(2.5+1)D Spatio-Temporal Scene Graphs for Video Question Answering

Anoop Cherian **Chiori Hori** **Tim K. Marks** **Jonathan Le Roux**

Mitsubishi Electric Research Labs (MERL), Cambridge, MA
{cherian, chori, tmarks, leroux}@merl.com

Abstract

Spatio-temporal scene-graph approaches to video-based reasoning tasks, such as video question-answering (QA), typically construct such graphs for every video frame. These approaches often ignore the fact that videos are essentially sequences of 2D "views" of events happening in a 3D space, and that the semantics of the 3D scene can thus be carried over from frame to frame. Leveraging this insight, we propose a (2.5+1)D scene graph representation to better capture the spatio-temporal information flows inside the videos. Specifically, we first create a 2.5D (pseudo-3D) scene graph by transforming every 2D frame to have an inferred 3D structure using an off-the-shelf 2D-to-3D transformation module, following which we register the video frames into a shared (2.5+1)D spatio-temporal space and ground each 2D scene graph within it. Such a (2.5+1)D graph is then segregated into a static sub-graph and a dynamic sub-graph, corresponding to whether the objects within them usually move in the world. The nodes in the dynamic graph are enriched with motion features capturing their interactions with other graph nodes. Next, for the video QA task, we present a novel transformer-based reasoning pipeline that embeds the (2.5+1)D graph into a spatio-temporal hierarchical latent space, where the sub-graphs and their interactions are captured at varied granularity. To demonstrate the effectiveness of our approach, we present experiments on the NExT-QA and AVSD-QA datasets. Our results show that our proposed (2.5+1)D representation leads to faster training and inference, while our hierarchical model showcases superior performance on the video QA task versus the state of the art.

Introduction

Recent advances in deep learning have made it possible to think beyond individual domains, such as computer vision and natural language processing, and consider tasks that are at their intersections. Visual question answering (VQA) is one such task that has witnessed a significant attention lately (**???????**). While earlier approaches to this task used holistic visuo-textual representations (**??**), it was found that decomposing a visual scene into its constituents (and their relationships) provided a better reasoning pipeline (**????**), perhaps because of the possibility for an easier disentanglement of the scene objects relevant to the given question. Such a disentanglement naturally leads to a graph representation of the scene, usually called a scene graph (**?**). Using such a graph representation allows one to use the powerful machinery of graph neural networks for VQA, and has demonstrated significant promise (**????**).

While visual scene graphs were originally proposed for image-based tasks, there have been direct adaptations of this data structure for video-based reasoning problems (**????**). Usually, in such problems, scene graphs are constructed for every video frame, followed by an inter-frame representation learning to produce holistic video level features for reasoning. However, having scene graphs for every frame may be redundant and could even become computationally detrimental for longer video sequences. Taking a step back, we note that videos are essentially 2D views of a 3D space in which various events happen temporally, and representing the scene in a 4D spatio-temporal space could thus potentially avoid such representational redundancies. Furthermore, object properties such as permanence (**?**) could be handled more effectively in a 3D space, as each object (that is visible in some video frame) gets a location therein, thereby disentangling the camera views from its spatial location (**??**). Using such a 3D representation thus would provide a natural way to avoid occlusions, which is a significant problem when working with 2D scene graphs.

Motivated by the above insight, we explore a novel spatio-temporal scene graph representation, where the graph nodes are not grounded on individual video frames, instead are mapped to a shared 3D world coordinate frame. While there are approaches in computer vision that could produce such a common 3D world (**?**), such methods usually assume: i) that the scene is static, without dynamic objects, ii) that the camera calibration information is known, or iii) that multiple overlapping views of the same scene are available; none of which may exist for arbitrary (internet) videos typically used in VQA tasks. Fortunately, there have been several recent advancements in 3D reconstruction from 2D images, such as (**??**); these works take as input an image and produces a realistic pseudo-3D structure for the image scene, typically called a 2.5D image. For every video frame, we leverage such a 2.5D reconstruction to impart an approximate 3D location for each graph node, thereby producing a (2.5+1)D spatio-temporal scene graph.

A technical challenge with the above (2.5+1)D scene graph representation is that each graph is still specific to a video frame, and is not registered to a shared space. Such

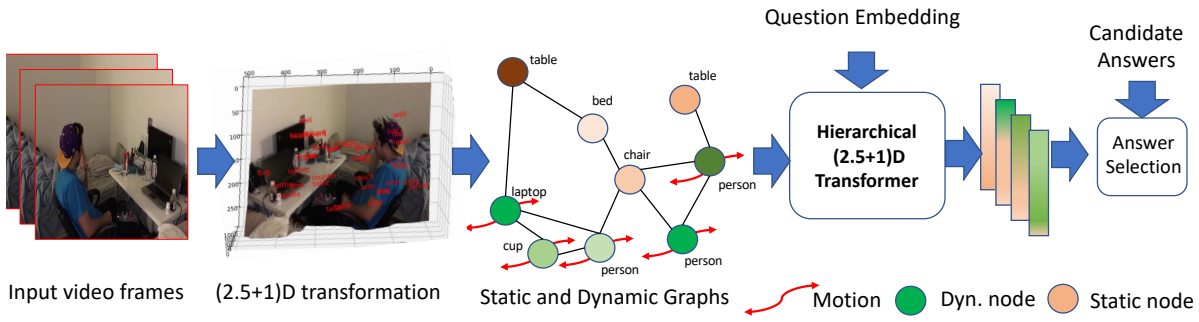


Figure 1: A schematic illustration of our proposed (2.5+1)D video QA reasoning pipeline.

a registration is confounded by the fact that objects in the scene may move from frame to frame. To this end, we propose to: (i) split the (2.5+1)D scene graph into a static 2.5D sub-graph and a dynamic (2.5+1)D sub-graph, depending on whether the class of the underlying scene graph node usually moves in scenes (e.g., a *person* class is dynamic, while a *table* class is considered static), (ii) merge the graph nodes corresponding to the static sub-graph based on their 3D spatial proximity across frames, thereby removing the node redundancy, and (iii) retain the nodes of the dynamic sub-graph from the original scene graph. As the dynamic sub-graph nodes are expected to not only capture the frame-level semantics, but to also potentially involve object actions (e.g., person *picking* a bottle), we enrich each dynamic graph node with motion features alongside its object-level feature representation. Thus, our proposed (2.5+1)D scene graph representation approximately summarizes the spatio-temporal activity happening in a scene in a computationally efficient framework.

The use of such a (2.5+1)D graph representation allows for developing rich inference schemes for VQA tasks. For example, to capture the interaction of a person with a static object in the scene, the inference algorithm needs to attend to regions in the (2.5+1)D graph where the spatio-temporal proximity between the respective graph nodes is minimized. Leveraging this intuition, we propose a hierarchical latent embedding of the (2.5+1)D graph where the graph edges are constructed via varied spatio-temporal proximities, thereby capturing the latent embeddings of the graph at multiple levels of granularity. We use such a graph within a Transformer reasoning pipeline (**?**) and conditioned on the VQA questions to retrieve the correct answer.

To validate the effectiveness of our approach, we present experiments on two recent video QA datasets, namely: (i) the NExT-QA dataset (**?**) and (ii) the QA task of the audio-visual scene aware dialog (AVSD) dataset (**?**). Our results on these datasets show that our proposed framework leads to about 4x speed up in training, while pruning 25–50% graph nodes, and showcases superior QA accuracy against recent state-of-the-art methods.

