Trajectory Planning for Reaching an Object on a Table Using Accumulated Action Data

Yusuke MORIYA, Nahum ALVAREZ, Kimitoshi YAMAZAKI

***Abstract*— This paper describes a robot trajectory planning method for grasping an object placed on a table. In conventional methods, the more obstacles are on the table, the more calculation costs are needed to find a course for avoiding the obstacles; to overcome this problem we use Accumulated Action Data from previous situations. The Action Data is composed of obstacles arrangement and manipulation trajectory and the manipulator acts on the information that is decided by comparing data arrangement with the present arrangement. For evaluating this method, we performed experiments in different conditions on a simulator.**

I. Introduction

When we think about a robot supporting people in daily activities, we can imagine an array of situations where the robot will have to take a concrete object from a surface containing other different objects, and of course, every time can have different features, like a different goal to grasp or different objects' arrangement, position or number. In order to effectively grasp the robot's objective we need to calculate a route for the robot's arm avoiding collisions with any potential obstacle. Having this in account, by evaluating the current status of the arm's joints and the environment's objects, the robot have to perform a collision check before every decision of the next joint angle for the arm. However, if the number of joints or objects increases, the solution space increases exponentially as well and as a consequence, this generates large processing times [1]. To overcome this issue, several methods have been proposed [2] [3] [4]. In this paper we propose a new method for calculating trajectories by exploring past experience, and avoiding to perform costly collision checks.

In our proposal scenario, we use previously recorded experience from examples with a number of obstacles displayed in the operation environment. These experiences, called "Action Data", are composed of the mapping of objects present in the environment and the data from the robot arm's joint angles, and are stored in large numbers in an Action database we called "Accumulated Action Data".

Such a method requires large quantities of data which is not practical for individual robots, especially if we want for them to adapt to different environments. However, if many robots share the Action database, this issue is lighten. In order to counter that problem, this system was deployed in a distributed server in the cloud, allowing to be used by several client robot platforms. Fig. 1 shows the concept architecture. In this architecture, each robot (assuming they have the same joint structure) initially contributes with its own environment to the creation of the Action database, which will be integrated on the cloud system. Afterwards, each robot can ask for trajectory planning to the remote system by sending its current environment data. Using this framework, we aim to solve the issue of needing a huge amount of Action Data. Also, as the whole process is held in the server, the robots do not need to spend important resources in terms of computing power.

This cloud architecture concept can be used with different action planning methods, but as we stated previously, an important part of the process is the method to search the appropriate data on action planning. The method we present contains a novel technique for trajectory planning that doesn't use collision checking, thus increasing the efficiency of the system. In other words, the contribution of our method is that calculates successful trajectories with a high success rate but doesn't spend time in checking collision with the potential obstacles. Instead of that, in order to select which action is more convenient to do, the trajectory planner will select the Action Data whose environment map is the most similar to the current environment and whose results didn't generate a collision. Once selected, the robot will adopt the joint angle

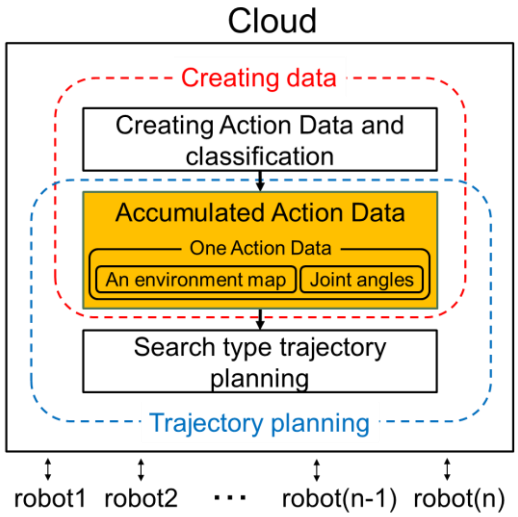


Fig. 1. Design concept of the planning in this paper. The same platform is able to access to cloud and use two programs: creating data and trajectory planning.

