and collision constraints are now prevalent. The basic idea is to form a graph-based representation of the state space starting from some state that satisfies the constraints. In the basic algorithm, an input is randomly chosen from the set of inputs to act for some time Δt, the equations of motion are integrated by calling the dynamic simulation module, and the state at time $t + \Delta t$ is obtained. If the entire path is collision free, the state is added as a node to the graph and the input is stored; otherwise, another input is chosen at random from the input set. The process is then repeated until the start and goal states belong to the same connected component of the graph. Any path on this graph from the start to the goal state gives a feasible motion plan. Note that the sampling is done over the space of possible inputs (or actions) to the system and the output of such sampling-based algorithms is a sequence of piecewise inputs.

In this paper, we present the use of complementarity-based algorithms for dynamic simulation and point out the advantages of such methods in the context of sampling-based kinodynamic motion planning problems where collision avoidance is a key requirement. The role of the dynamic simulation algorithms in current sampling-based motion planning methods is to ensure that the state trajectories obtained satisfy the differential constraints. Our approach, in addition to ensuring satisfaction of differential constraints, also ensures that we obtain state trajectories and inputs that satisfy collision constraints. The complementarity conditions encode the physical constraint that two objects cannot interpenetrate. If we use complementarity-based models for dynamic simulation, and simulate for a time Δt[1] using a given input we will get non-zero contact force values when the objects touch in that time interval; we can use a prescribed safety distance so that the forces become non-zero when the objects are a safe distance apart. The end state at time $t + \Delta t$ is collision free. Moreover, we can obtain the input that ensures that the whole path is collision-free by adding the (suitably transformed) virtual contact forces to the input forces. This has the following advantages: (a) We always get a collision-free path segment along with the corresponding input forces (provided the input forces do not violate actuator force constraints). Thus when the path is not collision free, we do not waste any computation unlike in the conventional method. (b) Using our method, we can obtain feasible inputs that are outside the set of primitives, thus essentially enhancing the input set available for planning, even when we start with a simple set of hand designed primitives. When the input set is just the set of motion primitives, there may be no input in this set of primitives

[1]This is not the same as the time step used in the dynamic simulation module, i.e., numerical integration of the differential equations which may be much smaller than Δt.

that gives a feasible path for that time step.

The outline of the rest of this paper is as follows: In Section II we provide a brief overview of the literature on robot motion planning with differential constraints. In Section III we provide a brief description of the complementarity-based dynamic state evolution models and sampling-based motion planning algorithms with differential constraints. In Section IV we describe the changes we propose to the basic RRT (rapidly exploring random tree) algorithm and discuss its advantages. In Section V we present two simple simulations to illustrate the key features of our algorithm. Finally, we present our conclusions and outline future work.

II. Related Work

The problem of finding an exact time-optimal trajectory, from a given start state to goal state, for a point mass with bounded velocity and acceleration moving among polyhedral obstacles is NP-hard [5]. Approximation algorithms have been proposed for systems with decoupled dynamics that are polynomial in the combinatorial complexity of the number of obstacles [4], [3]. However, these algorithms are exponential in the dimension of the configuration space. For practical purposes there are three basic approaches for kinodynamic motion planning problems: (a) Decoupled approach [18] (b) Potential field based methods [8], [9], [16] (c) Sampling-based approaches [12], [7]. In the decoupled approach the problem is divided into a path planning and a trajectory planning problem. In the path planning stage a path is obtained for the robot that satisfies the geometric (collision) constraints and in the trajectory planning stage this path is converted to a trajectory while satisfying the dynamics constraints. Both the path planning problem and the trajectory planning problem have been studied extensively [1], [11]. The limitation of this method is that it may not be possible to convert the path into a trajectory satisfying all the dynamics constraints.

In potential field methods, an artificial attractive field is generated in the configuration space with the goal as its minimum and repulsive fields are generated around the obstacles. The robot then follows the steepest gradient on the resultant field to reach the goal. However, the robot may get stuck in local minima and there is no guarantee that the robot will reach its goal (except for a class of environments called star-shaped environments [16], where the potential function is called a navigation function). In other words, neither of the two approaches described above are *complete*, i.e., they may not find a feasible solution even though one exists.

Sampling-based approaches build a graph-based representation of the free state space such that the start and goal states belong to the same connected component of the graph. There are two main approaches (notable exception being [10]) in the sampling-based randomized algorithms literature: (a) Rapidly Exploring Random Tree (RRT) algorithm [12][2] and (b) Expansive space tree algorithm [7] (terminology adopted from [1]). The basic difference between the two methods is in the procedure used to bias the connectivity graph (tree) towards unexplored regions of the state space (i.e., choice of the node to expand). The RRT algorithm generates a random sample and tries to connect the nearest node on the existing partial tree towards the random node. The algorithm in [7] maintains a weight at the nodes of the tree based on the number of samples that lie within a certain radius of the node. The algorithm then tries to expand the nodes having lower weight. There are various variations of these two basic approaches [15], [13], but all of these are concerned with heuristics on the choice of nodes to be expanded and the number of trees to be maintained in the exploration of the state space. For example, bi-directional RRT [13] is a variation of RRT in which two trees are grown, one from the initial state and one from the goal state. The common feature of all these algorithms is that they choose a random input and use numerical integration of the differential constraints followed by collision checking when expanding a node. *In our work we propose a modification to the local planning step of expanding a node and not the decision of which node to expand*. We will present our discussion based on the basic RRT algorithm. The changes that we propose to the basic RRT are also applicable to all the other variations.

