A Implementation details

A.1 Hyper-parameters

We use the same set of training hyper-parameters for all models during vision-language pre-training. We employ the AdamW optimizer [1] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and a weight decay of 0.05. We utilize a cosine learning rate decay with a peak learning rate of $2 \times 10^{-5}$ and a linear warm-up with a warm-up ratio of $5 \times 10^{-2}$. Our training data consists of images of size 224×224 that are augmented with random resized cropping and horizontal flipping. The maximum sequence length of the language model is set as 256.

A.2 Video encoding

Firstly, videos are distilled into distinct keyframes. These keyframes are then partitioned into three-dimensional (3D) patches. Leveraging the Conv3D technique, these 3D patches undergo a transformation into visual tokens. In the concluding step, these visual tokens seamlessly integrate with the vision transformer’s core architecture.

Specifically, we extract eight evenly distributed keyframes from each video. These keyframes are utilized as the primary inputs for our Conv3D module. The video gets divided into multiple three-dimensional (3D) patches, where each patch has a size defined by $\text{patch\_size} \times \text{patch\_size} \times \text{frame\_stride}$. As shown in Listing 1, these 3D patches are then embedded into visual tokens through the Conv3D module. Each of these 3D patches signifies a spatio-temporal cube within the video, thus efficiently capturing the visual content and the sequence of scenes, and incorporating the temporal dynamics of the videos.

```python
patch_embedding = nn.Conv3d(
in_channels=3,
out_channels=embed_dim,
kernel_size=(frame_stride, patch_size, patch_size),
stride=(frame_stride, patch_size, patch_size))
```

Listing 1: Pseudocode of the Conv3D for video encoding

In this code, the in_channels parameter is set to 3, representing the Red, Green, and Blue (RGB) channels of the video. The out_channels parameter corresponds to the dimensions of the embedding. The kernel_size and stride parameters specify the size and stride of the 3D patch, respectively.

After the division of the video into 3D patches, Conv3D captures the spatio-temporal features from the frame sequence and converts each keyframe into a set of visual tokens. These visual tokens then serve as inputs for the vision transformer encoder. The vision transformer leverages the temporal...
information within the visual tokens derived from the keyframes, which allows for effective encoding of the video content. By processing this sequence of visual tokens, we adapt the pretrained image encoder to manage video inputs efficiently.

A.3 Downstream policy learning

We adopt imitation learning as the method of policy learning in low-level control tasks, which leverages demonstration data provided by an expert to learn the desired behavior. This technique has found applications in various domains, such as robotics, autonomous driving, and game playing. We provide each task with 25 demonstrations, which are trajectories of observations and actions performed by an expert in the given task, and test the performance with 25 demonstrations and only 10 demonstrations respectively. The goal of imitation learning is to learn a policy, denoted as \( \pi \), that maps the agent’s observations to appropriate actions. The learned policy should be able to imitate the expert’s behavior accurately. Specifically, we use behavioral cloning to learn the downstream policy, which trains a supervised learning model to predict actions given states based on the expert demonstrations, and the loss function is shown as in Equation 1:

\[
L(\theta) = \sum [\pi_\theta(a|s) \log P^*(a|s)]
\]

Here, \( \theta \) represents the parameters of the policy model, \( \pi_\theta(a|s) \) denotes the predicted action probability distribution given a state \( s \), and \( P^*(a|s) \) represents the ground truth action probability distribution derived from the expert demonstrations.

For Franka Kitchen [2] tasks, the length of a demonstration is 50, which contains 50 state-action pairs. For Meta-World [3] tasks, the length of a demonstration is 500, which contains 500 state-action pairs. Our evaluation methodology is loosely inspired by R3M [4], but we only use visual observations to assess the effectiveness of EmbodiedGPT. We employ a visual representation \( z_t \) generated by EmbodiedGPT and train the policy \( \pi \) using a standard behavior cloning loss \( ||a_t - \pi(z_t)||_2^2 \). The parameterization of \( \pi \) consists of a two-layer MLP [5] with a preceding BatchNorm [6] at the input. The agent is trained for 20,000 steps, and we evaluate its performance in the environment every 1000 steps, reporting the best success rate achieved. For each task, we conduct 5 behavior cloning runs with different seeds. The final success rate reported for a method on a specific task represents the average across 5 seeds, 2 camera viewpoints, and 2 demo dataset sizes, resulting in a total of 20 runs.

B More demos of EmbodiedGPT

B.1 Visual Captioning

We assessed EmbodiedGPT on numerous visual captioning tasks spanning a range of embodied AI benchmarks. As shown in Figure 1, the model displayed an exceptional ability to accurately describe objects, characters, and spatial relationships relevant to embodied AI tasks. Furthermore, EmbodiedGPT exhibited robust zero-shot learning capabilities, evidenced by its strong performance across multiple benchmarks without the need for task-specific fine-tuning.

B.2 Embodied Planning with video input

Here, we present additional examples of embodied planning with video inputs. As shown in Figure 2, the examples demonstrate that EmbodiedGPT can generate high-quality and executable planning.

B.3 Embodied Planning with image input

Embodied Planning for Concrete Tasks: In the context of concrete task planning, such as making a cup of coffee, EmbodiedGPT effectively utilized visual information to pinpoint the required objects and their positional relationships within the environment. As shown in Figure 3, EmbodiedGPT produced coherent, multi-step plans, taking into consideration both the task requirements and environmental constraints. This capability demonstrates the model’s potential to facilitate real-world applications and tackle complex planning challenges within the realm of embodied AI.
We show Table 1 to outline the scoring criteria for a user study incorporating the above five evaluation metrics:

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Scoring Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual</td>
<td>Relies heavily on textual inputs, may not fully exploit visual cues.</td>
</tr>
<tr>
<td>Visual</td>
<td>Incorporates additional prompts to guide model's attention towards relevant objects.</td>
</tr>
<tr>
<td>Cross-modal</td>
<td>Combines textual and visual observations to generate concrete sub-task descriptions.</td>
</tr>
<tr>
<td>Adaptability</td>
<td>Demonstrates ability to adapt to diverse problem-solving contexts.</td>
</tr>
</tbody>
</table>

Through extensive experiments evaluating multi-modal large models in question answering tasks, we have identified a limitation in their ability to effectively utilize visual information. These models tend to rely heavily on textual inputs and may not fully exploit the visual cues present in the question-answering process. In order to address this limitation and enhance their performance, we aimed to investigate the impact of incorporating additional prompts that guide the model's attention toward the visual content.
This video depicts a robot performing a manipulation task in a kitchen backdrop. The scene includes a white robotic arm, a black platform with four gray-white circles that indicate the heating zones of an induction cooktop. In the lower left corner of the platform, there is a white kettle with a handle, and on the far left, a black microwave oven with a handle for opening. Above the platform, two blue-gray cabinets are situated; the left cabinet has two doors with handles, and the right cabinet has a sliding door with a handle. Below the cabinets, there are several knobs and a rod-shaped switch that can be toggled. The switch is located on the far left, while the other knobs are on the right. In the video, the robotic arm is moving the rod-shaped switch to the left to illuminate the workspace.

Assistant: The robotic arm is moving the rod-shaped switch to the left to illuminate the workspace.

Human: Describe the video in detail.

Assistant: The video features a red robot with a mechanical arm, a yellow table, a hammer with a red handle, and a yellow wooden box with a nail protruding from it. The robot is hammering the nail into the box. The mechanical arm first picks up the hammer, aligns the gray iron head with the nail, hammers it, and then slowly lowers the hammer.

(a) Embodied planning for the light opening task in Franka Kitchen.

(b) Embodied planning for hammering nail task in Meta-World.
To achieve this goal, we propose a straightforward yet highly effective approach: incorporating additional prompting into the model’s input. This involves introducing specific prompts such as “in the scene shown in this image/video” or allowing the model to describe the image/video as part of a multi-turn dialogue. By including these prompts, we aim to explicitly direct the model’s focus toward the visual information available and encourage it to utilize this information when generating answers. Our experiments have yielded promising results. As shown in Figure 5, the introduction of additional prompts has significantly improved the model’s ability to leverage visual information and provide accurate answers based on the visual content. By explicitly referencing the scene depicted in the image or video, the model’s attention is directed toward the relevant visual features, leading to a more comprehensive integration of visual and textual information. Consequently, the model’s reasoning ability is enhanced, resulting in more precise and contextually grounded answers.

References

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Recognition Accuracy</td>
<td>This metric measures the ability of a system to accurately identify objects from images or videos. A higher accuracy indicates that the system can correctly recognize the objects present in the given visual data.</td>
</tr>
<tr>
<td>Spatial Relationship Understanding</td>
<td>Spatial relationship understanding refers to the system’s capability to accurately discern the spatial relationships between objects in a scene. It evaluates whether the system can determine the relative positions, orientations, distances, and other spatial attributes of objects with precision.</td>
</tr>
<tr>
<td>Level of Redundancy in the Answer</td>
<td>The level of redundancy in the answer assesses the amount of unnecessary or repetitive information present in the system’s response. Lower redundancy indicates that the system provides concise and non-repetitive answers, which is generally preferred as it reduces verbosity and improves clarity.</td>
</tr>
<tr>
<td>Reasonability of the Planning</td>
<td>The reasonability of the planning metric gauges the logical coherence and appropriateness of the system’s planning process. It examines whether the system’s generated plans are sensible and align with the given goals or objectives.</td>
</tr>
<tr>
<td>Executability of the Planning</td>
<td>This metric evaluates the feasibility and practicality of the system’s generated plans. It assesses whether the plans can be executed successfully in the real world.</td>
</tr>
</tbody>
</table>

Table 1: Evaluation metric and scoring criteria for user study

---

Human: with this image, if you want to take off clothes and hang them in the closet, write your plan according to the image.

Assistant:
1. Find a location in the image where there is an empty hanger.
2. Identify the size or shape of the hanger to ensure it can accommodate your clothes.
3. Use the handheld device to remove the clothes from the image.
4. Position the clothes on the hanger in the image, ensuring they fit comfortably.
5. Once the clothes are positioned correctly, use the device to hang them on the hanger.
6. Once you have successfully hung the clothes, verify that they are visible in the image.

(a) Ask EmbodiedGPT to write the plan directly.

Human: with this image, if you want to take off clothes and hang them in the closet, identify where you need to go and what you need to do in the scene shown in this image.

Assistant:
1. To hang clothes in the closet, you need to identify the closet area or location within the room where you can place the clothes.
2. In the image, there is a closet visible on the far right side of the room. The location of the closet can vary depending on the specific setting or design of the room.
3. Once you have identified the closet, go to the closet and find a hanging rod or shelf to place the clothes on.
4. In the image, there is a hanging rod visible on the far right side of the door frame. Ensure that the hanging rod is large enough to accommodate the clothes you want to hang.
5. Take the clothes off the bed and place them on the hanging rod.

(b) Ask EmbodiedGPT to write the plan with image-related chain-of-thought.

Figure 5: Performance Comparison with different types prompt.


