Integration of Multimodal Inputs and Interaction Interfaces for Generating Reliable Human-Robot Collaborative Task Configurations

Shuvo Kumar Paul*

University of Nevada, Reno, NV, USA

Pourya Hoseini, Arjun Vettath Gopinath, Mircea Nicolescu, Monica Nicolescu†

University of Nevada, Reno, NV, USA

Abstract

As robots become more ubiquitous in our daily life, designing natural, easy to use, and meaningful interaction interfaces relevant to robotic tasks is vitally important as not only it can enhance user experience, but also can increase task reliability by proving supplementary information. This paper presents a flexible framework that integrates two natural interaction interfaces: speech, and pointing gesture with the sensor input streams to generate reliable task configurations for human-robot collaborative environment. The proposed framework takes the RGB image as input to detect the objects present in the scene and to recognize the pointing gestures, and it computes the corresponding pointing direction in the 2D image frame to infer the target object in the scene. At the same time, verbal instruction is received from the audio input which is then converted to text to either be fed into the proposed neural model or to compare against predefined grammar rules to extract relevant task parameters. All this information is used to resolve any missing or ambiguous task parameters. Structured task configurations are formed for the desired human-robot collaborative tasks. The proposed framework shows very promising results in integrating the relevant task parameters for the intended robotic tasks in different real-world interaction scenarios.

Key Words: Interaction interface; gesture recognition; multimodal inputs; input integration; robotics; human-robot interaction; natural language processing.

1 Introduction

Recent technological advances in automation, engineering, and artificial intelligence have provided the impetus for the rapidly accelerating robotics revolution. While in the past few decades the number of industrial robots skyrocketed, only recently robots are becoming more and more inclusive in our daily life. The rapid advances in AI and robotics have significantly shifted the focus of robotics research from industrial robotics to service robots; these robots lend a helping hand for tasks like cooking, cleaning, assistance, companionship, education, and so on. In contrast to the industrial robots, which repeatedly perform a specific task, a service robot is expected to interact with humans while performing tasks, and an ideal interaction should replicate a human-to-human interaction.

Designing interaction interfaces that are more intuitive and instinctive are prerequisites for ensuring the ease of use and inclusion of robots in our daily life. However, to sustain this inclusion, robots would need to build and maintain the trust of the user, particularly the trust that the robot can reliably perform a designated task. This warrants Human Robot Interaction (HRI) framework that not only can establish a natural interaction interface, but also can supplement the robotic task execution with additional data to make it more reliable.

Robotic entities are set to become an essential part of modern society and have the potential to shape our social experience. However, the inclusion of robots in our daily life will be dictated by one key factor: our trust in robots, specifically, the confidence of the human user that a robotic agent can accurately perform a task. To build and maintain this trust, it needs to be made sure that the robots can consistently execute the tasks in a proper manner.

Proper execution of robotic tasks requires instructions containing a set of parameters that defines a task configuration. We have identified two essential properties of a **complete** task configuration:

1. The task configuration needs to have all the relevant task parameter information for executing an intended task. Depending on the task, the task configurations can have different task parameters e.g. navigational task may only require the direction, while an assisting task may involve

*shuvo.k.paul@nevada.unr.edu

†hoseini,avettathgopinath@nevada.unr.edu;mircea,monica@unr.edu

knowledge of object(s), their attributes and locations in the environment, order of task information, etc. The relevant task parameters can be extracted from the robot sensors and filtered to form a set of structured instructions that can be correctly interpreted by the robotic entity.

2. The task configurations should be reliable enough for intended task execution. The robot may follow the instruction accordingly, but the instruction itself may contain erroneous task parameters; for instance, the robot may start moving toward its left, while the intended instruction was to turn left. This can result either from the noisy sensory data or ambiguities during task parameters extraction. To address this, the instructions need to be cross-validated to ascertain the intent of the user which can be done by integrating sensory information with different human robot interaction interfaces.

The first property is sufficient for executing a robotic task, however, the second property provides more assurance for the validity of the task configurations. Subsequently, if required task parameters can not be inferred from sensor input streams, then the interaction interfaces can be used to determine and/or verify the task parameters and vice versa. Although, achieving the second property may not be viable for every task configuration, verifying instructions from multiple input streams and interaction interfaces should definitely be one of the objectives in collaborative HRI design as it will help generate more reliable task configurations. More reliable task configuration would aid in proper task completion, which would help build more trust in robots as well.

Natural human interactions mostly involve gestures, speech, and facial expressions. Amongst these interaction interfaces, gestures and particularly pointing gestures are probably the most natural for humans and can serve as an effective device to convey a simple message or a command to the robot. Unlike speech, pointing gestures are not suitable for conveying sophisticated information; however, it provides with a more natural interface for simpler instructions that can be relayed even in noisy environments. Moreover, it provides the possibility of specifying objects and their locations intuitively and can be used as simple but meaningful commands. In addition, distinct human gestures represent specific information that can be used to convey the general intent of the user. This inferred intent information then can be compared or matched to predefined gesture configuration to provide additional information or appropriate command for the robot to execute certain tasks.

At the same time, speech can be used as a natural medium to express intricate commands which can then be used to effectively apply modern Natural Language Processing (NLP) techniques to parse and extract language information. Proper utilization of natural language can permit for a more intrinsic and faster dialogue between human and the robot. However, natural language communication needs to be translated into a formal language which the robot can process and make a decision upon. Subsequently, the robot needs to formulate a structured message for the successive communication which needs to be transformed into natural language to allow the user for easier comprehension.

Gestures information can simultaneously be integrated with verbal communication (speech recognition) to provide auxiliary information to further disambiguate natural language commands.

In our work, we focused on integrating two interaction modalities: 1) pointing gesture that can be used to direct the attention of the interacting robot toward an object or a certain location in the scene, and 2) verbal commands which are translated into simpler formal languages that are more interpretable for the robot; both of these are natural interaction interface for humans. Additional data containing the information of detected objects in the scene is also passed to the system to find and locate the **Object of Interest (OOI)**. **OOI** is the object that is requested by the human user to be manipulated by the robotic entity.

We propose a simple and reliable HRI framework that extracts a set of information from verbal instructions retrieved from the audio input, detects pointing gestures, estimates the general pointing direction and the object being pointed at from individual RGB frames, and finally, assigns them to the appropriate task parameters which then can be used to formulate structured instructions. We have focused primarily on simpler but more common collaborative task instructions targeting scenarios where a user provides navigational instructions e.g. "go to your right", "go there", etc. or commands that require object manipulation e.g. "give me the bowl", "bring that red book", etc. The task configuration has the following parameters: $\underset{\sim}{a}$: action, $\underset{\sim}{o}$: general object, $\underset{\sim}{r}$: object attribute, $\underset{\sim}{p}$: position of the object in the scene, $\underset{\sim}{d}$: general pointed direction, $\underset{\sim}{o_p}$: estimated pointed object. These extracted parameters can be used to generate a formal task description targeting a specified goal. Our approach relies on Google's speech to text API and the skeletal joint points extracted by AlphaPose [5, 15] from the RGB image to infer gesture.

The main contributions of this paper are of three folds:

1. We present a gesture recognition system that estimates whether the user is performing a pointing gesture and the pointing direction.
2. We leveraged a verbal command comprehension system to extract certain information from a user command relevant to specific robotic tasks by a) deep multi-task learning model and b) matching to pre-defined language patterns.
3. We have implemented an object detection and pose estimation system using template matching technique suitable for fast prototyping and demonstrated how the pointing gesture framework coupled with the extracted information from verbal command can be used to generate a list of task configurations corresponding to a sequence of tasks.

