CobraVPS: Code Template Optimization for Better Question Reasoning Accuracy with Visual Program Synthesis

Supplementary Material

1. Pesudo-code Template Examples

In this section, we show several code examples in different code templates for a given query class. We use query types *relVerifyCr* and *chooseAttr* as examples in Fig. 1(a) and Fig. 1(b), respectively. For the query class *relVerifyCr*, the query involves questions about the relative positions of two target objects, while the other query class *chooseAttr* consists of questions regarding the features/attributes of a single target object. For each query class, there are multiple logically correct code templates.

As seen in Fig. 1(a), for the query class *relVerifyCr*, Template 1 directly uses a `simple_query` function, powered by an end-to-end VQA model, such as BLIP[1], to obtain the answer. In contrast, Template 2 first detects the target objects. If they are detected, the positions of the objects are checked based on their horizontal center. If target objects are not detected, the question is asked directly to the end-to-end VQA model.

For the queries of type *chooseAttr*, Template 1 directly feeds the entire question into a `simple_query` function (through an end-to-end VQA model) as seen in Fig. 1(b). Template 2, on the other hand, first detects the target object, then calls the `simple_query` function on the detected target object patch. Different from Template 1 and Template 2, Template 3 first detects the target object, and then employs the `verify_property` function to check if the detected object has the specified feature/attribute.

Overall, these templates are all logically correct, but their VQA accuracy can vary significantly. This can be attributed to the strengths/weaknesses of different vision models when used with different input images and query types.

2. Ablation Study on Effect of Qwen

Index	Testing Prompt	LLM	Accuracy
1	Query+API + 3 fixed shot (ViperGPT)	CodeLlama-7B	54.40%
2	Query+API + 3 fixed shot + Intermediate Steps (ViperGPT)	CodeLlama-7B	56.42%
3	Query+API +3 shots in Best Template (CobraVPS)	CodeLlama-7B	63.09%

Table 1. **Effect of intermediate steps from Qwen.** VQA accuracy when different prompts are used with the Code LLM.

As outlined in Section 4.1 of our main paper, Qwen is utilized to generate intermediate steps during the generation of Code Base B. We have conducted an ablation study to analyze the impact of generating intermediate steps through Qwen, and reported the results in Table 1. For Experiments 1 and 2, we employed ViperGPT without and with the intermediate steps from Qwen, respectively. Comparing the first two rows of Table 1 shows that using the intermediate steps improves the accuracy of ViperGPT from 54.40% to 56.42%, by 2.02%. However, as shown in row 3 (Experiment 3), our proposed CobraVPS still outperforms the ViperGPT, using intermediate steps (Experiment 2), by 6.67% on the VQA accuracy.

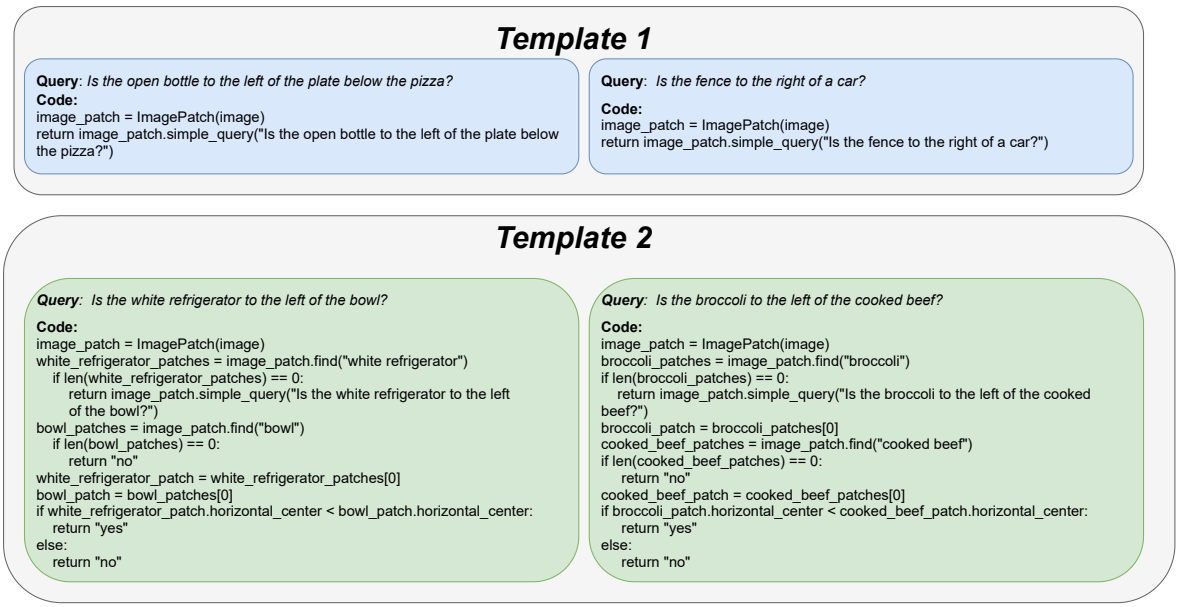
3. Qualitative Results

Figure 2 presents several example images and queries to provide a qualitative comparison between ViperGPT and our proposed CobraVPS. Since the code of the VisRep is not publicly available, in our main paper, we reported the numbers from the original paper, and were not able to run the code ourselves. For this reason, we only present the qualitative comparison with ViperGPT.