Related Work

We note that visual question answering has been a very active research area in the recent times, and thus interested readers may refer to excellent surveys, such as (**??**). In the following, we restrict our literature review to prior methods that are most similar to our contributions.

Scene graphs for QA: Since the seminal work of (**?**) in using scene graphs as a rich representation of an image scene, there have been extensions of this idea for video QA and captioning tasks (**?????**). Spatio-temporal scene graphs are combined with a knowledge distillation objective for video captioning in (**?**). Similarly, video scene graphs are combined with multimodal Transformers for video dialogs and QA in (**?**). In (**?**), a graph alignment framework is proposed that uses graph co-attention between visual and language cues for better video QA reasoning. In (**?**), a multi-step reasoning pipeline is presented that attends to visual and textual memories. We note that scene graphs have been explored for various action recognition tasks as well. For example, video action graphs are presented in (**???**). Action Genome (**??**) characterizes manually annotated spatio-temporal scene graphs for action recognition. In contrast to these prior methods, we seek a holistic and potentially minimal representation of a video scene via pseudo-3D scene graphs for the QA task.

3D scene graphs: Very similar to our motivations towards a comprehensive scene representation, 3D scene graphs have been proposed in (**?**). However, their focus is on efficient annotation and collection of such graphs from 3D sensors. Similarly, more recent efforts such as (**??**) are also targeted at improvising the efficiency of constructing a 3D scene graph from RGBD scans, while our focus is on constructing pseudo-3D graphs leveraging recent advancements in 2D-to-3D methods. We note that while precise 3D scene graphs may be important for several tasks such as robot navigation or manipulation, they need not be required for reasoning tasks such as what we consider in this paper, and for such tasks approximate 3D reasoning may be sufficient. We also note that 3D graphs have been explored for video prediction tasks in (**?**), however in a very controlled setting.

Graph Transformers: Similar to our contribution, connections between graphs and Transformers have been explored previously. For example, (**?**) has explored long-range atten-

tion using kernelized Transformers, while (**?**) presents a kernel view of Transformer attention, and Bello presents a long range attention using lambda layers that captures both position and content interactions (**?**). While there are some similarities between these works and ours in the use of kernels and positional details in computing the similarity matrix within a Transformer, our objective and goals are entirely different from these works. Specifically, our proposed architecture is to represent a pseudo-3D scene at multiple levels of spatio-temporal granularity for a reasoning task, which is entirely different from the focus of these prior works.

Proposed Method

In this section, we first present our setup for constructing (2.5+1)D scene graphs for a given video sequence, then explain our hierarchical spatio-temporal Transformer-based graph reasoning pipeline. See Fig. 1 for an overview of our framework.

Problem Setup

We assume that we have access to a set of N training video sequences, $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$, where the i-th video consists of n_i frames. In the following, we eliminate the subscripts for simplicity, and use S to denote a generic video sequence from $\mathcal{S}$ that has n frames. We assume that each video S is associated with at least one question Q, which is an ordered tuple of words from a predefined vocabulary (tokenized and embedded suitably). We define the task of video QA as that of retrieving a predicted answer, A_{pred}, from a collection of ℓ possible answers, $\mathcal{A} = \{A_1, A_2, \ldots, A_\ell\}$. Of these ℓ answers, we denote the ground-truth answer as A_{gt}. We propose to represent S as a (2.5+1)D spatio-temporal scene graph. The details of this process are described in the following subsections.

2D Scene Graph Construction

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a scene graph representation of a video sequence S of length n frames, where $\mathcal{V} = V_1 \cup V_2 \cup \cdots \cup V_n$ denotes the set of nodes, each V_t denotes the subset of nodes associated with frame t, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of graph edges (which are computed as part of our hierarchical Transformer framework explained later). To construct the scene graph $\mathcal{G}$, we follow the standard pipeline using an object detector. Specifically, we first extract frames from the video sequence and pass each frame as input to a Faster R-CNN (FRCNN) object detection model (**?**). The FRCNN implementation that we use is pre-trained on the Visual Genome dataset (**?**) and can thus detect 1601 object classes, which include a broad array of daily-life indoor and outdoor objects. In every frame, the FRCNN model detects m objects, each of which is represented by a graph node v that contains a tuple of FRCNN outputs $(f_v^o, c_v, \text{bbox}_v)$, where f_v^o is the object's neural representation, c_v is its label in the Visual Genome database, and bbox_v denotes its bounding box coordinates relative to the respective frame. Thus, for a video sequence with n frames, we will have mn graph nodes.[1] However, as alluded to above, several of these graph nodes may be redundant, thus motivating us to propose our (2.5+1)D scene graphs.

(2.5+1)D Scene Graphs

Suppose $D\colon \mathbb{R}^{h\times w\times 3} \to \mathbb{R}^{h\times w\times 4}$ denotes a neural network model that takes as input an RGB image and produces as output an RGBD image, where the depth is estimated. For a video frame (image) I, further let $d_I : \mathbb{R}^2 \to \mathbb{R}^3$ map a 2D pixel location (x, y) to a respective 3D coordinate, denoted $p = (x, y, z)$. To implement D, we use an off-the-shelf pre-trained 2D-to-3D deep learning framework. While there are several options for this network (**??**), we use the MiDAS model (**?**), due to its ease of use and its state-of-the-art performance in estimating realistic depth for a variety of real-world scenes. For a scene graph node $v \in V_t$ extracted from video frame t (image I_t), let $\overline{\text{bbox}}_v$ denote the centroid of the node's detected bounding box. To enrich the scene graph with (2.5+1)D spatio-temporal information, we expand the representation of node v to include depth and time by updating the tuple for v to be $(f_v^o, c_v, \text{bbox}_v, p_v, t)$, where $p_v = d_{I_t}(\overline{\text{bbox}}_v)$ can be interpreted as the 3D centroid of the bounding box. We denote the enriched graph as $\mathcal{G}_{3.5D}$.

Static and Dynamic Sub-graphs

While the nodes in $\mathcal{G}_{3.5D}$ are equipped with depth, they are still grounded in every video frame, which is potentially wasteful. This is because many of these nodes may correspond to objects in the scene that seldom move in the real world. If we can identify such objects, then we can prune their redundant scene graph nodes. To this end, we segregated the Visual Genome classes into two distinct categories, namely (i) a category $\mathcal{C}_s$ of static scene objects, such as *table*, *tree*, *sofa*, *television*, etc., and (ii) a category $\mathcal{C}_d$ of dynamic objects, such as *people*, *mobile*, *football*, *clouds*, etc. While the visual appearance of a static object may change from frame to frame, we assume that its semantics do not change and are sufficient for reasoning about the object in the QA task. However, such an assumption may not hold for a dynamic object, such as a *person* who may interact with various objects in the scene, and each interaction needs to be retained for reasoning. Using this class segregation, we split the graph $\mathcal{G}_{3.5D}$ into two distinct scene graphs, $\mathcal{G}_s$ and $\mathcal{G}_d$, corresponding to whether the object label c_v of a node $v \in \mathcal{V}$ belongs to $\mathcal{C}_s$ or $\mathcal{C}_d$, respectively.