This paper is outlined as follows: in the next section, we provide a brief overview of previous work on gesture recognition techniques and natural language understanding in

HRI design. Next, we describe the methodology of our work in detail. The following chapters include our evaluation including experimental results and observations. Finally, we conclude this paper by summarizing our work.

2 Literature Review

A large body of work has been published in the area of HRI that focuses on pointing gestures and natural language understanding. In the following sub-sections, some of the previous works have been discussed.

2.1 Pointing Gesture Recognition

Early pointing gesture interfaces were built with the help of wearable devices e.g. glove-based devices [25, 8]; Dipietro et al. [2] surveyed Data-GLove like systems and their application for gesture recognition. Kahn et al. [10, 11] introduced Perseus architecture which operated on several feature maps (intensity, edge, motion, disparity, color) to locate pointed objects by interpreting the pointing gestures. With the continuing advances in computer vision, gesture recognition research shifted more toward vision-based methods. Kadobayashi et al. presented VisTA-Walk, a gesture interpreter which could only infer left or right; it uses the output recognition result of Pfinder [32] which employed a multiclass statistical color and shape model to derive a 2D representation of head and hands from a wide range of viewing conditions. With the introduction of stereo cameras, multi-cameras, time-of-flight (TOF) cameras, or depth cameras like Kinect, Intel RealSense, etc. researchers were able to come up with different approaches for solving pointing gesture detection. [30, 12] used multi-camera setup while [3] used TOF camera to segment body, localize forearm and elbow for gesture detection, and subsequently used Gaussian Process Regression for pointing direction estimation.

Different Hidden Markov Model (HMM) have been used to detect pointing gestures. Wilson et al. [31] proposed parametric HMM for recognition, representation, and interpretation of parameterized gestures, such as pointing gestures. Jojic et al. [9] used only the dense disparity maps for gesture detection, while Nickel et al. [20] calculated the dense disparity maps to track the positions of a person's face and hands together with an HMM-based approach for detecting pointing gestures. Park et al. [21] applied Cascade HMM and particle filters but depended on a large number of HMM states for accurate gesture recognition which required large amounts of training data and thus, incurred higher processing time.

Richarz et al. [27] used Gabor wavelets to extract features and multilayer perceptron to approximate pointing direction estimator, but it was sensitive to pose variations. Pateraki et al. [22] exploited the prior information of the location of possible pointed targets and used the Dempster–Shafer theory of evidence to fuse the estimated head pose and hand pointing orientation information to locate the pointed target.

Rautaray et al [26] extensively reviewed hand gesture recognition and pointed out the well known limitations of the leading technologies related to the field.

In our work, we used the estimated (pixel) location of the forearm joints (elbow and wrist) of the user to 1) determine whether the person is performing the pointing gestures and 2) infer the general direction the user is pointing e.g. left, right, straight and estimate the pointing direction by computing the line that goes through the arm joints.

2.2 Natural Language Understanding in HRI

Natural language based interaction has been explored in various human-robot interaction tasks that include instructing the robot with direction for navigation, commanding for performing certain robotic tasks, specifying the object to manipulate, etc. Natural language understanding is also used as a modality along with other sensory information like vision to disambiguate human instructions or the scene configuration in general.

Kollar et al. [13] presented a system that infers the probable path for an agent by taking the environmental geometry and the detected objects as inputs along with the extracted sequence of spatial description clauses from the linguistic information. Matuszek et al. [18] investigated statistical machine translation techniques to follow natural language route instructions within a tractable manner. Macmohan et al. [17] introduced MARCO, an agent that infers implicit actions from knowledge of linguistic conditional phrases and spatial action information along with environmental configuration. This method performs the explicit, implicit actions required to reach the instructed state, and subsequently executes exploratory actions to learn about the environment. In [29] an approach was introduced to automatically generate a probabilistic graphical model with respect to the hierarchical and compositional semantic structure of natural language navigation or mobile manipulation commands.

Dzifcak et al. [4] proposed an integrated robotic architecture that translates natural language instructions incrementally and simultaneously generating logical goal representation and action language, which can be further analyzed to measure the achievability of the goal as well as to create new action scripts targeting specified goals. Kuo et al. [14] demonstrated that the combination of a hierarchical recurrent network with a sampling-based planner can be utilized to generate a model that learns to understand a sequence of natural language commands in a continuous configuration space. The use of spatial relationships to establish natural communication mechanism between humans and robots was investigated in [28]; a multimodal robotic interface comprising of linguistic spatial descriptions and other spatial information extracted from an evidence grid map was used to show how this information can be used in a natural, human-robot dialog. [1] described a robotic architecture featuring a planner that utilized discovered information by learning the pre and post conditions of previously unknown action sequences from natural language construction.

We have leveraged the recent natural language processing methods in our work to retrieve text from speech, generate grammatical patterns to match, and extract relevant command information from the user.

3 Methodology

The system receives gesture information and verbal communication from an RGB image, and an audio input stream respectively. The speech in audio is converted to text to further extract the action instruction and the object information corresponding to the robotic tasks. RGB image is used for pointing gesture recognition, pointed direction estimation along with object detection, and pointed object prediction. The overview of the system architecture is illustrated in the Figure 1; the green boxes indicate the extracted task parameters.

3.1 Information Extraction from Verbal Commands

In the course of a typical human-to-human collaborative interaction, the instructions that the participants interchange usually consists of a set of particular information; this includes the action to be performed, the object of interest, direction information for navigation, location of interest in the scene, etc. Furthermore, humans generally describe an object in terms of a general color, pattern, shape, size, and the relative position to disambiguate [24], e.g., "bring that red shirt", "The book on the left", "take the small box", etc. This information defines certain parameters of a task. We have developed two techniques: 1) neural network model, 2) natural language pattern matching, to extract the task parameters from verbal commands. These two approaches are outlined in the following sub sections.

3.1.1 Neural Network Model. For our work, we generated a dataset for collaborative robotic commands. Subsequently, we considered 8 different architectures for training our model and found single layer Bi-directional Long Short Term Memory (Bi-LSTM) based model to be optimum. As our goal was to extract multiple task parameters from the verbal commands, we formulated deep multi-task learning model. Multi task learning (MTL) is a sub-division of machine learning, where multiple tasks are jointly learned by a shared model. Deep multi task learning tries to produce a generalized representation that are powerful enough to be shared across multiple tasks; here, each task denotes a multi-class classification.

Dataset: We generated a dataset of commands where each of the commands contains action information, and information about one or more of the following three things: object name, object color, and object size. The dataset contains 60769 samples, each of which has four labels.

Model Architecture: The model contains three neural layers: an embedding layer, a Bi-LSTM layer, and a fully connected layer. Let's assume the vocabulary length of the dataset is $\mathtt{V}$, then every word is represented with a one-hot encoding of size $W \in \mathbb{R}^{1\times \mathtt{V}}$. The input sequences or sentences each contains n elements or words. These inputs are fed into an embedding layer $\mathscr{E}$.

An embedding is a mapping of a discrete or categorical values to a vector of continuous numbers. A neural netwrok embedding provides a low-dimensional, learned continuous vector representations of these discrete variables. The key advantage of neural network embeddings is that they can reduce the dimensionality of categorical variables to represent them in the transformed space. $\mathscr{E} \in \mathbb{R}^{\mathtt{V}\times \mathtt{d}}$, where $\mathtt{d} << \mathtt{V}$ denotes the lower dimensional embedding vector; this lower dimensional vector is then passed to the Bi-LSTM layer.

LSTMs [7] are a particular form of Recurrent Neural Network (RNN). LSTM (or Bi-LSTM) are ideal for sequential data such as text, speech, video, audio, etc. Our key motivation for choosing Bi-LSTM is that it can make use of both the past and future context information of a sentence, and can learn long-term temporal activities as well as avoid exploding or vanishing gradient that the traditional RNN suffers from during the back propagation optimization. In Figure 2, f_i and b_i denote forward and backward LSTM respectively.

