

# Thinking with Programming Vision: Towards a Unified View for Thinking with Images

## Supplementary Material

### Case Study

To provide a clearer understanding of our model’s behavior and the impact of our reward design, we present four case studies.

Figure 11 illustrates a common failure mode known as reward hacking, which we observe when training a model without our proposed constraint penalties. In this example, the agent is tasked with a problem that requires a 90-degree rotation. The agent correctly identifies and applies the `rotate90` tool in its first turn, successfully correcting the image’s orientation. However, driven solely by the goal of maximizing the strategy reward, it does not stop. Instead, it proceeds to call additional, unnecessary orientation tools in subsequent turns, which corrupts the already-correct image and ultimately leads to failure. This case underscores the critical importance of the penalty functions, which act as guardrails to prevent such inefficient and counterproductive behavior.

In contrast, Figure 12 showcases the robust, multi-step reasoning capabilities of our fully-trained CodeVision model. The task requires the model to read text from a specific region that is too small to be seen in the initial view. The model first performs a “crop” operation to zoom in. Recognizing that its initial crop was incomplete and missed part of the target area, it dynamically adjusts its strategy. In the next turn, it executes a second, more precise “crop” that successfully isolates the entire region of interest. With the full context now visible, the model is able to accurately provide the correct answer. This example highlights the model’s ability to self-correct and perform fine-grained, iterative adjustments, a key capability for solving complex real-world visual tasks.

Besides, Figure 13 presents a nuanced success case that reveals areas for further refinement. The task requires a sequence of orientation correction followed by localizing and reading a very small line of text. The model adeptly handles the first step by correcting the image’s orientation. It then correctly reasons about the text’s location and performs a crop. Although the cropped region successfully includes the target text, allowing the model to answer correctly, the crop itself is suboptimal—a long, narrow strip containing significant irrelevant context. This example demonstrates that while the model’s high-level reasoning is effective, its ability to generate precise, tight bounding boxes could be improved, suggesting it may sometimes adopt an overly cautious safe cropping strategy.

To further probe the model’s limitations, we present a

clear failure case in Figure 14. The task requires correcting the image orientation and then cropping a small, specific region to examine details. The model successfully completes the first two stages: it correctly adjusts the orientation and accurately reasons about the approximate location of the region of interest. However, it fails at the final, crucial step of precise localization. The coordinates it generates for the crop are slightly off, causing the cropped area to be adjacent to the target region but missing it entirely. This case clearly demonstrates that while the model excels at high-level reasoning and coarse localization, its fine-grained coordinate prediction remains a challenge and a key area for future improvement.

### MultiTool Benchmark Construction

To rigorously evaluate a model’s ability to compose multiple tools in sequence, we construct the MVToolBench. The construction process is designed to create challenging scenarios that necessitate the combined use of multiple tools. The pipeline consists of three main stages: data filtering, question generation, and multi-tool augmentation.

**Data Source and Filtering.** We build our benchmark upon the HierText dataset, which provides rich annotations including bounding box coordinates and corresponding text for words, lines, and paragraphs. To ensure that our benchmark effectively tests the model’s ability to perceive fine-grained details, we first perform a filtering step based on the relative size of the text annotations. For each image, we calculate the area of every annotated word, line, and paragraph as a percentage of the total image area. We then retain only the annotations that are exceptionally small. For instance, for word-level questions, only words occupying less than 0.01% of the total image area are selected as candidates. This filtering serves two critical purposes: 1) it guarantees a high level of difficulty, as the target text is not easily visible in the original view, and 2) it enforces a strong dependency on the “crop” tool, as zooming in becomes a prerequisite for successful recognition and reasoning.

**Question and Answer Generation.** Using the filtered, small-sized text annotations, we programmatically generate a variety of reasoning questions. These questions cover tasks such as text recognition (e.g., “What does the line beginning with “Busy” say?”), counting (e.g., “How many times does the letter “a” appear in the paragraph ending

**Question: How many pages did the reader read on average per month in 2023?**

**Model Response:**

Thought ...