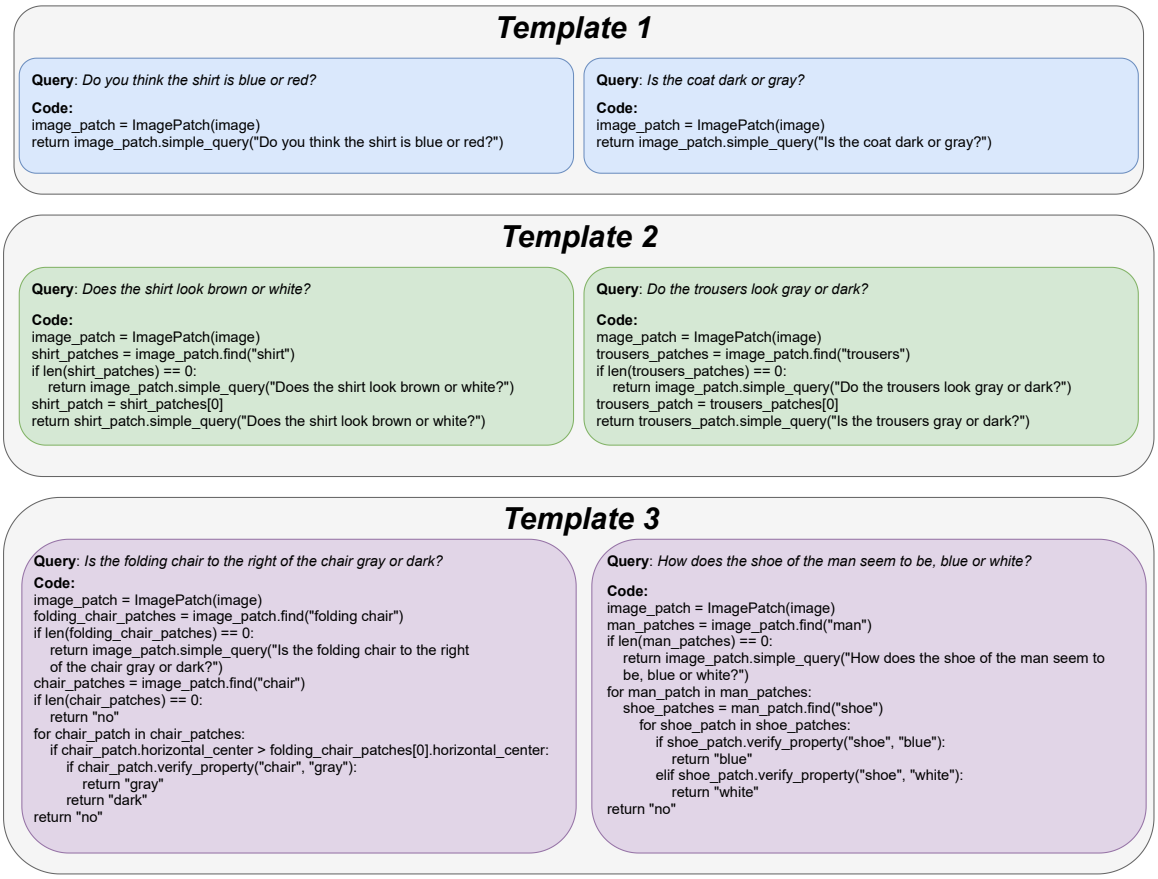
In the first example, the code outputted by ViperGPT first detects a fence then wire. If the wire patch overlaps with fence, it returns "yes," otherwise, it returns "no". Thus, based on this generated code, ViperGPT misinterprets the question as "*Is there a wire patch overlapping with a fence?*". Our proposed CobraVPS, on the other hand directly uses the `simple_query` function to obtain the answer for this query.

In the second example, both codes outputted by ViperGPT and our CobraVPS are logically correct, but in different code templates. The code template used by ViperGPT relies on the `find` and `verify_property` functions. Thus, after finding the persons, this code focuses on smaller person patches to try to verify their property as a skier or bartender, which apparently results in the wrong answer. Our CobraVPS, on the other hand, uses the best template chosen for this query type, which uses the `exists` function on the whole image possibly incorporating more context. This results in the correct answer to this question.

