Supplementary Materials for CAESAR: An Embodied Simulator for Generating Multimodal Referring Expression Datasets

Md Mofijul Islam, Reza Manuel Mirzaiee, Alexi Gladstone, Haley N. Green, Tariq Iqbal
School of Engineering and Applied Science, University of Virginia, Charlottesville, USA
{mi8uu, rmm3ya, abg4br, hng9vf, tiqbal}@virginia.edu

1 Access to Datasets, Source Codes, Benchmark Model Checkpoints, and Docker

The datasets we generated, source code for our simulator, benchmark learning models, trained model checkpoints, and simulator configuration guide can be accessed through the following links:

- **Project website:**
  `https://caesar-simulator.github.io`
- **CAESAR-XL dataset** (319 **GB):**
  `https://caesar-simulator.github.io/dataset.html`
- **CAESAR-L dataset** (181 **GB):**
  `https://caesar-simulator.github.io/dataset.html`
- **CAESAR-S dataset** (5.31 **GB):**
  `https://caesar-simulator.github.io/dataset.html`
- **Source code of data processing, and benchmark learning models:**
  `https://caesar-simulator.github.io/source_codes.html`
- **Docker for computing environment** (6.84 **GB):**
  `https://hub.docker.com/r/mmiakashs/pytorch_1-11_pl_1-6-1`
- **Source code of the CAESAR simulator** (11.8 **GB):**
  `https://caesar-simulator.github.io/source_codes.html`
- **The CAESAR simulator installation guide:**
  `https://drive.google.com/file/d/1_NixyzRAuedGy6U9Ngy14PkzGGgFqOay`
- **The CAESAR simulator configuration and data generation tool guide:**
  `https://youtu.be/KnKcpG7c2fk`

2 Source code and Datasets Accessibility

We used GitHub to host and publicly release all the source code for simulator, dataset parsing, and benchmark model experimentation. We released the future versions of our simulator through GitHub. We highly encourage other researchers to create a pull request to update the simulator, issue bugs, or resolve any known bugs. Additionally, researchers can request additional features by contacting us.

We host our datasets in two places: Google Drive and University of Virginia's secure storage system (Rivanna: https://www.rc.virginia.edu/userinfo/storage/). As the Rivanna storage system periodically backs the storage up, this storage is safer and ideal for long-term storage. We will share the datasets with other researchers via the Google drive links or provide read-access to the Rivanna storage system

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

using the Globus File sharing system (https://www.globus.org/data-sharing). Additionally, As our generated datasets are large in size, if researchers prefer, we can share the dataset via storage drives. However, the researchers need to bear a fixed charge to cover the cost of the storage drive and standard shipping. The authors of the paper maintain both of the generated datasets. Contact information: Md Mofijul Islam (Email: mi8uu@virginia.edu) and Prof. Tariq Iqbal (tiqbal@virginia.edu).

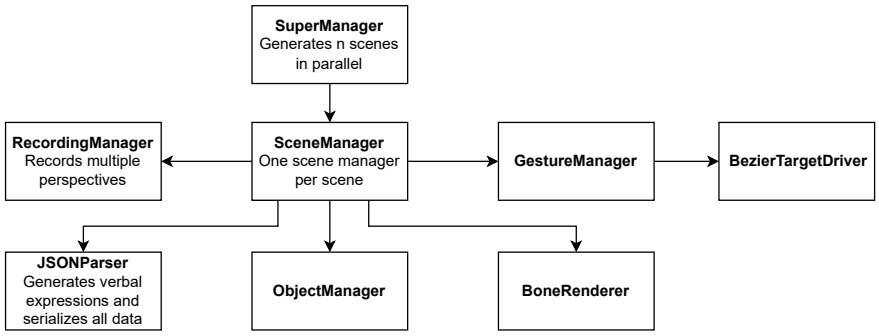


Figure 1: CAESAR simullator class diagram. One SuperManager instantiates different SceneManagers depending on the number of parallel scenes designated by the user. Each scene manager is then responsible for interactions between classes, managing different modules of the simulator including generating objects, generating human gestures, recording multiple views, generating verbal expressions, and serializing all labeled data.

3 CAESAR: An Embodied Simulator

3.1 CAESAR Simulator Structure

We visualize the general code structure of the CAESAR simulator in a class diagram shown in Fig. 1. During the data generation process in the CAESAR simulator, there is one SuperManager class that controls the start and stop of data generation as well as the spawning of each scene. The SuperManager class spawns the number of scenes designated to run in parallel, inputted by the user through the parameter *Parallel Scenes* in Table 1, where each scene is represented by a SceneManager class. Each SceneManager class controls all the other modules of a given scene, starting with the spawning of objects via the ObjectManager class. After all objects are spawned in, the SceneManager class activates the BoneRenderer class and starts recording from multiple views via the RecordingManager class. To start creating different nonverbal embodied referral expressions, the GestureManager class is called with varying parameters (gaze and gesture). The GestureManager class uses the BezierTargetDriver class to generate pointing gestures, using the hybrid approach described in Section 3.2 in the paper. While the human is moving, the BoneRenderer class continuously updates the bones for the skeletal pose view according to the human's joint positions. After all situations are created, the JSONParser class is called with all current scene data to serialize all information for the generated json as well as generate verbal embodied referral expressions according to what occurred in the scene. Please check the source code for further details regarding the way CAESAR simulator functions under the hood.

3.2 Human Avatars and Skeletal Poses

As one of the primary goals of our simulator CAESAR was to create a diverse dataset, we wanted our selection of humanoid characters to also promote diversity. Fig. 2 shows all humans used for data generation, where multiple different races are used and the gender split is $50/50$.

Many other works have had success in interpreting embodied instructions through the use of skeletal poses. For example, [1] used a PAF heatmap to identify human gestural patterns. To accomplish this, we have added a gestural pattern camera to our simulator allowing for ground truth skeletal pose

Figure 2: Human avatars used to generated nonverbal interactions in CAESAR simulator. Eight different human avatars were used with four males/females as well as several different races.

identification. This was accomplished by drawing lines between human bone joints using the Unity Vectrosity package [2], as normal Unity line renderers have several limitations. Cameras from all 3 views (ego, exo, and top) were copied and modified to cull all layers except the layer the skeletal pose was drawn on, effectively creating a skeletal pose camera. This can be seen in Fig 5, where the human is made obvious to be pointing and gazing at an object towards the right.

3.3 Object Library for the CAESAR-XL and CAESAR-L datasets

While developing the CAESAR simulator, we decided to create an object library based on objects generally found in household environments. Thus, most objects in the CAESAR-XL and CAESAR-L datasets are commonly found in a kitchen, office, bathroom, or living room. Additionally, a couple of unique objects were chosen to diversify further these datasets, such as a Decahedron or VR Headset. We display object libraries for both the CAESAR-L dataset as well as the CAESAR-XL dataset in Fig. 4 and Fig. 3 respectively. Note that the object library for the CAESAR-XL dataset (80 objects) is larger than the object library for the CAESAR-L dataset (61 objects) due to the removal of duplicate object categories as well as certain small objects that were too difficult to identify by humans.

