In IEEE Symposium Series on Computational Intelligence 2014. IEEE.

Grasping Novel Objects with a Dexterous Robotic Hand through Neuroevolution

Pei-Chi Huang*, Joel Lehman*, Aloysius K. Mok*, Risto Miikkulainen*, Luis Sentis†

*Department of Computer Science, The University of Texas at Austin
{*peggy, joel, mok, risto*}*@cs.utexas.edu*
†Department of Mechanical Engineering, The University of Texas at Austin
lsentis@austin.utexas.edu

***Abstract*— Robotic grasping of a target object without advance knowledge of its three-dimensional model is a challenging problem. Many studies indicate that robot learning from demonstration (LfD) is a promising way to improve grasping performance, but complete automation of the grasping task in unforeseen circumstances remains difficult. As an alternative to LfD, this paper leverages limited human supervision to achieve robotic grasping of unknown objects in unforeseen circumstances. The technical question is what form of human supervision best minimizes the effort of the human supervisor. The approach here applies a human-supplied bounding box to focus the robot's visual processing on the target object, thereby lessening the dimensionality of the robot's computer vision processing. After the human supervisor defines the bounding box through the man-machine interface, the rest of the grasping task is automated through a vision-based feature-extraction approach where the dexterous hand learns to grasp objects without relying on pre-computed object models through the NEAT neuroevolution algorithm. Given only low-level sensing data from a commercial depth sensor Kinect, our approach evolves neural networks to identify appropriate hand positions and orientations for grasping novel objects. Further, the machine learning results from simulation have been validated by transferring the training results to a physical robot called *Dreamer* made by the Meka Robotics company. The results demonstrate that grasping novel objects through exploiting neuroevolution from simulation to reality is possible.**

I. INTRODUCTION

The capability to autonomously grasp unknown objects can greatly aid robots in performing a wide range of tasks. However, a robot in an unstructured environment may encounter objects of which it has only limited *apriori* experience or knowledge. In such cases, successful grasping requires sophisticated perception, planning, and control. Yet because each of these problems are difficult, fully autonomous grasping of unforeseen objects in unstructured environments remains an unsolved problem.

Thus instead of assuming full autonomy of robots, this paper investigates the amount and type of human supervision required to enable a robot to grasp unforeseen objects in an unstructured environment. Our approach has two stages. In the first stage, the human supervisor defines a bounding box around the target object by means of the man-machine interface. In the second stage, the robot's vision system determines the robotic hand configuration, orientation, and location that possible accomplishes the grasping task, based solely on a three-dimensional (3D) vision-based object features as extracted by the robot's sensors. This collaborative method provides an incremental approach to robotic grasping inasmuch as it decomposes the difficult task into distinct stages, each of which may be automated independently. In this paper, we focus on the automation of the second stage, that of determining an effective grasp posture. The automation of the first stage will be reported in a future paper.

To automate the second stage, we apply a neuroevolution approach [1], i.e. evolving an artificial neural network (ANN) with an evolutionary algorithm (EA); in particular, we apply the widely-used NeuroEvolution of Augmenting Topologies (NEAT; [2]) algorithm which is most powerful in tasks where the optimal behavior to be learned is not known, but needs to be discovered by exploration. NEAT evolves robotic grasp controllers that are evaluated in simulated grasping scenarios to gauge their fitness. The controllers receive as input only low-level real-time vision features (i.e. depth information from a simulation Kinect sensor), and guide the robotic hand by specifying the desired grasp orientation and location. Supporting this approach, previous work [3], [4] has demonstrated the promise of the evolutionary approaches to enable dynamic behavior generation in autonomous robots. Additionally, to speed up the training process, our implementation parallelizes the learning simulation by multi-threading the NEAT algorithm. Importantly, the results of training in simulation indicate that neural networks can be evolved to help successfully grasp objects.

Beyond simulated results, learned policies are also transferred to reality. In particular, they are transferred to the *Dreamer* humanoid robot in the Human Centered Robotics Laboratory [5] at the University of Texas at Austin (UTA) from Meka Robotics Corporation. Low-level input is provided by Microsoft Kinect sensors attached to *Dreamer* that capture a depth image of the target objects and their surrounding environment. Through transferring policies learned in simulation to the physical robot, the *Dreamer*'s hand, called *Mekahand*, can pick up simple-shaped objects by using evolved neural networks, given only low-level depth information.

The remainder of this paper is organized as follows. Section II describes related work. The approach is detailed in Section III. Section IV introduces the learning process. In section V, the experiment, its implementation, and the results are presented. Finally, Section VI and Section VII conclude the paper with remaining problems and future work.

II. RELATED WORK

Over the last decade, multiple robotic manipulation systems have been developed that apply motion planning approaches to generate stable robotic grasps. Such approaches generally adopt control system models specifically calibrated for a particular robot in a specific environment [6]. Consequently they often prove fragile when deployed in unforeseen environments. Importantly, machine learning methods hold the promise to overcome such limitations. For example, Miller et al. [7] applied heuristic rules to produce and evaluate grasps, and Pelossof et al. [8] employed support vector machines to evaluate grasp quality. However, in both methods full two-dimensional (2D) or three-dimensional (3D) models of the target objects are given *apriori* to the algorithms, so these approaches focused only on control and planning. Methods of extracting visual information are also prominent approaches: Piater [9] and Coelho et al. [10] used K-means clustering to measure 2D hand orientation, and Kamon et al. [11] controlled an arm to approach and then grasp through Q-learning. Recently, impressive progress has been made in learning to grasp novel objects [12]–[15]. The fundamental principle is to track 2D and 3D information through computer vision (e.g. objects' shapes and segmentation). Tracking this information can be then generalized to grasp novel objects. To the best of our knowledge, previous methods use only simple hand models, such as the Barrett hand, which has only 6 Degrees of Freedom (DOFs) and lacks the dexterity and flexibility of more complex humanoid hands. Such methods may not be well suited for the target hand in this work (i.e. *Dreamer* robot's hand, *Mekahand*) which has 12 DOFs. Also, the migration from simulation to reality is challenging [16], [17], thus because many studies only generate simulated results [18], it is possible that they might not function robustly on a physical robot.

