An Approach to Design Language-Conditioned Low-Level Visual Controller

Tonmoay Deb[1]

Department of Computer Science, University of New Hampshire, Durham, NH 03824
td1165@wildcats.unh.edu

Abstract. In this research, we study designing robot's low-level controllers directly from visual data and apply on certain complex manipulation tasks. To understand the inherent goal from multiple objects, we incorporate language commands to train the robot to learn inherent syntactic properties inside the language, which can be utilized to accomplish certain tasks. In summary, we approach designing a language-modulated low-level controller for manipulation, where the robot does not have any certain information about the environment. To focus on a specific goal, we integrate a language-conditioned attention module, followed by a deep neural network to focus on specific target objects. To compare the efficacy of the attention mechanism, we evaluate our approach with another baseline that directly predicts control commands from raw language embedding. In addition, to demonstrate the limitations and emphasis further improvement, we derive another baseline that integrates attention with a dynamic system-based low-level control signal generator. We performed a full-scale Human-Robot Interaction (HRI) study on 7 people to evaluate the respective baseline and ours to illustrate the efficacy from a real-world perspective. Overall, our initiative certainly indicates the possibility of designing a vision-based low-level controller for dealing with complex tasks in uncertain environments, e.g., where environment dynamics change frequently, and the robot does not require explicitly knowing that (if only visual information is required) for manipulation.

1 Introduction

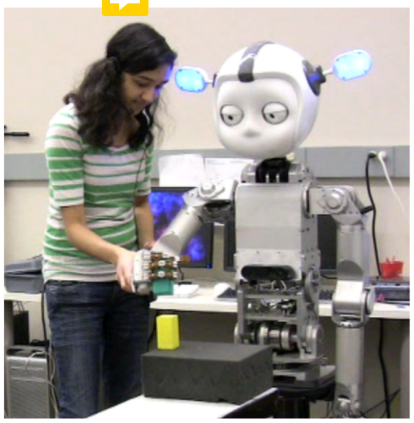
There have already been certain advancements in Robotics with numerous applications in several domains, e.g., healthcare, therapy, navigation, etc. [33]. Most of the problems in Robotics require the robot to learn certain skills to perform specific tasks, e.g., manipulation. The commonly used paradigm to teach the robot is imitation learning, where an expert teaches a robot to perform a certain task, and it is expected that the robot will perform that same task even with some perturbations involved, i.e., environment and context changes [36, 5]. Teaching a robot hand technically means training it to predict joint angles to perform a certain operation. For example, if a robot is about to pick an object from a certain location, at first, an expert has to do this exact task, which is called Kinesthetic Demonstration [1]. This approach is the fundamental way to

direct a robot on calculating the probable joint angles. In Figure 1, we can see two examples of kinesthetic demonstration. The left one is done for real-world robots by expert humans, whereas the right one is done in the simulator. For both cases, the robot hands try to manipulate the object, and the right one is relatively complex as it tries to perform a certain task from a multiple set of objects. For both cases, the main objective is to learn about generating control signals to reach the target. The control signal generation is typically done by a low-level controller. There have been several existing works done in designing and improving the performance of low-level controllers [15, 22, 26]. One of the widely accepted approaches in Dynamic Movement Primitives (DMP) [23, 10], which works as a time-series predictor model to calculate the joint angles given the current and goal position. However, DMP comes with a limitation of knowing the exact environment details prior to performing the tasks. To tackle this problem, another line of research included purely visual features [29] to design the control commands. However, for complex manipulation tasks like the right portion of Figure 1, there has been limited work. More specifically, kinesthetic demonstration with more abstract information, i.e., language, puts an extra hurdle to learn the specification accurately. In this work, we approach modulating the high-level language information with purely visual data to generate a low-level controller. One of the biggest advantages of our method is to let the robot learn latent information from kinesthetic demonstration with language so that it can understand how to predict joint angles in complex scenarios using visual attention [30, 36]. For example, in the right figure of Figure 1, if the robot is given an initial image with language *grasp the grail*, can the robot understand what is *grail* and generate control signals (joint angles) from the consecutive image features to reach that target? We further claim that the attention mechanism is important rather than raw language features for low-level controllers by a rigorous quantitative and Human-Computer Interaction (HRI) study done by real humans.

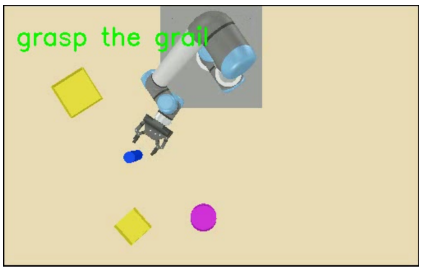
2 Related Work

2.1 Progress on Attention Mechanism

Attention mechanism have recently become popular to extract semantic information from high-dimensional data e.g., images [2, 32]. The attention mechanism is mainly inspired from the biological vision, where humans usually put focus on a specific region to use that sort of direction for doing further activities [17]. Attention has recently been popular for solving fine-grained classification activities, e.g., object, action recognition, etc [34, 20]. Because these literatures claim that the biological neurons do not need to be active on all regions of the focal point. The same applies to the context of deep learning models; focusing on a specific region based on visual inputs helps better classification [13]. This is also true for handling a large amount of data. We agree with the existing works and emphasize integrating attention mechanisms to facilitate robots focusing on specific objects or regions to perform certain tasks. Although there have been



(a) Real Robot



(b) Simulator Robot

Fig. 1: Performing Kinesthetic Demonstration on real and simulator robot.

some works on designing low-level controller by using attention modules [30], no works integrated pure vision-based low-level controller that integrates visual attention.

2.2 Bridging Between Language and Vision

A number of works have been recently focused on bridging multiple modalities, e.g., text, vision, speech, etc. [36, 3] to perform certain tasks. One of them is Visual Question Answering (VQA) [3]. In VQA, the goal is to use natural language to query an image and get answer from that. The main model mainly connects two modalities, language, and vision, to create an inter-modal representation for facilitating the query [6]. For the last few years, tremendous progress was made by integrating attention (discussed in the earlier section) mechanism for inter-modal feature representation. In our work, we take the concept from VQA, where our goal is to predict joint angles other than answering natural language questions. Our inherent attention module is a simplistic version of the recent VQA approach [36], where we create another module dedicated to predicting joint angles as discussed in the next section.

3 Problem Formulation: Network Architectures

This section will discuss the basic strategies to process two core components, language and image data, toward generating the control signals.