Yusuke Moriya is with the Shinshu University, Nagano, Japan (e-mail: 15tm129d@shinshu-u.ac.jp).

Nahum Alvarez is with the Shinshu University, Nagano, Japan (e-mail: nahum@shinshu-u.ac.jp)

Kimitoshi Yamazak is with the Shinshu University, Nagano, Japan (e-mail: kyamazaki@shinshu-u.ac.jp).

associated to that Action Data. This method avoids the processing time of joint angle planning and collision checking, depending only on the time spent in retrieving the past experience.

This paper is organized as follows: Section II describes other related systems, discussing their potential issues, Section III explains the architecture of our system, Section IV shows how the Action Data is created, and Section V details our trajectory planning method. Finally, Section VI describes the evaluation carried out for our system and its results, and Section VII presents our conclusions.

II. Related Work

Trajectory planning using a set of learnt data has been subject of extensive research using a number of different techniques [5]: often using learning by demonstration gives promising results [6] [7], but requires time and is not well prepared to counter variable situations or unexpected environments [8]. Also, learning techniques require expert-generated data, and in some cases is very specific to a certain task [9]. A possible solution for solving those problems is to use some form of dynamic or iterative planning, like in [10], but if the environment is too complex, especially when dealing with collisions, the process may require an unaffordable amount of time. In order to counter that issue, there has been different approaches: for example, it is possible to reduce the collision checking by doing it only in points that are likely to be the best trajectory, although it can generate sometimes non-optimal paths [2]; other works reduce the amount of calculation by planning with combinations of simple movements [11] [12]. However, in order to select the proper combinations, is necessary to have a large processing capability. Other approach, used in [3] generates a dynamic map based in precomputed roadmaps around obstacles, keeping a balance between stored data about the environment and processing effort. However, it still has some problems when objects in the environment change their position.

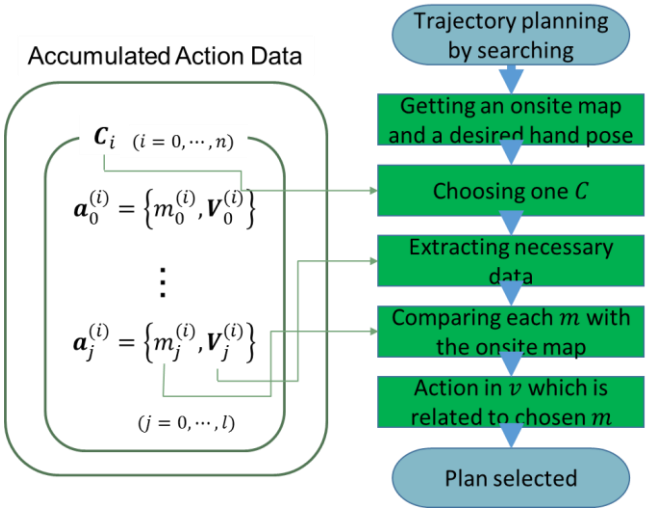


Fig. 2. The flow of trajectory planning using Accumulated Action Data. Action Data entries are grouped in classes (expressed as $\boldsymbol{C}$). Each entry (expressed as $\boldsymbol{a}$) is composed of a map (expressed as m) and a set of joint angles (expressed as **V**).

It would be desirable (and our motivation) to find a method having the adaptive features of the dynamic planning approaches but avoiding costly collision checking. Our approach is to use the data generated by a learning system in the training stages and apply it in the trajectory planning.

These data structures recorded in the learning database are an important factor too. Approaches like digital elevation maps (often called DEM or 2.5D map) have been used [14]. However, DEM is not suitable to our assumption, since it is difficult to express objects with parts over other objects or over the tabletop. Another popular method for mapping is to use voxels [13], which we use in our method as it fits better our application environment. Concretely, in each entry of our Accumulated Action Data we store two components: a representation of the environment map and the trajectory's set of joint angles.

