Beyond Text: An LLM Agent Approach to Multimodal Reference-Guided Image Editing

Advait Gupta
University of Maryland- College Park
College Park, MD, USA
advait25@umd.edu

Rishie Raj
University of Maryland- College Park
College Park, MD, USA
rraj27@umd.edu

Nithin Skantha Murugan
University of Maryland- College Park
College Park, MD, USA
nithin10@umd.edu

Abstract

Current agentic image editing frameworks, such as CoSTA, GenArtist, and VISPROG, typically depend on explicit, text-based instructions and struggle with integrating indirect guidance from diverse modalities like audio or reference images. Conversely, large multimodal models (LMMs), including Meerkat, Next-GPT, and CoDi, exhibit strong multimodal understanding but usually require extensive end-to-end training and lack flexibility in task expansion. To bridge this gap, we propose a novel LLM-driven agent framework that handles indirect, multimodal instructions without dedicated agent retraining. Our approach employs a two-stage planning process: first, interpreting multimodal inputs through targeted subtasks (e.g., VQA, Audio QA) to extract key entities or attributes, and second, synthesizing these elements with the user's main prompt into a precise step-by-step editing program using pre-trained tools. This modular approach enables complex, flexible editing workflows grounded across modalities without extensive retraining. Though demonstrated here for image editing, our framework readily extends to other multimodal tasks, offering a scalable solution for sophisticated content manipulation.*

1. Introduction

The landscape of AI-driven image editing is undergoing rapid transformation, with increasingly sophisticated demands for nuanced and complex manipulations. While text-to-image models [9, 12] have demonstrated remarkable capabilities in generating and modifying images from textual descriptions, they often falter when faced with composite instructions requiring multi-turn editing or, crucially, when guidance is provided through *indirect multimodal references*. For instance, instructing a system to "replace the object in the image that sounds like the provided audio clip with the style of the reference image" presents a significant challenge for current paradigms.

Agentic frameworks, such as VISPROG [7], CoSTA* [6], and GenArtist [17], have emerged as a promising direction. These systems excel at decomposing complex tasks into manageable sub-problems and orchestrating specialized AI tools to execute them. However, their primary reliance on explicit, purely textual instructions limits their ability to interpret and integrate richer, non-textual cues from modalities like audio snippets or visual exemplars. Consequently, they often struggle to ground instructions that are implicitly defined by these external references.

Concurrently, Large Multimodal Models (LMMs) like MEERKAT [4], CoDi [15], and Next-GPT [18] have shown strong prowess in joint multimodal understanding. These models can often process and relate information from various modalities. Nevertheless, they typically require extensive end-to-end training on large-scale multimodal datasets and can be less flexible in adapting to new tools or expanding their task repertoire without significant retraining. This monolithic nature can also make their decision-making process less transparent and harder to debug.

The core challenge, therefore, lies in creating an image editing system that is both versatile enough to understand nuanced, indirect guidance from diverse modalities and flexible enough to leverage a dynamic set of specialized tools without prohibitive retraining costs. How can an AI agent effectively interpret what a user *means* when

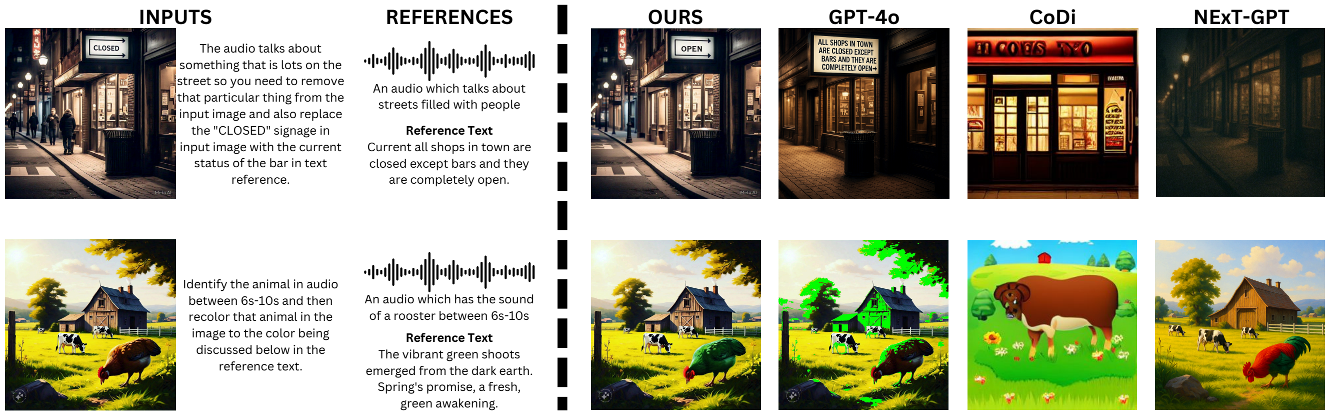

INPUTS
REFERENCES
OURS
GPT-4o
CoDi
NExT-GPT
CLOSED
The audio talks about something that is lots on the street so you need to remove that particular thing from the input image and also replace the "CLOSED" signage in input image with the current status of the bar in text reference.
An audio which talks about streets filled with people
Reference Text
Current all shops in town are closed except bars and they are completely open.
OPEN
ALL SHOPS IN TOWN ARE CLOSED EXCEPT BARS AND THEY ARE COMPLETELY OPEN
Identify the animal in audio between 6s-10s and then recolor that animal in the image to the color being discussed below in the reference text.
An audio which has the sound of a rooster between 6s-10s
Reference Text
The vibrant green shoots emerged from the dark earth. Spring's promise, a fresh, green awakening.

Figure 1. Qualitative comparison of *Beyond Text* with baseline models (GPT-4o, CoDi [15], Next-GPT [18]) on challenging image editing tasks guided by indirect multimodal references. **Top Row Example:** The instruction involves removing an object described in an audio reference (people on the street) and modifying signage text based on a textual reference indicating bars are open. *Beyond Text* correctly interprets both multimodal inputs to achieve the desired edit, while baselines struggle with one or both aspects. **Bottom Row Example:** This task requires identifying an animal (rooster) from a specific segment of an audio reference and then recoloring that animal in the main image according to a color cue ("green") provided in a separate text reference. *Beyond Text* successfully grounds the instruction across these disparate modalities, unlike the baseline methods which often fail to integrate the audio and text cues correctly for the targeted edit. These examples highlight *Beyond Text*'s enhanced capability in processing and integrating complex, indirect multimodal guidance.

they provide an audio cue or a reference image alongside a textual prompt, and then translate that understanding into a concrete sequence of editing actions?

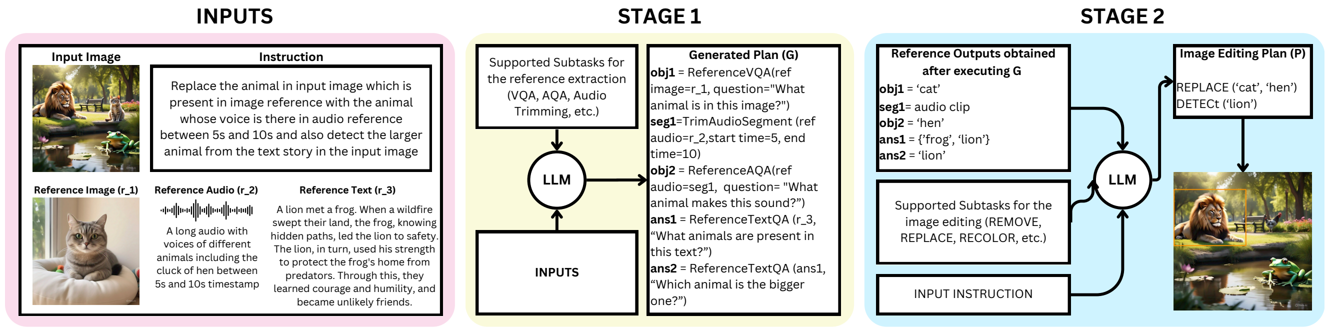
To address this gap, we introduce ***Beyond Text***, a novel LLM-driven agent framework designed explicitly for image editing tasks guided by indirect, multimodal instructions. Our approach circumvents the need for dedicated agent retraining or end-to-end LMM fine-tuning. The central hypothesis of our work is that the advanced reasoning and planning capabilities inherent in modern Large Language Models (LLMs) can be harnessed to (1) interpret and disambiguate multimodal references R by decomposing the analysis into targeted subtasks, and (2) subsequently orchestrate existing specialized tools $\mathcal{T}$ to achieve the desired edit.

