LLM-Guided Semantic Image Compression via CLIP-Guided Prompt Refinement

Jinheng Zhang, Jialun Yang, Ziao You and Ricardo Li

CIS 5300, University of Pennsylvania, Philadelphia, PA 19104

December 17, 2025

Roadmap

- Motivation: why semantic compression and why language.
- Task definition: what “semantic” and “compression” mean here.
- Method: CLIP-guided multi-prompt refinement loop.
- Setup: data, metric, rate, baselines.
- Results: CLIP curves + qualitative examples.
- Analysis: why it improves, error modes, and next steps.

Motivation: why semantic image compression?

- Standard compression transmits pixels. At very low rate, pixels become too expensive.
- Many use cases only need the **meaning**:
  - scene type (airport / city street / living room),
  - salient objects (people, vehicles, furniture),
  - overall context (night + rain + neon signs).
- Text is a compact representation of meaning.
- Modern generators can "fill in" details from strong learned priors.

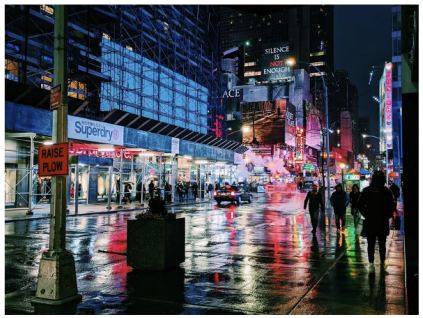
Key definitions

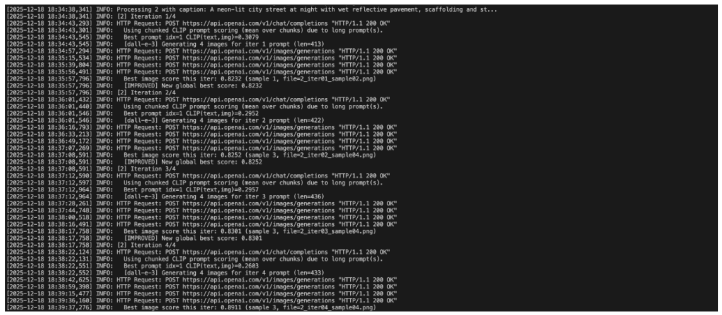
- **Semantic:** we do not require the same layout or exact pixels.
  - The reconstruction can shift viewpoint or geometry.
  - We only care if the meaning is similar.
- **Compression:** the transmitted code is a short prompt.
  - Typically a few hundred bytes.
  - This is the rate bottleneck, not compute at the decoder.

Goal: improve semantic similarity without increasing prompt length.

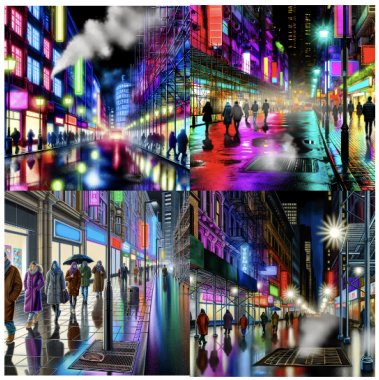
Illustrative example

Initial image





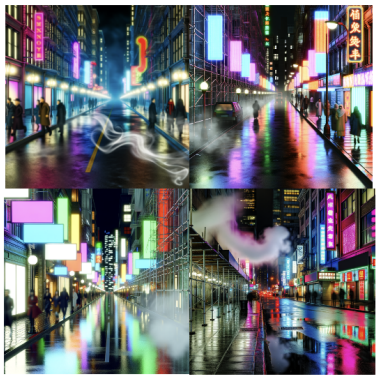
Iter 1 CLIP Score: 0.8232



Iter 2 CLIP Score: 0.8252

Iter 3 CLIP Score: 0.8301

Iter 4 CLIP Score: 0.8911



Improved 8.25% !

- We start from a short caption/prompt.
- Iteratively refine prompt $\Rightarrow$ higher CLIP similarity reconstructions.

