Proceeding of the 2nd
RSI/ISM International Conference on Robotics and Mechatronics
October 15-17, 2014, Tehran, Iran

Regrasp Planning through Throwing and Catching

Sayed Javad Mousavi, Ellips Masehian, and Shokraneh K. Moghaddam

Abstract—**Multi-fingered hands are the most sophisticated and complex robotic grippers which can be planned to exhibit highly versatile grasping tasks. A relatively new regrasp planning method is by throwing and catching objects, which cannot be considered as a quasi-static problem due to its high dynamism influenced by the gravity, impact forces, air resistance, friction, etc., and thus has rarely been researched. In this paper, regrasp planning is performed through throwing and catching of various objects by a five-fingered anthropomorphic hand attached to a PUMA 560 manipulator. The planner uses an MLP neural network for learning from past throwing and catching experiences of objects with 9 different geometries. For the test set, five distinct objects with different geometries, densities, and center of mass were designed and tested. Experimental results of throwing and catching these objects showed that the minimum and maximum number of throws for successful catching was either 2 or 3, among which three objects needed just 2 tries (throws) thanks to the correct prediction of the implemented neural network.**

I. Introduction

Multi-fingered hands are the most sophisticated and complex robotic grippers which can be planned to exhibit highly versatile grasping tasks. Examples of such hands are the Utah/MIT dexterous hand [1], the Shadow Dexterous Hand™ [2], the lightweight high speed multi-fingered hand system built in Ishikawa Watanabe laboratory at the University of Tokyo [3], and the Barrett Hand™ [4] which have the ability to perform complicated tasks and therefore require complex planning techniques. One of the main challenges of these hands is to achieve an appropriate grasp configuration for an object through reorienting and reconfiguring it, which is also a major issue in many manipulation planning tasks such as assembly planning. Manipulation planning for a robotic hand can be inspired by the way humans accomplish different tasks using their own hands [5]. Human hands have three main responsibilities: exploration, holding, and manipulation, which can each be defined for robotic hands as well.

In planning for anthropomorphic multi-fingered hands, Grasp planning is the first issue that comes to mind. According to [6], multi-finger grasping can be categorized into two major classes of *fingertip* and *envelope* grasping. Fingertip grasping is dexterous manipulation of an object in which the robot must have the ability to exert and control arbitrary contact forces on the object by its fingertips. Control signals are sent to pneumatic, hydraulic or servomotor actuators in order to be executed through connections to the joints directly or by means of tendons. Envelope grasping is performed by wrapping the fingers around the object, with the support of the palm. Both of the above methods are quite similar to fixturing as the object must be immobilized for further manipulation.

Sometimes the robot is unable to place an object in its desired final configuration from its current grasped configuration because of the existence of obstacles between the object and its goal, or due to the kinematic constraints of the hand and the arm to which the hand is attached [7]. Therefore, the object must be grasped and *regrasped* a number of times to reach its final configuration. In fact, reorienting a grasped object is referred to as *Regrasping* in many research articles [8]. For example, [9] categorizes regrasping methods into two ways of *in-hand* regrasping, in which the object is always in contact with the hand during regrasping, and *pick-and-place* regrasping, in which the robot is allowed to place the object on a stable surface during regrasping and grasp it again. However, an inclusive categorization of main regrasping approaches is presented in [10], which identifies them as Finger Gaiting [11-14], Finger Sliding [15, 16], Finger Rolling [17], Pick-and-Place [18], Bimanual regrasping [19], Human Imitation [20], and Throwing and Catching [21]. While in the first five methods the object is in constant contact with the hand during the regrasping operation (i.e., in-hand regrasping), the latter two methods do not require such a limitation. Also, force-closure or form-closure conditions must be maintained while in-hand regrasping is performed, but regrasping through pick-and-place or throwing and catching do not require such conditions.