Our next subgoal is to register $\mathcal{G}_{3.5D}$ in a shared 3D space. There are two key challenges in such a registration, namely that (i) the objects in the scene may move, and (ii) the camera may move. These two problems can be tackled easily if we extract registration features only from the static sub-graph nodes of the frames. Specifically, if there is camera motion, then one may find a frame-to-frame 3D projection

[1] While this may appear not too big a memory footprint, note that each visual feature f_v^o is usually a 2048D vector. Thus, with $m = 36$, videos of length $n \approx 50$ frames, and a batch size of 64, we would need about 15GB of GPU memory for forward propagation alone.

Algorithm 1: Identifying common ancestors for merging

for $v_1 \in V_1^s$ **do**
  $\text{ancestor}(v_1) := v_1$
for $t = 2$ to n **do**
  for $v_t \in V_t^s$ **do**
    if $\text{match}(v_t)$ exists **then**
      $\text{ancestor}(v_t) := \text{ancestor}\big(\text{match}(v_t)\big)$

matrix using point features, and then use this projection matrix to spatially map all the graph nodes (including the dynamic nodes) into a common coordinate frame.

While this setup is rather straightforward, we note that the objects in the static nodes are only defined by their bounding boxes, which are usually imprecise. Thus, to merge two static nodes, we first consider whether the nodes are from frames that are sufficiently close in time, with the same object labels, and with the intersection over union (IoU) of their bounding boxes above a threshold γ. Two nodes $v_t, v_{t'} \in \mathcal{G}_s$, from frames with timestamps $t \neq t'$ (where $|t - t'| < \delta$) are candidates for merging if the following criterion C is met:

$$\text{C}(v_t, v_{t'}) := \big(c_{v_t} = c_{v_{t'}}\big) \wedge \text{IoU}(\text{bbox}_{v_t}, \text{bbox}_{v_{t'}}) > \gamma. \quad (1)$$

If a static node v_t has multiple candidate nodes in the previous δ frames that satisfy criterion (1), the candidate with the nearest 3D centroid is selected as the matching node that will be merged:

$$\text{match}(v_t) = \underset{\substack{v_{t'} \in V^s_{t-\delta} \cup \cdots \cup V^s_{t-1} \\ \text{such that } \text{C}(v_t, v_{t'}) = 1}}{\arg\min} \big\| p_{v_t} - p_{v_{t'}} \big\|, \quad (2)$$

where $V_t^s = \{v_t \in V_t \mid v_t \in \mathcal{G}_s\}$ denotes the set of all static nodes from frame t. Since (2) chooses the best match from the past δ frames, rather than just from frame $t-1$, it can tolerate more noise in the estimates of the depth and the bounding boxes associated with the graph nodes.

We can apply this matching process recursively in order to determine larger equivalence classes of matched nodes to be merged, where an equivalence class is defined as the set of all nodes that share a single common ancestor. We accomplish this by looping over the frames t in temporal order, where for each node v_t for which $\text{match}(v_t)$ exists, we assign $\text{ancestor}(v_t) = \text{ancestor}\big(\text{match}(v_t)\big)$. This procedure is detailed in Algorithm 1. Finally, for each ancestor, all nodes that share that ancestor are merged into a single node. The feature f_v^o associated with the new node v is obtained by averaging the features from all of the nodes that were merged into it. We use the 3D coordinate p of the parent node for all the child nodes that are merged into it. Let $\mathcal{G}_{s'}$ denote the new reduced version of $\mathcal{G}_s$ after each equivalence class of matched nodes has been merged into one node.

Motion Features

To recap, so far we have segregated the graph $\mathcal{G}_{3.5D}$ into $\mathcal{G}_s$ and $\mathcal{G}_d$, where the nodes of $\mathcal{G}_s$ have been pruned and registered into a common shared 3D space to form an updated graph $\mathcal{G}_{s'}$, while the spatial locations of the nodes in the dynamic graph $\mathcal{G}_d$ have been updated via transformation matrices produced from $\mathcal{G}_{s'}$ into the same coordinate frame. An important step missing in our framework is that the dynamic sub-graph that we have constructed so far is still essentially a series of graph nodes produced from FRCNN, which is a static object detection model – that is, the nodes are devoid of any *action* features that are perhaps essential in capturing how a dynamic node *acts* within itself and on its environment (defined by the static objects). To this end, we propose to incorporate motion features into the nodes of the dynamic graph. Specifically, we use the I3D action recognition neural network (**?**), pre-trained on the Kinetics-400 dataset to produce convolutional features from short temporal video clips. These features are then ROI-pooled using the (original) bounding boxes associated with the dynamic graph nodes. Suppose $f^a_{v_t} = \text{ROIPool}(\text{I3D}(s_t), \text{bbox}_{v_t})$, where s_t denotes a short video clip around the t-th video frame of a video S, then we augment the FRCNN feature vector by concatenating the object and action features as $f^{oa}_v \leftarrow f^o_v \,||\, f^a_v$, for all $v \in V_d$, where $||$ denotes feature concatenation.

Hierarchical Graph Embedding

Using our (2.5+1)D scene graph thus constructed, we are now ready to present our video QA reasoning setup. As the questions in a QA task may need reasoning at various levels of abstraction (e.g., Q: *what is the color of a person's shirt?* A: *red*, Q: *why did the boy cry?*: A: *Because the ball hit him*, etc.), we decided to design our reasoning pipeline so that it can capture such a hierarchy. To set the stage, let us review a few basics on Transformers in our context. In the sequel, we assume the set of nodes in $\mathcal{G}_{3.5D}$ is given by: $V' = V_{s'} \cup V_d$.

Transformers: Suppose $F \in \mathbb{R}^{r \times |V'|}$ denotes a matrix of features computed from the static and dynamic graph nodes of a video S via projecting their original features into latent spaces of dimensionality r using multi-layer perceptrons, MLP_s and MLP_d; i.e., $F = \text{MLP}_s(f^o_{V_{s'}}) \,||\, \text{MLP}_d(f^{oa}_{V_d})$. If $\mathbf{Q}^i_F, \mathbf{K}^i_F, \mathbf{V}^i_F \in \mathbb{R}^{r_k \times |V'|}$ denote the i-th k-headed query, key, and value embeddings of F respectively, where $r_k = r/k$, then a multi-headed self-attention Transformer encoder produces features F' given by:

$$F' := \Big\|_{i=1}^{k} \text{softmax}\left(\frac{\mathbf{Q}^i_F {\mathbf{K}^i_F}^\top}{\sqrt{r_k}}\right)\mathbf{V}^i_F. \quad (3)$$

(2.5+1)D-Transformer: We note that in a standard Transformer described in (3), each output feature in F' is a mixture embedding of several input features, as decided by the similarity computed within the softmax. Our key idea in (2.5+1)D-Transformer is to use a similarity defined by the spatio-temporal proximity of the graph nodes as characterized by our (2.5+1)D scene graph. For two nodes $v_1, v_2 \in V'$, let a similarity kernel κ be defined as:

$$\kappa(v_1, v_2 | \sigma_S, \sigma_t) = \exp\left(-\frac{\|p_{v_1} - p_{v_2}\|^2}{\sigma_S^2} - \frac{\|t_{v_1} - t_{v_2}\|_1}{\sigma_T}\right), \quad (4)$$