All 80 objects in this library came from four purchased Unity Asset Store packages, that are linked below and covered under a Single Entity license.

- Furniture pack: `https://assetstore.unity.com/packages/3d/props/furniture/cabin-interior-household-items-furniture-pack-with-interactive-c-138278`
- Fruits and vegetables pack: `https://assetstore.unity.com/packages/3d/props/food/pbr-fruits-and-vegetables-hdrp-158808`
- Kitchen accessories pack: `https://assetstore.unity.com/packages/3d/props/interior/kitchen-accessories-200172`
- Gluttony pack: `https://assetstore.unity.com/packages/3d/props/food/supermarket-gluttony-pack-12042`

Additionally, we chose the object categories in this library carefully to ensure sufficient object diversity. Our object library contains objects we generally use in our daily life, such as kitchen and living room items, but also contains uncommon items, such as decahedrons and VR devices. Moreover, as the home and kitchen items are diverse and graspable, the model trained on these data samples can be transferred for robotic learning to implement human-robot interactive systems.

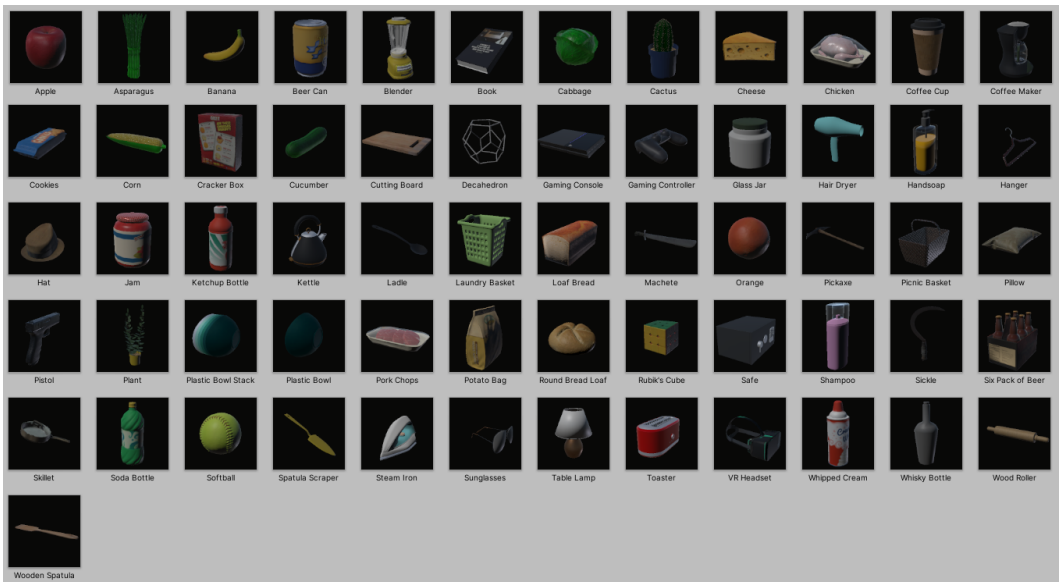


Figure 3: The CAESAR-L dataset Object Library, where a pool of 61 objects are used. All objects found in CAESAR-L are in CAESAR-XL, with the extra 19 objects in CAESAR-XL being different versions of objects as well as some objects that were commonly too small to identify with a smaller resolution.

Figure 4: The CAESAR-XL dataset Object Library, where a pool of 80 objects are used. Note that certain objects have multiple instances, such as pillow, cactus, and knife.

3.4 Bounding Box Annotation

As we can design the embodied referring expression comprehension task to identify a bounding box around the referred object [1], creating tight bounding boxes was a priority. Extensive testing was done to ensure bounding boxes were as accurate as possible from all three views. A naive approach for extracting object bounding boxes in Unity is to use the corners of an object's mesh renderers. However, our testing found that these bounding boxes tended to be overestimates for bounding boxes,

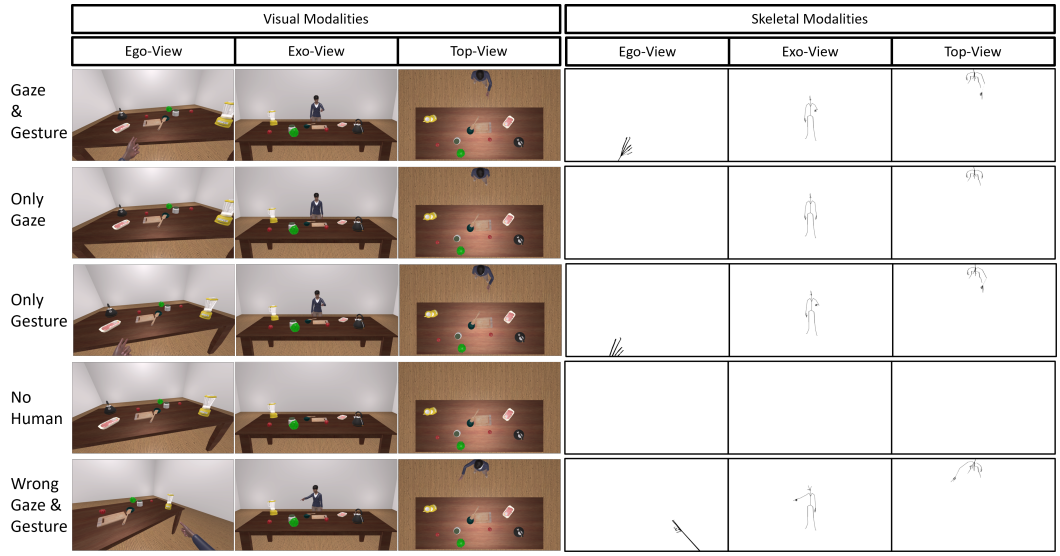


Figure 5: Generated data using our simulator CAESAR by varying non-verbal modalities (gaze and pointing gesture). Note that for the case of No Human there are no skeletal poses as there is no human in the scene.

especially those from the ego view, as the ego camera was not aligned by global scene coordinate axes. Therefore the vertices of every object was checked relative to each camera to compute bounding boxes. This created much tighter and accurate bounding boxes.

This process was repeated for all three views to properly identify every object's location from every view. For both the exo and top view the cameras remain stationary, meaning this computation only needed to be performed once. However, the ego view camera constantly changes position and rotation according to where the human gazes, meaning these bounding boxes had to be computed dynamically. We decided to use Unity's physics synchronized update function for synchronizing bounding box calculations with video frames, since Unity's physics engine runs at a constant frame rate regardless of the frame rate of the engine.

3.5 Ambiguous Sample Generator

Ambiguous samples, which differ from samples with contrasting instructions, are samples where the given modalities are not enough to completely identify the referred object. These were generated through supplying verbal utterances to the model in the *No Human* scenario. For example, the scenario shown in Fig. 5 along the *No Human* row supplied with the verbal instruction "the appl" would be deemed as ambiguous as it is impossible to determine which apple is"the apple" without knowing more information. However, the verbal instruction "The red apple to the left of the black kettle" would be deemed non-ambiguous as there is only one apple next to the cutting board.

