Scene Localization in Dense Images via Natural Language

Project Report

Author: Lakshay Pal
Date: August 15, 2025
Drive Link: Demo Vedio

Table of Contents

1. Introduction

In this project, I explored the application of state-of-the-art deep learning models for the task of scene localization in dense images using natural language. As visual data becomes increasingly ubiquitous, the ability to accurately identify and isolate specific events within a complex scene is of critical importance. My goal was to develop a robust inference pipeline by intelligently composing several powerful, pre-trained models and engineering custom modules to refine their output.

The project involved an iterative process of building, testing, and refining a sophisticated pipeline. The final model is a **GroundedSAM architecture with multiple custom modules for semantic refinement**, capable of processing free-form text queries to produce precise, segmented image crops as output. This report details the thought process, challenges overcome, and the final architecture of the prototype.

2. Thought Process & Initial Approach

I approached this project by first understanding the core challenge: bridging the semantic gap between unstructured text and pixel-based images. A successful system must not only detect objects but also understand the attributes, actions, and spatial relationships described in a query.

After an initial literature review, I recognized that:

- **Open-Vocabulary Detectors** like GroundingDINO are essential for handling the project's requirement of free-form text, moving beyond the limitations of fixed-class models.
- **Vision-Language Models** like CLIP possess a nuanced semantic understanding, making them ideal for re-ranking and validating the initial detections.
- **Spatial precision** was a critical requirement. Therefore, integrating a segmentation model like the **Segment Anything Model (SAM)** would be necessary to elevate the output from coarse bounding boxes to precise, object-level masks.
- A **multi-stage pipeline** would be more robust than a single end-to-end model, allowing for custom logic and quality gates at each step of the process.

This led to the choice of an architecture that composes these components, creating a pipeline that first proposes candidate regions, then refines them semantically, and finally segments them with high precision.

3. Final System Architecture

3.1 Pipeline Overview

The final system is a state-of-the-art GroundedSAM pipeline, which intelligently combines three powerful pre-trained models. The architecture is heavily influenced by the principle of **contrastive learning**, applying it at inference time to refine results.

3.2 Component Breakdown

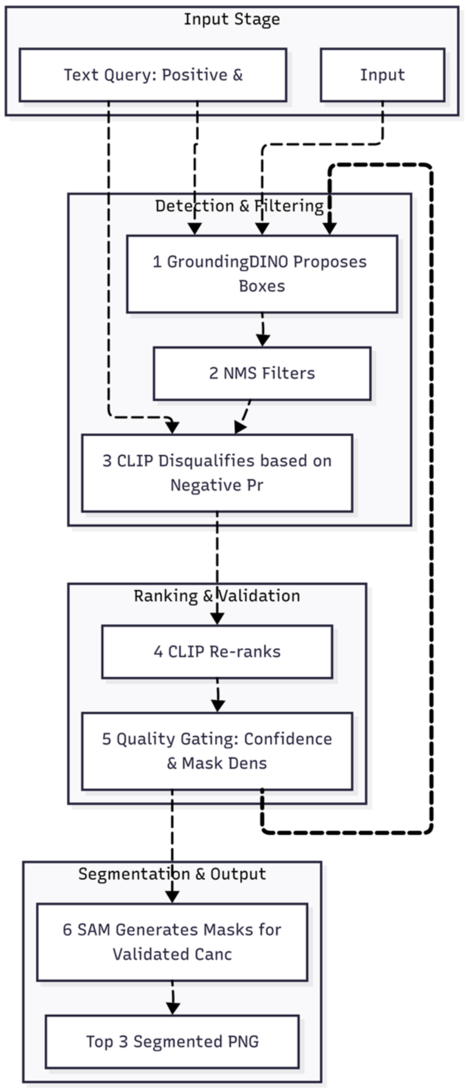
1. **Candidate Proposal (GroundingDINO):** The input image and text prompt are fed into a **GroundingDINO** model to identify potential bounding boxes. A Non-Maximum Suppression (NMS) step with an IoU threshold of 0.7 is applied to remove redundant, highly overlapping boxes.
2. **Semantic Disqualification (CLIP):** If a negative prompt is provided, this custom filter is activated. This module is a practical, inference-time application of the same principle behind training techniques like **Triplet Loss**. For each candidate, its semantic similarity to both the positive and negative prompts is calculated using **CLIP**. A candidate is **completely disqualified** if it matches the negative prompt more strongly than the positive one.
3. **Heuristic Re-ranking (DINO + CLIP):** All surviving candidates are ranked based on a combined_score (0.4 * dino_confidence + 0.6 * clip_score), balancing raw detection confidence with nuanced semantic understanding.

