Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models

Chenfei Wu   Shengming Yin   Weizhen Qi   Xiaodong Wang   Zecheng Tang   Nan Duan*

Microsoft Research Asia

{chewu, v-sheyin, t-weizhenqi, v-xiaodwang, v-zetang, nanduan}@microsoft.com

Abstract

*ChatGPT is attracting a cross-field interest as it provides a language interface with remarkable conversational competency and reasoning capabilities across many domains. However, since ChatGPT is trained with languages, it is currently not capable of processing or generating images from the visual world. At the same time, Visual Foundation Models, such as Visual Transformers or Stable Diffusion, although showing great visual understanding and generation capabilities, they are only experts on specific tasks with one-round fixed inputs and outputs. To this end, We build a system called **Visual ChatGPT**, incorporating different Visual Foundation Models, to enable the user to interact with ChatGPT by 1) sending and receiving not only languages but also images 2) providing complex visual questions or visual editing instructions that require the collaboration of multiple AI models with multi-steps. 3) providing feedback and asking for corrected results. We design a series of prompts to inject the visual model information into ChatGPT, considering models of multiple inputs/outputs and models that require visual feedback. Experiments show that Visual ChatGPT opens the door to investigating the visual roles of ChatGPT with the help of Visual Foundation Models. Our system is publicly available at https://github.com/microsoft/visual-chatgpt.*

1. Introduction

In recent years, the development of Large language models (LLMs) has shown incredible progress, such as T5 [32], BLOOM [36], and GPT-3 [5]. One of the most significant breakthroughs is ChatGPT, which is built upon Instruct-GPT [29], specifically trained to interact with users in a genuinely conversational manner, thus allowing it to maintain the context of the current conversation, handle follow-up questions, and correct answer produced by itself.

Although powerful, ChatGPT is limited in its ability to process visual information since it is trained with a
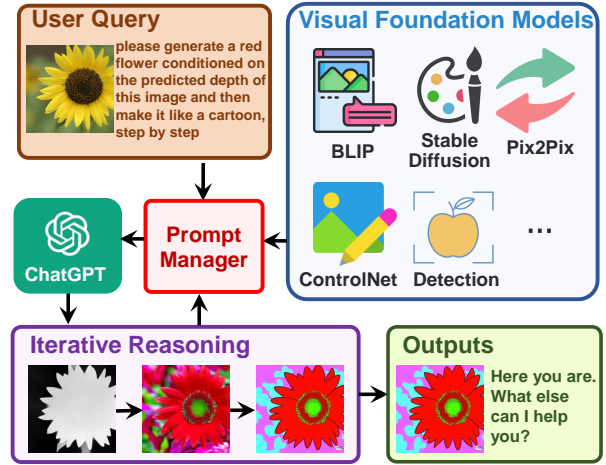

Figure 1. Architecture of Visual ChatGPT.

single language modality, while Visual Foundation Models (VFMs) have shown tremendous potential in computer vision, with their ability to understand and generate complex images. For instance, BLIP Model [22] is an expert in understanding and providing the description of an image. Stable Diffusion [35] is an expert in synthesizing an image based on text prompts. However, suffering from the task specification nature, the demanding and fixed input-output formats make the VFMs less flexible than conversational language models in human-machine interaction.

Could we build a ChatGPT-like system that also supports image understanding and generation? One intuitive idea is to train a multi-modal conversational model. However, building such a system would consume a large amount of data and computational resources. Besides, another challenge comes that what if we want to incorporate modalities beyond languages and images, like videos or voices? Would it be necessary to train a totally new multi-modality model every time when it comes to new modalities or functions?

We answer the above questions by proposing a system named **Visual ChatGPT**. Instead of training a new multi-modal ChatGPT from scratch, we build Visual ChatGPT directly based on ChatGPT and incorporate a variety of VFMs. To bridge the gap between ChatGPT and these VFMs, we propose a Prompt Manager which supports the following functions: 1) explicitly tells ChatGPT the capa-

---

*Corresponding author.

bility of each VFM and specifies the input-output formats; 2) converts different visual information, for instance, png images, the depth images and mask matrix, to language format to help ChatGPT understand; 3) handles the histories, priorities, and conflicts of different Visual Foundation Models. With the help of the Prompt Manager, ChatGPT can leverage these VFMs and receives their feedback in an iterative manner until it meets the requirements of users or reaches the ending condition.

As shown in Fig. 1, a user uploads an image of a yellow flower and enters a complex language instruction "please generate a red flower conditioned on the predicted depth of this image and then make it like a cartoon, step by step". With the help of Prompt Manager, Visual ChatGPT starts a chain of execution of related Visual Foundation Models. In this case, it first applies the depth estimation model to detect the depth information, then utilizes the depth-to-image model to generate a figure of a red flower with the depth information, and finally leverages the style transfer VFM based on the Stable Diffusion model to change the style of this image into a cartoon. During the above pipeline, Prompt Manager serves as a dispatcher for ChatGPT by providing the type of visual formats and recording the process of information transformation. Finally, when Visual ChatGPT obtains the hints of "cartoon" from Prompt Manager, it will end the execution pipeline and show the final result.

In summary, our contributions are as follows:

- We propose Visual ChatGPT, which opens the door of combining ChatGPT and Visual Foundation Models and enables ChatGPT to handle complex visual tasks;

- We design a Prompt Manager, in which we involve **22** different VFMs and define the internal correlation among them for better interaction and combination;

- Massive zero-shot experiments are conducted and abundant cases are shown to verify the understanding and generation ability of Visual ChatGPT.

2. Related Works

2.1. Natural Language and Vision

Surrounded by various modalities (sound, vision, video, *etc*), language and vision are the two main mediums transmitting information in our life. There is a natural link between the natural language and visions, and most questions require joint modeling of both two streams to produce the satisfied results [15, 26, 48], e.g., visual question answering (VQA) [2] takes an image and one corresponding question as input and requires to generate an answer according to the information in the given image. Owing to the success of large language models (LLMs) like InstructGPT [29], one can easily interact with the model or obtain feedback in the natural language format, but it is incapable for those LLMs to process the visual information. To fuse the vision processing ability into such LLMs, several challenges are lying ahead since it is hard to train either large language models or vision models, and the well-designed instructions [4, 55, 21] and cumbersome conversions [30, 52] are required to connect different modalities. Although several works have explored leveraging the pre-trained LLMs to improve the performance on the vision-language (VL) tasks, those methods supported several specific VL tasks (from language to version or from version to language) and required labeled data for training [38, 1, 22].

2.2. Pre-trained Models for VL tasks

To better extract visual features, frozen pre-trained image encoders are adopted in the early works [9, 25, 54], and recent LiT [52] apply the CLIP pre-training [30] with frozen ViT model [51]. From another perspective, exploiting the knowledge from LLMs also counts. Following the instruction of Transformer [39], pre-trained LLMs demonstrate a powerful text understanding and generation capability [31, 19, 37, 5], and such breakthroughs also benefit the VL modelling [13, 14, 3, 49], where these works add an extra adapter modules [17] in the pre-trained LLMs to align visual features to the text space. With the increased number of model parameters, it is hard to train those pretrained LLMs, thus more efforts have been paid to directly leverage the off-the-shelf frozen pre-trained LLMs for VL tasks [12, 38, 8, 46, 50].

