Motion Planning for Smooth Pickup of Moving Objects

Arjun Menon[1], Benjamin Cohen[2], and Maxim Likhachev[1]

Abstract— **Kinematic planning for robotic arms has been shown to be capable of planning for robust manipulation of static objects. However, this approach falls short when manipulating moving objects such as picking up a jar off of a conveyor belt at a bottling plant. The challenge in carefully picking up moving objects is that these actions require motions that do not involve large decelerations, to avoid jerking the object, as well as figuring out the proper time in which the object can be picked up. We present a search-based kinodynamic motion planning algorithm that generates a time-parameterized trajectory for both the arm and end-effector, capable of carefully picking up the object at the earliest feasible point in its trajectory. To combat the high-dimensionality of the time-parameterized kinodynamic planning problem, our approach employs informative heuristics and adaptive dynamic motion primitives. To validate our approach, we used a 7DOF manipulator on Willow Garage's PR2 robot to pickup objects off of a conveyor belt. We also provide a detailed set of results that demonstrate the planner's ability to generate consistent, low cost trajectories for manipulation.**

I. INTRODUCTION

A major goal in the field of industrial automation is to perform pick and place tasks using robotics technology when possible. Today, manipulators are being used in factories for many such tasks, including bin picking, box moving and the kitting of packages. A common theme among all of these tasks is that the robot arm is manipulating objects that are at rest. The manipulation of static objects is well researched and multiple approaches have been presented in the last decade that are fast and proven to be reliable. However, autonomous manipulation of moving objects poses new challenges to all pieces of the system - including motion planning for the arm and grasp planning for the end-effector.

Picking up moving objects is a challenging planning problem because kinematic path planning approaches that are used to manipulate static objects are no longer viable. The naive approach in which a kinematic planner for the arm is coupled with a grasp planner for the end-effector, leaves many questions unanswered. These questions include what should the timing of the trajectory be as well as at what time should the object be grasped in its trajectory. Another problem that arises is for a smooth and careful pickup motion to be performed, a grasping motion that tracks the velocity of the object until it's complete is required. Alternatively, if grasping the object is a jerky and abrupt motion that begins at rest, it may not be feasible to accelerate to the proper velocity in time for the pickup, causing a mishandling of

[1]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA {agmenon@andrew.cmu.edu, maxim@cs.cmu.edu}
[2]GRASP Laboratory, University of Pennsylvania, Philadelphia, PA bcohen@seas.upenn.edu

Fig. 1: A possible factory task outlining the careful pickup of moving objects for inspection.

the object. Such an action may be acceptable in a situation in which the objects being manipulated are rigid or robust to mishandling, however, in this paper we are presenting an approach that is capable of dealing with durable objects as well as fragile ones that require careful manipulation.

In this work, we show how the framework of search-based planning can be used to deal with the challenges described above. More specifically, we present a kinodynamic motion planning algorithm that is capable of generating time-parameterized trajectories for the entire pickup motion of a moving object. To generate these trajectories, motion planning is performed in a statespace that includes time in addition to the joint positions and joint velocities of the manipulator. This results in a very high dimensional statespace when planning for a state of the art robotic arm.

In our approach, we systematically construct a graph and search it with an anytime variant of heuristic search, known as ARA* [1]. The algorithm uses a set of dynamically feasible motion primitives to construct a graph in which, every edge is a valid motion with respect to the torque limits of the manipulator and acceleration constraints given by the object. To deal with the additional complexity posed by the grasping problem, we present an adaptive dynamic motion primitive, a primitive that is generated on-the-fly, that is capable of planning a dynamically feasible grasp for the object. To guide the search, we use a fast to compute heuristic that aids the search by providing valuable information regarding the complexities of the velocity and acceleration limits of

the manipulator.

II. RELATED WORK

A manipulation pipeline for picking up moving objects has been shown to consistently pickup objects successfully off of a conveyor belt even at high speeds [2]. In this approach, a kinematic search-based planner is used to plan to a pregrasp pose. From there, using an open loop inverse-kinematics based grasp-motion, starting from rest, the grasp is executed at the calculated time. A drawback is the open loop grasping motion may be unsafe or not synchronized properly with the object's motion. The approach we describe in this paper plans a smooth motion, tracking the object's motion as the gripper encloses it, giving it the capability to manipulate objects too fragile to be picked up with an abrupt scooping motion, such as a glass of water.

Another manipulation pipeline uses a combination of optical flow and a robust control strategy to produce arm motion that synchronizes to the motion of the object [3]. Our approach differs in that we seek to produce a feasible plan, accounting for kinematic feasibility in the environment and differing object geometries for manipulation, while producing a smoother pickup.