3.6 Additional Pointing Gesture Information

Our hybrid pointing gesture approach allows us to generate novel motion it has no reference to, since it is unconstrained by a motion-capture library. It also ensures that generated motion obeys observed patterns in motion-capture data, like acceleration, gesture timing, and arc-like arm paths. Our model is based on the five-phase gesture model described in Kendon et. al's work [3], with gesture phases including rest, preparation, stroke, hold, and retraction phases. Finally, unlike many classical physical simulation algorithms, our simulator does not rely on overtly-complex mathematical modeling of musculo-skeletal forces, which both significantly would increase computational time, as well as diminish our control over the resulting motion, which often results in "physically valid but unrealistic solutio" [4].

Algorithm 1: Data generation procedure in CAESAR simulator

Input: T: Total sample to generate; S: Number of parallel scenes designated in data generation settings
Output: D: Generated data sample

$D \leftarrow \emptyset$ ▷ Generated dataset to empty
for $i \leftarrow 1$ *to* T **do**
    LoadAssets() ▷ Load different assest, including objects, virtual avatars, etc.)
    GenerateObjects() ▷ Randomly spawn objects at different locations
    PrepareDifferentCamerasAndViews() ▷ Initialize cameras to capture nonverbal interactions.
    for $s \in S$ **do**
        PerformGazeOrGesture() ▷ Render gaze and pointing gestures based on the referred object location.
        CreateContrastiveSituation() ▷ Generated contrastive data samples.
        AnnotateImagesAndVideos()
    end
    $D \leftarrow D \cup$ ParserData() ▷ Parse data to json file to record data annotations.
end
return Generated dataset, D

3.7 Scalable Data Generation

We wanted the CAESAR simulator to effectively use a resource-intensive cluster computing system to generate data at scale. Additionally, Unity supports multithreaded rendering. Thus, supporting parallel data generation within our simulator would allow users to configure the speeds by which they generate data based on their computing system. Allowing for configurable parallelization also allows for scalable data generation. To accomplish this, parallel data generation within our simulator is done by spawning another data generation scene with all necessary objects while other data generation scenes are running.

4 Dataset Generation Procedure

Data generation of a given data sample begins through the creation of a scene's environment, with the floor, table, human, and walls being dynamically loaded. Proceeding this, all objects are loaded in based on the methods described in the paper (Section 3.1). Next, data generation for the eight different scenarios [1] starts generating a human gaze or gesture (Section 3.2 in the paper). Moreover, we generate contrastive samples based on the procedure described in the paper (Section 3.4). While data is generating videos are recorded (if selected to in the CAESAR settings), canonical frames are saved, and bounding boxes are computed and stored. Once all situations have been completed, all labels are passed into a JSON parser, responsible for creating the string to serialize into a JSON. During this process the verbal utterance templates are used to generate utterances describing the referred objects (Section 3.2 in the paper). This process is done in parallel according to the number of times designated by the user in the CAESAR settings, discussed more in Section 5. The entire data generation process is visualized in algorithm 1.

5 Configurable Data Generation Interface

While designing the simulator we wanted users to be able to configure settings of datasets they generate according to a variety of different model tasks. Thus, we have created an interface for configuring different settings of CAESAR via the inspector tab in Unity. To allow for environmental configuration, CAESAR dynamically loads all passed in humans/objects/floors/tables, so users can directly configure these. Additionally, we made various image setting configurations possible, various modalities possible to record/not record, and added general usability features. All configurable settings as well as a description for each setting can be found in Table 1. Additionally, we have

[1] There are four synced scenarios where a verbal utterance and non-verbal gesture refer to the same object: Using gaze, pointing gesture, and verbal utterance; Using gaze and verbal utterance, Using pointing gesture and verbal utterance; Using verbal utterance and no human avatar. There are four more contrastive scenarios for the four described synced scenarios, where the verbal utterance and non-verbal gesture refer to different objects (note that for no human avatar as there is no non-verbal gesture an object not within the scene is referenced.

Parameter	Explanation of Parameter
Object Pool	List of different objects to load into simulator. Does not use objects dragged into this field (loads all objects in Resources/objects folder) unless designated to by user in the *Dynamically Load Object* parameter.
Humans	List of different humans to choose at random.
Tables	List of different tables to choose at random.
Floor Materials	Different floor materials to choose at random.
Parallel Scenes	Number of scenes generating data concurrently.
Total Scenes Generated	Number of scenes to generate total.
Dynamically Load Objects	If toggled will dynamically load objects from those dragged into the *Object Pool* parameter.
Record	If toggled records video rather than just recording single canonical frames for each situation.
Activate Skeletal Camera	If toggled will activate skeletal pose cameras.
Draw Bounding Boxes	If toggled will draw bounding boxes onto canonical frames (as a diagnostic).
Png or Jpg	If toggled will save images/videos as png's, else will save images/videos as jpg's.
Use Different Directory	If toggled will use a different directory specified in the *Directory* parameter.
Directory	The absolute directory to generate data in, start with a / do not end with a /.
Width	The width of images to generate, must match the width specified in the resolution of the game tab inside Unity.
Height	The Height of images to generate, must match the Height specified in the resolution of the game tab inside Unity.
Origin Top L Or Bottom L	If toggled will generate bounding box coordinates from the top left, otherwise it will generate coordinates from the bottom left.
Use Preloaded Scene	If toggled will not dynamically generate objects and a humanoid but will create nonverbal signals from the preloaded scene, not used to generate data.

Table 1: Data generation interface with configurable parameters and descriptions.

prepared a video demonstrating how to configure parameters as well as use CAESAR to generate data: `https://youtu.be/KnKcpG7c2fk`.

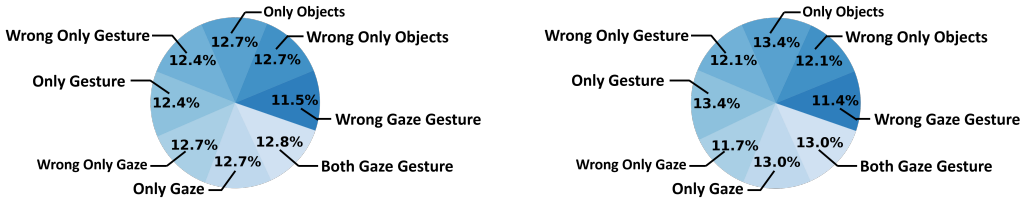
6 CAESAR-XL and CAESAR-L Dataset Analyses

In total there are 11,617,626 scenes in the CAESAR-XL dataset and 841,620 scenes in the CAESAR-L dataset. For each scene, we sample three verbal templates from 13 templates to generate verbal utterances. In total, each scene in the CAESAR-XL dataset (as this dataset only contains RGB images and no video) contains 15 images, with 3 images of each 4 nonverbal interactions settings (human using gaze and gestures, human using only gaze, human using only gestures, human using wrong gaze and gestures) as well as 3 images with no human settings. The number of images in the CAESAR-L dataset varies due to variable length videos, but the number of still canonical frames is 30 due to the inclusion of a skeletal pose modality. The number of generated frames for the CAESAR-L dataset (which was recorded at 15 FPS), ranges from 65 to 102.

