Motion-Aware Underwater Robotic Grasping

Robin Kr. Singh and Geoffrey A. Hollinger

***Abstract*— Grasping moving objects underwater introduces an entirely new set of challenges in contrast with structured terrestrial or industrial environments. Underwater environments are highly unstructured with waves causing the water particles to move in circular orbits that diminishes with depth which results the object to oscillate in a unique way. This causes a typically high confident and stable grasp to become an unstable and off-target grasp when the object pose changes, rendering the existing grasping methods useless. Dynamic grasp generation needs to be adaptive, real time, robust to noise, and have low computing needs. We present a novel dynamic grasping framework that processes a streaming point cloud generated from depth video in real time to generate 6-DOF 4D grasps. By tracking the successive position of point cloud, we predict the current motion of objects. Using the relative position and motion predictor, we decide on the most confident grasp. We recorded the motion of multiple objects under three different wave conditions at the O.H. Hinsdale Wave Research Laboratory at Oregon State University, USA, and tested them in a simulation environment using pyBullet. Our method achieved an average success rate of 70% across four different object configurations under three different wave conditions.**

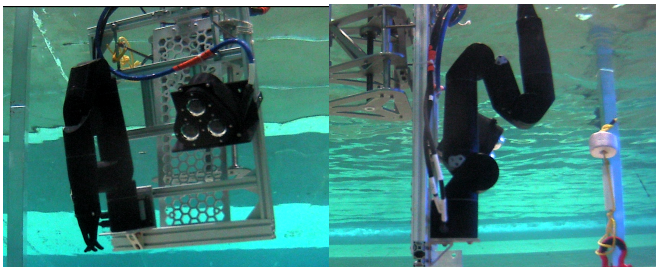
I. Introduction

There has been significant progress in developing grasping algorithms [1]; however, most of this work focuses on either static objects [2] or objects with uniform motion like an object on conveyor belt [3] which can be used in industries but fails in real life scenarios. This necessitates the development of grasping algorithms that are able to generate grasps and perform robustly in unstructured environments.

Many state-of-the-art grasping algorithms focus on detecting objects and generating as many grasps as possible using a trained network and then filter out grasps using multiple criteria to get the most confident grasp [2]. While this might work for static objects, it is not feasible for objects in motion as the object position is ever changing and the grasp generation is not fast enough to keep up with the change, even more so in dynamic environments where the motion might not be uniform. Currently dynamic grasping requires introduction of a number of assumptions such as grasping only when the object has come to rest or only attempting top-down grasps [4] or providing the object motion as input [5] or during training [3]. Other work focuses on grasping a static object in between moving obstacles [6], or dynamic grasping with static objects in slightly moving scene [7]. But these methods fail in dynamic environment even more so in unstructured environments like underwater scenarios.

The authors are with the Collaborative Robotics and Intelligent Systems Institute, Oregon State University, Corvallis, OR 97331, USA. {singhrob,geoff.hollinger}@oregonstate.edu

This work was supported in part by Office of Naval Research (ONR) awards N0014-21-1-2052 and N00014-22-1-2114.



(a) Front view of underwater setup inside testbed (b) Bravo arm next to an oscillating buoy under wave conditions

Fig. 1: Camera and Bravo arm [8] setup at O.H. Hinsdale Wave Research Laboratory. We recorded the streaming point cloud from the Trisect camera [9] in ROS bags under three different wave conditions with multiple objects attached to the seabed

In our work we focus on generating feasible grasps while the object is still in motion by utilizing motion predictor that generates future positions of the object by tracking the object's centroid. To relax some of the assumptions mentioned in other work we do not provide prior knowledge of object motion or wave conditions to our model. In addition to this unlike other works [3] we do not have prior information about the set of objects. Since the objects underwater can be deformed or have different shapes (like buoy) we only focus on generating feasible grasp at the object's centroid with as less interference to the object motion as possible. The main contributions of our work is our centroid trackng method which allows us to generalize motion of objects under different wave conditions. This allows us to predict the grasp in future pose based on current motion of the object. To the best of our knowledge, our work is the first attempt to generate 6-DOF grasps of a moving object underwater by predicting it's future position.