Related to the approach described here are previous ANNs approaches that simulate arm kinematics; Rezzoug and Gorge [18] proposed two separate ANNs that respectively learn finger inverse kinematics and appropriate arm configuration, with results obtained only in simulation; Pedro et al. [19] proposed that contact points can be calculated through computational geometry (e.g. Delaunay triangulation and Voronoi diagram) before a robot performs grasping action but their work only considered objects geometry problems. Other approaches use reinforcement learning techniques: Zhang and Bössler [20] defined a reward function according to simple geometrical features of objects to learn a grasp controller for a PUMA robot. Saxena et al. [12] used supervised learning to learn correct 3D grasping points for a gripper from visual features of objects. However, most methods assume that the current state of the system is completely known. If objects are occluded or the situation varies dynamically, it is difficult for such methods to differentiate between possible situations.

III. APPROACH

Our approach takes inspiration from Kohl et al. [21] who proposed that neuroevolution can develop effective automobile warning systems using only low-level sensor input (i.e. pixels) from a digital camera. Taking this hint, a similar vision-based feature-extraction approach is used here, where through neuroevolution the *Mekahand* learns appropriate hand positions and orientations for grasping by interacting with objects in the GraspIt! simulation environment, which is described next.

A. GraspIt! Simulation Implementation

To apply neuroevolution to learn where and how to grasp an object requires both training scenarios and a metric for evaluating performance. GraspIt! [22], [23] facilitates simulating the *Mekahand* robot in representative grasping tasks and aids in measuring the quality of resulting grips. GraspIt! has limited support for the *Mekahand* model, so to implement this architecture requires extending the simulator. In GraspIt!, the *Mekahand* is defined by one DOF for each knuckle in each finger, with an additional DOF for the thumb's rotator. The mechanics of this model are modified here to augment two aspects of the simulation. First, controlling the wrist is not modeled by default, but is an important DOF. Therefore, a wrist component was added to the *Mekahand* model supplied by GraspIt!. Second, most of the DOFs in the real *Mekahand* are not actuated, although they are modeled as actuated in the GraspIt! simulation. Each finger of the real *Mekahand* consists of three joints which are all connected by a single rubber tendon. Thus when the finger curls, all three knuckles curl in unison. Therefore, the torques in GraspIt! were adjusted such that the set of torques given to a single finger are equivalent to the torques initiated by stretching the rubber tendon in the real robot.

B. Grasp Quality Measure

An evolutionary search requires a fitness function that measures the quality of candidate solutions. Because robust grasping behaviors are desired in this experiment, an important consideration is how to measure the quality of grasps.

Much previous grasping quality research focuses only on contact types and positions, ignoring hand geometry and kinematics. Other measures assume simple of grippers. In contrast, Miller and Allen [24] proposed a more sophisticated approach, which is used here. Given a 3D object and posture of the hand, their measure can accurately identify the types of contacts points between the links of the hand and the object and compute the grasp's quality. Their approach was adopted in GraspIt! to measure grasp quality of the *Mekahand*, and is described below.

Let $\dot{c_i}$ denote a set of contact points, $1 \leq i \leq \dot{n}$, and $\dot{n}$ denote contact points are used to grasp an object, with each contact given a Coulomb friction with coefficient μ. In Figure 1(a), the applicable contact force $\dot{F}$ must certainly lie within a friction cone that has a $\dot{F_\perp}$ with half-angle $\tan^{-1}\mu$. In Figure 1(b), $\dot{F}$ a nonlinear cone can be approximated by a pyramid i with $\dot{m}$ sides. A unit grasp force $\dot{f_i}$ is represented as a convex combination of $\dot{m}$ force vectors [24]:

$$\dot{F_i} \approx \sum_{j=1}^{\dot{m}} \dot{\alpha_{ij}} \dot{f_{ij}},\ \dot{\alpha_{ij}} \geq 0,\ \sum_{j=1}^{\dot{m}} \dot{\alpha_{ij}} = 1, \tag{1}$$

where $\dot{f_{ij}}$ denotes the jth force vector around pyramid i, and $\dot{\alpha_{ij}}$ are convex weights. Assume that a reference point $\dot{r}$ is the

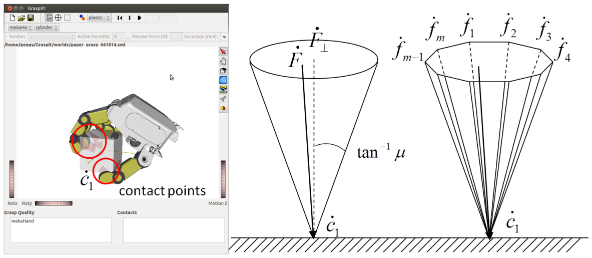


Fig. 1: (a) A single contact point in 3D illustrating the friction cone with half-angle $\tan^{-1}\mu$. (b) A unit grasp force $\dot{F}_i$ is represented by the convex combination of $\dot{m}$ vector.

object's center of gravity, that grasp forces $\dot{F}_i$ acting at a contact point on the object creates the torque $\dot{\tau}_i$, and that these forces and torque vectors can be concatenated to form a wrench, which is given by

$$\dot{w}_i = \begin{bmatrix} \dot{F}_i \\ (\dot{c}_i - \dot{r}) \times \dot{F}_i \end{bmatrix} \tag{2}$$

The grasp matrix $\dot{W}$ is formed by assembling a set of wrenches $\dot{w}_i$ for $\dot{n}$ contacts.

$$\dot{W} = \begin{bmatrix} \dot{w}_1 \dot{w}_2 \dots \dot{w}_n \end{bmatrix} \tag{3}$$

One grasp applied to an object, is defined as wrenches that are capable of keeping the object in static equilibrium when

$$\dot{W}_c = -\dot{w}_{ex}, \tag{4}$$

where c is the vector of contact wrench magnitudes and $\dot{w}_{ex}$ is the external disturbance wrench.