It is noteworthy that there is only one work in the literature based on throwing and catching method ([21]), in which the projectile's next position and orientation are estimated based on its dynamic equations as well as its previous position and orientation. Here we present a method in which estimating the position and orientation is based on objects' features and how it is thrown. This method does not require calculating the dynamic equations and all its corresponding processes can be performed offline.

Regrasping through throwing and catching is quite faster than other regrasping methods and using this method, robot does not need to place the object on a stable surface during regrasping procedure. Among all existing methods for regrasping, throwing and catching is particularly involved with objects' dynamics. Using a hand and an arm with high DOFs for regrasping require working with a very complex and complicated dynamic system. Handling such dynamic systems demands the design of a control system for arm and hand with the ability of performing many complex on-line calculations with a super-computer; implementing such system can be quite costly.

978-1-4799-6743-8/14/$31.00 ©2014 IEEE

For facing these challenges, based on a problem's configuration space, Artificial Intelligence can be implemented; if all the object's, arm's, and the environment's features could be distinguished and determined as the inputs of an artificial intelligent system and parameters of position, velocity and catching be the outputs of this AI system, many different objects can be thrown and caught using this system.

Using the above perfect AI system is impossible due to the fact that all the mentioned features can't be extracted at the same time, also designing such system would be very complicated and time consuming; therefore by abstracting the problem, the number of inputs and outputs of our AI system would be reduced and the problem can be implemented and solved. In this research abstracting the problem is used by omitting the robot and not considering the environment through similar throwing i.e., the same initial conditions are created for the robot every time it throws an object. Using the recreated throwing, we are able to correctly determine the catch position and velocity of the object in this position in offline mode. This abstraction causes a significant reduction of input and output parameters.

II. Problem Definition

There exists a simple physics rule for similar throwing and catching, that is, by recreating the initial conditions of a problem, the same results would be achieved; therefore, recreating the same initial conditions for a projectile which is about to be thrown, results the same ballistic motions. Despite the wide variety of objects shapes which inevitably increases total costs and also the unavailability of projectile's initial velocity vector, axis of rotation, angular velocity and other initial coordination, using the above fact enable us to recreate the same ballistic motions of an object by setting initial grasp configuration, joints' angles or joints' angular velocities (parts of robot configuration space within differential constraints [22]).

Here, we present a two phase method for tackling the complicated problem of regrasping. We focus on learning phase which in this phase there are sequentially *online* and *offline modes*, in the first online mode robot gathers information about throwing and catching parameters of an object and then calculates the grasp parameters in offline mode which are used in next online mode, which is like trial and error. In this phase an object is allowed to be thrown multiple times by recreating the same initial conditions. While doing so, position and orientation of the object are calculated and object's motion equation can further be estimated. In the real world, by implementing a simple offline image processing system, object's motion equation can be estimated using two different images of an object being thrown, however, available simulation software enable us to calculate the position and orientation of any object at any time. In the second phase i.e., *applying phase* robot, based on the learned throwing and catching parameters in the learning phase, is able to regrasp the exact same object with the exact same regrasp parameters attained in learning phase while performing a real task.

This paper mainly focuses on the learning phase hence the problem we address can be defined as follows: an object is grasped by a robotic hand using envelope grasping and its grasp configuration must change while it is not possible to place the object on a stable surface during performing the task. It is also impossible for the robot to carry out in-hand manipulation; therefore, regrasping is done through throwing and catching. Through similar throwing and catching(s), robot is able to learn which configuration is desirable to regrasp the object by, and while performing the real tasks, robot is able to catch the object properly before it hits the ground. In fact, in learning phase all the trials are performed assuming the same initial conditions for the robot and the only changing factor is properties of the objects. We have simulated robot and its throwing and catching motions with the help of Webots™ simulator software that entails the powerful physics module, ODE. Unlike previous researches, our robotic hand also has an arm which is a PUMA560 that is attached to an anthropomorphic five-fingered hand. Existence of the arm makes planning more complicated however more positions of the workspace would be accessible.