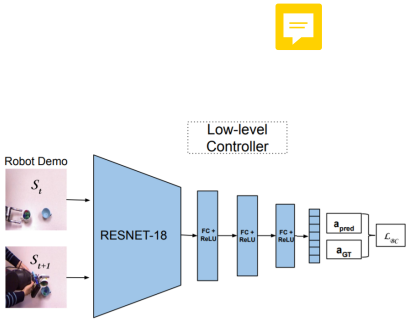
3.1 Language-Conditioned Attention Network

Processing Language There have been several earlier literature [8, 24] who discussed about efficient processing of language for Deep Learning. In our con-

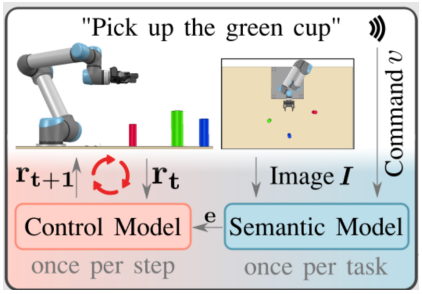
text, we will use language as a 'key' component, which is commonly seen in Visual Question and Answering (VQA) domain [6]. For input sentence S, at first, it is converted into lower-case, followed by removing punctuation. After that, S is split into multiple words by space. We Say $S = \{s_1, s_2, ..., s_n\}$, where n is the maximum number of words in a sentence. For fitting the model, we pad all sentences by maximum n n. For each s_i, we extract 300 dimensional GloVe [25] word embedding, which is is trained on a large linguistic corpus and expected to generate an unique, representative vector for a word. We define Globe Embedding as $G(s_i)$, which predicts $v_i \in \mathbb{R}^{300}$ for respective word s_i. The final language embedding is a $V = \{v_1, v_2, ..., v_i\} \in \mathbb{R}^{n \times 300}$. We further utilize this vector set either to integrate it directly (by averaging all sentence vectors) or passing through a sequential network, e.g., GRU/RNN to predict the final vector to taking certain decision, e.g., calculating attention based on these vector set.

Processing Image In our case, an image will work as a 'query' component, where the language query will be used to find the appropriate key, i.e., target object with the location. We used the existing state-of-the-art object detector Faster R-CNN [27] to detect the objects from the robot's environment image, followed by processing respective objective features for generating a salient map. More specifically, we initially pre-trained Faster R-CNN model and pass RGB image $I \in \mathbb{R}^{568 \times 330 \times 3}$ taken from the robot's viewpoint as the input. From there, Faster R-CNN first of all predicts o objects $\mathbb{X} = \{x_1^o, x_2^o, ..., x_p^o\}$ number of objects above certain confidence threshold $confidence(object) \in [0, 1]$. For our case, we have selected the minimum confidence threshold as 0.5. For each x_i^o, the object detector model predicts respective feature vector (that it extracted for object recognition) x_i^f and bounding box $x_i^b \in \mathbb{R}^4$. For this experiment, we utilize the object feature and bounding box location to design the attention network. We concatenate the object features with the language features in a Convolutional Neural Network during the training stage, where the output is usually the bounding box locations. The attention network learns to connect the sentence with objects to properly predict the correct bounding box (where to look for the target) region (essentially a 'value' component). During training, the target box is used as the positive sample, whereas the other ones are used as negative. The training component works as a triplet loss. We refer readers to Section 3.1 for a details discussion on the attention network construction. The initial Faster-RCNN model is trained on MSCOCO [18] dataset and uses ResNet-101 [9]. However, considering our object context, we had to fine-tune (re-train on our object information) and update the existing weights till convergence according to our new dataset. This allows Faster R-CNN to adjust to the novel environment.

Computing Attentive Feature Vector After the image and language features are processed, computing attention feature vectors is relatively straightforward. At first, the language features are passed into a GRU layer, and the output at final step 32 dimensional. Formally $s = GRU(V) \in \mathbb{R}^{32}$. After that,



(a) Basic low-level visual controller.



(b) Overall pipeline of 'Baseline 2'.

Fig. 2: A basic low-lever visual controller (left) and 'Baseline 2' pipeline (right).

the language feature s is concatenated with each candidate object x_i^f to compute the likelihood of each candidate as $a_i = w_a^T f_a([x_i^f, s])$. Here after concatenation of sentence of each object feature, the attention $f_a : \mathbb{R}^{32} \rightarrow \mathbb{R}^{64}$, which is further converted into scalar by multiplying with $w_a \in \mathbb{R}^{64}$. The f_a is a nonlinear transformer to map all candidate objects to respective likelihoods. Finally, we have $a = \text{softmax}([a_0, ..., a_c])$ for c candidate regions. The final weighted average for each candidate image feature x_i is calculated as $e' = \sum_{i=0}^{c} x_i a_i$ where $e' \in \mathbb{R}^5$ as we select top 5 object candidates. As per the recommendation from earlier works in attention [30], we re-introduce the sentence embedding and calculate the final feature $e = \text{ReLU}(W[e', s] + b)$, where W and b are weights that map $e \in \mathbb{R}^{32}$. For directing the controller better, we concatenate $e \in \mathbb{R}^{32}$ with $a \in \mathbb{R}^5$ and initial joint angles $j \in \mathbb{R}^7$ to calculate the final feature set $att \in \mathbb{R}^{44}$, which is further used in the proceeding networks (discussed in the next sections). Intuitively, the feature vector calculates sufficient information for robot to focus on a single target from the language-conditioned command.

3.2 Frame-Difference Continuous Control Value Updater Network (FCVN)

The fundamental goal of FCVN is to predict tentative joint angles for the reaching next timestep $t + 1$ based on the current frame t and previous frame $t - 1$ features. The core network architecture is based on this image feature mapping to joint angles [29] as illustrated in Figure 2a. However, authors of [29] only had one single goal, and the target position was always positioned in the same location with no shape difference. However, in our case, the robot needs to accomplish doing certain complex tasks based solely on natural language commands. For that purpose, an additional language-focused feature is introduced in FCVN. Figure 3 is a sample block diagram of FCVN. The initial t and $t - 1$ image features are concatenated and passed to several Multilayer Perceptron, e.g., learnable dense layers. In addition, we include the language-based feature,

