Multimodal Error Correction with Natural Language and Pointing Gestures

Stefan Constantin Fevziye Irem Eyiokur Dogucan Yaman Leonard Bärmann
Alex Waibel
Interactive Systems Lab, Karlsruhe Institute of Technology
Adenauerring 2, 76131 Karlsruhe, Germany
stefan.constantin@kit.edu

Abstract

Error correction is crucial in human-computer interaction, as it can provide supervision for incrementally learning artificial intelligence. If a system maps entities like objects or persons with unknown class to inappropriate existing classes, or misrecognizes entities from known classes when there is too high train-test discrepancy, error correction is a natural way for a user to improve the system. Provided an agent with visual perception, if such entity is in the view of the system, pointing gestures can dramatically simplify the error correction. Therefore, we propose a modularized system for multimodal error correction using natural language and pointing gestures. First, pointing line generation and region proposal detects whether there is a pointing gesture, and if yes, which candidate objects (i. e. RoIs) are on the pointing line. Second, these RoIs (if any) and the user's utterances are fed into a VL-T5 network to extract and link both the class name and the corresponding RoI of the referred entity, or to output that there is no error correction. In the latter case, the utterances can be passed to a downstream component for Natural Language Understanding. We use additional, challenging annotations for an existing real-world pointing gesture dataset to evaluate our proposed system. Furthermore, we demonstrate our approach by integrating it on a real-world steerable laser pointer robot, enabling interactive multimodal error correction and thus incremental learning of new objects.

1. Introduction

Generalization is a key strength of human intelligence. Humans can understand words which they have never heard by transferring the meaning from words with similar sound or context. In the domain of natural language processing, neural approaches are state-of-the-art because they have a comparable ability to generalize from similar examples. However, sometimes this generalization is too broad, and a new phrase or word is assigned to a wrong meaning. In this case, a correction of the system is necessary. For example, a system could generalize the word "forky" (which is a toy staring in an animation movie) to the concept "fork". But if a user wants the "forky", a robot should actually bring the "forky" and not a fork. From the user's viewpoint, a natural and easy way to correct the system is to use natural language to tell the system that something is not correct and to visually point at the correct object so that the robot can learn its appearance. An example of such scenario is shown in Figure 1.

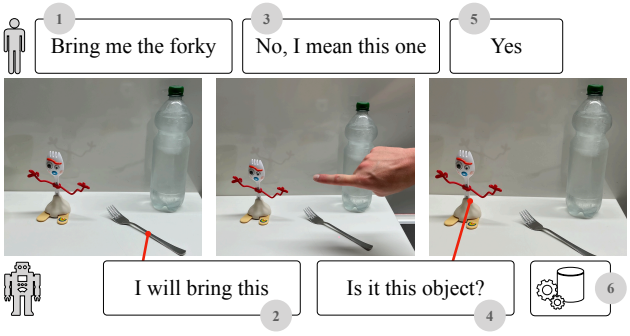


Figure 1: We tackle the task of multimodal error correction: A user gives a command (1) involving a potentially unknown object, which is misinterpreted by the system (2). In our showcase, the system is a small robot with a laser pointer (see Section 7). Because of the error, the user performs a multimodal correction through language and pointing gesture (3). The system then asks for confirmation (4), and if this is approved (5), the resulting information is used to perform the action with the correct object and incrementally learn the new object's name (6).

In this work, we tackle this problem by proposing a system with the following three components: pointing line generation, region proposal, and multimodal error correction. The pointing line generation is responsible for detecting the presence of a human pointing gesture from the image/video input, and computing the pointing line in case a gesture is

detected. Region proposal is utilized to find the Regions of Interest (RoIs) of candidate objects, as these are intersected with the pointing line to deliver pointing target candidates. If target candidates are available, they are forwarded to the multimodal error correction component. This component combines the information from vision and natural language to decide whether there is a correction and if that is the case, disambiguate the user's correction utterance. The result of the multimodal error correction module can then be used to trigger incremental learning of the new object, given its referring expression (word/phrase) as well as the visual grounding (RoI image).

We present the following contributions: First, we introduce a system for multimodal error correction, combining natural language and pointing gestures. Second, to assess the performance of multimodal error correction, we collected a test dataset of challenging natural language utterances that augment the pointing video dataset published in [10]. Third, we present extensive evaluation results of our individual components and the overall approach. Furthermore, we demonstrate the usefulness of the system by a showcase application on a small real-world laser pointer robot.

2. Related Work

2.1. Pointing Gesture and Line Detection

Pointing gestures are being used by researchers to improve human-robot interaction. There are two main types of approaches for recognizing pointing gestures: those that use a stereo camera or Kinect-based input to estimate 3D coordinates [24, 35, 34, 27, 20, 5, 16, 14] and those that use RGB camera input and estimate pointing direction based on 2D coordinates [43, 39, 21, 33, 32]. Different algorithms are used for pointing recognition, such as Hidden-Markov-Models-based [35, 34], probabilistic approaches [12, 46], and deep-learning-based methods [5, 20, 33, 32]. After recognizing a pointing gesture, various body parts such as the hand, forearm, and face can be used to calculate the line of sight between them. Several different approaches have been used to track pointing gestures, such as using dense disparity maps [24, 35], skin-color classification [35], and multi-view cameras [25]. Recently, deep-learning-based models have become more useful for both 3D and 2D pointing recognition. In these approaches, off-the-shelf models are used to calculate 3D vectors, estimate human body pose, and detect target objects. Azari et al. [5] primarily focused on detecting face and hand areas, while Hu et al. [20] used the estimation of human body pose to determine the pointing line using eye and wrist coordinates. In [33], a pipeline was built using inputs from an RGB drone camera, where the OpenPose model [7] was used to estimate human body pose and a YOLO-based detector [40] was used to identify target objects. In their subsequent work [32], the authors expanded on this approach by utilizing a monocular simultaneous localization and mapping algorithm to generate an unscaled point cloud, which allowed them to estimate the depth coordinates of both the hand and target objects using estimated 2D coordinates and camera calibration.