III. Trajectory Planning Architecture

Our system is used by the robot in order to select the most appropriate action for grasping a certain object in its environment. The environment we decided to work with is composed of a table with a number of random objects placed on it, as well as a goal object. Also, we divided the environment in tiles in order to classify different arrangements depending of where is placed the goal object; we will see later on that this classification allows us to decrease the process time of the system).

The robot objective is to use its manipulator hand for grasping the goal object and avoiding to collide with any of the other objects in its path. This process is carried out by selecting successful past actions in similar environments, stored in a database. The idea is that if we find a past experience similar enough, the same action will be successful as well.

Fig. 2 shows a flowchart depicting the trajectory planning process used by our system and the structure of the past actions database. The Accumulated Action Data, as we have called the database, is grouped in classes depending of the tile where the goal object is placed, expressed as "$\boldsymbol{C}_i (0 \leq i \leq n)$". Each class contains a list of action entries, "**a**", with two components: the first one is a map containing the arrangement of the objects placed in the environment, expressed as "**m**". The second component is composed of a list, "**V**", of joint angles sets "**v**" which are obtained from joint angle planning using calculation in the map situation, and contain the steps of joint angles to perform the trajectory for grasping the goal object. If the manipulator can grasp the goal object in several ways, there are several **v** per map.

Trajectory planning using our Accumulated Action Data is conducted by the following procedure:

1) The current objects arrangement on the table and target hand pose in that situation are obtained and mapped.

2) One class **C** is chosen in consideration of the goal object's tile.

3) The system selects the actions entries whose final joint angles in **V** correspond with the goal object.

4) From the list of actions selected in the previous step, the system selects the one with the map **m** most similar to the current arrangement.

5) The manipulator executes the movements described in the **v** component corresponding to the action selected in the previous step.

IV. Environment Mapping Based Planning

In order to construct the Accumulated Action Data our system needs a method for obtaining the components of each one of its entries: a map of the objects arrangement and the joint angles list for each plan. Also, with such a method it's possible to update the Accumulated Action Data anytime we want. This chapter describes how to obtain those two components.

A. Objects Arrangement Map

In this paper, environments are generated virtually on a simulator, so we generate the maps from digital data. However, it is also able to obtain the same maps from real space by getting its objects' arrangement information by using three-dimensional distance image sensor.

We define a voxel space dividing the space above the table plane into unit cubes (shown in fig. 3a). The voxels will be represented with a binary number in a tridimensional grid, with all initial voxel values being 0. If the referred voxel overlaps with any obstacles, its correspondent grid value is given a 1 (shown in fig. 3b as red-colored blocks). We call this set of voxels "Objects arrangement map". The grid is compressed and recorded, being ready for its use in next stages.

B. Joint Angles

After generating the Objects arrangement map, the hand's pose able to grasp the goal object is decided.

The manipulator is an articulated structure, so the geometric shape of the links and hand are known. Thus, we can use for trajectory planning the BiRRT method[15] and obtain an ordered list of joint angles describing a trajectory able to reach the goal pose without collision between two arm's links or between a link and any object. Also, we obtain trajectories for different possible goal hand poses, allowing solutions for environments where concrete goal hand poses are limited due to the objects' arrangement.

V. Trajectory Planning

Once we have enough samples in our Accumulated Action Data, we can proceed with the trajectory planning algorithm depicted in section III and in the Fig. 1. In this section we describe in depth the five steps composing the procedure carried out by the system.