The properties of the convex hull of the applied wrenches were utilized to measure grasp quality. Ferrari et al. [25], [26] proposed one method of finding the unit grasp wrench space, so the set of wrenches $\hat{W}$ can be applied to the object:

$$\hat{W} = ConvexHull(\dot{W}). \tag{5}$$

If the convex hull contains the wrench space origin and no external disturbance force, then the grasp is stable. In the convex hull, the distance from the origin of wrench space to the nearest facet can thus be used as a quality metric for the grasp [24].

C. Neuroevolution of Augmenting Topologies (NEAT)

In experiments in this paper, grasping behaviors are evolved that are controlled by ANNs. Thus a neuroevolution method is needed to underpin these experiments. The NEAT method is appropriate because it is widely applied and well understood.

The NEAT method was originally developed to evolve ANNs to solve difficult control and sequential decision tasks [2], [27]. Evolved ANNs control agents that select actions based on their sensory inputs. NEAT begins evolution with a population of small, simple networks and *complexifies* the network topology into diverse species over generations, leading to increasingly sophisticated behavior. To keep track of which gene is which while new genes are added, a historical marking is uniquely assigned to each new structural component. During crossover, genes with the same historical markings are aligned, producing meaningful offspring efficiently. Speciation in NEAT protects new structural innovations by reducing competition among differing structures and network complexities, thereby giving newer, more complex structures room to adjust. Networks are assigned to species based on the extent to which they share historical markings. Complexification, which resembles how genes are added over the course of natural evolution, is thus supported by both historical markings and speciation, allowing NEAT to establish high-level features early in evolution and then later elaborate on them. A more comprehensive description of NEAT can be found in [2].

D. Visual Bounding Box

In the experiment, ANNs through exploration learn how to grasp objects by integrating information from a high-dimensional depth image provided by a Kinect sensor. To better focus on the most important features of the depth image, a bounding box strategy was implemented. For each object extracted from the original scene, image data was only considered from within a supervisor-specified bounding box. The bounding box thus serves to minimize the number of irrelevant pixels considered to simplify the learning problem.

The training process with the bounding box method is as follows. GraspIt! loads a scene, and then two mouse clicks from the user specify a rectangular bounding box that encompasses the object. In the simulated implementation, because all relative 2D coordinates of each object can be determined, an encompassing bounding box is automatically generated which is centered on the desired object. For simplicity, all the computed bounding boxes are the same size. The boundary range can be mapped to four coordinates. For example, in Figure 2, a cube is chosen, so the bounding box is $(C_x, C_y), (C'_x, C_y), (C_x, C'_y), (C'_x, C'_y)$. The depth array of the bounding box is then divided into $M \times N$ pixels that are then given to the ANN being evaluated as input data.

To simplify the implementation, the position of the camera sensor is always set such that the origin O_{3d} $(0,0,0)$ in the GraspIt! scene is in the center of the 2D plane, as shown in Figure 2. Because the input is reduced to a small part of the overall depth image, after the ANN produces the output, the position of each object must be offset relative to the bounding box. For example, in Figure 2, for the cube, Δx and Δy should be added to the position of the output, for mapping to the normalized origin position.

IV. LEARNING PROCESS

To combine neuroevolution with the grasping task requires specifying the input and output layers of the target neural network, as well as a fitness function to evaluate grasps. A schematic description of the general framework combining GraspIt! and NEAT is depicted in Figure 3. Here, there is no supervised optimal performance that we are trying to copy, but instead only reinforcement feedback in terms of the quality of the grasp. Therefore, evolution will discover solutions that work well even when we don't know what they are ahead of time.

Each ANN evaluated by NEAT receives input data denoting the current state of the robot in its environment. It is thus

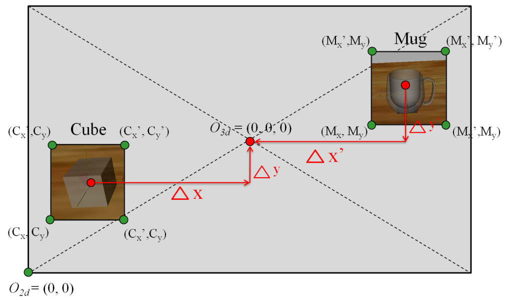


Fig. 2: Bounding boxes of a cube and mug, and the output shift offsets Δx and Δy ($\Delta x'$ and $\Delta y'$).

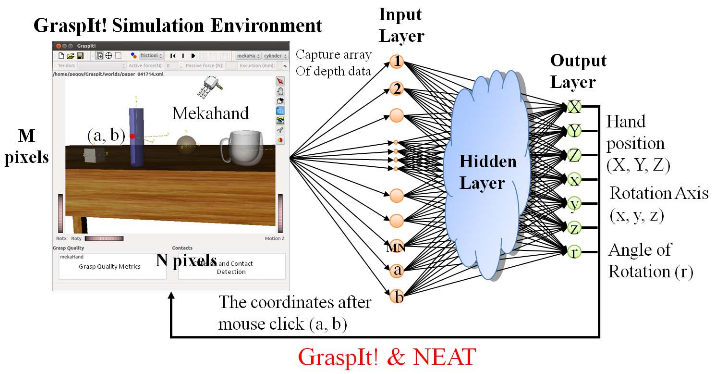


Fig. 3: Representation of the designed grasp controller network. The left side of the figure shows GraspIt! simulation environment; the right side of the figure shows a neural network receiving input consisting of depth data and the goal coordinate (a, b) on the GraspIt! visual input scene. The network has seven output nodes: hand position (X, Y, Z), rotation axis (x, y, z) and rotation angle (r). Note that NEAT can add internal hidden nodes as evolution progresses.

necessary to encode such state information, which includes the position of the target object as well as information about the object's shape. To eliminate dependency on high-level human-provided features of the grasped object, the object's state is given only by general low-level features provided by a depth map. In particular, each pixel in the depth information array is assigned a unique input node, as shown in Figure 3. In this way, the network can potentially learn to associate the state of an arbitrary object in an arbitrary environment with an appropriate grasping strategy.