Beyond Text effectuates this through a distinctive two-stage planning architecture (illustrated in Figure 3):

1. **Stage 1: Multimodal Reference Analysis Program Generation:** The LLM agent first receives the main textual prompt P_{main} and a set of multimodal references R (e.g., images, audio clips, or descriptive texts). It then generates an initial program, $Prog_{ref\_analysis}$, which calls upon specialized tools (e.g., Visual Question Answering (VQA), Audio Question Answering (AQA), etc.) to analyze these references. The goal is to extract key entities, attributes, or other salient information S_{ref} that are critical for understanding the editing intent conveyed through the references.
2. **Stage 2: Execution Program Generation:** Armed with the extracted structured information S_{ref} from the references and the original main prompt P_{main}, the LLM agent then synthesizes a precise, step-by-step editing program, $Prog_{exec}$. This program specifies the sequence of image editing operations to be performed by another set of specialized tools (e.g., object detectors, segmentation models, inpainting tools, style transfer models) to produce the final edited image.

This modular, program-based approach allows *Beyond Text* to handle complex multimodal grounding and compositional editing by leveraging existing, pre-trained specialized models. It offers enhanced interpretability, facilitates easier debugging and extension with new tools or modalities, and crucially, supports sophisticated content manipulation guided by indirect cues without requiring extensive retraining of the core agent. Examples of the programs generated in each stage are depicted in Figure 3. While this work primarily demonstrates the framework's efficacy for image editing, the underlying methodology is generalizable to a broader range of multimodal tasks requiring nuanced interpretation of diverse inputs. Our main contributions are:

- We propose *Beyond Text* a novel LLM-driven agent framework that uniquely addresses image editing guided by **indirect, multimodal references** (e.g., audio, images, text) without requiring end-to-end retraining of the agent.
- We introduce a **two-stage planning process**: first, generating a reference analysis program to interpret multimodal inputs and extract salient information, and second, generating an execution program to perform the image editing using specialized tools.
- Our framework demonstrates how to leverage the reasoning capabilities of LLMs to **explicitly ground tex-**


INPUTS
Input Image
Instruction
Replace the animal in input image which is present in image reference with the animal whose voice is there in audio reference between 5s and 10s and also detect the larger animal from the text story in the input image
Reference Image (r_1)
Reference Audio (r_2)
A long audio with voices of different animals including the cluck of hen between 5s and 10s timestamp
Reference Text (r_3)
A lion met a frog. When a wildfire swept their land, the frog, knowing hidden paths, led the lion to safety. The lion, in turn, used his strength to protect the frog's home from predators. Through this, they learned courage and humility, and became unlikely friends.
STAGE 1
Supported Subtasks for the reference extraction (VQA, AQA, Audio Trimming, etc.)
LLM
INPUTS
Generated Plan (G)
STAGE 2
Reference Outputs obtained after executing G
Supported Subtasks for the image editing (REMOVE, REPLACE, RECOLOR, etc.)
INPUT INSTRUCTION
LLM
Image Editing Plan (P)

```
obj1 = ReferenceVQA(ref image=r_1, question="What animal is in this image?")
seg1=TrimAudioSegment (ref audio=r_2,start time=5, end time=10)
obj2 = ReferenceAQA(ref audio=seg1,  question= "What animal makes this sound?”)
ans1 = ReferenceTextQA (r_3, “What animals are present in this text?”)
ans2 = ReferenceTextQA (ans1, “Which animal is the bigger one?”)
```

```
obj1 = ‘cat’
seg1= audio clip
obj2 = ‘hen’
ans1 = {‘frog’, ‘lion’}
ans2 = ‘lion’
```

```
REPLACE (‘cat’, ‘hen’)
DETECt (‘lion’)
```

Figure 2. The proposed two-stage agent process: Stage 1 uses an LLM plan (G) to extract information from multimodal references (image, audio, text) via QA subtasks. Stage 2 synthesizes this extracted info with the main instruction to generate the final image editing plan (P) which results in the final edited image.

tual instructions in information extracted from diverse modalities through targeted subtask execution before planning the final editing actions.

- We outline a modular and extensible architecture that facilitates the integration of various pre-trained unimodal and multimodal tools, offering a practical pathway towards more flexible and capable AI agents for complex content creation and manipulation.
- We release a new benchmark dataset, MultiRefEdit-Bench, specifically designed to evaluate image editing systems on tasks involving indirect multimodal references.

2. Related Work

Our research intersects with agentic frameworks, large multimodal models, and multimodal grounding techniques.

2.1. Agentic Frameworks for Vision & Language Tasks

Recent works leverage LLMs to automate tasks by orchestrating specialized tools. VISPROG [7] generates executable programs from text for visual tasks, while CoSTA* [6] emphasizes cost-sensitive planning. ViperGPT [14], HuggingGPT [13], and GenArtist [17] similarly explore task decomposition but primarily handle explicit instructions. Our framework introduces dedicated multimodal reference analysis, enhancing agent capability for interpreting indirect cues before generating final execution plans.

2.2. Large Multimodal Models & Multimodal Understanding

LMMs like MEERKAT [4], Flamingo [2], GPT-4V [10], and Gemini [5] achieve strong multimodal reasoning through extensive training on large datasets. These models demonstrate powerful capabilities but are less flexible and require substantial retraining to adapt. Our modular agent leverages components inspired by these models, maintaining flexibility without extensive end-to-end retraining.

2.3. Multimodal Grounding & Instruction Following

Multimodal grounding connects linguistic or cross-modal cues to specific data elements, as seen in works like MEERKAT [4], NExT GPT [18], and CoDi [15]. Unlike implicit grounding learned by LMMs, our approach explicitly grounds instructions through intermediate QA steps before generating the final editing program, enhancing interpretability and leveraging existing QA models effectively.

3. The *Beyond Text* Framework

This section details the architecture and operational flow of our proposed *Beyond Text* framework. Our core design philosophy is to empower a Large Language Model (LLM) agent to first deeply understand indirect multimodal references and then translate this understanding into precise image editing actions, all without requiring specialized agent retraining.