Problem formulation

- Reference image: I
- Prompt code: p with constraint $\mathrm{len}(p) \leq R$
- Decoder: text-to-image model G, reconstruction $\hat{I} \sim G(p)$
- Objective:

$$\max_{p:\ \mathrm{len}(p) \leq R} \ \mathbb{E}_{\hat{I} \sim G(p)} \Big[S(\hat{I}, I) \Big].$$

- We use $S(\hat{I}, I) = \mathrm{CLIPSim}(I, \hat{I})$ (CLIP image embedding cosine).

Evaluation metric: CLIPSim

- Use CLIP ViT-B/32 image embedding $f_{\text{img}}(\cdot)$.
- Normalize to unit vectors and compute cosine similarity:

$$\text{CLIPSim}(I, \hat{I}) = \frac{f_{\text{img}}(I)^\top f_{\text{img}}(\hat{I})}{\|f_{\text{img}}(I)\| \cdot \|f_{\text{img}}(\hat{I})\|}.$$

- Dataset score: mean over images.
- Why this metric fits our goal:
  - tolerant to layout drift,
  - rewards correct scene type + salient objects.

Rate metric: prompt bits-per-pixel

- We transmit only the prompt text.
- If prompt is B bytes and image size is $H \times W$:

$$\mathrm{bpp}_{\text{text}} \approx \frac{8B}{HW}.$$

- For $H = W = 1024$ and $B \in [400, 600]$ bytes:

$$\mathrm{bpp}_{\text{text}} \approx 0.003\text{–}0.005.$$

- This is our main reason for using **short prompts**.

Baselines (from Milestone 2)

- Simple baseline: low-resolution resize (short side = 32).
- Strong baseline: JPEG low quality (q = 10).
- Baseline CLIPSim (Milestone 2):

Method	CLIPSim	Notes
Low-res resize	0.8521	simple pixel baseline
JPEG (q=10)	0.9256	strong classical codec

- These baselines transmit **pixels**, so they preserve more layout.

Rate vs baselines trade-off

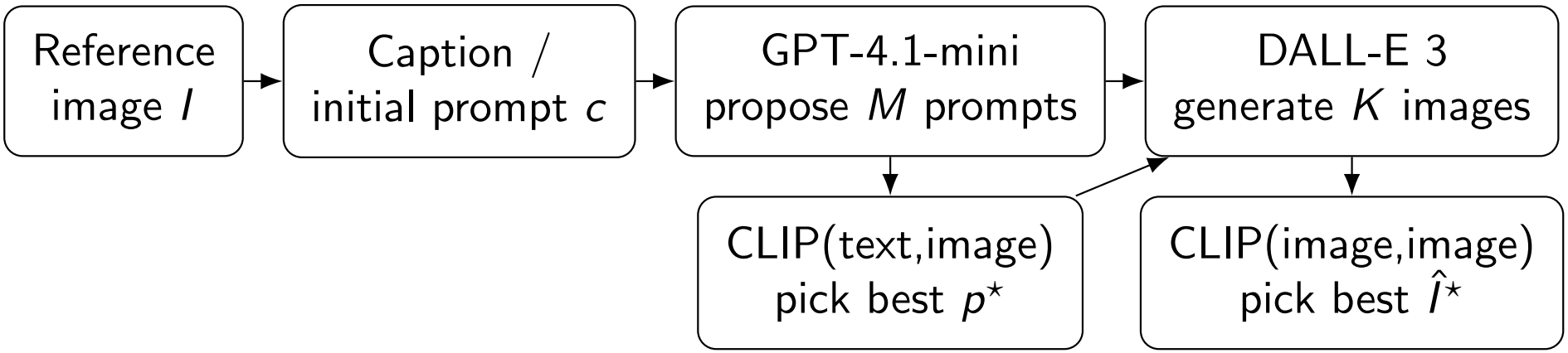
- Our prompt-based code: typically 400–600 bytes
  $\Rightarrow \mathrm{bpp}_{\text{text}} \approx 0.003$–$0.005$ at 1024^2.
- Low-res baseline implicitly sends a 32×32 RGB thumbnail:

$$32 \cdot 32 \cdot 3 = 3072 \text{ bytes} \Rightarrow \approx 0.023 \text{ bpp at } 1024^2.$$

- JPEG bitstreams are usually larger still (depends on quality and image).