Each ANN predicts where the object is and in what direction to grasp the object by outputting 3D hand positions and orientations. Note that each dimensional coordinate of the *Mekahand*'s position and orientation denotes one output neuron. Because the orientation is expressed as a axis angle format (e.g. a 3D axis vector and one angle), the total dimensionality is seven, meaning that the ANN has seven output neurons.

Evolution is initialized with ANNs with input nodes that are fully connected to a single hidden neuron, and with the hidden node fully connected to the output neurons. However, during evolution ANNs can accumulate additional connections and nodes through mutations in NEAT that augment network topology.

A. Grasping Fitness Function

The final aspect of the experimental design is to construct a fitness function to guide the search process for an appropriate ANN grasp controller.

In particular, in this experiment, the fitness of a network n with respect to an object O has three components:

1) fit_1: The reciprocal of Euclidean distance $Edist(\overrightarrow{P_i}, \overrightarrow{O_i})$ between the hand position computed by the neural network $(\overrightarrow{P_i})$ and a desired object $(\overrightarrow{O_i})$. Note that $\overrightarrow{P_i}$ and $\overrightarrow{O_i}$ are vectors.
2) fit_2: Grasp quality metric Q, described in Subsection III-B.
3) fit_3: The reciprocal of Euclidean distance $Edist(\overrightarrow{P_i}, \overrightarrow{S_i})$ between the hand position computed by the neural network $(\overrightarrow{P_i})$ and the actual hand coordinate after interacting with the environment $(\overrightarrow{S_i})$. Note that $\overrightarrow{P_i}$ and $\overrightarrow{S_i}$ are vectors.

Thus, the fitness function f of a network n is defined as follows:

$$\begin{aligned} f &= fit_1 + fit_2 + fit_3 \\ &= \frac{\beta}{Edist(\overrightarrow{P_i}, \overrightarrow{O_i}) + \alpha} + \gamma Q + \frac{\lambda}{Edist(\overrightarrow{P_i}, \overrightarrow{S_i}) + \epsilon} \end{aligned} \tag{6}$$

where α, β, γ, λ and ϵ are constants chosen to balance the various parameters.

The first term of the equation (fit_1) evaluates the distances between the hand and the object, while the second term (fit_2) is proportional to the grasp quality. During the initial phases of evolution, when the neural networks are mostly untrained, all networks may direct the *Mekahand* to grasp at positions where it cannot even touch the object. As a result, in early generations fit_2 is often effectively zero. Thus in this stage, fit_1, which rewards approaching the target object, is important for differentiating the fitness of the neural networks. After further evolution, when the hand can grasp the object, fit_2 then begins to dominate and the neural networks are ranked mostly by grasp quality. In addition, because the *Mekahand* may be blocked by obstacles (e.g. objects other than the target object), the third term (fit_3) evaluates the distance between the hand position given by the neural network and the resulting actual hand position after its interaction with the environment. Parameters α, β, γ, λ and ϵ adjust the relative effects of those three terms. Thus, this fitness function rewards neural networks to first learn to approach the object, and then to grasp the object in an increasingly appropriate way. The fitness algorithm is illustrated in more detail in Algorithm 1.

B. Speedup Learning Process: Parallel Implementation

The computational cost incurred by the *sequential* implementation of the fitness function computation is as follows. For one experiment, each generation consists of $\hat{o}$ ANNs, and each ANN is evaluated over $\hat{s}$ object combinations. Each object combination contains $\hat{b}$ objects, and each object is selected as $\hat{k}$ candidates to be an input. If one experiment runs for $\hat{g}$ generations, the total number of independent training

Alg 1 Computation of the Fitness Function

Input: P_i is the predicted position of hand for grasping by the network, O_i is the coordinate of the selected object after the mouse click, q is the grasp quality value after the execution of a single grasp, S_i is the actual hand coordinate after interacting with the environment.
Output: A fitness evaluation of a single grasp.
if quality $= 0$ **then**
  $fit_1 = \beta/(+\alpha)$;
  $fit_2 = 0$;
  $fit_3 = \lambda/ + \epsilon)$;
  return $sum = fit_1 + fit_2 + fit_3$;
else
  Assume that B_i is a set of three-dimensional coordinates of non-target objects in the environment;
  $A = \sum |O_i - S_i|$, where $i \in x, y, z$;
  $B = \sum |B_i - S_i|$, where $i \in x, y, z$;
  if A is smaller than B; **then**
    Mekahand is closer to the target object;
    $fit_1 = k$, where $k \leq 1000$;
    $fit_2 = \gamma$ quality, where $\gamma \geq 10000$;
    $fit_3 = \lambda/(\overline{N} + \epsilon)$;
  else
    $fit_1 = \beta/(\overline{M} + \alpha)$;
    $fit_2 = 0$;
    $fit_3 = \lambda/(\overline{N} + \epsilon)$;
  end if
  return $sum = fit_1 + fit_2 + fit_3$;
end if

simulations in GraspIt! T is $\hat{o}\times\hat{s}\times\hat{b}\times\hat{k}\times\hat{g}$. In our experiments, $\hat{o} = 120, \hat{s} = 5, \hat{b} = 2, \hat{k} = 10, \hat{g} = 100$. Thus, $T = 1,200,000$.