2.2. Multimodal Models

The fusion of visual and language input to one output is researched in a lot of studies [6, 19, 2], but these systems are tailored to one specific output task.

The VL-T5 model described in [9] offers an architecture that multiple tasks like VQA, Visual Grounding, and Grounded captioning can be handled by one model. It uses only one Transformer model and the text input as well as the vision input (36 RoIs) are fed into the Transformer encoder and the output of the Transformer decoder is a text sequence. It uses 220 million parameters. A multi-task pre-training (multimodal language modeling, visual question answering, image-text matching, visual grounding, and grounded captioning) can improve the performance for data-scarce tasks. PaLM-E [17] uses a similar approach by using only the Transformer decoder, but with its 562 billion parameters, it is much larger than the VL-T5 model. Further multimodal large language models are GPT-4 [36] and LLaVA [30].

2.3. Combining Pointing and Language

Chen et al. [8] introduced Embodied Reference Understanding. The idea is to benefit from a natural language referring expression in conjunction with a pointing gesture. Besides, a dataset that contains video and text was published. However, the authors focused on single-round reference understanding and also guided annotators to provide unambiguous referring expressions which are the main differences to our work and also with [10]. In [45], the authors proposed the Reasoning from Your Perspective model that uses a depth estimation map to acquire 3D coordinates in order to convert it to an embodied 3D coordinate by assuming the sender's position as the origin. Thus, the proposed method is able to model the relations between the objects, the sender and the receiver. Inspired by [8], the authors in [28] proposed a method that uses a CNN for embedding the visual input, BERT [15] for embedding the language input, and a Transformer model [52] for the fusion of both modalities. The main contribution of this paper is to benefit from the so-called virtual touch line which describes the line from the head pose instead of using the elbow-wrist line. According to the experimental results, it is found that the head pose has more robust and accurate information about the pointing than the elbow-wrist line. Therefore, the performance is increased compared to [8]. Finally, in [10], the authors propose an interactive, multi-

modal, task-oriented robot dialog system that also benefits from the pointing line generation to localize the referred objects. One of the main contributions of it is to provide an iterative approach to apply correction for enhancing the performance. Moreover, the authors collected and published an unconstrained test dataset that contains 222 videos. [10] is the most related paper to our work. However, our work differentiates from [10] in the different ways to be able to recognize unknown objects: (1) we benefit from a Region Proposal Network (RPN) to recognize the objects beyond the small set of known objects in the MSCOCO dataset and object detection models; (2) we are able to better disambiguate the objects by using a feature vector instead of the class name for an object and thereby can utilize descriptions like the color that are necessary for unknown objects.

2.4. Interactive Teaching of New Objects

In [18], a system can learn new objects by a user showing them and using natural language, but the objects must be shown to the system, to simply point at them is not possible. In contrast, in [53], a robot learns new objects by a user that is pointing at the new objects. Compared to our work, there is no interaction with natural language. In contrast, [51] combine pointing gestures and natural language to learn new objects, but the process is initiated by the robot. The process of learning new objects can be initiated by the user in [4] by explaining to the robot an object which the user is pointing at. However, the user cannot correct the robot. We argue that, in real-world scenarios, the user does often not know the internals of the robot's knowledge. Thus, it is necessary to correct the robot post execution instead of teaching objects in advance. Systems that implement error correction are presented in [47, 48, 11, 50], but none of these works explore the possibility of using pointing gestures for multimodal error correction.

3. Methodology

3.1. Overview

Our goal is to perform multimodal error correction. We thus have as input the most recent user utterances u_1, u_2 (where u_2 is the latest one), as well as the current camera image I (or the camera video $V = (I_1, ..., I_N)$). u_2 can either be a correction of the initial utterance u_1, i. e. $u_2 = c$, or it can be another independent command, i. e. $u_2 = u'$. We define $P_{\text{n}} = (u_1, u')$ to be a pair of independent utterances, and $P_{\text{c}} = (u_1, c)$ to be a pair involving a correction. Each image I shows a set of objects $O_I = \{o_1, ..., o_M\}$, and can either show a human pointing at the target object o_p, or no such gesture.

The multimodal error correction system $\mathcal{M}$ is supposed to behave as follows: In case there is no correction, the model is supposed to detect that, i. e. $\mathcal{M}(P_{\text{n}}, I) =$ *no correction*, and the utterance u' can be passed to a downstream component for Natural Language Understanding, e. g. to trigger appropriate actions. Afterwards, $u_1 := u_2$ and the system waits for the next utterance.

If there is a correction, $\mathcal{M}(P_{\text{c}}, I) = (o_p, l)$, where l is the label used in u_1 to refer to o_p. The resulting pair (o_p, l) can be passed to another component for incremental learning of the novel object. For the scope of this paper, as we focus on multimodal error correction, we assume that a pointing gesture is always present when there is a correction. Afterwards, u_2 is dropped and u_1 is kept so that the next utterance can either correct again or be independent.

3.2. Pointing Line Generation and Region Proposal

Pointing recognition is the process of recognizing and interpreting human gestures where an arm and index finger is extended in a specific direction to indicate an object or location by a subject. In order to recognize which object is pointed at by a subject, several different steps are necessary. The first step is to detect and track the hand. By using the detected hand's coordinates and tracking information, the intended pointing moment can be detected. For this, inspired by [10], we train a YOLOv5 model [40, 23]. Since recent studies have shown that using an additional classifier for hand classification works better than training an object detector considering the hand label [10], we train a lightweight MobileNetV2 model [42] to learn hand classification by using the detected and cropped hand, which we obtain with the YOLOv5 hand detection model, as an input. According to the hand detection and classification outputs, we consider the movement of the hand in the video in between the frames and employ a decision algorithm to decide the start and end time of the pointing movement. For this, the stable hand in k consecutive frames means the pointing has started. k is a hyperparameter that we empirically set to 10.

