

Dynamic Grasping with Reachability and Motion Awareness

Iretiayo Akinola*, Jingxi Xu*, Shuran Song and Peter K. Allen

Abstract—Grasping in dynamic environments presents a unique set of challenges. A stable and reachable grasp can become unreachable and unstable as the target object moves, motion planning needs to be adaptive and in real time, the delay in computation makes prediction necessary. In this paper, we present a dynamic grasping framework that is reachability-aware and motion-aware. Specifically, we model the reachability space of the robot using a signed distance field which enables us to quickly screen unreachable grasps. Also, we train a neural network to predict the grasp quality conditioned on the current motion of the target. Using these as ranking functions, we quickly filter a large grasp database to a few grasps in real time. In addition, we present a seeding approach for arm motion generation that utilizes solution from previous time step. This quickly generates a new arm trajectory that is close to the previous plan and prevents fluctuation. We implement a recurrent neural network (RNN) for modelling and predicting the object motion. Our extensive experiments demonstrate the importance of each of these components and we validate our pipeline on a real robot.

I. INTRODUCTION

Roboticians have made significant progress in developing algorithms and methods for robotic manipulation in static environments. However, robotic manipulation becomes much harder in dynamic environments which is often the case in the real world. For example, in dynamic grasping, ball catching, human-robot handover, etc., the targets and obstacles to be interacted with might be moving with an unknown motion. Providing robots with the ability to manipulate objects in dynamic environments, despite being less explored, can be extremely important in realizing automation in both industry and daily life. Figure 1 illustrates a conveyor belt setting; an ability to pick up the target object without pausing the conveyor belt or knowing the speed of the target object a priori can improve the overall efficiency of the system.

There are many challenges brought by dynamic environments. First, continuous changes in the environments require online and fast motion replanning. Sampling-based methods (RRT, PRM, etc.) are not well-suited for this requirement because the randomness of solutions leads to jerky and wavy motion due to the replanning at each time step. Optimization-based methods (CHOMP, STOMP, etc.) can be time-consuming in highly cluttered scenes, making fast replanning in dynamic environments extremely difficult. Second, most works in the grasp planning literature rarely

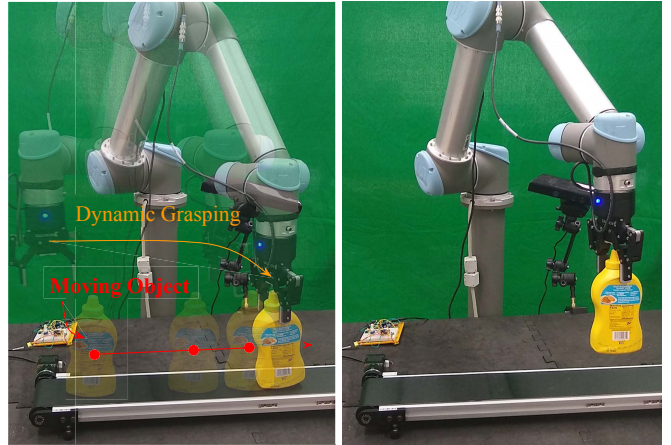


Fig. 1: Dynamic Grasping Problem: A moving target object is to be grasped and lifted. The object pose and motion is not known a priori and has to be estimated online. Full degree-of-freedom grasps should be explored to come up with feasible grasps that can pick-up the object before it escapes the robot’s workspace.

consider the approach and close motion of the grasp, which makes a difference for a moving target. For example, a grasp facing the moving direction of a target can have a higher success rate than a grasp catching the target from the back. Third, we need to understand and predict the motion of the object because computed plans are obsolete when executed.

Previous works have addressed dynamic grasping by introducing a number of assumptions such as prior knowledge of the object motion [1], waiting for the object to come to rest before grasping, limiting the grasping directions to a single direction (e.g. only top-down grasps [2]). In this work, we relax some of these assumptions and tackle the problem of robotic grasping for moving objects with no prior knowledge of the object’s motion profile and no restrictions on the possible grasping directions of the object. The increase in the range of possible grasping directions has the advantage of expanding the workspace of the robot leading to more grasp options that can be very useful in the dynamic setting. However, as the range of feasible grasp options grows, so does the range of infeasible ones. Without a notion of reachability, it is usually preemptively time-consuming to compute IKs for all the grasps in the database. Our method, illustrated in Figure 2, embraces the advantage of an expanded workspace for full degree-of-freedom (DOF) grasps and mitigates the reachability problem by constraining the grasp selection process to the more reachable and manipulable regions of the workspace. In addition, we observe that the robustness of a grasp may vary depending on the speed and direction

*Equal Contribution

This work was supported in part by National Science Foundation grants IIS-1734557 and IIS-1527747. Authors are with the Department of Computer Science, Columbia University, New York, NY 10027, USA. (iakinola, jxu, shurans, allen)@cs.columbia.edu,