III. Overview of the Method

The planning strategy which is used for throwing and catching in this paper, is more flexible towards object transformation, object density and position of object's center of mass; hence for each object, a *Parameter Space* (P-space) or a regrasping configuration space can be created and afterwards searched in order to achieve the final solution on how it must be thrown and caught. Searching is performed on the learning phase that contains two major stages, Online and Offline and executed consecutively.

During online stage, robot throws an object and the object's positions and orientations are gathered, then in the offline phase its catch positions and velocity along vertical axis (v_{catch}) on these catch positions are calculated. After that, robot, based on previous experiments (these experiments are carried out by neural network) decides how to catch the object and determines some parameters for online stage. During next online stage, robot throws the object and tries to catch it afterwards. If the catch is successful the process is complete otherwise robot refine the parameters by searching the parameter space (space created by catching parameters) and repeat throwing and catching until a solution is found. The main flowchart of learning phase process is shown in Fig. 1.

A. Assumptions

To implement throw and catch method in this paper, the following assumptions are made about the environment, objects, and robot which can all be created, implemented, deleted and edited using Webots™. Our robotic hand performs its tasks in an environment where real amount of gravity exists but wind effects, air resistance and objects' elasticity are not considered. Objects are rigid and, as well as different parts of the robot, have weight and density. Real-world dynamics applies to all objects and the robot, including velocity, acceleration, inertia, and momentum. Also, friction exists on the surfaces of objects according to Coulomb's friction laws. An object can either have nonuniform density or uniform density. In the

latter case, object's center of mass overlaps with its geometrical center.

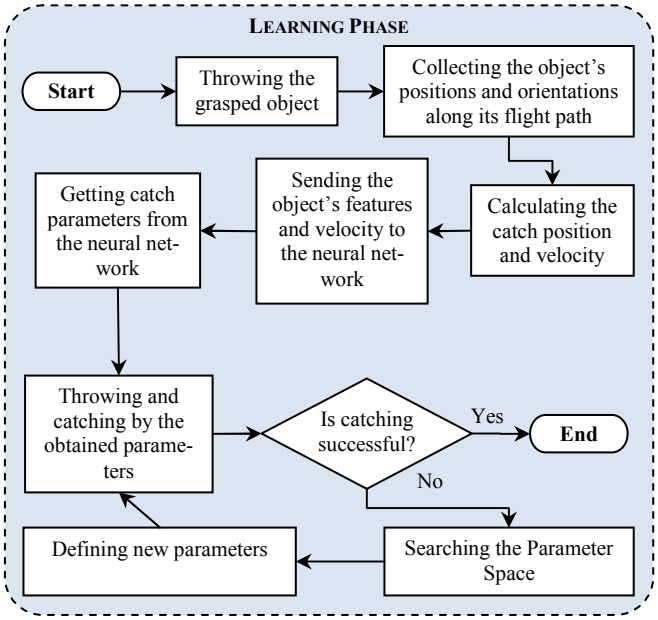


Fig. 1. Main flowchart of the proposed method.

To simplify the envelope grasping of an object, it is assumed that each object consists of at least a cylindrically symmetric piece and the position of object's center of mass is specified by a coordinate system attached to the geometrical center of this specific piece (Like Fig. 2). *P*-controllers are used for servos and servomotors power is within real servos power range (for example 150 N·m for the servomotor located in shoulder along vertical axis, 186.4 N·m for the servomotor in shoulder along horizontal axis, 89.4 N·m for the servomotor located in elbow, 24.2 N·m, 20.1 N·m and 21.3 N·m for all the three wrists' servos and 10 N·m for all the fingers' servos). Errors caused by neural network or sensors output are covered by envelope grasping tolerance (catching tolerance is higher in comparison to the mentioned errors).