After decision of pointing frames, next step is to determine which object or objects are considered pointing target candidates. For this, we propose to use a Region Proposal Network (RPN) to detect the candidate objects in the scene. We chose the RPN that was proposed in the Faster R-CNN object detection method [41], which is widely used in computer vision for object detection. The purpose of this component is to generate a set of RoIs, which are candidate object bounding boxes that may contain an object. An RPN takes an image as input and produces a set of object proposals as output. The RPN operates on a convolutional feature map that is produced by an initial convolutional neural network (CNN) that processes the input image. The RPN slides a small, fixed-size window over each location in the feature map, and for each window location, it generates a set of K anchor boxes of different scales and aspect ratios centered at that location. For each anchor box, the RPN

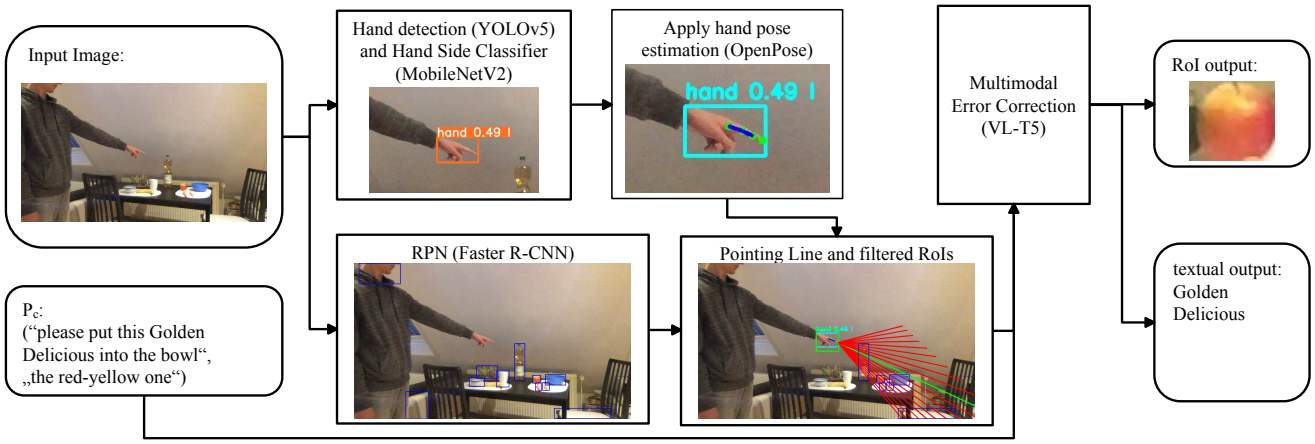


Figure 2: Information flow for an input sample including a correction. In case there was no correction, there would be no pointing line and therefore no RoIs are given to the multimodal error correction component, the textual output would be "no correction" and there would be no RoI output

predicts two scores: the probability that the anchor box contains an object, and the coordinates of the bounding box that best fits the object if there is one. It generates these scores by applying a fully connected layer to the features within each window. Then, these predicted scores are used to rank the anchor boxes by their likelihood of containing an object. The top-K anchor boxes with the highest objectness scores are selected as region proposals, and then passed on to a subsequent network for further refinement and classification. The result is a set $R = \{r_1, ..., r_K\}$ of RoIs, where each r_i consists of bounding box coordinates and the features associated with that RoI. Because of its ability to predict bounding boxes for candidate objects without limitation to known object classes, introducing an RPN (instead of object detection like in [10]) is an important step to go beyond known objects in the pointing region proposal task.

After we detect the candidate objects in the scene by our RPN (see Figure 2), using detected hand crop, we predict the hand pose with a pre-trained OpenPose pose estimation model [7] to localize the finger points to draw the pointing line. The finger provides a more accurate and robust pointing line estimation compared to other points such as elbow and wrists, although the hand pose estimation model may make wrong or less accurate predictions due to the difficulty of the pose as well as the lighting and resolution conditions. Then, the bounding boxes from the RPN overlapping with the estimated pointing line are selected as pointing candidate objects. However, due to the difference between the viewing perspective of the person and the camera as well as the pose, the line may not be precisely correct. In order to mitigate this problem, we propose to use a tolerance area while drawing the line from the finger. For this, we draw lines to adjacent the main line inside a predefined angle range. Afterwards, we choose the overlapping candidate object boxes with the resulting area, see Figure 2. However, as can be seen from Figure 2, we do not draw these tolerance lines as long as the original line. We empirically found out that the error caused by perceptual disorder is likely to occur more frequently for the objects that are close to the hand rather than the objects that are far from the hand. Therefore, when applying the tolerance area, we consider this aspect and shorten the lines based on the angular distance from the main line.

3.3. Multimodal Error Correction

For the multimodal error correction, we propose to finetune a VL-T5 model [9] with a dataset as described in Section 5. In the following, we describe the intended behavior of the model, as induced by training on that dataset. Note, however, that, since the model is a neural network, there are in fact no hard-coded rules in the system. VL-T5 is pre-trained in a multi-task setting, where each task is identified using a task-specific input prefix. Its pre-training is also multimodal, which means that the network receives both text as well as features of multiple images. In our case, the input to the model is constituted by 1) the prefix "correction visual grounding", 2) the text of the last two utterances u_1, u_2 of the user, and 3) the pointing candidate RoIs $R = \{r_1, ..., r_K\}$ (i. e., their bounding box coordinates and their features). First, the system detects if there is a correction. If no correction is detected, the textual output $t =$ "no correction", stating that the most recent utterance is not a correction of the previous one. In contrast, if a correction is detected, there are two outputs: first, the ID token of the RoI $r_{\hat{p}}$ that is most probably pointed at (given the cues from language and vision) and second, the name l of that entity as textual output, i. e. $t = l$. The name should be the one used in u_1 (i. e. not the correction) that lead to the