A. Obtaining the Current Arrangement Map

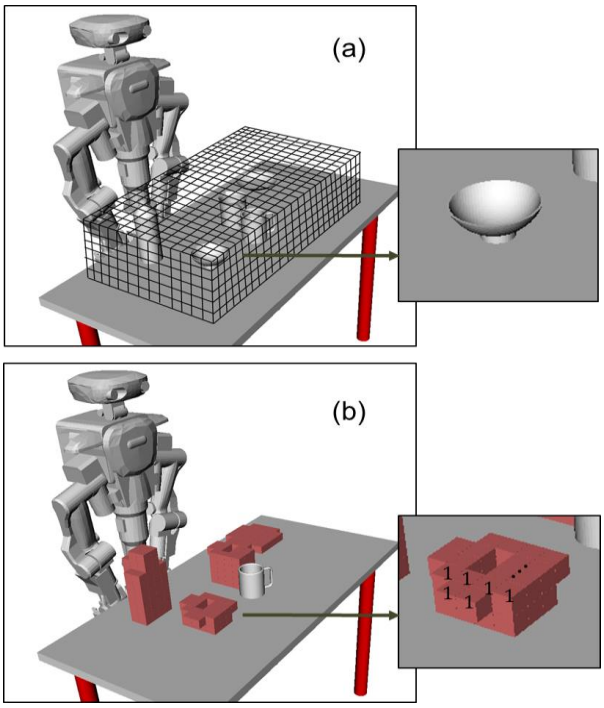


Fig. 3. Generating an object arrangement map. The space above the table top board is divided into unit cubes for defining a voxel map (a). Every voxel that overlap with any obstacles are filled up and is given the value of 1 (b).

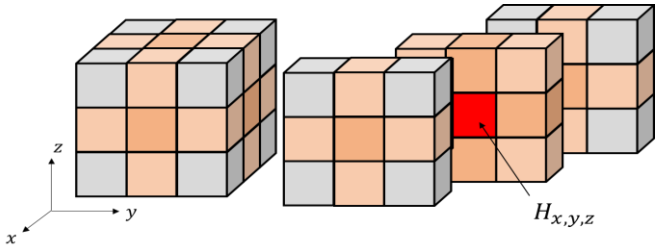


Fig. 4. A smoothing mask. The mask is composed of 3×3×3 voxels, and each voxel value is expressed as $\boldsymbol{H_{xyz}}$. The nearer voxel of mask is to the central voxel, the bigger value is substituted.

First, we obtain a voxel map of the current objects arrangement using the same method described in the previous section, from now we will call it "Current Arrangement Map". Then, the goal hand pose for grasping the current goal object is decided.

Next, the Current Arrangement Map has to be compared with each Objects arrangement map contained in the Accumulated Action Data and the degree of similarity between the two maps is calculated using their voxels' values. However, as a voxels' value is binary ('0' or '1'), it will result that there are several maps with the same degree of similarity, so we modify the values by masking the Current arrangement map with a smoothing function that gives each voxel a decimal value. This smoothing method is based on an image processing technique widely used [16]. Fig. 4 shows the smoothing mask. The mask is composed of 3×3×3 voxels, and one voxel size of mask is equivalent to the voxel size of maps. The value of each voxel in the mask is expressed as $H_{x,y,z}\ (-1 \le x \le 1, -1 \le y \le 1, -1 \le z \le 1)$, being its

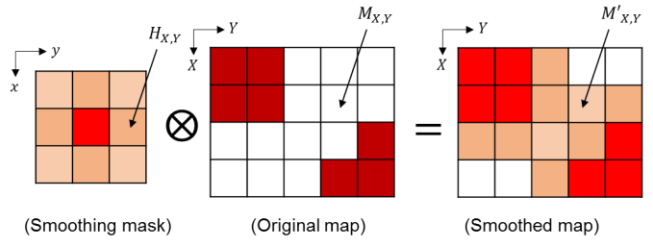


Fig. 5. The smoothing procedure. Dark red voxels in the original map are given the value of 1. The new map is obtained by smoothing the original map with the mask. The lighter the colors in the mask and the smoothed map are, the lower its values are given.

central voxel $H_{0,0,0}$. Each voxel's value is given by a Gaussian distribution, with its maximum value located in $H_{0,0,0}$ and decreasing in farther voxels.