B. Coordinate Frames

As mentioned earlier, a coordinate system is attached to all the objects. For a cylindrically symmetric object as shown in Fig.2 (a), the coordinate system's origin coincides with the object's geometrical center hence *y* axis would be along axis of symmetry, however for a cylindrically asymmetric object as shown in Fig.2 (b) origin of the coordinate system coincides with that part of the object which is cylindrically symmetric and also *y* axis would be along this part's axis of symmetry. In the latter case, object's center of mass is defined by two parameters, h_{cm} and r_{cm} that equals *y* coordinate and *x* coordinate of the object's center of mass respectively. It must be noted that a cylindrically asymmetric object's center of mass is determined in such way that the *z* component of its corresponding coordinate system equals zero.

It is also important mentioning that catching and grasping are two different terms that sometimes are used interchangeably. The term Catching is used when the robot grabs the object from the air and is not on its force, form, or frictional form closure but the term grasping is used after catching is finalized and force, form, or frictional form closure is maintained.

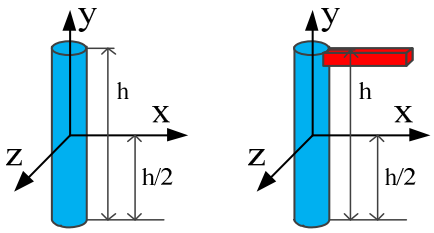


Fig.2. Coordinate systems attached to (a) a cylindrically symmetric, and (b) a cylindrically asymmetric objet.

IV. Implementation

Because of the low angular velocity of finger joints, when the object reaches a certain height above palm of the hand (h_{catch}), fingers must start retracting, otherwise due to its velocity, object might be tossed out of the robotic hand. Furthermore to reduce the effect of object's stroke and coordinate robotic hand movements with object's motion, the hand moves to a certain height lower than the catch point (h_{down}). After the object is thrown, hand goes to catch position then waits for the object's return and as soon as the object reaches h_{catch} above the hand, it moves to h_{down} while fingers slowly being retracted. When finally the object hits the hand, both robot and the object moves up concurrently until hand reaches catch point at which the grasping is finalized. h_{down} and h_{catch} are the output parameters of our proposed *neural network*.

V. Neural Network

The neural network which is used in this paper is a two layer MLP with 13 neurons in the hidden layer, 2 neurons in the output layer and has 5 inputs and 2 outputs (h_{catch} and h_{dwon} are the outputs) as shown in Fig.3.

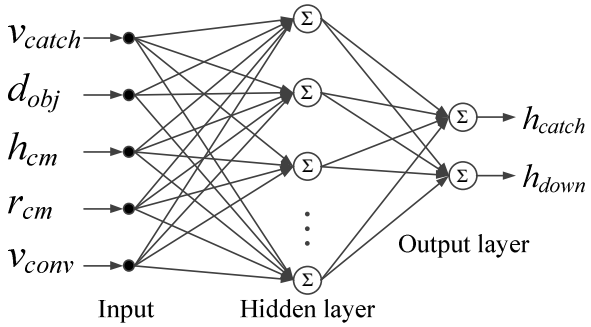


Fig.3. Presented neural network

To initially train the artificial neural network, 9 geometrically different objects are created (Fig. 4) that by changing their density and gravity centers 48 different objects would be at hand. Grasping parameters of these 48 objects are adjusted manually and since each object has many different grasping points 64 data is created for neural network learning that are shown in Fig. 4. Out of these 64 data, randomly, 44 are chosen for training, 10 are chosen for validation and

10 are chosen for testing the artificial neural network.

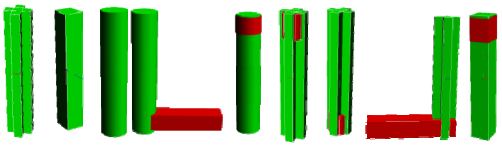


Fig. 4. Objects' shape

The input parameters of our artificial neural network are velocity of the object while being caught by the robot (v_{catch}), h_{cm}, r_{cm}, the density of the object (d_{obj}) and the number of convex vertices of the object (v_{conv}) (for example the first left object shown in the Fig. 4 has 8 convex vertices).