erroneous actions. When possible in the deployment scenario of the system, we add another level of interactivity to confirm the result before triggering an incremental learning component. For instance, this can be achieved by the system showing the user the cropped image of the output RoI on its display and asking whether this is correct. There are robots like Pepper [37] and ARMAR-6 [3] that have displays, or the robot can point at the object with a laser pointer (see Section 7). If the user does not confirm the result, the wrong RoI $r_{\hat{p}}$ is removed from R and the component is re-run again. This procedure can repeat until the correct RoI r_p is found, R is empty (in that case, the system could ask the user to do the pointing again) or the user aborts the task by natural language. In Section 6.2, we evaluate up to two re-runs.

So far, the system described deals with a single image I of a human pointing gesture. A more realistic but also more complex scenario is handling a video $V = (I_1, ..., I_N)$ instead (in this work, this only applies to the test dataset, see Section 5). In this case, we compare four approaches for deciding on the overall result. First, we select the middle frame of all frames of the longest consecutive pointing sequence (if the number is even, we chose the lower one) and use only the results of this frame. We call this approach "middle frame". The second approach also uses the middle frame, but with a reduced sequence of the first n frames where a pointing gesture is detected. If less than n frames are in the pointing sequence, all frames are used, i. e. we use $I_{\lfloor \min(N,n)/2 \rfloor}$. This approach is called "middle frame of the max. first n frames)" and aims to simulate an online system, which is desirable as it is unnatural to first finish the pointing and then get a system answer. The last two approaches average the individual results from multiple frames. Recall that for each frame I_i with $i \in \{1, ..., N\}$, the model produces a textual output t_i (either "no correction" or object name l_i) and, if a correction was detected, an RoI $r_{\hat{p},i}$ is outputted. The third approach uses all frames of the longest consecutive pointing sequence of the pointing video V. The overall textual output is decided by simple majority vote over all t_i. For averaging the output RoIs over all frames, the RoIs are clustered. For every RoI, it is checked if there is already a cluster where the RoI has an IoU higher than the threshold and if yes, it is added to this cluster. If not, a new cluster is created. Then, the largest cluster is selected, and all RoIs therein are averaged to an RoI bounding box that is the output of the system. We call this approach "averaged frames". The fourth approach is similar to the third approach, except that, as approach 2, a reduced pointing sequence of the first n frames where a pointing gesture is detected is used to simulate an online system, i. e. $i \in \{1, ..., \min(n, N)\}$. This approach is called "averaged frames of the max. first n frames". To average RoIs makes only sense for a static camera.

4. Implementation

For the hand detection and hand classification, we employ the official PyTorch [38] implementation of YOLOv5 [23] and the TorchVision [31] implementation of MobileNetV2 [42] models, respectively. We use the official Python implementation of OpenPose [7]. For the RoI feature vector generation, we utilize the Detectron2 [54] re-implementation [49] in PyTorch that uses the same model and weights as the Caffe VG Faster R-CNN provided in bottom-up-attention [1]. For the multimodal error correction, we use the PyTorch code provided with [9]. Instead of using 36 RoIs everytime, we use the RoIs provided by the pointing line generation and region proposal component.

5. Dataset

5.1. Pointing Line Generation and Region Proposal

To train our MobileNetV2 [42] hand classification model and YOLOv5 [40, 23] hand detection model, we employ the *100k Frames* version of the 100DOH dataset [44]. In this dataset, there are almost 100k frames and around 189k hand box annotations with the hand side labels. Since the dataset is a large-scale unconstrained dataset, our hand detection and classification models that were trained on this dataset are robust against challenging real-world conditions. On the other hand, for the RPN, we employ the pre-trained model without further fine-tuning.

5.2. Multimodal Error Correction

Our dataset for training and validating the multimodal error correction component is based on the EPIC-KITCHENS-100 dataset [13] for the textual data and the MSCOCO [29] dataset (data split from 2017) for the visual data. An annotation is a textual description of the action in the video, e. g. "put cup into cupboard", that we re-purpose as a robot command. There is also metadata like the set of objects and their classes used in the annotations. Every image of MSCOCO is annotated with bounding boxes, each associated with a category label for the object depicted inside that bounding box. For our dataset, first, we create a mapping between the objects in EPIC-KITCHENS-100 and the category labels of MSCOCO. We were able to find 29 MSCOCO category and EPIC-KITCHENS-100 class pairs. Subsequently, we use this mapping to create subsets of EPIC-KITCHENS-100 and MSCOCO, removing samples that do not include one of the mapped classes/categories.

Each sample in the dataset needs to consist of: 1) textual input, i. e. either an utterance pair P_c involving a correction or a pair P_n without a correction, 2) visual input, i. e. a set R of RoIs, 3) a textual target t, which is either the object name l or the text "no correction" and 4) in case there is a correction, a pointing target RoI output r_p.