3.1. Framework Overview and Motivation

The primary challenge in advanced AI-driven image editing is to move beyond explicit textual commands and enable systems to understand instructions grounded in diverse, often *indirect*, multimodal references $R = \{r_1, r_2, \ldots, r_n\}$. Each reference r_i (be it an image, audio clip, or auxiliary text) provides cues that qualify or specify aspects of a main textual instruction P_{main} for editing an input image I_{in}. For example, P_{main} might state, ”change the object’s color to match the one in the reference image r_1” or ”remove the entity that produces the sound in audio reference r_2.” Current agentic systems [6, 7, 17] primarily excel with direct textual inputs, while Large Multimodal Models (LMMs) [4, 15, 18], though capable of multimodal understanding, often require extensive training and offer less flex-

Table 1. Overview of Subtasks and Associated Tools

Stage 1: Reference Analysis Subtasks ($\mathcal{C}_{ref}$)		**Stage 2: Image Editing Subtasks ($\mathcal{C}_{edit}$)**	
Subtask Name	Tool Name	Subtask Name	Tool Name
`ReferenceVQA`	GPT-4o	`DETECT`	GroundingDINO [8]
`ReferenceAQA`	MEERKAT [4]	`REMOVE`	Stable Diffusion Erase [12]
`ReferenceTextQA`	OpenAI o1	`REPLACE`	Stable Diffusion Inpaint [12]
`TrimAudioSegment`	pydub	`RECOLOR`	Stable Diffusion Search & Recolor [12]
		`REPLACE_TEXT`	CRAFT [3] & EasyOCR [1] for extracting the text from image, an LLM (GPT-4o) for analyzing this extracted text and is location and giving updated text and corresponding location, DeepFont [16] for extracting font information, Stable Diffusion Erase [12] for removing text, python pillow library for writing new text.
		`REMOVE_TEXT`	CRAFT [3] & EasyOCR [1] for extracting the text from image, an LLM (GPT-4o) for analyzing this extracted text and is location and giving updated text and corresponding location, DeepFont [16] for extracting font information, Stable Diffusion Erase [12] for removing text.

ibility for integrating varied specialized tools.

To bridge this gap, *Beyond Text* introduces a novel LLM-driven agentic approach. It is specifically designed to interpret these indirect multimodal instructions and orchestrate a toolkit $\mathcal{T}$ of specialized pre-trained models without necessitating end-to-end agent retraining. At its heart, *Beyond Text* employs a distinct two-stage planning and execution process, managed by an LLM agent. This process is visualized in Figure 2 which contains the structure of generated programs and intermediate data as well.

The two stages are:

Stage 1: Intelligent Multimodal Reference Interpretation: This initial stage focuses on dissecting and understanding the provided multimodal references R within the context of the main prompt P_{main}. The LLM agent generates a *reference analysis program*, $Prog_{ref\_analysis}$. Executing this program utilizes various query and analysis tools to extract specific, salient pieces of information, collectively denoted as S_{ref}. Crucially, each piece of information in S_{ref} is tagged with its source modality and a unique identifier, facilitating precise grounding for the subsequent stage.
Stage 2: Grounded Execution Program Generation: With the structured and tagged information S_{ref} in hand, the LLM agent, in this second stage, processes the original main prompt P_{main} again. This time, it can substitute the abstract references in P_{main} with the concrete information from S_{ref}. Based on this fully grounded understanding, the agent synthesizes a precise *execution program*, $Prog_{exec}$, which is a sequence of specific image editing commands designed to modify I_{in} into the desired I_{out}.

This deliberate separation of interpretation and action allows *Beyond Text* to systematically address the complexity of indirect multimodal guidance.

3.2. Stage 1: Intelligent Multimodal Reference Interpretation

3.2.1. Mechanism and Program Structure

The core objective of Stage 1 is to transform the potentially rich and ambiguous information within multimodal references R into a concise, structured, and actionable set of facts S_{ref}, directly relevant to the editing goals stated in P_{main}. This involves the LLM agent analyzing P_{main} to identify phrases alluding to external references ($r_i \in R$). For each allusion, the LLM determines the specific information required and the most suitable reference analysis subtask to extract it.

Based on this, the LLM generates $Prog_{ref\_analysis}$ as a sequence of steps. Each step in this program typically invokes a tool from the Reference Analysis Subtask Set ($\mathcal{C}_{ref}$), specifying the reference data (an r_i or an intermediate variable from a previous step) and necessary parameters (e.g., a textual question). A key feature of $Prog_{ref\_analysis}$ is how it designates and structures the outputs that directly answer the queries posed by P_{main} regarding the references. The LLM is prompted such that:

- Output variables for steps yielding final, pertinent information for P_{main} are named using a specific convention, such as `ans1`, `ans2`, etc. Intermediate results not directly answering a reference query in P_{main} are assigned other variable names (e.g., `obj1`, `obj2`).
- For each step that produces such a final answer (e.g., assigned to `ans1`), the LLM also includes a `modality_tag` (e.g., ”image”, ”audio”, ”text”) within the program step itself. This tag is intelligently assigned to indicate the modality of the original reference r_i from which this specific piece of information was ultimately derived. This tagging is crucial and persists even if intermediate processing steps involved other modalities (e.g.,

using TextQA on an image caption would still result in an "image" tag if P_{main} referred to the reference image).

Outputs of program steps are assigned to variables, which can be used in subsequent steps. The structure of $Prog_{ref\_analysis}$ and the subsequent formation of S_{ref} are exemplified in Figure 3.

Upon execution of $Prog_{ref\_analysis}$, only the outputs from 'ans' steps are collected into the structured information set S_{ref}. This set typically takes the form of a dictionary or map, where each key is an `answer_id`, and the corresponding value is an object containing the extracted `value` and its associated `modality_tag`. For instance, S_{ref} might look like: {"ans1": "value": "cat", "modality": "image" , ...}. This tagged, structured information is then passed to Stage 2. The LLM's ability to generate such structured programs, including the crucial tagging, is guided by in-context learning from few-shot examples. The algorithm for this stage is included in Appendix (Algorithm 1).

3.2.2. Supported Reference Analysis Subtasks (C_{ref})

This set defines the repertoire of operations available to the agent for dissecting and understanding the multimodal references. Our initial implementation includes, but is not limited to:

- `ReferenceVQA(ref_image, question)`: Answers a natural language `question` about a given `ref_image`.
- `ReferenceAQA(ref_audio, question)`: Answers a `question` about a `ref_audio` clip.
- `ReferenceTextQA(ref_text, question)`: Answers a `question` about a `ref_text`.
- `TrimAudioSegment(ref_audio, start_time, end_time)`: Extracts a specific segment from an audio reference.

This toolkit is designed for extensibility, allowing future integration of more sophisticated analysis capabilities. The tools used for each subtask is listed in Table 1.

3.3. Stage 2: Grounded Execution Program Generation

3.3.1. Mechanism and Program Structure

After Stage 1 has distilled the multimodal references into the structured and tagged set S_{ref}, Stage 2 translates the user's original intent in P_{main}, now fully contextualized by S_{ref}, into a concrete image editing plan, $Prog_{exec}$. The LLM agent receives P_{main} and S_{ref}. It re-analyzes P_{main}, and for each part that previously referred to an external reference, it now knows the concrete information using the corresponding 'value' from S_{ref}, guided by the 'answer_id' and 'modality_tag'. For example, if P_{main} mentioned "the animal in the reference image" and S_{ref} contains '"ans1": "value": "cat", "modality": "image" ', the LLM understands this part of the prompt now refers to "cat".