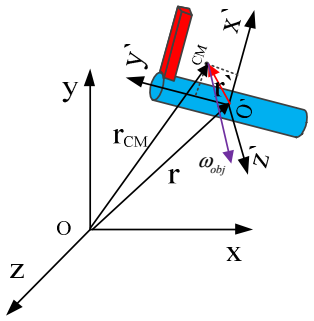


Fig.5. Position of object's gravity center and the universal coordinate system O.

v_{catch} is calculated for each appropriate catch point. Since the palm of the hand is horizontal, a proper catch point for the robot is where the object is horizontal as well; otherwise, if the object's angle with the hand be steep, the probability of catching the object will be reduced. To make sure that the hand is parallel with the object at the moment of catching, the rotation matrix of the coordinated system attached to it around O must be calculated as follows.

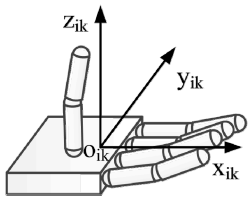


Fig. 6. The coordinate system attached to the hand.

If the catch point of an object in O be shown by r (Fig.5), unit vector of $\hat{r}_y$ shows the orientation of y' in O and the frames attached to the hand be as shown in Fig. 6, unit vectors along the coordinate system attached to the hand are calculated as follows:

$$\begin{aligned} r &= (x_r, y_r, z_r) \\ \hat{r}_{y'} &= (x_{\hat{r}_{y'}}, y_{\hat{r}_{y'}}, z_{\hat{r}_{y'}}) \end{aligned} \tag{1}$$

r must be rotated by a small degree around y axis in O in order to have a point which is always located at the right side with respect to y so we have:

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2}$$

Catch point's vector projection and its rotated projection on zx plane are calculated as follows:

$$\begin{aligned} r_p &= (x_r, 0, z_r) \\ r_{pr} &= r_i R_y(\theta) \end{aligned} \tag{3}$$

θ is equal to 0.3 radians and the vector which is pointing to the right side of the object with respect to y axis is calculated as follows:

$$r_R = r_{pr} - r_p \tag{4}$$

The unit vector along x_k is: ($\times$ is the cross product of two vectors)

$$\hat{r}_{z_k} = (\hat{r}_{y'} \times (\hat{r}_{y'} \times r_R)) / |\hat{r}_{y'}|\,|\hat{r}_{y'}|\,|r_R| \tag{5}$$

As a result z_k is always perpendicular to the axis of symmetry of the object's cylindrically symmetric part and points to the left side of y axis. Since palm of the hand is always pointing up, the other two axes are calculated as follows:

$$\begin{aligned} \hat{r}_{x_k} &= (0,1,0) \\ \hat{r}_{y_k} &= (\hat{r}_{x_k} \times \hat{r}_{y_k}) / |\hat{r}_{x_k}|\,|\hat{r}_{y_k}| \end{aligned} \tag{6}$$

Using the unit vectors we can calculate the rotation matrix of hand' coordination to its coordination in O and calculate the inverse cinematic of wrist as well.

$$R'_w = \begin{bmatrix} \hat{r}'_{x_k} & \hat{r}'_{y_k} & \hat{r}'_{z_k} \end{bmatrix} \tag{7}$$

A. The Parameter Space

If by using the neural network outputs, robot can't catch the object properly, planner starts searching a space called *parameter space* which is created by the two catching parameters, h_{catch} and h_{down} to determine new values for them. h_{down} and h_{catch} can each change in ranges of [−0.14, −0.01] and [0, 0.40] meters and since envelope grasping is used their tolerances are 0.01 and 0.04 respectively. Considering the tolerance amounts h_{down} creates 15 different states and h_{catch} creates 10 different states that result in 150 different search states combined. The exhaustive search method that is used for searching the parameter space in this paper is particularly similar to *wave front* method in which the search starts from a start point and then sequential layers of neighbors are searched completely for finding the final answer (Fig. 7).