To construct the textual input data (i. e. P_c, P_n, t) for our model, we start with the annotations of the EPIC-KITCHENS-100 subset. Since we want to be able to train the system to recognize unknown objects, we apply one the following three strategies (uniformly at random) for modifying the textual description of the object: replacing the noun with a brand name, putting a brand name before the noun, or leaving the noun unchanged. The brand names have no semantic connection to the object, but are a straightforward way to introduce rare words. The modified textual description is used as u_1. To construct a correction utterance c, we define 14 templates for the training dataset (e. g. "this is a LABEL") and with a probability of 50 % we randomly select (equally distributed) one of four prefixes ("no,", "look,", "i meant", and "please,"). The validation dataset is generated using three separate templates that differ from the templates of the training dataset. Some templates make use of an "attribute" placeholder that is filled with an attribute value assigned to the bounding box of the image (which is added in a next step). These attributes are classified by the Detectron2 model [54] with the Caffe VG Faster R-CNN weights provided in bottom-up-attention [1]. There are 400 attributes such as "black" and "round". The object's textual representation is used as the textual target $t = l$ for the model. For every annotation in EPIC-KITCHENS-100, we form multiple correction pairs P_c by combining the generated u_1 with 5 and 1 randomly selected correction templates for the training dataset and validation dataset, respectively. In addition, we augment the u_1 with different patterns to have more natural language variety, because all annotations start with the verb. We use different patterns for the training and validation dataset. Furthermore, since we also need samples without error correction, we take all these generated u_1 and shuffled them to form pairs P_n of independent u_1, u'.

After generating the textual data, we combine textual and visual data. We use up to 10 bounding boxes of every image of the MSCOCO dataset and each of them seeds a separate dataset example. For each bounding box, we select some P_c from the generated utterance pairs where the object class associated with the sample maps to the category of the bounding box. The selected P_c is then removed from the set of generated utterances, but if that set becomes empty, we reset it to its initial state. Because the pointing component might recognize any number of RoIs including zero (i. e., the pointing gesture must be improved), we must simulate this for the dataset to train the multimodal error correction. With a probability of 20 %, we combine P_c with zero RoIs and in the remaining 80 %, we randomly (equally distributed) add 1 to 35 additional RoIs (generated by our RPN). From there on, the reference bounding box (that seeded the sample) is treated as r_p and is added to R like any other RoI. If we add more than one RoI, we randomly choose its input sequence position. Thus, we now have a complete dataset sample of $S_\text{c} = (P_\text{c}, R, l, r_p)$, where l is defined by u_1. Every time we add such sample with a correction to the dataset, we also add one without a correction to the dataset to train the system to be able to detect if there is a correction or not. To gain robustness against false positives of the pointing recognition, with a probability of 50 %, we add the same RoIs to such P_n sample as for S_c. That means, we add a sample $S_\text{n} = (P_\text{n}, R, \text{"no correction"})$, where R is either the same as for the corresponding S_c, or the empty set.

In the end, we have 203 342 samples with a correction and the same number without a correction in our training dataset. The validation set is comprised of 7232 pairs each.

To test our component under real-world conditions, we use the 182 videos with pointing gestures and the 40 videos without pointing gestures from [10]. We added the following labels to the dataset: For every video that shows a pointing gesture, four humans annotated two P_c pairs as well as the reference bounding box r_p. The videos without pointing gesture were annotated with two P_n pairs by the same four humans.

6. Evaluation

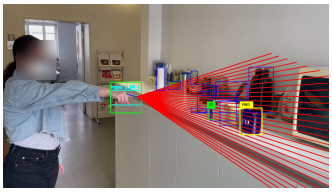
In this section, we will first evaluate the pointing line generation and region proposal component described in Section 3.2. Afterwards, we present results for multimodal error correction (introduced in Section 3.3), which builds on the results of the previous step.

6.1. Pointing Line Generation and Region Proposal

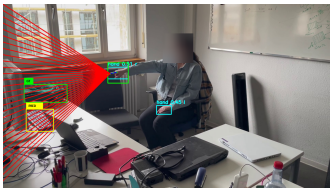
To evaluate the pointing line generation and region proposal component, we checked whether the annotated reference RoI r_p is included in the set of proposed RoIs $R = \{r_1, ..., r_K\}$. For this, we use the Intersection over Union (IoU) thresholds $\tau \in \{0.25, 0.5, 0.75\}$ and consider r_p to be included in R if $\exists i : IoU(r_i, r_p) > \tau$, i. e. the IoU between one of the proposed RoI and the reference is higher than the threshold τ. For the approaches that use multiple frames, we consider it as included if the above criterion is fulfilled for more than 50 % of the frames. The results are presented in Table 1. The best approach is using only the middle frame of the max. first 30 pointing frames where a pointing gesture is detected where 62.26 % of the reference RoIs are included in the proposed RoIs for an IoU threshold of $\tau = 0.25$, 55.10 % for $\tau = 0.5$ and 42.42 % for $\tau = 0.75$.

6.2. Multimodal Error Correction

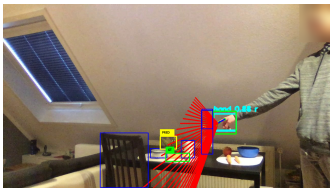
After giving a test sample to our system, we have the textual output $t =$ "no correction" or, in case a correction is detected, $t = l$ and an output RoI $r_{\hat{p}}$. Given the annotated target label l and reference RoI r_p, we compute the following evaluation metrics: First, we only want to evaluate the



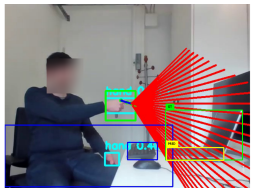
$u_1 =$ "please cut the tomatos using the cutlery"
$u_2 =$ "the cutlery is this one"
$\hat{l} =$ "knife" $\neq l =$ "cutlery"
$r_p \in R \wedge r_{\hat{p}} \neq r_p$



$u_1 =$ "I need you to pick up that knapsack and put it on the table."
$u_2 =$ "This is the knapsack that I asked"
$\hat{l} == l =$ "knapsack"
$r_p \notin R \Rightarrow r_{\hat{p}} \neq r_p$



$u_1 =$ "please fry the yellow fruit one"
$u_2 =$ "that banana!"
$\hat{l} == l =$ "yellow fruit"
$r_p \in R \wedge r_{\hat{p}} \neq r_p$



$u_1 =$ "Robot please look-behind that laptop to find my keys"
$u_2 =$ "behind this laptop"
$\hat{l} == l =$ "laptop"
$r_p \in R \wedge r_{\hat{p}} \neq r_p$