Figure 4.(a) illustrates the sequential method for each generation. To reduce execution time, the following computational steps were parallelized, as shown in Figure 4.(b). First, three commands are defined: *cmd_get_info* is to get the depth array, *cmd_get_quality* is to get the quality for each grasp, and *cmd_shuffle* is to change the position and orientation of each object. Here, assume that four instances of GraspIt! are run and waiting for commands. Two kinds of threads are created: *Organism tasks* accept ANNs from NEAT's main process to generate grasping tasks and collect the resulting fitness scores, while *GraspIt! tasks* accept ANNs and communicate with a GraspIt! process to send the output from a neural network for simulation in GraspIt!, and receive the resulting quality. The speedup rate depends on how many GraspIt! processes are run. Our results show that the run time is accelerated by at least a factor of three.

V. EXPERIMENTAL EVALUATION

In this section, training experiments are described that are conducted both with and without a bounding box. Next, testing experiments are described where fully trained networks are further tested in simulation and also transferred to the real robot.

A. Experimental Design

The raw depth data from the Kinect sensor is of high dimensionality, so for practical reasons the array is first down-scaled. Thus, before the input data is supplied to an ANN, the 640×480 pixel array was sampled to form a reduced 20×20

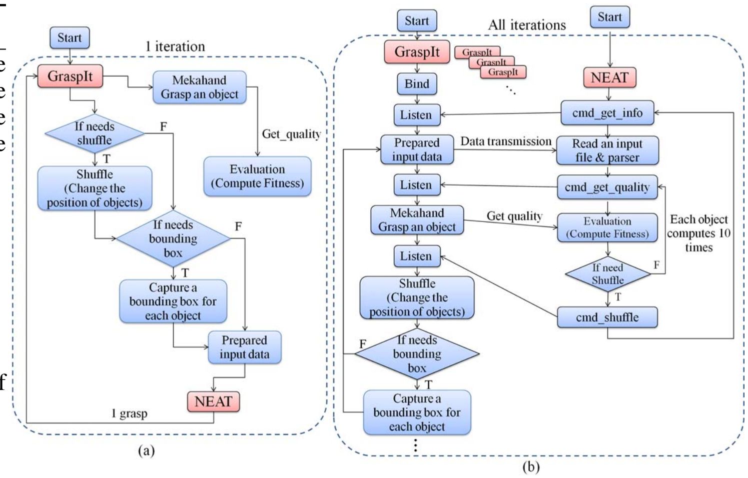
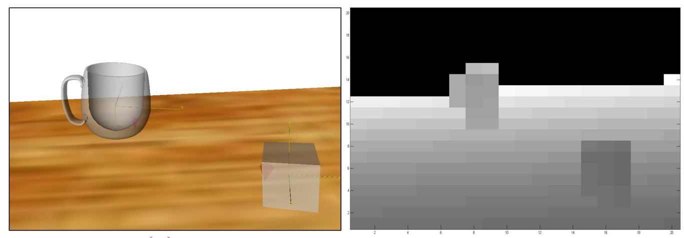


Fig. 4: (a) The original sequential method. (b) The faster parallel method.



(a) (b)

Fig. 5: Sample of input data for training neural networks (a) The RGB pixel data of the scene from the camera within GraspIt!. (b) The 20×20 depth data array supplied to the neural network as input. The depth data is normalized to a floating point number between [0, 1].

array. This smaller array was converted to gray-scale intensity values, and then normalized between zero and one; an example is shown in Figure 5. Here, the input data also includes a coordinate which represents the mouse click input from the user that specifies the target object. In the grasping experiments, the coordinate is chosen by randomly picking a different point on the target object in each trial. To increase accuracy in evaluating each network, they are each evaluated five times over different trials. In particular, for each evaluation, every possible target object is attempted five times, and then the average fitness value is calculated. To preserve generality, the position and orientation of the objects for each evaluation are randomized.

The experiments are divided into two parts: training and testing. For training, NEAT runs for 100 generations of evolution (neuroevolution does not require correct instances of behavior but learns them through optimizing the fitness function); for testing, the best neural network generated from training is further tested in simulations over objects placed in different locations. A final test applies a real scenario from *Dreamer* to the evolved neural networks. The flowchart for training and testing is shown in Figure 6. Detailed description of these parameters are given in [2].