The process of smoothing the Current arrangement map is shown in Fig. 5. For ease of understanding, Fig. 5 shows an example using a two-dimensional mask, instead the three-dimensional one we used. In the original map, each voxel's value is expressed as $M_{X,Y,Z}$, where X, Y, and Z are the values corresponding to the coordinates for its depth, width, and height inside the map, respectively. A new map whose shape is the same as the original map is prepared, and each voxel value in the new map is expressed as $M'_{X,Y,Z}$. The value for each $M'_{X,Y,Z}$ is given by (1).

$$M'_{X,Y,Z} = \sum_{x=-1}^{1} \sum_{y=-1}^{1} \sum_{z=-1}^{1} H_{x,y,z} M_{(X-x),(Y-y),(Z-z)} \qquad (1)$$

This new obtained map will be the Current Arrangement Map using for the planning process.

B. *Selecting a Similar Class*

Once we have the smoothed map we can select the class most similar to it. However, before explaining how to do it, we need to describe how classes are generated. The plane defined on the table top board is equally divided into tiles, and the center coordinates of each tile are calculated. Then, using the coordinates of the manipulator's goal pose contained in the Accumulated Action Data, the tile corresponding to those coordinates is obtained and the distance between the coordinates of the goal hand pose and the center of the tile is calculated. This processing is conducted for each joint angles' set "V" in the Accumulated Action Data, and its entry (containing the set of joint angles and the objects arrangement map) is grouped in classes depending of the tile of the goal hand.

In the process of selecting a similar class, the current goal hand coordinates is obtained from the current goal object, and the tile corresponding to those coordinates will give us the tile associated to the most similar class for the Current arrangement map.

C. *Extraction of Action Data*

We extract the Action Data of the class selected in the previous step. In this process, we focus not only on the coordinates but also on the hand's orientations. In order to reduce the processing time, we implemented a two steps extraction. Provided that we have N combinations of Objects arrangement maps and joint angles in the similar class, for the first step the nearest neighbor search FLANN method[17] is used, and k combinations with joint angles containing the goal hand coordinate similar to the current one are extracted. In the second step, we check values obtained from goal hand poses in k combinations. Then, we express the rotation matrix and the coordinates of one goal hand pose in k as $\boldsymbol{R}_D$ and $\boldsymbol{d}$. We obtain as well the rotation matrix and the coordinates from the current goal pose, expressed as $\boldsymbol{R}_G$ and $\boldsymbol{g}$. Then, we express the relative distance between these two poses as l, given by the next formula:

$$l = |\boldsymbol{d} - \boldsymbol{g}| \qquad (2)$$

Also, we are able to obtain a new matrix $\boldsymbol{R}$, whose elements are expressed as R_{ij}:

$$\boldsymbol{R} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} = \boldsymbol{R}_G^T \boldsymbol{R}_D \qquad (3)$$

Thresholds for distance and rotation (l_t and θ_t) are set in advance, and if the values obtained by the above process satisfy the following conditions, the combination is extracted.

$$\begin{cases} l < l_t \\ \cos^{-1}(R_{11}) < \theta_t \\ \cos^{-1}(R_{22}) < \theta_t \\ \cos^{-1}(R_{33}) < \theta_t \end{cases} \qquad (4)$$

Thus, the combinations extracted in the second step are put together and constructed. We call the resulting set "Reconstructed Action Data".

D. *Comparing Two Maps*

We compare the Current Arrangement Map with each map in the Reconstructed Action Data is conducted, and calculate the similarity degree between both. The similarity degree of a map is expressed as S and defined as (5).

$$\mathrm{S} = \sum_{X,Y,Z} M'_{X,Y,Z} D_{X,Y,Z} \qquad (5)$$

Being $M'_{X,Y,Z}$ the voxel value from the Current arrangement map, and $D_{X,Y,Z}$ the voxel value from the Reconstruction Action Data map, where X, Y and Z are the map coordinates of the voxel. In this equation, S increases if voxels of both maps in the same coordinates have positive values. As a result we obtain a set of similarity degrees, expressed as $\mathbf{S} = \{S_1, S_2, \cdots, S_n\}$, and the map of S_{max} which is the biggest value in $\mathbf{S}$ is extracted as the most similar map.