[2]Lavalle distinguishes between RRTs and rapidly exploring dense trees (RDT) in [11] where the distinction takes into consideration deterministic sampling. This distinction is not important here and we will discuss in the context of RRTs. All of the discussion holds for deterministic sampling.

III. Background and Problem Formulation

In this section, we present the complementarity-based model of collision-free dynamic state evolution of a robot and pose the kinodynamic motion planning problem in a formal setting. Let Q be the *configuration space* of dimension d, X be the *state space*, and U be the input (or action) space of the robot. We assume X to be a smooth manifold and $U \subset \mathbb{R}^m$ to be a bounded set. Let $\mathbf{x} = (\mathbf{q}, \boldsymbol{\nu})$ be the state of the robot where $\mathbf{q}$ is the configuration of the robot and $\boldsymbol{\nu} = \dot{\mathbf{q}}$ is the generalized velocity. The Lagrangian equations of motion of the robot are [14]

$$\mathbf{M}(\mathbf{q})\dot{\boldsymbol{\nu}} + \mathbf{C}(\mathbf{q}, \boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{V}(\mathbf{q}) = \boldsymbol{\tau} \tag{1}$$

where $\mathbf{M}(\mathbf{q})$ is the mass matrix of the robot, $\mathbf{C}(\mathbf{q}, \boldsymbol{\nu})$ is the coriolis matrix and $\mathbf{V}(\mathbf{q})$ arises due to gravity and other potential forces acting on the robot. If the robot is a rigid body, $\boldsymbol{\nu}$ may be chosen to be the concatenated vector of linear and angular velocities, whereas $\mathbf{q}$ may contain some parametrization of orientation like euler angles, quaternions, etc. In that case $\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\boldsymbol{\nu}$, where $\mathbf{G}^T\mathbf{G}$ is always a multiple of the identity matrix. The generalized force, $\boldsymbol{\tau}$, is the vector of all external forces on the robot and can be written as

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\mathrm{app}} + \boldsymbol{\tau}_{\mathrm{con}} + \boldsymbol{\tau}_{\mathrm{oth}} \tag{2}$$

where $\boldsymbol{\tau}_{\mathrm{app}}$ are the actuator inputs to the robots, $\boldsymbol{\tau}_{\mathrm{con}}$ are the contact forces (transformed to actuator space) acting on the robot, and $\boldsymbol{\tau}_{\mathrm{oth}}$ is the vector of any other types of forces acting on the robot. Note that inequality constraints on the states of the robot can be implemented by applying a virtual force $\boldsymbol{\tau}_{\mathrm{oth}}$.

The contact forces are usually unknown and have to be determined as part of the simulation process. The magnitude of the normal component of the contact forces is zero if the distance between the two objects is greater than zero and non-zero if the bodies are in contact. Thus, at each potential contact, the product of the magnitude of the normal contact force and distance between the two objects is always zero, or they are orthogonal to each other. This is encoded by the following equation (also known as a complementarity constraint)

$$0 \leq \lambda_{i\mathrm{n}} \perp \psi_{i\mathrm{n}}(\mathbf{q}, t) \geq 0 \tag{3}$$

where $\perp$ denotes orthogonality, $\psi_{i\mathrm{n}}$ is a signed distance function or *gap function* for the ith contact with the property $\psi_{i\mathrm{n}}(\mathbf{q}, t) > 0$ for separation, $\psi_{i\mathrm{n}}(\mathbf{q}, t) = 0$ for touching, and $\psi_{i\mathrm{n}}(\mathbf{q}, t) < 0$ for interpenetration. The wrench due to the normal contact force acting at any point on the body (say the center of gravity of the link) is $\mathbf{w}_i \lambda_{i\mathrm{n}}$, where $\mathbf{w}_i$ is a 6×1 concatenated vector of the negative of the unit normal at the contact point, $-\hat{\mathbf{n}}_i$, and $-\mathbf{r}_i \times \hat{\mathbf{n}}_i$, $\mathbf{r}_i$ being the vector from the center of gravity to the contact point. The generalized contact forces can be written as a sum of the generalized normal forces and the generalized friction forces, i.e., $\boldsymbol{\tau}_{\mathrm{con}} = \boldsymbol{\tau}_{\mathrm{n}} + \boldsymbol{\tau}_{\mathrm{f}}$. Since we shall be dealing with only virtual contacts in this paper, we can disregard the friction forces. The generalized normal forces at each contact i, $i = 1 \ldots n_c$, where n_c is the number of contacts are given by $\boldsymbol{\tau}_{i\mathrm{n}} = \mathbf{J}_i^T \mathbf{w}_i \lambda_{i\mathrm{n}}$ where $\mathbf{J}_i$ is the Jacobian upto ith contact point.

The state evolution model that satisfy the dynamics constraints and take into account the constraints on the state like collision avoidance, joint limits, and velocity limits is:

$$\begin{aligned} \mathbf{M}(\mathbf{q})\dot{\boldsymbol{\nu}} + \mathbf{C}(\mathbf{q}, \boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{V}(\mathbf{q}) &= \boldsymbol{\tau}_{\mathrm{app}} + \sum_{i=1}^{n_c} \mathbf{J}_i^T \mathbf{w}_i \lambda_{i\mathrm{n}} + \boldsymbol{\tau}_{\mathrm{oth}} \\ 0 \leq \lambda_{i\mathrm{n}} \perp \psi_{i\mathrm{n}}(\mathbf{q}, t) &\geq 0 \\ \mathbf{f}(\mathbf{q}, \boldsymbol{\nu}, t) &\leq 0 \end{aligned} \tag{4}$$