4. **Quality Gating & Final Selection:** The system iterates through the ranked list and applies two final gatekeeper filters:
   - **Confidence Gate:** A result is only accepted if its combined_score is above a minimum threshold (0.25) OR its raw dino_confidence is exceptionally high (0.90).
   - **Mask Density Gate:** The candidate box is passed to SAM. If the resulting mask is too sparse (less than 5% of its bounding box area), it is discarded as "pixel dust."

   The system collects the **first three candidates** that successfully pass all filters.

5. **Precise Segmentation (SAM):** For the final, validated candidates, the **Segment Anything Model (SAM)** is used to generate pixel-perfect, background-removed segmentation masks.

3.3 Architectural Flowchart


Input Stage
Text Query: Positive &
Input
Detection & Filtering
1 GroundingDINO Proposes Boxes
2 NMS Filters
3 CLIP Disqualifies based on Negative Pr
Ranking & Validation
4 CLIP Re-ranks
5 Quality Gating: Confidence & Mask Dens
Segmentation & Output
6 SAM Generates Masks for Validated Canc
Top 3 Segmented PNG

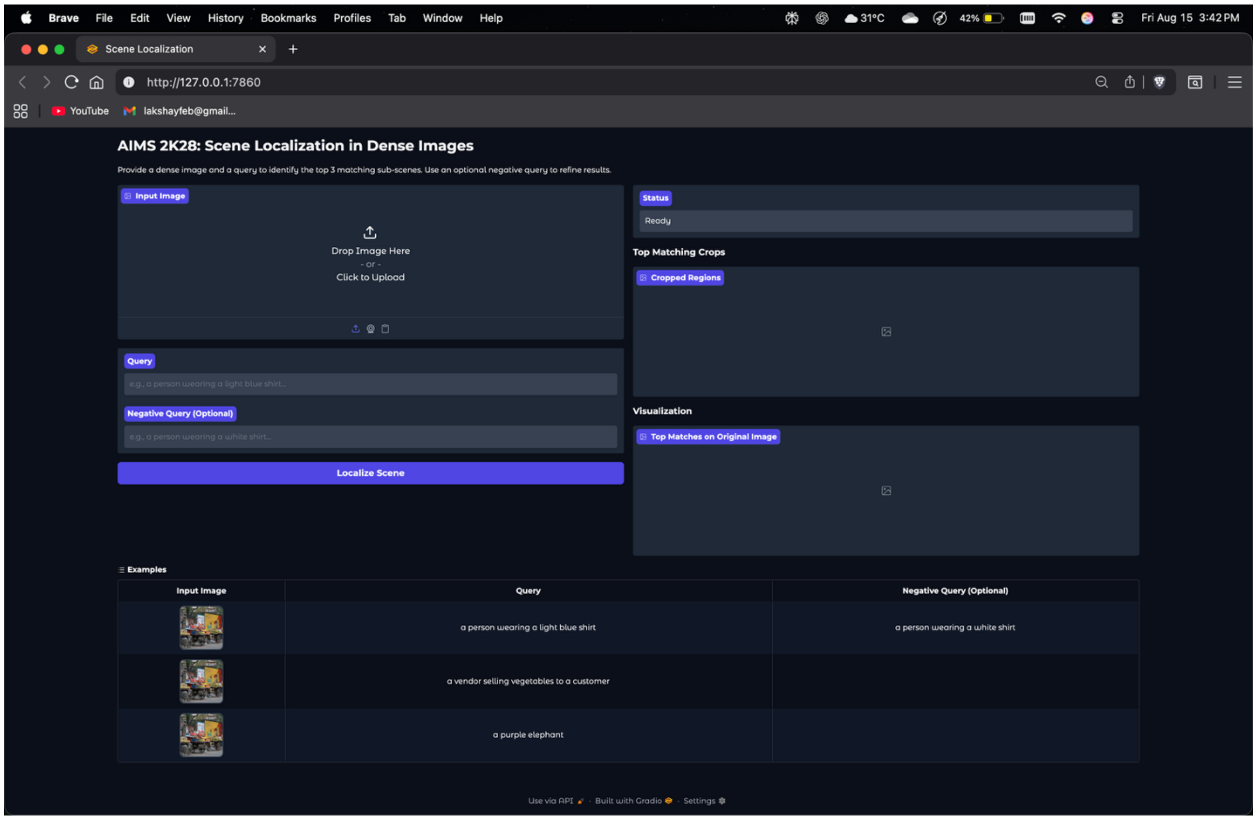
4. Development Journey: Blockers and Solutions

This project involved an iterative development process where several challenges were identified and overcome through systematic tuning and architectural improvements.

