PhotoScout: Synthesis-Powered Multi-Modal Image Search

Celeste Barnaby
University of Texas at Austin
USA
celestebarnaby@utexas.edu

Qiaochu Chen
University of Texas at Austin
USA
qchen@cs.utexas.edu

Chenglong Wang
Microsoft Research
USA
chenwang@microsoft.com

Işıl Dillig
University of Texas at Austin
USA
isil@cs.utexas.edu

ABSTRACT

Due to the availability of increasingly large amounts of visual data, there is a growing need for tools that can help users find relevant images. While existing tools can perform image retrieval based on similarity or metadata, they fall short in scenarios that necessitate semantic reasoning about the content of the image. This paper explores a new *multi-modal image search approach* that allows users to conveniently specify and perform semantic image search tasks. With our tool, PhotoScout, the user interactively provides natural language descriptions, positive and negative examples, and object tags to specify their search tasks. Under the hood, PhotoScout is powered by a program synthesis engine that generates visual queries in a domain-specific language and executes the synthesized program to retrieve the desired images. In a study with 25 participants, we observed that PhotoScout allows users to perform image retrieval tasks more accurately and with less manual effort.

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

Interface design, multi-modal interfaces, program synthesis

ACM Reference Format:
Celeste Barnaby, Qiaochu Chen, Chenglong Wang, and Işıl Dillig. 2024. PhotoScout: Synthesis-Powered Multi-Modal Image Search. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3613904.3642319

1 INTRODUCTION

With the advancement of camera technologies and the prevalence of social media, photography is more accessible than ever. Nowadays, people increasingly have access to large volumes of photographs, taken by themselves or shared by others, that capture unique moments of their lives. As this volume grows, the task of retrieving relevant images from one's personal library becomes more important yet also more challenging. Modern photo management tools like Google Photos allow the user to search for relevant images based on metadata constraints (e.g., presence of a specific person; the date the photo was taken) or visual similarity to another image or natural language query.

While existing interfaces work reasonably well for simple search tasks, they fall short in *structured image retrieval tasks*: that is, tasks that require semantic reasoning about the structure of objects in an image. Such structured image retrieval tasks are important both for professional photographers as well as regular users who increasingly have access to large amounts of visual data on their smartphones and the cloud. For example, event photographers often have a shot list describing certain images that they must deliver to a client, such as those where the bride and groom are walking down the aisle or images containing only the bride and her mother [1, 50]. However, such structured image search tasks also come up in everyday life for regular users. For example, someone who is writing a travel blog might want to retrieve those images in which they are standing in front of the Eiffel Tower, or someone mourning the loss of a pet might want to find all images in which their cat is sitting on their lap.

As illustrated by these examples, such structured image search tasks require reasoning about contents of the image as well as relationships between them. However, such tasks are not easy to specify using existing image search interfaces. For example, while they provide support for finding images that contain a specific person, they do not facilitate searching for images where that person is performing a certain action or has a certain property. In fact, a key characteristic of structured image search tasks is that they require the contents of the image to satisfy certain *logical constraints* and combinations thereof.

In this paper, we propose a new user interaction model that facilitates structured image search. In general, these structured image retrieval tasks pose two challenges: First, how can a user effectively communicate their intent to the image search tool? Second, how can the search tool plan and execute the search logic underlying the user's intent?

- **User specification challenge:** For some image search tasks, it is difficult for users to convey their intent with a single modality. In particular, an example image alone is often too ambiguous to convey complex search logic. On the other hand, natural language (NL) alone also has shortcomings. For instance, even ostensibly simple relational attributes like "next to" or "on top of" can have

CHI '24, May 11–16, 2024, Honolulu, HI, USA

ACM ISBN 979-8-4007-0330-0/24/05
https://doi.org/10.1145/3613904.3642319

multiple possible interpretations that are difficult to disambiguate without a visual example.
- **System development challenge:** Existing text or object-based image search tools are powered by vision-language models [6, 37]. Despite their object understanding capabilities, they have limited capability in reasoning about complex search logic involving multiple constraints and semantic relationships between different objects. Hence, even if the user is able to perfectly convey their intent, there are no existing techniques that can be used to execute complex image retrieval queries.

We propose to address the above challenges of structured image retrieval through a novel program synthesis-powered *multi-modal image search tool*, PhotoScout. With PhotoScout, users can communicate their intent using a combination of natural language, positive and negative example images, and interactive object tagging. Through PhotoScout's multi-modal specification interface, the user can start with an efficient NL description of the task, then iteratively refine the search results by responding to queries posed through this interface. Under the hood, PhotoScout's backend synthesizer generates a programmatic query expressed in a domain-specific language (DSL) for image retrieval. If the user's NL description is ambiguous, the generated program will be incomplete, allowing PhotoScout to ask clarifying questions to the user in a goal-directed way. Once all ambiguities are resolved through user interaction and PhotoScout generates a complete program, the resulting query is executed on all uploaded images and search results are displayed to the user. At that point, the user can inspect the search results and further refine them if needed.

To assess the efficacy of PhotoScout compared to alternatives, we have conducted a user study involving 25 participants. We find that, compared with a baseline image search tool (leveraging a state-of-the-art vision model), users see a 34% increase in the F1 score of their search results when using PhotoScout. Further, in post-study interviews, users report that they are better able to convey their intent via PhotoScout's multi-modal specification interface and have more trust that PhotoScout's results are correct.

To summarize, this paper makes the following contributions:

(1) We present a new multi-modal image search interface targeted towards *structured* image retrieval tasks that allows users to effectively communicate their intent in an interactive fashion.
(2) We describe a neuro-symbolic image query language that allows expressing the types of logical queries that underlie structured image retrieval tasks.
(3) We present a program synthesis technique that leverages all the different modalities of input that users can provide through our proposed interface.

2 RELATED WORK

2.1 Image Retrieval

PhotoScout performs content-based image retrieval (CBIR), a technology pivotal in organizing digital image archives by visual content [42]. Datta et al. [18] characterize CBIR tools from two perspectives: the user's and the system's. The user perspective depends on input query modalities, while the system perspective hinges on query processing methods and presentation of search results [18]. From the user perspective, PhotoScout is an interactive, multi-modal CBIR system that allows users to find relevant images from a large personal collection. In particular, PhotoScout is *multi-modal* in that the user provides a combination of natural language and example images, and it is *interactive* in that the user can refine the query results by providing feedback through the PhotoScout interface. From the system perspective, many prior CBIR tools search for target images using metadata (e.g., where or when an image was taken) [2–4] or based on features extracted through machine learning techniques (e.g., lighting conditions and position of an object) [48]. In contrast, the backend underlying PhotoScout is based on neuro-symbolic program synthesis — that is, it leverages the user's examples and natural language query to synthesize a *logical search query* utilizing pre-trained neural networks for object detection and classification. In the remainder of this section, we focus on prior work that is closely related to PhotoScout and refer the interested reader to existing surveys [18, 19, 30, 42] for a more comprehensive overview of CBIR.

Expressing User Intent in CBIR. A key challenge in image retrieval is the *intention gap*: the difficulty users face in articulating their task through queries [42]. Prior work aims to address this concern through different modalities of input [13, 21, 28, 47, 53] and multiple rounds of user interaction [15, 29, 31, 55]. One line of work similar to PhotoScout is *composed image retrieval* [9, 32, 51], which utilizes visual and textual modalities to *jointly* specify the user's intent. In this line of work, an example image illustrates the concepts that the user is looking for, while the text query specifies what should be *different* (e.g. "same dress but blue instead of red"). In contrast to such interfaces, PhotoScout uses natural language to *directly* convey the user's intent rather than specifying what should be *different* from a given image. In particular, users of PhotoScout utilize positive and negative images to *clarify* ambiguities in the natural language query rather than providing them as a starting point for visual similarity search.

Relevance Feedback-Based Search Paradigms. Relevance feedback (RF) is a paradigm for interactively refining search results based on user feedback [57]. In many systems, users provide relevance feedback in the form of positive and negative images where positive examples correspond to those that are relevant to the user's query while negative examples are not [17, 27, 33, 35, 39, 45, 46]. PhotoScout is similar to these approaches in that the user can refine the initial query results by providing positive and negative examples. However, in contrast to many RF systems where examples are used to re-rank the search results (e.g. [27, 35, 39]), PhotoScout uses positive and negative examples to extract *hard semantic constraints* that the query results should or should not satisfy.

Semantic Concepts for Images. Another significant hurdle in image retrieval is the *semantic gap*, which refers to the challenge of describing high-level semantic concepts using low-level visual features [42]. Past research has explored deep learning techniques based on Convolutional Neural Networks (CNNs), including architectures like SqueezeNet [25], VGG [40], and ResNet [23], to address this problem. PhotoScout builds on recent advances in this field and leverages pre-trained neural networks for object detection and classification. Prior work targets a variety of applications, including

geolocation [8, 52], medical diagnosis [7, 16, 36, 56], and interior decorating [11]. However, in contrast to most neural CBIR systems, PhotoScout learns new semantic concepts by composing existing neural networks via symbolic operators.

2.2 Neuro-symbolic Programming for Images

As mentioned earlier, PhotoScout performs image retrieval by synthesizing neuro-symbolic queries that combine pre-trained neural networks with symbolic operators. Specifically, PhotoScout first synthesizes a query that is consistent with the user-provided input and then retrieves the desired images by executing the query on the user's dataset. Hence, PhotoScout is related to a line of recent work on neuro-symbolic programming for images [10, 20, 24, 26, 34, 38, 44, 54].