The evolution of the state as a function of time, t, denoted by $\mathbf{x}(t) = (\mathbf{q}(t), \boldsymbol{\nu}(t))$, is called the *state trajectory* of the robot and the time history of the applied control input, $\boldsymbol{\tau}(t)$, is called the *action trajectory*. The state trajectory is called a *feasible state trajectory* if it satisfies Equation 4, i.e., it satisfies the equations of motion and avoids collision with obstacles at each time instant. We denote the workspace of the robot by $\mathcal{W} \subseteq \mathbb{R}^n, n = 2, 3$, the set representing the robot by $\mathcal{A}$, and the set representing the obstacles by $\mathcal{O}$. The motion planning problem with differential constraints can be formally stated as:

Input: The sets $\mathcal{W}, \mathcal{A}, \mathcal{O}, X, U$ as defined above, feasible initial state $\mathbf{x}_I$, feasible set of goal states $\mathbf{X}_G$, and a possibly unbounded interval $T = [0, T_f]$.

Output: An action trajectory $\boldsymbol{\tau}(t)$ for which the state trajectory $\mathbf{x}(t)$ is feasible, i.e., satisfies Equation 4, $\mathbf{x}(0) = \mathbf{x}_I$, and there exists some $t > 0$ such that $\boldsymbol{\tau}(t) = \boldsymbol{\tau}_T$ and $\mathbf{x}(t) \in \mathbf{X}_G$, where $\boldsymbol{\tau}_T$ is the termination input.

A. Discrete-time collision-free state evolution model

We now write down the discretized equations of motion for numerical integration. We use a velocity-level formulation and an Euler time-stepping scheme to discretize Equation 4. Let t_ℓ denote the current time, and h be the time step. Use the superscripts ℓ and $\ell+1$ to denote quantities at the beginning and end of the ℓth time step respectively. Using $\dot{\boldsymbol{\nu}} \approx (\boldsymbol{\nu}^{\ell+1} - \boldsymbol{\nu}^{\ell})/h$, $\dot{\mathbf{q}} \approx (\mathbf{q}^{\ell+1} - \mathbf{q}^{\ell})/h$, and $h\boldsymbol{\lambda}_{\mathrm{n}} = \mathbf{p}_{\mathrm{n}}$ we get the following discrete time system. The discretized equations of motion (considering only the collision constraints here for exposition) to be satisfied at each time step are [17]:

$$\begin{aligned} \mathbf{M}\boldsymbol{\nu}^{\ell+1} &= \mathbf{M}\boldsymbol{\nu}^{\ell} + h(\mathbf{W}_{\mathrm{n}}\boldsymbol{\lambda}_{\mathrm{n}}^{\ell+1} + \boldsymbol{\lambda}_{\mathrm{app}}^{\ell} + \boldsymbol{\lambda}_{\mathrm{vp}}^{\ell}) \\ \mathbf{q}^{\ell+1} &= \mathbf{q}^{\ell} + h\boldsymbol{\nu}^{\ell+1} \\ 0 \leq \mathbf{p}_{\mathrm{n}}^{\ell+1} &\perp \psi_{\mathrm{n}}(\mathbf{q}^{\ell+1}) \geq 0 \end{aligned} \tag{5}$$

where $\boldsymbol{\lambda}_{\mathrm{n}}$ is the concatenated vector of contact forces and each row of $\mathbf{W}_{\mathrm{n}}$ is $\mathbf{J}_i^T \mathbf{w}_i$, with $i = 1 \ldots n_c$, n_c being the number of contacts. We note that in Equation 5, if we approximate the distance function $\psi_{\mathrm{n}}(\mathbf{q}^{\ell+1})$ with a Taylor's series expansion, we will have a (mixed) linear complementarity system (i.e., (M)LCP, see [2] for a formal definition) to be solved at each step. The variables to be solved for at each step are $\mathbf{q}^{\ell+1}$, $\boldsymbol{\nu}^{\ell+1}$ and $\boldsymbol{\lambda}_n^{\ell+1}$. Since the mass matrix is positive definite, the matrix defining the LCP is positive definite. This implies that there exists a unique solution to the LCP at each time step that can be obtained in polynomial time (in contrast to general LCPs where the time taken for finding a solution may be exponential in the number of variables [2]).

B. Sampling-based kinodynamic motion planning

Algorithm 1 RRT algorithm

```
BUILDRRT(x_I)
Γ.init(x_I);
for k=1 to K do
    x_rand ← RANDOMSTATE();
    EXTEND(Γ, x_rand);
end for

EXTEND(Γ, x)
x_near ← NEARESTNEIGHBOR(x, Γ);
if NEWSTATE(x, x_near, x_new, u) then
    Γ.addvertex(x_new);
    Γ.addedge(x_near, x_new, u);
    if x_new = x then
        return Reached;
    else
        return Advanced;
    end if
end if
return Trapped;
```

The basic RRT algorithm is given in Algorithm 1 [12]. The tree representing the free configuration space is denoted by Γ. The algorithm starts from the initial node and at each iteration a new state that is biased towards a random state

$\mathbf{x}_{\text{rand}}$ is attempted to be added to the RRT (by calling the function EXTEND). The choice of the nearest vertex on the already existing tree $\mathbf{x}_{\text{near}}$ in function NEARESTNEIGHBOR depends on the definition of a metric in the state space. We propose to make modifications to the function NEWSTATE, which typically consists of the following steps:

1) An input $\mathbf{u} \in U$ is applied to the robot at state $\mathbf{x}(t) = \mathbf{x}_{\text{near}}$ for time Δt (note the difference between h and Δt, $h = N\Delta t$, where N is the number of integration steps) and the equations of motion of the robot are numerically integrated to obtain the state trajectory from $\mathbf{x}(t)$ to $\mathbf{x}(t + \Delta t)$. The time Δt is an input parameter to be chosen. The geometric collision constraints are checked after each step in the numerical integration and the simulation is terminated if there is a collision.
2) The input $\mathbf{u}$ may be chosen at random, or if the set U is finite then step 1 can be repeated for all possible inputs. In the former case, the state at time $t + \Delta t$ is added as a node to the tree and $\mathbf{u}$ is stored if the trajectory is collision-free. In the latter case, among all the collision-free trajectories, the one where $\mathbf{x}(t + \Delta t)$ is *nearest* to the random state is chosen and the corresponding input is stored.