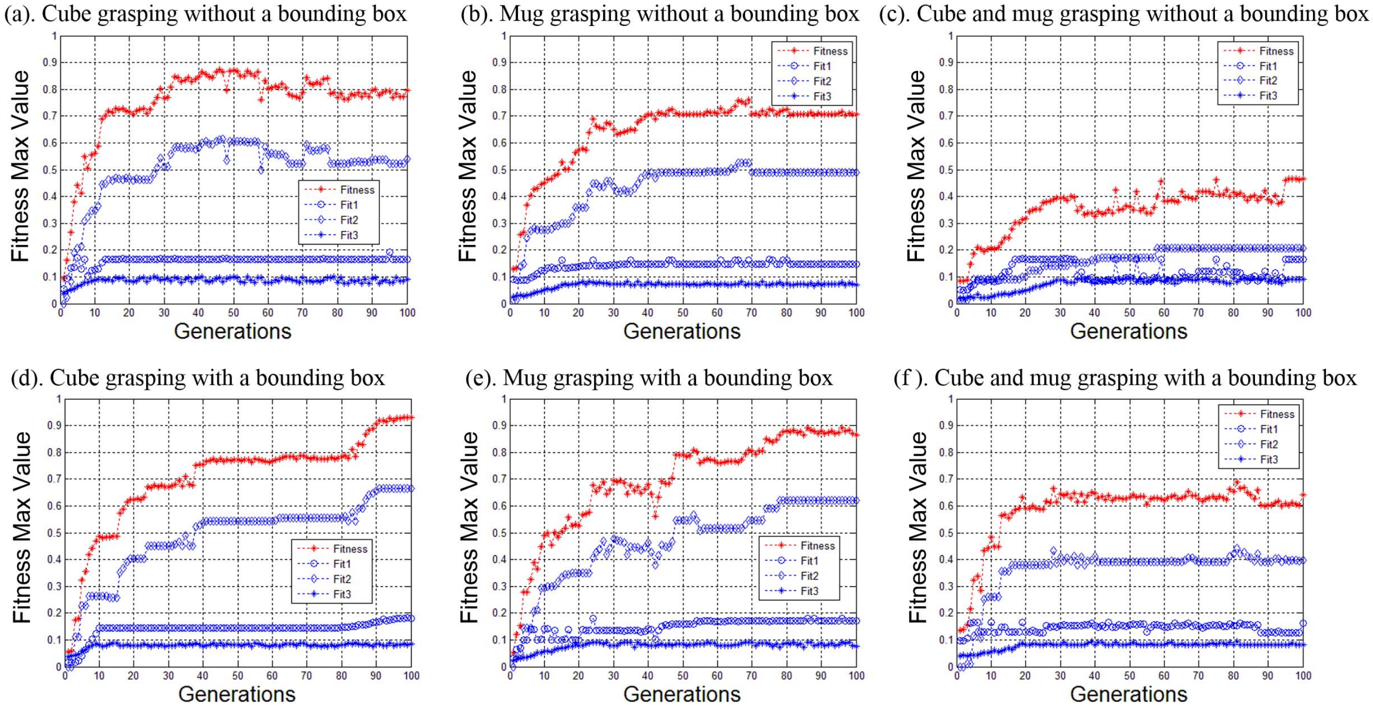


Fig. 7: Training performance. How fitness max values increase over generations is shown for each experiment. (a) and (d) show a scenario with a single cube on a table, (b) and (e) have a single mug on a table, while (c) and (f) include both a cube and a mug on a table. To explore if a bounding box helps improve performance, (a), (b), and (c) have no bounding box, while (d), (e), and (f) include the bounding box technique. The total fitness value is shown, as are the contributions from the three underlying normalized terms. The conclusion is that the bounding box increases performance, and that although including more than one object complicates evolution, all experiments eventually evolve ANNs able to grasp the objects in simulation.

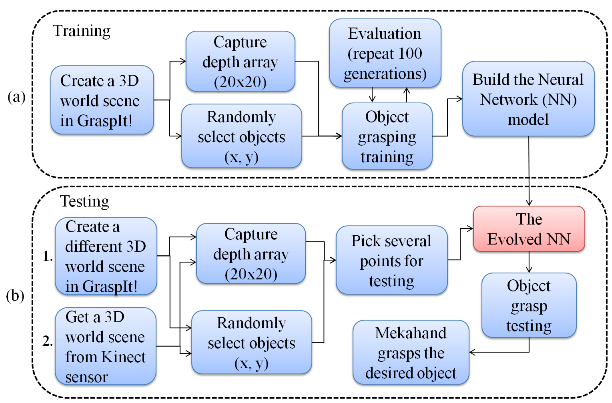


Fig. 6: The flowchart for the experiments. The training process is shown in (a), and the testing process is shown in (b).

B. *Experiments Results without and with a Bounding Box*

In the first set of experiments, three different training scenarios without a bounding box are performed with different target objects. Figure 7(a) and (b) shows training results for grasping a single cube object and a single mug object, respectively. Figure 7(c) shows training results for networks trained to grasp both a cube and a mug. These figures show how fitness values increase over the course of evolution. Note that larger fitness value implies better grasping quality; also, to differentiate the respective contributions from fit_1,fit_2, and fit_3, each term is normalized.

Because in practice only the best controller would be used, we present overall best-case results. Figure 7 shows that as expected, the best fitness value increases as evolution progresses. Note that the maximum score fit_1 can attain is 0.2, the maximum for fit_2 is 0.7, and the maximum for fit_3 is 0.1. Because fit_1 and fit_3 encourage approaching objects and avoiding obstacles (which only serve as secondary objectives), these terms are therefore given lower weights than that of fit_2, which represents grasp quality and is the most important performance metric.

To start evolution, individuals in the population are initialized with random weights and a simple topology (recall that input nodes are fully connected to one hidden node, and this hidden node is fully connected to the outputs). Because randomly generated policies generally do not cause the robot hand to approach the target objects initially, we expect to see low fitness scores. In this stage, fit_2 for all the networks is low, so the fitness scores of the networks are mainly determined by fit_1 and fit_3. These two terms guide evolution to produce networks that approach the objects without being blocked by obstacles. In accordance with this explanation, figure 7(a) shows that initially fit_1 is larger than fit_2 and fit_3. However, after 4 generations, fit_2 becomes dominant. Then, after 40 generations, fit_2 reaches its maximum value of 0.6, which means the *Mekahand* can grasp the object accurately with proper position and orientation.

Similar results appear in the other two experiments. In

Figure 7(b), after approximately 10 generations, fit_2 sharply increases, and the total fitness value steadily increases to reach a maximum value of 0.7. In Figure 7(c), two objects are used to train a neural network for each evaluation. Here, for the generality test, each object is shuffled three times. After 10 generations, the fitness value remains around 0.2. The reason is that distinguishing the two objects is difficult for the neural network. Comparing the three figures, it can be seen that the fitness scores of neural networks trained in the simple scene Figure 7(a) and (b) were larger than those trained in the more complicated one Figure 7(c). However, even in the more complicated scenario the networks all learned to approach the target objects and grasp them.