General Visual-Reasoning Tasks. Several recent works such as [22, 43] have proposed using neuro-symbolic programming to automate image-related reasoning tasks, such as visual question answering (VQA), image editing, and object tagging. In particular, VisProg [22] proposes a neuro-symbolic DSL targeting images and uses in-context learning to synthesize programs in this DSL based on natural language. ViperGPT [43] proposes a custom Python API for visual reasoning tasks and synthesizes Python programs using this API, also based natural language queries. The back-end of PhotoScout synthesizes neuro-symbolic programs; however, it uses a combination of natural language queries and positive and negative examples. In particular, PhotoScout generates a so-called *program sketch* by leveraging the natural language description and refines this sketch into a full query by utilizing the user-provided examples.

Specific Applications. While VisProg [22] and ViperGPT [43] propose general neuro-symbolic programming frameworks that can be adapted to several visual reasoning tasks, prior research has also developed more robust application-specific methods that use neuro-symbolic programming [10, 20, 24, 26, 34, 38, 44, 54]. Similar to our work, these efforts typically combine symbolic operators for higher-level reasoning with neural modules for perception, with the goal of learning new concepts in a few-shot manner. For example, Huang et al. [24] generate programmatic *referring expressions* that identify specific objects in an image in terms of their attributes and relationships to other objects. This work focuses on locating a single object, whereas our DSL expresses image search tasks that involve multiple objects. In addition, their focus is on a synthetic dataset with geometric shapes, while our focus is on more realistic images with faces, text, and arbitrary objects.

In the domain of image manipulation, ImageEye [10] allows users to automate batch image editing tasks using neuro-symbolic programming. In particular, ImageEye captures demonstrations of a user editing an image and then synthesizes neuro-symbolic programs that are consistent with the demonstration. In contrast to PhotoScout, ImageEye does not utilize natural language; instead, it requires the user to demonstrate the task by applying actions to selected parts of an image.

Another related work in this space is RAPID [49], which is a system for automated image labeling. The idea behind RAPID is to express new visual concepts (e.g., *chef*) as logical combinations of existing concepts and then learn these concept definitions from positive and negative examples. For instance, RAPID may learn that an image should be labeled "chef" if there is food or a bowl in the image. In contrast to PhotoScout, RAPID does not utilize natural language descriptions, and thus lacks the inductive bias for efficient image search. Additionally, RAPID uses a different learning approach based on first-order inductive logic learning.

3 USAGE SCENARIO

This section illustrates the interface and features of PhotoScout through a use case inspired by real-world scenarios described in online blogs [1]. In this example, a photographer, John, is preparing a wedding photo album and needs to locate specific images among hundreds of photos he took during the wedding. As part of this process, John needs to find photos in which the bride, Alice, and the groom, Bob, are next to each other and where Alice is holding flowers. For example, the first three images in Figure 1 meet John's requirements but the last one does not. John finds this task challenging to perform using existing similarity-based search tools, as there are a lot of other images containing Alice, Bob, and flowers, but many of these images do not match his *logical constraints* — for example, there is another person between Alice and Bob or Alice is not holding flowers. We now illustrate how John can use PhotoScout to perform this task and avoid significant manual labor.

Figure 2 shows the general interface of PhotoScout, which contains three main components: (1) a task specification panel (Figure 2–① to ④) that allows the user to communicate their intent using a combination of natural language queries and image labels, (2) a search result panel (Figure 2–⑤) that shows the results from the current search query, and (3) a saved images panel (Figure 2–⑥) for saving and exporting the desired images. Using this interface, John can complete his task by performing the following steps:

(1) *Load images.* John first loads all the images to PhotoScout and then sees the interface shown in Figure 2.
(2) *Write natural language query.* The user interface exposes a search box where the user can type a natural language query (Figure 2–①) and a panel displaying thumbnails for all uploaded photos (Figure 2–③). In a typical use case, the user starts by entering a natural language query, such as "Alice next to Bob holding flowers", and clicks the "Search" button.
(3) *Tag objects.* In this example, PhotoScout does not yet know who Alice and Bob are, so, in the search results panel (Figure 2–⑤), PhotoScout displays a message communicating this missing information. John resolves this ambiguity by selecting an image and tagging Alice and Bob's face in the labeling panel (Figure 2–④). Figure 3 provides a more detailed view of the labeling panel. When John selects a photo, PhotoScout shows the full-size photo in the center of the labeling panel. The photo is annotated with object detection and classification results to help the user understand what the underlying computer vision tools "see" in that image. For example, when the user hovers over a part of the photo, PhotoScout displays detected objects as a square box, as shown

Figure 1: Left: Three images that matches John's intent: the bride and groom are next to each other, with the bride holding flowers. Right: an image that is incorrect image because the bride is *not* holding flowers.

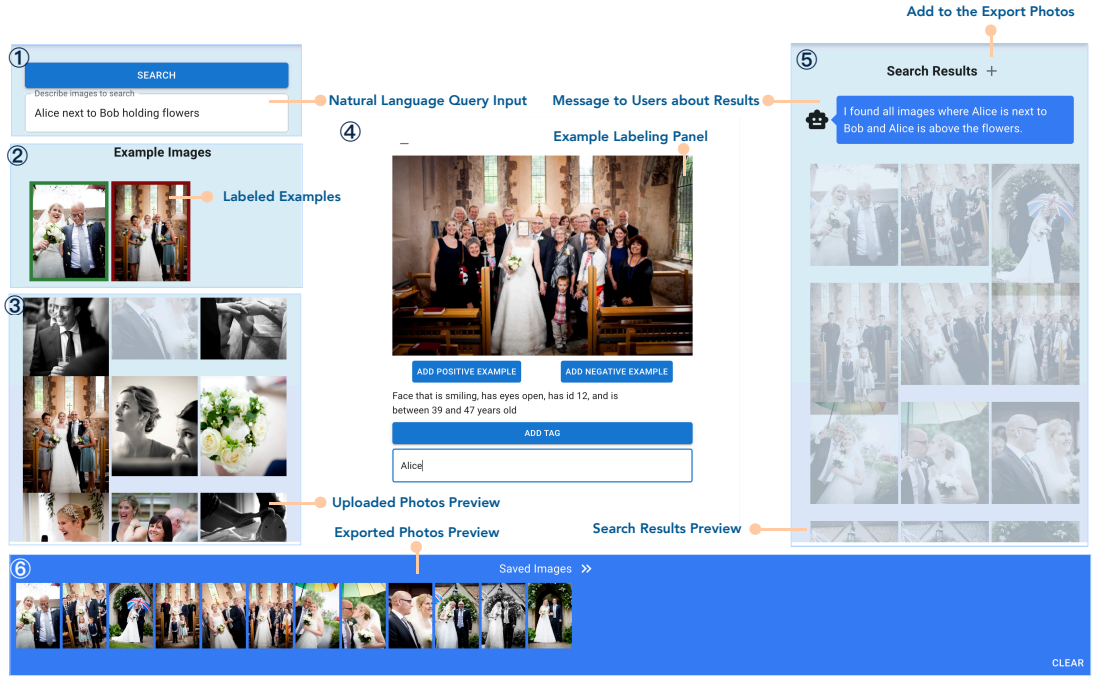


Figure 2: The PhotoScout interface has six main panels: (1) The user enters a natural language query describing the images to be searched. (2) The example images panel highlights all the positive and negative images that the user has already labeled. Positive examples are wrapped in a green box and negative examples are wrapped in a red box. (3) The album preview panel displays all the photos in the album to be searched from. (4) Once the user selects a photo to label, the example labeling panel displays the image and the example labeling buttons. (5) The search results panel shows all the images that PhotoScout finds that match both the natural language description and the labeled examples, along with a natural language explanation. (6) The photo export panel shows all the images selected by the user as the final search results.

in Figure 3. Additionally, the interface displays a natural language description of the classification results for that object. For example, Alice's face in Figure 3 is further categorized as smiling and between 31 to 41 years old. In this scenario, John clicks on the face of the bride and labels the face as Alice (see 3d Figure 3). At this point, PhotoScout learns to associate this face with Alice, ensuring that she can be referenced in future search queries without additional user interaction. John uses the same panel to similarly detect the groom's face and label it as Bob.

(4) *Select positive examples.* After tagging these faces, John clicks "Search" to see the updated results. This time, PhotoScout is not sure about the concept of "holding flowers" and asks John to illustrate this concept by providing examples. John labels the first image in Figure 1 as a positive example using the labeling panel (Figure 2–④) and clicks "Search" again.