Although the different kinodynamic planners differ on the details of which node to expand, the dynamic simulation subroutine in function NEWSTATE is virtually identical in all kinodynamic planners. Irrespective of the details, the performance of all sampling-based algorithms are limited by the following:

1) The geometric constraints (or collision constraints) are not considered during the dynamic simulation step (i.e., during each step of the numerical integration).
2) A finite set of inputs and a time of application of the input (Δt) is usually used in the dynamic simulation step. At each step, the results obtained will be dependent on Δt and the input set chosen; thus the algorithm may not give a collision-free trajectory for the step even if one exists.

In the next section, we argue that the use of complementarity-based dynamic simulation (i.e., using Equation 4 instead of Equation 1) can help in mitigating the above limitations.

IV. Complementarity-based Planner and its Advantages

In this section we propose the use of Equation 4 as the discrete-time dynamic state evolution model that takes into account the collision constraints. Since we want the bodies to avoid collision we can rewrite the complementarity constraint for collision in Equation 5 as

$$0 \leq \mathbf{p}_{\text{n}}^{\ell+1} \perp \psi_{\text{n}}(\mathbf{q}^{\ell+1}) - \epsilon \geq 0 \tag{6}$$

where $\epsilon \geq 0$ is a parameter specifying the safety distance to the obstacle that the robot must satisfy. Conceptually, we can think of this as a *virtual obstacle* that the robot can just touch. This ensures that any trajectory that satisfies Equation 5 with the collision constraints modified as above is collision free.

During the numerical integration of the equations of motion (i.e., from the solution of Equation 5) we obtain the contact wrenches ($\mathbf{w}_i\boldsymbol{\lambda}_{i\text{n}}$ for the ith contact) that arise when the robot comes in contact with the virtual obstacle. Thus, at the end of the simulation we have a time history of the *virtual contact wrenches* over the interval $[t, t + \Delta t]$ along with a trajectory where the robot maintains a safe distance from the obstacle boundary. If we transform the virtual contact wrenches to the actuator space and add it to the input, then we obtain an input ($\boldsymbol{\tau}_{\text{app}} + \sum_{i=1}^{n_c} \mathbf{J}_i^T \mathbf{w}_i \boldsymbol{\lambda}_{i\text{n}}$) that produces safe trajectories. If the input is within the set U, then we have a feasible input. Usually the set U is defined by simple upper and lower bound constraints. Moreover, in the equations of motion we have the values of contact wrenches transformed to the actuation space (i.e., $\mathbf{J}_i^T \mathbf{w}_i \boldsymbol{\lambda}_{i\text{n}}$). Thus, after each step of the numerical integration process, we can check if the input required to satisfy the safety distance is feasible or not at very little computational cost and terminate the simulation if the input becomes infeasible. Moreover, we can also check at each step of the numerical integration if the state of the robot is changing; if not, we can terminate the simulation because a stationary state implies that the robot is stuck when using the chosen input. Thus our proposed modification (of considering the geometric constraints during the dynamic simulation step) to the sampling-based algorithms has the following advantages:

1) We either get a collision-free path and corresponding feasible inputs at the end of the dynamic simulation step, or can terminate the simulation early if the input required to obtain a collision-free trajectory is infeasible. Moreover, we also get the information of whether the robot gets stuck (i.e., its state does not change over two consecutive iterations) for the chosen input and can terminate the simulation after the robot reaches that state. At this state, any input in the conic hull of the negative contact wrenches will not change the state of the robot. Thus, we can easily check if a chosen input vector changes the state of the robot when it gets stuck. Alternatively, we can easily find a vector that releases the robot from the stuck state by checking if it lies in the conic hull formed by the contact wrenches.
2) When the input set of primitives is a finite set that is a subset of the feasible input set obtained by discretization (say), there may be no input in the discretized set that provides a collisionfree trajectory in the conventional algorithms. However, with our algorithm, we may be able to obtain feasible inputs that lie outside the set of the primitives. Thus, we are not limited to the set of inputs that we start out with.

Effect of choice of Δt and input: As is evident from the discussion on sampling based algorithms in Section III-B the simulation time Δt is a parameter to be chosen. Moreover, when the input set is given as a compact set, the number of possible inputs is infinite. Thus, at each simulation step it is not possible to check all possible inputs, and a subset of the

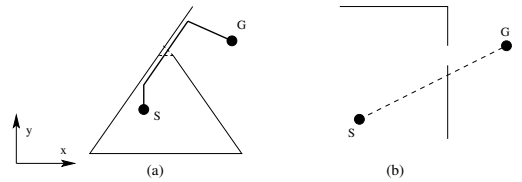

Fig. 1. Schematic sketch of a situation where it is difficult to find a path using motions only along two directions.

available inputs is chosen at each step. This subset may be randomly drawn or formed by discretizing the available input set. We illustrate the effect of this with a simple example shown in Figure 1(a). We consider a point particle that has to move from the start position S to goal G. The particle can be independently actuated along the x and y axis (horizontal and vertical directions in the figure). As can be seen from Figure 1(a), if we use Equation 1 for the dynamics the planning algorithms will find a solution if and only if the value of Δt is chosen small enough and the input is small enough such that an end state lies in the dashed triangular area. In contrast, the solution from our algorithm (shown by the bold line) is independent on the choice of Δt. In our case the choice of Δt determines how fast the solution is found but not whether the solution can be found or not.