- **Blocker: Model "Hallucinations" on Non-Existent Objects.**
    - **Problem:** Initial versions of the pipeline would find low-confidence, nonsensical matches for queries like "a purple elephant."
    - **Solution:** A **Confidence Gate** was implemented. This dual-threshold check acts as a quality filter, rejecting results that are not sufficiently confident and ensuring the system fails gracefully.
- **Blocker: Imprecise or "Junk" Segmentations.**
    - **Problem:** When SAM was given a vague bounding box, it sometimes produced fragmented, "pixel dust" masks.
    - **Solution:** A **Mask Density Filter** was added. This custom module calculates the ratio of object pixels to the total bounding box area and discards any mask that is too sparse.
- **Blocker: Ambiguous Results and Poor Precision.**
    - **Problem:** The model struggled to differentiate between visually similar concepts, such as a "light blue shirt" versus a "white shirt" in tricky lighting.
    - **Solution:** The **Negative Prompting** feature was designed. The final, robust solution was a **disqualification filter**, which completely removes a candidate if it matches the negative query more strongly than the positive one.
- **Blocker: Redundant Detections.**
    - **Problem:** The model would sometimes detect the same object multiple times with slightly different bounding boxes.
    - **Solution:** The **NMS threshold** was tuned from a permissive 0.8 to a more aggressive 0.7, effectively merging duplicate detections.

5. The Interactive Prototype

A key deliverable was a working prototype. I developed an interactive web interface using the Gradio library. This UI allows a user to upload an image, input positive and negative queries, and view the ranked, segmented results alongside a visualization of the detections on the original image. A status bar provides clear feedback on the outcome of each query.



AIMS 2K28: Scene Localization in Dense Images

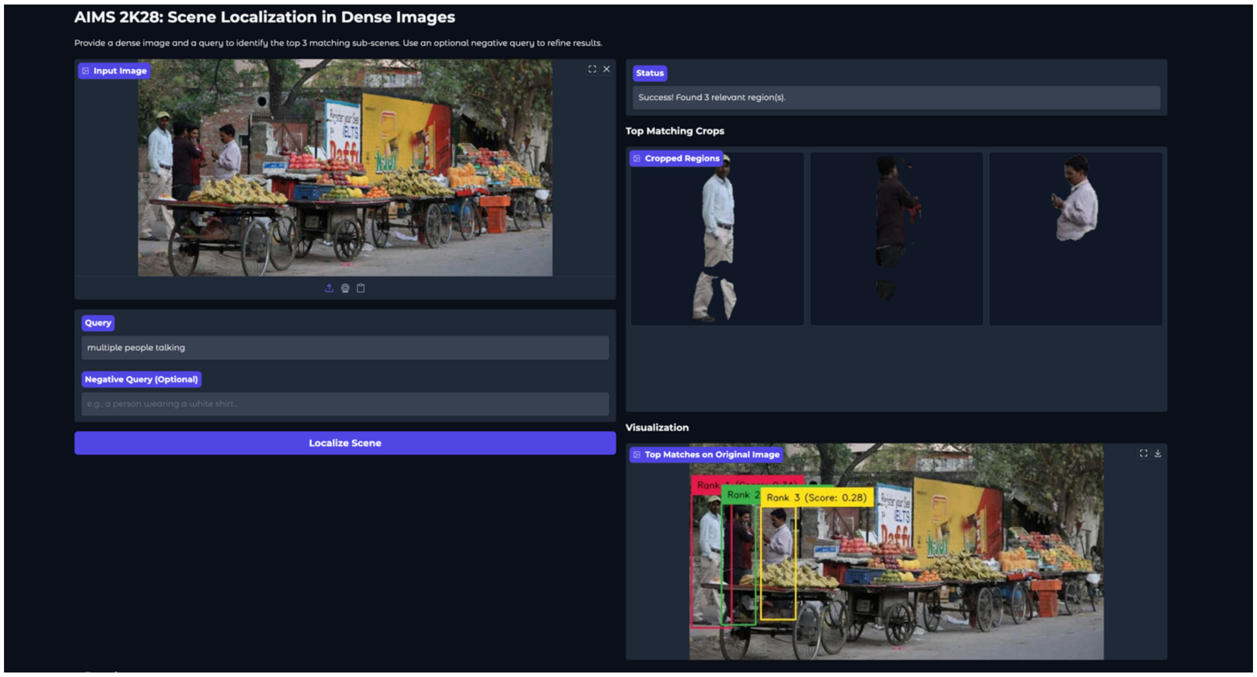
Provide a dense image and a query to identify the top 3 matching sub-scenes. Use an optional negative query to refine results.

