Prompted Segmentation for Drywall QA

Shahzad

November 16, 2025

1 Introduction

This report describes a text-conditioned segmentation system for drywall quality assurance (QA). The goal is to automatically segment relevant regions in drywall images using natural-language prompts. We focus on two datasets:

- **Dataset 1: Taping area** – images of drywall panels with taping / joints.
- **Dataset 2: Cracks** – images of surface cracks on concrete / drywall.

Given an input image and a short prompt such as `"segment taping area"` or `"segment crack"`, the model predicts a binary mask highlighting the requested region. This enables promptable QA workflows where the same image can be re-used with different prompts (e.g., taping vs. seams, different crack types).

2 Approach

2.1 Model

I fine-tune the **CLIPSeg** model [1] using the `CIDAS/clipseg-rd64-refined` checkpoint from HuggingFace. CLIPSeg combines a CLIP-style vision–language encoder with a lightweight segmentation head. Given an image and a text prompt, it predicts a dense segmentation logits map.

Key choices:

- **Backbone:** CLIPSeg (ViT-based encoder + segmentation head).
- **Input resolution:** images are resized to 224×224 (CLIP standard).
- **Output:** segmentation logits at 224×224, upsampled back to the original resolution (e.g. 640×640).
- **Loss:** binary cross-entropy with logits between predicted mask and ground truth.

2.2 Prompts

Each dataset is associated with a small set of prompts:

- **Taping dataset:**
  - “segment taping area”
  - “segment joint/tape”
  - “segment drywall seam”

- **Cracks dataset:**
  - "segment crack"
  - "segment wall crack"

During training I use one *canonical prompt* per dataset (`segment taping area` for taping and `segment crack` for cracks). During evaluation and inference, I run the model for *all* prompts and report separate metrics.

2.3 Preprocessing

- Images are loaded at their original resolution (approximately 640×640) and resized to 224×224.
- Masks are generated from COCO annotations:
  - For the taping dataset, COCO bounding boxes are rasterized to filled rectangular masks.
  - For the cracks dataset, polygon segmentations are rasterized to binary masks.
- Both images and masks are normalized to $[0, 1]$ and jointly resized so that training is consistent.

3 Datasets and Splits

The assignment provides two Roboflow-based datasets:

- **Cracks Segmentation:** https://universe.roboflow.com/ay-7aga/cracks-segmentation/dataset/2
- **Drywall Joint Detect (taping):** https://universe.roboflow.com/objectdetect-pu6rn/drywall-join-detect

In my local setup the data is arranged as:

```
/media/sdb_access/Assignment/
  Dataset_1/ (taping)
      train/
      train_annotations.coco.json
      valid/
      valid_annotations.coco.json
  Dataset_2/ (cracks)
       train/
       train_annotations.coco.json
       valid/
       valid_annotations.coco.json
       test/
       test_annotations.coco.json
```

Table 1 summarizes the splits that are used by the code. (The counts can be obtained directly from the JSON files or via the Python dataloaders.)

Dataset	Train	Validation	Test
Taping (Dataset 1)	820	202	–
Cracks (Dataset 2)	1431	34	35

Table 1: Data split counts used in my experiments. Validation for the taping dataset doubles as test-time evaluation.

4 Training Setup

All experiments were run using the same script:

```
python -m src.main \
  --mode full \
  --dataset {cracks|taping} \
  --data_root /media/sdb_access/Assignment \
  --output_root ./outputs \
  --batch_size 4 \
  --image_size 224 \
  --epochs 50 \
  --lr 1e-5
```

Key hyperparameters:

- Optimizer: AdamW with learning rate 1×10^{-5}.
- Batch size: 4.
- Epochs: 50 (for final runs; the example logs below use smaller epochs for quick testing).
- Random seed is fixed for reproducibility.

The training loop uses the canonical prompt and computes loss over the resized masks. After each epoch, validation is run for *every* prompt of that dataset. The best checkpoint is selected by validation Dice score.

5 Metrics

- **Binary cross-entropy loss** (BCE with logits).
- **Intersection-over-Union (mIoU)** for foreground.
- **Dice coefficient** for foreground.

5.1 Cracks Dataset Results

Table 2 shows the final validation performance for the cracks dataset (`val_final_full` split from the logs).

5.2 Taping Dataset Results

Table 3 shows validation performance for the taping dataset (`val_eval` split).

These numbers are consistent with the qualitative impressions from the visualizations: taping and crack regions are generally captured well, while some prompts such as `segment drywall seam` are slightly harder.

Prompt	**Loss**	**mIoU**	**Dice**
segment crack	0.0891	0.5222	0.6111
segment wall crack	0.1038	0.5766	0.6561

Table 2: Per-prompt metrics for the cracks dataset (validation).