For picking up a moving object, a purely kinematic planner is insufficient because the solution is a time parametrized trajectory. The problem is kinodynamic since there is a need to account for manipulator limits on velocity and torque. Time parametrized planning in the kinodynamic state-space is challenging due to the large number of dimensions, as well as an unbounded time dimension. One set of approaches that is robust to the curse of dimensionality is sampling based planners, such as RRT [4] or its variant RRT∗ [5]. Bidirectional RRT's are not applicable here since the goal state is partially defined. Even with forward search however, the goal state is degenerate (can be at any time, and at any arm configuration) which makes biasing the sampling towards the goal not as clear-cut. Additionally, for RRT∗ the rewiring step requires solving a two-point boundary value problem, which is challenging to solve in the case of redundant manipulators. Other approaches to kinodynamic planning include planning in a modified configuration space that accounts for the motion of obstacles [6], a hybrid approach that leverages graph-search and sampling-based planners [7], or algebraically constructing the solution [8]. These algorithms however are limited to lower dimensional problems.

Kinodynamic planning has been framed as a time-optimal control problem, for a given kinematic path [9], [10]. Extensions to this address different path parametrizations [11], and dealing with velocity and torque limits [12] or dynamic singularities [13] . These papers do not address the completeness since combining a kinematic path planner with retiming for kinodynamic constraints are independent steps. A possible solution is to combine the retiming step and planning in configuration space, while including dynamic constraints [14]. Alternatively, instead of time-optimal trajectories, *safety*-optimal trajectories which minimize the likeli-hood of failure, under uncertainty, were found by searching for a kinodynamic policy [15]. Neither of these approaches have considered search for time-parametrized trajectories where the goal is moving.

III. KINODYNAMIC MOTION PLANNING ALGORITHM

The kinodynamic motion planning problem presented in this paper is to pick up a moving object such as a bottle on a conveyor belt. Using prior knowledge of the object's trajectory, as well as a manually selected grasp on the object, we generate a trajectory for the manipulator that is kino-dynamically feasible by constructing and searching a graph using motion primitives that are predefined or generated on-the-fly. The trajectory generated minimizes time taken to retrieve the object (we consider the object retrieved if it is grasped and stationary).

We divide the motion plan into three phases which are differentiated based on the pose of end-effector with respect to the object:

- *Reaching*: end-effector is not close to the object, and the manipulator is moving to position the end-effector at the object.
- *Grasping*: end-effector is moving from near the object to contact with the object, and tracking it to secure its hold on the object. In this phase the manipulator needs to keep the relative position of the end effector and the object the same, otherwise the object would slip out of grasp.
- *Lifting*: the end-effector has achieved a stable grasp of the object which allows it to move the object as if it were rigidly attached. Here the manipulator would be free to move in space, or it could be constrained to the special dynamics of the object, such as a fluid filled container not being jerked.

Our planner performs a single search that finds a solution for all three phases, and solves for the least-time trajectory. The search is done in the space of joint angles, velocities, and in time in order to find a kinodynamically feasible time-parametrized trajectory that picks up the moving object. As discussed in Section II, existing techniques to kinodynamic planning are inapplicable because of the two-point boundary value problem and the degeneracy of the goal state.

We employ a graph search for finding the time parametrized trajectory as described below. As with any graph search it can be split into four parts: graph con-struction, cost function, heuristic function and the search algorithm itself.

A. Graph Construction

We represent the planning problem by an underlying lattice structure which is a discretization of the kinodynamic configuration-space into a set of states S and connections between the states E. This allows us to employ graph search algorithms for finding a motion plan. Using the notation, $G = (S, E)$ to describe the graph we search over, S

represents the state-space of the search and E represents the possible transitions between states.

For our specific task, the state-space S consists of the robot joint configurations, joint velocities, time. A candidate state $s = (\theta_1, \theta_2, ..., \theta_n, \dot{\theta}_1, \dot{\theta}_2, ..., \dot{\theta}_n, t)$ is then a tuple of discretized joint positions, velocities, and time. A dynamic environment naturally pirecludes the inclusion of time as a search dimension.

The transitions between states, E, consist of two sets of motion primitives which are E_{delta} and E_{grasp}.