II. Related Work

Grasping static objects can be achieved through various methods such as tactile sensing [10] [11], visual servoing [11] [12], and object classification [13]. However, grasping in unstructured environments presents unique challenges even for static objects [14]. While database-based methods [13] [15] boast high success rates, this is largely due to the availability of extensive labeled datasets [1]. Some labeled 3D model datasets, such as PartNet [16], ShapeNet [17], and YCB [18], contain detailed information including textures, shapes, hierarchies, weight, and rigidity. These datasets have proven invaluable for grasping static objects, as demonstrated by numerous studies over the years [1]. However, these datasets are less useful when the same objects are in motion, especially when the motion is influenced by the environment.

Despite exciting advancements [3] [19] [20] in leveraging large datasets to improve grasping of moving objects, these methods often fail in unstructured environments like underwater due to their inherent limitations of requiring object motion to be uniform [21] or from a given subset [3].

III. Problem Formulation

Grasping objects in dynamic and unstructured environments, such as underwater settings, presents a significant challenge due to the complex and unpredictable motion of the objects. In an underwater environment, objects are often subjected to oscillatory and irregular motions due to wave forces and other hydrodynamic effects. These movements are characterized by a combination of periodic oscillations and low-frequency perturbations, leading to complex motion patterns that are difficult to predict. The object's motion is further complicated by factors such as buoyancy, tethering, and the influence of the surrounding fluid dynamics.

The central problem in this context is to develop a grasping algorithm that can adapt to the dynamic nature of the underwater environment by accurately predicting the future pose of the moving object. This requires not only the ability to model the object's complex motion but also to generate and execute grasps in real-time, considering the object's future position and the robot's own motion constraints.

Formally, the problem can be stated as follows: Given an object with a known current pose, subjected to dynamic underwater conditions, the goal is to predict its future pose after a specified time interval and determine a grasp configuration that can be executed effectively within this time frame. This involves the challenges of motion prediction, grasp generation, and real-time trajectory planning, all of which are critical for successful grasping in such a complex environment.

IV. Methods

A. Data Collection Setup

The physical setup as shown in Figure [1] comprises an underwater stereo camera system, Trisect [9], and a Bravo robotic arm. The bravo robotic arm [8] is a 7-function manipulator arm designed for inspection-class ROVs (Remotely Operated Vehicles) developed by Reach robotics for subsea inspection. The Trisect camera, developed at the University of Washington, is designed for embedded, ROS-based computer vision applications. The Robot Operating System (ROS) [22] is an open-source framework for robotics software development. It provides essential services like hardware abstraction, device control, message-passing, and package management. ROS is designed to support code reuse and modularity, making it easier to develop complex robotic systems. The Trisect provides a continuous stream of point cloud data, which is essential for tracking and predicting the motion of objects in underwater environments. Our data collection was conducted at the O.H. Hinsdale Wave Research Laboratory, Oregon State University, where we recorded ROS bags under three different wave conditions as shown in Table [I] with multiple objects tethered to the seabed. The three wave conditions used are inspired by marine energy monitoring applications. [23] It should be noted that the wave condition information is not fed to our framework and is only used to measure performance.

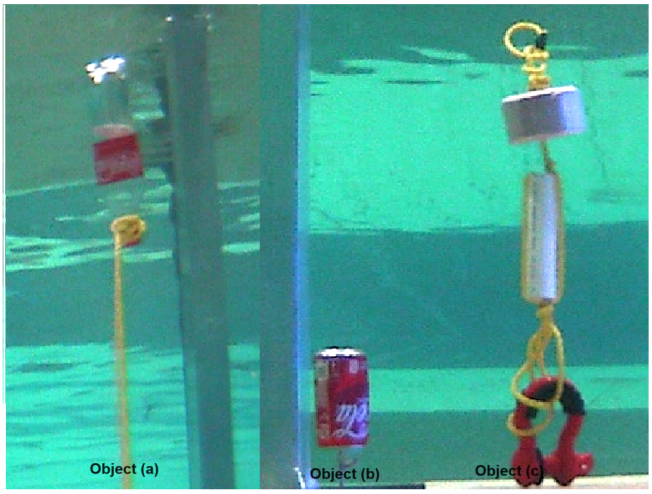


Fig. 2: The three objects used for data collection: (a) inverted bottle (b) beverage can (c) buoy

B. Point Cloud Processing

The first step in our method involves processing the continuous point cloud stream from the Trisect camera. The point cloud data is filtered to remove noise and irrelevant points, ensuring a clean and accurate representation of the object's surface. Point cloud data captured in underwater environments often contains noise and outliers due to various factors such as light refraction, particulate matter, and water movement. To address these challenges, we employ a series of simple filtering techniques.