2.3. Guidance of Pre-trained LLMs for VL tasks

To deal with complex tasks, e.g., commonsense reasoning [11], Chain-of-Thought (CoT) is proposed to elicit the multi-step reasoning abilities of LLMs [42]. More concretely, CoT asks the LLMs to generate the intermediate answers for the final results. Existing study [57] have divided such a technique into two categories: Few-Shot-CoT [56] and Zero-Shot-CoT [20]. For the few-shot setting, the LLMs perform CoT reasoning with several demonstrations [58, 41], and it turns out that the LLMs can acquire better abilities to solve complex problems. Further, recent studies [20, 47] have shown that LLMs can be self-improved by leveraging self-generated rationales under the zero-shot setting. The above studies mainly focus on a single modality, i.e., language. Recently, Multimodal-CoT [57] is proposed to incorporate language and vision modalities into a two-stage framework that separates rationale generation and answer inference. However, such a method merely shows superiority under specific scenarios, i.e., ScienceQA benchmark [28]. In a nutshell, our work extends the potentiality of CoT to massive tasks, including but not limited to text-to-image generation [27], image-to-image translation [18], image-to-text generation [40], *etc*.
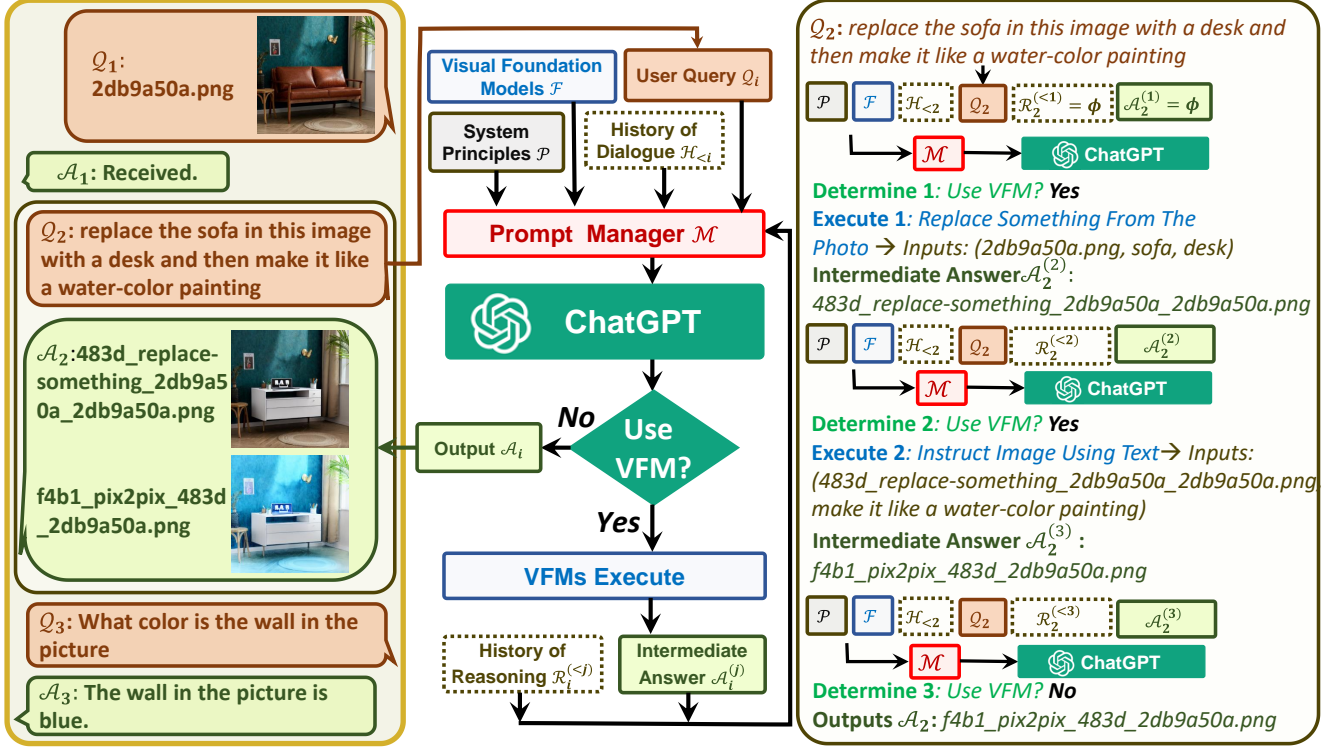
Figure 2. Overview of Visual ChatGPT. The left side shows a three-round dialogue, The middle side shows the flowchart of how Visual ChatGPT iteratively invokes Visual Foundation Models and provide answers. The right side shows the detailed process of the second QA.

3. Visual ChatGPT

Let $\mathcal{S} = \{(\mathcal{Q}_1, \mathcal{A}_1), (\mathcal{Q}_2, \mathcal{A}_2), ..., (\mathcal{Q}_N, \mathcal{A}_N)\}$ be a dialogue system with N question-answer pairs. To get the response $\mathcal{A}_i$ from the i-th round of conversation, a series of VFMs and intermediate outputs $\mathcal{A}_i^{(j)}$ from those models are involved, where j denotes the output from the j-th VFM ($\mathcal{F}$) in i-th round. More concretely, handling with Prompt Manager $\mathcal{M}$, the format of $\mathcal{A}_i^{(j)}$ is constantly modified to meet the input format of each $\mathcal{F}$. In the end, the system output $\mathcal{A}_i^{(j)}$ if it is denoted as the final response, and no more VFM is executed. Eq. (1) provides a formal definition of Visual ChatGPT:

$$\mathcal{A}_i^{(j+1)} = ChatGPT(\mathcal{M}(\mathcal{P}), \mathcal{M}(\mathcal{F}), \mathcal{M}(\mathcal{H}_{<i}), \mathcal{M}(\mathcal{Q}_i),$$
$$\mathcal{M}(\mathcal{R}_i^{(<j)}), \mathcal{M}(\mathcal{F}(\mathcal{A}_i^{(j)}))) \quad (1)$$

– System Principle $\mathcal{P}$: System Principle provides basic rules for Visual ChatGPT, e.g., it should be sensitive to the image filenames, and should use VFMs to handle images instead of generating the results based on the chat history.

– Visual Foundation Model $\mathcal{F}$: One core of Visual ChatGPT is the combination of various VFMs: $\mathcal{F} = \{f_1, f_2, ..., f_N\}$, where each foundation model f_i contains a determined function with explicit inputs and outputs.

– History of Dialogue $\mathcal{H}_{<i}$: We define the dialogue history of i-th round of conversation as the string concatenation of previous question answer pairs, i.e, $\{(\mathcal{Q}_1, \mathcal{A}_1), (\mathcal{Q}_2, \mathcal{A}_2), \cdots, (\mathcal{Q}_{i-1}, \mathcal{A}_{i-1})\}$. Besides, we truncate the dialogue history with a maximum length threshold to meet the input length of ChatGPT model.

– User query $\mathcal{Q}_i$: In visual ChatGPT, query is a general term, since it can include both linguistic and visual queries. For instance, Fig. 1 shows an example of a query containing both the query text and the corresponding image.

– History of Reasoning $\mathcal{R}_i^{(<j)}$: To solve a complex question, Visual ChatGPT may require the collaboration of multiple VFMs. For the i-th round of conversation, $\mathcal{R}_i^{(<j)}$ is all the previous reasoning histories from j invoked VFMs.

– Intermediate Answer $\mathcal{A}^{(j)}$: When handling a complex query, Visual ChatGPT will try to obtain the final answer step-by-step by invoking different VFMs logically, thus producing multiple intermediate answers.