The second set of experiments tests evolution in the same three scenarios, but adds a bounding box. The first experiment is shown in Figure 7(d). The fitness value gradually increases, and after 40 generations, the values are better than Figure 7(a) and achieve a value of 0.92 after 90 generations. Also, the fitness values are more stable and do not oscillate around the maximum value as they do in the experiments without a bounding box. Similar results are seen in Figure 7(e). Figure 7(f) illustrates that with a bounding box, more complex object configurations can still produce consistent results; although the trend is comparatively slow, the maximum fitness value still approaches 0.7.

The experiments intuitively suggest that the more complex the training scenario (i.e. the number of different kinds of objects in the scene), the more difficult it is to train the neural network. Furthermore, if a facet is obscured or the depth array values of an object are similar to the background, then even if the object to be grasped is simple, the training results will remain poor. However, applying the bounding box technique can significantly improve the results.

C. Validating the Generality of Evolved Neural Networks

The training methodology results in neural networks evolved to grasp objects in simulation. To validate such networks, they can be further tested in a variety of novel (i.e. not explicitly trained for) situations through GraspIt!. Most objects in the scenes were not seen during evolution, and the experiment measures how general the evolved solutions are. A successful case is recorded if the *Mekahand* can grasp the object; otherwise it is recorded as a failure.

For this generality test, each object was tested 100 times. The grasping procedures were executed under test conditions arbitrarily placing a cylinder, a cuboid, a cube, a sphere, a plated mug, as shown in Figure 8, at different positions and orientations on the table. The success rate in Table I compares the neural networks with/without a bounding box. The results show that despite its simplicity, the proposed bounding box method still performs reasonably well in grasping novel objects. However, if the target object is too far from the center of the image frame, the neural networks often perform unreliably, indicating the training process may need further refinement to deal with such boundary cases. Table I shows the best results from among all the experiments. Also, in some cases the *Mekahand* collides with objects while grasping, because

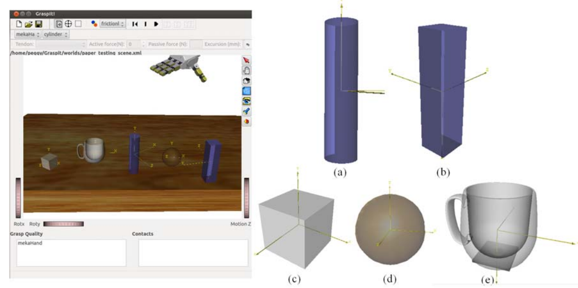


Fig. 8: Testing the objects in different locations, including a cylinder, a cuboid, a cube, sphere, and a plated mug.

#	Objects	without Bounding Box	with Bounding Box
(a)	cylinder	52%	72%
(b)	cuboid	65%	82%
(c)	cube	82%	92%
(d)	sphere	54%	80%
(e)	plated mug	61%	79%
	Mean/Std	62.80 ± 11.95%	81.00 ± 7.21%

TABLE I: Grasping objects at different positions with the evolved networks

many objects are placed on the table. A potential remedy is to decompose the movement into more steps to avoid such collisions. This can be done most easily with the human supervisor's input to provide assistance.

D. Validating in Reality with Dreamer

To validate the simulation results against reality, a physical (i.e. not simulated) Kinect sensor is applied to capture object depth array information. This information is provided as input to an evolved neural network to guide *Dreamer* robot's grasp.

To carry out an experiment, the human chooses a target either without or with a bounding box in the color image from the laptop screen with the Kinect sensor by clicking on it. After designating the target, a copy of the color image is copied to the target object panel, and a red dot is added on the image to indicate where the user has clicked. The position of the click on the image and the depth data at that point are used to calculate the approximate position of the object to be grasped. This specifies the grasping task for the robot to perform. Note that the grasping behavior was not evolved on the actual robot, but was transferred from simulation.

The video[1] demonstrates robotic grasping of novel objects in our implementation. Figure 9 shows screen captures taken from a proof-of-concept demonstration of grasping by the real *Dreamer* robot, that demonstrates robotic grasping of a tennis ball, a bottle using an evolved controller. The results illustrate that *Dreamer* can successfully approach and grasp target objects when controlled by an evolved neural network. These objects were not seen during evolution, so that the experiment demonstrates two things: that the learning transfers

[1]http://www.cs.utexas.edu/~peggy/ssci2014.html

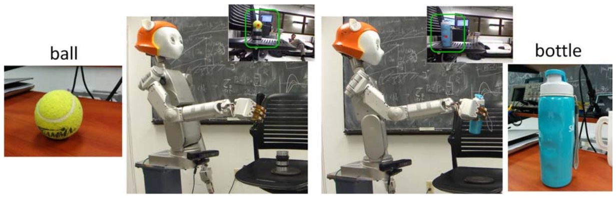


Fig. 9: Screen captures from videos demonstrating *Dreamer* grasping a ball, a bottle through an evolved controller. The small pictures with red dots are the snapshots from the laptop screen with the Kinect sensor.

from simulation to the real world, and that it generalizes to novel objects. However, quantifying how well grasping in a real environment still needs a metric for the assessment of the quality of a real grasp, so further work to incorporate real sensor data on the *Mekahand* (e.g. touch pressure) is ongoing.

VI. FUTURE WORK

For future work, further improvement is required to produce more robust grasping with more real objects; it appears that one constraining factor is the limited precision of the Kinect sensor at close range. A possible remedy is to use the bounding box technique for continuous visual tracking of the robot's hand to overcome this limitation. Another future direction is to investigate off-line techniques to reduce model and sensing uncertainties by exploiting pre-computed information about object types, rather than relying entirely upon low-level features of the object to be grasped. Finally, we aim to develop an algorithm that can take sensory inputs and return a measure of the physical *Dreamer* robot's grasp quality to quantify concretely how well grasping controllers transfer to reality.