The text in the image appears reversed left-to-right, suggesting the entire figure is mirrored. I will try a horizontal mirror first to see if it makes the text readable.

Tool Call to Image O

# Try mirroring horizontally to make text readable  
result = ImageOps.mirror(image)

Tool Return: Image 1

Thought ...

The previous mirror made the text backward but still unreadable. The text appears reversed in orientation, suggesting the image is mirrored top-to-bottom. I will try a vertical flip to make the text upright and readable.

Tool Call to Image O

# Try vertical flip to make text upright and readable  
result = ImageOps.flip(image)

Tool Return: Image 2

Thought ...

The text remains backward and mirrored; the title and all labels read backward, indicating the image is reversed left-to-right. To make the text readable, I should rotate the image 180° to correct the orientation.

Tool Call to Image O

# Rotate 180° to make text upright and readable  
result = image.rotate(180, expand=True)

Tool Return: Image 3

Thought ...

The text is now readable and the layout is correct; the table shows total pages read (324) and the number of months (12). To find the average pages per month:  $324 \text{ pages} / 12 \text{ months} = 27 \text{ pages per month}$ .

Answer. 27



Image 0 (inputted)



Image 1

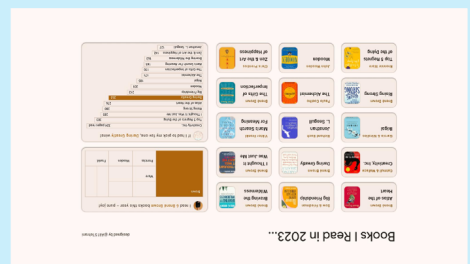


Image 2



Image3

Figure 11. An example of reward hacking from a model trained without constraint penalties. After correctly applying rotate90, the agent continues to call superfluous tools, leading to task failure.

with “Television”?), and question answering that requires information retrieval from specific paragraphs. A crucial design principle during this stage is the deliberate avoidance of any explicit positional cues. We do not use phrases like “the word on the left” or provide bounding box coordinates in the prompts. This forces the model to rely entirely on its own grounding capabilities to first locate the relevant region and then perform an accurate crop, significantly increasing

the challenge of the benchmark.

**Multi-Tool Augmentation.** After the question-answer pairs are finalized, we perform a final augmentation step on the images to create a multi-tool requirement. Each image is randomly subjected to one of several transformations: rotation (90, 180, or 270 degrees), horizontal flip, or vertical flip. This step ensures that solving a problem requires

**Question:** What is the last word of the line beginning with the phrase "Bouncy with..."?

**Model Response:**

Thought ...

The phrase "Bouncy with..." appears in the lower-left area of the webpage within the "Awards" section as the caption for the small thumbnail image labeled "Bouncy with toon shading." To answer the question about the last word of that line, I should crop tightly around the text area that contains this caption to isolate both the word and trailing punctuation so the word at the end is clear.

Tool Call to Image O

```
# Crop the region containing the line starting with 'Bouncy with...'.  
# Approximate ROI identified visually around the lower-left awards text.  
xmin, ymin, xmax, ymax = 763, 605, 917, 619  
result = image.crop((xmin, ymin, xmax, ymax))
```

Tool Return: Image 1

Thought ...

The initial crop around the caption shows the full phrase "Bouncy with toon shading," but the right edge and the word at the far end are cut off. To clearly read the last word of that line, I should expand the crop to include a bit more space toward the right so the trailing punctuation and the full word "shading" are fully visible. A wider crop centered on that line will allow me to confirm the complete wording and identify the final word.

Tool Call to Image O

```
# Improve crop around the line starting with 'Bouncy with...'.  
xmin, ymin, xmax, ymax = 737, 603, 943, 624  
result = image.crop((xmin, ymin, xmax, ymax))
```

Tool Return: Image 2

Thought ...

The refined crop shows the caption clearly as "Bouncy with toon shading." The sequence reads as a single phrase with the word "shading" at the end followed by a trailing space, indicating the last word on that line is "shading."