– Prompt Manager $\mathcal{M}$: A prompt manager is designed to convert all the visual signals into language so that Chat-GPT model can understand. In the following subsections, we focus on introducing how $\mathcal{M}$ manages above different parts: $\mathcal{P}, \mathcal{F}, \mathcal{Q}_i, \mathcal{F}(\mathcal{A}_i^{(j)})$.
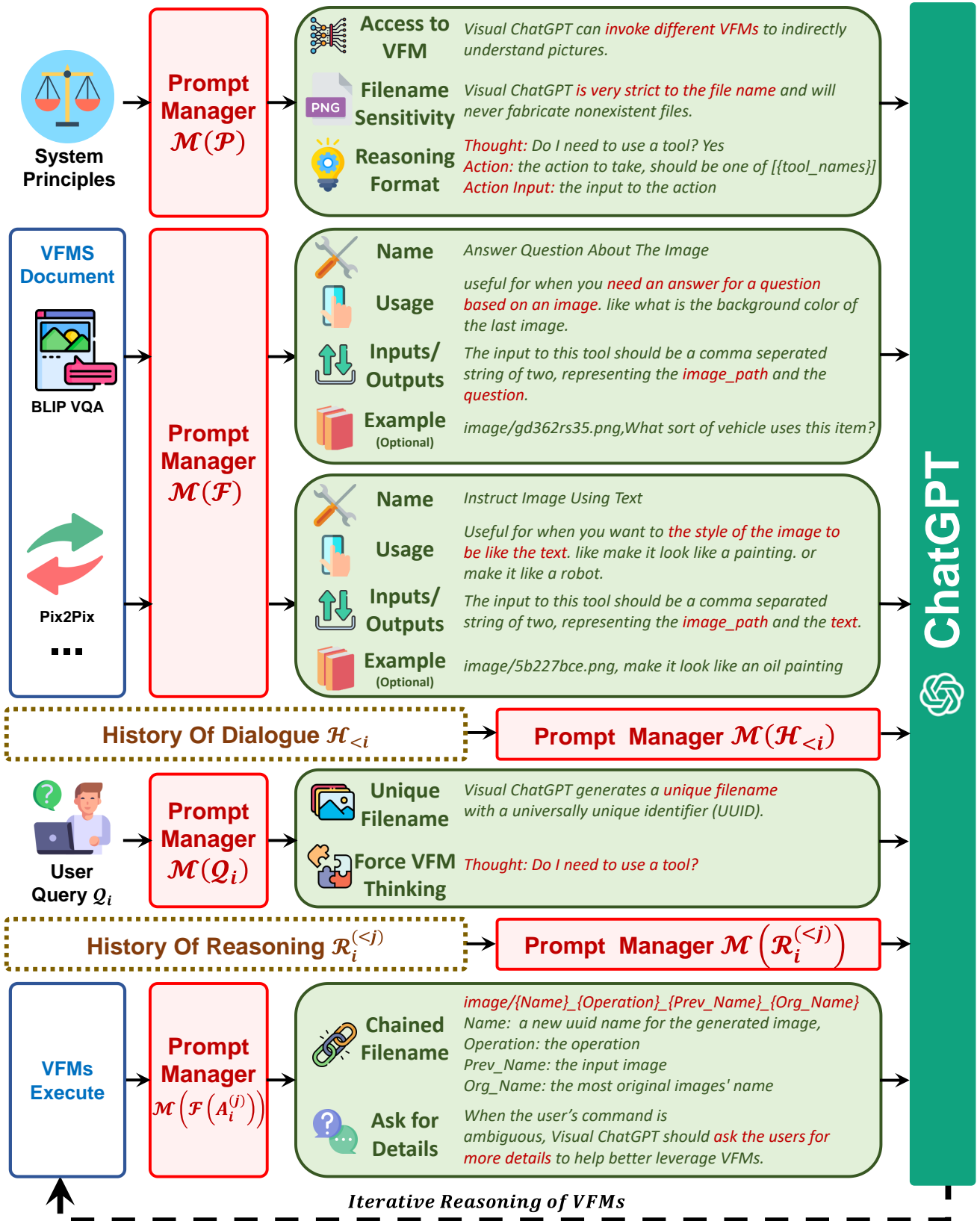
Figure 3. Overview of Prompt Manager. It coverts all non-language signals into language so that ChatGPT can understand.

3.1. Prompt Managing of System Principles $\mathcal{M}(\mathcal{P})$

Visual ChatGPT is a system that integrates different VFMs to understand visual information and generation corresponding answers. To accomplish this, some system principles need to be customized, which are then transferred into prompts that ChatGPT can understand. These prompts serve several purposes, including:

- **Role of Visual ChatGPT** Visual ChatGPT is designed to assist with a range of text and visual-related tasks, such as VQA, image generation, and editing.

- **VFMs Accessibility** Visual ChatGPT has access to a list of VFMs to solve various VL tasks. The decision of which foundation model to use is entirely made by the ChatGPT model itself, thus making it easy to support new VFMs and VL tasks.

- **Filename Sensitivity** Visual ChatGPT accesses image files according to the filename, and it is crucial to use precise filenames to avoid ambiguity since one round of conversation may contain multiple images and their different updated versions and the misuse of filenames will lead to the confusion about which image is currently being discussed. Therefore, Visual ChatGPT is designed to be strict about filename usage, ensuring that it retrieves and manipulates the correct image files.

- **Chain-of-Thought** As shown in Fig. 1, to cope with one seemingly simple command may require multiple VFMs, e.g., the query of "generate a red flower conditioned on the predicted depth of this image and then make it like a cartoon" requires depth estimation, depth-to-image and the style transfer VFMs. To tackle more challenging queries by decomposing them into subproblems, CoT is introduced in Visual ChatGPT to help decide, leverage and dispatch multiple VFMs.

- **Reasoning Format Strictness** Visual ChatGPT must follow strict reasoning formats. Thus we parse the intermediate reasoning results with the elaborate regex matching algorithms, and construct the rational input format for ChatGPT model to help it determine the next execution, e.g., triggering a new VFM or returning the final response.

- **Reliability** As a language model, Visual ChatGPT may fabricate fake image filenames or facts, which can make the system unreliable. To handle such issues, we design prompts that require Visual ChatGPT to be loyal to the output of the vision foundation models and not fabricate image content or filenames. Besides, the collaboration of multiple VFMs can increase system reliability, thus the prompt we construct will guide ChatGPT to leverage VFMs preferentially instead of generating results based on conversation history.

Table 1. Foundation models supported by Visual ChatGPT.

Remove Objects from Image [10, 35]	
Replace Objects from Image [10, 35]	
Change Image by the Text [35]	
Image Question Answering [23]	
Image-to-Text [23]	Text-to-Image [35]
Image-to-Edge [45]	Edge-to-Image [53]
Image-to-Line [16]	Line-to-Image [53]
Image-to-Hed [44]	Hed-to-Image [53]
Image-to-Seg [24]	Seg-to-Image [53]
Image-to-Depth [34, 33]	Depth-to-Image [53]
Image-to-NormalMap [34, 33]	NormalMap-to-Image [53]
Image-to-Sketch [44]	Sketch-to-Image [53]
Image-to-Pose [6]	Pose-to-Image [53]

3.2. Prompt Managing of Foundation Models $\mathcal{M}(\mathcal{F})$

Visual ChatGPT is equipped with multiple VFMs to handle various VL tasks. Since these different VFMs may share some similarities, e.g., the replacement of objects in the image can be regarded as generating a new image, and both Image-to-Text (I2T) task and Image Question Answering (VQA) task can be understood as giving the response according to the provided image, it is critical to distinguish among them. As shown in Fig. 3, the Prompt Manager specifically defines the following aspects to help Visual ChatGPT accurately understand and handle the VL tasks:

- **Name** The name prompt provides an abstract of the overall function for each VFM, e.g., answer question about the image, and it not only helps Visual ChatGPT to understand the purpose of VFM in a concise manner but also severs as the entry to VFM.

- **Usage** The usage prompt describes the specific scenario where the VFM should be used. For example, the Pix2Pix model [35] is suitable for changing the style of an image. Providing this information helps Visual ChatGPT make informed decisions about which VFM to use for the particular task.