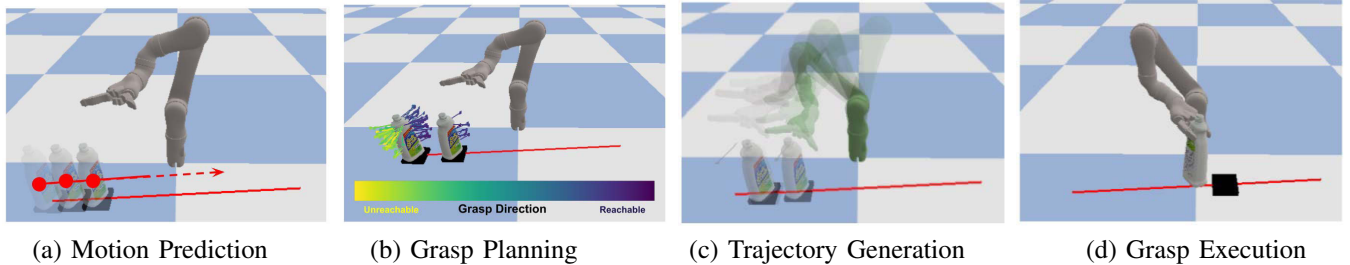


Fig. 2: Dynamic Grasping Framework. **a)** Instantaneous pose estimation runs continuously to keep track of the moving object and we use a recurrent neural network to model the motion of the target object and predict its future pose. **b)** Full grasp database are ranked and filtered based on reachability. **c)** Pick the grasp from filtered list that is closest to the current arm configuration. Arm trajectory is generated based on the future pose of the moving object. Arm trajectory from previous time step is used to seed the planner in current step. **d)** Approach and grasp are executed when CANGRASPs condition is satisfied.

of the moving object. To handle this, we learn a function that predicts the robustness of grasps given the motion of the object. This is used to rank and select a robust reachable grasp. To generate arm motion, rather than planning from scratch each time, we seed the planning process by the solution from the previous time step. This method allows the newly planned trajectory to be similar and also speed up the computation. In summary, our main contributions are:

*a) **Reachability and motion-aware grasp planning:*** ranking functions that predict the reachability and success probability for different grasps based on target pose and motion of the target. These ranking functions are used for real-time grasp filtering.

*b) **Adaptive motion generation:*** an effective trajectory generation approach that incorporates the solution from previous timestep to seed the search process to achieve quicker and smoother transition between different motion plans.

*c) **Simulation and real robot evaluation:*** procedure of systematically evaluating dynamic grasping performance in simulation with randomized linear/nonlinear motion with different objects, and a real robot demonstration to pick up objects moving on a conveyor belt.

II. RELATED WORKS

1) Grasping in Dynamic Environments: Grasping of static objects can be achieved via visual servoing [3], [4], [5], [6], [7], [8], [9], [10]; however, grasping in a changing environment presents a unique challenge. The robot not only tracks the object but also has to reason about the geometry of the object to determine how to pick it up. Some learning-based grasping systems [11] have been applied to slightly moving scenes. Previous works demonstrate grasping of a static object in the midst of moving obstacles [12]. Our work deals with picking a moving object while avoiding collision with static obstacles.

2) Database-based Robotic Grasping: Previous works [13], [14] have looked at the idea of grasping using a pre-computed database. Most of these methods sample different grasps and evaluate them in simulation using a geometry-based metric. These metrics use static analysis which does not account for the dynamics of the approach and lift process. Some other methods [15], [16] generate grasp database using a real robot which can be valuable but such data is very

expensive to collect. A recent concurrent work [17] used this technique to examine different approaches for sampling grasps when generating a grasp database and evaluated their coverage of possible grasping directions. They very densely sample “billions” of grasping direction and measure the robustness of each grasp candidate using the success rate of it’s neighbours. [18] collects grasps with randomly added perturbations on the object poses.

3) Object Tracking: Visual feedback is crucial to grasping and manipulation in dynamic environments. For a position-based system like ours, the visual input from a camera (color and/or depth) is continuously processed into the pose (position and orientation) of the objects in the environment. Bayesian methods [19], [20] or deep learning techniques [21], [22] can be applied to the input image stream to produce object poses in the camera’s frame of reference. The noisy pose results from object pose detection systems can be filtered into more stable values using methods such as Kalman filtering [23]. Since Kalman filtering builds a model for the motion, this model serves as a good predictor for the future pose of the object being tracked.

4) Motion Generation: When obstacles are present, reaching a moving target requires some trajectory planning (such as RRT [24], PRM [25]) or trajectory optimization methods (such as CHOMP [26], STOMP [27]) that are able to generate collision-free paths for the arm. Recent works [28], [29] presented an approach to generate a sequence of constraint-based controllers to reactively execute a plan while respecting specified constraints like collision avoidance. Our work is more similar to works that generate arm motion from a library of stored arm motions [30], [31], [32]. Building on these works, we propose an approach that only keeps the solution from previous time step without a precomputed database of arm motions. This previous solution is used to initialize tree/roadmap for sampling-based methods or as a seed for trajectory optimization solver.

A recent work [2] presented an approach that uses motion prediction to grasp a moving block using top-down grasps; they illustrated their approach using simulated experiments. Our work differs from [2] in that we do not limit the grasp direction to only top-down direction, we handle different objects and we incorporate a notion of reachability [33] to

guide the grasping process. In addition, we demonstrate our method on real hardware. Another recent work [34] looked holistically at the problem of dynamic grasping especially during handover between a human and a real robot. As the object moves, approximate inverse kinematics (IK) are computed on a database of pre-computed grasps and the quality of the IK solutions are computed and ranked. In our work, we compare the IK of filtered grasps to the current robot joint values and pick the closest grasp.

III. PROBLEM DEFINITION

The task is for a robot to pick up a moving object whose motion is not known a priori and avoid colliding with the surrounding obstacles. We assume that the models of the objects and obstacles exist so the system can model the environment using object detection and pose estimation. The task is successful if the robot is able to pick up and lift the correct target object without knocking over the surrounding objects/obstacles. We also want target object to be picked up as fast as possible. This task imitates many warehouse conveyor belt scenarios when both the obstacle and target objects are fragile and moveable with unplanned contact.