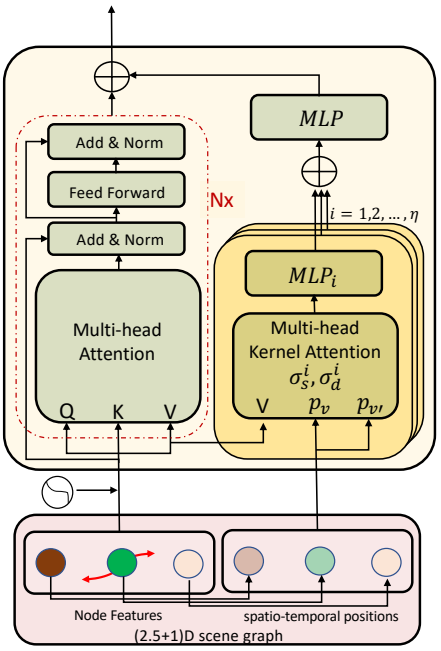


Figure 2: The architecture of the proposed Hierarchical (2.5+1)D-Transformer for encoding (2.5+1)D scene graphs. The left module in red (N×) is the standard Transformer.

capturing the spatial-temporal proximity between v_1 and v_2 for scales σ_S and σ_T for spatial and temporal cues, respectively. Then, our (2.5+1)D-Transformer is given by:

$$F'_{3.5D} := \Big\|_{i=1}^{k} \text{softmax}\,\mathbf{K}(V', V'|\sigma_S, \sigma_T)\mathbf{V}_F^i, \qquad (5)$$

where we use $\mathbf{K}$ to denote the spatio-temporal kernel matrix constructed on V' using (4) between every pair of nodes. Such a similarity kernel merges features from nodes in the graph that are spatio-temporally nearby – such as for example, *a person interacting with an object*, or the dynamics of objects in $\mathcal{G}_d$. Further, the kernel is computed on a union of the graph nodes in $\mathcal{G}_{s'}$ and $\mathcal{G}_d$, and thus directly captures the interactions between the static and dynamic graphs.

Hierarchical (2.5+1)D-Transformer: Note that our (2.5+1)D-Transformer in (5) captures the spatio-temporal features at a single granularity as defined by σ_S and σ_T. However, we can improve this representation towards a hierarchical abstraction of the scene graph at multiple granularities. Let σ_S^j, σ_T^j, $j = 1, \ldots, \eta$ be a set of scale, and let $\text{MLP}_j, j = 1, \ldots, \eta$ be a series of multilayer perceptrons, then combining (3) and (5), we define our hierarchical (2.5+1)D-Transformer producing features $F^H_{3.5D}$ as:

$$F^H_{3.5D} = \sum_{j=1}^{\eta} \underset{j}{\text{MLP}} \Big\|_{i=1}^{k} \left(\text{softmax}\,\mathbf{K}(V', V'|\sigma_S^j, \sigma_T^j)\mathbf{V}_F^i\right). \qquad (6)$$

In words, (6) computes spatial-temporal kernels at various bandwidths and merges the respective scene graph node features and embeds them into a hierarchical representation space via the MLPs. In practice, we find that it is useful to combine the kernel similarity in (6) with the feature similarity in (3) and add (6) with (3) (after an MLP) to produce the final graph features. Figure 2 shows the architecture of the proposed Transformer.

Question Conditioning: For the video QA task, we first use a standard Transformer architecture in (3) to produce multi-headed self-attention on the embeddings of the given question. This step precedes attending the encoded questions on $F^H_{3.5D}$ via a multi-headed cross-attention Transformer, followed by average pooling to produce question-conditioned features $F^Q_{3.5D}$. In this case, the source to the cross-Transformer is the set of $F^H_{3.5D}$ features, while the target sequence corresponds to the self-attended question embeddings.

Training Losses: To predict an answer A_{pred} for a given video S and a question Q, we use the question-conditioned (2.5+1)D-Transformer features produced in the previous step and compute its similarities with the set of candidate answers. Specifically, the predicted answer is defined as $A_{\text{pred}} = \text{softmax}({F^Q_{3.5D}}^\top \lambda(\mathcal{A}))$, where $\lambda(\mathcal{A})$ represents embeddings of candidate answers. For training the model, we use cross-entropy loss between A_{pred} and the ground truth answer A_{gt}. Empirically, we find that rather than computing the cross-entropy loss against one of ℓ answers, if we compute the loss against $b \times \ell$ answers produced via concatenating all answers in a batch, that produced better gradients and training. Such a concatenation is usually possible as the text answers for various questions are often different.

Experiments

In this section, we provide experiments demonstrating the empirical benefits of our proposed representation and inference pipeline. We first review the datasets used in our experiments, following which we describe in detail our setup, before presenting our numerical and qualitative results.

Datasets

We used two recent video QA datasets for evaluating our task, namely NExT-QA (**?**) and AVSD-QA (**?**).

NExT-QA Dataset is a very recent video question answering dataset that goes beyond traditional VQA tasks, and incorporates a significant number of *why* and *how* questions, that often demand higher level abstractions and semantic reasoning about the videos. The dataset consists of 3,870 training, 570 validation, and 1,000 test videos. The dataset provides 34,132, 4,996, and 8,564 multiple choice questions in the training, validation, and test sets respectively, and the task is to select one of the five candidate answers. As the test video labels are with-held for online evaluation, we report performances on the validation set in our experiments. We use the code provided by the authors of (**?**) for our experiments, which we modified to incorporate our Transformer pipeline.

AVSD-QA Dataset is a variant of the Audio-Visual Scene Aware Dialog dataset (**?**), repurposed for the QA task. The dataset consists of not only QA pairs, but also provides a human generated conversation history, and captions for each

video. In the QA version of this dataset, the task is to use the last question from dialog history about the video to select an answer from one of a hundred candidate answers. The dataset consists of 11,816 video clips and 118,160 QA pairs, of which we follow the standard splits to use 7,985, 1,863, and 1,968 clips for training, validation, and test. We report the performances on the test set. For this dataset, we used an implementation that is shared by the authors of (**?**) and incorporated our modules.

Experimental Setup and Training

Visual Features: As mentioned earlier, we use public implementations for constructing our scene graphs and (2.5+1)D graphs, these steps being done offline. Specifically, the video frames are sub-sampled at fixed 0.5 fps for constructing the scene graphs using Faster RCNN, and each frame is processed by a MiDAS pre-trained model[2] for computing the RGBD images. The FRCNN and depth images are then combined in a pre-processing stage for pruning the scene graph nodes as described earlier. Out of 1600 object classes in the Visual Genome dataset, we classified 1128 of the classes as *dynamic* and used those for constructing the dynamic scene graph. Next, we used the I3D action recognition model (**?**) to extract motion features from the dynamic graph nodes. For this model, we used the videos at their original frame rate, but averaged the spatio-temporal volumes via conditioning on the pruned FRCNN bounding boxes for every dynamic object anchored at the frame corresponding to the frame rate used in the object detection model. This setup produced 2048D features for the static graph nodes and (2048+1024)D features for the dynamic graph nodes. These features are then separately projected into a latent space of 256 for NExT-QA and 128 dimensions for AVSD-QA datasets on which the Transformers operate.