- **Inputs/Outputs** The inputs and the outputs prompt outlines the format of inputs and outputs required by each VFM since the format can vary significantly and it is crucial to provide clear guideline for Visual ChatGPT to execute the VFMs correctly.

- **Example(Optional)** The example prompt is optional, but it can be helpful for Visual ChatGPT to better understand how to use particular VFM under the specific input template and deal with more complex queries.

3.3. Prompt Managing of User Querie $\mathcal{M}(\mathcal{Q}_i)$

Visual ChatGPT supports a variety of user queries, including languages or images, simple or complex ones, and

the reference of multiple images. Prompt Manager handles user queries in the following two aspects:

- **Generate Unique Filename** Visual ChatGPT can handle two types of image-related queries: those that involve newly uploaded images and those that involve reference to existing images. For newly uploaded images, Visual ChatGPT generates a unique filename with a universally unique identifier (UUID) and adds a prefix string "image" representing the relative directory, e.g., "image/{uuid}.png". Although the newly uploaded image will not be fed into ChatGPT, a fake dialogue history is generated with a question stating the image's filename and an answer indicating that the image has been received. This fake dialogue history assists in the following dialogues. For queries that involve reference to existing images, Visual ChatGPT ignores the filename check. This approach has been proven beneficial since ChatGPT has the ability to understand fuzzy matching of user queries if it does not lead to ambiguity, e.g., UUID names.

- **Force VFM Thinking** To ensure the successful trigger of VFMs for Visual ChatGPT, we append a suffix prompt to $(\mathcal{Q}_i)$: "Since Visual ChatGPT is a text language model, Visual ChatGPT must use tools to observe images rather than imagination. The thoughts and observations are only visible for Visual ChatGPT, Visual ChatGPT should remember to repeat important information in the final response for Human. Thought: Do I need to use a tool?". This prompt serves two purposes: 1) it prompts Visual ChatGPT to use foundation models instead of relying solely on its imagination; 2) it encourages Visual ChatGPT to provide specific outputs generated by the foundation models, rather than generic responses such as "here you are".

3.4. Prompt Managing of Foundation Model Outputs $\mathcal{M}(\mathcal{F}(\mathcal{A}_i^{(j)}))$

For the intermediate outputs from different VFMs $\mathcal{F}(\mathcal{A}_i^{(j)})$, Visual ChatGPT will implicitly summarize and feed them to the ChatGPT for subsequent interaction, i.e., calling other VFMs for further operations until reaching the ending condition or giving the feedback to the users. The inner steps can be summarized below:
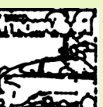
- **Generate Chained Filename** Since the intermediate outputs of Visual ChatGPT will become the inputs for the next implicit conversational round, we should make those outputs more logical to help the LLMs better understand the reasoning process. Specifically, the image generated from the Visual Foundation Models are saved under the "image/" folder, which hints the following strings representing an image name. Then, the image is named as "{Name}_{Operation}_{Prev_Name}_{Org_Name}", where {Name} is the UUID name mentioned above, with {Operation} as the operation name, {Prev_Name} as the input image unique identifier, and {Org_Name} as the original name of the image uploaded by users or generated by VFMs. For instance, "image/ui3c_edge-of_o0ec_nji9dcgf.png" is a canny edge image named "ui3c" of input "o0ec", and the original name of this image is "nji9dcgf". With such a naming rule, it can hint ChatGPT of the intermediate result attributes,i.e., image, and how it was generated from a series of operations.

- **Call for More VFMs** One core of Visual ChatGPT is that it can automatically call for more VFMs to finish the user's command. More concretely, we make the ChatGPT keep asking itself whether it needs VFMs to solve the current problem by extending one suffix "Thought: " at the end of each generation.

- **Ask for More Details** When the user's command is ambiguous, Visual ChatGPT should ask the users for more details to help better leverage VFMs. This design is safe and critical since the LLMs are not permitted to arbitrarily tamper with or speculate about the user's intention without basis, especially when the input information is insufficient.

4. Experiments

4.1. Setup

We implement the LLM with ChatGPT [29] (OpenAI "text-davinci-003" version), and guide the LLM with LangChain [7][1]. We collect foundation models from HuggingFace Transformers [43][2], Maskformer [10][3] and ControlNet [53][4]. The fully deployment of all the 22 VFMs requires 4 Nvidia V100 GPUs, but users are allowed to deploy fewer foundation models to save GPU resources flexibly. The maximum length of chat history is 2,000 and excessive tokens are truncated to meet the input length of ChatGPT.

4.2. A full Case of Multiple Rounds Dialogue

Fig. 4 shows a 16-round multi-modal dialogue case of Visual ChatGPT. In this case, the user asks for both text and image questions, and Visual ChatGPT responses with both text and images. The dialogue involves a discussion of multiple images, processing with multiple fundamental models, and handling questions that require multi-steps.
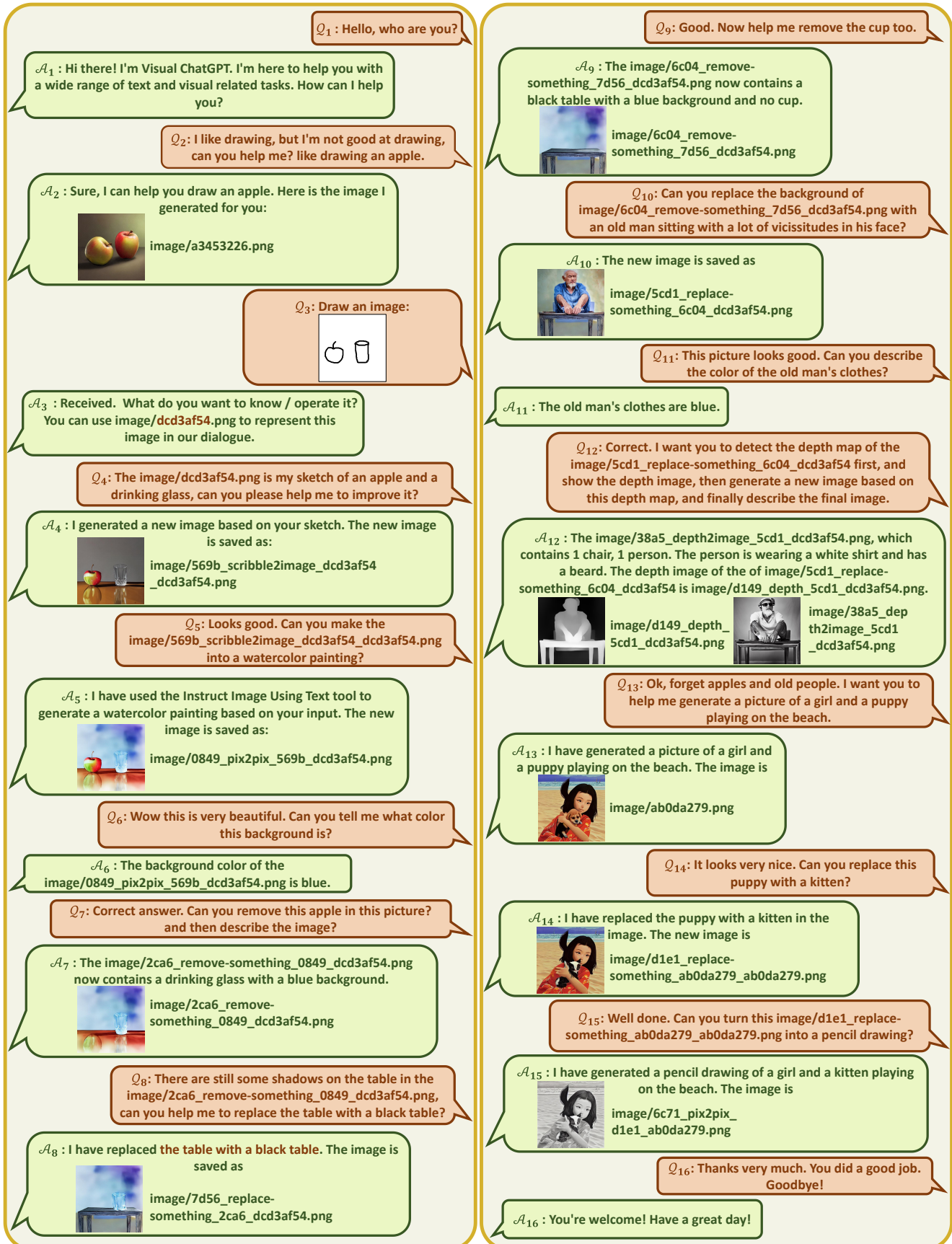
---

[1] https://github.com/hwchase17/langchain
[2] https://huggingface.co/docs/diffusers/index
[3] https://github.com/facebookresearch/MaskFormer
[4] https://github.com/lllyasviel/ControlNet

Q_1: Hello, who are you?