We also note that in our algorithm there is also more flexibility to the choice of inputs. In the example in Figure 1(a), we will get a solution if we choose the maximum possible inputs along the two directions and an input directly towards the goal. This is what we use in our implementations; an input along each of the independently actuated coordinates and an input directly towards the goal. Using the inputs along the coordinates only is not enough as can be seen in Figure 1(b). When the normal to the obstacle at the point of impact directly opposes the input the robot gets stuck.

When we have an input that takes the robot directly towards the goal, in our algorithm, the robot may reach the goal even if the line joining the start and goal intersects an obstacle. This happens if the obstacles are arranged such that the line joining the robot's current state to the goal is not normal to any of the obstacles it is intersecting with. For example, in Figure 1(b), if the robot moves along the line joining S and G it will reach the goal. However, in Figure 1(a), the robot will not reach the goal. Note that if the projection of the motion of the robot in a time-step on the line joining the start and goal is always positive then the robot always moves toward the goal. When the line joining the robot's state and goal is not normal to the surface of the obstacles it intersects, the above condition is true. Hence the robot always reaches its goal. The input required to reach the goal is obtained from the applied input plus the transformed contact wrenches.

Implication for Cluttered Environments: The choice of the simulation time Δt and the input set becomes critical in cluttered environments because the probability of collision is high. In such environments, it is difficult to find a feasible input, and sampling-based algorithms for kinodynamic planning can perform poorly in such environments. Thus, in cluttered environments, i.e., environments characterized by the presence of narrow passages, our modification would improve the performance of sampling based algorithm since it always finds a collision-free trajectory and the corresponding input. More specifically, *if we arrive near a narrow passage during the construction of the graph, then we can easily find inputs that will take us through the passage*. Our algorithm does not say anything about how to get near the narrow passage, which is an open problem.

Completeness properties of our algorithm: The change we are proposing to the existing sampling-based randomized planning techniques enhances the reachability set at each point. In other words, for any variation of the randomized sampling-based techniques, the set of reachable points from any given state, using our complementarity-based dynamic simulation algorithm is a superset of the set of reachable points using the conventional dynamic simulation. Thus all the probabilistic completeness [7], [12] results proved for sampling-based algorithms still hold for our variation.

V. Simulation Results

In this section we present some simulation examples illustrating our method. The first example shows a scenario where a point robot reaches the goal under the action of a single force directed towards the goal. The obstacles hinder the motion along the direction of motion. The environment in this case is not a star shaped environment and hence it is not possible to design navigation functions that ensure that the goal can be reached. The second example is that of a $2R$ planar manipulator moving among three obstacles. The problem is hand-designed so as to contain two narrow passages in the free configuration space connected by a relatively large free space region. The start and the goal configurations are placed at the two ends of the narrow passage. In this case we use our modified version of the RRT algorithm to obtain the solution trajectory.

A. Example 1: Planar point robot

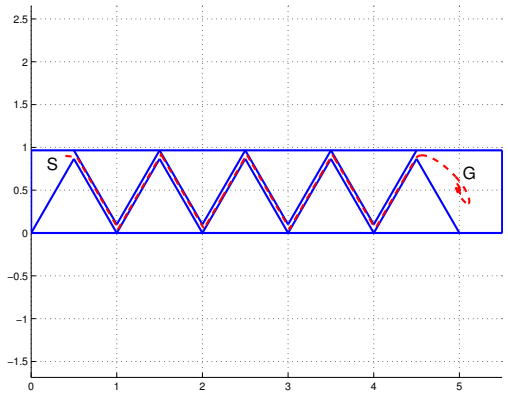

Fig. 2. Path of a point robot moving through a narrow passage formed by the sawtooth shaped obstacles.

This example illustrates planning with a single force towards the goal (obtained with a potential field with minima

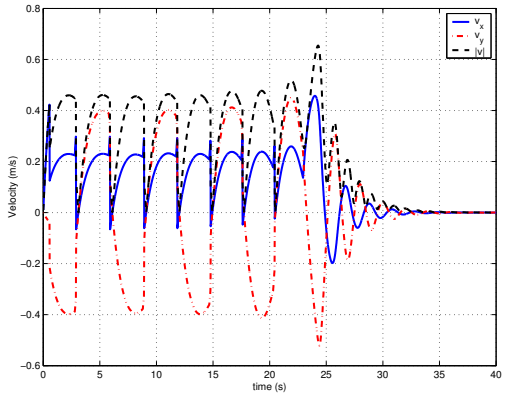

Fig. 3. Velocity of the point robot.