Text Features: For the NExT-QA dataset, we use the provided BERT features for every question embedding. These are 768D features, which we project into 256D latent space to be combined with our visual features. Each candidate answer is concatenated with the question, and BERT features are computed before matching them with the visual features for selecting the answer. For NExT-QA, we also augment the BERT features with the recent CLIP features (**?**) that are known to have better vision-language alignment. For AVSD-QA, we used the provided implementation to encode the question and the answers using an LSTM into a 128D feature space. We used the same LSTM to encode the dialog history and the caption features; these features are then combined with the visual features using multi-headed shuffled Transformers as suggested in (**?**).

Evaluation Protocol: We used the classification accuracy on NExT-QA, while we use mean retrieval rank on the AVSD-QA dataset; the latter measure ranks the correct answer among the selections made by an algorithm and reports the mean rank over the test set. Thus, a lower mean rank suggests better performance.

Training Details: We use an Adam optimizer for training both the models. For NExT-QA, we used a learning rate of 5e-5 as suggested in the paper with a batch size of 64 and trained for 50 epochs, while AVSD-QA used a learning rate of 1e-3 and a batch size of 100, and trained for 20 epochs.

[2]https://pytorch.org/hub/intelisl_midas_v2/

Table 1: NExT-QA: Comparisons to the state of the art. Results for the various competitive methods are taken from (**?**).

Method	Accuracy (%)↑
Spatio-Temporal VQA (**?**)	47.94
Co-Memory-QA (**?**)	48.04
Hier. Relation n/w (**?**)	48.20
Multi-modal Attn VQA (**?**)	48.72
graph-alignment VQA (**?**)	49.74
(2.5+1)D-Transformer (ours)	**53.40**

Hyperparameters: There are two key hyperparameters in our model, namely (i) the number of spatial abstraction levels in the hierarchical Transformer, and (ii) the bandwidths for the spatio-temporal kernels. We found that for the NExT-QA dataset, a four layer hierarchy with $\sigma_S \in \{0.01, 0.1, 1, 10\}$ showed the best results, while for AVSD-QA, we used $\sigma_S \in \{1, 10\}$. As for the temporal scale, we divided the frame index t_v by the maximum number of video frames in the dataset (making the temporal span of the video to be in the unit interval), and used $\sigma_T = \sigma_S$. We found that using a larger number of hierarchical levels did not change the performance for NExT-QA, while it showed slightly inferior performance on AVSD-QA. For the Transformer, we used a 4-headed attention for NExT-QA, and a 2-headed attention for AVSD-QA.

Results

In this section, we provide numerical results of our approach against state of the art, as well as analyze the contribution of each component in our setup.

State-of-the-art Comparisons: In Tables 1 and 2, we compare the performance of our full (2.5+1)D-Transformer pipeline against recent state-of-the-art methods. Notably, on NExT-QA we compare with methods that use spatio-temporal models for VQA such as spatio-temporal reasoning (**?**), graph alignment (**?**), hierarchical relation models (**?**), against which our proposed model shows a significant $\sim$4% improvement, clearly showing benefits. On AVSD-QA, as provided in Table 2, we compare against the state of the art STSGR model (**?**), as well as older multi-modal Transformers (**?**), outperforming them in the mean rank of the retrieved answer. We found that when our AVSD-QA model is combined with other text cues (such as dialog history and captions), the mean rank improves to nearly 1.4, suggesting a significant bias between the questions and the text-cues. Thus, we restrict our analysis only to using the visual features.

Ablation Studies

In Table 3, we provide an ablation study on the importance of each component in our setup on both the datasets. Our

Table 2: AVSD-QA: Comparisons to the state of the art. The prior results are taken from (**?**).

Method	Mean Rank ↓
Question Only (**?**)	7.63
Multimodal Transformers (**?**)	7.23
Question + Video (**?**)	6.86
MTN (**?**)	6.85
ST Scene Graphs (**?**)	5.91
(2.5+1)D-Transformer (ours)	**5.84**

results show that without the static or the dynamic subgraphs, the performance drops. Without I3D features, the performance drops significantly for both the datasets, underlining the importance of motion features in the graph pipeline. We find that without the hierarchical Transformer, the performance drops from 53.4→52.9 on NExT-QA, and 5.84→5.97 on AVSD-QA. Further, the trick of using augmented answers in the learning process as described in the section on Training Losses seems to help improve the training of the models. We also evaluate the importance of question conditioning, which appears to contribute to the final performance. As our proposed pipeline is sequential in nature, we may also study the performance via removing individual modules from the pipeline. In Table 4 rows 1-4, we show the results of this experiment. Our results show that our proposed Transformer module leads to nearly 3% improvement, and using the pseudo-depth improves by a further 1%.

In Table 5, we ablate on the performance of the hierarchy in the (2.5+1)D-Transformer. Specifically, we show results for various bandwidths and their combinations on the NExT-QA dataset. The results suggest that including more bandwidths (hierarchies) leads to better modeling of the video scene and better performance. We use only the (2.5+1)D-Transformer for this experiment without using the combination with the standard Transformer to clearly separate out the benefits.

Computational Benefits: In Table 4 last row, we further show ablations on NExT-QA dataset, when the full set of graph nodes are used for inference. As expected in this case, the performance improves mildly, however our experiments show that the time taken for every training iteration in this case slows down 4-fold (from ∼1.5 s per iteration to ∼6 s on a single RTX6000 GPU). In Table 6, we compare the number of nodes in the static and dynamic graphs, and compare it to the total number of nodes in the unpruned graph for both the datasets. As the results show, our method prunes nearly 54% of graph nodes on AVSD-QA dataset and 24% on NExT-QA. We believe the higher pruning ratio for AVSD-QA is perhaps due to the fact that most of its videos do not contain shot-switches and use a stationary camera, which is not the case with NExT-QA.

Question-Category Level Performance: In Table 7, we compare the performance of our (2.5+1)D approach on various question categories in the NExT-QA dataset. Specifi-

Table 3: Ablation study on NExT-QA and AVSD-QA. Below, *Txr* is the standard Transformer and *I3D+FRCNN* is the averaged I3D and FRCNN features per frame (no graph), *V(2+1)D Txr*: without depth.

	NExT-QA	AVSD-QA
Method	Acc (%)↑	mean rank↓
No dynamic graph	52.49	5.97
No static graph	53.00	6.03
No I3D	52.65	6.09
No hier. kernel	52.90	5.97
No ans. augment (AA)	49.98	5.92
No question condition (QC)	50.39	5.96
Full Model	**53.40**	**5.84**

Table 4: Ablation study on NExT-QA by removing modules in our pipeline.

#	Ablation	Accuracy (%)↑
1	Txr + I3D + FRCNN + QC	47.90
2	(1) + AA	49.80
3	Txr + V(2+1)D Txr + AA + QC	52.40
4	Txr + V(2.5+1)D Txr + AA + QC	53.40
5	(4) using all nodes (no pruning)	**53.50**

Table 5: Ablation study on NExT-QA using different spatio-temporal hierarchies defined by the kernel bandwidth σ.

Hier. levels	bandwidths σ	Accuracy
1-level	0.01	52.13
2-levels	$\{0.01, 0.1\}$	52.58
3-levels	$\{0.01, 0.1, 1.0\}$	52.97
4-levels	$\{0.01, 0.1, 1.0, 10.0\}$	53.20
5-levels	$\{0.01, 0.1, 1.0, 10, 20.0\}$	53.00

Table 6: Computational benefits of the proposed approach. The numbers indicate the average number of graph nodes per video sequence in each dataset.