(5) *Select negative examples.* This time, instead of asking for clarification, PhotoScout shows all relevant images in the result panel (Figure 2–⑤), along with an natural language explanation of how it generated these results. After looking at the explanation and inspecting the results, John notices

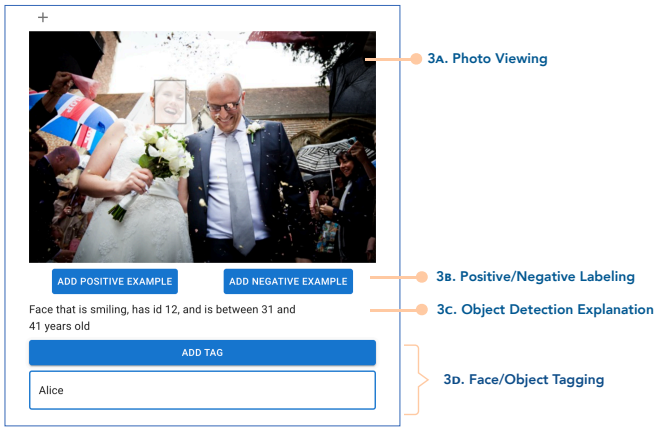


Figure 3: The example labeling panel consists of 4 elements. (a) A view of the photo to be labeled. When a user hovers over it, each object identified by the detector is highlighted with a square box, with the detailed description of the detected object shown in (c). (b) asks the user to label the photo either as a positive or negative example. (d) is a tagging interface so the user can give semantic meanings to the detected face or object. In this particular example, the user is tagging the bride with the name "Alice" so that they can refer to the bride in the query.

that the results contain all relevant photos but also some extra ones, specifically those where there are flowers, but Alice is not holding them (e.g., the last photo in Figure 1, where Bob's boutonnière is visible). To further refine the search results, John labels this photo as a negative example and does another round of search. This time, PhotoScout returns all photos of Alice next to Bob with Alice holding flowers. As a final step, John clicks on the "+" sign located at the top of the search results section (Figure 2–⑤) and all the added photos are displayed in Figure 2–⑥.

(6) *Manually add/remove images.* Upon inspection, John finds that there is one photo in the results in which Alice's flowers are sitting in front of her on a table, but she is not holding them. To exclude that photo from the search results, John selects the photo from Figure 2–⑥ and clicks the "-" button on the top left of Figure 2–④ to remove this image from the export results. Once John is happy about the results, he clicks the "»" button in Figure 2–⑥ to export the results to a user-defined directory.

In summary, John is able to find all the photos he wants to retrieve by first providing a natural language query and then iteratively refining this query by tagging objects and labeling photos as positive or negative examples. In this process, he benefits from the following design decisions behind PhotoScout:

- **Multimodal Inputs.** PhotoScout grants John the versatility to articulate his search criteria both using natural language prompts and positive and negative examples. On one hand, solely relying on natural language introduces several potential ambiguities: For example, who are Alice and Bob, and who should be holding the flowers? On the other hand, solely relying on examples would be quite cumbersome, as John would need to provide several more examples to convey his intent. In contrast, the *combination* of natural language and image annotations allows John to succinctly and efficiently convey his intent.
- **Semantic search.** In our example, John's search query is quite specific: First, Alice and Bob must be next to each other, and, second, Alice should be holding flowers. Such search queries are out of scope for existing image retrieval systems, as they cannot reason about *relationships* between objects within an image. In contrast, our approach performs search by first synthesizing a *neuro-symbolic program* and then executing that program on all images. This synthesis-based approach allows PhotoScout to perform structured image search tasks where the goal is to find images that conform to non-trivial logical constraints.
- **Feedback-guided refinement.** Rather than presuming John to deliver flawless instructions from the outset, PhotoScout employs an interactive feedback mechanism. As illustrated in our example, PhotoScout recognizes ambiguous elements in John's description and proactively seeks clarification via natural language prompts.
- **Fast synthesis procedure.** To ensure that John does not have to wait a long time when interacting with the tool, PhotoScout adopts an efficient synthesis approach to find useful programs. Each synthesis run takes between 0.36 and 4.8 seconds, making it feasible to use PhotoScout in an interactive fashion.

4 SYSTEM ARCHITECTURE AND IMPLEMENTATION

In this section, we discuss the design and implementation of PhotoScout. As mentioned earlier, PhotoScout performs image search by first synthesizing a program in a neuro-symbolic domain-specific language (DSL) and then applying that program to all images in a collection. In this section, we first provide an overview of the image search DSL and then explain the internal workflow of PhotoScout in more detail.

4.1 Image Search DSL

PhotoScout's DSL, shown in Figure 5, is designed to express a wide array of image search tasks. At a high level, a program in this DSL is similar to a first-order logic formula, and evaluates to either true or false given an input image.

The primitives in this DSL are predicates of the form $r(t_1, \ldots, t_n)$ where r is an n-ary relation and each t_i is a term (constant or variable). The PhotoScout DSL contains many built-in predicates such as the binary relations HasEmotion(t, c) and HasType(t, c), as well as ternary relations such as HasRelation(t_1, t_2, c). Note that the semantics of the predicates are determined using neural models; hence, we refer to this DSL as *neuro-symbolic*. For example, HasType(o, Car) is determined by using an object classification model to check whether o is classified as a car. Similarly, the truth

Figure 4: An example image.

$$
\begin{array}{rcl}
E & := & r(t_1, \ldots, t_n) \\
 & | & E \rightarrow E \mid E \wedge E \mid E \vee E \mid \neg E \mid \exists x.E \mid \forall x.E \\
r & := & \mathsf{HasType} \mid \mathsf{HasEmotion} \\
 & | & \mathsf{HasRelation} \mid \mathsf{HasProperty} \\
t & := & x \mid c
\end{array}
$$

Figure 5: Image Search DSL. All predicates are binary except for HasRelation (ternary).

value of $\mathsf{HasRelation}(o_1, o_2, \mathsf{Above})$ can be determined by using an object classification model that identifies bounding boxes around objects o_1, o_2 and then using the resulting coordinates to check whether o_1 is above o_2. As standard in first-order logic, predicates can be combined using boolean connectives. Additionally, our DSL allows quantification over variables to test whether an image contains *some* object with a given property (requiring existential quantification) or whether *all* objects have a certain property (requiring universal quantification).

In our implementation, the truth value of atomic predicates is determined using the Amazon Rekognition library [5]. The pretrained neural nets supported by this library can detect and locate a wide array of objects in image. In particular, this library can be used to identify bounding boxes for different objects in the image, and to determine their types (e.g., cat, car, person etc). Rekognition can also detect properties of human faces (e.g. whether a face is smiling or has open eyes) and identify the same face across multiple images. Overall, it is this *combination* of logical operators and neural models that allows PhotoScout to express a rich class of structured image search tasks.

Example 4.1. Consider the following program:

$$\forall x.(\mathsf{HasType}(x, \mathsf{Face}) \rightarrow (\mathsf{HasProperty}(x, \mathsf{Smiling}) \wedge \exists y.$$
$$(\mathsf{HasType}(y, \mathsf{Flower}) \wedge \mathsf{HasRelation}(x, y, \mathsf{Above}))))$$

In this program, the universal quantifier $\forall x$ indicates that every object x identified in the image must obey the subsequent condition. In particular, if x is identified to be a human face, then x must be smiling, *and* there must exist an object y in the image such that y is identified to be a flower, where x is above y. Put simply, this program can be used to find images where every person is smiling and holding flowers, as in Figure 4. Note that the concept of "x holds y" is approximated by checking a spatial relationship between x and y.

4.2 PhotoScout Synthesizer

In this section, we describe PhotoScout's underlying synthesis engine, which is depicted schematically in Figure 6. Given the initial natural language query, PhotoScout starts by generating a *program sketch* containing holes (i.e., unknowns denoted as $\square$). Intuitively, PhotoScout cannot directly generate a program from the natural language query because some of the concepts used in the NL description have to be grounded. For example, given a query like "Alice is holding flowers," the synthesizer has no idea what Alice corresponds to or how the concept of "holding" can be implemented in our DSL. To instantiate the program sketch into a complete program, PhotoScout interacts with the user by asking them to tag objects or provide examples (Step 2 in Figure 6). In the third step, the synthesizer fills the holes in the sketch by performing enumerative search over the space of sketch completions and discarding those programs that do not satisfy the examples. In the final step, the synthesized program is applied to all uploaded images and displayed to the user. If the search results are unsatisfactory, the user can refine the query by providing more positive and negative examples. We now explain each of the steps in this process in more detail.

Step 1: Generate program sketches. Motivated by the success of few-shot prompting in similar domains [12, 14, 58], PhotoScout obtains program sketches by prompting GPT-3.5 Turbo.[1] The key idea is to provide GPT with examples of representative natural language and program pairs and then ask it to generate a program for the user's NL query. Figure 7 shows an example of such a prompt where we provide the LLM with a manually curated set of representative (query, program) pairs as well as the natural language query of interest.. PhotoScout asks GPT to generate 20 answers to this prompt in order to increase the likelihood that *one* of the results match the user's intention. For each result returned by GPT, PhotoScout attempts to parse the string into a program in our DSL. If, during parsing, PhotoScout encounters a predicate or constant that it does not recognize, it replaces that construct with a hole $\square$. If parsing fails for a different reason, that program sketch is discarded.