4	4	4	4	4	5	6
3	3	3	3	4	5	6
2	2	2	3	4	5	6
2	1	2	3	4	5	6
2	2	2	3	4	5	6
3	3	3	3	4	5	6
4	4	4	4	4	5	6

Fig. 7. Searching the parameter space

The search goes on until the answer is found otherwise, process would be stopped. In our presented algorithm, start

point (the yellow square which is labeled 1) of the searching process for each object is in a different place of the search space and is, in fact, the value attained by the neural network output. Pseudo code of the search algorithm is shown in Fig. 8 and Fig. 9.

```
Algorithm Search(P_start,Grasp_position)
1     Grasp_Flag ← 0
2     Calculate h_catch and h_down
3     Grasp(h_catch, h_down, Grasp_position)
4     If grasping is successful
5        Grasp_Flag ← 1
6     Else
7        Set A_nbr as the set of all the neighbors created by sequential
         ly adding and/or subtracting 1 unit to x and y coordinates of
      P_start
8     End
9     While Grasp_Flag = 0
11       For all the elements in A_nbr
12          Grasp(h_catch, h_down, Grasp_position)
13          If  grasping is successful
14             Grasp_Flag ← 1
15          End
14       End
15    End
```

Fig. 8. Pseudocode of the search algorithm.

```
Algorithm Grasp(h_catch, h_down, Grasp_position)
1     set joints_angle by Grasp_position
2     set Wrist_position by joint_angle
3     While ||Wrist_position- Grasp_position|| > 0.001m
4        set Wrist_position by joint_angle
5     End
6     set Infr as Infrared(4:13) output
7     While Infr > h_catch
8        set Infr as Infrared(4:13) output
9     End
10    set finger_joints_angle for grasp
11    set Grasp_position(y) as (Grasp_position(y)+ h_down)
12    While ||Wrist_position- Grasp_position|| > 0.001m
13       If (Infrared(1:3)<600)
14          breack;
15       End
16       set Wrist_position by joint_angle
17    End
18    set joints_angle by Grasp_position
```

Fig. 9. Pseudocode of the search algorithm.

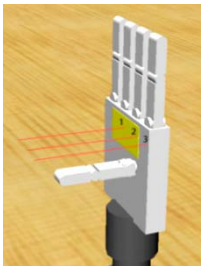


Fig. 10. The yellow rectangle on the palm shows the tactile bumper sensor and the red lines are infrared sensor range.

B. Grasp Recognition

Infrared and tactile bumper sensors are used for grasp recognition. Fig. 10 shows the emplacement of each sensor on the hand. The tactile bumper sensor has the dimensions of 3×6×0.1 cm and infrared sensors have the ability to sense the objects that are within their 10 cm and are divided into 1000 parts as shown in Fig. 11. If all the infrared sensors record less than a certain value simultaneously for a long period of time or the tactile bumper sensor output be equal to one for more than a certain amount of time, the object is recognized as grasped i.e., robotic hand is grasping the object by envelope method. Infrared and tactile bumper sensors are both prone to mistakes due to their certain amount of tolerances; hence both of them are used to reduce the probability of false recognition. Output of three IR sensors and one tactile sensor are shown in Fig. 11.

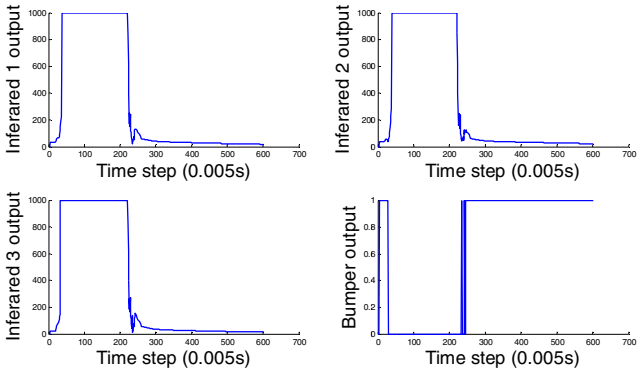
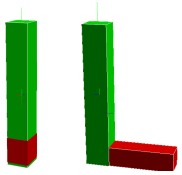


Fig. 11. Tactile bumper and infrared sensors output

VI. Results

After training the neural network, using the existing data, catch parameters of a totally new object are predictable. To test the accuracy of the neural network‘s outputs, two object's shape (shown in Fig.12) with different densities, and centers of mass were designed, which generated 5 distinct objects.