Input Image	Query	Negative Query (Optional)
	a person wearing a light blue shirt	a person wearing a white shirt
	a vendor selling vegetables to a customer	
	a purple elephant	

6. Results & Examples

6.1 Example 1: Complex Scene Interpretation

- **Query:** multiple people talking
- **Result:** The system correctly identifies the primary interaction as Rank 1, while also identifying partially relevant scenes (the person standing , the person talking) as lower-ranked results, demonstrating a nuanced understanding.



AIMS 2K28: Scene Localization in Dense Images
Provide a dense image and a query to identify the top 3 matching sub-scenes. Use an optional negative query to refine results.
Input Image
Status
Top Matching Crops
Cropped Regions
Query
Negative Query (Optional)
Localize Scene
Visualization
Top Matches on Original Image
Rank 1
Rank 2
Rank 3 (Score: 0.28)

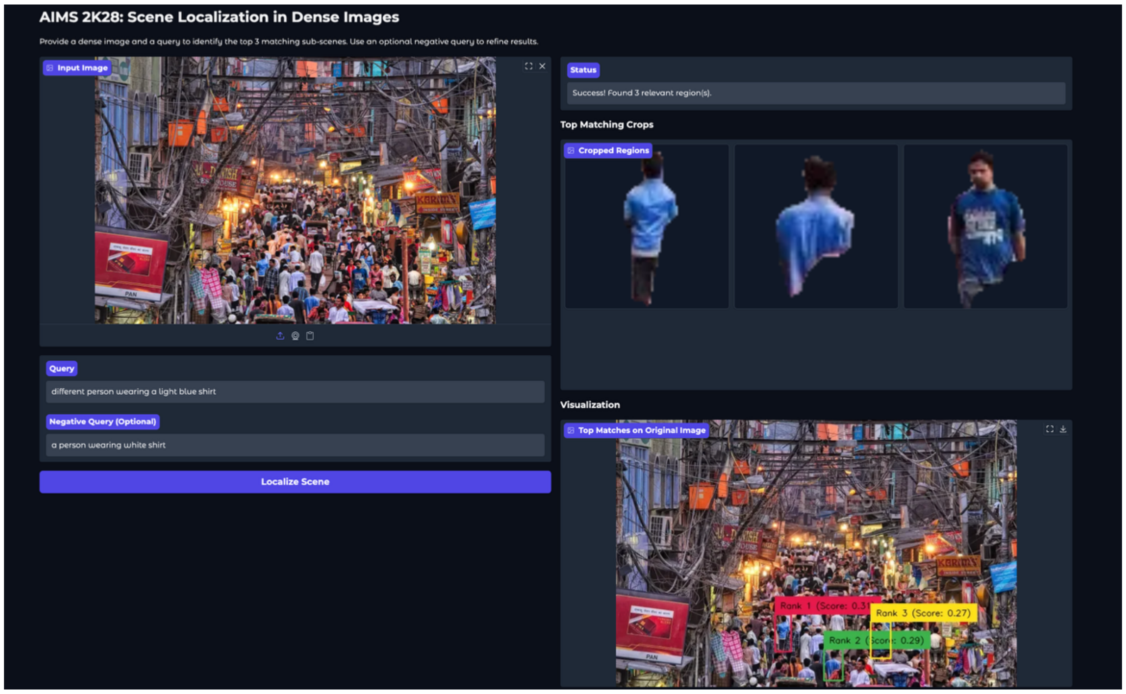
Success! Found 3 relevant region(s).

multiple people talking

e.g., a person wearing a white shirt...

6.2 Example 2: Semantic Disambiguation with Negative Prompting

- **Query:** a person wearing a light blue shirt
- **Negative Query:** a person wearing a white shirt
- **Result:** The negative query successfully disqualifies candidates wearing white, leading to a much more precise and accurate set of final results that correctly isolate people wearing blue.