The output of the Bi-LSTM cells are concatenated and fed into the four fully connected (FCN) layers. Finally, the output of the FCN layers goes through softmax activation to classify four task parameters. For each classifier, we measured the Cross Entropy loss $\mathscr{L}_c$ and used the mean of these losses $\mathscr{L}_m = \frac{1}{4}\Sigma_{c=1}^{4}\mathscr{L}_c$ to update our model.

3.1.2 Natural Language Pattern Matching. A set of language patterns can provide sufficient information about the parameters of collaborative interaction while excluding other verbal communications that may not contain a command. The language patterns are arranged in Table 1. These patterns are represented in terms of regular expressions with different language elements as terms. The angled brackets(<>) encloses each term. The terms with capital letters indicate different language elements, while the exact words are enclosed within quotes. An array of words, surrounded by square brackets e.g. [”right”, ”left”, ”front”, ”back”], specifies only one of them needs to be present. The symbols after each term e.g. +, ?, * are the quantifiers that represent how many times the preceding term needs to be matched; + indicates the term needs to be matched 1 or more times, * means 0 or more, ? means 0 or 1, and if there are no quantifiers then the preceding term needs to be matched exactly once.

The **Action** pattern extracts the general task needed to be executed; simultaneously, it can distinguish between a traditional instructive command and a non-instructive verbal communication. The **Object** pattern is almost similar to **Action** pattern except for that it needs to match with a *NOUN* at the end which would contain the name of the object. The **Attribute** pattern identifies certain visual characteristics of the object and

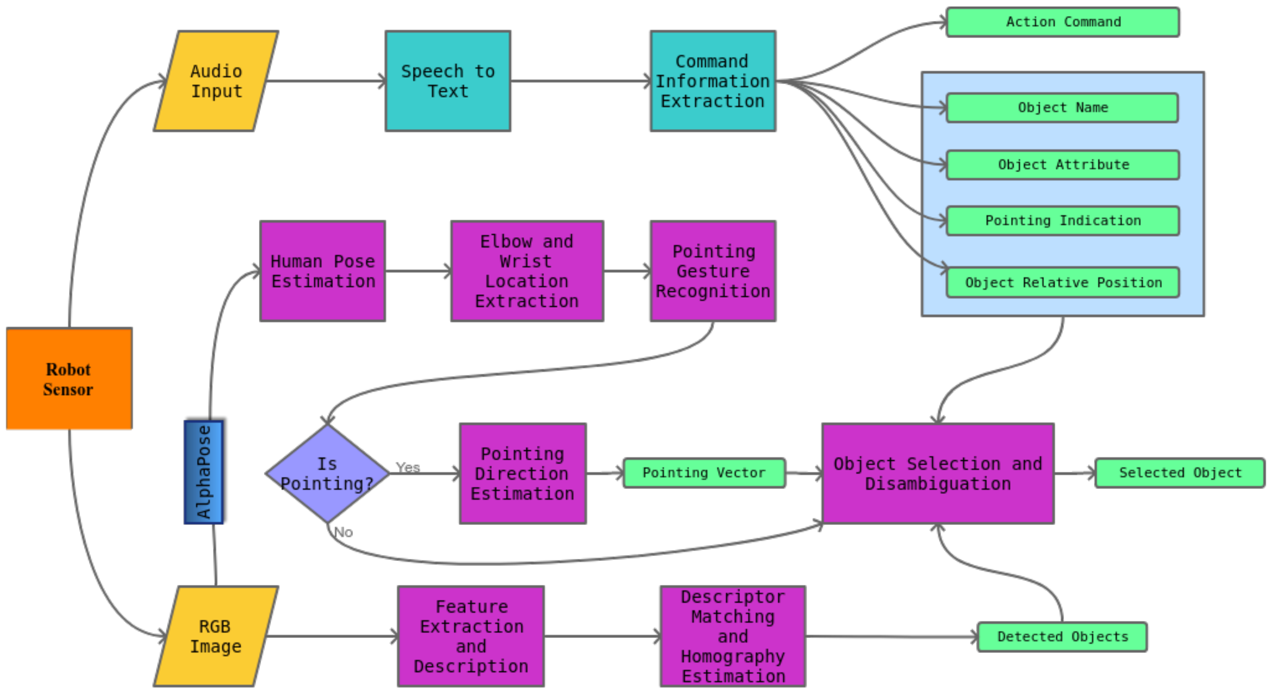

Audio Input
Speech to Text
Command Information Extraction
Action Command
Object Name
Object Attribute
Pointing Indication
Object Relative Position
Human Pose Estimation
Elbow and Wrist Location Extraction
Pointing Gesture Recognition
Robot Sensor
AlphaPose
Is Pointing?
Yes
No
Pointing Direction Estimation
Pointing Vector
Object Selection and Disambiguation
Selected Object
RGB Image
Feature Extraction and Description
Descriptor Matching and Homography Estimation
Detected Objects

Figure 1: System architecture

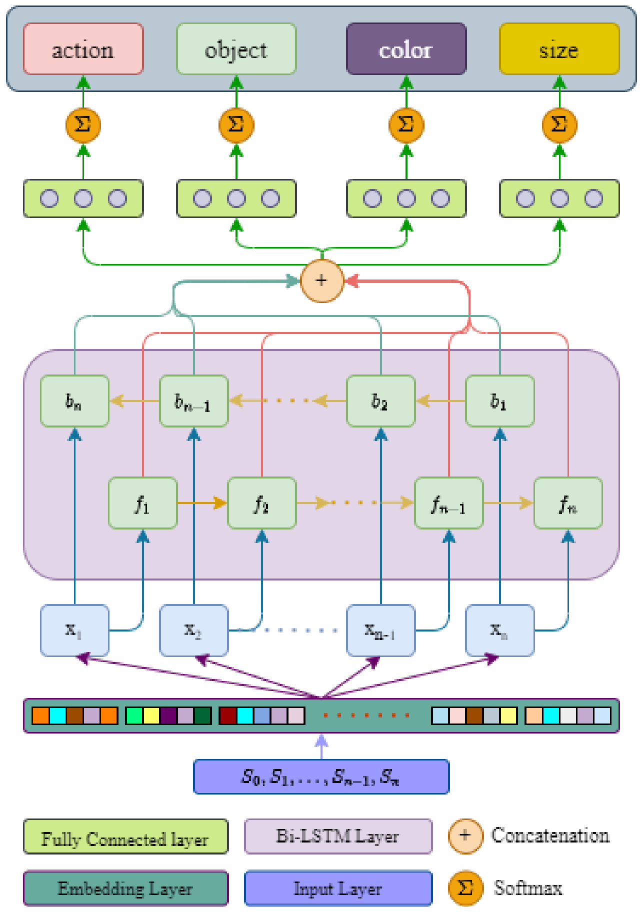


Figure 2: NN model for parameter extraction from verbal commands.