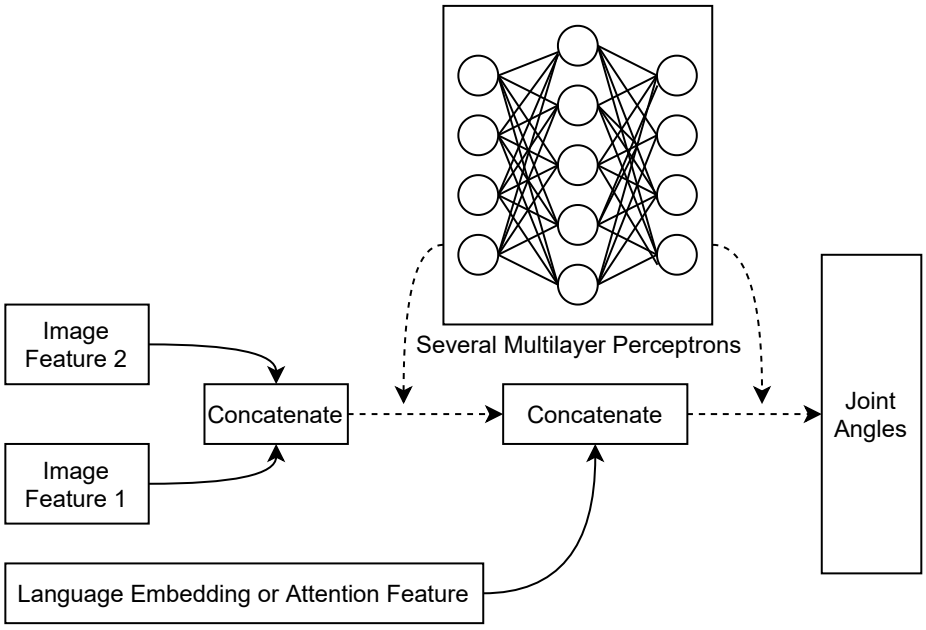


Fig. 3: A simple outline of Frame-Difference Continuous Control Value Updater Network, with basic functionalities. The 'Multilayer Perceptons' are the components that are used in between the concatenations. There are several copies of them, which learns weights. Parameters are more discussed in Section 4.4.

e.g., sole language feature or language-modulated attention feature, to concatenate with the immediate dense layer and pass through additional dense layers to predict the joint angles. The reason for using multiple dense layers is to let the controller module enough parameters to learn to map the feature vectors. A detailed description of how they have been used is discussed in the following sections.

3.3 Baseline 1: Visual Controller with only Language Information (without Attention)

As discussed in the earlier section, we have processed the language data as 'key' for 'querying' image to predict 'value', which is the bounding box region to put attention. Technically, by 'image' we meant the first static image of the robot scene. To reach a specific goal or perform a specific task, processing continuous frames is an important goal. Also, our main objective was to claim that the attention mechanism improves low-level visual controllers. For claiming this, we are designing 'Baseline 1' Network, which does not include attention, rather uses only raw language features and continuous frame features to predict the trajectory. The implication here is that we are not explicitly using the image as

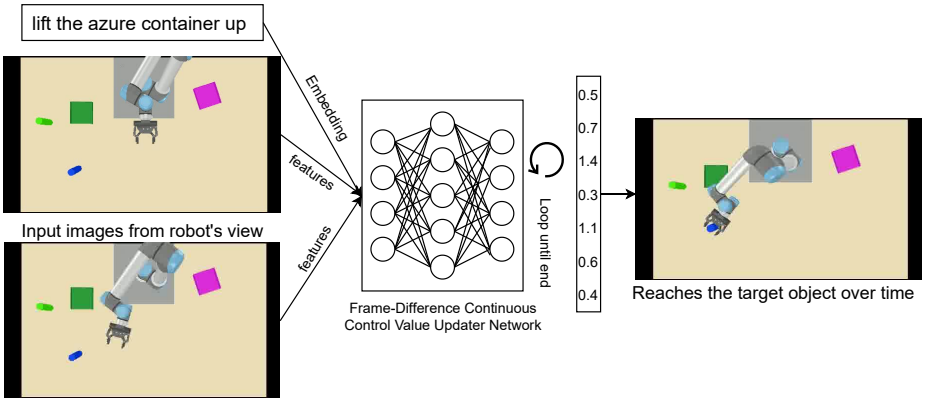


Fig. 4: Network architecture of 'Baseline 1'. It directly uses language information without attention for joint angle prediction.

a query or language sentence as a key. Also, we never provide the ground truth object bounding box. We expect that the robot will learn how to reach certain objects without a primary target, i.e., by analyzing the target location from the continuous joint angle values. In this line of assumption, we integrate the FCVN network that takes language features once per demonstration and frames over time steps to map the image difference toward predicting the robot joint angles. Figure 4 illustrates the overall pipeline. In the top left row, we can see the sentence, which is further embedded and averaged as discussed in Section 3.1. After that, we take the robot's two initial frames by using the ground truth trajectories. We can say that the bottom left image is t and the upper one is $t-1$. Further, for each of these images, we predict the global image feature using the pre-trained ResNet-18 [9] model. All three features are passed as input to the FCVN as discussed in Section 3.2. The network predicts the joint angles for the further timestep $t+1$. After this step, the frame at $t+1$ and t timestep is passed to the network for predicting joint angles at $t+2$ timestep, which keeps continuing until the number of max steps mentioned in the ground truth. We note that we use the language feature once for one demonstration, which relatively works as a static feature set during the timesteps.

3.4 Ours: Visual Controller with Language-Modulated Attention

We argue that the sole language feature and the global image feature descriptor can not properly identify the low-level controller network (FCVN) in the proper direction. Because only language information, when passed as a compressed vector (by averaging), may lose important syntactic components, e.g., shape and color information. Moreover, when FCVN tries to connect the language features with visual ones, they may not properly converge due to the large modality gap. The issue regarding the modality gap has been widely studied in the VQA

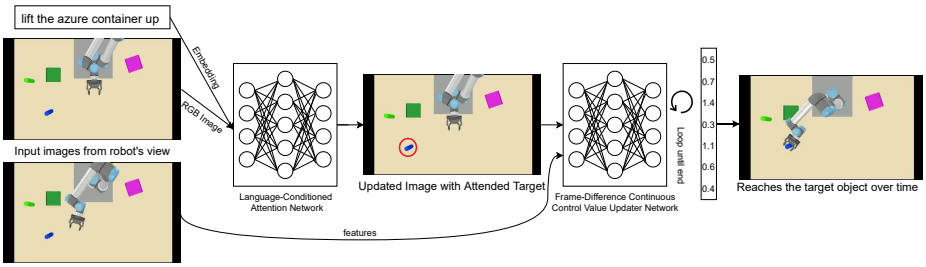


Fig. 5: Our proposed pipeline, which uses attention network to modulate the language for fixing the target and better directing FCVN model.

domain [6]. We also argue at the same point and utilize the attention mechanism as discussed in Section 3.1 to process the language modulation with the initial frame first, mainly to fix the probable target bounding box derived from the network. The next consecutive frames will only use the feature information of the target, which will lead the FCVN to generalize better prediction over time. Figure 5 illustrated our overall pipeline. We can see that the initial language embedding and the image (without feature extraction) are passed to the Language-Conditioned Attention Network. The network processes the language vectors using the GRU unit and connects the features with Faster R-CNN object information as discussed in Section 3.1 to produce the feature of the target box (shown in the red circle of the figure). Later, this attention feature is used as static information and passed to the FCVN model with the consecutive frames with regular ResNet-18 features, as we discussed in the earlier section. Here we note that the FCVN now focuses only on the image features, one is coming from the attention module, and the others are the timestep image features. So, technically, there is no modality gap that was present in 'Baseline 1'. We expect that this approach should work better compared to 'Baseline 1' for this particular reason.