E. *Action execution*

Once we have selected the most similar map, we extract its related joint angles list. Then, the manipulator moves in accordance with those joint angles.

VI. Evaluation

A. Condition

We performed an evaluation experiment with a multiple-jointed arm HIRO robot from Kawada industries Inc. Specifications of the robot are shown in table 1. In the operations, the robot only used the six-jointed left arm and the joint in the waist axis. A table was used as operations environment and we placed there a mug, acting as the goal object of the experiment, and a pet bottle, a bowl, a beaker and a dish used as obstacles. All objects were placed randomly on the table. We used OpenRAVE[18] with BiRRT trajectory planning in order to select the movements for grasping the goal object. Figure 6 shows how the grasping is performed in an OpenRAVE simulation.

We designed a voxel space for the operation environment of 570mm wide, 1080mm long and 240mm tall. The length of the voxels were 30mm. As described in section V-C, the error threshold distance was 3mm and the threshold angle was 5º. We prepared two conditions for the experiment: condition (1), with the goal object's coordinates and orientation always constant and predefined, and condition (2), where only its coordinate in the Y axis was constant and predefined. We also defined a simple method to grasp the goal object for the robot. In case of condition (1), the first step extraction described in section V-C was skipped because the extracted combinations from the Action Data are not big enough to generate performance issues, so we don't need the reduction step (however the second step is always mandatory to extract the Reconstructed Action Data). In case of condition (2), as the orientation of the object may not be suitable for that grasping posture, the robot would adjust it if necessary. Also, for condition (1) we tested with increasing amounts of Action Data from 100 to 7000 samples, using increments of 100 up to reaching 1000 samples, and then using increments of 500, and for condition (2) we used amounts from 1000 to 100000 samples with increments of 1000. Finally, we performed 300 runs of the experiment.

TABLE 1
HIRO's specification

Name	HIRO
Development	KAWADA Industries Inc.
Flexibility	15 axes (Arm: 6 axes×2, Neck: 2 axes, Waist: 1 axes)
Joint structure	Arm: Sholder-Y, P Elbow-P Wrist-Y, P, R Neck: Neck-Y, P Body: Waist-Y
Used Joints	Left Arm + Waist
CPU	ATOM N270 1.6GHz

We use the model of HIRO in an Openrave simulator. The robot only used the six-jointed left arm and the joint in the waist axis.

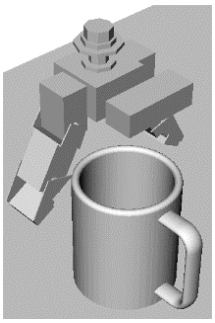
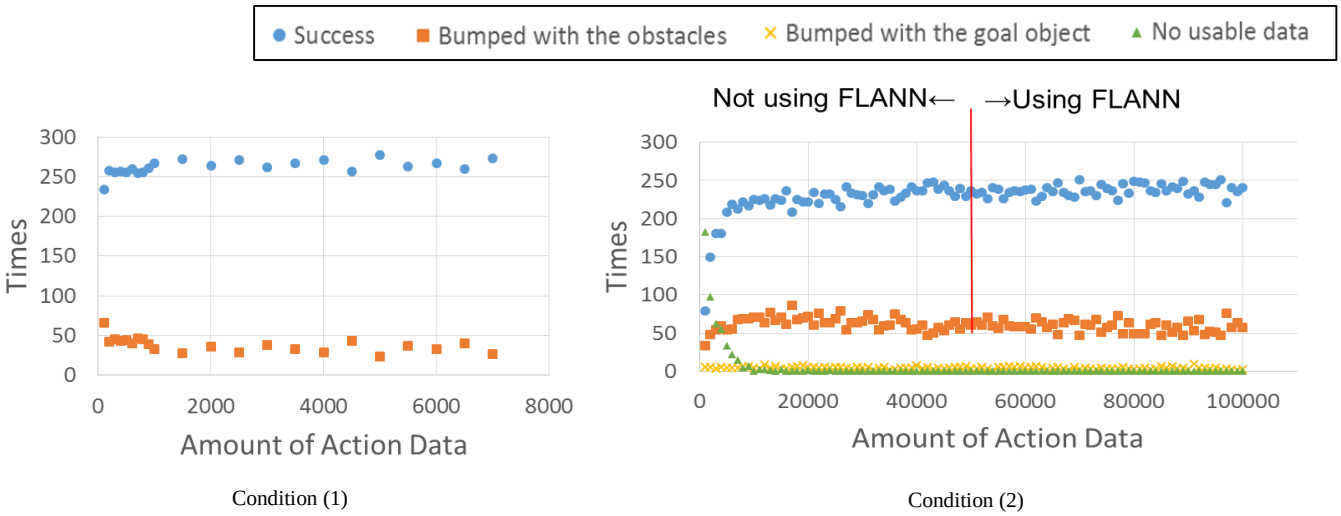