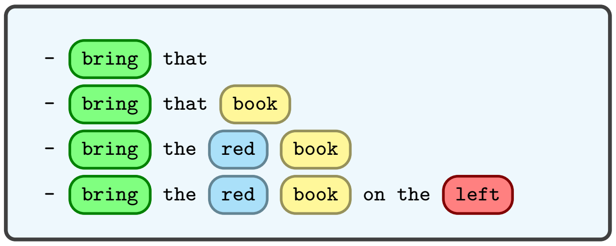
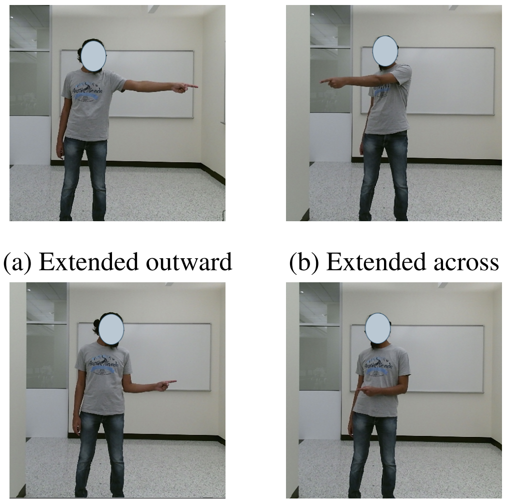


Figure 3: Green box represents the task action; red box indicates the location of the object in the scene; yellow and blue boxes specify the object of interest and the corresponding attributes

the **Position** pattern determines the general location, both of which can be utilized as identifiers/specifiers to disambiguate objects that fall into similar categories. Figure 3 presents a set of sample instructions iterated with additional information and illustrates the retrieved relevant information as well.



(a) Extended outward (b) Extended across

(c) Bent outward (d) Bent across

Figure 4: Gesture categories

3.2 Pointing Gesture Recognition

AlphaPose [5] was used to extract the skeletal joint locations to predict the pointing gestures and the general pointing direction. For simplicity, we assumed the user is using one hand at a time for pointing. Park et al. [21] categorized the pointing gestures into large and small pointing gestures, which we labeled them as extended (Figure 4(a, b)) and bent (Figure 4(c, d)) arm gestures. Additionally, for the pointing gesture, the forearm's relative direction with respect to the body can be generalized in three categories: across (Figure 4(b, d)), outward (Figure 4(a, c)), and straight (Figure 5(b)).

For across and outward pointing gestures, we can simply measure the angle θ_a (Figure 5(a)) of each forearm with respect to a vertical line and compare it to some smaller angle threshold θ_t to determine whether the user is performing the pointing gesture or not. In other words, if the user is not pointing (Figure 5(b)) then the forearm would produce a smaller angle compared to when the user is pointing. However, if the user points straight with respect to the camera (robot's vision) (Figure 5(c)), the angle would be close to 0 and to address this, we measured the ratio of the lengths of the forearms ρ_a; intuitively, if the user is not pointing, the lengths of the detected forearms should be virtually the same (Figure 5(c)), but if one of the forearm's length is significantly shorter than the other, it can be assumed that the user is pointing straight (or its proximity) toward the camera using that corresponding arm (Figure 5(b)). Furthermore, the general direction d of pointing was determined from the relative position of the wrist and the elbow of the pointing arm to supplement navigational command.

3.2.1 θ_a Calculation from Wrist and Elbow Location.

From the extracted skeletal joints' locations, only the following joints were needed: left elbow, left wrist, right elbow, and right wrist. This means even if some body part is occluded, our approach should still work as long as pointing hand's joints are detected. Let us define the skeletal joint coordinate of the elbow as (x_1, y_1), the wrist as (x_2, y_2); the pointing 2D vector centered at the origin can be defined as $\vec{a} = (x_2 - x_1, y_2 - y_1$ and the vertical vector to compare with is set to $\vec{v} = (0, 1)$. The pointing angle θ_a is measured using equation 1.

$$\theta_a = \cos^{-1} \frac{\vec{a} \cdot \vec{v}}{|a||v|} \tag{1}$$

If $\theta_a > \theta_t$, then we consider the corresponding forearm is performing the pointing gesture and subsequently, compare the x coordinate of the wrist and the elbow to further detect the general pointing direction towards the left or right of the scene (across or away with respect to the body). Next, the ratio of the length of the forearms $\rho_a = \frac{\text{Length of the arm of interest}}{\text{Length of the other arm}}$ is compared against a ratio ρ_t to further decide whether the user is pointing straight. We set 0.8 as the value of ρ_t and 15° for θ_t.

3.3 Object Detection

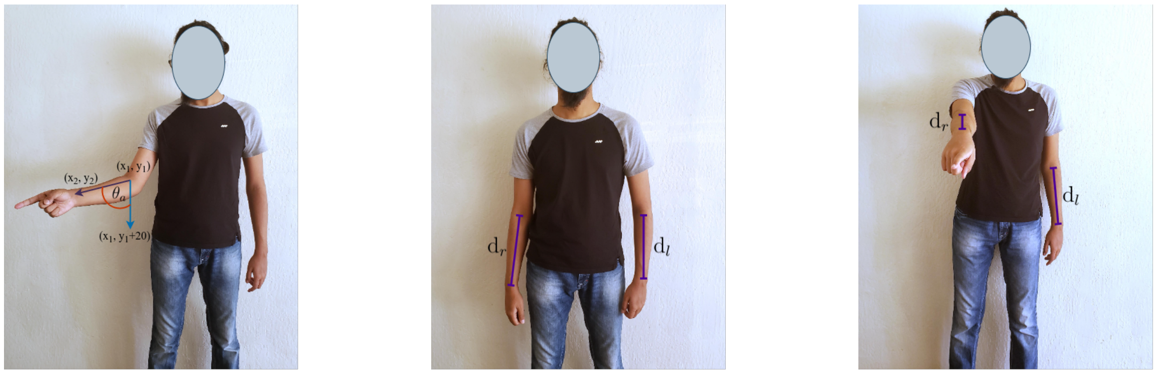
Our object detection module consists of two phases: (i) feature extraction and matching, and (ii) homography estimation and perspective transformation. In the following sections, we will focus on the detailed description of the aforementioned steps.

3.3.1 Feature Extraction and Matching.

Our object detection starts with extracting features from the images of the planar objects and then matching them with the features found in the images acquired from the camera. Image features are patterns in images based on which we can describe the image. A feature detecting algorithm takes an image and returns the locations of these patterns - they can be edges, corners or interest points, blobs or regions of interest points, ridges, etc. This feature information then needs to be transformed into a vector

Table 1: Grammatical patterns for information extraction

Information Type	Grammatical Pattern	Example
Action	<VERB><"me">?<DET[‡] >?<["this", "that"]>?<ADJ[§]>*<>?	bring me that; give me that book;bring it here
Object	<VERB><"me">?<DET>?<["this", "that"]>?<ADJ>*<NOUN>	take the red cup; bring me that dress
Attribute	<ADJ>+<NOUN>	red dress; large blue bowl
Position	<["right", "left", "front", "back"]>	the box on the left



(a) (b) (c)

Figure 5: (a) Generated angle θ_a , (b) length of forearms $\mathrm{d_l}, \mathrm{d_r}$when not pointing, and (c) straight

space using a feature descriptor, so that it gives us the possibility to execute numerical operations on them. A feature descriptor encodes these patterns into a series of numerical values that can be used to match, compare, and differentiate one feature to another; for example, we can use these feature vectors to find the similarities in different images to detect objects as well as distinguish each object in the scene. Ideally, this information should be invariant to image transformations e.g. change in illumination of the scene, different degrees of image blur and compression, variance in viewpoint, etc. However, every feature detector and descriptor is unique and can be tolerant to certain image transformations and to certain degrees. We selected SIFT [16] as both the feature detector and descriptor for our work as it provides reliable detection with adequate speed as reported in [23].

Once the features are extracted and transformed into vectors, we compare the features to determine the presence of an object in the scene. Usually, the Nearest Neighbor algorithm is used to find matches, however, finding the nearest neighbor matches within high dimensional data is computationally expensive, and with more objects introduced it can affect the process of updating the pose in real-time. To counter this issue to some extent, we used the FLANN [19] implementation of KD-Tree Nearest Neighbor Search, which is an approximation of the K-Nearest Neighbor algorithm that is optimized for high dimensional features. Finally, if we have more than ten matches, we presume the object is present in the scene.