As the task addressed in this paper was the *embodied spatial relation grounding* task, we needed to generate contrastive situations 50% of the time. Fig. 6 shows this, where contrastive situations represent about 50% of all samples (note that the model was trained on a $50/50$ contrastive to non-contrastive sample ratio). In total, there are eight different situations contained within this task, which were discussed in the paper in Sections 3.3 and 3.6.

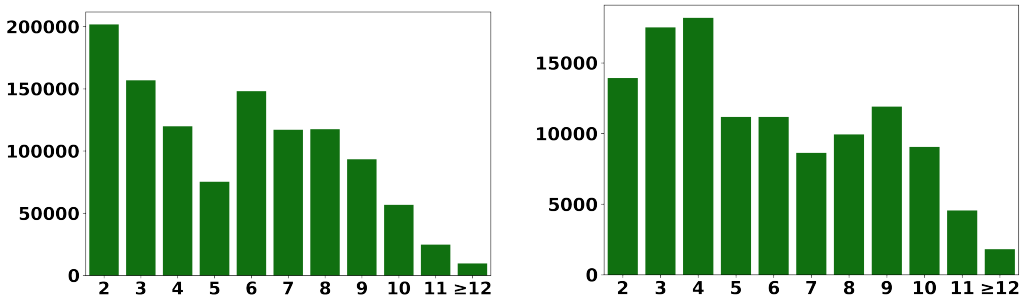
6.1 Object Category and Verbal Expression

We visualize the lengths of verbal utterances and the most frequent object categories used in our generated datasets CAESAR-XL and CAESAR-L (Fig. 8 & 7). In Fig. 8, all the object categories are used a similar number of times except the Knife, Cactus, and Pillow categories in the CAESAR-XL dataset. These three object categories had multiple instances with different attributes (size and color)



(a) CAESAR-XL (b) CAESAR-L

Figure 6: Distribution of different situations in the CAESAR-XL and CAESAR-L datasets.



(a) CAESAR-XL dataset. (b) CAESAR-L dataset.

Figure 7: Frequency distribution of verbal utterances lengths in the CAESAR-XL and CAESAR-XL datasets.

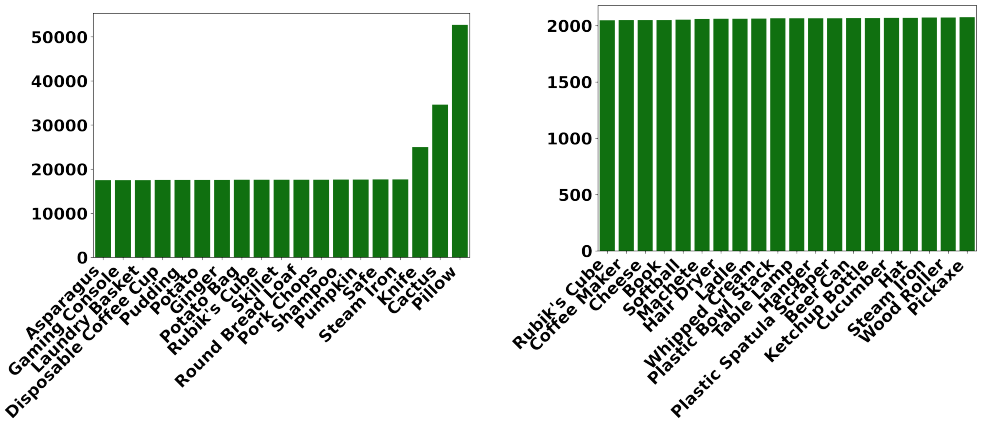
in the object library used to generate the CAESAR-XL dataset (Fig. 3). Similarly, all object categories are used close to the same number of times in the CAESAR-L dataset. As an additional verbal expression visualization, Fig. 9 shows the most common words of all verbal utterances found in the CAESAR-XL dataset through a wordcloud, where the largest words are the words appearing most frequently. Moreover, the worldcloud visualization indicates that the most frequent words describe sizes of objects, spatial relations, spatial locations, and colors.

6.2 Spatial Location and Relation Visualizations

One of the goals of CAESAR was to reduce the spatial bias from previous datasets with respect to spatial location/relation terms, as addressed in Section 4 in the paper. Particularly, we wanted to ensure models were learning the perspective-taking and nonverbal signals necessary to perform in real-world embodied situations. Thus, object locations plotted with respect to spatial location/relational terms in verbal utterances that can be used ambiguously from multiple perspectives should be non-separable, to ensure models do not bias verbal utterances or 2D visual cues during training (Fig. 10, 11,12, 13). Aligning with this, Figs. 10 & 12 show visualizations for different terms when specifying object spatial locations, where the only locational term with separable points being the spatial location "center". This makes sense as the center is the only locational term used in CAESAR that is objective from both the exo and ego perspectives. Similarly, Figs. 11 & 13 show visualizations for different terms when specifying object spatial relations, where no terms provide a separable pairs of points. Please note that the term "Next to" is not compared to any other terms as it is a standalone term used in all locations with no complements, and thus does not lead to model bias or point separability.

7 Human-Subject Study

We conducted a study on Amazon Mechanical Turk (MTurk) to evaluate the model. 300 data samples were collected in which participants looked at an image from the exo view and answered a survey question on their observations. All participants were located in the United States and at least 18 years of age or older. Participants were also required to have a Human Intelligence Task (HIT) approval rating approval rating of 95% or greater. Moreover, to obtain a variety of participant responses, participants were limited to three tasks. Participants were compensated $0.02 for the 10 second task.



(a) CAESAR-XL dataset. (b) CAESAR-L dataset.

Figure 8: Most referred object categories in the CAESAR-XL and CAESAR-XL datasets.

(a) CAESAR-XL dataset (b) CAESAR-L dataset

Figure 9: A wordcloud for all verbal expressions in the CAESAR-XL and CAESAR-L datasets, where the size of words represents their relative frequencies. The most frequent words describe sizes of objects, colors, and spatial relations/locations.

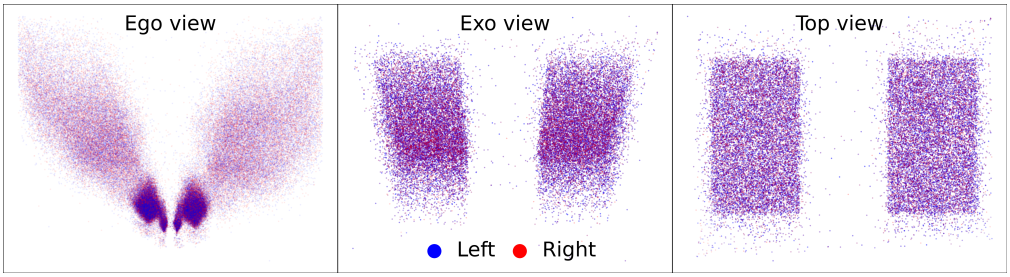
To generate the task, we uploaded a CSV file containing the links for the 300 exo view images to MTurk. The CSV file also contained the verbal descriptions for the images and whether the verbal and non-verbal descriptions were in contrast. MTurk randomized the order in which the images were shown. We performed three trials of the study on MTurk, so each image was shown to three participants for a total of 900 evaluations.