at the goal) in the presence of obstacles. We consider a $2D$ point robot moving in a rectangular environment with sawtooth shaped obstacles (see Figure 2). The sawtooth shaped obstacles form a narrow passage in the environment and the start and goal configuration are located on opposite sides of the obstacles. Let $\mathbf{q} = (x, y)$ be the configuration and $\boldsymbol{\nu} = \dot{\mathbf{q}}$ be the velocity of the robot. The discrete time dynamic model for state evolution of the system is

$$\begin{aligned} 0 &= -\mathbf{M}\boldsymbol{\nu}^{\ell+1} + \mathbf{M}\boldsymbol{\nu}^{\ell} + \mathbf{W}_{\mathrm{n}} p_{\mathrm{n}} + \mathbf{p}_{\mathrm{app}} \\ 0 &\leq p_{\mathrm{n}} \perp \psi_{\mathrm{n}}(\mathbf{q}^{\ell}) + \mathbf{W}_{\mathrm{n}}^{T} \boldsymbol{\nu}^{\ell+1} \geq \epsilon \end{aligned} \tag{7}$$

where $\mathbf{M}$ is a 2×2 diagonal matrix with the mass m of the robot as the diagonal entries, $\mathbf{W}_{\mathrm{n}}$ is a 2×1 vector giving the normal at the contact point, $\psi_n(\mathbf{q}^{\ell})$ is the distance of the robot to the obstacle at time ℓ, and ϵ is the safety distance from the obstacle. Both the distance and the normal at time ℓ are obtained from a collision detection algorithm. The applied impulse is given by:

$$\mathbf{p}_{\mathrm{app}} = -h(K_p(\mathbf{q}^{\ell} - \mathbf{q}_{\mathrm{g}}) + K_d \dot{\mathbf{q}}^{\ell}) \tag{8}$$

where K_p and K_d are diagonal matrices with positive entries, h is the time step, and $(\mathbf{q}, \boldsymbol{\nu}) = (\mathbf{q}_g, \mathbf{0})$ is the goal state. The unknowns in Equation 7 are $\boldsymbol{\nu}^{\ell+1}$ and p_{n}. Thus, we have a system of 3 equations and 3 unknowns.

For generating the results, we have used the following data: $m = 1, \epsilon = 0.01, h = 0.01$, each diagonal entry of K_p and K_d is 1, initial state is $(0.4, 0.9, 0, 0)$ and goal state is $(5, 0.5, 0, 0)$. We simulated the system in Equation 7 using $\mathbf{p}_{\mathrm{app}}$ given by Equation 8. The contact impulse obtained from the simulation $\mathbf{W}_{\mathrm{n}} p_n$ is then added to the applied impulse to obtain the collision-free input impulse. The path taken by the point robot to reach the goal under the action of the collision-free input impulse is shown in Figure 2. The approach of the robot to the goal state (when the x-coordinate of the robot is greater than 4.5) clearly shows the effect of dynamics. The robot does not go straight to the goal because of its non-zero momentum orthogonal to the goal direction. Ignoring the dynamics would give a solution path straight to the goal at the end. Figure 3 shows the velocities of the robot as it moves towards the goal. When the robot reaches the safety distance from the wall, the x-component of its velocity drops to 0 and then becomes less than 0 as the y-momentum carries it downward along the obstacle. Near the end, when there are no obstacles, the velocities decay to 0 because of the damping term in the applied force. The basic RRT method could not find a solution to this problem with $\Delta t = 0.1$ seconds and the input set consisting of a random vector, a force towards the goal given by Equation 8, and

$$\mathbf{u} = \{\begin{pmatrix}1\\0\end{pmatrix} \begin{pmatrix}-1\\0\end{pmatrix} \begin{pmatrix}0\\1\end{pmatrix} \begin{pmatrix}0\\-1\end{pmatrix}\} \tag{9}$$

Note that this is an illustrative example only, and it is easy to find start and goal positions in this example that cannot be reached by only a single force towards the goal. However, using the force towards the goal along with the input set in Equation 9 we can reach the goal.

B. Example 2: 2R Manipulator

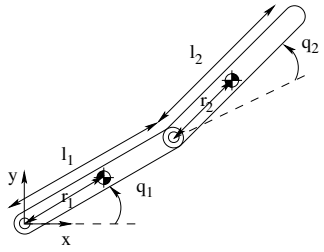

Fig. 4. A planar manipulator with 2 revolute joints ($2R$ manipulator).

The second example is that of a $2R$ manipulator (a planar manipulator with 2 revolute joints, see Figure 4) that has to move from a start to goal configuration in a gravity free environment with three circular obstacles (see Figure 9). The free space of the manipulator relevant to the problem along with the start and goal configurations is shown in Figure 5. The joint angles $\mathbf{q} = (q_1, q_2)$ are the configuration of the robot and $\boldsymbol{\nu} = \dot{\mathbf{q}}$ are the joint angle rates. The discrete time dynamic model for state evolution of the system is

$$\begin{aligned} 0 &= -\mathbf{M}\boldsymbol{\nu}^{\ell+1} + \mathbf{M}\boldsymbol{\nu}^{\ell} - \mathbf{p}_{\mathrm{vp}} + \mathbf{W}_{\mathrm{n}}\mathbf{p}_{\mathrm{n}} + \mathbf{p}_{\mathrm{app}} \\ 0 &\leq \mathbf{p}_{\mathrm{n}} \perp \psi_{\mathrm{n}}(\mathbf{q}^{\ell}) + \mathbf{W}_{\mathrm{n}}^{T} \boldsymbol{\nu}^{\ell+1} \geq \epsilon \end{aligned} \tag{10}$$

where the mass matrix and Coriolis force term are given by

$$\begin{aligned} \mathbf{M} &= \begin{bmatrix} \alpha + 2\beta\cos(q_2) & \delta + \beta\cos(q_2) \\ \delta + \beta\cos(q_2) & \delta \end{bmatrix} \\ \mathbf{p}_{\mathrm{vp}} &= \begin{bmatrix} -\beta\sin(q_2)\dot{q}_2^{\ell} & -\beta\sin(q_2)(\dot{q}_1^{\ell} + \dot{q}_2^{\ell}) \\ \beta\sin(q_2)\dot{q}_1^{\ell} & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1^{\ell} \\ \dot{q}_2^{\ell} \end{bmatrix} \\ \alpha &= \mathcal{I}_{z1} + \mathcal{I}_{z2} + m_1 r_1^2 + m_2(l_1^2 + r_1^2) \\ \beta &= m_2 l_1 r_2 \\ \delta &= \mathcal{I}_{z2} + m_2 r_2^2 \end{aligned} \tag{11}$$