[‡]DET refers to the determinant i.e. "the"
[§]ADJ refers to adjective part of speech e.g. small, red, etc.

3.3.2 Homography Estimation and Perspective Transformation. A homography is an invertible mapping of points and lines on the projective plane that describes a 2D planar projective transformation (Figure 6) that can be estimated from a given pair of images. In simple terms, a homography is a matrix that maps a set of points in one image to the corresponding set of points in another image. We can use a homography matrix **H** to find the corresponding points using equation 2 and 3, which defines the relation of projected point $(x^{'}, y^{'})$ (Figure 6) on the rotated plane to the reference point (x, y).

A 2D point (x, y) in an image can be represented as a 3D vector $(x, y, 1)$ which is called the homogeneous representation of a point that lies on the reference plane or image of the planar object. In equation (2), **H** represents the homography matrix and $[x\ y\ 1]^T$ is the homogeneous representation of the reference point (x, y) and we can use the values of a, b, c to estimate the projected point $(x^{'}, y^{'})$ in equation (3).

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

$$\begin{cases} x^{'} = \dfrac{a}{c} \\ y^{'} = \dfrac{b}{c} \end{cases} \tag{3}$$

We estimate the homography using the matches found from the nearest neighbor search as input; often these matches can have completely false correspondences, meaning they

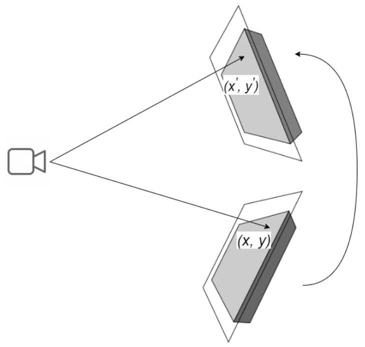


Figure 6: Object in different orientations from the camera

don't correspond to the same real-world feature at all which can be a problem in estimating the homography. So, we chose RANSAC [6] to robustly estimate the homography by considering only inlier matches as it tries to estimate the underlying model parameters and detect outliers by generating candidate solutions through random sampling using a minimum number of observations.

While the other techniques use as much data as possible to find the model parameters and then pruning the outliers, RANSAC uses the smallest set of data point possible to estimate the model, thus making it faster and more efficient than the conventional solutions. This estimated homography can also be effectively used for the planar pose estimation of textured objects [23].

3.4 OOI Estimation from Pointing Gesture

For each detected object, the bounding box can be defined as a list of four 2D line segments $BB = [s_1, s_2, s_3, s_4]$; s_i is defined by the following parametric equation:

$$s_i = (a_i, tb_i) = \left(V_i, \begin{cases} t(V_{i+1} - V_i) & \text{if } i < 4, \\ t(V_1 - V_i) & \text{else} \end{cases} \right) \tag{4}$$

where, V_i represents the $i^{th}(1 \le i \le 4)$ vertex of the quadrangle bounding box, $0 \le t_i \le 1$, and the value of t_i determines the location of a point on the segment; if $t_i = 0$ then it's the initial point and if it's equal to 1 then it's the final point in the segment (Figure 7). Additionally, the center of each detected object is computed by taking the average of the four vertices.

Similarly to equation 4, we can estimate the pointing direction by computing a 2D line from the pixel location of the arm joints (equation 5).

$$l_p = ((x_1, y_1), t(x_2 - x_1, y_2 - y_1)) = (\mathrm{a_p}, \mathrm{tb_p}) \tag{5}$$

where, l_p denotes the pointing direction in the image frame, $(x_1, y_1), (x_2, y_2)$ corresponds to the 2D pixel locations of the elbow and the wrist respectfully, and $-\infty < t < +\infty$.

For each s_i we can solve for t using equation 6 and find the intersecting point $p_i = a_i + t_i b_i$. Next, the distance from

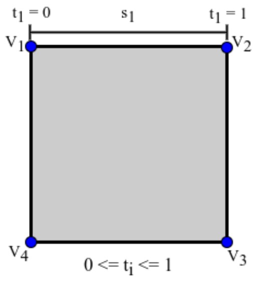


Figure 7: Visualization of the parametric equation of a segment

the object center to each corresponding intersecting point is measured and the minimum distance δ is computed and specified as the object distance δ. Algorithm 1 lists the steps for computing minimum distance δ for each detected object. Object with the least δ is estimated to be the pointed object.

$$t_i = (\mathrm{a_p} - a_i) \times \frac{\mathrm{b_p}}{b_i \times \mathrm{b_p}} \tag{6}$$

Algorithm 1: Minimum distance computation given 2D pointed vector and object boundary vertices

```
1  MinimumObjectDistance ($l_p, V$);
   Input  : $l_p$ is the 2D pointing vector
            $V$ is a list of vertices representing
            the bounding box
   Output: $\delta$ least distance from the object center
2  /* C is the center of the object          */
3  $C = \frac{1}{4}\sum_{i=1}^{4} V_i$
4  $\delta = null$
5  for $i = 1$ to 4 do
6      $s_i \leftarrow$ equation 4
7      $t_i \leftarrow$ equation 6
8      $p_i \leftarrow a_i + tb_i$
9      $d = ||p_i - C||$
10     if $\delta == null$ or $d < \delta$ then
11         $\delta = d$
12     end if
13 end for
14 return $\delta$
```

Algorithm 2 describes the steps for extracting relevant task parameters.

4 Experimental Results

We set up experiments that involved the participants performing a specific pointing gesture within a predefined scenario. The scene contained three objects: two books, and a cheez-it box, and the user could point at one object at a given time. For instance, in one of the scenarios the user was

Algorithm 2: Task parameters extraction

```
1  ExtractTaskParameters ($l_p, O$);
   Input  : $\mathscr{I}$: RGB image
            $\mathscr{C}$: verbal command
   Output: Parameters of a robotic task
2  $[\mathrm{a}, \mathrm{o_g}, \mathrm{r}, \mathrm{p}] \leftarrow$ extracted from $\mathscr{C}$ using the language
     patterns arranged in Table 1
3  $[\mathrm{O}, \mathrm{BB}] \leftarrow$ detected object name and the corresponding
     estimated boundary from image $\mathscr{I}$
4  $J_l, J_r \leftarrow$ Extract elbow and wrist joints location of left
     and right forearm using AlphaPose
5  Estimate if the user is pointing and the pointing arm from
     $J_l, J_r$ according to section 3.2
6  if user is pointing then
7      $l_p \leftarrow$ compute the 2D pointing vector as described in
         section 3.2.1
8      $d \leftarrow$ estimated general pointing direction
9      /* object with minimum distance is the estimated
          pointed object $\mathrm{o_p}$                    */
10     $\mathrm{o_p} =$
         $\arg\min_V \{(l_p, V) | LeastObjectDistance(l_p, V), V \in BB\}$
11     return $[\mathrm{a}, \mathrm{o_g}, \mathrm{r}, \mathrm{p}, \mathrm{o_s}, d]$
12 else
13     return $[\mathrm{a}, \mathrm{o_g}, \mathrm{r}, \mathrm{p}]$
```

instructed to point at the leftmost object by extending their right hand; therefore, for this data sample, we know the user was performing a pointing gesture using their right hand, pointing to their left, and pointing at the rightmost object (leftmost from user's perspective). This information was set as the ground truth for quantitative evaluation. The participants were positioned at the center of the image frame and were instructed to point at different parts of the scene. The pointing direction was labeled as either away, across, or straight for the operating/directing hand and "not pointing" for the other. These experiments were carried out where the user was standing at 1.22, 2.44, 3.66, and 4.88 meters distance from the camera.