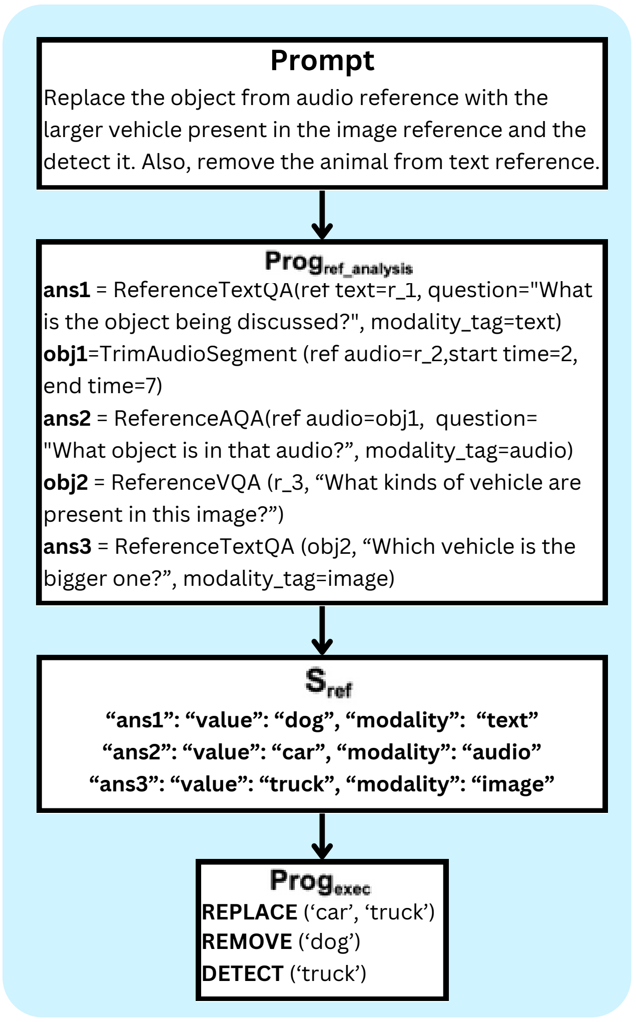

Prompt
Replace the object from audio reference with the larger vehicle present in the image reference and the detect it. Also, remove the animal from text reference.
Prog_ref_analysis
ans1 = ReferenceTextQA(ref text=r_1, question="What is the object being discussed?", modality_tag=text)
obj1=TrimAudioSegment (ref audio=r_2,start time=2, end time=7)
ans2 = ReferenceAQA(ref audio=obj1, question= "What object is in that audio?", modality_tag=audio)
obj2 = ReferenceVQA (r_3, "What kinds of vehicle are present in this image?")
ans3 = ReferenceTextQA (obj2, "Which vehicle is the bigger one?", modality_tag=image)
S_ref
"ans1": "value": "dog", "modality": "text"
"ans2": "value": "car", "modality": "audio"
"ans3": "value": "truck", "modality": "image"
Prog_exec
REPLACE ('car', 'truck')
REMOVE ('dog')
DETECT ('truck')

Figure 3. Exemplar programs part of the *Beyond Text* framework as discussed in Section 3. Given an initial textual prompt and associated multimodal references (not explicitly shown here but implied by r_1, r_2, r_3 in the $Prog_{ref\_analysis}$), Stage 1 generates and executes $Prog_{ref\_analysis}$ to produce the structured, modality-tagged information S_{ref}. Stage 2 then utilizes P_{main} (the original prompt) and S_{ref} to generate the final image editing plan, $Prog_{exec}$. This figure illustrates the transformation from the initial prompt to the intermediate S_{ref} and finally to the executable $Prog_{exec}$.

With all references resolved, the LLM identifies the core editing actions (e.g., replace, remove, detect) and translates these grounded instructions into an ordered sequence of commands from a predefined editing command set, $\mathcal{C}_{edit}$. This sequence forms $Prog_{exec}$. An example of such a $Prog_{exec}$ is shown in Figure 3 (see 'Image Editing Plan (P)'). The generation of $Prog_{exec}$ is guided by few-shot examples provided to the LLM, demonstrating how to map the combined understanding of P_{main} and S_{ref} to the editing command language. The algorithm for this stage is included

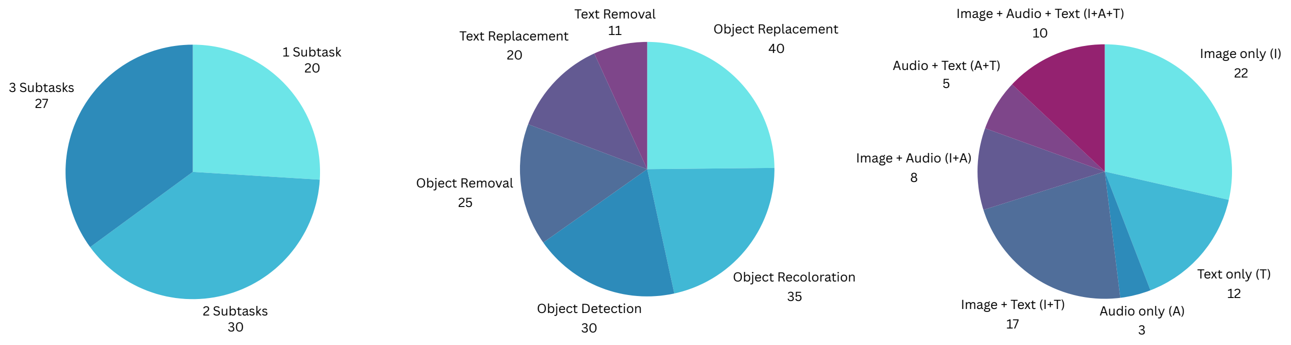


Figure 4. Distribution of the MultiRefEditBench dataset. (**Left**) Distribution of the 77 tasks by the number of subtasks per task (1, 2, or 3 subtasks). (**Center**) Distribution of the 161 total subtask instances across six primary editing categories (e.g., Object Replacement, Object Recoloration). (**Right**) Distribution of the 77 tasks based on the combination of reference modalities provided (Image only (I), Audio only (A), Text only (T), and their various combinations like I+A, I+T, A+T, I+A+T).

in Appendix (Algorithm 2).

3.3.2. Supported Image Editing Commands (C_{edit})

This set defines the elementary image manipulation operations the agent can command at a high level. Our initial set includes:

- `DETECT(object_prompt)`: Detects and localizes an object described by `object_prompt`.
- `REMOVE(object_prompt)`: Removes the specified object.
- `REPLACE(target_object_prompt, replacement_object_prompt)`: Replaces one specified object with another.
- `RECOLOR(object_prompt, color_description)`: Changes the color of a specified object.
- `REPLACE_TEXT(target_text, new_text)`: Replaces specified text within the image.
- `REMOVE_TEXT(target_text)`: Removes specified text from the image.

These commands are abstract and assume the existence of underlying specialized models capable of their execution. The tools used to execute these subtasks can be seen in Table 1.

3.4. Execution of Editing Plan

The generated $Prog_{exec}$ is a simple, ordered list of high-level editing commands. An execution engine then processes this program sequentially. For each command in $Prog_{exec}$ (e.g., `REPLACE('cat', 'hen')`), the engine invokes the corresponding specialized image editing tool from the toolkit $\mathcal{T}$. For example, a `REPLACE` command might internally trigger an object detection tool to find the 'cat', a segmentation tool to create a mask, and then an inpainting or generative fill tool with the prompt 'hen' to perform the replacement. The output of one command (the modified image) becomes the input for the next, until all commands in $Prog_{exec}$ are executed, yielding the final edited image I_{out}.

4. Experiments & Results

This section details the experimental setup designed to evaluate our proposed two-stage agent framework, the evaluation methodology employed, the baselines used for comparison, and presents results on current version of the benchmark dataset.

4.1. Experimental Setup

Dataset To rigorously evaluate the agent's ability to handle diverse multimodal references and complex instructions, we are constructing a manually curated benchmark dataset, named MultiRefEditBench. This dataset aims to include over 150 unique image editing scenarios. Each scenario consists of an input image (I_{in}), a main textual prompt (P_{main}), a set of multimodal references (R), and the ground truth edited image (I_{gt}). The dataset is designed to provide comprehensive coverage across all supported editing subtasks ($\mathcal{C}_{edit}$) and all combinations of reference modalities (Image-only, Audio-only, Text-only, Image+Audio, Image+Text, Audio+Text, Image+Audio+Text) with each task having 1-3 subtasks. The construction of this benchmark is currently underway and currently has around 77 tasks which are used for evaluation. More details about the dataset construction are included in Appendix 7. Figure 4 shows the distribution of different types of tasks and references in the current dataset.

4.2. Evaluation Criteria & Methodology

To assess the performance of our agent and the baselines, we employ a rigorous human evaluation process, focusing on the semantic correctness and fidelity of the final edited

Table 2. Results: Performance ($A(T)$) Across Editing Subtasks and Reference Types. Best "Ours" performance per subtask (row-wise) is in red, and best "Ours" performance per reference modality (column-wise, if not already a row maximum) is in green.