$\mathcal{A}_1$: Hi there! I'm Visual ChatGPT. I'm here to help you with a wide range of text and visual related tasks. How can I help you?

Q_2: I like drawing, but I'm not good at drawing, can you help me? like drawing an apple.

$\mathcal{A}_2$: Sure, I can help you draw an apple. Here is the image I generated for you:

image/a3453226.png

Q_3: Draw an image:

$\mathcal{A}_3$: Received. What do you want to know / operate it? You can use image/dcd3af54.png to represent this image in our dialogue.

Q_4: The image/dcd3af54.png is my sketch of an apple and a drinking glass, can you please help me to improve it?

$\mathcal{A}_4$: I generated a new image based on your sketch. The new image is saved as:

image/569b_scribble2image_dcd3af54_dcd3af54.png

Q_5: Looks good. Can you make the image/569b_scribble2image_dcd3af54_dcd3af54.png into a watercolor painting?

$\mathcal{A}_5$: I have used the Instruct Image Using Text tool to generate a watercolor painting based on your input. The new image is saved as:

image/0849_pix2pix_569b_dcd3af54.png

Q_6: Wow this is very beautiful. Can you tell me what color this background is?

$\mathcal{A}_6$: The background color of the image/0849_pix2pix_569b_dcd3af54.png is blue.

Q_7: Correct answer. Can you remove this apple in this picture? and then describe the image?

$\mathcal{A}_7$: The image/2ca6_remove-something_0849_dcd3af54.png now contains a drinking glass with a blue background.

image/2ca6_remove-something_0849_dcd3af54.png

Q_8: There are still some shadows on the table in the image/2ca6_remove-something_0849_dcd3af54.png, can you help me to replace the table with a black table?

$\mathcal{A}_8$: I have replaced the table with a black table. The image is saved as

image/7d56_replace-something_2ca6_dcd3af54.png

Q_9: Good. Now help me remove the cup too.

$\mathcal{A}_9$: The image/6c04_remove-something_7d56_dcd3af54.png now contains a black table with a blue background and no cup.

image/6c04_remove-something_7d56_dcd3af54.png

Q_{10}: Can you replace the background of image/6c04_remove-something_7d56_dcd3af54.png with an old man sitting with a lot of vicissitudes in his face?

$\mathcal{A}_{10}$: The new image is saved as

image/5cd1_replace-something_6c04_dcd3af54.png

Q_{11}: This picture looks good. Can you describe the color of the old man's clothes?

$\mathcal{A}_{11}$: The old man's clothes are blue.

Q_{12}: Correct. I want you to detect the depth map of the image/5cd1_replace-something_6c04_dcd3af54 first, and show the depth image, then generate a new image based on this depth map, and finally describe the final image.

$\mathcal{A}_{12}$: The image/38a5_depth2image_5cd1_dcd3af54.png, which contains 1 chair, 1 person. The person is wearing a white shirt and has a beard. The depth image of the of image/5cd1_replace-something_6c04_dcd3af54 is image/d149_depth_5cd1_dcd3af54.png.

image/d149_depth_5cd1_dcd3af54.png

image/38a5_depth2image_5cd1_dcd3af54.png

Q_{13}: Ok, forget apples and old people. I want you to help me generate a picture of a girl and a puppy playing on the beach.

$\mathcal{A}_{13}$: I have generated a picture of a girl and a puppy playing on the beach. The image is

image/ab0da279.png

Q_{14}: It looks very nice. Can you replace this puppy with a kitten?

$\mathcal{A}_{14}$: I have replaced the puppy with a kitten in the image. The new image is

image/d1e1_replace-something_ab0da279_ab0da279.png

Q_{15}: Well done. Can you turn this image/d1e1_replace-something_ab0da279_ab0da279.png into a pencil drawing?

$\mathcal{A}_{15}$: I have generated a pencil drawing of a girl and a kitten playing on the beach. The image is

image/6c71_pix2pix_d1e1_ab0da279.png

Q_{16}: Thanks very much. You did a good job. Goodbye!

$\mathcal{A}_{16}$: You're welcome! Have a great day!

Figure 4. Multiple rounds of dialogue between humans and Visual ChatGPT. In the dialogues, Visual ChatGPT can understand human intents, support the language and image inputs, and accomplish complex visual tasks such as generation, question, and editing.

Figure 5. Case study of prompt managing of system principles. We qualitatively analyze the four proposals: file name sensitivity, reasoning format strictness, reliability, and chain of thoughts. The top-left shows whether emphasizing the file name sensitivity in $\mathcal{M}(\mathcal{P})$ affects the file reference accuracy. Further parsing cannot be performed for the top-right points without reasoning format strictness. The bottom-left shows the difference in whether to tell Visual ChatGPT to be loyal to tool observation rather than faking image content. The bottom-right shows emphasizing the ability to use tools in a chain will help the decision.
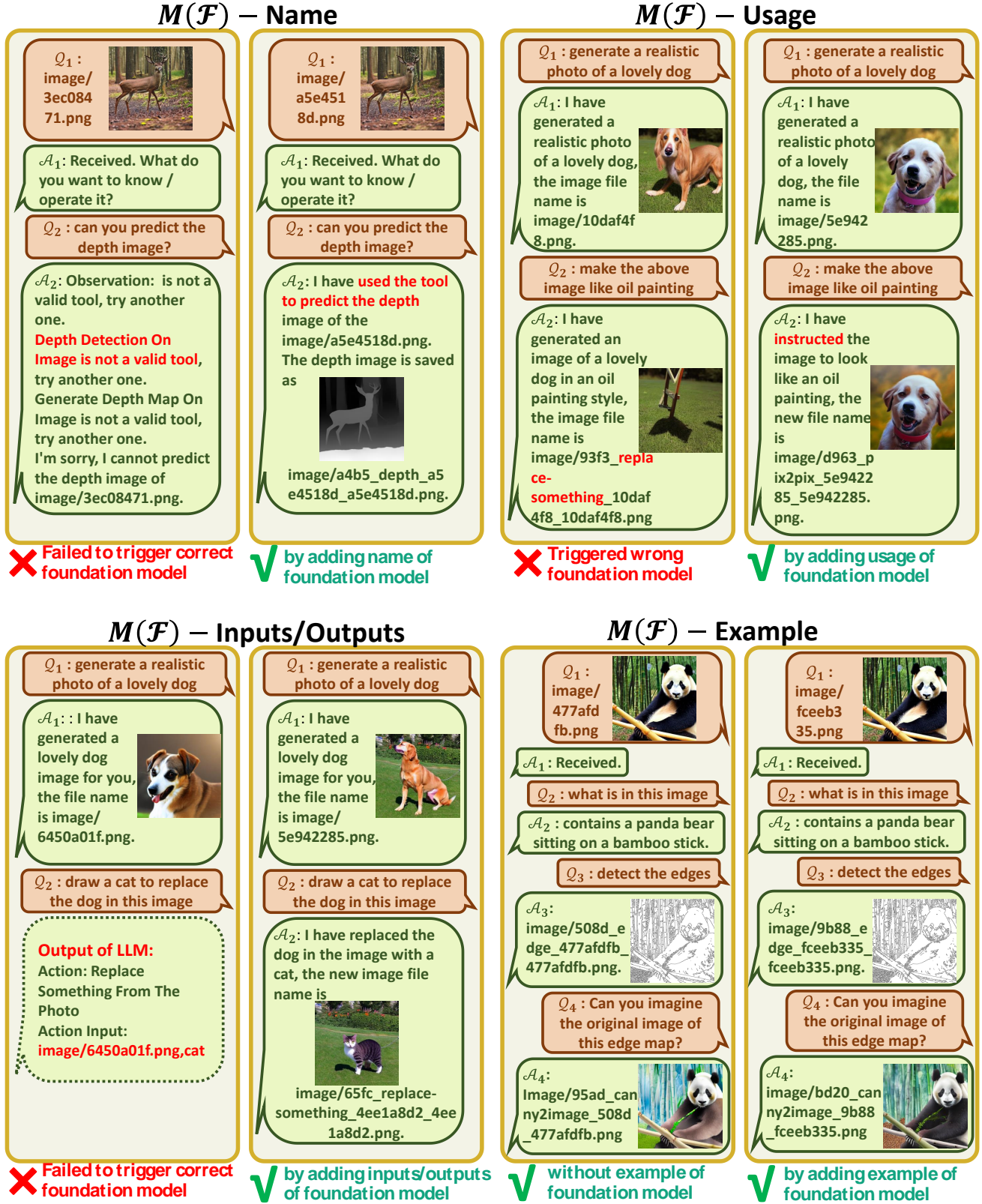
$M(\mathcal{F})$ − Name