**Answer.** shading

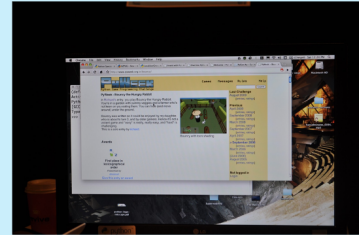


Image 0 (inputted)

Bouncy with toon shading

Image 1

Bouncy with toon shading

Image 2

Figure 12. A successful multi-step reasoning case. The model first performs an initial crop, then recognizes its incompleteness and refines the cropped region in a second step to include all necessary information before providing the final answer.

not only a “crop” operation but also an initial orientation-correction tool. By setting the proportion of each transformation to be equal, we maintain a balanced and unbiased dataset. This three-stage process results in a comprehensive benchmark that effectively evaluates the model’s capacity for complex, multi-step tool composition.

## Limitation and Future Work

In this work, we have focused on a core set of tools—primarily orientation correction and cropping—to establish the CodeVision framework and demonstrate its effectiveness in addressing model brittleness. While this targeted approach has proven successful, creating a truly general-purpose visual agent requires scaling and expansion in several key dimensions. Our work reveals several limitations that open up exciting avenues for future research:

**Expanding Tool Diversity and Compositionality.** Our current research deliberately concentrates on a limited toolset. To enhance the model’s generalization capabilities, future work should incorporate a much wider variety of tool types and data. This includes training on tasks that require more complex tool compositions and even multi-image tool use (e.g., comparing, merging, or analyzing multiple images simultaneously). Furthermore, the “code-as-tool” paradigm can be extended beyond standard Python libraries to include custom tools, such as proprietary search engines or generative models. By exposing only a simple API endpoint to the model, we can empower it to leverage powerful, black-box functionalities, further broadening its problem-solving horizons.

**Refining Process Supervision with Beneficial Tools.** Our current training methodology relies on a “must-use” tool list to provide strong process supervision. While ef-

Question: How many words are on the line that starts with the text "Ripped...?"

Model Response:

Thought ...

The page text and UI elements are mirrored horizontally, indicating a left-right reversal. I will flip the image horizontally to restore the correct orientation.

Tool Call to Image 0

result = ImageOps.mirror(image)

Tool Return: Image 1

Thought ...

The current page is now correctly oriented. I need to isolate the line that begins with the text "Ripped..." on the right sidebar, which contains the description reading "Ripped Freak endelig i Norge. Et av de mest populære kosttillskuddene i USA/Canada for maksimal fettforbrømming. 349 kr for 60 kapsler." I will crop a tight region around that specific line.

Tool Call to Image 1

# Crop the line starting with 'Ripped' on the right sidebar

xmin, ymin, xmax, ymax = 35, 266, 1004, 283

result = image.crop((xmin, ymin, xmax, ymax))

Tool Return: Image 2

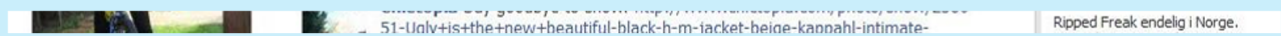


Image 2

Thought ...

The cropped line reads: "Ripped Freak endelig i Norge." Counting words: Ripped(1), Freak(2), endelig(3), i(4), Norge.(5).

Answer: 5



Image 0 (inputted)

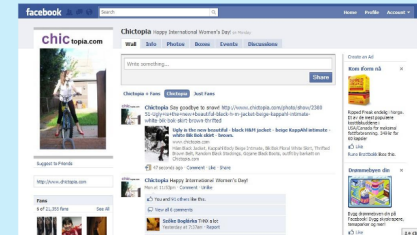


Image 1

Figure 13. A successful but inefficient localization case. The model corrects the orientation and successfully crops a region containing the target text. However, the crop is a long, narrow strip that includes much irrelevant information, pointing to potential improvements in generating more precise bounding boxes.

fective, this can be expanded to a more flexible framework. Future iterations could incorporate a broader set of "beneficial" tools. These are tools that are not strictly necessary to solve a task but are highly likely to improve performance, such as contrast enhancement for low-light images or a preliminary crop for focusing attention. By rewarding the use of such advantageous tools, we can encourage the model to learn more nuanced strategies and generalize better to tasks where the optimal toolset is not rigidly defined.