Figure 3: Failure cases from the test dataset; the predicted RoI $r_{\hat{p}}$ has a yellow bounding box with the label "PRED", the reference RoI r_p has a green bounding box with the label "GT", the detected hands have a cyan bounding box with the label "hand", the detected pointing hand has a green bounding box that covers the cyan bounding box, and the other RoIs on the pointing line have blue bounding boxes

approach	IoU 0.25	IoU 0.5	IoU 0.75
middle frame	60.06 %	54.55 %	40.22 %
middle frame of the max. first 30 frames	62.26 %	55.10 %	42.42 %
averaged frames	57.85 %	53.44 %	39.12 %
averaged frames of the max. first 30 frames	58.40 %	53.44 %	40.22 %

Table 1: evaluation results of the pointing line generation and region proposal component, is the reference RoI in the set of proposed RoI, evaluated with the IoU thresholds of 0.25, 0.5, and 0.75

textual output. For that, we compare t and $\hat{t}$ to compute the *textual accuracy*. Second, we want to evaluate the RoI proposals separately. For that, we compute the $IoU(r_p, r_{\hat{p}})$, and report the *RoI accuracy* as the percentage of samples surpassing a threshold τ of 0.25, 0.5, or 0.75, respectively. Third, we want to evaluate the overall system with the *complete output accuracy*, where we consider a sample to be correctly solved if it is either detected correctly as "no correction" or both $t = \hat{t}$ and $IoU(r_p, r_{\hat{p}}) > \tau$. For the clustering algorithm of the averaged frames approaches (see Section 3.3), we always set the IoU threshold equal to the one used for evaluating the RoIs.

The highest textual accuracy of 81.04 % is produced by the approach that average the max. first 30 frames where a pointing gesture is detected. The lowest accuracy with 79.01 % is only slightly worse (middle frame of the max. first 30 frames where a pointing gesture is detected). The middle frame approach (79.68 %) and the averaged frame approach (80.81 %) have accuracies in between.

The evaluation of the proposed RoIs depends on the results delivered by the pointing line generation and region proposal component. An RoI can only be outputted if it is included in the RoIs proposed by the pointing line generation and region proposal component. Thus, for the two middle frame approaches, the results of the pointing line generation and region proposal component are an upper bound. In contrast, for the average frames approaches, the results in Table 1 are only an approximation of an upper bound, since the evaluation of the pointing/RPN results is stricter than the criterion used to average the result of the average frames approaches: When evaluating the pointing/RPN result, the reference RoI must have sufficient IoU with one of the proposed RoIs in more than 50 % of the frames. In contrast, for the average frame approaches, it depends on the largest cluster of the output RoI over all frames, and whether the average of this largest cluster has sufficient IoU with the reference RoI. Since the largest cluster might be created by less than half of the frames, this criterion is less strict. Pointing/RPN evaluation is performed with the 50 % criterion so that it is possible (for a perfect multimodal error correction system) to build the largest cluster which can fulfill the IoU threshold. If the component did not find the reference RoI on the first try, up to two further tries were done. In Table 2, the evaluation results of the RoI accuracy on the examples of the test dataset involving a correction are presented and in Figure 3 failure examples of the test dataset are depicted. A correct example of the test dataset is depicted in Figure 2. Again, the approach that averages the max. first 30 frames where a pointing gesture is detected performs best. For up to three tries, it has the best performance (45.33 % ($\tau = 0.25$), 38.31 % ($\tau = 0.5$), and 29.86 % ($\tau = 0.75$)) and for only one try and up to two tries, it performs comparable to the other approaches.

Our evaluation results for the complete output accuracy on the full test set are shown in Table 3. The numbers are better than the results for RoI accuracy on the test set restricted to examples involving a correction since our component in all cases detected the "no correction" label correctly. This is favorable, as it is better to cautiously han-

	IoU 0.25			IoU 0.5			IoU 0.75		
approach	1 try	$\leq$ 2 tries	$\leq$ 3 tries	1 try	$\leq$ 2 tries	$\leq$ 3 tries	1 try	$\leq$ 2 tries	$\leq$ 3 tries
middle frame	22.87 %	34.93 %	42.54 %	19.01 %	30.99 %	37.75 %	13.50 %	23.38 %	27.32 %
middle frame of the max. first 30 frames	22.59 %	33.80 %	40.00 %	18.73 %	28.73 %	34.08 %	15.43 %	23.66 %	26.48 %
averaged frames	22.31 %	34.37 %	43.38 %	18.46 %	30.14 %	37.46 %	14.33 %	23.66 %	28.73 %
averaged frames of the max. first 30 frames	22.59 %	35.77 %	45.35 %	18.18 %	30.70 %	38.31 %	14.60 %	25.07 %	29.86 %

Table 2: RoI accuracy on the examples of the test dataset involving a correction

	IoU 0.25			IoU 0.5			IoU 0.75		
approach	1 try	$\leq$ 2 tries	$\leq$ 3 tries	1 try	$\leq$ 2 tries	$\leq$ 3 tries	1 try	$\leq$ 2 tries	$\leq$ 3 tries
middle frame	34.76 %	42.44 %	47.63 %	31.83 %	39.50 %	44.24 %	27.77 %	34.09 %	36.79 %
middle frame of the max. first 30 frames	34.31 %	41.08 %	45.37 %	31.38 %	37.70 %	41.08 %	28.67 %	33.86 %	35.21 %
averaged frames	34.99 %	42.89 %	49.21 %	32.05 %	39.73 %	44.92 %	28.89 %	34.99 %	38.60 %
averaged frames of the max. first 30 frames	34.76 %	43.12 %	49.44 %	31.38 %	39.28 %	44.47 %	28.67 %	35.44 %	38.37 %

Table 3: Complete output accuracy on the full test dataset

dle corrections, in order not to correct something which is already fine. The error correction component should only improve the overall system and never decrease its performance. The best two approaches are the ones that use averaged frames, and both of them have similar results. Therefore, we can choose the averaged frames approach of the max. first 30 frames, since this is the more realistic scenario: A user wants to see results while doing the pointing gesture and not after finishing it. The averaged frames of the max. first 30 frames approach has a correctness of 34.76 % for only one try and 49.44 % for up to three tries with the IoU threshold of 0.25 and 28.67 % for only one try and 38.37 % for up to three tries (IoU threshold of 0.75).