After filtering, the clean point cloud is converted into a continuous mesh. A polygonal mesh is created that approximates the surface defined by the point cloud data. The resulting mesh is a more structured representation of the object, facilitating easier manipulation and analysis in the pyBullet simulator. This mesh is further smoothened by using point cloud data from multiple positions. The refined mesh is used to validate generated grasps and detect collisions.

C. Centroid Tracking

Within the pyBullet environment, we identify the object's centroid and track its oscillatory motion. In our research, we have focused on uniform objects, as depicted in figure [2], to simplify the process of locating the centroid. This assumption obviates the need for point cloud completion - a separate and complex research problem necessary for finding the centroid of non-uniform objects. The object, being buoyant and tethered underwater by a string, moves in the circumference of a sphere with the ocean bottom acting as the center. This motion is influenced by both periodic oscillations and a regular sway of the oscillation plane by a few degrees. By accurately tracking the centroid's movement, we make predictions for the object's future pose.

Let *C(t)=(x(t),y(t),z(t))* represent the position of the object's centroid at time t in 3D space. The centroid's motion is tracked continuously, capturing its oscillatory behavior influenced by underwater wave conditions. Given the object's buoyancy and tethering, its motion can be approximated as periodic oscillations around the point where the string is anchored to the seabed.

D. Motion Model

The motion of the centroid can be described by a combination of harmonic functions to account for the periodic motions of wave and additional terms for the sway of the oscillation plane. The position of the centroid, *C(t)* can be modeled as follows:

$$x(t) = A_x \sin(\omega t + \phi_x) + \delta_x(t), \tag{1}$$

$$y(t) = A_y \sin(\omega t + \phi_y) + \delta_y(t), \tag{2}$$

$$z(t) = A_z \cos(\omega t + \phi_z) + \delta_z(t). \tag{3}$$

Here, A_x, A_y, and A_z are the amplitudes of oscillation in the x, y, and z directions, respectively. ϕ_x, ϕ_y, and ϕ_z are the phase shifts. The terms $\delta_x(t)$, $\delta_y(t)$, and $\delta_z(t)$ represent the sway of the oscillation plane, which can be modeled as low-frequency perturbations.

E. Future Pose Prediction

To predict the future pose of the centroid after a time interval Δt, we utilize the tracked motion data up to the current time t. The future position $C(t + \Delta t)$ is given by:

$$x(t + \Delta t) = A_x \sin(\omega(t + \Delta t) + \phi_x) + \delta_x(t + \Delta t), \tag{4}$$

$$y(t + \Delta t) = A_y \sin(\omega(t + \Delta t) + \phi_y) + \delta_y(t + \Delta t), \tag{5}$$

$$z(t + \Delta t) = A_z \cos(\omega(t + \Delta t) + \phi_z) + \delta_z(t + \Delta t). \tag{6}$$

The time interval Δt is typically chosen as the time it takes for the object to oscillate from one end of its motion to the other. This period, T, is given by:

$$T = \frac{2\pi}{\omega}, \tag{7}$$

By choosing $\Delta t = T$, we ensure that the motion prediction aligns with the most uniform phase of the object's oscillation, providing a reliable estimate of the future position.

F. Grasp Trajectory Planning

With the final grasp position determined using the motion predictor, the next crucial step involves planning a grasp trajectory that ensures the robotic arm does not interfere with the object's oscillating path and that the grasp can be executed within the time interval t. This approach ensures the object reaches its predicted position, allowing for a successful grasp at the appropriate moment.

To generate a suitable grasp direction, the trajectory must be designed to approach the object in a way that minimizes interference with its motion. The grasp direction should not be entirely parallel to the object's motion but should converge towards the predicted position towards the end of the motion cycle. This strategy ensures the robotic arm can position itself optimally without disrupting the object's oscillation.

Let P_f represent the predicted final position of the object's centroid at time $t + \Delta t$. The trajectory planning process involves the following steps:

1) Initial Approach Path: Define an initial approach path that positions the robotic arm near the object's predicted motion plane but outside the direct path of oscillation. The initial approach position P_{init} can be determined by offsetting P_f along a direction orthogonal to the object's motion.

$$P_{init} = P_f + \alpha \mathbf{n}, \tag{8}$$

where α is a scalar defining the distance from the object's path, and $\mathbf{n}$ is a unit vector orthogonal to the object's motion direction. This ensures the arm approaches from a safe distance, avoiding interference.