In the third example, the code generated by ViperGPT involves incorrect logic, limiting the answer to either "car" or "truck." In contrast, the code generated by our proposed CobraVPS first detects vehicle(s) and then calls the `simple_query` function with the cropped vehicle image patch to provide the correct answer in this example as well.



(a) relVerifyCr



(b) chooseAttr

Figure 1. This picture shows example codes from different template for query class relVerifyCr and chooseAttr respectively

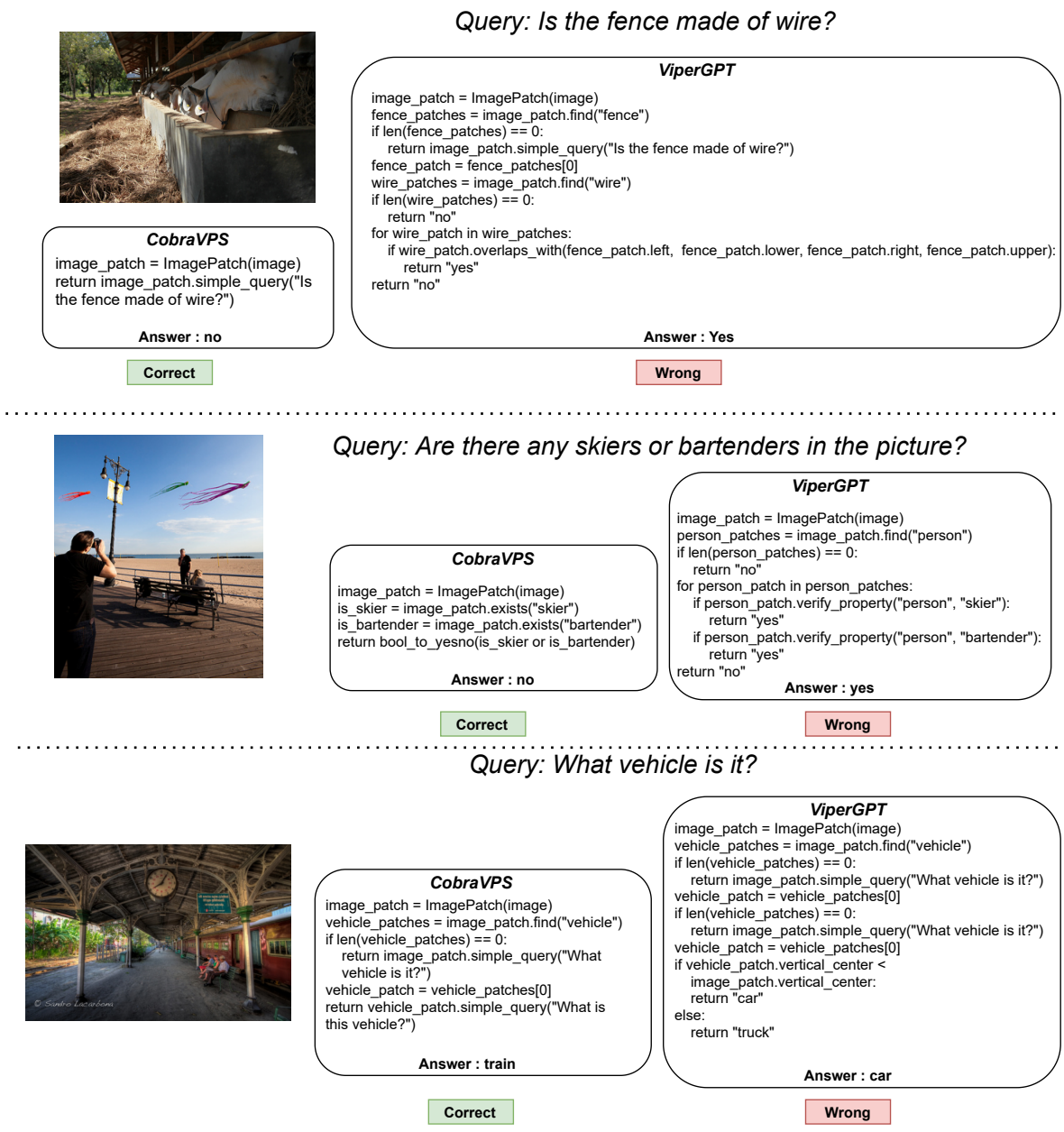


Figure 2. Qualitative results comparing our proposed CobraVPS with ViperGPT on three different VQA examples.

References

[1] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023. 1