IV. METHOD

In this section we describe the various components of our system (illustrated in Figure 2). First, we describe the visual processing unit that detects object poses. We then discuss the predictive component that estimates the future pose of the object. Next, we discuss the online grasp planning component that produces motion-conditioned reachable and stable grasps in real time. Finally, we present our arm motion generation method.

A. Overview

The overall algorithm is presented in Algorithm 1. Each grasp consists of a grasp pose and a pregrasp pose generated by backing off the grasp pose for distance b . Our pipeline takes in a known object O . It first retrieves a pre-computed database of grasps G_{DB} for the target object; grasps in G_{DB} are all in object frame. In the dynamic grasping loop, it estimates the current pose p_c of the target and predicts a future pose p_f with duration t . t is defined as a step function of the euclidean distance d from the arm end-effector to the planned pregrasp: $t = 2s$ if $d > 0.3m$, $t = 1s$ if $0.1m < d \leq 0.3m$, and $t = 0s$ if $d \leq 0.1m$. We convert the grasps in G_{DB} from object frame to robot frame according to predicted pose p_f and then filtered the grasps using the reachability and motion-aware ranking functions described later in Section IV-C and keep the shortlisted top 10 grasps G_F . We pick the grasp g_c from G_F that is closest to the current robot configuration and move the arm if p_c is reachable otherwise we continue to the next loop. We keep executing the algorithm until the condition for executing the grasp is satisfied. Define the euclidean distance between the end-effector position and the planned grasp position to be d_p , and the absolute quaternion distance between the end-effector orientation and planned grasp pose orientation to be

d_q , then CANGRASP returns true if $d_p \leq 1.1b$ and $d_q \leq 20^\circ$, where b is the back-off distance.

After the condition is met, we get an updated estimate of the object pose, predict with horizon $t' = 1s$, and convert g_c using the newly predicted pose p'_f . The arm is then moved to g_c . The hand is closed while moving with the target for another $t'' = 0.1s$. In our pipeline, t , t' , and t'' are configured experimentally but the optimal prediction horizon should be a function of the end-effector speed, the distance between the end-effector and the planned grasp pose, and the motion of the target. We leave this for future research. Finally, we check if the object has been lifted to determine success.

Algorithm 1 Dynamic Grasping Pipeline

```

1: function DYNAMICGRASP( $O$ )
2:    $G_{DB} \leftarrow \text{RETRIEVEGRASPDATABASE}(O)$ 
3:   while True do
4:      $p_c \leftarrow \text{DETECTPOSE}(O)$ 
5:      $p_f \leftarrow \text{PREDICT}(p_c, t)$ 
6:      $G_W \leftarrow \text{CONVERTGRASPS}(G_{DB}, p_f)$ 
7:      $G_F \leftarrow \text{FILTERGRASPS}(G_W, p_f)$ 
8:      $g_c \leftarrow \text{PICKGRASP}(G_F)$ 
9:     Continue to next iteration if  $g_c$  is not reachable
10:    Move arm to  $g_c$ 
11:    if CANGRASP() then
12:       $p'_c \leftarrow \text{DETECTPOSE}(O)$ 
13:       $p'_f \leftarrow \text{PREDICT}(p'_c, t')$ 
14:       $g_c \leftarrow \text{CONVERTGRASPS}(g_c, p'_f)$ 
15:      Move arm to  $g_c$ 
16:      Close hand while moving with the target for  $t''$ 
17:      Break loop
18:    end if
19:  end while
20:  return CHECKSUCCESS()
21: end function

```

B. Object Motion Modelling

Picking up moving objects requires instantaneously detecting the relevant objects in the scene. We continuously track/model the motion of the object to be able to handle cases where the motion profile changes with time.

1) *Object Detection and Tracking*: In real-world experiments of this work, we use a recent learning-based method (DOPE [21]) to get instantaneous poses of moving objects in the scene. DOPE trains a neural network model that takes an RGB image as input and outputs the pose of a target object relative to the camera frame. A different model is trained for each object of interest and each model can detect multiple instances of their target object. Images of the grasping scene are captured using a kinect and passed through the DOPE models to detect objects and obstacles in the scene. To achieve robustness, we use the published model [21] that was trained on data collected in different lighting condition. In simulation, we directly access the pose of the objects and obstacles in the grasping scene.

2) *Recursive State Estimation/Object Pose Prediction*: Grasp/motion planning has a time cost and a computed grasp/motion plan can become obsolete very quickly as the object moves. As a result, an ability to predict the future pose of the target object can improve the overall success of a dynamic grasping system. The motion prediction ability is

needed for both planning a grasp and executing a grasp (the approach and close motion). While Kalman filtering (KF) [23] is a practical approach for linear motion prediction, we adopt a recurrent neural network (RNN) approach to be able to generalize to non-linear motions as well. The RNN continuously takes in a sequence of instantaneous pose measurements $(p_{t-n}, \dots, p_{t-1}, p_t)$ to update its internal state which is used to predict future pose at different prediction horizon lengths $(p_{t_{f1}}, p_{t_{f2}}, \dots, p_{t_{fm}})$. To train the RNN¹, we create a dataset contains planar linear, circular and sinusoidal sequence of waypoints (2000 each) randomly generated along different directions with different start points. To aid learning and generalization, each sequence data point is normalized to the start of the sequence i.e. $((0, \dots, p_{t-1} - p_{t-n}, p_t - p_{t-n}) \rightarrow (p_{t_{f1}} - p_{t-n}, p_{t_{f2}} - p_{t-n}))$. This RNN approach can also be used to model object motion during human-robot handovers.