Interpretation: baselines get higher CLIPSim partly because they transmit much more visual structure.

Our method: pipeline overview



- We do **search in prompt space** with CLIP as the critic.
- Keep best-so-far for stability (hill-climb).

Algorithm details

Per image:

1. Compute reference CLIP image embedding once.
2. For iteration $t = 1, \ldots, N$:
   - LLM produces M prompts: initial ($t = 1$) or variations of best prompt ($t>1$).
   - Score prompts using CLIP(text,image), pick best $p^\star$.
   - Generate K images with DALL-E 3 using $p^\star$.
   - Score images using CLIP(image,image), pick best $\hat{I}^\star$.
   - Update global best if score improves.

Implementation

- Refiner: `gpt-4.1-mini` (cheap enough for development + sweeps).
- Decoder: `dall-e-3`, `quality=hd`, `style=natural`.
- Critic: CLIP ViT-B/32.
- Hyperparameters in the main run:

$$N = 4, \quad M = 3, \quad K = 4.$$

- Compute per image (roughly):
  - LLM calls: N (one per iteration),
  - image generations: $N \cdot K$ (16 images),
  - CLIP scoring: cheap compared to generation.

Extension: conditional chunked prompt scoring

- CLIP text encoder has a context length limit.
- If a prompt would be truncated, we:
  - split prompt into CLIP-fitting chunks,
  - score each chunk,
  - use **mean chunk score** as the prompt score.
- In our main runs prompts are short on purpose, but this prevents silent truncation when prompts get longer.

Data

- 20 complex scene images total (city, indoor, public spaces, etc.).
- 5 are included in the submission.
- Captions in `captions.json` are used as the initial semantic description.
- We run refinement for $N = 4$ rounds because improvements often saturate early.