3.5 Baseline 2: Non-Visual Controller with Language-Modulated Attention

We have discussed generating low-level control signals (joint angles) solely from the visual data in the last two sections. More specifically, the FCVN is dominated by image features, and no specific locations are fixed. So, based on only pixel-level information, the joint angles are being predicted. However, is another line of research, which takes the trajectory generation problem as time-series prediction [35] and continues predicting by not requiring consecutive image data, rather from the target location and current location and distance between them. To accomplish, it uses Dynamic Movement Primitives (DMP) [23, 10] to predict the angles as the abstract process illustrated in Figure 2b. DMP is a widely studied field for generating smoother trajectories in mostly ideal environments, e.g., a simulator. In this case, the target location is set by the attention network,

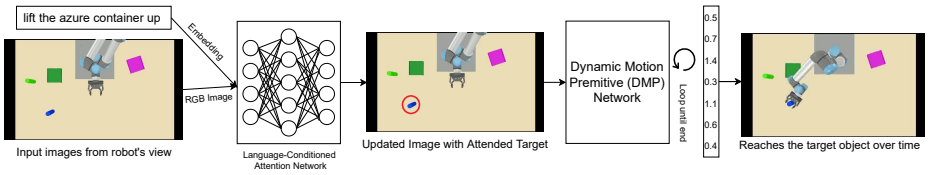


Fig. 6: Pipeline of 'Baseline 2'. It uses simple DMP to predict the trajectories from the goal set by the attention module.

and the DMP module predicts continuous joint angles based on that. Figure 6 illustrates the overall pipeline of 'Baseline 2.' We can note the similarity of this pipeline with ours till the attention network. After the target box $x_i^b \in \mathbb{R}^4$ is predicted from the module as discussed in Section 3.1, it is passed to the DMP module. Here we note that the DMP does not get the pixel-level target information as ours and 'Baseline 1' had. Rather it gets the relative information of that object with respect to the environment of the robot simulator. During the continuous steps, it tries to reach that exact simulator location. We can note that, although simpler, DMP requires the environment information beforehand. Based on several studies [7, 21, 19], it fails when it comes to a real-world context, i.e., coping up with environment changes, noisy information. On the other hand, our approach can adapt to any context as it does not need any environment information and works on pixel-level data from the images. However, we expect that our approach will perform closer to 'Baseline 2' even it does not have access to the environment properties.

4 Experimental Analysis

4.1 Simulator Details

We have conducted all experiments in simulation concerning the COVID-19 pandemic[1]. For that, we used CoppeliaSim [11], i.e., V-REP simulator, which provides dynamics simulation at a $20Hz$ update rate. For our experiment, similar to the setup from [30], we used cups and bowls with 20 total variations (two sizes, two shapes, and five colors). We used UR5 Manipulator Robot [14] for the experiments. The object setup was random and the simulator itself generated the ground truth trajectories. Both object setup and trajectories were recorded for the dataset creation.

4.2 Dataset and Subsampling

For our experiment, we have incorporated the dataset of [30] for a fair comparison with respect to the simulation context. The authors of [30] generated random movements with objects done by the UR5 robot in simulation and asked several

[1] https://www.cdc.gov/coronavirus/2019-ncov/index.html

```
class Controller_NN(nn.Module):
    def __init__(self):
        super(Controller_NN, self).__init__()
        resnet18 = models.resnet18(pretrained=True)
        self.modified_pretrained = nn.Sequential(*list(resnet18.children())[:-1])
        self.relu = nn.ReLU()
        self.linear1 = nn.Linear(512*2,512)
        self.linear2 = nn.Linear(512,256)
        self.linear3 = nn.Linear(256,128)
        self.linear4 = nn.Linear(128,44)
        self.linear5 = nn.Linear(44*2,32)
        self.linear = nn.Linear(32, 7)


    def forward(self, input1,input2,input3):
        i_1 = self.modified_pretrained(input1)
        i_2 = self.modified_pretrained(input2)
        i_combined = torch.cat((i_1,i_2),1)
        i1 = self.relu(self.linear1(i_combined.squeeze(2).squeeze(2)))
        i2 = self.relu(self.linear2(i1))
        i3 = self.relu(self.linear3(i2))
        i4 = self.relu(self.linear4(i3))

        att_combined = torch.cat((i4,input3),1)

        h_t = self.linear5(att_combined)
        output = self.linear(h_t)
        return output
```

Fig. 7: FCVN with attention features included.

human evaluators to provide language commands for that task. The core annotation was done on 200 tasks ranged over 20 total categories. Based on that, they extracted multiple demonstrations and augmented sentences (by using synonyms and switching syntactic structure) to generate in total 45000 demonstrations and natural language commands. There, 22500 consisted 'picking', and the rest 22500 was pouring tasks. In total, they trained 'Baseline 2' 3.5 on 40000 demonstrations, evaluated on 4000 and tested on 1000. For our experiment, we narrowed down the problem by selecting only the 'picking' task. Out of that 22500 tasks, we randomly sub-sampled 500 demonstrations and maintained standard 70% train, 10% validation, and 20% test. It resulted in 350 for training, 50 for validation, and the rest 100 for testing.

4.3 Preprocessing

The preprocessing was performed in two levels. The first objective was to find out the objects correctly from the simulation images. To accomplish that, we have used Faster R-CNN [27] object detector model. However, as the Faster-RCNN model was trained on real-world data and ours are from simulation, we further fine-tuned the model with all possible ground truth object data to perform up to adapt with the environment. For processing the language, we have

```
class Controller_NN(nn.Module):
    def __init__(self):
        super(Controller_NN, self).__init__()
        resnet18 = models.resnet18(pretrained=True)
        self.modified_pretrained = nn.Sequential(*list(resnet18.children())[:-1])
        self.relu = nn.ReLU()
        self.linear1 = nn.Linear(512*2,512)
        self.linear2 = nn.Linear(512,300)
        self.linear3 = nn.Linear(300*300,128)
        self.linear4 = nn.Linear(128,64)
        self.linear5 = nn.Linear(64, 32)
        self.linear = nn.Linear(32, 7)


    def forward(self, input1,input2,input3):
        i_1 = self.modified_pretrained(input1)
        i_2 = self.modified_pretrained(input2)
        i_combined = torch.cat((i_1,i_2),1)
        i1 = self.relu(self.linear1(i_combined.squeeze(2).squeeze(2)))
        i2 = self.relu(self.linear2(i1))

        lang_combined = torch.cat((i2,input3),1)

        i3 = self.relu(self.linear3(lang_combined))
        i4 = self.relu(self.linear4(i3))
        h_t = self.linear5(i4)
        output = self.linear(h_t)
        return output
```

Fig. 8: FCVN with language embedding, for 'Baseline 1'.

used a popular Word Embedding Model (GloVe) [25], which maps each word into 300 dimensional vector space. For processing the attention from image and language data, we have adopted the 'Semantic Model' from [30], which trained an attention network with language embedding to influence the CNN feature maps. This process was done to find out the candidate region by weighting using probability distribution on all objects (found by Faster R-CNN) and re-weighting the winning candidate object with language embedding to train. We have utilized the final feature vector for our attention-based controller network as discussed in Section 3.4.