C. Grasp Planning for Moving Objects

1) *Grasp Database Generation*: To generate grasps for moving objects, we pre-compute a database of grasps for all target objects while they remain static. Similar to [17], this database was collected and evaluated purely in simulation with dynamics turned on. First, we densely generate 5000 stable grasps for each object using a simulated annealing approach [35], and we then evaluate all the grasps in simulation; each grasp is executed to lift the object 50 times and each time we add random noise to the object pose [18]. The success rate gives a measure of the robustness of the grasp and we choose the top 100 robust grasps.

2) *Reachability-Aware Grasping*: In the dynamic grasping setting, it is important to have a fast way to choose a feasible grasp out of the list of stable and robust grasps. Generating collision-free IK for all the grasps can be time-consuming; instead, we use our pre-computed reachability space to quickly rank the grasps for the given object pose estimate (See [33] for more details). The larger reachability value the grasp has, with higher probability a valid IK can be found for that grasp. Reachability can also be an index of manipulability and it follows the intuition that the most reachable grasp has higher probability to continue being reachable in the future, reducing the number of grasp switches while the target moves around. The interpolation and indexing of a pre-computed 6D space gives a fast way to reduce the grasp database to a few more reachable grasps whose IK can then be found. This approach is much faster than computing IK for the entire database and is important in dynamic settings. We can use this reachability computation as a rank function for FILTERGRASP in Algorithm 1. An example using reachability is shown in Figure 2.

3) *Motion-Aware Grasping*: We observe that the success rate of a stable grasp varies depending on the motion of the object. For example, picking up an object from behind as it moves away can result in different success rate statistics compared to approaching in the direction opposite its motion. To

address this, we learn a neural network model $\mathcal{M}(g, pg, v, \theta)$ that predicts the success probability of a grasp g given the motion profile of the object (speed v and motion direction θ). The input into the model includes:

- The 6D grasp pose ($g \in \mathbb{R}^6$) i.e. the $\{x, y, z\}$ position and $\{roll, pitch, yaw\}$ orientation in the object's frame of reference.
- The 6D pre-grasp pose ($pg \in \mathbb{R}^6$) which is the grasp pose backed off (5 cm for Mico hand and 7.5cm for robotiq hand) along the approach direction (i.e. a vector pointing from the end-effector towards the object).
- The speed $v \in \mathbb{R}$ of the object.
- The motion direction $\theta \in [0, 2\pi]$. We assume a 2D planar motion parameterized by a polar angle direction around the z-axis of the object frame.

The model has two hidden layers (512 each) and an output predicting the success probability. We generated a dataset of 10000 grasp attempts each on 7 different objects in simulation using the robot's end-effector only. For each grasp attempt, the end-effector starts at the pregrasp pose and moves towards the object while the object moves in a randomly sampled planar direction, at a speed sampled uniformly between 0.5cm/s and 5cm/s. We record the result of the grasp attempts and use this as supervision to train the models (one for each object). We train 100 epochs for each object. The average training time is ~ 5 mins and the average validation accuracy is 0.963 with False Positive Rate (FPR) 0.017 and False Negative Rate (FNR) 0.117. Ultimately, the probability of success output by the network can be used as a motion-aware quality conditioned on the object motion. We can use this network to quickly filter grasps that has the highest motion-aware quality for FILTERGRASP in Algorithm 1. In general, the motion-aware model prefers grasps facing the moving direction of the target.

4) *Combining Reachability and Motion-aware*: We want to include grasps in the shortlisted pool G_F that are both reachable and are stable conditioned on the object motion. There are many different ways to combine the reachability and motion-aware quality for each grasp in the database. We empirically find that simply including the top 5 grasps with highest reachability and the top 5 grasps with highest motion-aware quality outperforms other ways of combination, including the weighted sum of two values or filtering by reachability and then motion-aware quality, etc.

D. Motion Generation and Grasp Execution

There are three stages of motion generation when picking up an object: reaching, grasping and lifting [36]. In our implementation, we transform the planned grasp to match the predicted future pose of the object and generate arm motion for all three phases.

To be able to generate and update the reaching trajectories as the object moves, we introduce the idea of trajectory seeding that uses the trajectory solution of a previous time step as an initialization for finding a new trajectory at the current time-step. For sampling-based methods like RRT or PRM, this entails initializing the sampling tree or roadmap with the

¹RNN model: LSTM(100), $2 \times$ Dense(100), Dense(output_shape).
output_shape = num_future \times dim

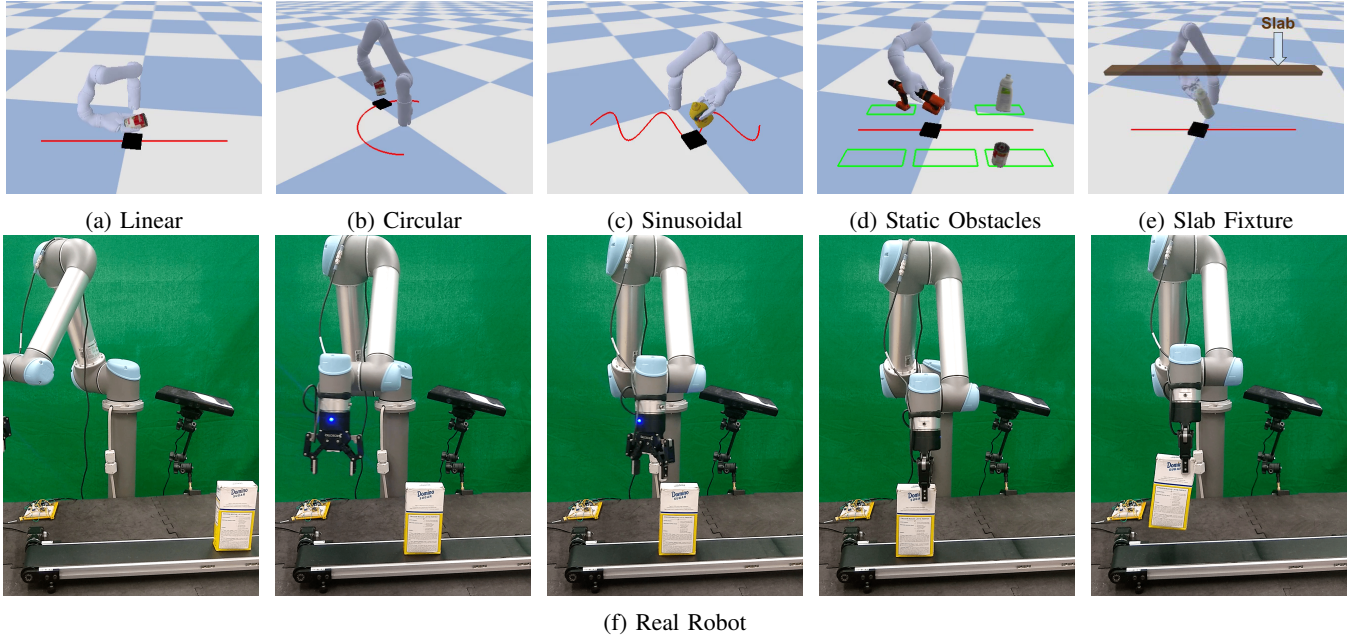


Fig. 3: Dynamic Grasping Tasks. Experimental scenarios for picking up objects on a conveyor belt. The red line shows the conveyor belt trajectory. (a), (b), (c) Linear, circular and sinusoidal motion of target object with no surrounding obstacles. (d) Linear motion with surrounding static obstacles. Green rectangles are the sub-regions where we sample obstacle locations. (e) Linear motion with slab fixture that limits feasible grasping directions. (f) Real Robot Demo: Linear motion of target object moving at 4.46 cm/s.

waypoints found from the previous time step. This seeds the search to be quite close to the previous path and empirically helps find new solutions that are not drastically different from the previous solution. An unconstrained sampling based approach can return drastically different trajectories in subsequent trajectories which can be disadvantageous in dynamic settings. Another benefit of our seeding approach is that a good initialization from seeding can reduce the time used to find a valid solution. We implement our seeding approach on CHOMP, RRT and PRM and find that PRM works best for our experimental tasks.

Note that the motion plan is executed once generated interrupting the previous trajectory that was being executed. To ensure that the arm does not slow down as new trajectories are computed and updated, we retime the trajectory solution from the solver so that it blends with the current arm velocities and it moves as fast as possible while also respecting joint limits [37]. We use cartesian control to move the arm during the grasping and lifting stages.

V. EXPERIMENTS

We extensively evaluate the performance of our algorithm picking different target objects in randomized linear/nonlinear motion with/without static obstacles in simulation. We then demonstrate that our method works reliably on a real robot. Videos showing some of the experiments can be found at our project website http://crlab.cs.columbia.edu/dynamic_grasping.

A. Experimental Setup

We create different scenarios illustrated below using the Bullet simulator [38] to evaluate the performance of our methods on two robot arms with parallel jaw grippers:

the Kinova Mico and the UR5-Robotiq robots. These two robots have different workspace dimensions, manifolds, joint limits as well as different gripper spans/width. UR5 arm has a wider span and moves quicker than Mico arm, so we intentionally make the tasks for UR5-Robotiq harder. For each robot, we simulate the task of linear and non-linear conveyor belt pickup, which plays a significant role in warehouse packaging and assembly lines. In these scenarios, there is a target object moving on a belt, possibly among surrounding static obstacles. Both the target and the obstacle objects can be fragile and we cannot knock them over.

a) Linear Motion: The target object moves linearly at a constant speed (3cm/s for Mico and 5cm/s for UR5-Robotiq) as shown in Figure 3a. The conveyor trajectories are randomized using 4 parameters as shown in Figure 4. θ specifies the counter clockwise angle of the line perpendicular to the linear motion, connecting the middle point of the motion and the base of the arm. r is the distance between the linear trajectory and the arm base. l is the length of the linear trajectory. $d \in \{+1, -1\}$ indicates the direction of the motion, where +1 means moving counter clockwise and -1 means moving clockwise. We set $0 \leq \theta < 2\pi$ (radians), $0.15m \leq r \leq 0.4m$ for Mico and $0.3m \leq r \leq 0.7m$ for UR5-Robotiq, and $l = 1m$.

b) Linear with Obstacles: We add 3 static obstacles to the linear motion in the grasping scene. Specifically, we divide the near region (distance between 0.15m and 0.25m) surrounding the linear motion into 5 sub-regions, as shown in Figure 3d. For each obstacle, we randomly pick a sub-region and uniformly sample a location in the sub-region. We make sure there is no collision between obstacle and robot or between two obstacles. The arm has to avoid hitting both

obstacles and the target.

c) *Linear with Top Slab*: A 2cm-thick slab of width 10cm is placed 40cm directly on top of the conveyor belt. This limits the grasping directions (e.g. top-down grasps) and makes motion planning/grasping more challenging.

d) *Linear with Z Motion*: We relieve the constraint of the linear motion so that the object can also move in the Z axis. The starting height and the end height is randomly sampled between 0.01m and 0.4m.

e) *Linear with Varying Speed*: The target is accelerated from 1cm/s to 3cm/s and from 3cm/s to 5cm/s for Mico and UR5-Robotiq respectively.

f) *Circular Motion*: A smooth non-linear circular motion as shown in Figure 3b. The speed of the conveyor belt is constant (2cm/s for Mico and 3cm/s for UR5-Robotiq). The circular motion trajectory is also randomized by 4 parameters as shown in Figure 4. θ controls the angle of the starting position on the circle. r is the radius of the circle. l specifies the length of the motion. $d \in \{+1, -1\}$ indicates the direction of the motion, where +1 means moving counter clockwise and -1 means moving clockwise. We set $0 \leq \theta < 2\pi$ (radians), $0.15\text{m} \leq r \leq 0.4\text{m}$ for Mico and $0.3\text{m} \leq r \leq 0.7\text{m}$ for UR5-Robotiq, and $l = 1\text{m}$.

g) *Sinusoidal Motion*: This is a more challenging non-linear motion where the object moves along a sinusoidal path as shown in Figure 3c. To do this, a sinusoid is superimposed on the randomly generated linear motion as shown in Figure 4. In addition to the parameters of the linear motion (θ, r, l, d), we specify the amplitude A and frequency f of the sinusoid. We set $A = l/8\text{m}$ and $f = 2\pi/(l/3)\text{Hz}$.

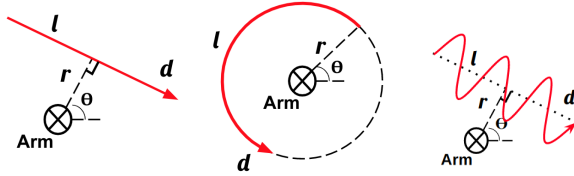


Fig. 4: A bird's-eye view of randomized linear, circular and sinusoidal conveyor belt motion generation process. A random experiment motion is parameterized by angle θ , distance r , direction d , and length l . The cross indicates the position of the robot base. The red line shows the motion of the conveyor belt, with an arrow indicating the direction. The horizontal dashed line indicates the x -axis of the world frame. Left: linear motion. Middle: circular motion. Right: sinusoidal motion.

Each experiment in simulation is run on 7 different target objects shown in Figure 5 whose sizes can physically fit in the robots' hands. The randomized process for generating conveyor belt motion ensures that the results are not biased to a specific robot configuration. For example, a particular starting pose might be close to the robot arm end-effector and will have a higher success rate. For each object, we run each 100 times and report the average success rate and grasping time across 700 trials. In each setting, we compare the performance of the below methods.

- **Ours (R+M)**. This is our proposed method that uses all the discussed modules and filters grasp in the grasp

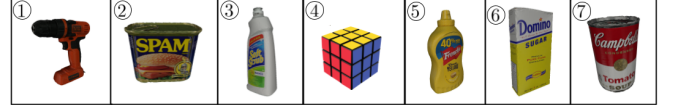


Fig. 5: Seven objects from the YCB Object Database selected as the graspable objects in our experiments. All seven are used for simulation experiments while the last three are used for the real robot experimentation.

database combining both reachability and motion-aware quality as discussed in Section IV-C.4.

- **Ours (Reachability)**. Same as *Ours (R+M)* except the grasps are filtered with only reachability.
- **Ours (Motion-aware)**. Same as *Ours (R+M)* except the grasps are filtered with only motion-aware quality.
- **Baseline**. Using randomly sampled 10 grasps from the grasp database as the filtered grasps. We need to use a subset because checking IK for all grasps from the database during dynamic grasping is unfeasible with very low success rate that is not worth comparing.
- **No Traj. Seeding**. This ablation study picks the model from above with best performance and removes the trajectory seeding module to study its importance.
- **No Prediction**. Similar to *No Traj. Seeding*, this removes the prediction module to study its importance.

We evaluate a subset of the experiments on a real robot hardware to validate our approach. For this, we use the UR5 robot arm fitted with a Robotiq parallel jaw gripper to pick up an object moving on a conveyor belt.

B. Experimental Results and Discussion

Shown in Table I, our proposed methods with reachability and motion awarenesses or a combination of both outperform the baseline in all cases. The ablation studies demonstrate the importance of the trajectory seeding and motion prediction components.

1) *Effect of Grasp Planning*: The results show that reachability and motion awarenesses help extensively in dynamic grasping tasks. This is because our methods leverage these two ranking functions to quickly filter grasps that are likely to be reachable or stable conditioned on the current motion of the object. They help to focus on those grasps which are near-optimal given the object pose and motion and reduce unnecessary IK calls. Without reachability or motion awareness, even though all grasps in the grasp database are stable in static cases, the selected 10 grasps might not be as optimal given the current location where the object has moved or the current motion the object is following.

We also notice that *Ours (Reachability)* almost always performs better than *Ours (Motion-aware)*. This shows that in dynamic grasping setting, reachability awareness can be a slightly more important factor than motion-aware. Though being stable for the motion of the object, the selected grasps by motion-aware might still be unreachable and waste some IK computing time. We further investigate the relationship between reachability performance and the distance of the conveyor motion to the robot base, using linear motion

TABLE I: Simulation Experiments for the Kinova Mico (Top) and UR5 (Bottom) robot arms. For each entry, run on 7 objects and 100 trials each. We report success rate, dynamic grasping time (s) averaged over 700 trials.