Fig. 6. The method of grasping. This is the goal hand pose in the experiment.

B. Results

Figure 7 shows the success rate of the robot operation for each condition in the 300 runs of the experiment. We can see that in the majority of cases the robot managed to take the goal object without touching any of the obstacles

For condition (1) training data was composed of 7000 samples, and we didn't found any result where the data was not usable or where the robot bumped with the goal object. Also, once the robot was trained with more than 100 samples, the success rate reached 78%, and reaching 92% at most.



Condition (1)

Condition (2)

Fig. 7. The success rate of the robot operation for two conditions. The blue plots shows success times in 300 trial. The red, yellow, and green plots shows fail times, and each cause is bumping with obstacles, bumping with the goal object, and no usable samples. There is no fail times of yellow and green in case of condition (1).

For condition (2) training data was composed of 100000 samples, and the first step extraction is skipped in cases when the amount of Action Data were less than 50000 due to the same reason for doing it in condition (1). The highest success rate we obtained was 83%. When the training data was sparse, we can observe cases where there is no usable samples but after reaching 25000 samples, those results decrease to almost 0. Also, the cases where the robotic arm bumped with the goal object were sparse but constant with any number of training samples. Finally, when using 50000 training samples, the time for selecting the most suitable joint angle from the database was 0.202 seconds on average, around 10 times faster than BiRRT.

C. Discussion

By seeing the results, in both (1) and (2) conditions the success rate was constant, but also constant (albeit low) was the failing rate due to bumping with obstacles. We think these cases appeared due to the most similar map still containing important differences with the current arrangement map, containing obstacles in the manipulator trajectory. As our similarity function does not give more importance to the space around the proposed trajectory of the manipulator, we are planning to improve it by adding such functionality. Additionally, the failing cases where the grasping method we decided (shown in fig. 6) was not feasible were due to that obstacles being too near to the goal hand coordinates. Also, condition (2) showed that in some cases the robot bumped also with the goal object when trying to grasp it. The cause of these misses is due to slight differences in the goal object's position and orientation, making the manipulator to narrowly failing to grasp the object.

Therefore, in order to increase the success rate and confront these issues, we are planning to add additional information related to the possible ways of grasping, and implement a method to select which one should be the best one given the current arrangement and its similarity with the selected Action Data. We think this simple solution can reduce greatly the failing cases we described previously.

VII. Conclusion

In this paper, we presented a new method for robot's trajectory planning that obtains a success rate of around 80% without performing collision's checking. Due to this feature, we are able to obtain a significant reduction in the computational load and calculation times, which is a problem in current methods. Also, this technique allows for cloud deployment, lightening more the systems that use it. Our technique uses a database of past samples in order to select the current actions for grasping a goal object. The search of the most suitable sample from the database is performed by comparing its samples with the current environment and selecting the most similar one. We conducted an evaluation for our method obtaining positive results in terms of success rate: just by having more than 100 samples of Accumulated Action Data it reached between 78% and 92% of success. It also performed ten times faster than trajectory planners like BiRRT. In some sparse results, issues with our method were detected and we devised a way for solving them in the future. For the next steps we plan to solve those problems applying the improvement mentioned in section VI-C and also to perform more complex evaluation experiments with more obstacles and in different environments in order to identify other possible issues.