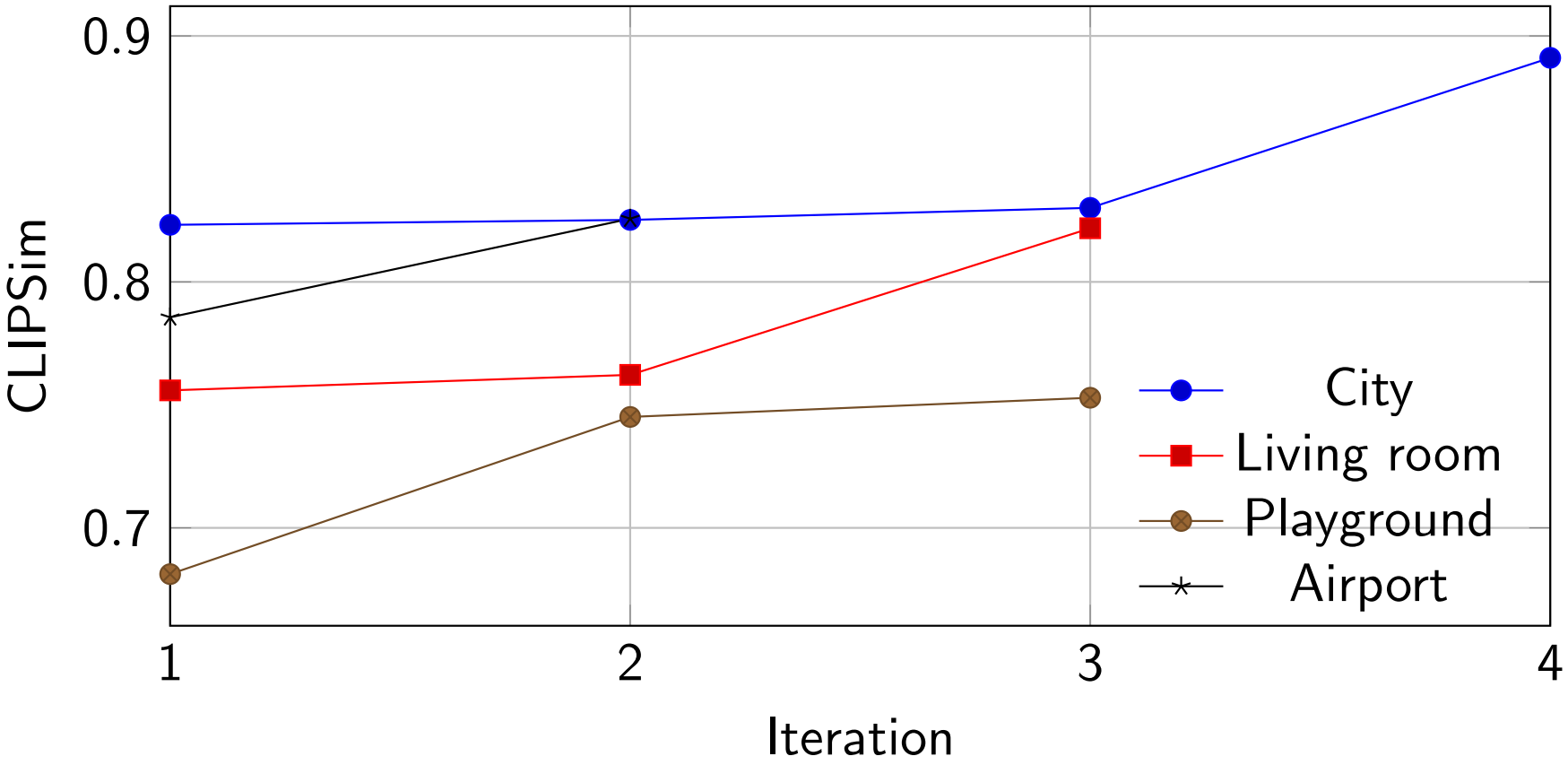
Main dataset-level observation

From the 20-image run:

- About **half** of the images show **steady improvement** during the first 3 iterations.
- The other half are not monotone, but their best CLIPSim often happens at **iteration 2 or iteration 4**.
- So refinement reliably finds higher-scoring reconstructions, even when the trajectory is noisy.

Reason: image generation is stochastic; selection and best-so-far make the loop robust.

CLIPSim over iterations



- Early improvement is strong; later iterations often saturate.

Qualitative result: Living room

Initial image

```
INFO: [4] Iteration 1/4
INFO: HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
INFO:   Best prompt idx=0 CLIP(text,img)=0.2942
INFO:   [dall-e-3] Generating 4 images for iter 1 prompt (len=533)
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO:   Best image score this iter: 0.7559 (sample 2, file=4_iter01_sample03.png)
INFO:   [IMPROVED] New global best score: 0.7559
INFO: [4] Iteration 2/4
INFO: HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
INFO:   Best prompt idx=2 CLIP(text,img)=0.3059
INFO:   [dall-e-3] Generating 4 images for iter 2 prompt (len=498)
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO:   Best image score this iter: 0.7622 (sample 0, file=4_iter02_sample01.png)
INFO:   [IMPROVED] New global best score: 0.7622
INFO: [4] Iteration 3/4
INFO: HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
INFO:   Best prompt idx=1 CLIP(text,img)=0.3042
INFO:   [dall-e-3] Generating 4 images for iter 3 prompt (len=500)
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO:   Best image score this iter: 0.8218 (sample 1, file=4_iter03_sample02.png)
INFO:   [IMPROVED] New global best score: 0.8218
```

Iter 1 CLIP Score: 0.7559

Iter 2 CLIP Score: 0.7622

Iter 3 CLIP Score: 0.8218

Improved 8.72% !