2) Intermediate Waypoints: To smoothly transition from P_{init} to P_f, we define intermediate waypoints P_i that guide the trajectory. These waypoints are strategically placed to ensure the arm's motion gradually aligns with the object's motion direction while not interfering with it's motion. The number of waypoints N and their positions can be defined as follows:

$$P_i = P_{init} + \frac{i}{N}(P_f - P_{init}) + \beta_i \mathbf{n}, \tag{9}$$

where $i = 1, 2, \ldots, N-1$ and β_i are small perturbations ensuring a gradual alignment with the object's motion direction. The movement to final waypoint P_N coincides with the predicted final position P_f and is in the exact opposite direction as the motion of the object giving higher chance of success.

3) Time-Constrained Path Generation: The generated trajectory must be executable within the time interval Δt. To ensure this, the velocity profile of the robotic arm is constrained such that the entire path from P_{init} to P_f can be completed within Δt.

The path length L is calculated as the sum of the distances between successive waypoints:

$$L = \sum_{i=0}^{N-1} \|P_{i+1} - P_i\|. \tag{10}$$

To complete the trajectory in time Δt, the average speed v of the robotic arm must satisfy:

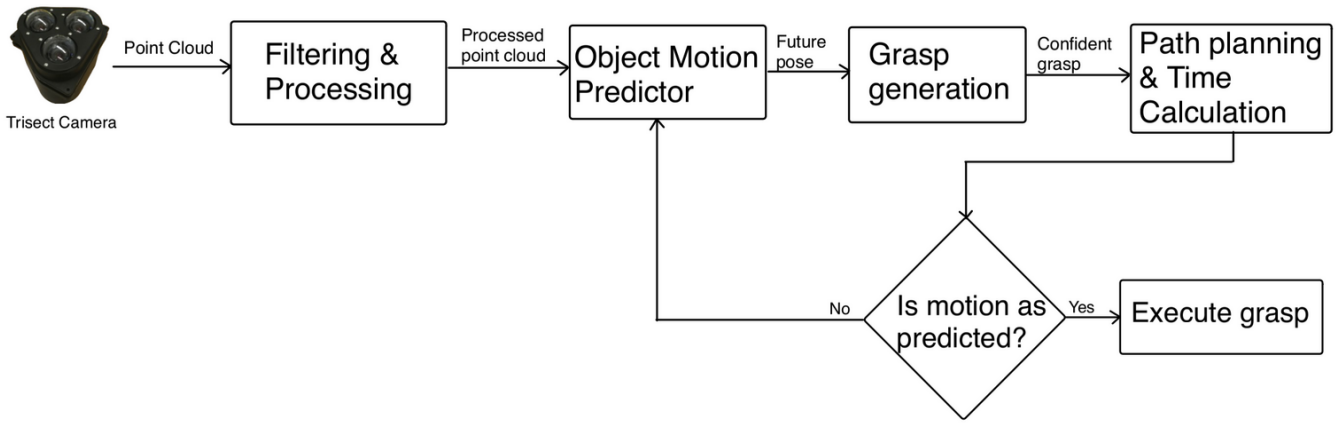


Fig. 3: Proposed motion-aware grasping framework. An underwater 3D camera camera provides continuous stream of point cloud which is filtered, processed, and fed to the object motion predictor that generates the future pose of object. This predicted pose is used to generate possible grasps and the most confident grasp is used for path planning and time calculation. The decision-making module makes sure that the motion hasn't changes drastically and only then grasp is executed.

$$v = \frac{L}{\Delta t}. \tag{11}$$

The maximum allowable speed and acceleration of the robotic arm are taken into account to generate a feasible velocity profile. The trajectory is then planned using a time-scaling algorithm that adjusts the arm's speed to ensure the grasp is executed within Δt.

4) Smooth Path Execution: Using the initial approach position and intermediate waypoints, a smooth trajectory is generated. This trajectory can be represented as a cubic spline that interpolates through P_{init}, P_i, and P_f. The spline ensures a smooth transition and minimizes abrupt changes in direction.

Let $\mathbf{T}(t)$ represent the trajectory of the robotic arm from time t to $t + \Delta t$. The cubic spline is defined such that:

$$\mathbf{T}(t) = \sum_{j=0}^{3} \mathbf{a}_j t^j, \tag{12}$$

where $\mathbf{a}_j$ are the spline coefficients determined by the boundary conditions and the positions of P_{init}, P_i, and P_f.

G. Path Execution and Grasp

The planned trajectory $\mathbf{T}(t)$ is executed by the robotic arm, ensuring it approaches the object without disrupting its oscillation and within the specified time interval Δt. The arm moves through the intermediate waypoints, gradually aligning with the object's motion direction. At the end of the trajectory, the arm reaches the predicted final position P_f and executes the grasp.

By carefully designing the grasp trajectory to avoid interference with the object's oscillation and ensuring it can be completed within Δt, the robotic arm can successfully perform grasps on moving objects in dynamic underwater environments. This approach ensures robust and reliable grasping, accommodating the complexities of underwater motion.

H. Decision-Making Module

The final component of our framework is decision-making module, which is integral to the dynamic grasping framework, ensuring that the robotic arm only attempts a grasp if the predicted motion closely matches the actual recorded movement of the object. We continuously track the predicted and actual movement, using a defined metric to evaluate the accuracy of the predictions.

To quantify the accuracy of the motion predictions, we define an error metric based on the Euclidean distance between the predicted and actual positions of the object's centroid. Let $e(t)$ represent the error at time t:

$$e(t) = \|\hat{C}(t) - C(t)\|. \tag{13}$$

This error metric provides a measure of the deviation between the predicted and actual positions. A smaller error indicates a more accurate prediction, while a larger error signifies a discrepancy between the predicted and actual movements.

The decision-making module uses the error metric to determine whether to proceed with the grasp or restart the prediction and planning process. A threshold ϵ is defined to evaluate the accuracy of the predictions. If the average error over a time window Δt_w is below the threshold, the prediction is considered accurate enough to proceed with the grasp. Otherwise, the prediction is deemed unreliable, and the process is restarted.

The average error $\bar{e}$ over the time window Δt_w is calculated as:

$$\bar{e} = \frac{1}{\Delta t_w} \int_{t-\Delta t_w}^{t} e(\tau)\, d\tau. \tag{14}$$

The decision criteria can be expressed as:

- If $\bar{e} \leq \epsilon$, the prediction is accurate, and the grasp is executed.
- If $\bar{e} > \epsilon$, the prediction is inaccurate, and the steps are restarted.

When the decision-making module determines that the prediction is accurate (i.e., $\bar{e} \leq \epsilon$), it gives a green light to proceed with the grasp. The robotic arm follows the planned trajectory and executes the grasp at the predicted final position P_f.

If the prediction is not accurate (i.e., $\bar{e} > \epsilon$), the module restarts the process by re-evaluating the motion data, updating the motion predictor, and generating a new grasp trajectory. This iterative process ensures that the robotic arm only attempts a grasp when there is a high level of confidence in the motion predictions.

V. Results

To validate the effectiveness of our dynamic grasping algorithm, we implemented our algorithm in PyBullet [24] [25] simulator. PyBullet is a python module designed for physics simulation, particularly in robotics, and machine learning. It allows users to load articulated bodies from various file formats like URDF, SDF, and MJCF and supports forward and inverse dynamics, kinematics, collision detection, and ray intersection queries. Using ROS we were able to record a continuous stream of point cloud data as ROS bags. We filtered the recorded point clouds to remove noise and outliers to get only the object's point cloud as best as possible. This filtered point cloud data was converted to mesh such that it can be directly loaded to PyBullet. The motion of these meshes were the same as the object under wave conditions, shown in Table I allowing us to replicate the object motion in simulation.

Wave Condition	Wave Height (m)	Peak Period (s)	Max Horizontal Velocity (m/s)	Max Vertical Velocity (m/s)
1	0.25	1.25	0.2218	0.2157
2	0.135	1.08	0.1036	0.0894
3	0.15	1.2	0.0523	0.0012

TABLE I: Wave characteristics

The motion of three different objects tethered to the seabed was recorded under each wave condition. The third object, a beverage can, was recorded twice: once when it was entirely empty (more buoyant) and once when it was half empty (less buoyant). This approach allowed us to assess how our algorithm performs with the same object and wave condition but different motion characteristics. In total, we had four different object configurations subjected to three different wave conditions.

We conducted 20 grasping trials for each object configuration using our method. If the first attempt failed, we restarted and made two more attempts. We evaluated the performance of our dynamic grasping algorithm, and the results are presented in Figure 4. Our success rate ranged from 50% to 80%, with higher rates observed for the buoy and the empty beverage can. After reattempts we see improvements in success rates for all objects ranging from 5% to 20%.

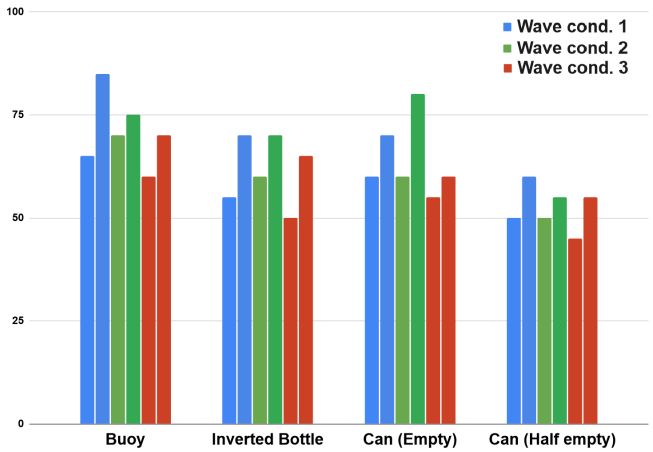


Fig. 4: Final success rate of object under different wave conditions. Blue, Green, and Red represents wave conditions 1,2, and 3 respectively from Table (I)

We documented all the reasons for unsuccessful grasps for our method. These causes of errors are systematically categorized and presented in Table II which provides a clear and comprehensive overview of the factors contributing to grasping failures.

Reason behind failures	Quantity (%)
Motion prediction error	15
Collision	40
Timing (execution)	15
Abrupt change before grasp	30

TABLE II: Distribution of Errors in Dynamic Grasping Algorithm

Motion prediction errors encompass failures due to inaccuracies in predicting an object's future position. Our algorithm relies on motion prediction to anticipate the object's location at the time of grasp. Errors in this prediction can lead to unsuccessful attempts, as the gripper may not align correctly with the object's actual position. It was observed that motion prediction was the smallest cause of error. The highest percentage of failures was due to collisions, which occur when the gripper and the object collide. These collisions mostly happened during grasp attempts, but sometimes the gripper deviated from its trajectory, causing it to collide with the object before reaching the predicted position. While motion prediction errors might also lead to collisions, we did not categorize them as such since the prediction itself was flawed.

Timing-related failures occur when there is a delay or mismatch between the predicted motion and the actual execution of the grasp. This can be due to system latency or delays in the gripper's response, causing the grasp to miss the object. Finally, abrupt changes before grasping are caused by sudden, unpredictable changes in the object's motion just before the grasp attempt. There is always a probability of such abrupt changes, and our algorithm is not yet able to compensate for them quickly enough.

Since the motion predictors in prior work are designed for terrestrial structured environments, we compared our method to a general linear motion predictor. The linear motion predictor assumes that the object's motion can be

approximated by a uniform trajectory, which is a common approach in terrestrial structured environments. This predictor calculates the object's future position based on its current velocity and direction, without accounting for the complex dynamics of underwater environments.

Similar to our approach, if the first grasp attempt using the linear motion predictor failed, we restarted the process and made two additional attempts for each wave condition. This ensured a fair comparison between the two methods under identical conditions.

We then compared the average success rate for each object configuration between our method and the linear motion predictor. By doing so, we aimed to highlight the advantages of our algorithm. The results of our comparison are presented in Figure 5.

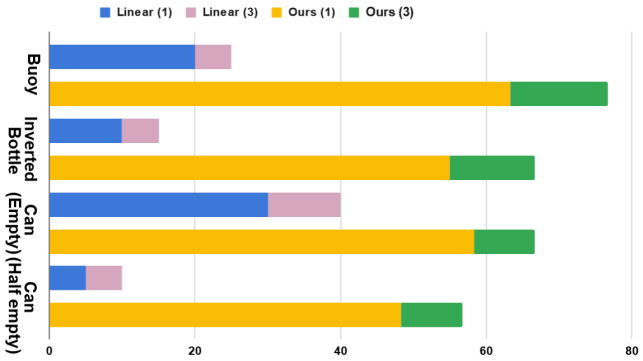


Fig. 5: Average success rate over 3 different wave conditions linear vs ours

Our method demonstrates improved performance on the first attempt, and the two reattempts further enhance the overall success rate as shown in Figure5.

VI. Future Work and Discussion

We introduced a dynamic grasping algorithm designed for 6-DOF grasp synthesis in underwater environments, focusing on real-time prediction and execution. Our method utilizes a motion predictor to estimate the future pose of an object based on its current oscillatory motion, enabling the generation and planning of viable grasps at precise future positions. Our simulations conducted under various wave conditions demonstrated the ability to robustly track and predict object movement, resulting in successful grasps approximately 70% of the time for different object configuration in challenging dynamic settings.

This approach is particularly relevant for underwater mobile manipulation, where the unpredictability of the environment necessitates rapid and adaptive grasping strategies. By predicting future object positions and planning grasps accordingly, our algorithm enables the robotic system to execute grasps in synchronization with the object's motion, significantly improving the robustness and efficiency of the grasping process.

The ability to perform dynamic grasping in unstructured underwater environments represents a significant step forward in the field of robotic manipulation. Future work will focus on integrating this grasp in physical robot and verifying if this method generalizes with other non-uniform shaped objects. Additionally, exploring active perception strategies to optimize object tracking and prediction under varying environmental conditions presents an exciting avenue for improving the overall grasping performance in complex unstructured scenarios.

References

[1] L. X. Xie Z and R. C, "Learning-based robotic grasping: A review," *Front. Robot. AI*, 2023.

[2] R. N. et al., "Deep learning approaches to grasp synthesis: A review," in *IEEE Transactions on Robotics*, vol. 39, no. 05, 10 2023.

[3] S. S. I. Akinola, J. Xu and P. K. Allen, "Dynamic grasping with reachability and motion awareness," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[4] C.-C. Wong, M.-Y. Chien, R.-J. Chen, H. Aoyama, and K.-Y. Wong, "Moving object prediction and grasping system of robot manipulator," *IEEE Access*, vol. 10, pp. 20 159–20 172, 2022.

[5] P. 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.

[6] 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.

[7] P. C. D. Morrison and J. Leitne, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," 2018.

[8] "Reach bravo: Larger platform rov manipulator - reach robotics." [Online]. Available: https://reachrobotics.com/products/manipulators/reach-bravo/

[9] "Trisect underwater stereo camera." [Online]. Available: https://trisect-perception-sensor.gitlab.io/

[10] R. Howe, N. Popp, P. Akella, I. Kao, and M. Cutkosky, "Grasping, manipulation, and control with tactile sensing," in *Proceedings., IEEE International Conference on Robotics and Automation*, 1990, pp. 1258–1263 vol.2.

[11] D. Guo, F. Sun, B. Fang, C. Yang, and N. Xi, "Robotic grasping using visual and tactile sensing," *Information Sciences*, vol. 417, pp. 274–286, 2017.

[12] 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.

[13] F. Sun, C. Liu, W. Huang, and J. Zhang, "Object classification and grasp planning using visual and tactile sensing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 7, pp. 969–979, 2016.

[14] T. R. Player, D. Chang, L. Fuxin, and G. A. Hollinger, "Real-time generative grasping with spatio-temporal sparse convolution," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 7981–7987.

[15] Q. Bai, S. Li, J. Yang, Q. Song, Z. Li, and X. Zhang, "Object detection recognition and robot grasping based on machine learning: A survey," *IEEE Access*, vol. 8, pp. 181 855–181 879, 2020.

[16] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding," in *IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 909–918.

[17] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[18] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *International conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 510–517.

[19] J. Liu, R. Zhang, H.-S. Fang, M. Gou, H. Fang, C. Wang, S. Xu, H. Yan, and C. Lu, "Target-referenced reactive grasping for dynamic objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 8824–8833.

[20] P. Chen and W. Lu, "Deep reinforcement learning based moving object grasping," *Information Sciences*, vol. 565, pp. 62–76, 2021.

[21] 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)*, 2018, pp. 644–649.
[22] "Ros (robot operating system)," https://wiki.ros.org/.
[23] R. Vivekanandan, D. Chang, and G. A. Hollinger, "Autonomous underwater docking using flow state estimation and model predictive control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1062–1068.
[24] "Pybullet documentation." [Online]. Available: https://usermanual.wiki/Document/pybullet20quickstart20guide.479068914.pdf
[25] "Pybullet github repo." [Online]. Available: https://github.com/bulletphysics/bullet3/blob/master/docs/pybullet_quickstart_guide/PyBulletQuickstartGuide.md.html