References

[1] F. Schwarzer, M. Saha, and J. C. Latombe. "Adaptive dynamic collision checking for single and multiple articulated robots in complex environments." Robotics, IEEE Transactions on 21.3 (2005): 338-353.

[2] K. Hauser. "Lazy collision checking in asymptotically-optimal motion planning," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on* , vol., no., pp.2951-2957, 26-30 May 2015

[3] M. Kallmann, A. Aubel, T. Abaci and D. Thalmann. "Planning Collision‐Free Reaching Motions for Interactive Object Manipulation and Grasping". In Computer Graphics Forum (Vol. 22, No. 3, pp. 313-322). Blackwell Publishing, Inc. Sep. 2003

[4] J. Schulman, Y. Duan, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, P. Abbeel. "Motion planning with sequential convex optimization and convex collision checking". I. J. Robotic Res. 33(9): 1251-1270 (2014)

[5] S. M. LaValle. "Planning algorithms". Cambridge university press, 2006.

[6] A. Ude, C. G. Atkeson and M. Riley. "Planning of joint trajectories for humanoid robots using B-spline wavelets". Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on (Vol. 3, pp. 2223-2228). IEEE, 2000.

[7] D. Costantinescu and E. A. Croft. "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths". Journal of robotic systems, 17(5), 233-249. 2000.

[8] G. E. Hovland, P. Sikka, and B. J. McCarragher. "Skill acquisition from human demonstration using a hidden markov model". Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on (Vol. 3, pp. 2706-2711). IEEE, April, 1996.

[9] N. Melchior and R. Simmons. "Graph-based trajectory planning through programming by demonstration". In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on (pp. 1929-1936). IEEE. Oct. 2012.

[10] J. Luo and K. Hauser. "Robust Trajectory Optimization Under Frictional Contact with Iterative Learning". Robotics: Science and Systems, Rome, Italy, Jul. 2015.

[11] T. Sekiguchi, Y. Kobayashi, A. Shimizu and T. Kaneko, "Online learning of optimal robot behavior for object manipulation using mode switching", Proc. of IEEE Int. Symposium on Robotic and Sensors Environments, pp61-66, Magdeburg, Germany, Nov. 2012.

[12] H. Kato, T. Tsuruta, Y. Ishihara, M. Shimizu and M. Hashimoto. Development of Robot Motion Performance Platform for auto generation of Robot Motion Planning. In *SICE Annual Conference (SICE), 2012 Proceedings of* (pp. 1685-1690). IEEE, Aug. 2012.

[13] J-S. Gutmann, M. Fukuchi, and M. Fujita. "3D Perception and Environment Map Generation for Humanoid Robot Navigation". The International Journal of Robotics Research October 2008vol. 27 no. 10 1117-1134

[14] Robert Collins, Yanghai Tsin, J. Ryan Miller, and Alan Lipton, "Using a DEM to Determine Geospatial Object Trajectories", in DARPA Image Understanding, Workshop; 1998. p. 115-122

[15] J.J. Kuffner and S.M. LaValle, "RRT-connect: An efficient approach to single-query path planning", Proc. of IEEE Int'l Conf. on Robotics and Automation, pp995-1001, April 2000, San Francisco, CA

[16] D. Holz, S. Behnke, "Approximate triangulation and region growing for efficient segmentation and smoothing of range images", Robotics and Autonomous Systems62(9): 1282-1293 (2014)

[17] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithmic configuration", In Proc. VISAPP, 2009

[18] R. Diankov. "An Analysis of the ROS and OpenRAVE Open-Source Tools Towards Intelligent Manipulation". Journal of the Robotics Society of Japan, vol. 28, no. 5, pp585~588, Jun 2010.