(1) (2)

Fig.12. Newly created objects for evaluating the algorithm

Table 1. Results of Throwing and Catching on Newly Created Objects

	Object			Results	
Row	ID	Density	Position of center of mass	Time (s)	Number of trials
1	(1)	150	(0.000, −0.010, 0.000)	68	3
2	(1)	200	(0.000, −0.010, 0.000)	44	2
3	(1)	200	(0.000, −0.020, 0.000)	68	3
4	(2)	300	(0.000, −0.055, 0.043)	42	2
5	(2)	350	(0.000, −0.055, 0.043)	42	2

Experimental results of throwing and catching these objects are shown in Table 1 indicating that the minimum and maximum number of throws for a successful catch was either 2 or 3. In cases 2, 4 and 5, robot was able to catch the object only by 2 trials. The first trial is in fact the one happening in first online stage of the learning phase where robot throws the object and instead of catching it, gathers information about the proper grasping positions of the object. During the second trial robot is able to successfully catch the object in second online stage using neural network outputs.

The above fact indicates the network ability to predict correct catch point parameters within envelope grasping tolerance. Throwing and catching steps of object (1) in the first row of Table 1, during the second trial are shown in Fig.13. In cases 1 and 3 robot caught the object by 3 trails which contained 3 online phases. During the third online phase the catch parameters were all achieved by searching the parameter space. Also, according to these cases (both involving the same object), robot was unable to catch the object by throwing it less than 3 times.

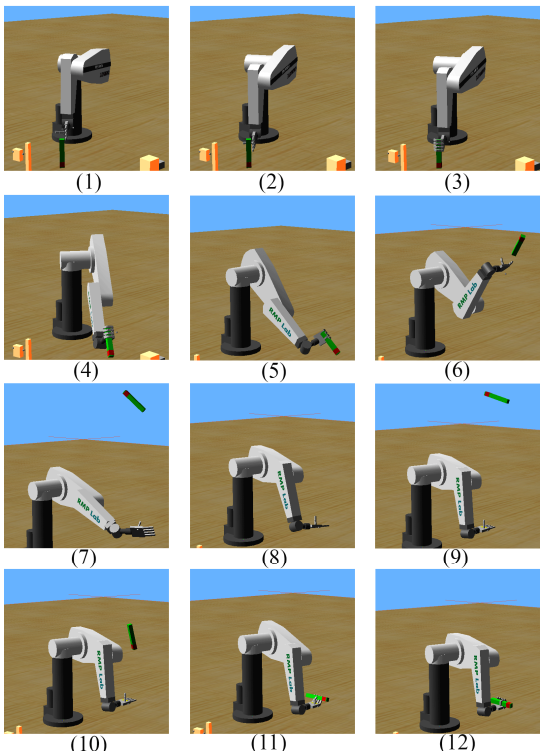


Fig.13. Steps of Throwing and catching an object

VII. Conclusion

Regrasping through throwing and catching can be a very useful method for reorienting and reconfiguring objects in the future. During performing this method no stable surface would be required for an object to be placed on and hence the object's reorientation can be performed faster. Also robotic hand performance while handling complex situations of regrasping and placing different items would be very similar to the way human beings handle them.

For the future work, by designing two different neural networks, it is possible to omit the stage where robot throws an object without catching it. The first network can be designed and trained using the parameters (i.e., velocity of the object while being caught by the hand and the catch point as network output) that affect grasping position of an object and based on the information acquired by this network the second neural network can be created with the capability of accurately predicting grasping parameters. For better neural network' training, it is also possible to add more variety to the geometry of the designed objects.

References

[1] Jacobsen, S. C., et al. "Design of the Utah/MIT dexterous hand," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1986, pp.1520-1532.