Prompt	**Loss**	**mIoU**	**Dice**
segment taping area	0.2020	0.5405	0.6834
segment joint/tape	0.3026	0.4088	0.5548
segment drywall seam	0.4626	0.3820	0.5178

Table 3: Per-prompt metrics for the taping dataset (validation).

6 Runtime and Model Footprint

6.1 Inference Timing

The code measures inference timing over all images and prompts.

Cracks dataset. For 35 test images and 2 prompts:

- Total inference time: 2.264 seconds.
- Average time per image per prompt: 0.0323 seconds.

Taping dataset. For 202 validation images and 3 prompts:

- Total inference time: 17.021 seconds.
- Average time per image per prompt: 0.0281 seconds.

The fine-tuned CLIPSeg model has roughly 150M parameters, corresponding to ~600 MB of trainable weights (in 32-bit floats). The code computes and logs the exact number of parameters and their memory footprint.

7 Qualitative Results

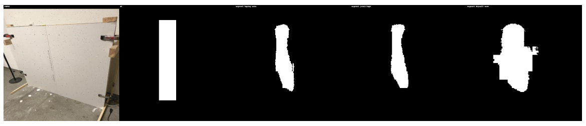
Figures 1 and 2 show representative qualitative results for both datasets. Each panel is a *stacked visualization* created by the code:

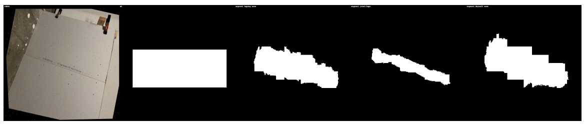
```
[Input] | [GT] | [Prompt 1 prediction] | [Prompt 2 prediction] | ...
```

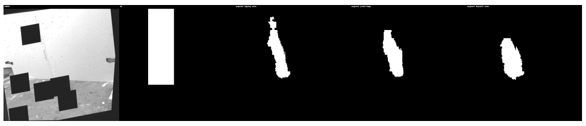
The prompt text is written along the top border of each mask tile.

8 Conclusion

I implemented and evaluated a text-conditioned segmentation system for drywall QA using CLIPSeg. The system supports multiple prompts per dataset, produces competitive mIoU / Dice scores on both cracks and taping tasks, and runs efficiently (tens of milliseconds per image per prompt).


(a) Taping sample 1.


(b) Taping sample 2.


(c) Taping sample 3.

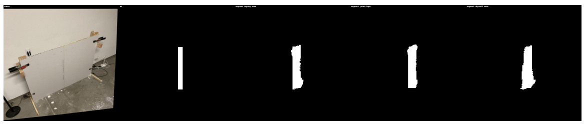
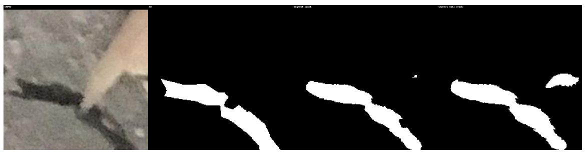
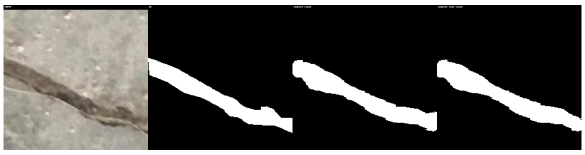
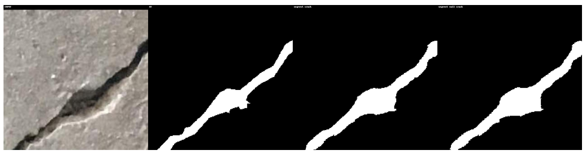

(d) Taping sample 4.

Figure 1: Qualitative results on the taping dataset. Each panel shows Input — GT — segment taping area — segment joint/tape — segment drywall seam.


(a) Crack sample 1.


(b) Crack sample 2.


(c) Crack sample 3.

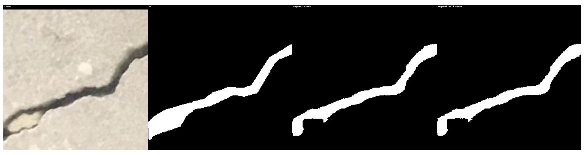

(d) Crack sample 4.

Figure 2: Qualitative results on the cracks dataset. Each panel shows Input — GT — segment crack — segment wall crack.

This approach is attractive because a *single* model can be reused for multiple QA tasks simply by changing the text prompt, without retraining. Future work could explore few-shot prompt tuning, better handling of ultra-thin structures, and joint training on both datasets to leverage shared visual cues.

Acknowledgements

The CLIPSeg model and codebase are built on top of the excellent open-source work by Lüddecke and Wörtwein [1], and the original CLIP architecture by Radford et al. [2].

References

[1] Timo Lüddecke and Theodor Wörtwein. CLIPSeg: Image Segmentation Using Text and Image Prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[2] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, and others. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning (ICML)*, 2021.