Editing Subtask	Beyond Text (Ours) Accuracy								Baselines Accuracy		
	Img	Aud	Txt	I+A	I+T	A+T	I+A+T	Avg. Ours	CoDi tang2023codi	GPT-4o	Next-GPT wu2023nextgpt
REPLACE	0.93	0.87	0.95	0.85	0.94	0.86	0.83	**0.91**	0.31	0.76	0.30
RECOLOR	0.94	0.88	0.96	0.85	0.95	0.87	0.84	**0.92**	0.30	0.80	0.30
DETECT	0.96	0.90	0.97	0.87	0.95	0.88	0.86	**0.93**	X	X	X
REMOVE	0.94	0.87	0.96	0.85	0.95	0.89	0.85	**0.92**	0.30	0.79	0.31
REPLACE_TEXT	0.91	0.85	0.91	0.82	0.88	0.88	0.80	**0.88**	0.15	0.45	0.14
REMOVE_TEXT	0.92	0.86	0.92	0.82	0.91	0.87	0.81	**0.89**	0.18	0.68	0.18

image I_{out} with respect to the main prompt P_{main} and the provided references R.

Subtask-Level Scoring For each test case, human evaluators assess the output of every individual editing subtask implicitly performed (e.g., was the correct object identified based on the reference? Was the replacement accurate? Was the detection correct?). Each subtask s_i within a task T is assigned a score $A(s_i) \in [0, 1]$. A score of 1 indicates full correctness, 0 indicates failure, and partial scores ($x \in (0, 1)$, e.g., 0.3, 0.5, 0.7, 0.9) are assigned based on predefined criteria for partial success (e.g., correct object replaced but with minor visual artifacts, correct detection but slightly inaccurate bounding box). The predefined criteria are the same as used by authors of CoSTA* [6].

Task-Level Accuracy The accuracy for a complete task T, denoted $A(T)$, is calculated as the mean correctness score across all its constituent subtasks: $A(T) = \frac{1}{|S_T|}\sum_{i=1}^{|S_T|} A(s_i)$, where S_T is the set of subtasks for task T.

Overall Accuracy The final reported accuracy for a category (e.g., all tasks involving REPLACE with audio+text references) or the entire dataset is the average of the task-level accuracies $A(T)$ over all evaluated tasks in that category or the dataset, respectively.

This multi-level evaluation provides a nuanced understanding of the agent's performance, pinpointing failures at both the subtask and overall task levels. More details about the evaluation strategy are included in the Appendix 6.

4.3. Emphasis on Human Assessment

While automated metrics such as CLIP similarity [11] are commonly employed in evaluating image generation and editing tasks, we primarily rely on human evaluation to assess the performance of *Beyond Text* for tasks involving complex, multi-step, and indirect multimodal instructions.

The decision to prioritize human assessment stems from the limitations of current automated metrics in capturing the nuanced aspects of success in such scenarios. For instance, CLIP scores may reflect overall visual similarity but can often overlook critical local errors, semantic inconsistencies arising from misinterpretation of indirect references (e.g., an incorrect object identified from an audio cue), or the subtle but incorrect application of an edit based on a visual reference. The complexity of grounding instructions across multiple modalities and then executing a sequence of edits means that a final image might be globally similar to a target yet fail in specific, instruction-critical ways that automated metrics might not penalize appropriately.

The COSTA paper [6] provides a detailed analysis, including correlation studies (see Section 5.2 and Appendix J in [6]), which indicate a weak correlation between CLIP similarity scores and human judgments of accuracy for complex, multi-turn editing tasks. Their findings show that images with high CLIP similarity can still contain significant errors as perceived by humans. Given that our framework, *Beyond Text*, deals with an added layer of complexity through indirect multimodal reference interpretation, we anticipate similar, if not more pronounced, discrepancies. Therefore, to ensure a robust and reliable assessment of whether the agent correctly interprets and executes the full intent behind the combined textual and multimodal instructions, human evaluation of semantic correctness and edit fidelity is indispensable.

4.4. Baselines

We compare our proposed agent against several state-of-the-art models and frameworks:

1. **CoDi [15] / Next-GPT [18]:** Two great contenders supporting series of different kinds of tasks and modalities as inputs. We compare them on tasks which can be supported by them using the current mechanism.
2. **GPT-4o:** We test the ability of this VLM to perform the editing task directly by providing all inputs (input image, main prompt, text descriptions/transcriptions of references) and evaluating the generated output image or its description.

Comparisons are made only on the subset of tasks and reference types that each baseline can plausibly support. Unsupported scenarios are marked appropriately in the results tables.

4.5. Results and Analysis

Table 2 presents the quantitative outcomes of our *Beyond Text* framework compared against various baselines across different multimodal reference scenarios. The performance of our agent (`Ours`) is broken down by the combination of reference modalities used. Baselines are evaluated on scenarios they can plausibly support, with 'X' denoting an unsupported task.

Quantitative Performance Our results consistently demonstrate that *Beyond Text* outperforms the baseline approaches across nearly all evaluated scenarios and editing subtasks. The two-stage approach, focusing first on explicit multimodal reference interpretation (Stage 1) before grounded execution planning (Stage 2), proves highly effective in correctly understanding and applying user instructions. Notably, performance on tasks involving audio references (e.g., 'Aud' or 'I+A' columns) tends to be slightly lower than those relying solely on image or text references. This can be attributed to the inherent complexities and potential ambiguities in interpreting audio content compared to the more structured nature of visual or textual information. However, even in these challenging audio-grounded scenarios, *Beyond Text* maintains a significant advantage over baselines that struggle to incorporate such indirect cues.

Qualitative Analysis Beyond quantitative metrics, qualitative examples provide crucial insights into the practical capabilities of *Beyond Text*. Figure 1 showcases several successful editing instances achieved by our framework across diverse and complex multimodal reference scenarios, alongside comparative outputs from baseline methods. These examples visually demonstrate *Beyond Text*'s superior ability to accurately interpret indirect guidance from combined image, audio, and text references, leading to edits that more faithfully reflect the user's nuanced intent.

Summary of Results In summary, the experimental results validate the core hypothesis of our work: by explicitly dedicating a stage to interpreting multimodal references before planning execution, *Beyond Text* achieves a superior understanding of complex, indirect instructions. This leads to significantly improved performance in multimodal reference-guided image editing compared to existing approaches. The framework's modularity and reliance on LLM reasoning offer a flexible and powerful paradigm for tackling sophisticated content manipulation tasks.

4.6. Ablation Studies

To understand the contribution of key architectural decisions in *Beyond Text*, we conducted several ablation studies. We focus on the impact of our two-stage planning process.

1. **Stage 1 Removed (Direct Execution Planning):** In this configuration, we bypass Stage 1 entirely. The LLM agent is provided with the input image I_{in}, the main prompt P_{main}, and all raw multimodal references R directly. It is then tasked with generating the final image editing plan ($Prog_{exec}$) in a single step, attempting to interpret references and plan execution simultaneously.
2. **Single-Stage Combined Planning:** Here, while we don't entirely remove the concept of reference analysis, we merge Stage 1 and Stage 2. The LLM agent is asked to generate a single, unified program that interleaves reference analysis steps (like VQA, AQA) with image editing commands. The agent must manage the flow of information from reference analysis to editing within this single plan.

Table 3 compares the overall accuracy of our full *Beyond Text* framework against these two ablated versions.

Table 3. Ablation Study: Impact of Framework Stages on Overall Accuracy.

Approach Configuration	**Overall Accuracy (%)**
Beyond Text (Ours - Two Stages)	**0.92%**
Stage 1 Removed - Direct Execution Planning	**0.65%**
Single-Stage Combined Planning	**0.73%**

The results from these ablations demonstrate a significant drop in accuracy for both ablated versions, underscoring the importance of the dedicated reference interpretation stage (Stage 1) and the structured two-stage approach for handling complex multimodal instructions effectively. Removing or overly simplifying the reference interpretation process leada to increased ambiguity and misinterpretation of the indirect multimodal cues, resulting in less accurate final edits.