[2] Shadow Robot Company, "Shadow Dexterous Hand", [Online]. Available: http://www.shadowrobot.com/products/dexterous-hand.htm.

[3] A. Namiki, Y. Imai, M. Ishikawa, and M. Kaneko, "Development of a high-speed multifingered hand system and its application to catching," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 2003, pp. 2666-2671.

[4] Barrett Technology, "BarrettHand," [Online]. Available: http://www.barrett.com/robot/products-hand.htm.

[5] R. L. Klatzky and S. Lederman, "Intelligent Exploration by the Human Hand," in *Dextrous Robot Hands*, 1990, pp. 66-81.

[6] A. Namiki, Y. Imai, M. Ishikawa, and M. Kaneko, "Development of a high-speed multifingered hand system and its application to catching," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 2003, pp. 2666-2671.

[7] F. Rohrdanz and F. M. Wahl, "Generating and Evaluating Regrasp Operations," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 1997, pp. 2013-2018.

[8] T. Omata and K. Nagata, "Planning reorientation of an object with a multifingered hand," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1994, pp. 3104-3110.

[9] A. Sudsang and T. Phoka, "Regrasp planning for a 4-fingered hand manipulating a polygon," in *Proc. IEEE Int. Conf. the Robotics and Automation*, 2003, pp. 2671-2676.

[10] S. J. Mousavi and E. Masehian, "A New Taxonomy of Robotic Regrasp Planning Approaches and Methods," in *Proc. IEEE Int. Conf. on Robotics and Mechatronics*, 2013, pp. 90 – 95.

[11] M. A. Farooqi, T. Tanaka, Y. Ikezawa, T. Omata, and K. Nagata, "Sensor based control for the execution of regrasping primitives on a multifingered robot hand," in *IEEE Int. Conf. on Robotics and Automation*, 1999, pp. 3217-3223.

[12] T. Omata and M. A. Farooqi, "Reorientation planning for a multifingered hand based on orientation states network using regrasp primitives," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 1997, pp. 285-290.

[13] T. Schlegl and M. Buss, "Hybrid closed-loop control of robotic hand regrasping," in *IEEE Int. Conf. on Robotics and Automation*, 1998, pp. 3026-3031.

[14] T. Schlegl and M. Buss, "Dextrous hand regrasping using hybrid system models," in *IEEE/ASME Int. Conf. on Advanced Intelligent and Mechatronics*, 1997.

[15] A. A. Cole, P. Hsu, and S. S. Sastry, "Dynamic control of sliding by robot hands for regrasping," in *Proc. IEEE Trans. On Robotics and Automation*, vol. 8, no. 1, pp. 42-52, 1992.

[16] A. A. Cole, P. Hsu, and S. S. Sastry, "Dynamic regrasping by coordinated control of sliding for a multifingered hand," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1989, pp. 781-786.

[17] L. Han and J. C. Trinkle, "Dextrous manipulation by rolling and finger gaiting," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1998, pp. 730-735.

[18] P. Tournassoud, T. Lozano-Perez, and E. Mazer, "Regrasping," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1987, pp. 1924-1928.

[19] E. Yoshida, M. Poirier, J.-P. Laumond, O. Kanoun, F. Lamiraux, R. Alami, and K. Yokoi, "Regrasp planning for pivoting manipulation by a humanoid robot," in *Proc. IEEE Int. Conf. on the Robotics and Automation*, 2009, pp. 2467-2472.

[20] P. Vinayavekhin, S. Kudohf, and K. Ikeuchi, "Towards an automatic robot regrasping movement based on human demonstration using tangle topology," in *Proc. IEEE Int. Conf. on the Robotics and Automation* (*ICRA*), 2011, pp. 3332-3339.

[21] N. Furukawa, A. Namiki, S. Taku, and M. Ishikawa, "Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system," in *IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 181 – 187.

[22] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006, Chapter 4 "The Configuration Space", pp. 127-184.