VII. CONCLUSION

This paper describes an approach to exploit neuroevolution as a training method to enable a real robotic system to grasp unforeseen objects in an unstructured environment. Given only low-level depth data from a Microsoft Kinect sensor as input, our approach evolves neural networks to predict the appropriate hand configuration (i.e. positions and orientations) for grasping a target object. The right solutions are not known but discovered by NEAT in order to maximize the quality of the grasp. One contribution was to show that introducing a visual bounding box is an effective way of increasing grasping performance. Further, the approach bypasses the expensive and time-consuming process of training with a physical robot in a real environment, by training off-line in a simulated environment and later transferring the resulting policies to the physical *Dreamer* robot. Indeed, through combining GraspIt! and NEAT, successful grasping controllers were attained in simulation, and the initial transfer of ANNs to the real *Dreamer* also produced promising results. The main thesis of our approach is that while unattainable today, full autonomy in many robotic tasks may best be first tackled by a collaborative approach wherein the robot is supervised by a human supervisor. Supporting this idea, the results here demonstrate how this approach can achieve robotic grasping of unforeseen objects in unstructured environments.

REFERENCES

[1] J. Lehman and R. Miikkulainen, "Neuroevolution," *Scholarpedia*, vol. 8, no. 6, p. 30977, 2013. [Online]. Available: http://nn.cs.utexas.edu/?lehman:scholarpedia13
[2] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ec02
[3] J. C. Bongard, "Behavior chaining: incremental behavioral integration for evolutionary robotics," in *Artificial Life XI*. Cambridge, Massachusetts: MIT Press, 2008, pp. 64–71.
[4] J. C. Bongard, "The utility of evolving simulated robot morphology increases with task complexity for object manipulation," *Artificial Life*, vol. 16, no. 3, pp. 201–223, 2010.
[5] "The Human Centered Robotics Laboratory," The University of Texas at Austin, http://www.me.utexas.edu/~hcrl/.
[6] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
[7] A. T. Miller, S. Knoop, P. K. Allen, and H. I. Christensen, "Automatic grasp planning using shape primitives," *In International Conference on Robotics and Automation (ICRA)*, 2003.
[8] R. Pelossof, A. Miller, P. Allen, and T. Jebara, "An svm learning approach to robotic grasping," *In International Conference on Robotics and Automation (ICRA)*, 2004.
[9] J. H. Piater, "Learning visual features to predict hand orientations," *In ICML Workshop on Machine Learning of Spatial Knowledge*, 2002.
[10] J. Coelho, J. Piater, and R. Grupen, "Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot," *Robotics and Autonomous Systems*, vol. 37, pp. 195–218, 2001.
[11] I. Kamon, T. Flash, and S. Edelman, "Learning to grasp using visual information," *In International Conference on Robotics and Automation (ICRA)*, 1996.
[12] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *International Journal of Robotics Research (IJRR)*, vol. 27, no. 2, pp. 157–173, 2008.
[13] A. Saxena, "Monocular depth perception and robotic grasping of novel objects," Ph.D. dissertation, Stanford University, Stanford, USA, 2009.
[14] D. Rao, Q. V. Le, T. Phoka, M. Quigley, A. Sudsang, and A. Y. Ng, "Grasping novel objects with depth segmentation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2578–2585.
[15] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, "Learning to grasp objects with multiple contact points," in *Proceedings of ICRA*, 2010, pp. 5062–5069.
[16] N. Jakobi, "Minimal simulations for evolutionary robotics," Ph.D. dissertation, University of Sussex, School of Cognitive and Computing Sciences, 1998.
[17] H. Lipson, J. C. Bongard, V. Zykov, and E. Malone, "Evolutionary robotics for legged machines: From simulation to physical reality," in *Proceedings of the 9th International Conference on Intelligent Autonomous Systems*, 2006, pp. 11–58.
[18] N. Rezzoug and P. Gorce, "Robotic grasping: A generic neural network architecture," *IEEE Trans. Robotics*, 2006.
[19] L. M. Pedro, V. L. Belini, and G. A. Caurin, "Learning how to grasp based on neural network retraining," *Advanced Robotics*, vol. 27, no. 10, pp. 785–797, 2013.
[20] J. Zhang and B. Bössler, "Self-valuing learning and generalization with application in visually guided grasping of complex objects," *Robotics and Autonomous Systems*, vol. 47, pp. 117–127, 2004.
[21] N. Kohl, K. Stanley, R. Miikkulainen, M. Samples, and R. Sherony, "Evolving a real-world vehicle warning system," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2006. [Online]. Available: http://nn.cs.utexas.edu/?kohl:gecco06
[22] A. Miller, P. K. Allen, V. Santos, and F. Valero-Cuevas, "From robot hands to human hands: A visualization and simulation engine for grasping research," *Industrial Robot*, vol. 32, no. 1, pp. 55–63, Jan. 2005.
[23] "GraspIt! Software," http://www.cs.columbia.edu/~cmatei/graspit/.
[24] A. T. Miller and P. K. Allen, "Examples of 3d grasp quality computations," *IEEE International Conference on Robotics and Automation*, vol. 8, pp. 1240–1246, 1999.
[25] R. M. Murray, Z. Li, and S. S. Sastry, "A mathematical introduction to robotic manipulation," *CRC Press*, 1994.
[26] C. Ferrari and J. Canny, "Planning optimal grasps," *IEEE International Conference Robot Automation*, vol. 3, pp. 2290–2295, 1992.
[27] K. O. Stanley and R. Miikkulainen, "Competitive coevolution through evolutionary complexification," *Journal of Art. Intell. Research*, vol. 21, pp. 63–100, 2004.