4.4 Model Parameters

We will discuss about the dimensions of each of the terminal input and outputs in detail. We have disussed the structure of attention model in Section 3.1. For FCVN, we pass the timestep image features using ResNet-18. To fit the image into the input of ResNet-18, we reshape the image into $224 \times 224 \times 3$ dimension. The model predicts output of 512 dimensions. Features from two consecutive frames add up a 1024 dimensional vector, which is further passed with fully connected layers toward predicting 7 dimensional. This is the mainstream network.

However, we have language feature vector as input for 'Baseline 1' and attention information for Our model. For that, we additionally, pass the 300 dimensional sentence embedding to a dense layer for making it till 64 dimension. later, we concatenate this vector with the mainstream FCVN's 64 dimensional output and pass to the further layers to generate the final 7 dimensional joint angles. However, for the case of attention, the number of features are 44. So the mainstream FCVN's 64 dimensional output is compressed to 44 and concatenated with the attention vector for passing to the next layers. Figure 8 and Figure 7 illustrates the discussed FCVN model configuration and parameters of 'Baseline 1' and Our approach, respectively.

4.5 Training and Evaluation Criteria

For training the 'Baseline 1' in Section 3.3 and Our model in Section 3.5, we used the 350 sub-sampled videos along with trajectory and language information. We pre-computed the sentence embeddings for training 'Baseline 1' and utilized the averaged (over all words) vector only once per video demonstration. We pre-computed the attention vector for training our model based on the model discussed in the earlier section and used the same vector multiple times per video demonstration. For both cases, the 'Control Value Updater Network' was trained based on the *Mean-Squared Error Loss* between the model prediction and actual (ground-truth) joint angles. However, for 'Baseline 2' in Section 3.5, we used the pre-trained model from [30] as it was trained on identical simulation and dataset we are using. On each model epoch, we evaluate the model capabilities via validating the performance on the validation split. The neural network models were designed using Python-based deep learning libraries: PyTorch[2] and Tensorflow[3]. We rely on the HRI Experiment to test the outcome, where we evaluate the outputs by real humans, as discussed in Section 6. The qualitative analysis for model training is demonstrated in the next section.

5 Quantitative Analysis

In this section, we interpret and compare the training loss of *'Baseline 1'* and Ours. Ass mentioned in the earlier section, for each demonstration, the loss is computed for each time step t based on the joint angles predicted by the 'Control Value Updater Network' and ground truth angles (by simulator). As both values are normalized to be in between 0 to 1, for visualization, we amplify the average loss by multiplying with 60^2. The overall loss per epoch (single iteration over all videos) is reported accordingly. In Figure 9, we can notice that the training loss of 'Baseline 1' is unstable with respect to the epochs, which indicates that the model can not generalize or converge with the given visual and language information. Compared to this, our training approach, as mentioned in Figure 10 relatively

[2] https://pytorch.org
[3] https://www.tensorflow.org

achieves better performance with lower training loss than the 'Baseline 1'. Also, the convergence is satisfactory and relatively less unstable. During the validation stages (testing error on unseen data), we can notice that our approach in Figure 12 gains a loss which is less than 500 in one stage, whereas, the 'Baseline 1' validation loss in Figure 11 can hardly achieve less than 600. From this analysis, we can claim that solely language embedding can not indicate a robot controller to set its goal; rather, assistance from the attention mechanism (as we did) can direct the robot more explicitly to focus on a specific object/region for manipulation. Hereby, attention is important in the process of designing a low-level controller. As we have only used the pre-trained model of 'Baseline 2', we have omitted the quantitative comparison. However, we have compared all three approaches on the test set as an HRI study (discussed in the next section).

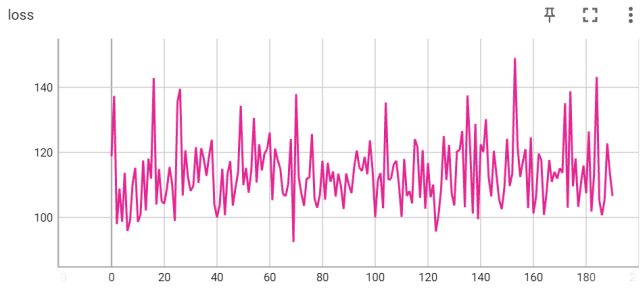


Fig. 9: Training Loss of 'Baseline 1' (no attention, only language features)

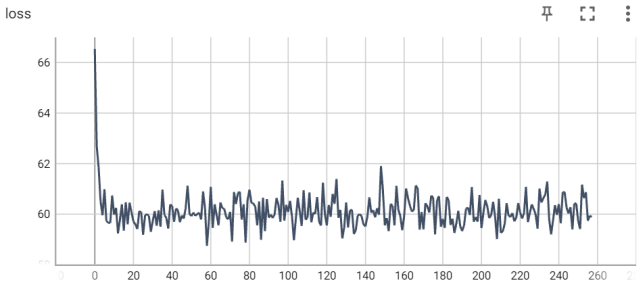


Fig. 10: Training Loss of Our Network (language-modulated attention)

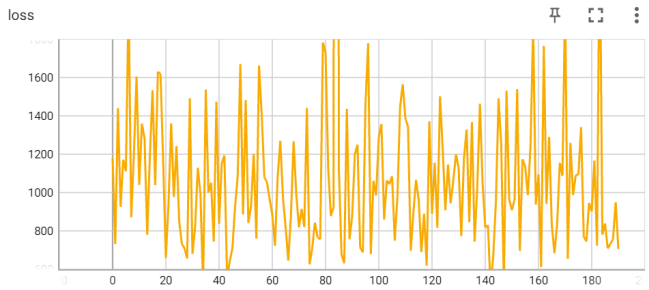


Fig. 11: Validation Loss of 'Baseline 1' (no attention, only language features)

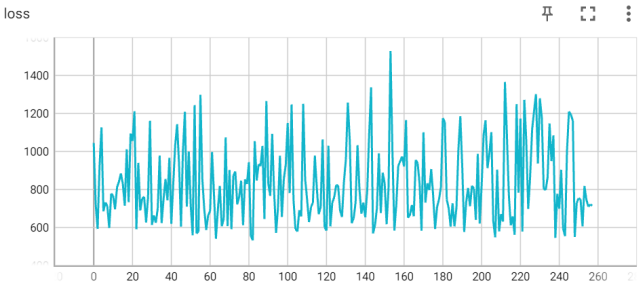


Fig. 12: Validation Loss of Our Network (language-modulated attention)

6 Qualitative Analysis: An HRI Study

To demonstrate the effectiveness of a low-level controller, a real-life evaluation is necessary, as discussed in the earlier works [4, 31]. This is due to the concern that the loss function we are evaluating might be convincing from the perspective of our trained model; however, in reality, the training might not be fruitful. For example, the controller might not generate sufficient angles to direct the robot to reach that specific location over time. To overcome this uncertainty, we have performed an HRI study to evaluate 4 contexts, i.e., 'Baseline 1', Our Approach, 'Baseline 2', and Ground Truth. The experiment design is discussed in the following subsection.

6.1 Experimental Design

In this section, we will explain the standard procedure/protocol we followed for conducting the experiment.

Hypothesis: *Attention mechanism can train low-level controllers to accomplish relatively complex tasks than training directly with language embedding.*

Subjects: 7 Male (age: 23-33), Engineering Graduate students with no prior expertise in Robotics.

Measurement Process: Each of the persons had to evaluate 10 demonstrations based on 4 following criteria based on the likert scale (0-5):

- **Accuracy:** Could the robot understand language content properly and reach the target object?
- **Closeness:** How close the robot could reach to the target?
- **Smoothness:** How smooth the trajectory was during the reach?
- **Overall Pick Approach:** Could the robot made satisfactory progress in picking the object?

We can consider the above criterions as independent variables, from which we analyze the dependent variable, i.e., Hypothesis's feasibility.

Prior to the survey, every candidate was given a 30 minute demonstration of the objective of their work. They were also trained how to accurately select the dependent variable measurements, followed by a brief introduction with the objects shapes and names. We have randomly selected 10 demonstrations out of 100 from the test set and used these to generate the robot manipulation for Baseline 1, 'Baseline 2', Ours, and Ground Truth. We kept Ground Truth as a evaluation approach to depict the human perspective and validating their measurements. The respective language text were embed with the video for better user experience. The videos of resulting demonstrations were randomized and each user did not know the category over 40 demonstrations. Figure 13 illustrates the survey process built with Python backend. After each video played, the user have to input their score for the four independent variable question we set. After each submission, the user information was saved into a file with their name as prefix as shown in Figure 15. We can see a person performing annotation in Figure 14. Here we note that, '_noatt' suffix means 'Baseline 1', '_att' is Our method, '_sim' is 'Baseline 2', and '_gt_new' is the Ground Truth.

6.2 Results and Discussion

After collecting survey result from 7 persons, we grouped and averaged the score for each approach (4 in total) on 4 independent variables. To measure the fluctuation among the evaluators' perception, we recorded standard deviation as well. Table 1 depicts the overall performance measurement based on likert scale for each approaches. From the result, we can notice an interesting phenomenon regarding Ground Truth, which is, not a perfect 5 for any independent variables as we initially expected. There can be one possible explanation regarding this case: the evaluators could not interpret the language clearly, which led them to score lower even if the correct object was picked. It can be clearly seen that the improvement of the visual controller after replacing language embedding with language-modulated attention mechanism. For all cases, our proposed controller outperforms 'Baseline 1'. The highest score of our approach was achieved in 'Closeness' variable, which implies that our approach was fairly close enough to reach the target. However, the performance of 'Baseline 2' exceeded ours in all aspects, also it was closer to the performance of ground truth. There can be two

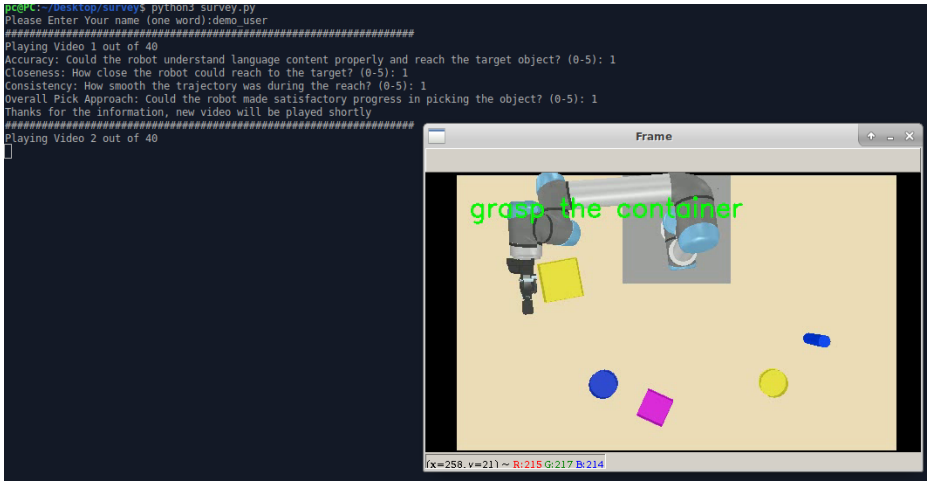


Fig. 13: The Terminal User Interface (UI) for recording the evaluations.

explanations regarding this. First of all, we used the pre-trained model of 'Baseline 2'. It can be possible that the 10 demonstrations we have used for evaluation could have been in the training set of 'Baseline 2'. Also, they utilized Dynamic Motion Primitive (DMP) [10], which is an established method for trajectory generation and robustly performs in given sequential environment information and target location over time. However according to several researches [7, 21, 19], DMP controller could fail if it was about to perform in uncontrolled environment, whereas our proposed low-level controller only relies on visual data, which might perform better for the real-world environments where the dynamics change frequently and having a complete information of the environment is fairly impossible as opposed to the simulator. Similarly, we notice that the 'Baseline 1' drastically fails in human evaluation. From Figure 16, we can also analyze that the robot is not producing any plausible motion that could reach the target. This issue can be connected with our low training examples, which clearly fails the joint angle prediction network to converge. However, we can conclude that if we train from scratch, even with low data, our language-modulated controller can generalize objects and produce relatively fair joint angles whereas only language embedding can not, perhaps need more data to learn how to predict the angles.

If we take a look at the Standard Deviation of the results, it can be noticed that, the scoring fluctuation was highest in Our method. The ground truth also showed this phenomenon close to us. This issue could be because of some confounding variables regarding human perception and judgment, which we leave for further research.

Although we have trained the visual controller solely from video data, we have noticed another phenomenon while executing the model prediction in simulator. When we analyze the result videos produced by our models and captured by the simulator camera that we used during training, we can see that the hand

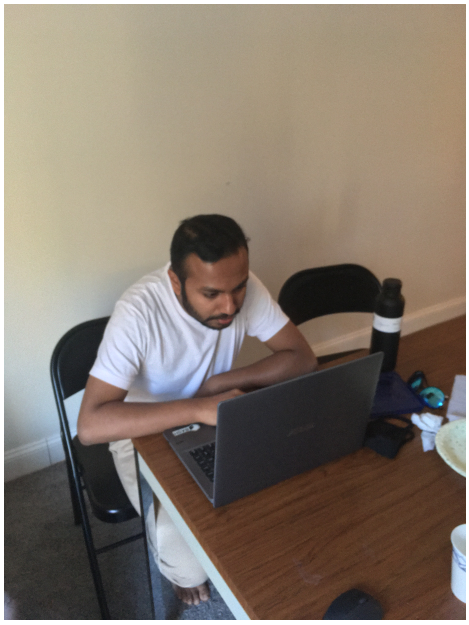


Fig. 14: An HRI experiment participant is working on evaluating the videos.

reaches very close to the object. However, in real simulator environment, we can notice that it does not go down to reach that object. Figure 17 illustrated a clearer example of this phenomenon. In the left side, we can see that the overall performance looks reasonable from the top-view, however, from another viewpoint, there is a large gap (right figure). We argue that, this is not the entire fault by the model, but the training data. If we analyze the ground truth of the same video in Figure 18, it can be noticed that when the hand reaches the object, there is no visual difference, which resulted our model to ignore this constraint. Because, our model takes two consecutive frame and calculates feature different, which was near to 0 in this specific case. The issue can be solved to incorporate similar concept like TCN [28], where can feed the model from multiple visual data and it can learn more latent constraints. Moreover, we can introduce depth information as another input feature to train our morel. Nevertheless, dependence on depth cameras in real-world scenarios might plummet the performance as studied in [12]. Hereby, we prefer multi-view input to extend our controller model. Here is the YouTube Playlist[4] of the 10 HRI experiment videos for each approach we discussed. We are planning to extend it with a large scale training.

[4] https://youtube.com/playlist?list=PL0BfYeyasxHMrNloFM9_LwoH7mCBirDdf

	A	B	C	D	E
1	video_id	accuracy	closeness	smooth	overall
2	Salman_video_data_sim/Q	5	5	5	5
3	Salman_video_data_sim/D	4	4	3	3
4	Salman_video_data_gt_nev	4	4	4	4
5	Salman_video_data_sim/8	3	4	3	4
6	Salman_video_data_gt_nev	3	4	3	4
7	Salman_video_data_sim/Q	4	3	4	3
8	Salman_video_data_gt_nev	5	5	5	5
9	Salman_video_data_att/48	0	0	1	0
10	Salman_video_data_noatt/	0	0	0	0
11	Salman_video_data_sim/1j	4	4	3	4

Fig. 15: A sample evaluation output done by evaluator named 'Salman'.

7 Conclusion and Future Possible Extensions

In this research, we explored designing a low-level controller to perform manipulation tasks that only depend on visual data. We explicitly used language information to specify the object properties, e.g., color, shape, to direct the controller. To accomplish a smoother training process, we used language-modulated attention to generate unique feature sets for the controller model convergence. To validate our experiment, we have compared our method with 'Baseline 1', which uses language embedding features directly to predict the joint angles and fails due to having noisy information. In that perspective, our proposed pipeline outperforms 'Baseline 1' in terms of quantitative (model training and validation loss) and qualitative analysis performed by real humans. Concerning our model's relatively lower performance compared to 'Baseline 2', which uses DMP to generate angles, we have several rooms for further extensions of this work. First of all, the model only performs for simulation data. However, a similar-scale experiment can be integrated for real-world data. As the deep learning

Table 1: Averaged outcome of each approach based on the likert scale.

	Accuracy	Closeness	Consistency	Overall
Baseline 1	0.55 ± 0.85	0.58 ± 0.85	0.53 ± 0.79	0.56 ± 0.86
Ours (attention)	2.19 ± 1.28	2.26 ± 1.38	2.21 ± 1.32	1.63 ± 1.65
Baseline 2	4.33 ± 1.10	4.51 ± 0.67	4.35 ± 0.79	4.45 ± 0.89
Ground Truth	4.40 ± 1.04	4.40 ± 0.98	4.45 ± 1.32	4.65 ± 1.64

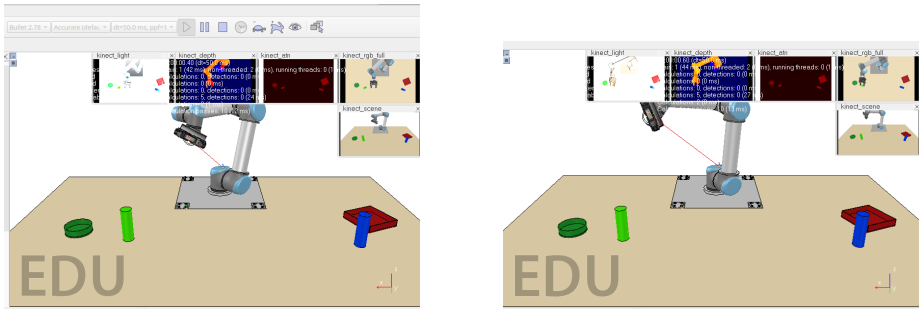


Fig. 16: An example of 'Baseline 1' outcome that never makes a valid trajectory.

models can be easily adaptable to new scenarios by transfer learning [37], we can extend existing work with this line of adaptation. The current network only relies on two consecutive image differences to predict trajectories. We argue that this can fail with several similar examples and may bias the network. Hereby, a memory-based module, e.g., Long Short-Term memory [16] can be integrated to keep track of the earlier joint angles prediction robustness to bias. Finally, we expect the low-level controller to be perturbations-proof for having a robust performance in uncertain environments. So, we can introduce adversaries (random or human-made) to train the model to avoid them while reaching the target.

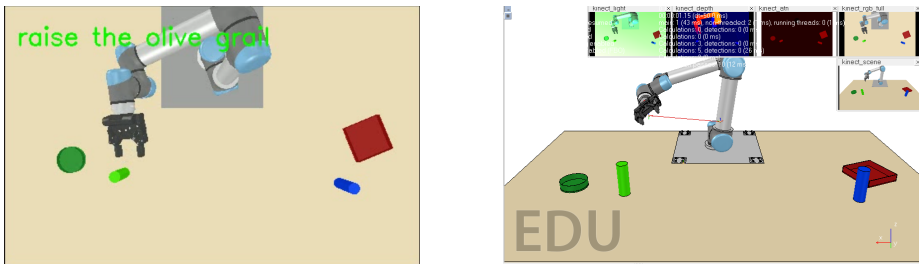


Fig. 17: An example of the outcome our method that reaches closer from top view, but does not get below due to the lack of training examples.