Example 4.2. Given the text query "`Alice is holding flowers`", GPT may generate the program

$$\exists x.\exists y.\ (\mathsf{HasType}(x, \mathsf{Alice}) \wedge \mathsf{HasType}(y, \mathsf{Flowers}) \wedge \mathsf{HasRelation}(x, y, \mathsf{Holding})) \quad (1)$$

However, since Alice is not an object category recognized by the object detector and Holding is not a predicate in the DSL's grammar, these constructs will be replaced with holes. Thus, the following program sketch will be produced:

$$\exists x.\exists y.\ (\mathsf{HasType}(x, \square_1) \wedge \mathsf{HasType}(y, \mathsf{Flowers}) \wedge \mathsf{HasRelation}(x, y, \square_2)) \quad (2)$$

Step 2: Query the user. If the program generated in Step 1 contains holes, PhotoScout prompts the user to provide additional information by (1) tagging objects that are not recognized by the object detector and (2) adding example images that clarify the meaning of unknown predicates. Tags and examples images both allow the user to clarify the meaning of their natural language query, but in complementary ways. Tags allow grounding unknown terms like people's names in the user's NL query, whereas positive and

[1]While a different LLM could be used for the purpose of sketch generation, we use GPT in our implementation because we found it to more effective than alternative models that we tried.

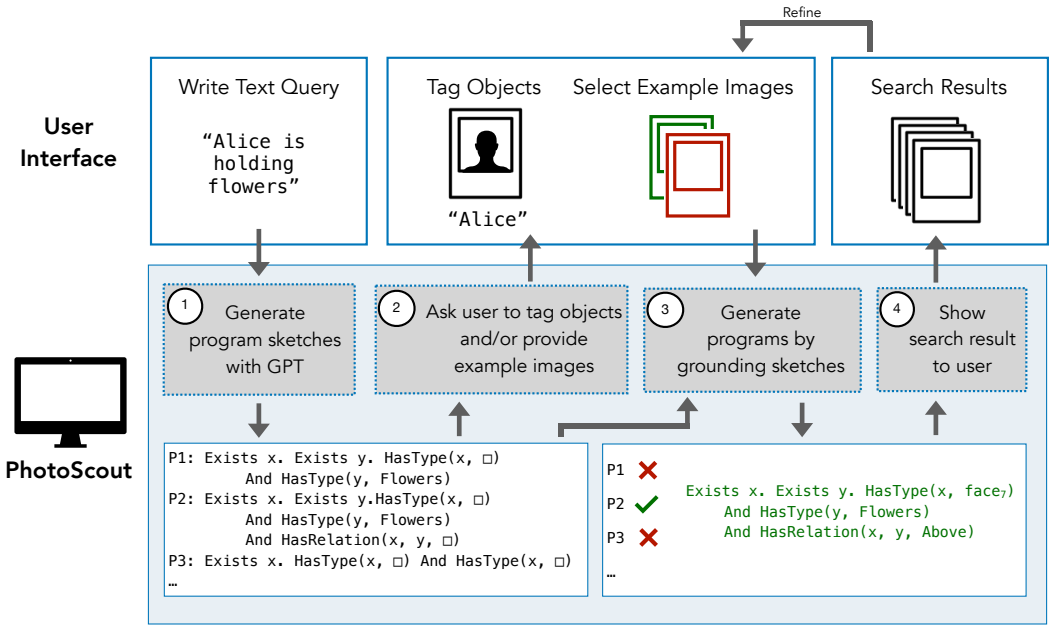


Figure 6: The architecture of the PhotoScout system.

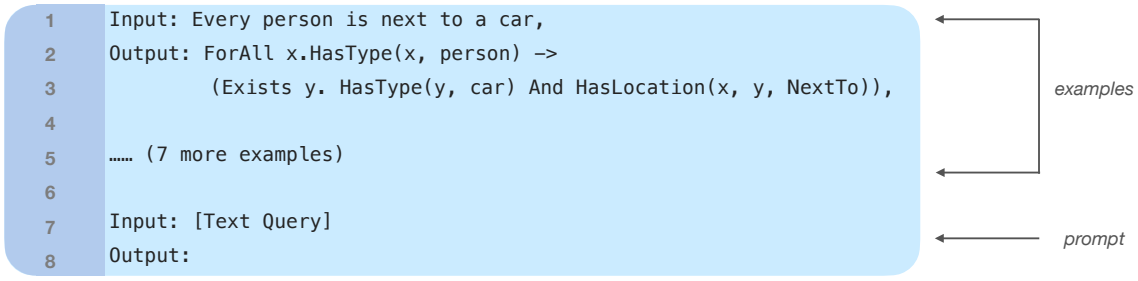


Figure 7: GPT prompt for generating program sketches from a user's text query.

negative examples allow the synthesizer to learn logical constraints and concepts (e.g., the concept of "holding") in a data efficient way.

Example 4.3. Given the program sketch

$$\exists x.\exists y.(\text{HasType}(x, \square_1) \wedge \text{HasType}(y, \text{Flowers}) \wedge \text{HasRelation}(x, y, \square_2)) \quad (3)$$

where $\square_1$ and $\square_2$ were derived from Alice and Holding, respectively, PhotoScout will display the following message to the user: "I don't know the terms 'Alice' and 'Holding'. Can you provide a few positive and negative examples and/or tags to show me what you mean?" The user can easily ground the name "Alice" by using the PhotoScout interface to add a tag. However, the concept of "holding" is harder to explain through a tagging, as it corresponds to a binary relationship between two objects. In this case, the user can help PhotoScout learn this concept by providing a few positive examples where Alice is holding the flower and a few negative examples of those where there is a flower in the picture but Alice is *not* holding them.

Step 3: Sketch completion. While object tagging helps resolve many sources of ambiguity in the natural language, PhotoScout needs to perform enumerative search over possible sketch completions to find a program that is consistent with all positive and negative examples. Given a program sketch P and a a set of positive and negative examples $\mathcal{E}^+ \cup \mathcal{E}^-$, PhotoScout enumerates possible completions of P by instantiating each hole with a constant and then evaluating the resulting query Q on each example in $\mathcal{E}^+$ and $\mathcal{E}^-$. If Q evaluates to true (resp. false) for all examples in $\mathcal{E}^+$ (resp. $\mathcal{E}^-$), then Q is retained as a viable completion of the sketch. Among all programs that are consistent with the examples, PhotoScout chooses the simplest program, where simplicity is defined in terms of the number of nodes in the program's abstract syntax tree. Note that enumerative search is tractable in this context because each sketch contains no more than a few holes.

Example 4.4. Consider the following partial program:

$$\exists x.\exists y.\ (\text{HasType}(x, \text{face}_n) \wedge \text{HasType}(y, \text{Flowers}) \wedge \text{HasRelation}(x, y, \square)) \quad (4)$$

Suppose that the user has added the first three images in Figure 1 as positive examples and the last one as negative. Consider the completion P' of this partial program where $\square$ has been filled with NextTo. For each positive example image $\mathcal{I}^+$, $P'(\mathcal{I}^+)$ is true, as Alice is adjacent to flowers in each of these images. However, for the negative example image $\mathcal{I}^+$, $P'(\mathcal{I}^-)$ is also true. Thus, P' is not a valid completion of the program. However, the completion P'' where $\square$ has been filled with Above is a valid completion, as Alice's face is below flowers in every positive example, but not in the negative example.

Note that this step is useful even if none of the program sketches contain holes, as example images will filter out complete programs that do not match the user's intent.

Step 4: Display search results. The last step in the process is to execute the synthesized program P on all input images and display those images I for which $P(I)$ yields true. Since the number of search results may be quite large, PhotoScout also generates a natural language explanation of what the program does. Such explanations are intended to help users quickly uncover unintended behaviors without having to look through a large set of images and inspecting each one. PhotoScout generates these NL descriptions through few-shot prompting of an LLM: In particular, given a few examples of programs and their corresponding NL description, PhotoScout prompts GPT to produce an natural language description of the programmatic query.

Example 4.5. Suppose that P is the program

$$\exists x.\exists y.\text{HasType}(x, \text{Alice}) \wedge \text{HasType}(y, \text{Flowers}) \wedge \text{HasRelation}(x, y, \text{Above}). \tag{5}$$

Then GPT may generate the following natural language explanation: "`I have found all images that contain Alice and flowers and where Alice's face is directly above the flowers.`"

Even after PhotoScout generates a correct program, it may not produce *exactly* the desired output for two main reasons: First, some concepts such as "holding" may not be perfectly expressible in our DSL. For instance, in our running example, we approximate the concept of holding through a coarse spatial relationship between objects (e.g., if face x is directly above object y, then x is holding y). Second, even when all concepts are perfectly expressible, the program may not produce the desired output due to inaccuracies in the underlying neural model. For instance, if the face recognition model does not correctly classify Alice's face, then a photo containing Alice may not appear in the search results even though it should. PhotoScout deals with this problem by allowing users to manually add or remove images through the Saved Images panel of the user interface.

4.3 Design Considerations

We conclude this section by summarizing and justifying some of the design considerations underlying PhotoScout.

User Interface. The design principles of PhotoScout's user interface reflect the requirements of structured image search tasks. As seen in the usage scenario, a structured search task may be simple and intuitive to describe in natural language, but contain ambiguities that are easier to resolve through visual examples. Hence, our interface allows the user to interactively refine the search results. In a typical workflow, the user begins their search by writing a natural language query, which may contain unknown terms and concepts that need to be *grounded*. To help the user understand which terms need to be grounded via user interaction, PhotoScout generates natural language explanations of what it does *not* understand. The user then can then interact with PhotoScout to teach it new concepts. In particular, constants such as people's names are natural to teach via object tagging, whereas new predicates (e.g., "holding") can be demonstrated using positive and negative examples. Furthermore, the user can provide these examples in a piecemeal fashion by providing one example at a time, re-running the synthesizer, and inspecting the search results.