The participants were first instructed to review the electronic study information document for consent and the task instructions. In the task, participants were asked to determine whether the person was verbally and non-verbally (pointing, gaze, etc.) describing the same object. Prior to the task, participants were shown examples for cases in which the person *was* and *was not* verbally and non-verbally identifying the same object (Fig. 14).

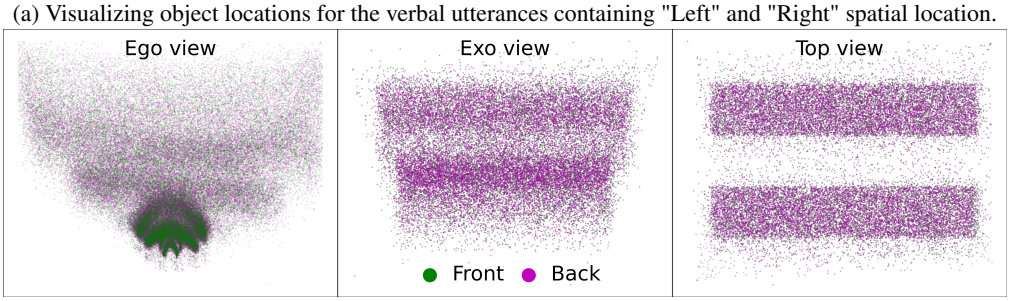
During the task, participants were shown the exo view image, the verbal instruction, and a reference image containing the potential objects (Fig. 15). After completing the task, participants were debriefed and compensated. The results of the study suggest that the participants correctly validated the relations 80.66% of the time.

8 Embodied Relation Grounding Models

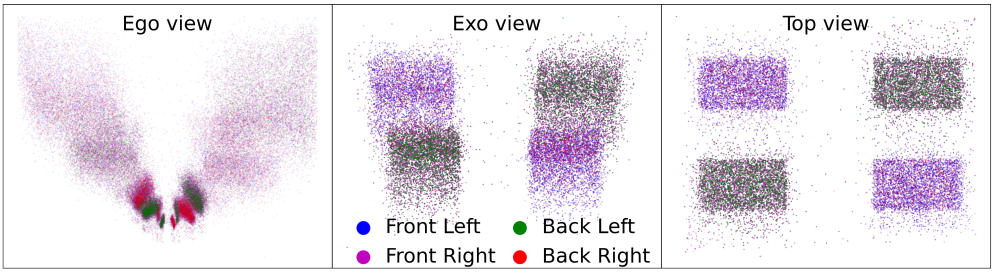
We have adopted visual-language models to develop three representation learning models for the embodied spatial relation grounding task: a CLIP Model [5], a Dual-Encoder (ViT [6] + BERT [7]) model and a Late Fusion (ResNet [8] + BERT [7]) model (Fig. 16).



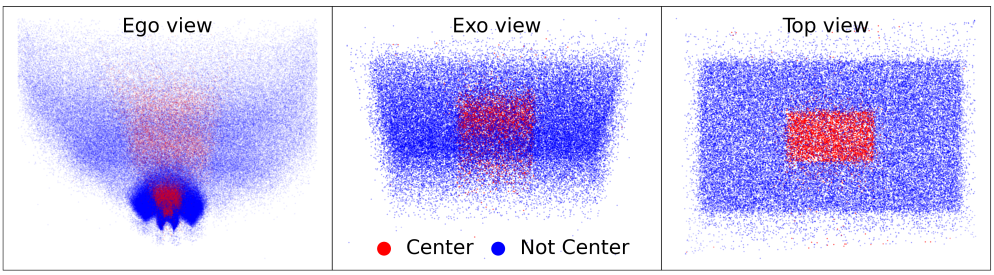
(a) Visualizing object locations for the verbal utterances containing "Left" and "Right" spatial location.



(b) Visualizing object locations for the verbal utterances containing "Front" and "Back" spatial location.



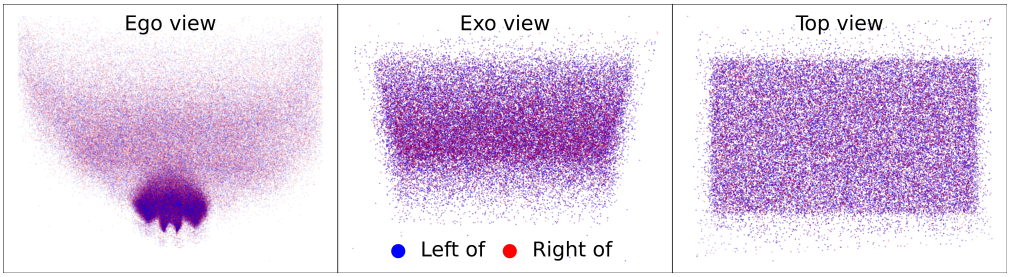
(c) Visualizing object locations for the verbal utterances containing four corner spatial locations.



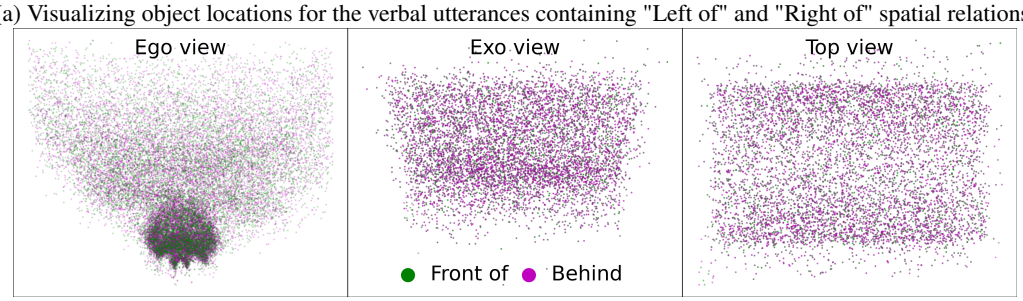
(d) Visualizing object Locations containing "center" term vs. all other spatial location terms.

Figure 10: Analysis of object spatial locations throughout different terms in verbal utterances in the CAESAR-XL dataset. These visualizations show that the contrastive spatial locations in verbal utterances, such as *left* and *right*, are not separable based on the object locations in visual modalities. This property of our dataset ensures the model does not benefit from exploiting 2D visual cues and verbal modalities to ground embodied spatial relations, thus ensuring the model learns the perspective-taking necessary for real world embodied situations. Therefore, using our datasets, we can train a model that can effectively learn to attend to the salient portion of nonverbal signals to ground embodied spatial relations.