E_{delta} is a fixed set of motion primitives of the form $(\ddot{\theta}_1, \ddot{\theta}_2, ..., \ddot{\theta}_n, \Delta T)$, which are used for reaching the moving object and intercepting it. A successor state is generated by numerically integrating under these acceleration profiles for a fixed time-step from the current state. We integrate using the Runge-Kutta-Fehlberg and the second-order differential dynamics model of the manipulator of n degrees of freedom:

$$M(\theta)\ddot{\theta} + \dot{\theta}^T C(\theta)\dot{\theta} + G(\theta) = \tau \qquad (1)$$

where θ is the $n \times 1$ vector of joint positions, $\dot{\theta}$ is the vector of joint velocities, $\ddot{\theta}$ is the vector of joint accelerations, M is the $n \times n$ inertia matrix of the manipulator, C is the $n \times n \times n$ Coriolis tensor, and G is the $n \times 1$ vector of gravity forces, and τ is the $n \times 1$ vector of joint torques.

E_{grasp} is a set of *adaptive* motion primitives [16] that uses Jacobian pseudoinverse control to generate a collision-free, dynamically feasible trajectory of the arm that brings the hand in close contact with the object and maneuvers the hand from cradling the object to securely holding the object. This achieves the sub-task of grasping the object. The motion primitive is designed to produce a grasp on the object that is manually determined to work on the object being manipulated, i.e. we hand-pick the grasp and pregrasp pose for the object. As an side note, we bundle lifting sub-task into E_{grasp} in the planner by appending a waypoint that lifts the object, and it can generated either by motions produced through E_{delta} or Jacobian pseudoinverse control.

In our experiments, we use one of the PR2's end-effectors, a two fingered pincher, to pickup cylindrical objects. In this case, the set of candidate grasps are the poses radially arranged around the main axis of the cylindrical object. The motion primitives of E_{grasp} is agnostic to the grasp desired, and the set of motion primitives that are included in E_{grasp} can be expanded to include ranked grasps from grasp-tables, provided appropriate grasp trajectories can be generated using a jacobian motion from the pregrasp pose.

Algorithm 1

graspPrimitive($s, o, t_{pregrasp}$)

1: $P^o_{pregrasp} \leftarrow$ pregrasp pose for object o at $t_{pregrasp}$
2: $\theta_{pregrasp} \leftarrow$ inverse kinematics to reach $P^o_{pregrasp}$
3: `traj` $\leftarrow JacobianMotion(\theta_{pregrasp}, t_{pregrasp})$
4: **if** `traj` is collision-free and dynamically-feasible **then**
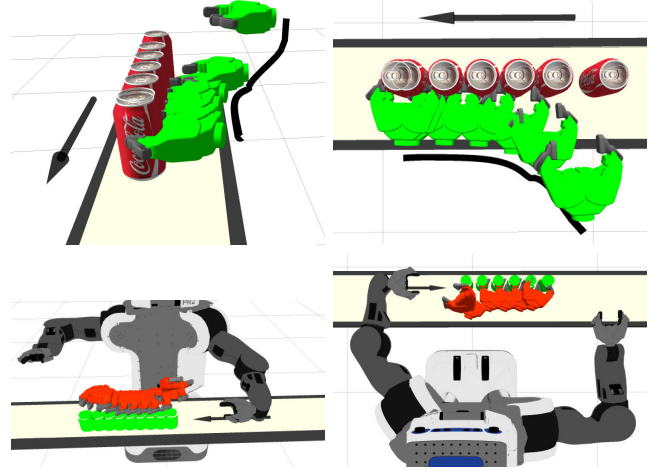5:   **return** `traj`
6: **end if**

Fig. 2: Example gripper trajectories as generated by the adaptive motion primitives in E_{grasp}, showing pregrasp, grasp, and lifting. The black arrow denotes the forward-in-time direction of motion of the object.

As seen in Algorithm 1, a grasp trajectory is generated from the configuration state s, for object o at time $t_{pregrasp}$. A configuration of the arm, $\theta_{pregrasp}$, is solved for using standard inverse kinematics, at a given pose $P^o_{pregrasp}$, which is determined for the object at time $t_{pregrasp}$. For our specific problem, we require the end-effector is facing the radial axis of the object, as visualized in Figure 3a. However there are many poses around the object varying orientation around the object, and we test a handful of such poses. Then using inverse velocity relation from Equation 2, the Jacobian motion trajectory is generated using the control law,

$$\dot{\theta} = J^\dagger \dot{X} + (I - J^\dagger J)\dot{\theta}_0 \qquad (2)$$