- Best CLIPSim: $0.7559 \rightarrow 0.7622 \rightarrow 0.8218$
- Relative improvement: $+8.72\%$

Qualitative result: Playground

Initial image

```
INFO: HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
INFO:   Best prompt idx=0 CLIP(text,img)=0.3440
INFO:   [dall-e-3] Generating 4 images for iter 1 prompt (len=415)
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO:   Best image score this iter: 0.6812 (sample 0, file=5_iter01_sample01.png)
INFO:   [IMPROVED] New global best score: 0.6812
INFO: [5] Iteration 2/4
INFO: HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
INFO:   Best prompt idx=0 CLIP(text,img)=0.3442
INFO:   [dall-e-3] Generating 4 images for iter 2 prompt (len=431)
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO:   Best image score this iter: 0.7451 (sample 1, file=5_iter02_sample02.png)
INFO:   [IMPROVED] New global best score: 0.7451
INFO: [5] Iteration 3/4
INFO: HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
INFO:   Best prompt idx=0 CLIP(text,img)=0.3452
INFO:   [dall-e-3] Generating 4 images for iter 3 prompt (len=439)
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
INFO:   Best image score this iter: 0.7529 (sample 0, file=5_iter03_sample01.png)
INFO:   [IMPROVED] New global best score: 0.7529
```

Iter 1 CLIP Score: 0.6812

Iter 2 CLIP Score: 0.7451

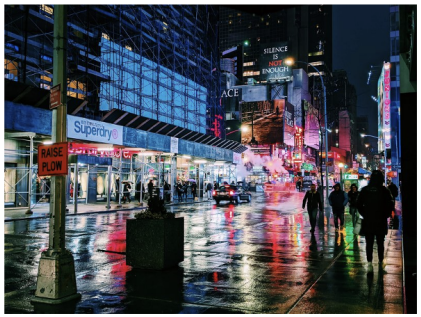
Iter 3 CLIP Score: 0.7529

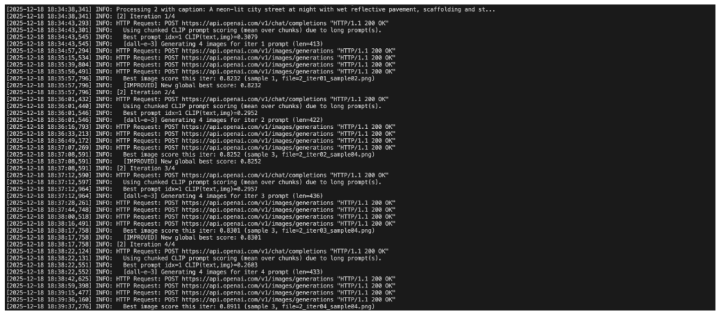
Improved 10.53% !

- Best CLIPSim: $0.6812 \rightarrow 0.7451 \rightarrow 0.7529$
- Relative improvement: $+10.53\%$

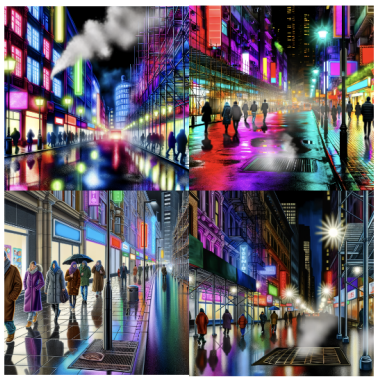
Qualitative result: City street

Initial image





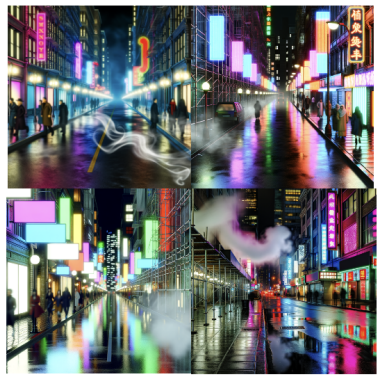
Iter 1 CLIP Score: 0.8232



Iter 2 CLIP Score: 0.8252

Iter 3 CLIP Score: 0.8301

Iter 4 CLIP Score: 0.8911



Improved 8.25% !