As our work depends on several modules, we have decoupled them to evaluate how each of the modules performs. These modules include extracting task parameters from verbal commands, detecting the operating hand, estimating the pointing direction, and predicting the object of interest. Finally, we have presented the final result in terms of extracted task parameters in a tabular form.

The system receives the verbal command and extracts the parameters using the NLP techniques described in section 3.1.1 and 3.1.2. We examined RNN and LSTM based model each with 4 different variants: bidirectional and non bidirectional; 1 and 2 layers. We found the single layer Bi-LSTM network to be superior in performance. Figure 8 illustrates the superior performance of the Bi-LSTM network compared to other neural network architectures in terms of validation accuracy.

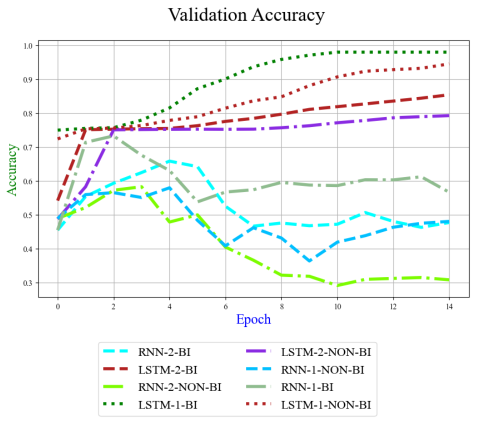


Figure 8: Validation accuracy

These parameters are stored so that each task can be executed sequentially. Table 2 arranges different verbal commands received from the user and the corresponding extracted task parameters; if no matches found, the corresponding parameters are set to ***None***. Each command initiates a task and is stored according to the order of task initiation (Table 3).

For each reliable frame, we compared the prediction with the label and measured the accuracy, precision, and recall. For a sample frame that has a label of "Right hand: pointing; Left hand: not pointing", if the prediction is "Right hand: pointing" then the sample is labeled as True Positive, else False Negative, if the prediction is "Left hand: pointing" then the sample is labeled as False Positive, else True Negative. Table 4 tabulates the accuracy, precision, and recall for varying distances. Figure 11 illustrates the system output for different pointing scenarios.

Next, we experimented on scenarios where multiple objects were placed on top of the table, each of which had predefined attributes (Figure 9). Additionally, the participants were instructed to point at a specific object. The system takes this information along with the natural language instruction to devise the task parameters and thereafter emits a follow-up response in the presence of any ambiguities. Two example scenarios are illustrated in Figure 10. Sample scenario configuration along with extracted task parameters have been arranged in Table 5. The *Structured Information* column exhibits the information extracted from the Pointing State and the Verbal Command. The first column indicates whether the user was pointing or not, the second column enumerates experiments (shortened to "Exp") for corresponding pointing states, the third column lists different verbal commands with fixed task action *"bring"*, the fourth presents the extracted information from the verbal commands and the simultaneous pointing state, the fifth column arranges the predicted object of interest (OOI) that needs the action to be performed upon, and the sixth column tabulates the corresponding response formulated by the system. Light blue cells indicate the presence of ambiguity.

Table 2: Extracted task parameters from different verbal commands

Verbal command: "give me the plate"
Object: plate | Action: give | Attributes: None | Position: None

Verbal command: "bring me that red cup"
Object: cup | Action: bring | Attributes: [red] | Position: None

Verbal command: "go left"
Object: None | Action: go | Attributes: None | Position: left

Verbal command: "grab the large green box on your right"
Object: box | Action: grab | Attributes: [green, large] | Position: right

Verbal command: "put the jar on the table"
Object: jar | Action: put | Attributes: None | Position: None

Table 3: Stored sequential task parameters

```
+====+========+========+===============+==========+
| NO | Object | Action |   Attributes  | Position |
+====+========+========+===============+==========+
| 1  | plate  | give   | None          | None     |
+====+========+========+===============+==========+
| 2  | cup    | bring  | [red]         | None     |
+====+========+========+===============+==========+
| 3  | None   | go     | None          | left     |
+====+========+========+===============+==========+
| 4  | box    | grab   | [green, large] | right    |
+====+========+========+===============+==========+
| 5  | jar    | put    | None          | None     |
+====+========+========+===============+==========+
```

Figure 9: Object attributes

Table 4: Pointing gesture recognition

Distance	**Accuracy**	**Precision**	**Recall**
4.88	1	1	1
3.66	0.995	1	0.99
2.44	0.995	1	0.99
1.22	0.995	1	0.99

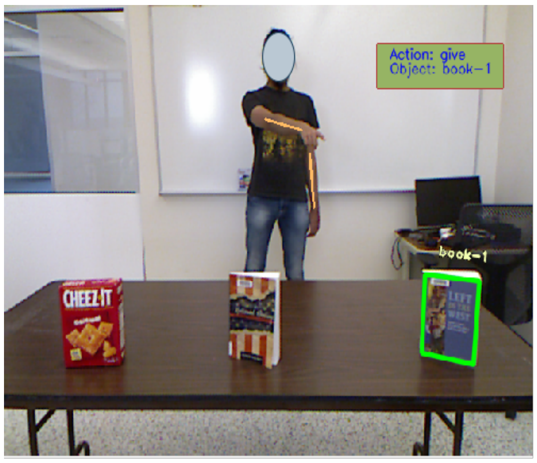
Ambiguity occurs when the **OOI** cannot be determined from the given verbal command and pointing gesture; the system fails to identify the object of interest and responds with the feedback *"Need additional information to identify object"*. Subsequently, the system waits for the user to perform the pointing gesture and/or amend the command; once these inputs are received, it repeats the entire process.

From the Table 5 we can see, for the *"Not Pointing"* state, ambiguity occurred when there was a lack of object attribute(s) (Exp 1, 3) for uniquely identifying the **OOI**, and thus, the system asked for additional information. For the *"Pointing"* state, the ambiguity arises when the pointing direction does not intersect with any of the object boundaries; in these scenarios, verbal commands can alleviate the ambiguity by providing additional information. Ambiguity can also arise if the inferred objects from the extracted identifiers and the pointing gesture are different. However, the system prioritizes the object inferred from the pointing gesture as the speech-to-text module may sometimes miss transcribe.

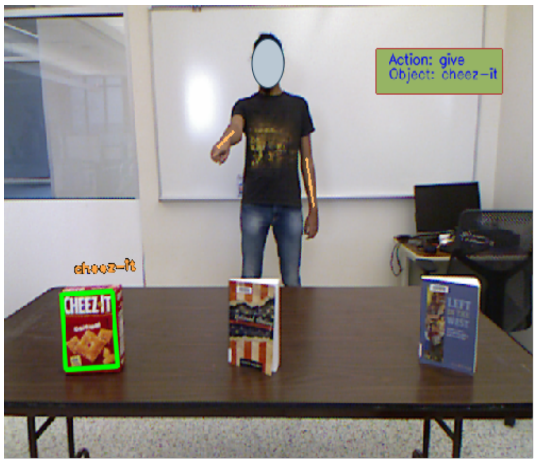
5 Conclusion and Further Research

In this paper, we have presented a HRI framework for extracting human-robot collaborative task parameters taking multiple inputs, and processing them simultaneously in real-time. Verbal communication is used to extract intricate information about the task e.g. action command, object attributes, etc., which is supported by pointing gesture recognition, the general direction along pointed object estimation to facilitate a natural interaction interface for the user. This information is assembled into a set of named parameters and can then be further analyzed to form a structured command that can be passed to and easily translated by a robotic entity.

We presented a pointing gesture recognition approach from a 2D image frame that can detect the gesture, and estimate the pointing direction and the pointed object in the scene. We implemented a template matching based object detection and planar pose estimation technique that provides information about the object present in the scene. We have also devised two approaches to extract the task information prevalent in collaborative interactions. The verbal command is received from the sensor and then converted to text to further match with the previously formulated language patterns to extract relevant task specific parameters. All this information is integrated to form the final task configuration; if an essential parameter is missing or there are ambiguities, the system responds with appropriate feedback.