5. Conclusion

In this paper, we introduced *Beyond Text*, a novel LLM-driven agent framework designed to address the critical challenge of enabling AI image editing systems to interpret and act upon complex instructions grounded in indirect multimodal references—a common limitation in existing agentic frameworks and large multimodal models. Our core contribution is a distinct two-stage planning process: an initial stage for intelligent multimodal reference interpretation, where an LLM agent generates a program to analyze references (images, audio, text) and extract salient, modality-tagged information; and a second stage that uses this structured information to ground the main textual prompt, enabling the synthesis of a precise execution program. Experiments demonstrate that this explicit separation of interpretation and execution allows *Beyond Text* to achieve a superior understanding of nuanced, indirect instructions, leading to more accurate and semantically consistent image edits. The modular architecture of *Beyond Text* not only facilitates easier debugging and extension but also offers a practical, flexible pathway towards more capable AI agents. While demonstrated for image editing, the underlying principles of decomposing complex multimodal understanding, explicit information grounding, and leveraging LLM reasoning for program synthesis are broadly applicable.

References

[1] Jaided AI. Easyocr. `https://github.com/JaidedAI/EasyOCR`, 2020. 4

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Nando de Freitas Garcia, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 23716–23736, 2022. 3

[3] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. CRAFT: Character region awareness for text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9365–9374, 2019. 4

[4] Sanjoy Chowdhury, Sayan Nag, Subhrajyoti Dasgupta, Jun Chen, Mohamed Elhoseiny, Ruohan Gao, and Dinesh Manocha. MEERKAT: Audio-visual large language model for grounding in space and time, 2024. 1, 3, 4

[5] Gemini Team and Google. Gemini: A family of highly capable multimodal models, 2023. 3

[6] Advait Gupta, NandaKiran Velaga, Dang Nguyen, and Tianyi Zhou. COSTA: Cost-sensitive toolpath agent for multi-turn image editing, 2024. 1, 3, 7

[7] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14953–14962, 2023. 1, 3

[8] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection, 2023. 4

[9] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models, 2022. 1

[10] OpenAI. Gpt-4v(ision) system card. `https://cdn.openai.com/papers/GPTV_System_Card.pdf`, 2023. 3

[11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 7

[12] 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 (CVPR)*, pages 10684–10695, 2022. 1, 4

[13] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving ai tasks with chatgpt and its friends in hugging face, 2023. 3

[14] Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19957–19968, 2023. 3

[15] Zineng Tang, Ziyi Zhang, Zhitong Liu, Yougan Jiang, Yingyan Wang, Wenchang Wang, Zili Luo, Ze Liu, Jing Liu, Michael Zeng, Weiming Chen, and Yong Zhang. CoDi: Any-to-any generation via composable diffusion, 2023. 1, 2, 3, 7

[16] Zhangyang Wang, Jianchao Yang, Hailin Jin, Eli Shechtman, Aseem Agarwala, Jonathan Brandt, and Thomas S. Huang. Deepfont: Identify your font from an image, 2015. 4

[17] Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. GenArtist: Multimodal llm as an agent for unified image generation and editing, 2024. 1, 3

[18] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. Next-GPT: Any-to-any multimodal llm, 2023. 1, 2, 3, 7

Beyond Text: An LLM Agent Approach to Multimodal Reference-Guided Image Editing

Supplementary Material

Table 4. Predefined Rules for Assigning Partial Correctness Scores in Human Evaluation (identical to COSTA [6], Table 8).

Task Type	Evaluation Criteria	Assigned Score
Image-Only Tasks	Minor artifacts, barely noticeable distortions	0.9
	Some visible artifacts, but main content is unaffected	0.8
	Noticeable distortions, but retains basic correctness	0.7
	Significant artifacts or blending issues	0.5
	Major distortions or loss of key content	0.3
	Output is almost unusable, but some attempt is visible	0.1
Text+Image Tasks	Text is correctly placed but slightly misaligned	0.9
	Font or color inconsistencies, but legible	0.8
	Noticeable alignment or formatting issues	0.7
	Some missing or incorrect words but mostly readable	0.5
	Major formatting errors or loss of intended meaning	0.3
	Text placement is incorrect, missing, or unreadable	0.1

6. Human Evaluation Methodology for Accuracy Calculation

To ensure a robust and reliable assessment of our framework's performance, particularly for tasks involving complex, multi-step, and indirect multimodal instructions, we employ a detailed human evaluation process to calculate accuracy. This methodology is consistent with the approach detailed in COSTA [6] (see Appendix B in [6]). Automated metrics often fall short in capturing nuanced errors, semantic inconsistencies, or the successful interpretation of indirect cues in such complex scenarios. Human evaluation, therefore, provides a more faithful measure of task success. This section outlines our evaluation methodology, including scoring criteria and the aggregation process for determining overall performance.

6.1. Subtask-Level Accuracy Scoring

For each test case, human evaluators meticulously assess the output corresponding to every individual editing subtask s_i implicitly or explicitly performed within a broader task T. The objective is to determine if each subtask (e.g., correctly identifying an object based on an audio reference, accurately applying a color from an image reference, successfully executing a removal) was completed as intended by the main prompt P_{main} and the provided multimodal references R.

Each subtask s_i is assigned a correctness score, $A(s_i)$, based on the following scale:

$$A(s_i) = \begin{cases} 1, & \text{if the subtask is completed fully and correctly.} \\ x, & \text{if the subtask is partially correct, where} \\ & x \in \{0.1, 0.3, 0.5, 0.7, 0.8, 0.9\}. \\ 0, & \text{if the subtask execution failed entirely or} \\ & \text{produced an unusable/incorrect result.} \end{cases} \tag{1}$$

The specific value for partial correctness (x) is determined using a predefined set of task-specific criteria. These criteria, detailed in Table 4, ensure consistency in scoring across different types of editing operations and evaluators.

6.2. Task-Level Accuracy Calculation

The accuracy for a complete task T, denoted as $A(T)$, is calculated as the arithmetic mean of the correctness scores $A(s_i)$ of all its constituent subtasks $S_T = \{s_1, s_2, \dots, s_k\}$:

$$A(T) = \frac{1}{|S_T|} \sum_{s_i \in S_T} A(s_i) \tag{2}$$

This approach ensures that the task-level accuracy provides a comprehensive reflection of the performance across all required editing steps involved in fulfilling the user's complex instruction.

6.3. Overall System Accuracy

To evaluate the overall performance of *Beyond Text* (and comparative baselines) across the entire benchmark dataset, the overall accuracy, A_{overall}, is computed. This is the average of the task-level accuracies $A(T_j)$ for all evaluated tasks T_j in the dataset:

$$A_{\text{overall}} = \frac{1}{N_T} \sum_{j=1}^{N_T} A(T_j) \tag{3}$$

where N_T is the total number of unique tasks evaluated in the dataset. This multi-level evaluation strategy (subtask, task, and overall) provides a nuanced understanding of the system's capabilities and pinpoints areas of strength and potential failure.

7. Benchmark Dataset Construction: MultiRefEditBench

To rigorously evaluate the capabilities of *Beyond Text* in handling indirect multimodal references, we constructed a new benchmark dataset, termed **MultiRefEditBench**. The generation of this dataset was a structured, multi-step process designed to ensure a diverse range of editing tasks, reference types, and contextual scenarios.

7.1. Initial Prompt Generation and Curation

The foundation of our dataset lies in the main textual prompts (P_{main}) that describe the editing tasks.

1. **Automatic Prompt Generation**: We utilized a Large Language Model (LLM), specifically GPT-4o, to generate a wide array of structured prompts. These prompts were designed to cover various editing operations (e.g., object replacement, recoloration, removal, detection) and to necessitate the use of different combinations of multimodal references (image, audio, text). The LLM was guided to create scenarios where the references provided indirect cues crucial for task completion.
2. **Manual Curation and Refinement**: Each LLM-generated prompt underwent a thorough manual review and curation process by human annotators. This step was critical to:
   - Ensure logical feasibility and clarity of the editing instructions.
   - Verify that the references described were plausible and could realistically guide the edit.
   - Refine ambiguous phrasing and ensure that the prompts were challenging yet unambiguous for an ideal system.

7.2. Generation of Image and Multimodal Reference Materials

Once the main textual prompts were finalized, the corresponding visual and other reference materials were generated:

1. **Input Image Generation** (I_{in}): For each curated prompt, a synthetic input image was generated using Meta AI's generative image models. Human annotators provided detailed textual descriptions to the image generation model, derived from the prompt, to ensure that:
   - All key objects and scene elements mentioned or implied in the prompt (and its references) were present in the generated image.
   - The visual content was clear and suitable for the intended editing operations.
2. **Image Reference Generation** (r_i **- image modality**): When a task required an image reference, a separate image was generated, again using Meta AI models, specifically to contain the visual cue (e.g., a specific style, color, or object) mentioned in P_{main}.
3. **Text Reference Generation** (r_i **- text modality**): For tasks requiring textual references (e.g., a story from which an attribute is to be inferred), GPT-4o was employed to generate relevant text snippets. These were crafted to be contextually consistent with the main prompt and the input image.
4. **Audio Reference Generation** (r_i **- audio modality**): Generating specific audio references presented a unique challenge. Our process involved:
   - Searching online repositories and sound effect libraries for audio clips matching the descriptions in the prompts (e.g., a specific animal sound, a type of musical instrument).
   - Manually editing these clips, which sometimes involved trimming, isolating specific sounds, or concatenating multiple clips to create the desired audio reference.
   - Ensuring the audio quality was sufficient for clear interpretation.

All generated images and reference materials were manually reviewed to ensure they aligned with the curated prompt and were suitable for the benchmark. This iterative process of prompt generation, curation, and multimodal asset creation, with human oversight at each step, allowed us to build a dataset tailored to evaluating indirect multimodal reference grounding.

7.3. Dataset Status and Overview

The construction of MultiRefEditBench is an ongoing effort. To date, we have curated approximately seventy unique image editing tasks, each comprising an input image, a main textual prompt, and one or more multimodal references. We are continuously working to expand the dataset to cover an even broader range of complexities and reference combinations.

A detailed breakdown of the dataset's current distribution, including the types of editing tasks and the combinations of reference modalities, is provided in Figure 4.

8. Algorithms for *Beyond Text* Framework

This section provides pseudocode for the two main stages of the *Beyond Text* framework: Stage 1 (Intelligent Multimodal Reference Interpretation) and Stage 2 (Grounded Execution Program Generation).

Algorithm 1: Stage 1: Intelligent Multimodal Reference Interpretation

```
Input: $P_{main}$ (main textual prompt), $R$ (set of
    multimodal references)
Output: $S_{ref}$ (structured, tagged reference
    information)
$LLM_{agent} \leftarrow$ Initialize LLM agent;
$Prog_{ref\_analysis} \leftarrow$ empty list of program steps;
$identified\_queries \leftarrow$
  $LLM_{agent}$.AnalyzeForReferenceCues($P_{main}, R$)
foreach $query\_info \in identified\_queries$ do
    $ref\_data \leftarrow$
      GetRelevantReferenceData($query\_info, R$);
    $subtask\_tool \leftarrow$
      $LLM_{agent}$.SelectReferenceAnalysisTool($query\_info$,
    $C_{ref}$);
    $tool\_params \leftarrow$
      $LLM_{agent}$.FormulateToolParameters($query\_info$,
    $ref\_data$) $output\_var\_name \leftarrow$
      $LLM_{agent}$.DetermineOutputVarName($query\_info$)
      e.g., "ansX_description" or "tempY"
      $modality\_tag \leftarrow$ empty string;
    if $output\_var\_name$ starts with "ans" then
        $modality\_tag \leftarrow$
          $LLM_{agent}$.DetermineModalityTag($query\_info, R$)
          Tag based on original reference $r_i$
    AddStepToProgram
    ($Prog_{ref\_analysis}$, $output\_var\_name$, $subtask\_tool$,
    $tool\_params$, $modality\_tag$)
$S_{ref} \leftarrow$ empty map;
$execution\_context \leftarrow$ empty map foreach
  $step \in Prog_{ref\_analysis}$ do
    $current\_ref\_data \leftarrow$
      ResolveData($step$.tool_params.ref_input,
    $execution\_context$, R) $result$
      $\leftarrow$ ExecuteTool($step$.
    tool, $current\_ref\_data$,
      $step$.tool_params.other) $execution\_context$[$step$.
    output_var_name] $\leftarrow$ $result$;
    if $step.output\_var\_name$ starts with "ans" then
        $S_{ref}$[$step$.output_var_name] $\leftarrow$
          {value: $result$, modality: $step$.modality_tag};
return $S_{ref}$;
```

Algorithm 2: Stage 2: Grounded Execution Program Generation

```
Input: $P_{main}$ (main textual prompt), $S_{ref}$
    (structured, tagged reference information)
Output: $Prog_{exec}$ (list of editing commands)
$LLM_{agent} \leftarrow$ Initialize LLM agent;
$grounded\_P_{main} \leftarrow P_{main}$;
foreach $(ans\_id, info) \in S_{ref}$ do
    $placeholder \leftarrow$
      $LLM_{agent}$.FindPlaceholderInPrompt($ans\_id, P_{main}$)
      Identifies part of $P_{main}$ corresponding to
      $ans\_id$ $grounded\_P_{main} \leftarrow$
      Replace($grounded\_P_{main}$, $placeholder$, $info$.value);
$editing\_actions \leftarrow$
  $LLM_{agent}$.IdentifyEditingActions($grounded\_P_{main}$);
$Prog_{exec} \leftarrow$ empty list of editing commands;
foreach $action \in editing\_actions$ do
    $command \leftarrow$
      $LLM_{agent}$.SelectEditingCommand($action, \mathcal{C}_{edit}$);
    $command\_params \leftarrow$
      $LLM_{agent}$.ExtractCommandParameters($action$,
    $grounded\_P_{main}$);
    AddCommandToList($Prog_{exec}$, $command$,
    $command\_params$)$Guided by few-shot examples$
return $Prog_{exec}$;
```

9. LLM Prompt for Stage 1: Reference Analysis Program Generation

The following text is the detailed prompt provided to the Large Language Model (LLM) to guide the generation of the ‘Prog_ref_analysis‘ (Reference Analysis Program). This program is central to Stage 1 of the *Beyond Text* framework, as described in Section 3.2. The prompt outlines the task, expected inputs (including the main prompt P_{main}, multimodal references R, and available tools $\mathcal{C}_{ref}$), the precise output structure for ‘Prog_ref_analysis‘, key rules for variable naming and modality tagging, the reasoning process the LLM should follow, and several few-shot examples to illustrate the desired behavior.

You are an expert AI agent responsible for interpreting complex user instructions that involve multimodal references. Your primary task is to generate a structured "Reference Analysis Program" (Prog_ref_analysis). This program will be executed to extract specific, salient information from provided multimodal references, which will then be used to ground the user's main textual instruction.

Your Goal

Given a main textual prompt (P_main) from the user and a set of available multimodal references (R), generate a Prog_ref_analysis. This program should be a sequence of steps, where each step invokes a specific reference analysis tool to extract information.