7. Showcase Application

To demonstrate the use of our proposed system, we deployed it on a real-world steerable laser pointer robot. It is based on an open-source 3D-printed robot platform [22] and has two degrees of freedom (azimuth and altitude). Furthermore, we added an integrated camera and software-controllable laser pointer on the head. We implemented a simple control layer, that can approximately steer the head to given camera image coordinates and then turn the laser pointer on and off. That way, we can point at objects in the scene by passing the center coordinate of their detected RoI bounding box to the controller.

We deployed our proposed multimodal error correction system in a modified variant: instead of passing on results not including a correction to a downstream NLU component, we directly trained the NLU component into the same VL-T5 network. This is straightforward as we have only one intent the robot can perform ("point at some object") and thus only need to extract the relevant object. Visual grounding of extracted object names is done by a pre-trained YOLO network [40, 23] if the object's name is a known class. Otherwise, a simple feature clustering approach is utilized for incremental object learning based on the results of the multimodal error correction. A sample scenario can be seen in Figure 1.

8. Conclusions

In this work, we show that it is possible to combine natural language and pointing gestures to perform multimodal error correction. In particular, we present a system that identifies the name of an unknown or misclassified object as well as the corresponding RoI in the camera image. These two outputs can be used for incremental learning of new objects. If an example does not involve a correction, our system also detects this in 100 % of the test cases, and the corresponding utterances can be passed to another component or trained into the same network in future work. That means our proposed method adds value to an interactive robot system without decreasing the performance. The overall correctness results of 34.76 % on the first try and 49.44 % for up to three tries are a good step in the direction of a solid component. One major problem is that the pointing line generation and region proposal only finds the reference RoI in 58.40 % of the cases. To improve the performance of the pointing component, different region proposal approaches like using [26] should be tested. Furthermore, different multimodal large language models, like LLaVa [30], should be evaluated to decrease the gap between the upper bound and the actual results. In future work, we also want to use incremental output from automatic speech recognition instead of clean utterance pairs.

Acknowledgements. The research leading to these results has received funding from the Baden-Württemberg Ministry of Science, Research and the Arts (MWK) as part of the state's "digital@bw" digitization strategy in the context of the Real-World Lab "Robotics AI".

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the project OML (01IS18040A).

References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018. 5, 6

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, and Margaret Mitchell et al. VQA: Visual question answering. In *2015 IEEE International Conference on Computer Vision (ICCV)*, volume abs/1505.00468. IEEE, dec 2015. 2

[3] Tamim Asfour, Lukas Kaul, Mirko Wächter, Simon Ottenhaus, Pascal Weiner, Samuel Rader, Raphael Grimm, You Zhou, Markus Grotz, Fabian Paus, Dmitriy Shingarey, and Hans Haubert. Armar-6: A collaborative humanoid robot for industrial environments. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 447–454, 2018. 5

[4] Pablo Azagra, Javier Civera, and Ana C. Murillo. Incremental learning of object models from natural human–robot interactions. *IEEE Transactions on Automation Science and Engineering*, 17(4):1883–1900, 2020. 3

[5] Bita Azari, Angelica Lim, and Richard Vaughan. Commodifying pointing in hri: simple and fast pointing gesture detection from rgb-d images. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 174–180. IEEE, 2019. 2

[6] Michael Bett, Ralph Gross, Hua Yu, Xiaojin Zhu, Yue Pan, Jie Yang, and Alex Waibel. Multimodal meeting tracker. In *Proceedings of 6th International Conference on Computer-Assisted Information Retrieval (Recherche d'Information et ses Applications) (RIAO '00)*, pages 32–45, April 2000. 2

[7] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017. 2, 4, 5

[8] Yixin Chen, Qing Li, Deqian Kong, Yik Lun Kei, Song-Chun Zhu, Tao Gao, Yixin Zhu, and Siyuan Huang. YouRefIt: Embodied reference understanding with language and gesture. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1385–1395, 2021. 2

[9] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. Unifying vision-and-language tasks via text generation. In *ICML*, 2021. 2, 4, 5

[10] Stefan Constantin, Fevziye Irem Eyiokur, Dogucan Yaman, Leonard Bärmann, and Alex Waibel. Interactive multimodal robot dialog using pointing gesture recognition. In *European Conference on Computer Vision*, pages 640–657. Springer, 2022. 2, 3, 4, 6

[11] Stefan Constantin and Alex Waibel. Error correction and extraction in request dialogs. In *Proceedings of the 5th International Conference on Natural Language and Speech Processing (ICNLSP 2022)*, pages 2–11, 2022. 3

[12] Akansel Cosgun, Alexander JB Trevor, and Henrik I Christensen. Did you mean this object?: Detecting ambiguity in pointing gesture targets. In *HRI'15 Towards a Framework for Joint Action Workshop*, 2015. 2

[13] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision. *International Journal of Computer Vision*, 130(1):33–55, 2022. 5

[14] Shome S Das. A data-set and a method for pointing direction estimation from depth images for human-robot interaction and vr applications. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11485–11491. IEEE, 2021. 2

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NACL)*, 2019. 2

[16] Naina Dhingra, Eugenio Valli, and Andreas Kunz. Recognition and localisation of pointing gestures using a rgb-d camera. In *International Conference on Human-Computer Interaction*, pages 205–212. Springer, 2020. 2

[17] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023. 2

[18] Hartwig Holzapfel, Daniel Neubig, and Alex Waibel. A dialogue approach to learning object descriptions and semantic categories. *Robotics and Autonomous Systems*, 56(11):1004–1013, 2008. Semantic Knowledge in Robotics. 3

[19] Hartwig Holzapfel, Kai Nickel, and Rainer Stiefelhagen. Implementation and evaluation of a constraint-based multimodal fusion system for speech and 3d pointing gestures. In *Proceedings of the 6th International Conference on Multimodal Interfaces*, ICMI '04, page 175–182, New York, NY, USA, 2004. Association for Computing Machinery. 2

[20] Jun Hu, Zhongyu Jiang, Xionghao Ding, Taijiang Mu, and Peter Hall. Vgpn: Voice-guided pointing robot navigation for humans. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1107–1112. IEEE, 2018. 2

[21] Shruti Jaiswal, Pratyush Mishra, and GC Nandi. Deep learning based command pointing direction estimation using a single rgb camera. In *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pages 1–6. IEEE, 2018. 2

[22] JJRobots. Remotely controlled laser pointer robot. https://jjrobots.com/remotely-controlled-laser-pointer. 8

[23] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu , changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai.

ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements, Oct. 2020. 3, 5, 8

[24] Nebojsa Jojic, Barry Brumitt, Brian Meyers, Steve Harris, and Thomas 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)*, pages 468–475. IEEE, 2000. 2

[25] Roland Kehl and Luc Van Gool. Real-time pointing gesture recognition for an immersive environment. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pages 577–582. IEEE, 2004. 2

[26] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. 8

[27] Yuhui Lai, Chen Wang, Yanan Li, Shuzhi Sam Ge, and Deqing Huang. 3d pointing gesture recognition for human-robot interaction. In *2016 Chinese Control and Decision Conference (CCDC)*, pages 4959–4964. IEEE, 2016. 2

[28] Yang Li, Xiaoxue Chen, Hao Zhao, Jiangtao Gong, Guyue Zhou, Federico Rossano, and Yixin Zhu. Understanding embodied reference with touch-line transformer. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 5

[30] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. 2, 8

[31] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016. 5

[32] Anna CS Medeiros, Photchara Ratsamee, Jason Orlosky, Yuki Uranishi, Manabu Higashida, and Haruo Takemura. 3d pointing gestures as target selection tools: guiding monocular uavs during window selection in an outdoor environment. *ROBOMECH journal*, 8:1–19, 2021. 2

[33] Anna CS Medeiros, Photchara Ratsamee, Yuki Uranishi, Tomohiro Mashita, and Haruo Takemura. Human-drone interaction: Using pointing gesture to define a target object. In *Human-Computer Interaction. Multimodal and Natural Interaction: Thematic Area, HCI 2020, Held as Part of the 22nd International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II 22*, pages 688–705. Springer, 2020. 2

[34] Kai Nickel, Edgar Scemann, and Rainer Stiefelhagen. 3d-tracking of head and hands for pointing gesture recognition in a human-robot interaction scenario. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pages 565–570. IEEE, 2004. 2

[35] Kai Nickel and Rainer Stiefelhagen. Pointing gesture recognition based on 3d-tracking of face, hands and head orientation. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 140–146, 2003. 2

[36] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. 2

[37] Amit Kumar Pandey and Rodolphe Gelin. A mass-produced sociable humanoid robot: Pepper: The first machine of its kind. *IEEE Robotics & Automation Magazine*, PP:1–1, 07 2018. 5

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[39] Maria Pateraki, Haris Baltzakis, and Panos Trahanias. Visual estimation of pointed targets for robot guidance via fusion of face pose and hand orientation. *Computer Vision and Image Understanding*, 120:1–13, 2014. 2

[40] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2, 3, 5, 8

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 3

[42] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 3, 5

[43] Boris Schauerte and Gernot A Fink. Focusing computational visual attention in multi-modal human-robot interaction. In *International conference on multimodal interfaces and the workshop on machine learning for multimodal interaction*, pages 1–8, 2010. 2

[44] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9869–9878, 2020. 5

[45] Cheng Shi and Sibei Yang. Spatial and visual perspective-taking via view rotation and relation reasoning for embodied reference understanding. In *European Conference on Computer Vision*, pages 201–218. Springer, 2022. 2

[46] Dadhichi Shukla, Ozgur Erkent, and Justus Piater. Probabilistic detection of pointing directions for human-robot interaction. In *2015 international conference on digital image computing: techniques and applications (DICTA)*, pages 1–8. IEEE, 2015. 2

[47] Bernhard Suhm, Brad Myers, and Alex Waibel. Model-based and empirical evaluation of multimodal interactive error correction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, page 584–591, New York, NY, USA, 1999. Association for Computing Machinery. 3

[48] Bernhard Suhm, Brad A. Myers, and Alex Waibel. Multimodal error correction for speech user interfaces. *ACM Trans. Comput. Hum. Interact.*, 8(1):60–98, 2001. 3

[49] Hao Tan. Pytorch bottom-up attention with detectron2. https://github.com/airsplay/py-bottom-up-attention, 2019. 5

[50] Niket Tandon, Aman Madaan, Peter Clark, and Yiming Yang. Learning to repair: Repairing model output errors after deployment using a dynamic memory of feedback. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 339–352, Seattle, United States, July 2022. Association for Computational Linguistics. 3

[51] Sepehr Valipour, Camilo Perez, and Martin Jagersand. Incremental learning for robot perception through hri. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2772–2777, 2017. 3

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. 2

[53] Sagar Gubbi Venkatesh, Raviteja Upadrashta, Shishir Kolathaya, and Bharadwaj Amrutur. Teaching robots novel objects by pointing at them. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1101–1106, 2020. 3

[54] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 5, 6