	AVSD-QA	NExT-QA
Full graph	502.43	656.30
Static graph	97.26	68.68
Dynamic graph	136.10	430.83
% node reduction	53.6	23.9

Table 7: Comparison of our answer selection on various categories in NExT-QA dataset against other recent methods. The numbers for competing methods are taken from (**?**).

Method	Why (W)	How (H)	Avg. (W+H)	Prev&Next (P&N)	Present (P)	Avg. (P&N+P)	Count (C)	Location (L)	Other (O)	Avg. (C+L+O)	Overall
STVQA, IJCV'19	45.37	43.05	44.76	47.52	51.73	49.26	43.50	65.42	53.77	55.86	47.94
CoMem, CVPR'18	46.15	42.61	45.22	48.16	50.38	49.07	41.81	67.12	51.80	55.34	48.04
HCRN, CVPR'20	46.99	42.90	45.91	48.16	50.83	49.26	40.68	65.42	49.84	53.67	48.20
HME, CVPR'19	46.52	45.24	46.18	47.52	49.17	48.20	45.20	73.56	51.15	58.30	48.72
HGA, AAAI'20	46.99	44.22	46.26	49.53	52.49	50.74	44.07	72.54	55.41	59.33	49.74
Ours	**52.39**	**48.36**	**51.33**	**50.91**	**54.28**	52.30	**46.02**	**77.08**	**58.31**	**62.58**	**53.4**
% improvement	+5.4	+3.12	+5.07	+1.38	+1.79	+1.56	+0.82	+3.52	+2.91	+3.25	+3.66

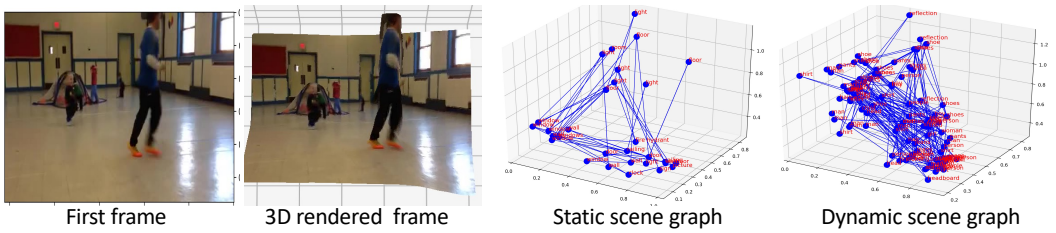


Figure 3: An example illustration of (2.5+1)D scene graphs produced by our method. The figure shows a video frame from the NExT-QA dataset, its pseduo-3D rendering, and the (2.5+1)D static and dynamic graphs computed on all frames of the video.

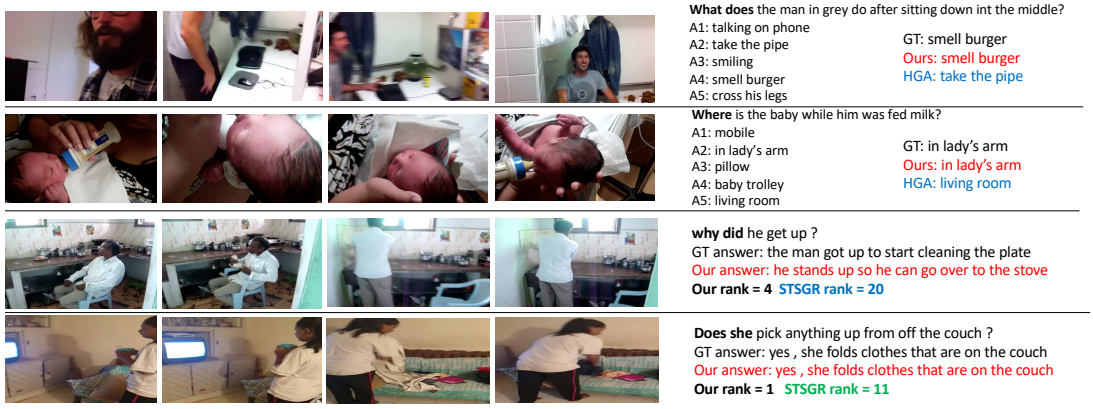


Figure 4: Qualitative responses: First two rows show the results on the NExT-QA dataset and the last two on the AVSD-QA dataset. We compare our results to those produced by HGA (**?**) on NExT-QA, and STSGR (**?**) on AVSD-QA datasets.

cally, the dataset categorizes its questions into 7 reasoning classes: (i) why, (ii) how, (iii) previous&next, (iv) present (v) counting, (vi) spatial location related, and (vii) all other questions. From Table 7, we see that our proposed representation fares well in all the categories against the state of the art. More interestingly, our method works significantly outperforms to the next best scheme HGA (**?**), by more than 5% on *why*-related questions and 3.5% on *location*-related questions, perhaps due to better spatio-temporal localization of the objects in the scenes as well as the spatio-temporal reasoning.

Qualitative Results: Figure 3 gives an example static-dynamic scene graph pair on a scene from NExT-QA. In Figure 4, we present qualitative QA results and compare against the responses produced by two recent methods. See Figures 5, 6, 7, and 8 for more results.

Conclusions

In this paper, we presented a novel (2.5+1)D representation for the task of video question answering. We use 2.5D pseudo-depth of scene objects to be disentangled in 3D space, allowing the pruning of redundant detections. Using the 3D setup, we further disentangled the scene into a set of dynamic objects that interact within themselves or with the environment (defined by static nodes); such interactions are characterized in a latent space via spatio-temporal hierarchical transformers, that produce varied abstractions of the scene at different scales. Such abstractions are then combined with text queries in the video QA task to select answers. Our experiments demonstrate state-of-the-art results on two recent and challenging real-world datasets.

Officia fugiat error adipisci fuga eius labore ab quos, odit eius magni, amet illo architecto nemo animi, voluptatum commodi accusamus molestiae minus cum sed eius quis