- Best CLIPSim: $0.8232 \rightarrow 0.8252 \rightarrow 0.8301 \rightarrow 0.8911$
- Relative improvement: $+8.25\%$

Qualitative result: Airport

Initial image

```
[2025-12-18 18:39:37,886] INFO: [3] Iteration 1/4
[2025-12-18 18:39:41,178] INFO: HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
[2025-12-18 18:39:41,185] INFO:   Using chunked CLIP prompt scoring (mean over chunks) due to long prompt(s).
[2025-12-18 18:39:41,296] INFO:   Best prompt idx=2 CLIP(text,img)=0.3149
[2025-12-18 18:39:41,296] INFO:   [dall-e-3] Generating 4 images for iter 1 prompt (len=453)
[2025-12-18 18:39:55,821] INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
[2025-12-18 18:40:12,000] INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
[2025-12-18 18:40:30,741] INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
[2025-12-18 18:40:45,791] INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
[2025-12-18 18:40:47,116] INFO:   Best image score this iter: 0.7856 (sample 3, file=3_iter01_sample04.png)
[2025-12-18 18:40:47,116] INFO:   [IMPROVED] New global best score: 0.7856
[2025-12-18 18:40:47,116] INFO: [3] Iteration 2/4
[2025-12-18 18:40:50,962] INFO: HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
[2025-12-18 18:40:50,968] INFO:   Using chunked CLIP prompt scoring (mean over chunks) due to long prompt(s).
[2025-12-18 18:40:51,072] INFO:   Best prompt idx=0 CLIP(text,img)=0.3159
[2025-12-18 18:40:51,072] INFO:   [dall-e-3] Generating 4 images for iter 2 prompt (len=402)
[2025-12-18 18:41:05,252] INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
[2025-12-18 18:41:19,977] INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
[2025-12-18 18:41:35,440] INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
[2025-12-18 18:41:51,428] INFO: HTTP Request: POST https://api.openai.com/v1/images/generations "HTTP/1.1 200 OK"
[2025-12-18 18:41:52,740] INFO:   Best image score this iter: 0.8257 (sample 3, file=3_iter02_sample04.png)
[2025-12-18 18:41:52,740] INFO:   [IMPROVED] New global best score: 0.8257
```

Iter 1 CLIP Score: 0.7856

Iter 2 CLIP Score: 0.8257

Improved 5.10% !

- Best CLIPSim: 0.7856 $\rightarrow$ 0.8257
- Relative improvement: $+5.10\%$

Error analysis

Common failure modes we observed:

- **Style drift:** sometimes outputs become too illustrative / cinematic.
- **Fine details:** wrong object counts, missing small objects, incorrect signage text.
- **Layout mismatch:** camera viewpoint and geometry differ (allowed semantically).

Why refinement helps

- DALL-E decoding is stochastic: one sample may miss key semantics.
- Our loop reduces variance by:
  - sampling M prompts and picking the best by CLIP(text,image),
  - sampling K images and picking the best by CLIP(image,image),
  - keeping a global best-so-far (hill-climb).
- Result: higher probability to find a better semantic match within few rounds.

Next steps

- Improve prompt quality *without increasing rate too much*:
  - more structured prompts (sections: scene / objects / lighting),
  - remove redundancy, keep only high-value details.
- If we want layout fidelity (not our current goal):
  - add lightweight side information (e.g., sketch as in Text+Sketch).
- Add more analysis:
  - rate–semantic-fidelity plots,
  - prompt length vs improvement tradeoff.

Conclusions

- We implemented a complete semantic compression pipeline:
  - prompt as ultra-low-rate code,
  - DALL-E 3 as decoder,
  - CLIPSim as semantic distortion,
  - GPT-4.1-mini as prompt refiner.
- On 20 images, we observe consistent improvements early, with representative relative gains of about 5–11%.
- The method is simple, reproducible, and matches our semantic definition.