where $J^\dagger$ is the Moore-Penrose pseudoinverse of the manipulator jacobian J, $\dot{X}$ is the workspace twist velocity desired, and $\dot{\theta}_0$ is a velocity that makes use of the redundancy in the manipulator to perform auxiliary actions in the trajectory such as maximizing manipulability or avoiding joint limits [17], [18], [19]. In our experimental work, we opted for joint limit avoidance given by

$$\dot{\theta}_{0,i} = \begin{cases} -\dot{\theta}^J, & \theta_i \geq \theta_i^U - \theta^D & (3a) \\ \dot{\theta}^J, & \theta_i \leq \theta_i^L + \theta^D & (3b) \\ 0, & \theta_i > \theta_i^L + \theta^D, \theta_i < \theta_i^U - \theta^D & (3c) \end{cases}$$

where θ_i^L and θ_i^U are the lower and upper joint limits of a joint i, θ^D defines a band near upper and lower joint limits, and $\dot{\theta}^J$ is a constant velocity to get the joint out of the deadband.

In Equation 2, $\dot{X}$ is chosen such that the hand approaches the object from the normal direction while keeping up with the object. The workspace velocity found moves the hand radially towards the object while keeping pace, as shown in Figure 3b, after which it tracks the object's position while
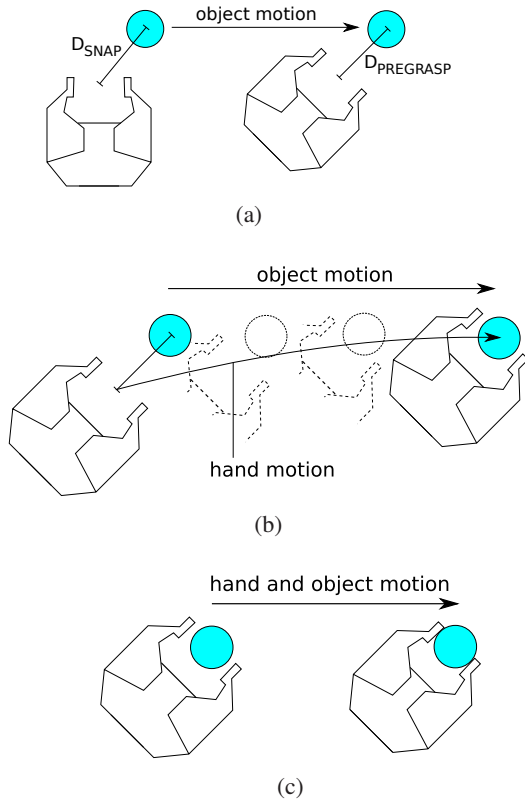
(a)

(b)

(c)

Fig. 3: Trajectory produced by E_{grasp} consisting of (a) pregrasp motion, (b) grasp motion, and (c) tracking motion.

the gripper closes around the object, in Figure 3c. A sample generated gripper trajectory with respect to the object is shown in Figure 2.

After the trajectory is generated, a spline is fitted to the trajectory to get a time parameterization that we can validate easily for infeasible joint torques, velocities, and collisions with the moving goal. Collisions with static obstacles, and joint limits are handled during the rollout of the trajectory. If the trajectory is invalidated for any of the reasons provided, we discard it and move on to testing other pregrasp poses and times for feasible grasp motions.

If a feasible trajectory `traj` is found, this constitutes a valid edge and the successor state is labeled as one of the goal states. The cost of this edge is set to the cost of the trajectory for the motion primitive, i.e. duration of the jacobian motion and pregrasp motion.

If the set of candidate pregrasp states is exhausted, no successors are generated by E_{grasp}.

B. Cost Function

The goal of the cost function is to minimize the execution time to complete the entire pickup task. Therefore the cost of each motion primitive in E_{delta} is the time computed by the forward-integration of the dynamics of the arm. The cost of a motion primitive in E_{grasp} is the time taken to perform all three components - pregrasping, grasping, and tracking. Additionally the time for lifting is accounted for in this.

C. Heuristic

The purpose of a heuristic function is to improve the efficiency of the search by guiding it in promising directions. Heuristic-based search algorithms require that the heuristic function is admissible (underestimating the cost to goal) and consistent [20]. A typical method for formulating a heuristic is to use the results from a simplified problem, such as from a lower-dimensional version of a similar search problem. This is usually achieved by relaxing some of the constraints of the original problem. The heuristic needs to capture the key complexities of the problem at hand to efficiently guide the search in the most informative way. In our problem, we found that it is important for the heuristic to account for the velocity and the acceleration constraints on the arm.

The heuristic estimates the cost (time) to the goal, $t_{heuristic}$, by summing the individual time estimates for the phases of the task, Δt_{reach} and Δt_{grasp} (see Algorithm 2 and Figure 4a).

Algorithm 2

heuristicLeastTime(s, o)

1: $t_{current} \leftarrow$ time of state s
2: $P^e \leftarrow$ pose of end-effector at state s
3: $\dot{X}^e \leftarrow$ velocity of end-effector at state s
4: $t_{sample} \leftarrow t_{current}$
5: **while** $t_{sample} < t_{current} + \Delta T_{max}$ **do**
6:     $P^o \leftarrow$ object o's pose at t_{sample}
7:     $\dot{X}^o \leftarrow$ object o's velocity at t_{sample}
8:     $\Delta t_{reach} = timeOfFlight(P^e, \dot{X}^e, P^o, \dot{X}^o)$
9:     $\Delta t_{grasp} = timeOfGrasp()$
10:    $\Delta t_{estimate} = \Delta t_{reach} + \Delta t_{grasp}$
11:    **if** $\Delta t_{estimate} < t_{sample} - t_{current}$ **then**
12:       **return** $\Delta t_{estimate}$
13:    **end if**
14:    $t_{sample} = t_{sample} + \Delta t_{sample}$
15: **end while**

Algorithm 3

timeOfFlight($P^e, \dot{X}^e, P^o, \dot{X}^o$)

1: $dist =$ euclidean distance between P^e and P^o
2: $axis =$ line from P^e to P^o
3: $v_{start} \leftarrow$ project instantaneous velocity of $\dot{X}^e$ onto $axis$
4: $v_{end} \leftarrow$ project instantaneous velocity of $\dot{X}^o$ onto $axis$
5: $v(t) \leftarrow$ triangular velocity profile subject to $\int_0^T v(t)\,dt = dist, v(0) = v_{start}, v(T) = v_{end}$
6: **if** $\max_t v(t) > v_{max}^e$ **then**
7:    $v(t) \leftarrow$ trapezoidal velocity profile subject to same constraints, and $\max_t v(t) = v_{max}^e$
8: **end if**
9: **return** T from $v(t)$

Assume we are computing the heuristic value of a state s. In order to estimate Δt_{reach}, we use the sub-routine described by Algorithm 3 and visualized in Figure 4b. We
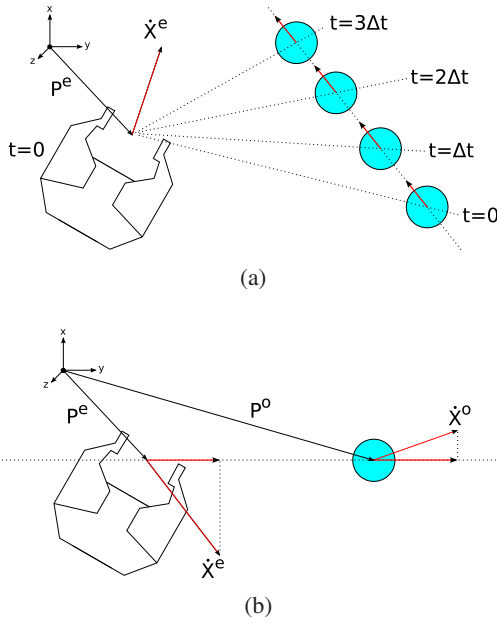
(a)

(b)

Fig. 4: Heuristic function outline showing (a) sampling of multiple points along the object trajectory and then, (b) one dimensional projection to the straight line connecting object and hand.

start by sampling a position P^o and velocity $\dot{X}^o$ (Line 6 and 7) of the object at t_{sample}.

To estimate Δt_{reach} we use Algorithm 3, and this sample pose and velocity. We connect the end-effector at $t_{current}$ and the object at t_{sample} by a straight line path. By projecting the object and end-effector cartesian velocities, $\dot{X}^o$ and $\dot{X}^e$, to this one dimensional path, we solve for a velocity profile $v(t)$, trapezoidal or triangular, using bounded linear velocity v_{max}^e and linear acceleration a_{max}^e. This velocity profile directly gives us Δt_{reach}.

An important step in Algorithm 2, is the check whether Δt_{reach} is greater than $t_{sample} - t_{current}$ for which it was generated. If it is greater, then it takes longer for the free-flying end-effector to reach the pose at t_{sample} than it does for the object. If this is the case t_{sample} is incremented. Once Δt_{reach} is less than $t_{sample} - t_{current}$ then the object can be reached and the heuristic is returned. ΔT_{max} can be chosen as the time till the object leaves the robot workspace. At which point the heuristic can only return infinite costs to the goal.

To compute Δt_{grasp}, we account for the time taken to grasp and lift the object. Since we use a control policy to generate the motion primitive, we set a nominal time which approximates least time it would take to grab the object, i.e. the time to close the gripper. The time to lift the object is also a constant time, which is added to the gripper closing time to give Δt_{grasp}.

The summation of Δt_{reach} and Δt_{grasp} gives us our heuristic estimate for the current state in the search, $t_{heuristic}$.

In order to select the bounded end-effector velocity v_{max}^e, we choose a v_{max}^e that is the the fastest we can move the end effector in the workspace, while respecting the joint velocity limits. To do this, we maximize $\dot{X}^T \dot{X}$, subject to the constraints

$$\dot{X} = J(\theta)\dot{\theta} \qquad (4)$$

$$\dot{\theta}_{min,i} < \dot{\theta}_i < \dot{\theta}_{max,i} \qquad (5)$$

where v_{max}^e is given by $\dot{X}^T \dot{X}$. Equation (5) describes a bounding box in the space of joint velocities. Since the Jacobian of the arm J is affine, we can check the same corners projected into workspace velocity space through Equation (4) and select the highest norm for our estimate of v_{max}^e. Using Equation (5) by permuting each $\dot{\theta}_i$ whether it is a maximum or a minimum for the index i, we compute the corresponding workspace velocity and find the max speed of the end-effector in the workspace as v_{max}^e.

Since this maximum v_{max}^e is dependent on the configuration θ as seen in (4), as a preprocessing step, we test a large number of arm configurations to get an estimate for the top speed of the end-effector in the workspace. The approach to finding the bound on the linear acceleration, a_{max}^e, of the end effector is similarly done to [21]. However for our purposes the acceleration bound is calculated on a stationary arm.

D. Search

We use the anytime variant of A∗, ARA∗, to search a graph G for the solution. ARA∗ tries to find the best solution it can within the given time period. Beginning with an initial suboptimal solution, the algorithm improves the solution while deliberation time remains. ARA∗ is complete for the given graph G and has theoretical bounds on on sub-optimality of solutions. Using an inflated heuristic, by $\epsilon \geq 1$, the graph-search reduces with time the ϵ to 1, at which point the return solution is the least cost path.

IV. EXPERIMENTAL RESULTS

Our experimental analysis includes two components. First, we present a battery of tests in simulation to measure the computational performance of the planner. Then we present experiments on an actual robot in which it picks up moving objects. The purpose of the two stage analysis is that it is difficult to assess whether the robot successfully grasped the object or not in our simulated environment. In addition, without object detection and perception in the test setup on the actual robot, the process of performing a large battery of tests becomes tedious and time consuming. Instead, we performed a large set of planning requests in simulation to benchmark the performance and robustness of the planner and a small subset of those were repeated on the actual robot to verify the success of the grasping motions and included in the attached video. We performed all of our experiments on Willow Garage's PR2 robot.
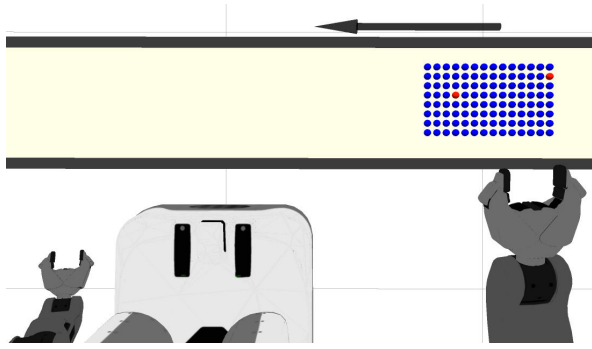
Fig. 5: The dots represent the initial poses of the moving object in the experiments.

	$\ddot{\theta}_0$	$\ddot{\theta}_1$	$\ddot{\theta}_2$	$\ddot{\theta}_3$	$\ddot{\theta}_4$	$\ddot{\theta}_5$	$\ddot{\theta}_6$	Δt
	± 1.0	0	0	0	0	0	0	0.2
	0	± 1.0	0	0	0	0	0	0.2
	0	0	± 1.0	0	0	0	0	0.2
	0	0	0	± 1.0	0	0	0	0.2
	0	0	0	0	± 1.0	0	0	0.2
	0	0	0	0	0	± 1.0	0	0.2
	0	0	0	0	0	0	± 1.0	0.2

TABLE II: List of motion primitives in E_{delta}

Fig. 6: The objects used in our experiments on the PR2.