where $m_i, l_i, r_i, \mathcal{I}_{zi}, i = 1, 2$, are the mass, length, distance of center of gravity (cg) from joint frame, and moment of inertia about an axis passing through the cg and perpendicular to the plane for the two links respectively. The vector of contact impulse magnitudes in Equation 10 is of size 6×1 and that of $\mathbf{W}_{\mathrm{n}}$ is 2×6. Each column of $\mathbf{W}_{\mathrm{n}}$ is $\mathbf{J}_k^T \hat{\mathbf{w}}_k$ where $\hat{\mathbf{w}}_k$ is the 2×1 unit normal vector at the contact point and

J_k is the 2×2 Jacobian up to the contact point. Depending upon the link in contact the Jacobian is

$$J^{\text{link1}} = \begin{bmatrix} x_{c1} & 0 \\ y_{c1} & 0 \end{bmatrix} \quad J^{\text{link2}} = \begin{bmatrix} x_{c2} & x_{c2} - l_1 \cos(q_1) \\ y_{c2} & y_{c2} - l_1 \sin(q_1) \end{bmatrix}$$

The input set consists of a force directed towards the goal given by Equation 8 and those given by Equation 9. From the initial state, we apply these 5 inputs. We repeat this procedure starting with the states obtained at the previous level till the robot reaches the goal. If the robot gets stuck and a particular input does not move it from that state, we reject the input after two time-steps of dynamic simulation. Note that this is a very basic version of RDT where we are growing the tree by expanding all the nodes. We obtain a solution to our problem in three iterations. The number of nodes in the tree is 27. In our simulations we have used the following data: $m_1 = m_2 = 1,\ l_1 = l_2 = 1,\ r_1 = r_2 = 0.5, \Delta t = 2\text{s}, \text{h} = 0.01\text{s}$, initial state $[0.22\pi, 1.94\pi, 0, 0]$, and goal state $[0.5\pi, 0.1\pi, 0, 0]$. The proportional and derivative gains, K_p and K_d, in Equation 8 are assumed to be 2×2 diagonal matrices. Each diagonal entry of K_p is 3 and K_d is 4. The basic RRT could not find a solution to this problem with $\Delta t = 0.1$ seconds.

Figure 5 shows the collision-free path of the robot in configuration space that we obtained. Figure 6 shows the variation of joint angle rates in our solution as the manipulator moves from start to goal state. Figures 7(a) and 8(a) show the input torques at joint 1 and joint 2 respectively, during the search stage. The contact impulses projected to the actuator space are shown in Figures 7(b) and 8(b). The final collision-free input impulses at each joint that takes the manipulator from start to goal state are shown in Figures 7(c) and 8(c). Figures 5, 9, and 10 show some snapshots of the path of the manipulator in configuration space and work space.

Fig. 5. Configuration space of the 2R manipulator. The white region shows the free space. The black squares show configurations along the collision-free trajectory. The positions of the manipulator in the workspace corresponding to the configurations are shown in Figures 9 and 10. The points S and G are the start and goal configurations respectively.

VI. Conclusion and Future Work

In this paper, we have shown the use of complementarity-based dynamic simulation in kinodynamic motion planning problems and pointed out its advantages in solving motion planning problems in cluttered environments. Since the contact constraints are directly incorporated in complementarity-based dynamic models, the numerical integration always provides us with a collision-free trajectory and the contact forces when "contact" occurs. We can thus transform the contact forces to the actuator space and obtain an input that gives us a collision-free trajectory by adding it to the input applied for the simulation. This observation can be used for kinodynamic planning in cluttered environments to reduce the sensitivity of the planning algorithm to the choice of the initial discrete input set and duration of application of the input (Δt). We have illustrated this using two simple examples. We are currently working on an implementation to illustrate our algorithm in $3D$ environments and to perform detailed comparison with existing algorithms.

Future Work: There are various directions for extending the current work. Firstly, we note that the point particle example in our paper also shows that even in the presence of obstacles along the direction of steepest gradient, using our algorithm, we can reach the goal by using a single attractive potential function at the goal. This problem cannot be solved

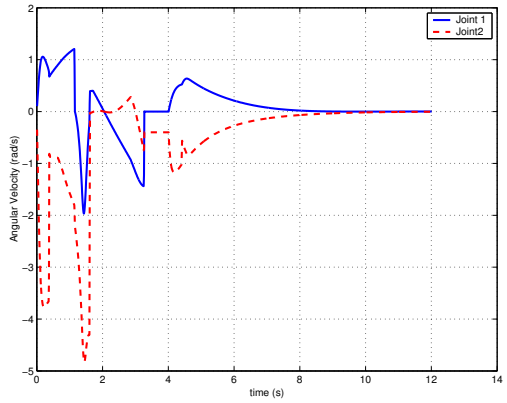

Fig. 6. Variation of the joint angle rates of the 2R robot for motion from start to goal configuration in Figure 5.