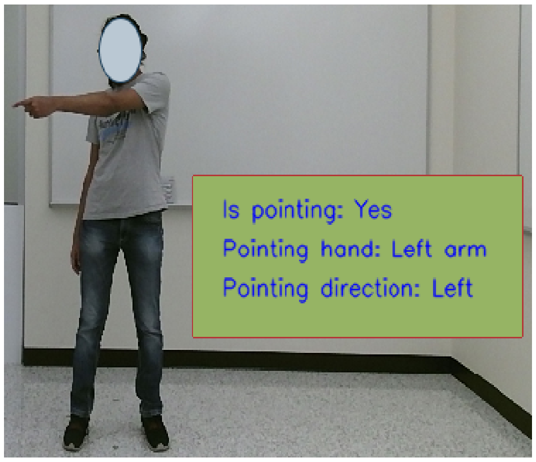

Action: give
Object: book-1

(a) Pointing to the object labeled "**book-1**"

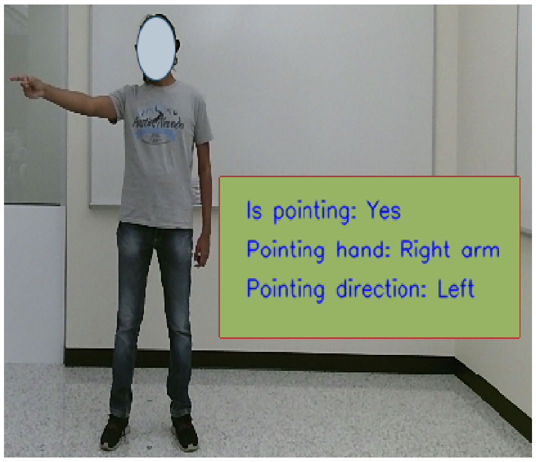

Action: give
Object: cheez-it

(b) Pointing to the object labeled "**cheez-it**"

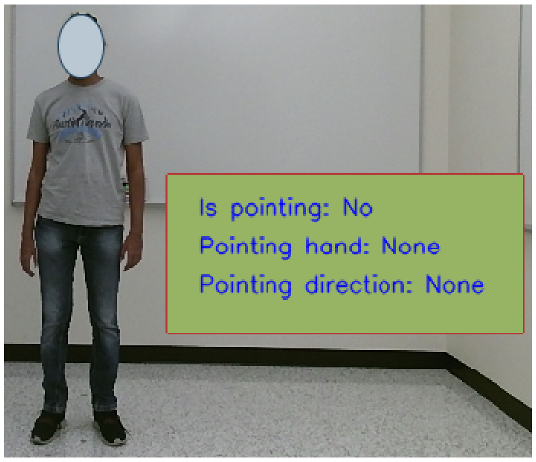
Figure 10: Example scenarios where the user points to different objects while voicing the command *"give me that"*


Is pointing: Yes
Pointing hand: Left arm
Pointing direction: Left

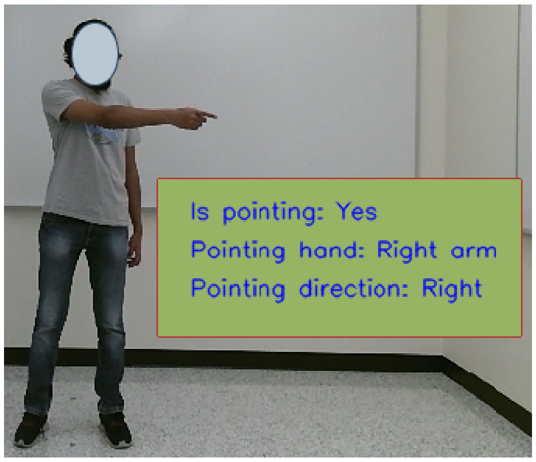
(a) Pointing across with left hand


Is pointing: Yes
Pointing hand: Right arm
Pointing direction: Left

(b) Pointing away with right hand


Is pointing: No
Pointing hand: None
Pointing direction: None

(c) No pointing


Is pointing: Yes
Pointing hand: Right arm
Pointing direction: Right

(d) Pointing across with right hand

Figure 11: System output with different pointing scenarios

Table 5: Generated task parameters

Pointing State	Exp #	Verbal Command	Structured information	Identified Object	Feedback
Pointing	1	bring that, bring me that	*{action: "bring", pointing_identifier: True, object: "book", object_identifiers: {attributes: null, position: null}}*	"book-1"	None
	2	bring the red book	*{action: "bring", pointing_identifier: True, object: "book", object_identifiers: {attributes: "red", position: }}*	"book-2"	None
	3	bring that red thing	*{action: "bring", pointing_identifier: True, object: null, object_identifiers: {attributes: "red", position: }}*	"cheez-it"	None
Not Pointing	1	bring that, bring me that	*{action: "bring", pointing_identifier: False, object: null, object_identifiers: {attributes: null, position: null}}*	None (ambiguous)	"Need additional information to identify object"
	2	bring the red book	*{action: "bring", pointing_identifier: False, object: "book", object_identifiers: {attributes: "red", position: null}}*	"book-2"	None
	3	bring that red thing	*{action: "bring", pointing_identifier: False, object: null, object_identifiers: {attributes: "red", position: "right"}}*	None (ambiguous)	"Need additional information to identify object"

We carried out experiments to measure the performance of our pointing gesture recognition system at different distances and it was able to recognize the state of the pointing gesture with very high accuracy. Next, we investigated the formation of the task configurations by passing different natural language instructions along with different gesture states. We have tabulated the extracted task parameters for different verbal commands along with how the task instructions are stored in a sequential list. Subsequently, we have illustrated the integration of these sensor data and interaction information with sample experimental results. Finally, we have listed the final task configurations along with the system feedback for different interaction scenarios.

The system can be further improved by introducing reliable 3D information. Having reliable depth information will effectively eliminate these limitations. Furthermore, more complex interaction scenarios comprising of multiple users and more intricate dialogues can be investigated to further develop more meaningful HRI systems.

References

[1] R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz, “"Tell Me When and Why to Do It! Run-Time Planner Model Updates via Natural Language Instruction",” in *"Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction"*, 2012, pp. 471–478.

[2] L. Dipietro, A. M. Sabatini, and P. Dario, “"A Survey of Glove-Based Systems and Their Applications",” *"Ieee Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)"*, vol. 38, no. 4, pp. 461–482, 2008.

[3] D. Droeschel, J. St"uckler, and S. Behnke, “"Learning to Interpret Pointing Gestures With a Time-of-Flight Camera",” in *"Proceedings of the 6th International Conference on Human-Robot Interaction"*, 2011, pp. 481–488.

[4] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn, “"What to Do and How to Do It: Translating Natural Language Directives Into Temporal and Dynamic Logic Representation for Goal Management and Action Execution",” in *"2009 IEEE International Conference on Robotics and Automation"*. IEEE, 2009, pp. 4163–4168.

[5] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, “"",” in *"Iccv"*, 2017.

[6] M. A. Fischler and R. C. Bolles, “"Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography",” *"Commun. ACM"*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[7] S. Hochreiter and J. Schmidhuber, “"Long Short-Term Memory",” *"Neural Computation"*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] S. Iba, J. M. V. Weghe, C. J. Paredis, and P. K. Khosla, “"An Architecture for Gesture-Based Control of Mobile Robots",” in *"Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots With High Intelligence and Emotional Quotients (Cat. No. 99CH36289)"*, vol. 2. IEEE, 1999, pp. 851–857.