Methods	Linear (3cm/s)	Linear (3cm/s) with Obstacles	Linear (2cm/s) with Top Slab	Linear (3cm/s) with Z Motion	Linear (1-3cm/s) Varying Speed	Circular (2cm/s)	Sinusoidal (1cm/s)
Ours (R + M)	0.799, 10.23s	0.796, 11.43s	0.781, 21.92s	0.814, 10.39s	0.869, 9.988s	0.875, 10.38s	0.895, 8.661s
Ours (Reachability)	0.802, 10.21s	0.795, 10.13s	0.759, 19.91s	0.806, 9.836s	0.836, 9.609s	0.861, 9.421s	0.867, 7.857s
Ours (Motion-aware)	0.722, 15.30s	0.780, 14.37s	0.697, 25.07s	0.710, 14.61s	0.819, 14.45s	0.857, 15.43s	0.827, 13.19s
Baseline	0.439, 23.85s	0.419, 24.15s	0.431, 38.12s	0.430, 23.19s	0.610, 25.72s	0.716, 24.89s	0.643, 20.22s
No Traj. seeding	0.769, 10.61s	0.777, 10.83s	0.706, 22.29s	0.800, 10.41s	0.827, 10.23s	0.857, 10.01s	0.827, 8.588s
No Prediction	0.610, 12.91s	0.614, 13.30s	0.609, 25.84s	0.737, 12.60s	0.761, 11.96s	0.767, 13.11s	0.807, 9.045s

Methods	Linear (5cm/s)	Linear (5cm/s) with Obstacles	Linear (3cm/s) with Top Slab	Linear (5cm/s) with Z Motion	Linear (3-5cm/s) Varying Speed	Circular (3cm/s)	Sinusoidal (1cm/s)
Ours (R + M)	0.874, 8.134s	0.854, 9.104s	0.748, 19.86s	0.858, 8.166s	0.917, 7.895s	0.909, 8.311s	0.946, 9.243s
Ours (Reachability)	0.857, 8.730s	0.841, 9.646s	0.652, 18.52s	0.872, 8.864s	0.907, 8.748s	0.890, 9.512s	0.925, 8.608s
Ours (Motion-aware)	0.744, 9.976s	0.752, 9.896s	0.675, 19.86s	0.717, 10.47s	0.854, 8.935s	0.840, 10.68s	0.930, 9.645s
Baseline	0.676, 12.64s	0.576, 13.63s	0.606, 22.89s	0.659, 12.65s	0.788, 12.84s	0.810, 14.09s	0.716, 17.19s
No Traj. seeding	0.849, 9.107s	0.810, 10.22s	0.631, 20.01s	0.836, 9.047s	0.906, 8.859s	0.899, 9.610s	0.904, 11.17s
No Prediction	0.284, 10.31s	0.269, 11.41s	0.457, 20.04s	0.261, 10.89s	0.344, 10.44s	0.594, 9.891s	0.310, 11.96s

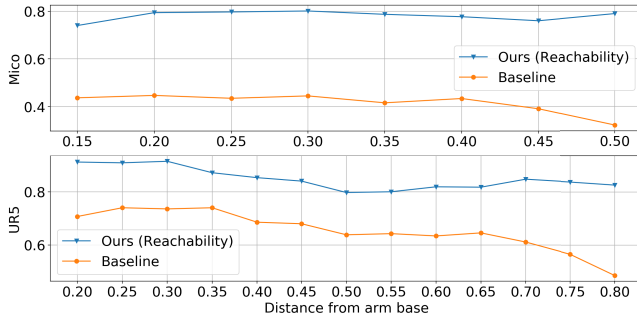


Fig. 6: Success rate vs. distance. Improvement from reachability awareness becomes more significant when the moving object is extremely close to or far from the robot.

while varying r , as demonstrated in Figure 6. We find that reachability being especially beneficial in difficult-to-reach near and far portions of the workspace.

Ours (R + M) outperforms *Ours (Reachability)* in all cases except linear motion for Mico and linear with Z motion for UR5 arm. Combining reachability and motion awareness include grasps that are both reachable and robust for the current motion. This combines the advantages of both methods and provides a pool of wider variety for PICKGRASP to choose from. For example, even though a grasp might not be the most reachable, but it can also be included in the filtered grasps because of high motion-aware quality. For the two cases where *Ours (Reachability)* outperforms *Ours (R+M)*, we believe it is because the most motion-aware grasps happen to have very low reachability but are closer to the current robot configuration. They are picked but cannot remain reachable and result in unnecessary grasp switches.

2) *Effect of Seeding in Arm Trajectory Generation:* We observe that the value of seeding during trajectory generation becomes significant for tasks when slab fixture is above the conveyor belt limiting the range of motion of the arm (column 3 of Table I, Mico and UR5 show a performance

drop of 7.5% and 11.7% respectively), and when the motion is sinusoidal and hard to model (column 7 of Table I, Mico and UR5 show a performance drop of 6.8% and 4.2% respectively). Without seeding, computing arm trajectory from scratch at each time step is computationally expensive. Besides, seeding makes the new trajectory solution similar to the previous one. Qualitatively, we noticed that seeding makes the arm motion less wavy given that the arm trajectories generated in subsequent time steps is seeded to be similar to the immediate previous one.

3) *Effect of Object Motion Prediction:* Our results also show that object motion prediction is an important component for dynamic grasping and we see the biggest drop in performance without motion prediction. This is expected as we can imagine without prediction in dynamic grasping, even with perfect grasp planning and optimal motion generation, the gripper will never catch the target because of the delay from computation. Its importance becomes more pronounced as the object's motion is hard to predict (non-linear / varying speed) and also when the gripper width is smaller. We observe that there is a bigger drop in performance for the UR5-Robotiq robot, compared to the Mico robot. Qualitatively, we observe that a lot of the failure cases occur during the approach-and-grasp phase where the robot finger narrowly knocks off the object. The wider span of the Mico gripper enables it to be more robust in this sense.