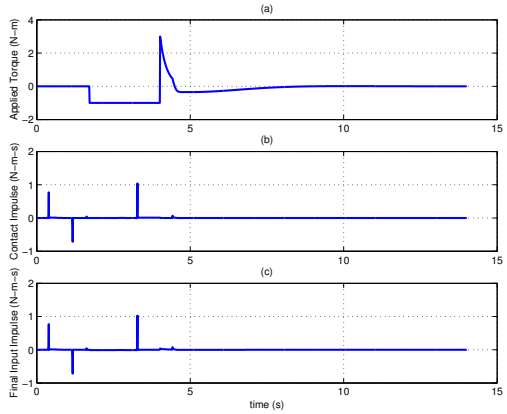

Fig. 7. (a) The applied torque at joint 1 used for dynamic simulation (b) The contact impulses transformed to joint 1 obtained from dynamic simulation using the input in subplot (a). (c) The final collision-free input torque impulse for joint 1.

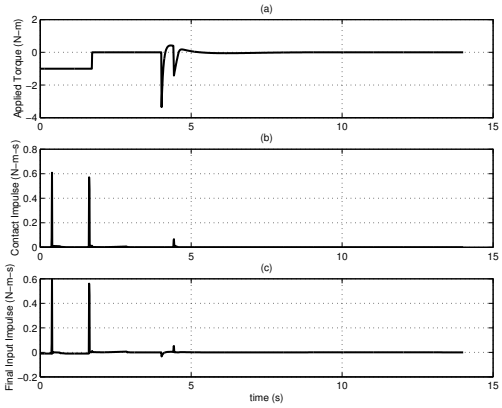

Fig. 8. (a) The applied torque at joint 2 used for dynamic simulation (b) The contact impulses transformed to joint 2 obtained from dynamic simulation using the input in subplot (a). (c) The final collision-free input torque impulse for joint 2.

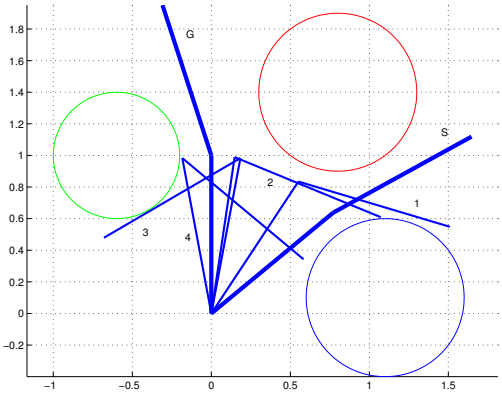

Fig. 9. The first four configurations in the path of the 2R robot for motion from start to goal configuration in Figure 5.

by the method of [16] since the environment is not star-shaped and it may not be possible to design a navigation function. Intuitively, the complementarity constraints in the dynamic model can be thought of as a non-smooth repulsive potential that is activated only when the point reaches the boundary and is zero otherwise. Even in this example, there are start and goal positions for which our algorithm will get stuck if we use just a single attractive potential. An interesting question is the connection of our method to potential field methods. Secondly, we note that we have restricted our attention to problems without contact (or for problems involving collision avoidance only). In principle our method also applies to applications where the robot may be in contact with other objects (such as in assembly planning). We also want to explore this avenue further.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation under grant CCF-0729161.

REFERENCES

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005.

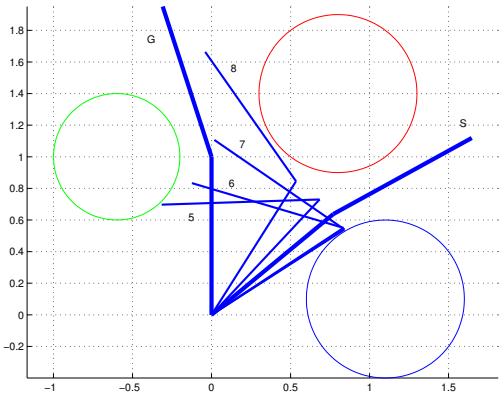

Fig. 10. The last four configurations in the path of the 2R robot for motion from start to goal configuration in Figure 5.

[2] R. W. Cottle, J. Pang, and R. E. Stone. *The Linear Complementarity Problem*. Academic Press, 1992.
[3] B. R. Donald and P. G. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open chained manipulators. *Algorithmica*, 14:480–530, 1995.
[4] B. R. Donald and P. G. Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, 14:443–479, 1995.
[5] B. R. Donald, P. G. Xavier, J. Canny, and J. Reif. Kinodynamic planning. *Journal of the ACM*, 40:1048–66, Nov. 1993.
[6] E. Gilbert and D. Johnson. Distance functions and their application to robot path planning in the presence of obstacles. *IEEE Journal of Robotics and Automation*, 1(1):21–30, March 1985.
[7] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, March 2002.
[8] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.
[9] J.-O. Kim and P. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(3):338–349, June 1992.
[10] A. M. Ladd and L. E. Kavraki. Fast tree-based exploration of state space for robots with dynamics. In *Algorithmic Foundations of Robotics VI*, pages 297–312. Springer, STAR 17, 2005.
[11] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.
[12] S. M. Lavalle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.
[13] S. M. Lavalle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.
[14] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL, 1994.
[15] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Discrete search leading continuous exploration for kinodynamic motion planning. In *Robotics: Science and Systems*, pages 326–333, Atlanta, Georgia, June 2007. MIT Press.
[16] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, October 1992.
[17] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction. *International Journal of Numerical Methods in Engineering*, 39:2673–2691, 1996.
[18] E. Yoshida, C. Esteves, I. Belousov, J.-P. Laumond, T. Sakaguchi, and K. Yokoi. Planning 3-d collision-free dynamic robotic motion through iterative reshaping. *IEEE Transactions on Robotics*, 24(5):1186–1198, Oct. 2008.