$\mathcal{Q}_1$: image/3ec08471.png

$\mathcal{A}_1$: Received. What do you want to know / operate it?

$\mathcal{Q}_2$: can you predict the depth image?

$\mathcal{A}_2$: Observation: is not a valid tool, try another one. **Depth Detection On Image is not a valid tool**, try another one. **Generate Depth Map On Image is not a valid tool,** try another one. I'm sorry, I cannot predict the depth image of image/3ec08471.png.

❌ **Failed to trigger correct foundation model**

$\mathcal{Q}_1$: image/a5e4518d.png

$\mathcal{A}_1$: Received. What do you want to know / operate it?

$\mathcal{Q}_2$: can you predict the depth image?

$\mathcal{A}_2$: I have **used the tool to predict the depth** image of the image/a5e4518d.png. The depth image is saved as

image/a4b5_depth_a5e4518d_a5e4518d.png.

✓ **by adding name of foundation model**

$M(\mathcal{F})$ − Usage

$\mathcal{Q}_1$: generate a realistic photo of a lovely dog

$\mathcal{A}_1$: I have generated a realistic photo of a lovely dog, the image file name is image/10daf4f8.png.

$\mathcal{Q}_2$: make the above image like oil painting

$\mathcal{A}_2$: I have generated an image of a lovely dog in an oil painting style, the image file name is image/93f3_**replace-something**_10daf4f8_10daf4f8.png

❌ **Triggered wrong foundation model**

$\mathcal{Q}_1$: generate a realistic photo of a lovely dog

$\mathcal{A}_1$: I have generated a realistic photo of a lovely dog, the file name is image/5e942285.png.

$\mathcal{Q}_2$: make the above image like oil painting

$\mathcal{A}_2$: I have **instructed** the image to look like an oil painting, the new file name is image/d963_pix2pix_5e9422 85_5e942285.png.

✓ **by adding usage of foundation model**

$M(\mathcal{F})$ − Inputs/Outputs

$\mathcal{Q}_1$: generate a realistic photo of a lovely dog

$\mathcal{A}_1$: : I have generated a lovely dog image for you, the file name is image/6450a01f.png.

$\mathcal{Q}_2$: draw a cat to replace the dog in this image

Output of LLM:
Action: Replace Something From The Photo
Action Input: image/6450a01f.png,cat

❌ **Failed to trigger correct foundation model**

$\mathcal{Q}_1$: generate a realistic photo of a lovely dog

$\mathcal{A}_1$: I have generated a lovely dog image for you, the file name is image/5e942285.png.

$\mathcal{Q}_2$: draw a cat to replace the dog in this image

$\mathcal{A}_2$: I have replaced the dog in the image with a cat, the new image file name is

image/65fc_replace-something_4ee1a8d2_4ee1a8d2.png.

✓ **by adding inputs/outputs of foundation model**

$M(\mathcal{F})$ − Example

$\mathcal{Q}_1$: image/477afdfb.png

$\mathcal{A}_1$: Received.

$\mathcal{Q}_2$: what is in this image

$\mathcal{A}_2$: contains a panda bear sitting on a bamboo stick.

$\mathcal{Q}_3$: detect the edges

$\mathcal{A}_3$: image/508d_edge_477afdfb_477afdfb.png.

$\mathcal{Q}_4$: Can you imagine the original image of this edge map?

$\mathcal{A}_4$: Image/95ad_canny2image_508d_477afdfb.png

✓ **without example of foundation model**

$\mathcal{Q}_1$: image/fceeb335.png

$\mathcal{A}_1$: Received.

$\mathcal{Q}_2$: what is in this image

$\mathcal{A}_2$: contains a panda bear sitting on a bamboo stick.

$\mathcal{Q}_3$: detect the edges

$\mathcal{A}_3$: image/9b88_edge_fceeb335_fceeb335.png.

$\mathcal{Q}_4$: Can you imagine the original image of this edge map?

$\mathcal{A}_4$: image/bd20_canny2image_9b88_fceeb335.png

✓ **by adding example of foundation model**

Figure 6. Case study of prompt managing of foundation models. We qualitatively analyze the four proposals: Name, Usage, Inputs/Outputs, and Example. The top-left shows that Visual ChatGPT will guess the tool name and then fails to use the correct tool without the tool name. The top-right shows that when the usage of the tool name is missing or unclear, it will call other tools or encounters an error. The bottom-left shows that the lack of inputs/outputs format requirements will lead to wrong parameters. The bottom-right shows that the example sometimes is optional because the ChatGPT is able to summarize the historical information and human intents to use the correct tool.

Figure 7. Case study of prompt managing of the user query and model output. We qualitatively analyze the four proposals: unique file names, force VFM thinking, chained file names, and ask for more details. The top-left shows that unique file names avoid overwriting. The top-right shows force VFM thinking encourages tool invoking and strict thinking format. The bottom-left shows chained file naming helps understand files and can be successfully observed and concluded by Visual ChatGPT. The bottom-right shows Visual ChatGPT is able to detect ambiguous references and ask for more details.

4.3. Case Study of Prompt Manager

Case Study of prompt managing of system principles is analyzed in Fig. 5. To validate the effectiveness of our system principle prompts, we remove different parts from it to compare model performance. Each removal will result in different capacity degradation.

Case Study of prompt managing of foundation models is analyzed in Fig. 6. The name of the VFM is the most important and needs to be clearly defined. When the name is missing or ambiguous, Visual ChatGPT will guess it many times until it finds an existing VFM, or encounters an error, as the top-left figure shows. The VFM usage should clearly describe the specific scenario where a model should be used to avoid the wrong responses. The top-right figure shows that the style transfer is mishandled to the replacement. The input and output format should be prompted accurately to avoid parameter errors, as shown in the bottom-left. The example prompt can help the model deal with complex usages but is optional. As shown in the bottom-right figure, although we delete the example prompt, ChatGPT can also summarize the dialogue history and human intents to use the correct VFM. The complete visual foundation model prompts is shown in Appendix A.

Case Study of prompt managing of user query is analyzed in Fig7 upper part. The top-left figure shows that without image file unique naming, newly uploaded image file might be renamed to avoid overwritten and result in wrong reference. As shown in the top-right figure, by moving the thought guidance from $\mathcal{M}(\mathcal{P})$ to $\mathcal{M}(\mathcal{Q})$ and making it spoken in Visual ChatGPT's voice as a force thinking, invoking more VFM is emphasized rather than imagination based on textual context as compared in $\mathcal{Q}_2$. By forcing Visual ChatGPT to say "Thought: Do I need to use a tool?", $\mathcal{M}(\mathcal{Q})$ makes it easier to pass regex match correctly. In contrast, without force thinking, $\mathcal{A}_3$ may wrongly generate the end of thoughts token and directly consider all of its ChatGPT outputs as the final response.

Case Study of prompt managing of model outputs is analyzed in Fig7 bottom part. The bottom-left picture compares the performance of removing and keeping the chained naming rule. With the chained naming rule, Visual ChatGPT can recognize the file type, trigger the correct VFM, and conclude the file dependency relationship naming rule. It shows that the chained naming rule does help Visual ChatGPT to understand. The bottom-right picture gives an example of asking for more details when the item inference is ambiguous, which also indicates the safety of our system.

5. Limitations

Although Visual ChatGTP is a promising approach for multi-modal dialogue, it has some limitations, including:

- **Dependence on ChatGPT and VFMs** Visual ChatGPT relies heavily on ChatGPT to assign tasks and on VFMs to execute them. The performance of Visual ChatGPT is thus heavily influenced by the accuracy and effectiveness of these models.

- **Heavy Prompt Engineering** Visual ChatGPT requires a significant amount of prompt engineering to convert VFMs into language and make these model descriptions distinguishable. This process can be time-consuming and requires expertise in both computer vision and natural language processing.

- **Limited Real-time Capabilities** Visual ChatGPT is designed to be general. It tries to decompose a complex task into several subtasks automatically. Thus, when handling a specific task, Visual ChatGPT may invoke multiple VFMs, resulting in limited real-time capabilities compared to expert models specifically trained for a particular task.

- **Token Length Limitation** The maximum token length in ChatGPT may limit the number of foundation models that can be used. If there are thousands or millions of foundation models, a pre-filter module may be necessary to limit the VFMs fed to ChatGPT.

- **Security and Privacy** The ability to easily plug and unplug foundation models may raise security and privacy concerns, particularly for remote models accessed via APIs. Careful consideration and automatic check must be given to ensure that sensitive data should not be exposed or compromised.

6. Conclusion

In this work, we propose Visual ChatGPT, an open system incorporating different VFMs and enabling users to interact with ChatGPT beyond language format. To build such a system, we meticulously design a series of prompts to help inject the visual information into ChatGPT, which thus can solve the complex visual questions step-by-step. Massive experiments and selected cases have demonstrated the great potential and competence of Visual ChatGPT for different tasks. Apart from the aforementioned limitations, another concern is that some generation results are unsatisfied due to the failure of VFMs and the instability of the prompt. Thus, one self-correction module is necessary for checking the consistency between execution results and human intentions and accordingly making the corresponding editing. Such self-correction behavior can lead to more complex thinking of the model, significantly increasing the inference time. We will solve such an issue in the future.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, 2022.

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[3] Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *arXiv preprint arXiv:2111.02358*, 2021.

[4] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800*, 2022.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

[7] Harrison Chase. LangChain, 10 2022.

[8] Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18030–18040, 2022.

[9] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX*, pages 104–120. Springer, 2020.

[10] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021.

[11] Ernest Davis and Gary Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 2015.

[12] Constantin Eichenberg, Sidney Black, Samuel Weinbach, Letitia Parcalabescu, and Anette Frank. Magma–multimodal augmentation of generative models through adapter-based finetuning. *arXiv preprint arXiv:2112.05253*, 2021.

[13] Tsu-Jui Fu, Linjie Li, Zhe Gan, Kevin Lin, William Yang Wang, Lijuan Wang, and Zicheng Liu. Violet: End-to-end video-language transformers with masked visual-token modeling. *arXiv preprint arXiv:2111.12681*, 2021.

[14] Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. Large-scale adversarial training for vision-and-language representation learning. *Advances in Neural Information Processing Systems*, 33:6616–6628, 2020.

[15] Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. Semantic compositional networks for visual captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5630–5639, 2017.

[16] Geonmo Gu, Byungsoo Ko, SeoungHyun Go, Sung-Hyun Lee, Jingeun Lee, and Minchul Shin. Towards light-weight and real-time line segment detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 726–734, 2022.

[17] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[20] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

[21] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. Manigan: Text-guided image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7880–7889, 2020.

[22] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[23] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022.

[24] Kunchang Li, Yali Wang, Junhao Zhang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unifying convolution and self-attention for visual recognition. *arXiv preprint arXiv:2201.09450*, 2022.

[25] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*, pages 121–137. Springer, 2020.

[26] Yangguang Li, Feng Liang, Lichen Zhao, Yufeng Cui, Wanli Ouyang, Jing Shao, Fengwei Yu, and Junjie Yan. Supervision exists everywhere: A data efficient contrastive

language-image pre-training paradigm. In *International Conference on Learning Representations*.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[28] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, 2023.

[29] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[30] 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.

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[33] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021.

[34] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020.

[35] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[36] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[37] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[38] Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[41] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

[43] 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, Oct. 2020. Association for Computational Linguistics.

[44] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

[45] Zhao Xu, Xu Baojie, and Wu Guoxin. Canny edge detection based on open cv. In *2017 13th IEEE international conference on electronic measurement & instruments (ICEMI)*, pages 53–56. IEEE, 2017.

[46] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Yumao Lu, Zicheng Liu, and Lijuan Wang. An empirical study of gpt-3 for few-shot knowledge-based vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3081–3089, 2022.

[47] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

[48] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6720–6731, 2019.

[49] Rowan Zellers, Jiasen Lu, Ximing Lu, Youngjae Yu, Yanpeng Zhao, Mohammadreza Salehi, Aditya Kusupati, Jack Hessel, Ali Farhadi, and Yejin Choi. Merlot reserve: Neural script knowledge through vision and language and sound. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16375–16387, 2022.

[50] Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo,

Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[51] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

[52] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18123–18133, 2022.

[53] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.

[54] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. Vinvl: Making visual representations matter in vision-language models. *arXiv preprint arXiv:2101.00529*, 1(6):8, 2021.

[55] Tianhao Zhang, Hung-Yu Tseng, Lu Jiang, Weilong Yang, Honglak Lee, and Irfan Essa. Text as neural operator: Image manipulation by text instruction. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1893–1902, 2021.

[56] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

[57] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

[58] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

A. Tool Details

- **Remove Something From The Photo:**

  Model: "runwayml/stable-diffusion-inpainting" from Huggingface library, StableDiffusionInpaint-Pipeline model; "CIDAS/clipseg-rd64-refined" from Huggingface library, CLIPSegForImage-Segmentation model.

  InOut: image_path, textual what to remove → image_path

  Prompt: Remove something from the photo: useful for when you want to remove and object or something from the photo from its description or location. The input to this tool should be a comma seperated string of two, representing the image_path and the object need to be removed.