Inputs You Will Receive

1. **P_main (Main Textual Prompt):**
   * A natural language string containing the user's primary instruction. This prompt will often contain phrases that refer to information that needs to be extracted from the multimodal references.
   * Example: "Change the color of the car to the one shown in reference image 'r_1' and replace the background music with the genre described in reference text 'r_2'."
2. **R (Set of Multimodal References):**
   * A list or dictionary of available references. Each reference will have:
     ◦ An identifier (e.g., 'r_1', 'r_2', 'r_3').
     ◦ A type (e.g., 'image', 'audio', 'text').
     ◦ (The actual data for the reference will be available to the tools during execution, not directly in this prompt).
   * Example:
     ◦ 'r_1': type 'image' (e.g., a picture of a blue car)
     ◦ 'r_2': type 'text' (e.g., "The desired music genre is upbeat jazz.")
     ◦ 'r_3': type 'audio' (e.g., a sound clip of a bird chirping)
3. **C_ref (Available Reference Analysis Subtask Tools):**
   * A predefined list of tools you can use in your program. Each tool has specific parameters.
   * Available Tools:
     ◦ `ReferenceVQA(ref_image_id, question)`: Answers a textual 'question' about the image identified by 'ref_image_id'.
     ◦ `ReferenceAQA(ref_audio_id, question)`: Answers a textual 'question' about the audio clip identified by 'ref_audio_id'.
     ◦ `ReferenceTextQA(ref_text_id_or_variable, question)`: Answers a textual 'question' about the text identified by 'ref_text_id_or_variable' (this can be an initial reference ID like 'r_2' or an output variable from a previous step).
     ◦ `TrimAudioSegment(ref_audio_id, start_time, end_time)`: Extracts a segment from the audio clip 'ref_audio_id'. 'start_time' and 'end_time' are in seconds.

Output Program (Prog_ref_analysis) Structure

Your generated program must be a sequence of lines, each representing a tool call. The format for each line is: `output_variable_name = ToolName(param1=value1, param2=value2, ..., modality_tag="tag_if_final_ans")`

Key Rules for 'output_variable_name' and 'modality_tag'

1. **Final Answer Variables:**
   * If a step directly extracts a piece of information that is needed to resolve a reference in P_main (i.e., it's a "final answer" for a part of the multimodal query), its 'output_variable_name' **MUST** follow the convention: 'ans¡N¿_¡descriptive_suffix¿'.
     ∘ '¡N¿' is a sequential number (1, 2, 3,...).
     ∘ '¡descriptive_suffix¿' should briefly indicate what the answer represents (e.g., 'ans1_color_from_image', 'ans2_sound_type', 'ans3_object_in_text').
   * For these "ans" variables ONLY, you **MUST** include the 'modality_tag' parameter in the tool call.
     ∘ 'modality_tag' should be one of: '"image"', '"audio"', or '"text"'.
     ∘ This tag **MUST** reflect the modality of the *original reference* (r_i) from which this piece of information was ultimately derived, even if intermediate steps involved other tools or modalities. For example, if you use `ReferenceTextQA` on a caption generated by `DescribeImage(r_1)`, and this answers a query about r_1, the 'modality_tag' should still be '"image"'.
2. **Intermediate Variables:**
   * If a step produces an intermediate result that is used by subsequent steps in Prog_ref_analysis but is NOT itself a final answer for P_main, its 'output_variable_name' should be different (e.g., 'temp_description', 'trimmed_clip', 'extracted_entities').
   * Do **NOT** include the 'modality_tag' parameter in the tool call for these intermediate steps.

Your Reasoning Process to Generate 'Prog_ref_analysis'

1. **Understand 'P_main':** Carefully parse 'P_main' to identify all parts that require information from the references in 'R'. Note down what information is needed and which reference it seems to relate to.
2. **Plan Extraction Steps:** For each piece of information needed:
   * Select the correct reference from 'R' (e.g., 'r_1', 'r_2').
   * Choose the most appropriate tool from 'C_ref'.
   * Formulate the necessary parameters for the tool (e.g., the question for a QA tool, time for trimming).
   * Determine if this step produces a "final answer" for 'P_main'.
     ∘ If YES: Name the output variable 'ans¡N¿_¡suffix¿' and add the correct 'modality_tag' parameter to the tool call, reflecting the original reference's type.
     ∘ If NO (it's an intermediate result): Use a temporary variable name and do not add the 'modality_tag' parameter.
3. **Handle Dependencies:** Ensure steps are ordered correctly if one step depends on the output of another (e.g., 'TrimAudioSegment' before 'ReferenceAQA' on the segment).
4. **Construct the Program:** Write out the sequence of tool calls line by line.

Few-Shot Examples

Example 1

* **P_main:** "Replace the animal in input image which is present in image reference 'r_1' with the animal whose voice is there in audio reference 'r_2' between 5s and 10s. Also, detect the larger animal from the text story in reference 'r_3'."
* **R:**
  ∘ 'r_1': type 'image' (contains a cat)
  ∘ 'r_2': type 'audio' (contains a hen's cluck between 5s-10s)
  ∘ 'r_3': type 'text' (story about a lion and a frog, lion is bigger)
* **Expected 'Prog_ref_analysis':**

```
ans1_image_animal = ReferenceVQA(ref_image_id=r_1, question="What animal is in this image
temp_audio_segment = TrimAudioSegment(ref_audio_id=r_2, start_time=5, end_time=10)
ans2_audio_animal = ReferenceAQA(ref_audio_id=temp_audio_segment, question="What animal m
temp_text_animals = ReferenceTextQA(ref_text_id_or_variable=r_3, question="What animals a
ans3_text_larger_animal = ReferenceTextQA(ref_text_id_or_variable=temp_text_animals, ques
```

Example 2

* **P_main:** "Change the color of the sky in the input image to the dominant color found in reference image 'r_1'. Then, identify the object making the sound in the provided audio reference 'r_2' and add a small icon of this object to the bottom-left corner. Finally, write a short, two-word poetic phrase related to the theme of reference text 'r_3' at the top of the image."
* **R:**
  ◦ 'r_1': type 'image' (sunset with orange/purple sky)
  ◦ 'r_2': type 'audio' (cat meowing)
  ◦ 'r_3': type 'text' (story about a lighthouse)
* **Expected 'Prog_ref_analysis':**

```
ans1_sky_color = IdentifyDominantColor(ref_image_id=r_1, region_prompt="sky", modality_ta
ans2_sound_object = ReferenceAQA(ref_audio_id=r_2, question="What animal or object is mak
temp_theme_extraction = ReferenceTextQA(ref_text_id_or_variable=r_3, question="What is th
ans3_poetic_phrase = ReferenceTextQA(ref_text_id_or_variable=temp_theme_extraction, quest
```

Example 3 (from user provided image)

* **P_main:** "Replace the object from audio reference 'r_2' with the larger vehicle present in the image reference 'r_3' and then detect it. Also, remove the animal from text reference 'r_1'."
* **R:**
  ◦ 'r_1': type 'text' (discusses a "dog")
  ◦ 'r_2': type 'audio' (contains sound of a "car" between 2s-7s)
  ◦ 'r_3': type 'image' (shows different vehicles, a "truck" is the largest)
* **Expected 'Prog_ref_analysis':**

```
ans1_text_animal_to_remove = ReferenceTextQA(ref_text_id_or_variable=r_1, question="What
temp_trimmed_audio = TrimAudioSegment(ref_audio_id=r_2, start_time=2, end_time=7)
ans2_audio_object_to_replace = ReferenceAQA(ref_audio_id=temp_trimmed_audio, question="Wh
temp_vehicles_in_image = ReferenceVQA(ref_image_id=r_3, question="What kinds of vehicles
ans3_image_larger_vehicle = ReferenceTextQA(ref_text_id_or_variable=temp_vehicles_in_imag
```

—

Now, given the following P_main, R, and C_ref, please generate the Prog_ref_analysis.