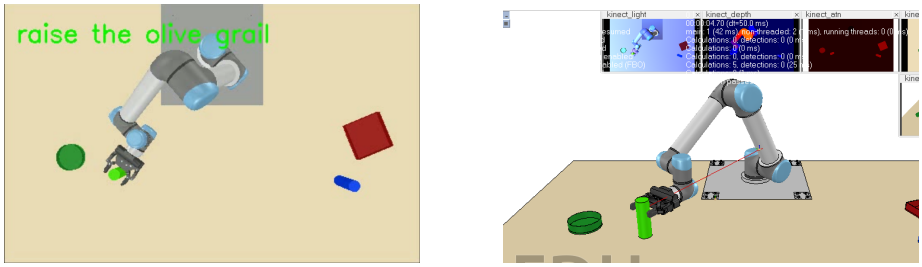


Fig. 18: The ground truth of the example which shows the ideal case.

References

1. Akgun, B., Cakmak, M., Yoo, J.W., Thomaz, A.L.: Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In: Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction. pp. 391–398 (2012)
2. Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L.: Bottom-up and top-down attention for image captioning and visual question answering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6077–6086 (2018)
3. Cadene, R., Ben-Younes, H., Cord, M., Thome, N.: Murel: Multimodal relational reasoning for visual question answering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1989–1998 (2019)
4. Choi, Y., Choi, M., Oh, M., Kim, S.: Service robots in hotels: understanding the service quality perceptions of human-robot interaction. Journal of Hospitality Marketing & Management **29**(6), 613–635 (2020)
5. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 4693–4700. IEEE (2018)
6. Gao, P., Jiang, Z., You, H., Lu, P., Hoi, S.C., Wang, X., Li, H.: Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6639–6648 (2019)
7. Giszter, S.F.: Motor primitives—new data and future questions. Current opinion in neurobiology **33**, 156–165 (2015)
8. Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Shi, X., Wang, C., Xie, J., Zha, S., et al.: Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. Journal of Machine Learning Research **21**(23), 1–7 (2020)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
10. Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., Schaal, S.: Dynamical movement primitives: learning attractor models for motor behaviors. Neural computation **25**(2), 328–373 (2013)
11. James, S., Freese, M., Davison, A.J.: Pyrep: Bringing v-rep to deep robot learning. arXiv preprint arXiv:1906.11176 (2019)
12. Jing, C., Potgieter, J., Noble, F., Wang, R.: A comparison and analysis of rgb-d cameras' depth performance for robotics application. In: 2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP). pp. 1–6. IEEE (2017)
13. Karthik, R., Hariharan, M., Anand, S., Mathikshara, P., Johnson, A., Menaka, R.: Attention embedded residual cnn for disease detection in tomato leaves. Applied Soft Computing **86**, 105933 (2020)
14. Kebria, P.M., Al-Wais, S., Abdi, H., Nahavandi, S.: Kinematic and dynamic modelling of ur5 manipulator. In: 2016 IEEE international conference on systems, man, and cybernetics (SMC). pp. 004229–004234. IEEE (2016)
15. Lambert, N.O., Drew, D.S., Yaconelli, J., Levine, S., Calandra, R., Pister, K.S.: Low-level control of a quadrotor with deep model-based reinforcement learning. IEEE Robotics and Automation Letters **4**(4), 4224–4230 (2019)

16. Li, J., Tan, H., Bansal, M.: Improving cross-modal alignment in vision language navigation via syntactic information. arXiv preprint arXiv:2104.09580 (2021)
17. Li, W., Zhu, X., Gong, S.: Harmonious attention network for person re-identification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2285–2294 (2018)
18. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
19. Liu, C., Geng, W., Liu, M., Chen, Q.: Workspace trajectory generation method for humanoid adaptive walking with dynamic motion primitives. IEEE Access **8**, 54652–54662 (2020)
20. Liu, J., Wang, G., Hu, P., Duan, L.Y., Kot, A.C.: Global context-aware attention lstm networks for 3d action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1647–1656 (2017)
21. Liu, Y., Romeres, D., Jha, D.K., Nikovski, D.: Understanding multi-modal perception using behavioral cloning for peg-in-a-hole insertion tasks. arXiv preprint arXiv:2007.11646 (2020)
22. Molchanov, A., Chen, T., Hönig, W., Preiss, J.A., Ayanian, N., Sukhatme, G.S.: Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors. arXiv preprint arXiv:1903.04628 (2019)
23. Nah, M.C., Krotov, A., Russo, M., Sternad, D., Hogan, N.: Dynamic primitives facilitate manipulating a whip. In: 2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob). pp. 685–691. IEEE (2020)
24. Otter, D.W., Medina, J.R., Kalita, J.K.: A survey of the usages of deep learning for natural language processing. IEEE Transactions on Neural Networks and Learning Systems (2020)
25. Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1162, https://www.aclweb.org/anthology/D14-1162
26. Petersen, K.H., Napp, N., Stuart-Smith, R., Rus, D., Kovac, M.: A review of collective robotic construction. Science Robotics **4**(28) (2019)
27. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497 (2015)
28. Sermanet, P., Lynch, C., Hsu, J., Levine, S.: Time-contrastive networks: Self-supervised learning from multi-view observation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 486–487. IEEE (2017)
29. Sharma, P., Pathak, D., Gupta, A.: Third-person visual imitation learning via decoupled hierarchical controller. arXiv preprint arXiv:1911.09676 (2019)
30. Stepputtis, S., Campbell, J., Phielipp, M., Lee, S., Baral, C., Amor, H.B.: Language-conditioned imitation learning for robot manipulation tasks. arXiv preprint arXiv:2010.12083 (2020)
31. Taheri, A., Meghdari, A., Alemi, M., Pouretemad, H.: Human–robot interaction in autism treatment: a case study on three pairs of autistic children as twins, siblings, and classmates. International Journal of Social Robotics **10**(1), 93–113 (2018)
32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. arXiv preprint arXiv:1706.03762 (2017)

33. Villani, V., Pini, F., Leali, F., Secchi, C., Fantuzzi, C.: Survey on human-robot interaction for robot programming in industrial applications. IFAC-PapersOnLine **51**(11), 66–71 (2018)
34. Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X.: Residual attention network for image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3156–3164 (2017)
35. Wang, Y., Sheng, Y., Wang, J., Zhang, W.: Optimal collision-free robot trajectory generation based on time series prediction of human motion. IEEE Robotics and Automation Letters **3**(1), 226–233 (2017)
36. Zhu, F., Zhu, Y., Chang, X., Liang, X.: Vision-language navigation with self-supervised auxiliary reasoning tasks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10012–10022 (2020)
37. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proceedings of the IEEE **109**(1), 43–76 (2020)