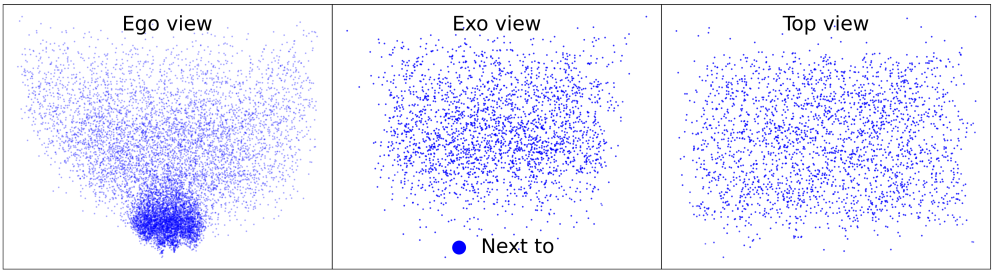
CLIP-based Model: The CLIP model excels at aligning visual and language modalities [5]. Thus, we use the CLIP model to detect whether nonverbal cues and verbal utterances of an embodied expression refer to the same object. The CLIP model architecture processes an image-text pair and produces verbal and visual representations. For this reason, we pair the verbal expression, T, to each of the views of the nonverbal expression (Ego (V_{ego}), Exo (V_{exo}), and Top (V_{top})) and pass each modality pair to CLIP models: $E_i^v, E_i^t = CLIP(V_i, T), i \in (ego, exo, top)$. Here, $E_i^v \in \mathbb{R}^{B \times S}$ and $E_i^t \in \mathbb{R}^{B \times S}$ are the visual and verbal embeddings from CLIP models, respectively (B is the



(a) Visualizing object locations for the verbal utterances containing "Left of" and "Right of" spatial relations.



(b) Visualizing object locations for the verbal utterances containing "Front of" and "Behind" spatial relations.



(c) Visualizing object locations for the verbal utterances containing "Next to" spatial relations.

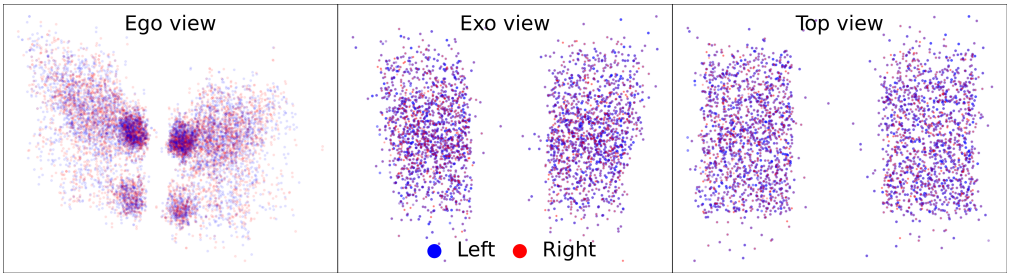
Figure 11: Analyses of spatial relations given through verbal utterances in the CAESAR-XL dataset. These visualizations show that contrastive spatial relations, such as *left of* and *right of*, are not separable based on object locations in the visual modalities.

batch size and S is the embedding dimension). The adopted CLIP model architecture is depicted in Fig. 16(a).

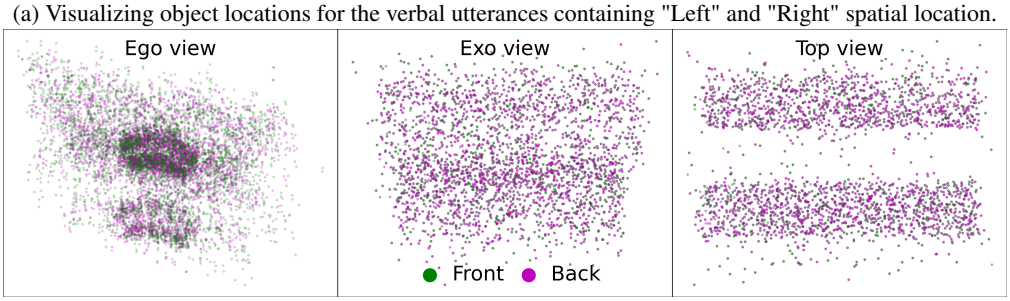
We used a Huggingface library [9] to implement the CLIP model. We also used the CLIPProcessor to tokenize the input verbal data as well as process the visual data and produce the input tensors. Finally, these processed data were fed into the CLIP model to extract verbal and visual embeddings. The verbal embedding is the projected pooled output of a CLIPTextModel. Similarly, The visual embedding is the projected pooled output of a CLIPVisionModel. Both verbal and visual embeddings were a tensor of length 512.

Dual-Encoder Model: Similar to CLIP models, we pass each pair of visual and verbal modalities to Dual-Encoder models to extract verbal and nonverbal representations (Fig. 16(b)). We used ViT and BERT in a Dual-Encoder model to encode visual and verbal modalities, respectively. The ViT model (*google/vit-base-patch16-224*)we used was pretrained on a Huggingface library. We also used a BERT pretrained model (*bert-base-uncased*) from Huggingface. This Dual-Encoder Model is pretrained to align visual-text embeddings using a CLIP like contrastive image-text training approach for zero-shot training tasks, such as image classification and image retrieval. The verbal embedding is the projected pooled output of the language model portion of the Dual-Encoder model. Similarly, The visual embedding is the projected pooled output of the vision model aspect of the Dual-Encoder model. Both verbal and visual embeddings are a tensor with length 768.

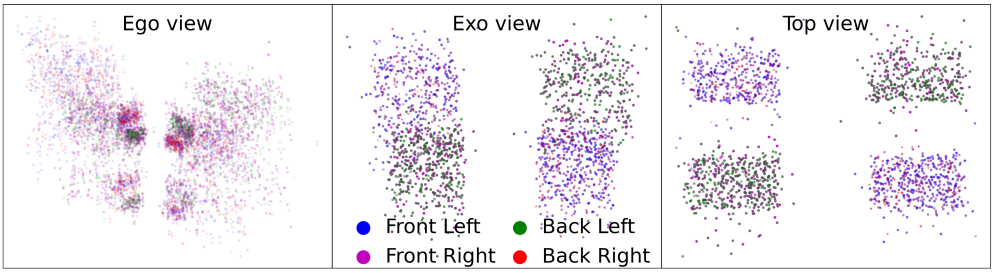
Late Fusion Model: In the Late Fusion model all the visual and verbal modalities are encoded independently using ResNet-50 [8] and BERT [7] language models. First, we independently encode



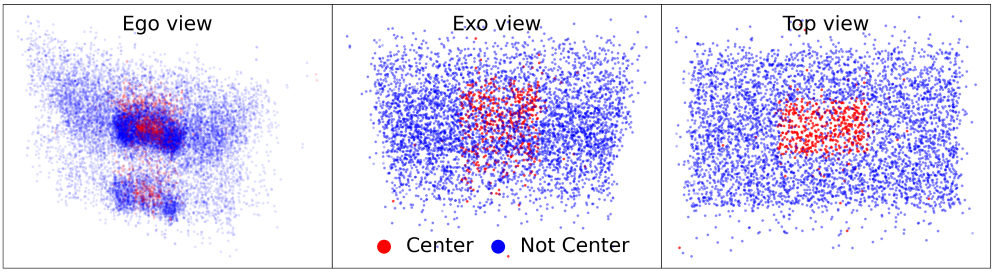
(a) Visualizing object locations for the verbal utterances containing "Left" and "Right" spatial location.