A. Simulation

Our performance tests were run in simulation on a computer running Ubuntu 11.10 with an Intel Core2 Quad Q9550 CPU (2.83Ghz) with 6GB of RAM. We generated a set of 112 tests, each with initial object poses varying in their position in xy on the conveyor belt. In all of these tests, the belt speed is set to $0.1 m/s$, we are planning for the right arm, and the object that is being grasped is a soda can. The initial poses of the object are each separated by $2cm$ in a $14cm$ by $26cm$ rectangle. In Figure 5, the initial poses are displayed as blue and red dots and the belt moves along the direction of the arrow. The initial joint configuration of the right arm is the same for all experiments and can be partially seen in the figure.

In each of these trials, the planner is given an allocated planning time of 30 seconds and if a solution is not found it is considered a planning failure. Of the 112 tests performed, the planner successfully computed plans for 110 of them, yielding a success rate of 98.2%. In Table I we present a summary of the successful planning requests. In each test, the initial $\epsilon = 100$ and the reported results are computed using the first solution found.

	plan. time(s)	expans.	sol. cost	execution time (s)
average	2.25	10.6	3,296	5.04
st. dev	2.84	10.2	290	0.29

TABLE I: Summary of performance results.

In these experiments, we use a basic set of motion primitives for E_{delta} in which each joint is accelerated one at a time. We chose this set because they provide a dense coverage of the workspace. However, more complicated acceleration profiles can be used in E_{delta}. We state the values of the joint velocities in radians per second squared and Δt in seconds. The actual motion primitives in E_{delta} that were used in these experiments are listed in Table II.

We manually select the grasp points of the object during our experiments performed on the PR2 that would result in the most consistent pickups. The E_{grasp} motion primitives are considered at any state whose end effector was within a threshold of 0.10m. Additionally we estimated the time to close the gripper was 2.0s.

B. PR2 Experiments

In experiments on the PR2 we use its left arm to place an object on the conveyor and right arm for the delicate pickup of the object. This imitated a scenario where the motion of the object is detected perfectly using external sensors such as laser range finders arrayed along the conveyor. The velocity of the object is known to the PR2 (and planner). During the experiment the object is initially held slightly above the belt in the left hand at its set initial pose. The left gripper then opens, placing the object on the moving belt, and triggers the execution of the planned path.

For our experiments, we chose objects that would help demonstrate the smoothness of the planned motion. The objects we used include a soda can, martini glass and a toy bird balancing on a pointed stand mounted to the top of another soda can. Screenshots of the objects in the video can be seen in Figure 6. Note that the martini glass is double-walled with clear liquid inside that is free to move around. The placidity of the liquid during the pickup gives a sense of how smooth the planned trajectory is. The same is true with the balancing bird. In the accompanying video, the toy bird's weighted beak makes it adept at balancing on its tip however it fails to maintain its balance given jerky or abrupt motions.

V. DISCUSSION

Currently in our implementation E_{grasp} is only one kind of motion, that uses Jacobian pseudoinverse control. This is not exhaustive in the space of possible pregrasp motions, for manipulators with more complicated hands, and for redundant manipulators. E_{grasp} uses a controller whose gains are not automatically determined by the planner, and hence requires some manual configuration to produce a set of feasible grasping motions. The grasp poses are limited to cylindrical objects due to their simplicity.
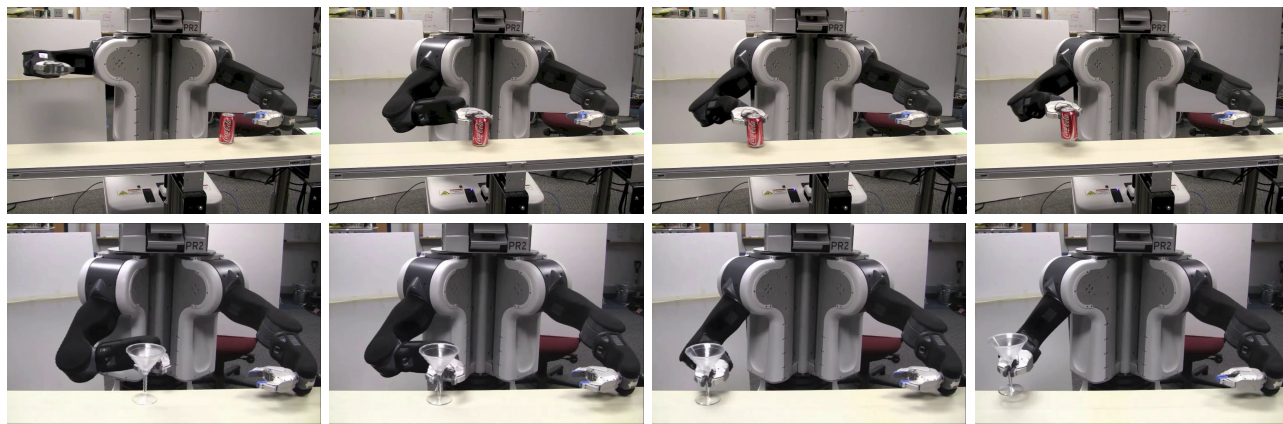
Fig. 7: Snapshots of the robot picking up a soda can and a martini glass off of a conveyor belt.