[9] N. Jojic, B. Brumitt, B. Meyers, S. Harris, and T. Huang, “"Detection and Estimation of Pointing Gestures in Dense Disparity Maps",” in *"Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)"*. IEEE, 2000, pp. 468–475.

[10] R. E. Kahn and M. J. Swain, “"Understanding People Pointing: The Perseus System",” in *"Proceedings of International Symposium on Computer Vision-Iscv"*. IEEE, 1995, pp. 569–574.

[11] R. E. Kahn, M. J. Swain, P. N. Prokopowicz, and R. J. Firby, “"Gesture Recognition Using the Perseus Architecture",” in *"Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition"*. IEEE, 1996, pp. 734–741.

[12] R. Kehl and L. Van Gool, “"Real-Time Pointing Gesture Recognition for an Immersive Environment",” in *"Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings."*. IEEE, 2004, pp. 577–582.

[13] T. Kollar, S. Tellex, D. Roy, and N. Roy, “"Toward Understanding Natural Language Directions",” in *"2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)"*. IEEE, 2010, pp. 259–266.

[14] Y.-L. Kuo, B. Katz, and A. Barbu, “"Deep Compositional Robotic Planners That Follow Natural Language Commands",” in *"2020 IEEE International Conference on Robotics and Automation (ICRA)"*. IEEE, 2020, pp. 4906–4912.

[15] J. Li, C. Wang, H. Zhu, Y. Mao, H.-S. Fang, and C. Lu, “"CrowdPose: Efficient Crowded Scenes Pose Estimation and a New Benchmark",” *"arXiv Preprint arXiv:1812.00324"*, 2018.

[16] D. G. Lowe, “"",” *"International Journal of Computer Vision"*, 2004.

[17] M. MacMahon, B. Stankiewicz, and B. Kuipers, “"Walk the Talk: Connecting Language, Knowledge, and Action in Route Instructions",” *"Def"*, vol. 2, no. 6, p. 4, 2006.

[18] C. Matuszek, D. Fox, and K. Koscher, “"Following Directions Using Statistical Machine Translation",” in *"2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)"*. IEEE, 2010, pp. 251–258.

[19] M. Muja and D. G. Lowe, “"Fast Approximate Nearest Neighbors With Automatic Algorithm Configuration",” in *"International Conference on Computer Vision Theory and Application VISSAPP('09)"*. INSTICC Press, 2009, pp. 331–340.

[20] K. Nickel and R. Stiefelhagen, “"Pointing Gesture Recognition Based on 3d-Tracking of Face, Hands and Head Orientation",” in *"Proceedings of the 5th International Conference on Multimodal Interfaces"*, 2003, pp. 140–146.

[21] C.-B. Park and S.-W. Lee, “"Real-Time 3D Pointing Gesture Recognition for Mobile Robots With Cascade HMM and Particle Filter",” *"Image and Vision Computing"*, vol. 29, no. 1, pp. 51–63, 2011.

[22] M. Pateraki, H. Baltzakis, and P. Trahanias, “"Visual Estimation of Pointed Targets for Robot Guidance via Fusion of Face Pose and Hand Orientation",” *"Computer Vision and Image Understanding"*, vol. 120, pp. 1–13, 2014.

[23] S. K. Paul, M. T. Chowdhury, M. Nicolescu, M. Nicolescu, and D. Feil-Seifer, “"Object Detection and Pose Estimation From RGB and Depth Data for Real-Time, Adaptive Robotic Grasping",” in *"Advances in Computer Vision and Computational Biology"*. Springer, 2021, pp. 121–142.

[24] F. H. Previc, “"The Neuropsychology of 3-D Space.",” *"Psychological Bulletin"*, vol. 124, no. 2, p. 123, 1998.

[25] D. L. Quam, “"Gesture Recognition With a Dataglove",” in *"IEEE Conference on Aerospace and Electronics"*. IEEE, 1990, pp. 755–760.

[26] S. S. Rautaray and A. Agrawal, “"Vision Based Hand Gesture Recognition for Human Computer Interaction: A Survey",” *"Artificial Intelligence Review"*, vol. 43, no. 1, pp. 1–54, 2015.

[27] J. Richarz, A. Scheidig, C. Martin, S. M"uller, and H.-M. Gross, “"A Monocular Pointing Pose Estimator for Gestural Instruction of a Mobile Robot",” *"International Journal of Advanced Robotic Systems"*, vol. 4, no. 1, p. 17, 2007.

[28] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock, “"Spatial Language for Human-Robot Dialogs",” *"IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)"*, vol. 34, no. 2, pp. 154–167, 2004.

[29] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy, “"Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation",” in *"Proceedings of the AAAI Conference on Artificial Intelligence"*, vol. 25, no. 1, 2011.

[30] H. Watanabe, H. Hongo, M. Yasumoto, and K. Yamamoto, “"Detection and Estimation of Omni-Directional Pointing Gestures Using Multiple Cameras.",” in *"Mva"*, 2000, pp. 345–348.

[31] A. D. Wilson and A. F. Bobick, “"Parametric Hidden Markov Models for Gesture Recognition",” *"IEEE Transactions on Pattern Analysis and Machine Intelligence"*, vol. 21, no. 9, pp. 884–900, 1999.

[32] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, “"Pfinder: Real-Time Tracking of the Human Body",” *"IEEE Transactions on Pattern Analysis and Machine Intelligence"*, vol. 19, no. 7, pp. 780–785, 1997.

Shuvo Kumar Paul received his M.S. in 2020 in computer science and engineering from University of Nevada, Reno. He completed his B.S. from North South University, Bangladesh. Before joining UNR, he worked as a research associate at the AGENCY lab (previously CVCR). His research interests include machine learning, computer vision, computational linguistics, and robotics

Pourya Hoseini received his Ph.D. in 2020 and M.S. in 2017, both in computer science and engineering from University of Nevada, Reno. He also received a M.S. degree in 2011 and a B.S. in 2008 in electrical engineering from Urmia University and Azad University, Iran, respectively. His research interests are machine learning, computer vision, and evolutionary computing.

Arjun Vettath Gopinath is currently working as a Software Developer at N2N Services inc. He graduated with a degree in M.S. in Computer Science and Engineering from the University of Nevada, Reno in May 2021. He received his B.S. from SCMS School of Technology and Management, Kerala, India. His interests are Machine Learning, Human-Computer Interaction and Software Development.

Mircea Nicolescu is a Professor of Computer Science and Engineering at the University of Nevada, Reno and co-director of the UNR Computer Vision Laboratory. He received a PhD degree from the University of Southern California in 2003, a MS degree from USC in 1999, and a BS degree from the Polytechnic University Bucharest, Romania in 1995, all in Computer Science. His research interests include visual motion analysis, perceptual organization, vision-based surveillance, and activity recognition. Dr. Nicolescu's research has been funded by the Department of Homeland Security, the Office of Naval Research, the National Science Foundation and NASA. He is a member of the IEEE Computer Society.

Monica Nicolescu is a Professor with the Computer Science and Engineering Department at the University of Nevada, Reno and is the Director of the UNR Robotics Research Lab. Dr. Nicolescu earned her PhD degree in Computer Science from the University of Southern California (2003) at the Center for Robotics and Embedded Systems. She obtained her MS degree in Computer Science from USC (1999) and a BS in Computer Science at the Polytechnic University Bucharest (Romania, 1995). Her research interests are in the areas of human-robot interaction, robot control, learning, and multi-robot systems. Dr. Nicolescu's research has been supported by the National Science Foundation, the Office of Naval Research, the Army Research Laboratory, the Department of Energy and Nevada Nanotech Systems. In 2006 she was a recipient of the NSF Early Career Development Award (CAREER) Award for her work on robot learning by demonstration