System Implementation. Recall that PhotoScout's system represents search tasks as programs in a DSL. Utilizing DSLs for visual tasks is an approach established in prior work [10, 22, 41, 43]. In the context of structured image retrieval, we believe that such a DSL-based approach is a particularly good fit, as the user wishes to find images that satisfy certain logical constraints.

We note that any DSL imposes a tradeoff between *expressiveness* and *reliability*: the more expressive the DSL, the larger the space of tasks it can represent, leading to a harder *synthesis* problem. On the other hand, if the DSL is too restrictive, it may not be able to express image search queries that arise in practice. Our DSL maintains a balance between these two properties, capturing a wide array of structured image search tasks while keeping a compact structure similar to first-order logic that facilitates synthesis.

PhotoScout generates image search queries by using an LLM to "translate" the user's NL description to a program sketch in the DSL. This approach allows the synthesizer to extract as much information as possible from a coarse search query expressed in natural language. However, because the user's description may be ambiguous or contain new concepts that are not captured via pre-trained neural models, PhotoScout grounds new concepts via user interaction, which takes two forms: Object tagging allows the user to conveniently ground names, whereas positive and negative examples allow grounding unknown relations and resolving ambiguities.

5 USER STUDY

To understand how people interact with PhotoScout and gain deeper insight about the strengths and limitations of the proposed interface, we conducted a within-subjects evaluation centered on the following questions:

- RQ1. Does PhotoScout improve user efficiency and accuracy compared to a baseline image search tool?
- RQ2. Do users find PhotoScout easier to use than a baseline tool?
- RQ3. Are users more confident about the accuracy of the search results compared to the baseline tool?
- RQ4. Does the proposed multi-modal interface help users express their intent?
- RQ5. What strategies do people adopt when interacting with PhotoScout?

In the remainder of this section, we first describe the baseline tool and our user study procedure. Afterwards, we present both quantitative and qualitative analyses of the user study results.

5.1 Baseline Tool: ClipWrapper

As a basis of comparison, we implemented a graphical user interface around OpenAI's CLIP model, which is a state-of-the-art neural network for learning visual concepts from natural language supervision. Given a dataset $\mathcal{D}$ of images and a query (in the form of text or image), the CLIP model assigns a score to each image in $\mathcal{D}$ that reflects its similarity to the given query.

Our baseline tool, henceforth called ClipWrapper, is a wrapper around this CLIP model. Specifically, ClipWrapper implements a graphical user interface that allows users to input a query on a set of uploaded images. The query can either be a text description of the search task or a photograph that exemplifies the target search results. ClipWrapper simply queries the CLIP model and returns all images in the dataset whose score exceeds some threshold. The ClipWrapper interface allows users to further refine the search results by manually adding or removing images to and from the result returned by the CLIP model.

ClipWrapper allows users to search for images that are similar to either a query image or an open-ended text query. ClipWrapper does not utilize any hard constraints and may return images that do not precisely match a user's query. A central question of this user study is: does ClipWrapper suffice for performing structured image retrieval tasks, or is a tool specially designed for such tasks necessary? Further, we explore the specific features of ClipWrapper that make structured image search difficult, compared with PhotoScout. ClipWrapper's interface mirrors PhotoScout as closely as possible so as to reduce the number of confounding factors in our comparison.

5.2 User Study Procedure

We recruited a total of 25 participants for our user study. Among these participants, 23 are in the 18-24 age bracket, and the remaining two are between 25 and 35 years old. In terms of gender, 14 (resp. 9) of the participants self identify as female (resp. male) and 2 self-identify as "other". Our only criteria for selecting participants was that they have prior experience using a computer and that they do not have impaired vision. The entire user study took place over the course of three weeks.

During the user study, each participant was asked to first complete a training session and then perform four image search tasks, two using PhotoScout and two using ClipWrapper. The order of tasks, as well as which tool to use for a given task, was randomly selected. The training session involved completing a tutorial about both tools and performing two practice tasks, one with PhotoScout and one with ClipWrapper. The users had access to the tutorial throughout the user study and were explicitly told that they could reference it whenever they wished to do so. The participants were given a total of 5 minutes to complete the practice tasks and each of the four image search tasks. Participants were told that they could end a task whenever they were satisfied with the results; however, participants opted to use all the time available to them in most cases.

In the course of the study, participants were asked to talk about their search strategies while completing each task. To aid subsequent analysis, we collected both audio and screen recordings throughout the process. Upon completion of the four tasks, the participants were asked to reflect on their experience and answer some interview questions. The total session, including the tutorial and interview, took less than 90 minutes for each participant.

5.3 Tasks

Given a dataset of images, the goal of each task in the user study was to identify a subset of the images matching a certain criteria. Specifically, the tasks involved the following three sets of images:

- *Transportation*: A set of 70 images of bicycles, cars, and people, mostly taken on public roads.
- *Festival*: A set of 420 images from a music festival, comprised of images of performances, venues, and attendees.
- *Wedding*: A set of 352 images from a wedding, including staged photos of the wedding party and candid photos of the ceremony and reception.

Each task targeted one of these datasets. Since PhotoScout is intended for use on personal images, we collected these datasets from image galleries shared on Flickr. As such, the datasets vary in size. Participants were provided with task descriptions, along with a description of the corresponding dataset. The task descriptions were as follows:

(0) Find all images that contain a car and a bicycle.
(1) Find all images that contain a guitar and a microphone.
(2) Find all images that contain no people. An image contains a person if you can see any discernible part of someone's body.
(3) Find all images where the bride is to the left of the groom. An image contains a person if their face is visible.
(4) Find all images that contain the bride but not the groom. An image contains a person if their face is visible.

Task 0 was the practice task and involved the transportation dataset. Tasks 1 and 2 used the festival dataset, and the last two tasks involved the wedding dataset. Note that tasks 3 and 4 involve searching for *particular* faces in an image. For these tasks, the participant was given example images with the bride and groom's faces.

6 USER STUDY RESULTS

6.1 Quantitative Results

Search Result Accuracy. One of the key metrics for evaluating the efficacy of each tool is *accuracy of search results*. That is, within the 5 minute time limit, how close were the saved results to the ground truth? To answer this question, Table 1 reports the F1 score of the search results when participants use PhotoScout and ClipWrapper. We report two different accuracy results, namely *before* and *after post-processing*. To understand what we mean by this, recall that people first interact with the underlying tool (ML model or synthesizer) to get an initial set of results, and then manually add/remove images to refine the search results before finally saving them. The columns labeled *before post-processing* show the F1 score for the search result automatically generated by the tool before

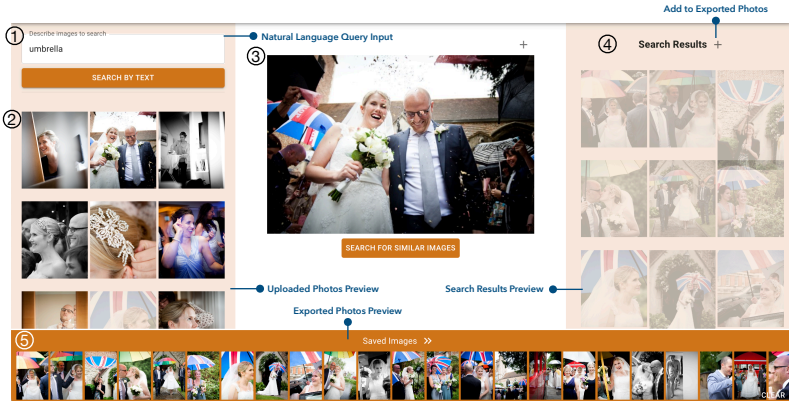


Figure 8: The ClipWrapper interface has five main panels: (1) The user enters a natural language query describing the images to be searched. (2) The album preview panel displays all photos in the target album. (3) The photo view panel displays an image and allows users to search for similar images. (4) The search results panel shows all images that ClipWrapper finds that match the user's query. (5) The photo export panel shows all images selected by the user as the final search result.

manual intervention.[2] As we can see, the initial search results for PhotoScout are significantly better (0.45 vs 0.76 in terms of average F1 score). Furthermore, using the Wilcoxon rank sum test, we find that these results are statically significant, with a p-value of less than 0.02, for all tasks.

The columns labeled *after post-processing* show the results after the users have manually refined the search results within the 5 minute time limit. Overall, the F1 scores for PhotoScout are higher compared to those of ClipWrapper, and overall difference in F1 score is statistically significant (p-value of $< 3.1e{-}7$ for the Wilcoxon rank sum test). However, if we run the same test for each individual benchmark, we find that the result is statistically significant for only the Guitar and Microphone task and the No People task. For the Bride and Not Groom task, there was one participant who mistook a wedding guest for the bride and completed the task by searching for images containing that guest. When this outlier is removed from the dataset, the result for the Bride and Not Groom task is significant as well. A discussion of why the Bride Left of Groom task does not have a significant result is included in Section 6.3.

Search Efficiency. The average time per search query (i.e. the time the system takes to perform a search for a given query) for PhotoScout and ClipWrapper is presented in Table 1. For ClipWrapper, search time is consistent across all tasks and queries. For PhotoScout, search time varies depending on what inputs the user provides. For instance, if the user provides an example image with a lot of different objects, then sketch completion will take longer, as there are more ways that the sketch could be filled in. While PhotoScout, on average, takes longer than ClipWrapper, both tools are efficient enough for interactive online use.