(b) Visualizing object locations for the verbal utterances containing "Front" and "Back" spatial location.



(c) Visualizing object locations for the verbal utterances containing four corner spatial locations.

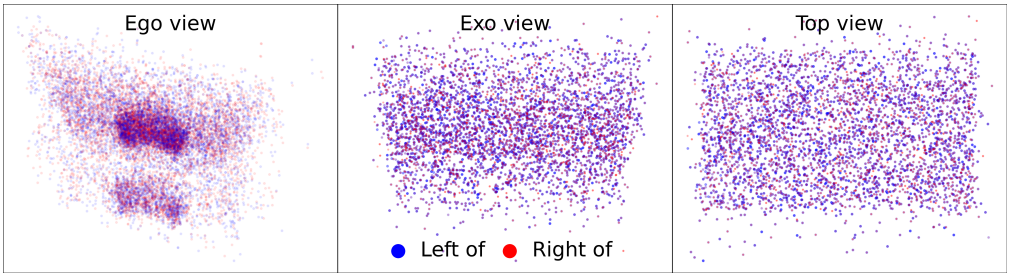


(d) Visualizing object Locations containing "center" term vs. all other locations.

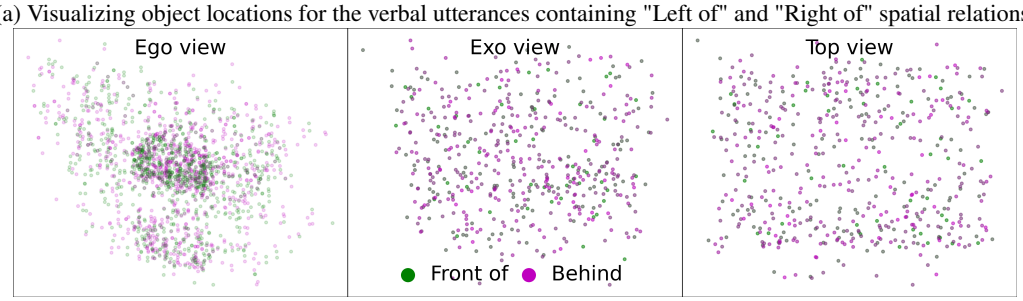
Figure 12: Analyses of spatial locations through verbal utterances in the CAESAR-L dataset.

all the visual data modalities using a pretrained ResNet-50 model on ImageNet. Second, we encode verbal data modalities using a pretrained BERT model (*bert-base-uncased*) from a Huggingface pretrained library. We projected the extracted verbal and visual modalities representations to a fixed-sized embedding of 768.

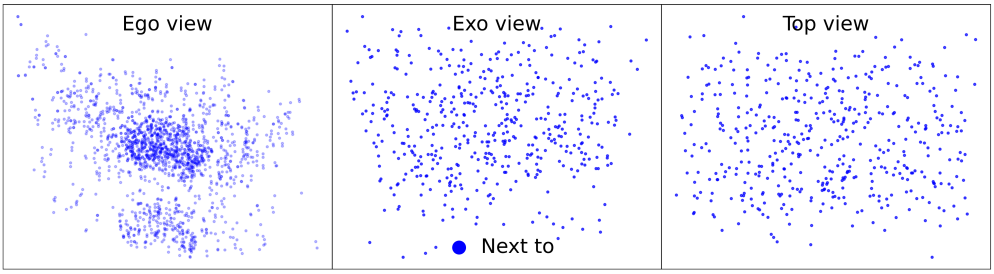
Multimodal Fusion: We fused the extracted verbal and visual representations from the above-mentioned models to produced multimodal representations, which are used to detect embodied spatial relations. We used four fusion approaches: SUM, CONCAT, Self-Attention, and Cross-Attention. The first two fusion approaches summed and concatenated the verbal and visual representations. The self-attention approach is similar to the transformer-style self-attention [10] which attends each of the verbal and visual representations and sums the attended representations. In our implementation of the self-attention model, we used 4 heads and single layers of multi-head attention models from PyTorch. We also used a dropout of 0.1 in this cross-attention model. We have also employed a Cross-Attention



(a) Visualizing object locations for the verbal utterances containing "Left of" and "Right of" spatial relations.



(b) Visualizing object locations for the verbal utterances containing "Front of" and "Behind" spatial relations.



(c) Visualizing object locations for the verbal utterances containing "Next to" spatial relations.

Figure 13: Analyses of spatial relations given through verbal utterances in the CAESAR-L dataset.

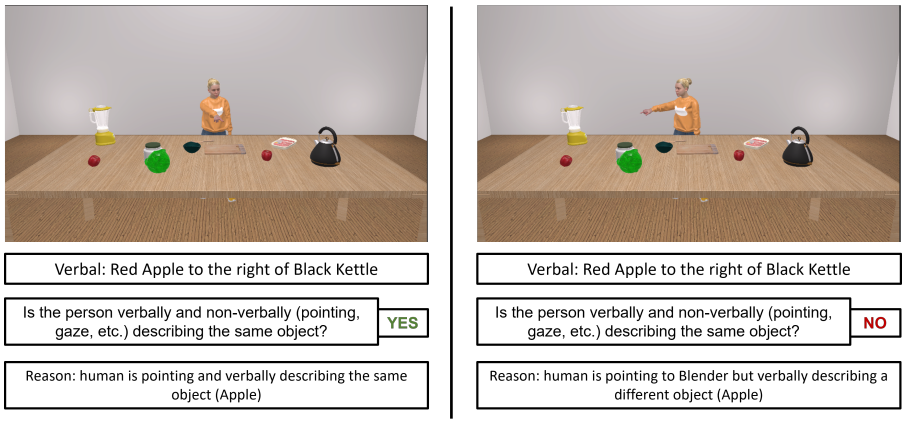


Figure 14: Instructions given to the Amazon Mechanical Turk participants.

approach, which is similar to the co-attention approach from ViLBERT [11]. Cross-Attention is essentially a query-key-value style attention approach, where verbal embeddings are used as queries and visual embeddings are used as keys and values. In our implementation of the cross-attention model, we used 4 heads and a single layer of multi-head attention model from PyTorch, with a dropout of 0.1 in this cross-attention model. Finally, the fused embedding is passed through a multilayer perceptron to detect embodied spatial relations.