Additionally E_{delta} is lacking *sleep* motion primitives, i.e. motion primitives that bring the arm to rest. These primitives would enable the planner to deal with objects that do not enter the manipulator workspace for a long time.

The planner may not compute a trajectory in real time if the object is too close to the robot, from our experimental results. A feasible use-case is precomputing a library of such plans, which can be indexed based on the object position and trajectory for a variety of scenarios. Planner optimizations are the next step for us to improve its applicability to real time scenarios.

Possible extensions to the planner to improve its applicability also include closed loop planning. Formulating it as an incremental search problem (dynamic replanning) may enable the use of object trajectory feedback.

VI. CONCLUSIONS

In this paper we have presented a search-based kinodynamic motion planning algorithm that is capable of generating time-parameterized trajectories to pickup moving objects. Our approach generates a smooth trajectory, capable of matching the velocity of the object throughout the grasping motion while being feasible with respect to joint torques and velocity limits. To efficiently deal with the high dimensionality of the planning problem, we introduced adaptive dynamic motion primitives, which are motions generated on the fly that use jacobian pseudoinverse control to simplify the planning of the grasping motion. The algorithm relies on an anytime graph search to generate solutions quickly, as well as to provide consistency and theoretical guarantees on the completeness, consistency and provides a bounds on the suboptimality of the solution cost. The search is guided efficiently by an informative heuristic that aids in coping with the velocity and acceleration limits of the arm. Our video and experimental results demonstrate the success of the approach, however, for it to be used in real time, further optimization is needed.

REFERENCES

[1] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems (NIPS) 16*.   Cambridge, MA: MIT Press, 2003.

[2] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and motion planning for pick-and-place of dynamic objects," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[3] P. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, "Automated tracking and grasping of a moving object with a robotic hand-eye system," *Robotics and Automation, IEEE Transactions on*, vol. 9, no. 2, pp. 152–165, 1993.

[4] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001. [Online]. Available: http://ijr.sagepub.com/content/20/5/378.abstract

[5] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Decision and Control (CDC), 2010 49th IEEE Conference on*.   IEEE, 2010, pp. 7681–7687.

[6] E. Owen and L. Montano, "Motion planning in dynamic environments using the velocity space," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*.   IEEE, 2005, pp. 2833–2838.

[7] E. Plaku, L. Kavraki, and M. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[8] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.

[9] J. E. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.

[10] K. Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *Automatic Control, IEEE Transactions on*, vol. 30, no. 6, pp. 531–541, 1985.

[11] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," in *Robotics: Science and Systems*, July 2012, pp. 09–13.

[12] L. Zlajpah, "On time optimal path control of manipulators with bounded joint velocities and torques," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2, 1996, pp. 1572–1577 vol.2.

[13] Z. Shiller and H. Lu., "Computation of path constrained time optimal motions with dynamic singularities," *Journal of dynamic systems, measurement, and control*, vol. 114, no. 34, 1992.

[14] Q.-C. Pham, S. Caron, and Y. Nakamura, "Kinodynamic planning in the configuration space via velocity interval propagation," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[15] K. Byl, "Optimal kinodynamic planning for compliant mobile manipulators," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2010.

[16] B. J. Cohen, G. Subramanian, S. Chitta, and M. Likhachev, "Planning for Manipulation with Adaptive Motion Primitives," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[17] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[18] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *Control Systems Technology, IEEE Transactions on*, vol. 2, no. 2, pp. 123–134, 1994.

[19] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Journal of Dynamic Systems, Measurement and Control*, vol. 108, pp. 163–171, September 1986.

[20] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[21] A. Bowling and O. Khatib, "The motion isotropy hypersurface: a characterization of acceleration capability," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 2, 1998, pp. 965–971 vol.2.