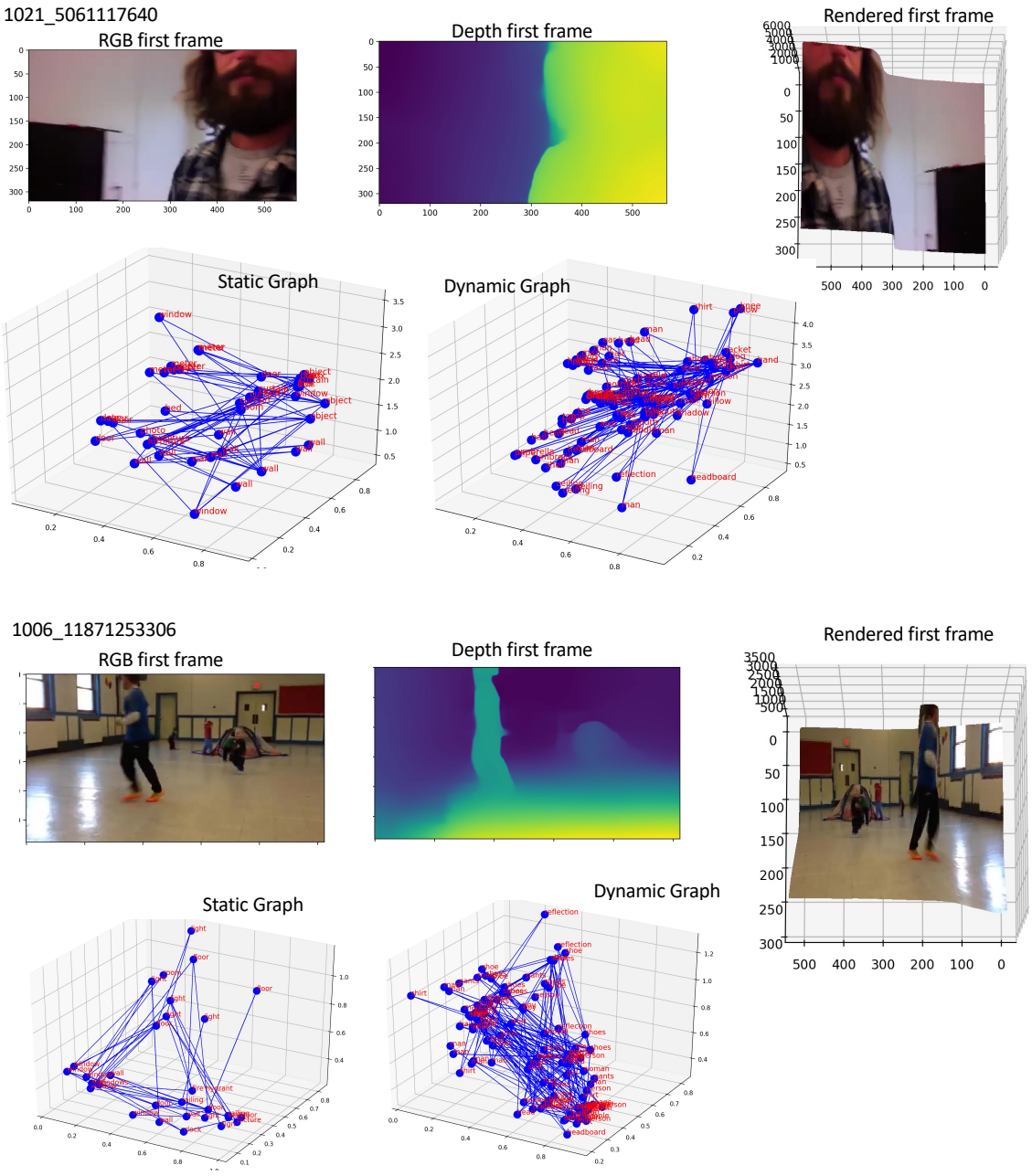


Figure 5: RGB images, depth images produced using (**?**), the depth rendered RGB images, the static scene graphs, and the dynamic scene graphs for the respective videos. See below for the questions and answers for these videos.

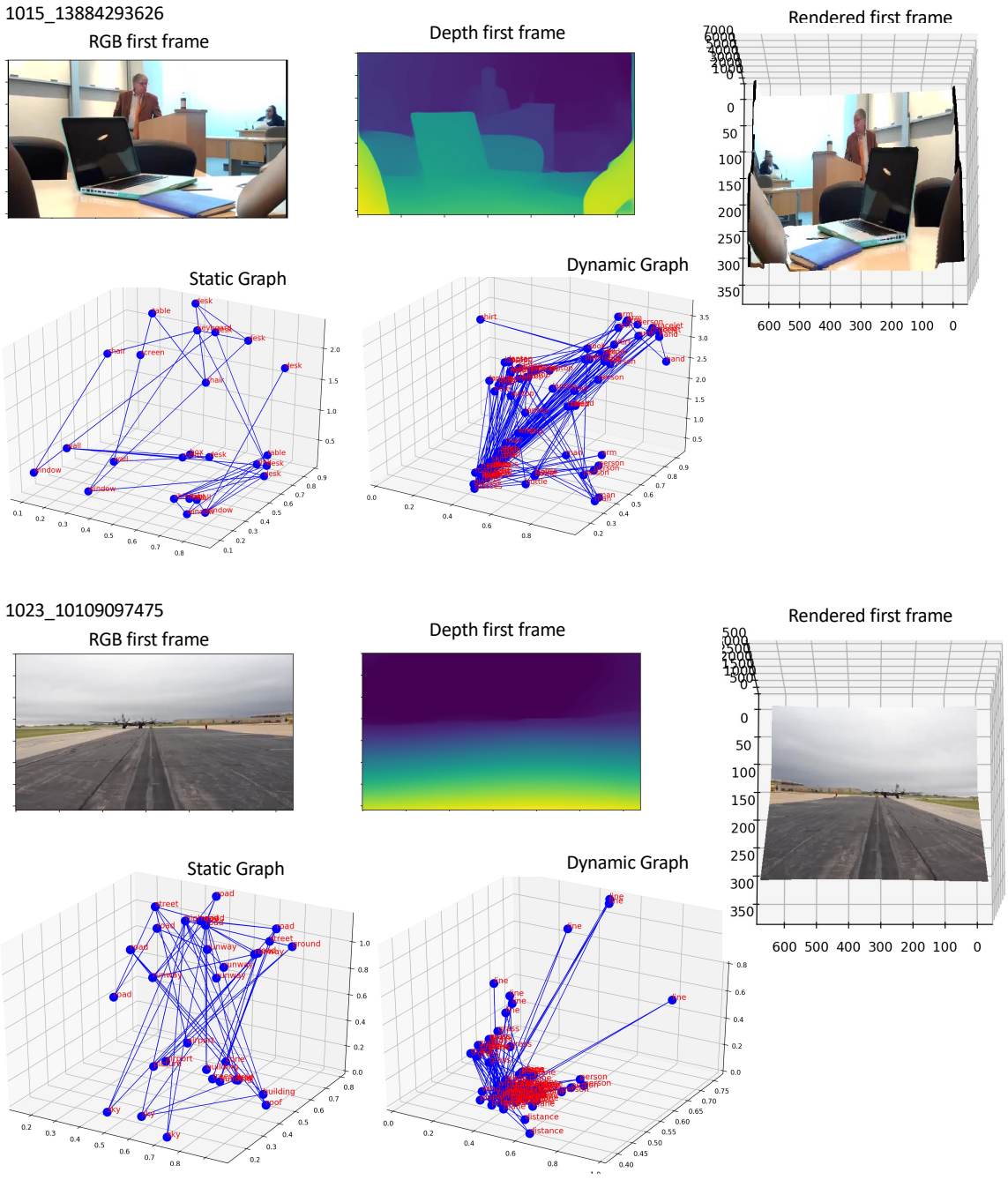


Figure 6: RGB images, depth images produced using (**?**), the depth rendered RGB images, the static scene graphs, and the dynamic scene graphs for the respective videos. See below for the questions and answers for these videos.

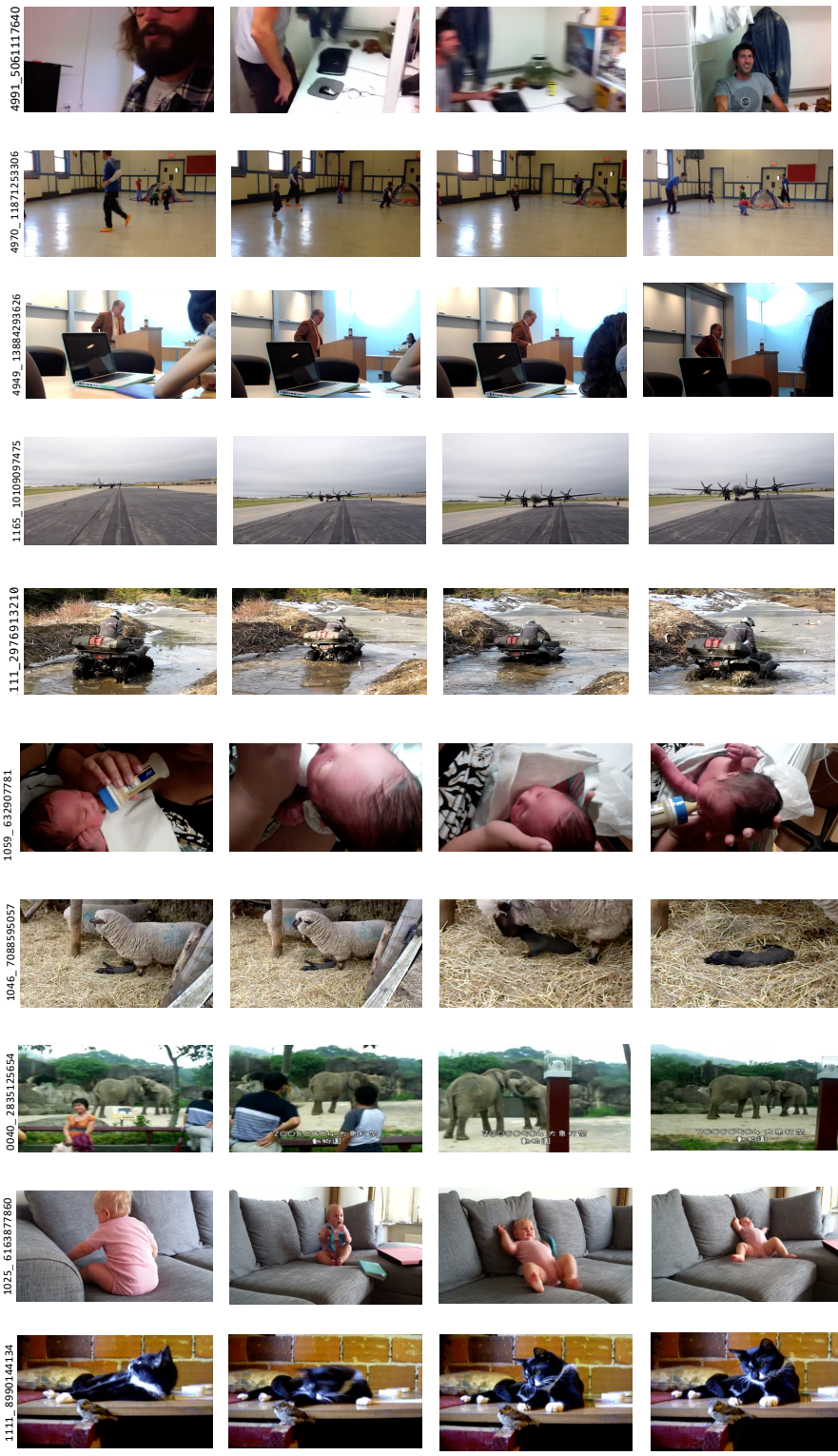
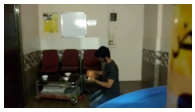
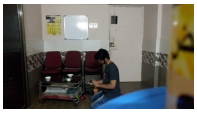
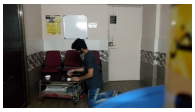
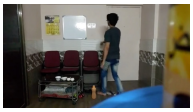


Figure 7: Qualitative responses from the NExT-QA dataset for various types of questions.

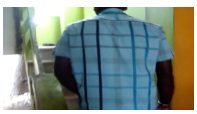
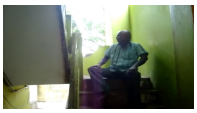





#1: PVP3C
Q: does it look more like a home or public place ?
GT answer: it looks like a public place , like dorm , apartment
Our answer: it looks like a public place , like dorm , apartment
Our rank = 1 **STSGR rank = 4**

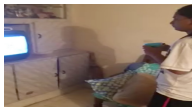
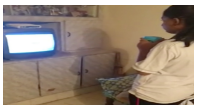


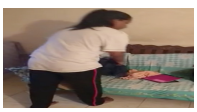

#2: CWERM
Q: why did he get up ?
GT answer: the man got up to start cleaning the plate
Our answer: he stands up so he can go over to the stove
Our rank = 4 **STSGR rank = 20**



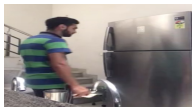



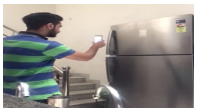
#3: J7DQX
Q: what is the man carrying ?
GT answer: he has a bottle of water , a cloth and a small
Our answer: he has a bottle of water , a cloth and a small
Our rank = 1 **STSGR rank = 22**



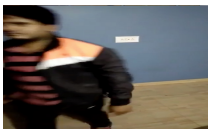
#4: 008UL
Q: does she pick anything up from off the couch ?
GT answer: yes , she folds clothes that are on the couch
Our answer: yes , she folds clothes that are on the couch
Our rank = 1 **STSGR rank = 11**






#5: 3ZPGZ
Q: why is he standing at the fridge ?
GT answer: to take pictures of the fridge from various
Our answer: to take pictures of the fridge from various
Our rank = 1 **STSGR rank = 13**



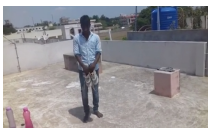



#6: PHCZQ
Q: where does he grab it from ?
GT answer: the same place he drops the other one off camera
Our best answer: he has it from the beginning of the video
Our correct answer rank = 5 **STSGR rank = 9**

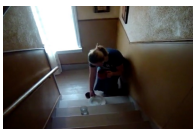
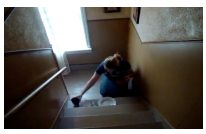
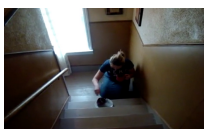



#7: 4YNPN
Q: can you tell why he stands there ?
GT answer: i cannot tell why he is standing still after closing the doc
Our best answer: no , it looks like he spends most of his time trying to open the one package
Our correct answer rank = 3 **STSGR rank = 8**

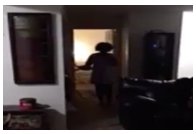
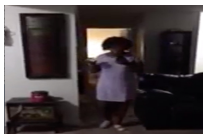


#8: TH0BF
Q: is standing by the shelves in the beginning , or does he walk over to the shelves ?
GT answer: he is already standing there
Our best answer: he is carrying plates and a mirror and seems to leave the room
Our correct answer rank = 3 **STSGR rank = 8**

#9: SHGOT
Q: does she enter from somewhere else or is she there the entire ti
GT answer: the only person is a young woman
Our answer: she seems to already be there , but doesn 't walk in fro the camera for a few seconds
Our rank = 8 **STSGR rank = 18**


#10: HLITK
Q: who is in the room ?
GT answer: there is only one
Our answer: a man , the person holding the broom is a man
Our rank = 25 **STSGR rank = 2**

Figure 8: Qualitative responses from the AVSD-QA dataset for various types of questions.

in?Inventore ad consequatur minima sint aspernatur libero fugit, ipsa beatae quibusdam voluptate impedit odit inventore, magni maiores quis autem inventore suscipit nulla