- **Replace Something From The Photo:**

  Model: "runwayml/stable-diffusion-inpainting" from Huggingface library, StableDiffusionInpaint-Pipeline model; "CIDAS/clipseg-rd64-refined" from Huggingface library, CLIPSegForImage-Segmentation model.

  InOut: image_path, textual what to replace, textual what to add → image_path

  Prompt: Replace something from the photo: useful for when you want to replace an object from the object description or location with another object from its description. The input to this tool should be a comma seperated string of three, representing the image_path, the object to be replaced, the object to be replaced with.

- **Instruct Image Using Text:**

  Model: "timbrooks/instruct-pix2pix" from Hugging-Face, StableDiffusionInstructPix2PixPipeline model.

  InOut: image_path, textual how to modify → image_path

  Prompt: Instruct image using text: useful for when you want to the style of the image to be like the text. like: make it look like a painting. or make it like a robot. The input to this tool should be a comma seperated string of two, representing the image_path and the text.

- **Answer Question About The Image:**

  Model: "Salesforce/blip-vqa-base" from HuggingFace, BlipForQuestionAnswering model.

  InOut: image_path, question → answer

  Prompt: useful when you need an answer for a question based on an image like: what is the background color of the last image, how many cats in this figure, what is in this figure.

- **Get Photo Description:**

  Model: "Salesforce/blip-image-captioning-base" from HuggingFace library, BlipForConditionalGeneration model.

  InOut: image_path → natural language description

  Prompt: Get photo description: useful for when you want to know what is inside the photo. The input to this tool should be a string, representing the image_path.

- **Generate Image From User Input Text:**

  Model: "runwayml/stable-diffusion-v1-5" from HuggingFace library, StableDiffusionPipeline model.

  InOut: textual description → image path

  Prompt: Generate image from user input text: useful for when you want to generate an image from a user input text and it saved it to a file. The input to this tool should be a string, representing the text used to generate image.

- **Edge Detection On Image :**

  Model: Canny Edge Detector from OpenCV

  InOut: image_path → edge_image_path

  Prompt: Edge Detection On Image : useful for when you want to detect the edge of the image. like: detect the edges of this image, or canny detection on image, or peform edge detection on this image, or detect the canny image of this image. The input to this tool should be a string, representing the image_path.

- **Image Generation Condition On Canny Image:**

  Model: ControlNet for Canny Edge.

  InOut: edge_image_path, textual description → image_path

  Prompt: useful for when you want to generate a new real image from both the user desciption and a canny image. like: generate a real image of a object or something from this canny image, or generate a new real image of a object or something from this edge image. The input to this tool should be a comma seperated string of two, representing the image_path and the user description.

- **Line Detection On Image :**

  Model: M-LSD Detector for Straight Line

  InOut: image_path → line_image_path

  Prompt: Line Detection On Image : useful for when you want to detect the straight line of the image. like: detect the straight lines of this image, or straight line detection on image, or peform straight line detection on this image, or detect the straight line image of this image. The input to this tool should be a string, representing the image_path

- **Generate Image Condition On Line Image:**

  Model: ControlNet for M-LSD Lines.

  InOut: line_image_path, textual description → image_path

  Prompt: useful for when you want to generate a new real image from both the user desciption and a straight line image. like: generate a real image of a object or something from this straight line image, or generate a new real image of a object or something from this straight lines. The input to this tool should be a comma seperated string of two, representing the image_path and the user description.

- **Hed Detection On Image :**

  Model: HED Boundary Detector

  InOut: image_path → hed_image_path

  Prompt: Hed Detection On Image: useful for when you want to detect the soft hed boundary of the image. like: detect the soft hed boundary of this image, or hed boundary detection on image, or peform hed boundary detection on this image, or detect soft hed boundary image of this image. The input to this tool should be a string, representing the image_path

- **Generate Image Condition On Soft Hed Boundary Image :**

  Model: ControlNet for HED.

  InOut: hed_image_path, textual description → image_path

  Prompt: Generate Image Condition On Soft Hed Boundary Image: useful for when you want to generate a new real image from both the user desciption and a soft hed boundary image. like: generate a real image of a object or something from this soft hed boundary image, or generate a new real image of a object or something from this hed boundary. The input to this tool should be a comma seperated string of two, representing the image_path and the user description

- **Segmentation On Image :**

  Model: Uniformer Segmentation

  InOut: image_path → segment_image_path

  Prompt: useful for when you want to detect segmentations of the image. like: segment this image, or generate segmentations on this image, or peform segmentation on this image. The input to this tool should be a string, representing the image_path

- **Generate Image Condition On Segmentations :**

  Model: ControlNet for Segmentation.

  InOut: segment_image_path, textual description → image_path

  Prompt: useful for when you want to generate a new real image from both the user desciption and segmentations. like: generate a real image of a object or something from this segmentation image, or generate a new real image of a object or something from these segmentations. The input to this tool should be a comma seperated string of two, representing the image_path and the user description

- **Predict Depth On Image :**

  Model: MiDaS Depth Estimation

  InOut: image_path → depth_image_path

  Prompt: Predict Depth Map On Image : useful for when you want to detect depth of the image. like: generate the depth from this image, or detect the depth map on this image, or predict the depth for this image, the input to this tool should be a string, representing the image_path.

- **Generate Image Condition On Depth:**

  Model: ControlNet for Depth.

  InOut: depth_image_path, textual description → image_path

  Prompt: Generate Image Condition On Depth Map : useful for when you want to generate a new real image from both the user desciption and depth image. like: generate a real image of a object or something from this depth image, or generate a new real image of a object or something from the depth map, The input to this tool should be a comma seperated string of two, representing the image_path and the user description.

- **Predict Normal Map On Image :**

  Model: MiDaS Depth Estimation for Normal Map

  InOut: image_path → norm_image_path

Prompt: Predict Normal Map On Image : useful for when you want to detect norm map of the image. like: generate normal map from this image, or predict normal map of this image The input to this tool should be a string, representing the image_path

- **Generate Image Condition On Normal Map :**

  Model: ControlNet for Normal Map.

  InOut: norm_image_path, textual description → image_path

  Prompt: Generate Image Condition On Normal Map : useful for when you want to generate a new real image from both the user desciption and normal map. like: generate a real image of a object or something from this normal map, or generate a new real image of a object or something from the normal map. The input to this tool should be a comma seperated string of two, representing the image_path and the user description

- **Sketch Detection On Image :**

  Model: HED Boundary Detector

  InOut: image_path → sketch_image_path

  Prompt: Sketch Detection On Image: useful for when you want to generate a scribble of the image. like: generate a scribble of this image, or generate a sketch from this image, detect the sketch from this image. The input to this tool should be a string, representing the image_path

- **Generate Image Condition On Sketch Image :**

  Model: ControlNet for Scribble.

  InOut: sketch_image_path, textual description → image_path

  Prompt: useful for when you want to generate a new real image from both the user desciption and a scribble image. like: generate a real image of a object or something from this scribble image, or generate a new real image of a object or something from this sketch. The input to this tool should be a comma seperated string of two, representing the image_path and the user description

- **Pose Detection On Image :**

  Model: Openpose Detector

  InOut: image_path → pos_image_path

  Prompt: Pose Detection On Image: useful for when you want to detect the human pose of the image. like: generate human poses of this image, or generate a pose image from this image. The input to

this tool should be a string, representing the image_path

- **Generate Image Condition On Pose Image :**

  Model: ControlNet for Human Pose.

  InOut: pos_image_path, textual description → image_path

  Prompt: Generate Image Condition On Pose Image: useful for when you want to generate a new real image from both the user desciption and a human pose image. like: generate a real image of a human from this human pose image, or generate a new real image of a human from this pose. The input to this tool should be a comma seperated string of two, representing the image_path and the user description