Manual effort. Another important metric for evaluating the efficacy of a tool is the amount of manual effort. That is, how many objects did the user tag, and how many images did the user have to manually add or remove before they were satisfied with the results or reached the 5 minute time limit? The use of tagging was extremely consistent. Participants only used tags when completing the two tasks using the wedding dataset, as these tasks involved reasoning about specific faces whose names were not known by the object detector. Participants completed these tasks with PhotoScout 24 times. In 23 of these instances, the participant used tagging to assign names to the bride and groom (i.e. the subjects of the task). For instance, P14 tagged the bride and groom as "Emily" and "John," respectively, and wrote the query "*Emily is to the left of John.*"

The results for other metrics of manual effort are presented in Table 1. In particular, for PhotoScout, we report two different numbers: (a) the total number of examples provided when using the synthesizer, and (b) the number of added/removed images to refine search results. We can take the sum of (a) and (b) to be the proxy for manual effort. The difference in manual effort between PhotoScout and ClipWrapper is statistically significant for all tasks, with a p-value of less than 0.02. Note that, for all tasks, and the Bride Left of Groom task in particular, ClipWrapper users manually added and removed a significant number of images in proportion to the size of the ground truth dataset. Participants using ClipWrapper often resorted to extra manual efforts to add and remove images from their initial search results to achieve a higher accuracy; however, this required a greater cognitive load to complete their task: e.g., P14 mentioned *"it felt like I had to basically look through every image."*

> **Result for RQ1:** On average, across all tasks, participants achieved higher accuracy with PhotoScout than with ClipWrapper while expending less manual effort.

Task Questionnaire. For the last part of our quantitative study, we analyze the results of the questionnaire that each participant was

[2]For PhotoScout, this refers to the result after the user is done running the synthesizer.

asked to complete *after* finishing a task. Specifically, participants were asked the following questions upon completion of each task:

(1) On a scale of 1-5, with 5 being very easy and 1 being not easy at all, how easy was it to complete the task using this tool?
(2) On a scale of 1-5, with 5 being very confident and 1 being not at all confident, how confident are you that your results are correct? "Correct" means that all of your saved images match the task, and none of the unsaved images match the task.

Figure 9 summarizes the results of this questionnaire. Across all tasks, participants gave an average score of 4.0 for PhotoScout and 2.7 for ClipWrapper on question 1, and an average score of 3.8 for PhotoScout and 2.6 for ClipWrapper on question 2. For question 1, the difference in scores between PhotoScout and ClipWrapper was statistically significant for all tasks, with a p-values less than 0.03. For question 2, the difference in scores was significant for the Guitar and Microphone task and the No People task.

We also asked participants for qualitative input on each question. When answering question 1 (ease of use), some participants noted that PhotoScout *"has a steeper learning curve"* than ClipWrapper due to its more sophisticated search features, but that *"once you have done the setup, the results it gives are pretty accurate"* (P1). Further, when answering question 2 (confidence in results), some participants noted that they had confidence in their results with ClipWrapper *because* of the manual effort they had expended going through the images themselves. Despite these aspects working in ClipWrapper's favor, the scores for each question are consistently higher for PhotoScout than for ClipWrapper.

Result for RQ2: Across all tasks, participants gave PhotoScout an average rating of 4.0 out of 5 for ease of use, compared with an average rating of 2.7 out of 5 for ClipWrapper. The difference in scores was statistically significant for all tasks.

Result for RQ3: Across all tasks, participants rated their confidence in PhotoScout's as 3.8 out of 5, compared with an average rating of 2.6 out of 5 for ClipWrapper. The difference in scores was statistically significant for two of the four tasks.

6.2 Qualitative Results

We conducted a semi-structured interview about the participants' experiences using PhotoScout and ClipWrapper. We asked participants about their search strategies and results using both tools. In addition, we instructed participants to think aloud while completing each task, kept notes on comments that participants made. One of the authors coded participants' responses to each question and comments on each task, and two authors reviewed and discussed the results collaboratively. This analysis addresses RQ4 and RQ5. We report the following key findings:

KF1: PhotoScout*'s synthesis-based search procedure makes structured image search easier and more efficient.* 16 of the 25 participants reported that they thought their results using PhotoScout were more accurate than their results using ClipWrapper. Out of the other 9 participants, only one said that their results with ClipWrapper were more accurate; the other 8 were unsure. Participants expressed confidence in their results with PhotoScout: *"[PhotoScout] was actually really good at getting what I asked. I think [PhotoScout] was pretty trustworthy overall"* (P21). Similarly, they expressed a lack of trust in ClipWrapper: *"I don't know, I just didn't have that much faith in [ClipWrapper]"* (P5).

In particular, participants observed that PhotoScout was better than ClipWrapper at finding images that were consistent with logical and positional elements of their queries. When completing the No People task with ClipWrapper, P9 noted, *"I put no people in the search bar, and it gave me a bunch of images with people."* Many participants who used ClipWrapper for this task developed a strategy of finding one image without people, and searching for similar images. This strategy allowed them to find certain types of images without people (e.g. closeup images of signage at the festival), but caused them to miss other types of images that were not visually similar (e.g. photos of venues before performances had taken place). By contrast, participants who used PhotoScout could efficiently write a text query, add a few example images, and see a set of accurate search results matching the logical intention of their query.

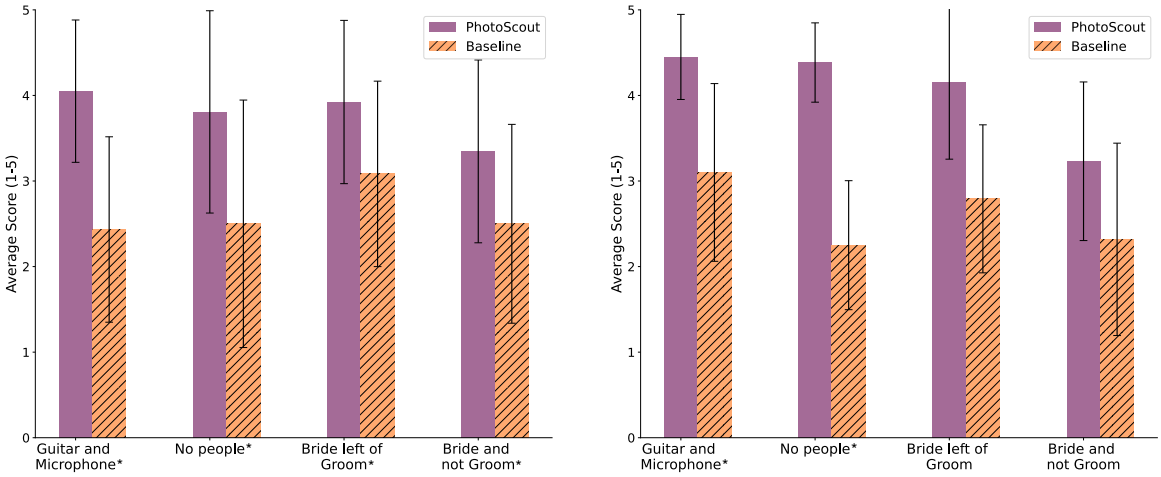
Similarly, when completing the Bride Left of Groom task, P18 said *"[ClipWrapper] doesn't seem to know its lefts and rights that well."* Participants using ClipWrapper were able to find images containing the bride and groom without much difficulty, but finding images where the bride and groom were oriented correctly could only be accomplished through manually filtering. Meanwhile, participants using PhotoScout could use a text query and example images to specify that they only wanted photos were the bride is to the left of the groom, and saw results that reflected this intent. P12 stated, *"I noticed that [PhotoScout] is more functional when it comes to relational statements"*.

Interestingly, most participants did *not* make use of the natural language explanations of the search results in PhotoScout. Only 2 out of 25 participants reported that they found the explanations useful, and many participants did not notice the explanations, even though the tutorial pointed out this feature. While we cannot determine exactly why participants did not make use of NL explanations, we can conclude that this feature had little to do with participants' confidence in PhotoScout's search results. Future work could explore alternative methods of explaining search results to the user. One such method allows users to visualize why a particular image appeared, or did not appear, in the search results. This visualization could include annotations and/or text that highlight the parts of an image that match or do not match the query. Several participants noted the potential utility of this feature when reviewing their search results.

KF2: Example images convey information that text alone cannot. 22 out of 25 participants indicated that example images provided additional information that they could not convey in text. P7 said *"[Examples] can describe what you're looking for better than text.... A picture is worth a thousand words."* P15 noted, *"I like that I was able to provide example images, because it helped me clarify [my intent]."* Participants noticed that example images and text queries worked

Table 1: A quantitative comparison of PhotoScout (abbr. P) and ClipWrapper (abbr. C). # Ground Truth lists the number of images in the ground truth dataset of each task. # Assigned lists the number of participants who were assigned with each tool for each task. Avg. F1 Score Before and Avg. F1 Score After list, respectively, the average F1 score of the search results before and after performing post-processing (i.e. manually adding and removing images from the search results) with each task and tool. Manual Effort lists the average number of images post-processed (i.e. added and removed from search results) for each task, and, in the case of PhotoScout, the number of images selected as examples.

Task Description	# Ground Truth	# Assigned		Avg. F1 Score Before		Avg. F1 Score After		Avg. Time Per Query (s)		Manual Effort - PhotoScout		Manual Effort - ClipWrapper
		P	C	P	C	P	C	P	C	Avg. # Examples	Avg. # Post-processed	Avg. # Post-processed
Guitar and Microphone	63	10	14	0.82	0.58	0.84	0.63	1.79	0.09	3.7	6.1	28.2
No People	24	13	12	0.78	0.29	0.77	0.66	2.08	0.08	5.1	4.2	21.4
Bride Left of Groom	42	13	12	0.77	0.47	0.78	0.66	2.07	0.08	5.0	4.2	38.4
Bride and Not Groom	40	11	13	0.67	0.43	0.68	0.54	2.30	0.08	4.9	5.8	21.7
Overall	-	**49**	**49**	**0.76**	**0.45**	**0.82**	**0.61**	**2.11**	**0.08**	**4.8**	**5.0**	**27.6**



(a) User-reported ease of use. **(b) User-reported confidence in results.**

Figure 9: Results of post-task questionnaire, with standard deviation. Tasks with a statistically significant difference in scores are denoted with an asterisk.

synergistically: *"I gave more specific text queries in [PhotoScout], because I could back them up with examples"* (P25). By contrast, some participants noted that ClipWrapper required more general text queries: *"I tried to give [ClipWrapper] as little ambiguity as possible"* (P4).

During the study, example images clarified ambiguous or erroneous text queries. For instance, when completing the Guitar and Microphone task with PhotoScout, many participants made the text query "guitar and microphone." Based on this text alone, it is unclear whether the user wants all images containing a guitar and a microphone, or all images containing a guitar and all images containing a microphone. A negative example image containing just one of these objects quickly resolves this ambiguity. In another instance, when completing the No People task with PhotoScout, P14 made the text query "music festival containing no people." The term "music festival" in this query was unnecessary (as all images in the dataset were from a music festival) and could have added noise to the search results. However, because the participant had already added a set of example images for the task, PhotoScout figured out that this part of the query was extraneous, and output images containing no people.

KF3: Selecting example images is an intuitive process. Every participant utilized example images when completing tasks with PhotoScout, and selecting positive and negative example images was an easy process for most tasks. Usually, the participant quickly found positive and negative example images by scanning through the full image dataset. In some instances, participants made an initial text query, and then selected example images from the smaller set of preliminary search results.

During the tutorial, we explained what positive and negative examples were, but did not offer any insight into what makes a good or bad example image. Even so, during the study 11 out of the 25 participants noted that they intentionally selected diverse example images. For instance, P10 noted, *"for negative examples, I chose things that could be confusing."* Similarly, P7 said, *"I tried to find edge cases with one image that was totally different,"* and that especially for negative examples *"I tried to find sort of tricky cases where an image was almost correct."*

This strategy likely helps to produce correct results in PhotoScout, as the example images will filter out synthesized programs that are almost correct but are missing one key component. Even though participants were not given any information about the underlying search procedure, they independently found an effective strategy for selecting example images. This behavior suggests that example images are an intuitive addition to text-based image search.

6.3 Description of Failure Cases

Limitations of object detector. Because PhotoScout performs image retrieval by executing neuro-symbolic queries, its performance is dependent upon the accuracy of the underlying neural models for object detection. If an image contains a particular object present in the user's query, but PhotoScout does not detect that object (e.g. because it is partially obscured), then that image will not appear in the search results.

This issue is more apparent if the object detector does not work well on the images that the user selects as positive and negative examples. In particular, because PhotoScout synthesizes a program that matches all positive examples and rejects all negative ones, PhotoScout may fail to synthesize *any* programs if the object detector misclassifies relevant objects in the example images. Indeed, this limitation of PhotoScout proved problematic in the Bride Left of Groom task. Several participants selected an example image of the bride and groom dancing, where only the back of the bride's head is visible. While participants could easily infer that this person was the bride, PhotoScout did not classify her correctly. Hence, PhotoScout could not generate a program that matched both the user's text query and this example image, and the user was prompted to adjust their query. A common response was for participants to then add more example images in an attempt to correct this error. However, they would continue to get poor results as long as they had any example images where relevant objects were misclassified by the object detector.

As a result, participants sometimes felt more frustrated when using PhotoScout than when using ClipWrapper: 8 out of 25 participants noted instances where PhotoScout should have detected an object but did not. In some cases, this caused participants to lose trust in PhotoScout. During the Bride Left of Groom task, P5 noted, *"it decreases my confidence to know that [PhotoScout] misclassified the face I was looking for."* When completing the Guitar and Microphone task, P8 said, *"it's annoying when [PhotoScout] doesn't recognize a microphone in an image."*

Limitations of LLM. It is also the case that PhotoScout's framework may fail to output results when GPT is unable to produce a program sketch from the user's text query. If the user provides a query that is very dissimilar from any of the example text queries in the prompt provided to GPT, then the output programs may fail to parse. This was the case when P22 made the text query *"solo images of anna"* during the Bride and Not Groom task (where they had already tagged the bride as "anna"). If this happens, the user will see no search results and will be prompted to adjust their query.

Inspiration for future work. An interesting direction for future work is to explore interaction models that balance the *structure* of PhotoScout with the *flexibility* of ClipWrapper. ClipWrapper will also fail to detect objects, and often misinterprets text queries. However, ClipWrapper is designed for similarity-based search queries, and does not extract any hard constraints from queries. As such, users will almost always get *some* results from any query they provide to ClipWrapper. Even if those results are inaccurate, users may feel more encouraged to continue trying other queries or to edit their results manually. One user suggested that PhotoScout could allow users to edit image labels in cases where the object detector is incorrect. Several other users reported that they would like a fusion of the two tools, wherein they could explore the dataset with open-ended text or image queries in a separate panel, without having to adjust the text query and example images that would determine the hard constraints of their task.

Another line of future work could involve expanding the DSL to support tasks involving more fine-grained relationships between objects. In our current synthesis procedure, logical constraints involving multiple objects are approximated by predicates such as HasRelation(x, y, Above). This predicate can accurately describe a concept like "Bob is playing guitar," unless there are photos where Bob is above a guitar, but he is not playing it. Predicates that consider additional information, such as the distance between objects and the relative sizes of objects, or that involve more than two objects, could increase the expressivity of our DSL.

7 CONCLUSION

We have presented PhotoScout, a new multi-modal synthesis-based interface for automating image search tasks. With PhotoScout, users provide natural language descriptions of their search tasks, then interactively select example images and tag objects to refine their search. Our approach uses an LLM to synthesize program sketches in a neuro-symbolic DSL and then grounds those sketches using a PBE approach. We have evaluated our proposed approach by conducting a user study with 25 participants, wherein users completed image search tasks with PhotoScout and a deep learning-based image search tool. We found that participants performed tasks more accurately and with less manual work using PhotoScout.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for the useful and constructive feedback on this paper. This work was conducted in a research group supported by NSF awards CCF-1762299, CCF-1918889, CNS-1908304, CCF-1901376, CNS-2120696, CCF- 2210831, and CCF-2319471.

REFERENCES

[1] 2021. *Essential Planning: Your Wedding Photo Checklist.* https://onefabday.com/wedding-photo-checklist/

[2] 2023. Google Photos. https://photos.google.com/. Accessed: Dec 1, 2023.

[3] 2023. Photos Support. https://support.apple.com/photos. Accessed: Dec 1, 2023.

[4] 2023. Piktures - Beautiful Gallery. https://www.piktures.app/. Accessed: Dec 1, 2023.

[5] 2024. Amazon Rekognition. https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html. Accessed: Jan 22, 2024.

[6] 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, Sina Samangooei, Marianne Monteiro, Jacob L Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikoł aj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 23716–23736. https://proceedings.neurips.cc/paper_files/paper/2022/file/960a172bc7fbf0177ccccbb411a7d800-Paper-Conference.pdf

[7] Stefano Allegretti, Federico Bolelli, Federico Pollastri, Sabrina Longhitano, Giovanni Pellacani, and Costantino Grana. 2021. Supporting skin lesion diagnosis with content-based image retrieval. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 8053–8060.

[8] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. 2016. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5297–5307.

[9] Alberto Baldrati, Marco Bertini, Tiberio Uricchio, and A. Bimbo. 2023. Composed Image Retrieval using Contrastive Learning and Task-oriented CLIP-based Features. *ACM Transactions on Multimedia Computing, Communications and Applications* (2023). https://api.semanticscholar.org/CorpusID:261065158

[10] Celeste Barnaby, Qiaochu Chen, Roopsha Samanta, and Işıl Dillig. 2023. ImageEye: Batch Image Processing Using Program Synthesis. *Proc. ACM Program. Lang.* 7, PLDI, Article 134 (jun 2023), 26 pages. https://doi.org/10.1145/3591248

[11] Sean Bell and Kavita Bala. 2015. Learning visual similarity for product design with convolutional neural networks. *ACM transactions on graphics (TOG)* 34, 4 (2015), 1–10.