4) *Real Robot Demonstration:* We demonstrate our algorithm on the real robot by picking up objects 5, 6, 7 shown in Figure 5 as each object moves on a conveyor belt with no surrounding obstacles. We repeat this experiment 5 times and the success rates for objects 5, 6, 7 are 4/5, 5/5 and 3/5 respectively. Even though the object is moving relatively fast (4.46 cm/s), our method is able to pick the objects 5 and 6 reliably well. The robot is able to align its gripper along the narrow axis of the objects and pick them up while moving. The failure cases for object 7 is because the radius of the

tomato can be slightly smaller than the gripper span with a tight margin for error in the approach and grasp stage.

VI. CONCLUSION

This work presents a novel pipeline for dynamic grasping moving objects with reachability and motion awareness. We demonstrate its ability with a RNN motion predictor and adaptive motion planning with seeding. We show in experiments with various settings that these elements are important to improve the performance. This work is a model-based visual-pose feedback system. A future work will be the image-based analogue where arm-hand trajectory commands are directly generated based on image/depth image features using learning-based techniques.

REFERENCES

- [1] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, "Automated tracking and grasping of a moving object with a robotic hand-eye system," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 2, pp. 152–165, 1993.
- [2] X. Ye and S. Liu, "Velocity decomposition based planning algorithm for grasping moving object," in *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, pp. 644–649.
- [3] D. Kragic, H. I. Christensen *et al.*, "Survey on visual servoing for manipulation," *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, vol. 15, p. 2002, 2002.
- [4] N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour, and R. Dillmann, "Visual servoing for humanoid grasping and manipulation tasks," in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2008, pp. 406–412.
- [5] D. Kragic and H. I. Christensen, "A framework for visual servoing," in *International Conference on Computer Vision Systems*. Springer, 2003, pp. 345–354.
- [6] N. Houshang, "Control of a robotic manipulator to grasp a moving target using vision," in *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE, 1990, pp. 604–609.
- [7] W. J. Wilson, C. W. Hulls, and G. S. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 684–696, 1996.
- [8] N. Vahrenkamp, C. Böge, K. Welke, T. Asfour, J. Walter, and R. Dillmann, "Visual servoing for dual arm motions on a humanoid robot," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2009, pp. 208–214.
- [9] R. Mahony, P. Corke, and T. Hamel, "Dynamic image-based visual servo control using centroid and optic flow features," *Journal of Dynamic Systems, Measurement, and Control*, vol. 130, no. 1, p. 011005, 2008.
- [10] H. Xie, G. Li, Y. Wang, Z. Fu, and F. Zhou, "Research on visual servo grasping of household objects for nonholonomic mobile manipulator," *Journal of Control Science and Engineering*, vol. 2014, p. 16, 2014.
- [11] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," 2018.
- [12] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg, "Real-time perception meets reactive motion generation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1864–1871, 2018.
- [13] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 1710–1716.
- [14] C. Goldfeder and P. K. Allen, "Data-driven grasping," *Autonomous Robots*, vol. 31, no. 1, pp. 1–20, 2011.
- [15] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4–5, pp. 705–724, 2015.
- [16] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 3406–3413.
- [17] C. Eppner, A. Mousavian, and D. Fox, "A billion ways to grasp: An evaluation of grasp sampling schemes on a dense, physics-based grasp data set," *arXiv preprint arXiv:1912.05604*, 2019.
- [18] J. Weisz and P. K. Allen, "Pose error robust grasping from contact wrench space metrics," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 557–562.
- [19] J. Issac, M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, and S. Schaal, "Depth-based object tracking using a robust gaussian filter," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 608–615.
- [20] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal, "Probabilistic object tracking using a range camera," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3195–3202.
- [21] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," *arXiv preprint arXiv:1809.10790*, 2018.
- [22] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," *arXiv preprint arXiv:1901.04780*, 2019.
- [23] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
- [24] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.
- [25] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [26] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 489–494.
- [27] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*.
- [28] P. S. Schmitt, F. Wirschofer, K. M. Wurm, G. v. Wichert, and W. Burgard, "Planning reactive manipulation in dynamic environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 136–143.
- [29] P. S. Schmitt, F. Wirschofer, K. M. Wurm, G. v. Wichert, and W. Burgard, "Modeling and planning manipulation in dynamic environments," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 176–182.
- [30] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3671–3678.
- [31] D. Coleman, I. A. Şucan, M. Moll, K. Okada, and N. Correll, "Experience-based planning with sparse roadmap spanners," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 900–905.
- [32] F. Islam, O. Salzman, A. Agarwal, and M. Likhachev, "Provably constant-time planning and replanning for real-time grasping objects off a conveyor belt," *arXiv preprint arXiv:2101.07148*, 2021.
- [33] I. Akinola, J. Varley, B. Chen, and P. K. Allen, "Workspace aware online grasp planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2917–2924.
- [34] N. Marturi, M. Kopicki, A. Rastegarpanah, V. Rajasekaran, M. Adjigble, R. Stolkin, A. Leonardis, and Y. Bekiroglu, "Dynamic grasp and trajectory planning for moving objects," *Autonomous Robots*, vol. 43, no. 5, pp. 1241–1256, 2019.
- [35] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *IJRR*, vol. 28, no. 7, 2009.
- [36] A. Menon, B. Cohen, and M. Likhachev, "Motion planning for smooth pickup of moving objects," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 453–460.
- [37] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.
- [38] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," *GitHub repository*, 2016.