**Scaling Data, Tasks, and Model Exploration.** Our experiments indicate that the model's learning has not yet reached saturation. The entropy of its policy suggests that there is still significant room for exploration and improvement. Moreover, as shown in Figure 15, we observe that performance on our benchmarks continue to increase steadily throughout training with no signs of plateauing. This strongly suggests that our approach would benefit from scaling up. By sourcing more diverse data, incorporating a wider array of tool types, and designing more varied tasks, we believe there is a clear path toward developing even more capable and robust visual agents.

Question: What is the full text of the paragraph opening with the words "RECHERCHE JESSICA...?"

**Model Response:**

Thought ...

The text and headings in the image appear upside down—the large title and all labels are inverted top-to-bottom while not mirrored left-to-right. This indicates the entire page is vertically flipped. To read the paragraph that begins with “RECHERCHE JESSICA...”, I should correct the orientation by flipping the image vertically. I will apply a vertical flip using the tool and then read the paragraph.

Tool Call to Image 0

```
# Flip the image vertically to correct upside-down orientation
result = ImageOps.flip(image)
```

Tool Return: Image 1

Thought ...

The corrected image now shows all text upright and readable. The paragraph beginning with “RECHERCHE JESSICA...” appears as a small line of attribution text in the bottom-right area of the page. I will crop that specific line so the full text can be read clearly.

Tool Call to Image 1

```
# Crop the attribution line beginning with 'RECHERCHE JESSICA...' in the
bottom-right
```

```
w, h = image.size
```

```
# Estimated ROI for the attribution text
```

```
left, top, right, bottom = 670, 1530, 1270, 1550
```

```
result = image.crop((left, top, right, bottom))
```

Tool Return: Image 2

Thought ...

The cropped bottom-right attribution line is sharp and fully legible. It reads the complete attribution text beginning with “RECHERCHE JESSICA...”. I can now provide the exact text.

Answer. RECHERCHE JESSICA NADEAU — INFOGRAPHIE JUSTIN STAHLMAN, JDÉM



Image 0 (inputted)

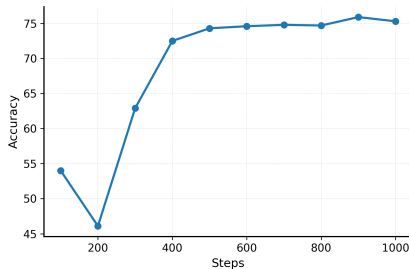


Image 1

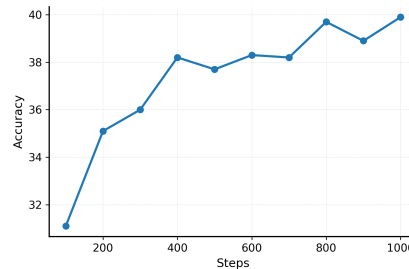


Image 2

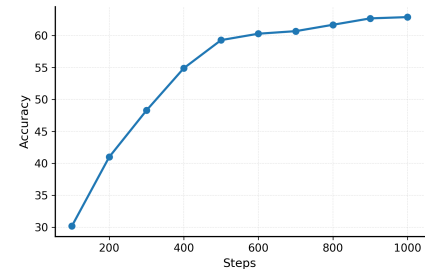
Figure 14. A failure case highlighting the model’s limitations in precise localization. The model correctly adjusts the orientation and identifies the general area of interest, but provides slightly inaccurate coordinates for the crop, resulting in a cropped region that just misses the target.



(a) OCRBench-Rot90



(b) ChartQAPro-Hori



(c) MVToolBench

Figure 15. Accuracy on OCRBench, ChartQAPro, and MVToolBench during RL training. The steady improvement across all benchmarks indicates that the model’s learning has not yet saturated and could benefit from further scaling of data and tasks.