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

[13] A. Chalechale, G. Naghdy, and A. Mertins. 2005. Sketch-based image matching Using Angular partitioning. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 35, 1 (2005), 28–41. https://doi.org/10.1109/TSMCA.2004.838464

[14] Qiaochu Chen, Arko Banerjee, Çağatay Demiralp, Greg Durrett, and Isil Dillig. 2023. Data Extraction via Semantic Regular Expression Synthesis. arXiv:2305.10401 [cs.PL] https://arxiv.org/abs/2305.10401

[15] Yanbei Chen, Shaogang Gong, and Loris Bazzani. 2020. Image search with text feedback by visiolinguistic attention learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3001–3011.

[16] Jooae Choe, Hye Jeon Hwang, Joon Beom Seo, Sang Min Lee, Jihye Yun, Min-Ju Kim, Jewon Jeong, Youngsoo Lee, Kiok Jin, Rohee Park, et al. 2022. Content-based image retrieval by using deep learning for interstitial lung disease diagnosis with chest CT. *Radiology* 302, 1 (2022), 187–197.

[17] I.J. Cox, M.L. Miller, T.P. Minka, and P.N. Yianilos. 1998. An optimized interaction strategy for Bayesian relevance feedback. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*. 553–558. https://doi.org/10.1109/CVPR.1998.698660

[18] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. 2008. Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM Comput. Surv.* 40, 2, Article 5 (may 2008), 60 pages. https://doi.org/10.1145/1348246.1348248

[19] Shiv Ram Dubey. 2022. A Decade Survey of Content Based Image Retrieval Using Deep Learning. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 5 (2022), 2687–2704. https://doi.org/10.1109/TCSVT.2021.3080920

[20] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. 2018. Learning to Infer Graphics Programs from Hand-Drawn Images. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/6788076842014c83cedadbe6b0ba0314-Paper.pdf

[21] Yuchun Fang, Donald Geman, and Nozha Boujemaa. 2005. An Interactive System for Mental Face Retrieval. In *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval* (Hilton, Singapore) *(MIR '05)*. Association for Computing Machinery, New York, NY, USA, 193–200. https://doi.org/10.1145/1101826.1101858

[22] Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual Programming: Compositional visual reasoning without training. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 14953–14962. https://doi.org/10.1109/CVPR52729.2023.01436

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[24] Jiani Huang, Calvin Smith, Osbert Bastani, Rishabh Singh, Aws Albarghouthi, and Mayur Naik. 2020. Generating Programmatic Referring Expressions via Program Synthesis. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 4495–4506. https://proceedings.mlr.press/v119/huang20h.html

[25] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[26] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. Inferring and Executing Programs for Visual Reasoning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[27] Deok-Hwan Kim and Chin-Wan Chung. 2003. QCluster: Relevance Feedback Using Adaptive Clustering for Content-Based Image Retrieval. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data* (San Diego, California) *(SIGMOD '03)*. Association for Computing Machinery, New York, NY, USA, 599–610. https://doi.org/10.1145/872757.872829

[28] Gunhee Kim, Seungwhan Moon, and Leonid Sigal. 2015. Ranking and retrieval of image sequences from multiple paragraph queries. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1993–2001. https://doi.org/10.1109/CVPR.2015.7298810

[29] Adriana Kovashka, Devi Parikh, and Kristen Grauman. 2012. Whittlesearch: Image search with relative attribute feedback. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2973–2980.

[30] Afshan Latif, Aqsa Rasheed, Umer Sajid, Jameel Ahmed, Nouman Ali, Naeem Iqbal Ratyal, Bushra Zafar, Saadat Hanif Dar, Muhammad Sajid, Tehmina Khalil, et al. 2019. Content-based image retrieval and feature extraction: a comprehensive review. *Mathematical problems in engineering* 2019 (2019).

[31] Seungmin Lee, Dongwan Kim, and Bohyung Han. 2021. Cosmo: Content-style modulation for image retrieval with text feedback. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 802–812.

[32] Zheyuan Liu, Cristian Rodriguez-Opazo, Damien Teney, and Stephen Gould. 2021. Image Retrieval on Real-Life Images With Pre-Trained Vision-and-Language Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2125–2134.

[33] Ye Lu, Chunhui Hu, Xingquan Zhu, HongJiang Zhang, and Qiang Yang. 2000. A Unified Framework for Semantics and Feature Based Relevance Feedback in Image Retrieval Systems. In *Proceedings of the Eighth ACM International Conference on Multimedia* (Marina del Rey, California, USA) *(MULTIMEDIA '00)*. Association for Computing Machinery, New York, NY, USA, 31–37. https://doi.org/10.1145/354384.354403

[34] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. 2019. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJgMlhRctm

[35] C. Nastar, M. Mitschke, and C. Meilhac. 1998. Efficient query refinement for image retrieval. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*. 547–552. https://doi.org/10.1109/CVPR.1998.698659

[36] Adnan Qayyum, Syed Muhammad Anwar, Muhammad Awais, and Muhammad Majid. 2017. Medical image retrieval using deep convolutional neural network. *Neurocomputing* 266 (2017), 8–20.

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

[38] Scott E. Reed and Nando de Freitas. 2016. Neural Programmer-Interpreters. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.06279

[39] Yong Rui, T.S. Huang, M. Ortega, and S. Mehrotra. 1998. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology* 8, 5 (1998), 644–655. https://doi.org/10.1109/76.718510

[40] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[41] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 11523–11530. https://doi.org/10.1109/ICRA48891.2023.10161317

[42] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. 2000. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 12 (2000), 1349–1380. https://doi.org/10.1109/34.895972

[43] Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. ViperGPT: Visual Inference via Python Execution for Reasoning. arXiv:2303.08128 [cs.CV] https://arxiv.org/abs/2303.08128

[44] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. 2019. Learning to Infer and Execute 3D Shape Programs. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rylNH20qFQ

[45] Kinh Tieu and Paul Viola. 2004. Boosting Image Retrieval. *International Journal of Computer Vision* 56, 1 (2004), 17–36. https://doi.org/10.1023/B:VISI.0000004830.93820.78

[46] Nuno Vasconcelos and Andrew Lippman. 1999. Learning from User Feedback in Image Retrieval Systems. In *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller (Eds.), Vol. 12. MIT Press. https://proceedings.neurips.cc/paper_files/paper/1999/file/7283518d47a05a09d33779a17adf1707-Paper.pdf

[47] J.Z. Wang, Jia Li, and G. Wiederhold. 2001. SIMPLIcity: semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 9 (2001), 947–963. https://doi.org/10.1109/34.955109

[48] Xianwang Wang, Tong Zhang, Daniel R. Tretter, and Qian Lin. 2013. Personal Clothing Retrieval on Photo Collections by Color and Attributes. *IEEE Transactions on Multimedia* 15, 8 (2013), 2035–2045. https://doi.org/10.1109/TMM.2013.2279658

[49] Yifeng Wang, Zhi Tu, Yiwen Xiang, Shiyuan Zhou, Xiyuan Chen, Bingxuan Li, and Tianyi Zhang. 2023. Rapid Image Labeling via Neuro-Symbolic Learning. *arXiv preprint arXiv:2306.10490* (2023).

[50] Wedgewood Weddings. [n. d.]. *Ultimate Wedding Shot List: Photography Guide*. https://www.wedgewoodweddings.com/blog/ultimate-wedding-shot-list

[51] Haokun Wen, Xian Zhang, Xuemeng Song, Yin wei Wei, and Liqiang Nie. 2023. Target-Guided Composed Image Retrieval. *Proceedings of the 31st ACM International Conference on Multimedia* (2023). https://api.semanticscholar.org/CorpusID:261530782

[52] Tobias Weyand, Ilya Kostrikov, and James Philbin. 2016. Planet-photo geolocation with convolutional neural networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*. Springer, 37–55.

[53] Hao Xu, Jingdong Wang, Xian-Sheng Hua, and Shipeng Li. 2010. Image Search by Concept Map. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Geneva, Switzerland) *(SIGIR '10)*. Association for Computing Machinery, New York, NY, USA, 275–282. https://doi.org/10.1145/1835449.1835497

[54] Halley Young, Osbert Bastani, and Mayur Naik. 2019. Learning Neurosymbolic Generative Models via Program Synthesis. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 7144–7153. https://proceedings.mlr.press/v97/young19a.html

[55] Feifei Zhang, Mingliang Xu, and Changsheng Xu. 2022. Tell, imagine, and search: End-to-end learning for composing text and image to image retrieval. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 18, 2 (2022), 1–23.

[56] Kai Zhang, Shouliang Qi, Jiumei Cai, Dan Zhao, Tao Yu, Yong Yue, Yudong Yao, and Wei Qian. 2022. Content-based image retrieval with a Convolutional Siamese Neural Network: Distinguishing lung cancer and tuberculosis in CT images. *Computers in biology and medicine* 140 (2022), 105096.

[57] Xiang Sean Zhou and Thomas S. Huang. 2003. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems* 8, 6 (2003), 536–544. https://doi.org/10.1007/s00530-002-0070-3

[58] Terry Yue Zhuo, Zhuang Li, Yujin Huang, Yuan-Fang Li, Weiqing Wang, Gholamreza Haffari, and Fatemeh Shiri. 2023. On robustness of prompt-based semantic parsing with large pre-trained language model: An empirical study on codex. *arXiv preprint arXiv:2301.12868* (2023).

Received 13 September 2023; revised 13 December 2023; accepted 22 February 2024