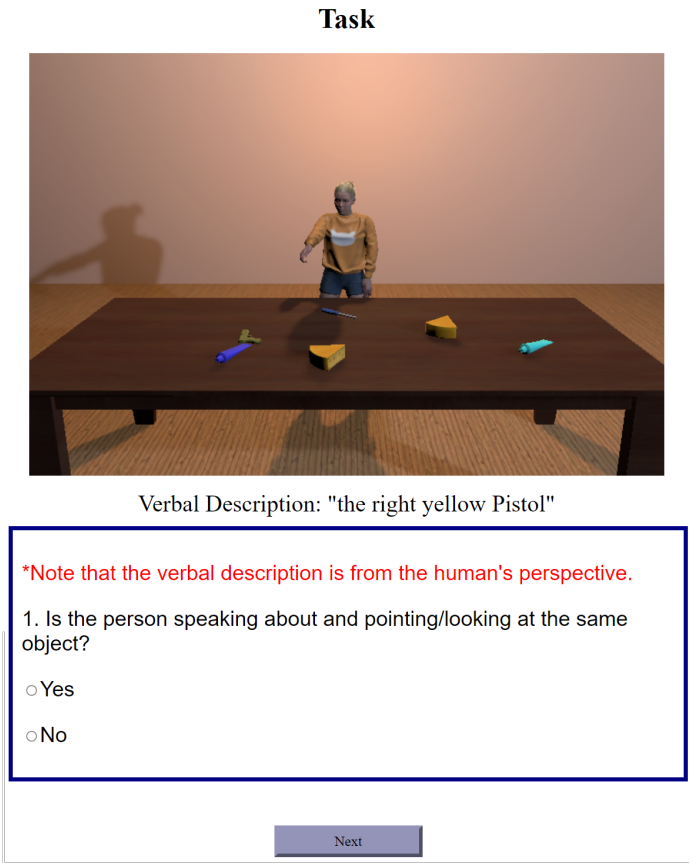


Figure 15: A sample task shown to the Amazon Mechanical Turk participants.

Model Training Environment Setup: We projected the visual and verbal embeddings from CLIP, Dual-Encoder, and Late Fusion models to 512, 768, and 768 sized embeddings, respectively. We fused all the embeddings from multiple views and passed them through a multilayer perceptron to classify whether verbal and nonverbal expressions referred to the same object. We used the PyTorch-1.7 and PyTorch-Lightning-1.0.8 deep learning frameworks to implement models for training on the CAESAR-XL dataset. We used the Adam optimizer with weight decay regularization, and cosine annealing warm restarts [12] with an initial learning rate set to $3e^{-4}$ to train the evaluated approaches. To train the learning model, we set both the cycle length (T_0) and cycle multiplier (T_{mult}) to 2. We used a batch size 8 to train the model. We trained each evaluated model for 4 epochs on the CAESAR-XL dataset, using cross-entropy loss. We trained all the models in a distributed GPU cluster environment, where each node contains 4-8 GPUs. Please check the source code for further details of the model implementations. We have also released a docker container to replicate the execution environment (6.84 GB):
`https://hub.docker.com/r/mmiakashs/pytorch_1-11_pl_1-6-1`.

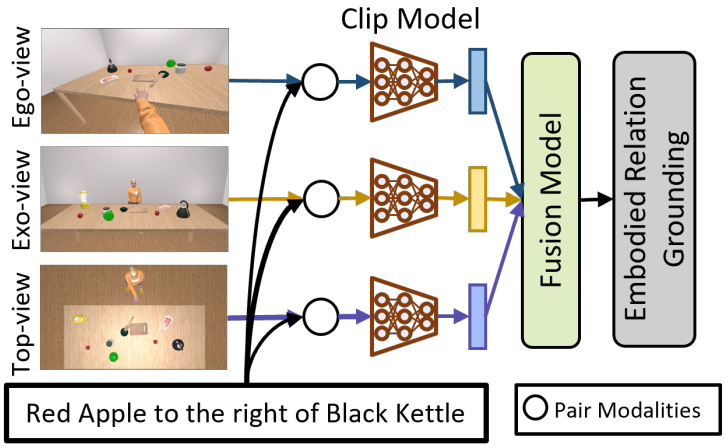
References

[1] 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.

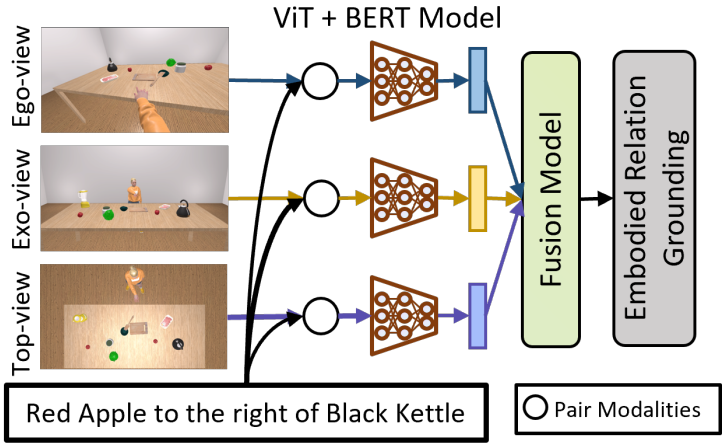
[2] Unity vectrosity package. `https://assetstore.unity.com/packages/tools/particles-effects/vectrosity-82`. Accessed: 2022-06-03.

[3] Adam Kendon. Gesture. *Annual Review of Anthropology*, 26:109–128, 1997. ISSN 00846570. URL `http://www.jstor.org/stable/2952517`.

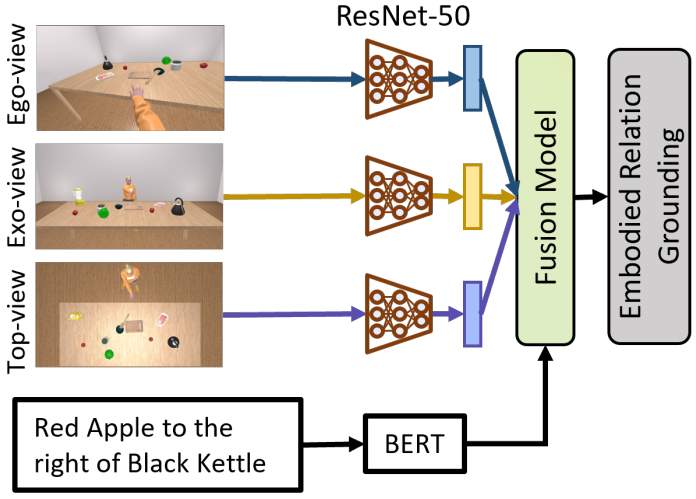
[4] Cheng-Yun Liu and Zoran Popovic. Towards a generative model of natural motion. 2005.



(a) Clip Model



(b) Dual Encoder (ViT+BERT)



(c) Late Fusion Model (ResNet + BERT)

Figure 16: (a) Clip model takes a pair of image, capturing nonverbal interactions, and verbal descriptions of the referred object. The clip model produces a pair of visual and verbal representations for each pair of image and verbal data. All the extracted representations are fused to produce multimodal representations for embodied relation grounding. (b) The Dual-Encoder takes a pair of image and verbal descriptions and produces a pair of verbal and visual representations, similar to the Clip models. However, Dual-Encoder model uses ViT and BERT models to encode visual and verbal modalities. (c) Late Fusion model independently encodes visual modalities using ResNet-50 model. Late Fusion model encodes verbal modalities using a pre-trained BERT model.

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[9] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, pages 5999–6009, 2017. ISSN 10495258.

[11] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019.

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2017.