AIMS 2K28: Scene Localization in Dense Images
Provide a dense image and a query to identify the top 3 matching sub-scenes. Use an optional negative query to refine results.
Input Image
Status
Top Matching Crops
Cropped Regions
Query
Negative Query (Optional)
Localize Scene
Visualization
Top Matches on Original Image
Rank 1 (Score: 0.31)
Rank 3 (Score: 0.27)
Rank 2 (Score: 0.29)

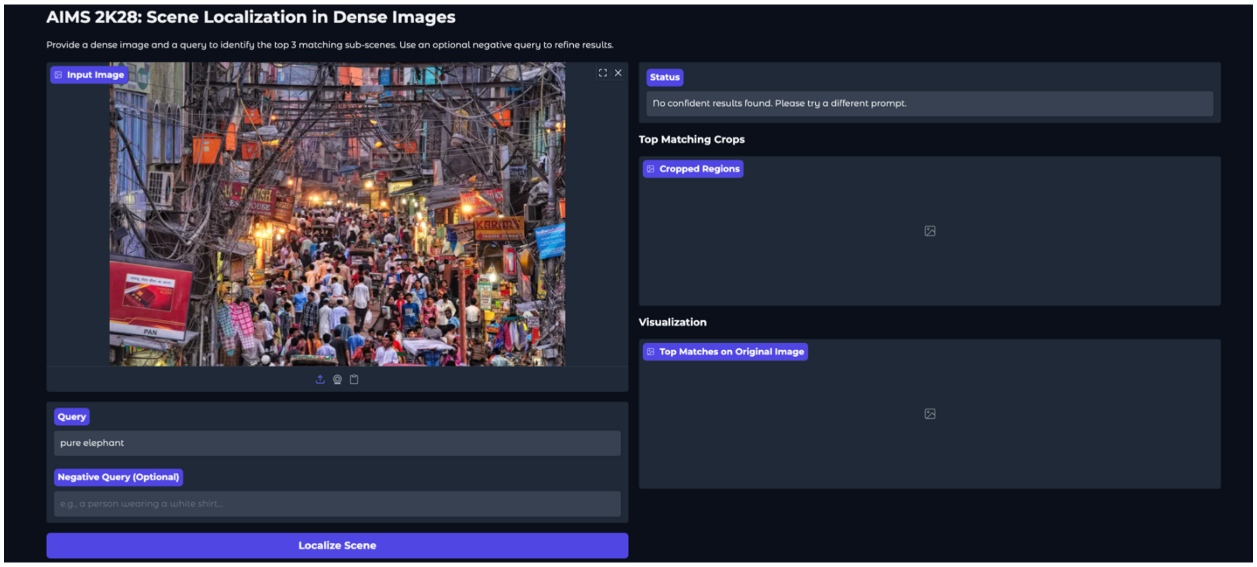
Success! Found 3 relevant region(s).

different person wearing a light blue shirt

a person wearing white shirt

6.3 Example 3: Robust Failure on Non-Existent Objects

- **Query:** a purple elephant
- **Result:** The system's quality gates correctly identify that all candidate detections have very low confidence and rejects them. The UI informs the user that no confident results were found.



AIMS 2K28: Scene Localization in Dense Images

Provide a dense image and a query to identify the top 3 matching sub-scenes. Use an optional negative query to refine results.

Status: No confident results found. Please try a different prompt.

Query: pure elephant

Negative Query (Optional): e.g., a person wearing a white shirt...

7. Future Work & Unimplemented Ideas

While the current prototype is a robust inference pipeline, several advanced phases were planned but not implemented due to the project's timeframe.

- **Quantitative Evaluation (mAP):** The logical next step is to establish a rigorous, quantitative benchmark for the system by creating a manually-labeled ground-truth test set and calculating the **Mean Average Precision (mAP)**score.
- **Model Fine-Tuning (Triplet Loss):** To address the model's core limitations, a full fine-tuning phase was planned. This would involve using the **Visual Genome** dataset to re-train the CLIP vision encoder with a **Triplet Loss**function, teaching the model a more robust, domain-specific semantic understanding. This requires significant cloud compute resources (GPU/TPU).

8. References

- GroundingDINO: Open-Set Object Detection with Language-Image Pre-training
- Segment Anything (SAM)
- CLIP: Learning Transferable Visual Models From Natural Language Supervision
- Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations