# Vision Language Frontier Maps for Zero-Shot Semantic Navigation

Resources

Article
Yokoyama, VLFM: Vision-Language Frontier Maps for Zero-Shot Semantic Navigation

GitHub
bdaiinstitute/vlfm

Abstract

Understanding how humans leverage semantic knowledge to navigate unfamiliar environments and decide where to explore next is pivotal for developing robots capable of human-like search behaviors. We introduce a zero-shot navigation approach, Vision-Language Frontier Maps (VLFM), which is inspired by human reasoning and designed to *navigate towards unseen semantic objects in novel environments*. VLFM **builds occupancy maps from depth observations** to identify frontiers, and leverages **RGB observations** and a pre-trained **vision-language model** to generate a language-grounded value map. VLFM then uses this map to identify the **most promising frontier** to explore for *finding an instance of a given target object category*. We evaluate VLFM in photo-realistic environments from the Gibson, Habitat-Matterport 3D (HM3D), and Matterport 3D (MP3D) datasets within the Habitat simulator. Remarkably, VLFM achieves state-of-the-art results on all three datasets as measured by success weighted by path length (SPL) for the Object Goal Navigation task. Furthermore, we show that VLFM's zero-shot nature enables it to be **readily deployed on real-world** robots such as the Boston Dynamics Spot mobile manipulation platform. We deploy VLFM on Spot and demonstrate its capability to efficiently navigate to target objects within an office building in the real world, without any prior knowledge of the environment. The accomplishments of VLFM

underscore the promising potential of vision-language models in advancing the field of semantic navigation.

# Introduction

## Human Navigation 🚶

- Human navigation in unfamiliar environmets involves complex processes:
  - Explicit maps
  - **Internal knowledge**: Accumulation of semantic knowledge. Used to infer-
    - Layout of a space
    - Locations of specific objects
    - Geometric configurations

- **e.g.** We know toilets and showers are located together in the bathrooms, which are often located near bedrooms.
- Natural language can be used to enhance this semantic knowledge.

## Robots Capable of Human-Like Navigation 🤖

Foundation models which can learn to mimic this **human reasoning process** are invaluable.

### Zero-Shot Methods

- Zero-Shot Methods 🔗 utilize these models to facilitate semantic navigation *wihout any task-specific training or fine-tuning*.
- Zero shot methods are convenient because:
  - **Easily adaptable** for new tasks.
  - Repurposed for **future robotics systems** performing complex tasks.

- Provide **intermediate representations** which improves *interpretability*.

> The remarkable performance of large language models (LLMs) and vision-language models (VLMs) has facilitated task-independent solutions for the *semantic inference of out-of-view scene information*.

VLFM-Zero-Shot-Semantic-Navigation, page 1

## Proposed Idea 💡

**Vision Language Frontier Maps**

- It is a **zero-shot** approach for **target-driven semantic navigation**
- The **goal** is to navigate to an *unseen object* in a *novel environment*.

### Occupancy Maps & Frontier Identification

- VLFM builds occupancy maps from **depth observations** to identify **frontiers** of the *explored map region*.
- To find the **semantic target objects**, VLFM **prompts a pre-trained VLM** *"Which of these frontiers is most likely to lead to the semantic target?"*

## Difference from Previous Methods

- In contrast to prior language-base zero-shot semantic navigation methods, VLFM **does not rely on object detectors and language models** to evaluate frontiers using *text-only semantic reasoning*.
- VLFM uses a **vision-language model** to directly extract semantic values from RGB images in the form of a *cosine similarity score* with a text-prompt involving the target object.

- These scores are used to create a **language-ground value map**, which is used to identify the most promising frontier to explore.

> This spatially- grounded joint vision-language-based semantic reasoning increases computational inference speed and overall semantic navigation performance.

[VLFM-Zero-Shot-Semantic-Navigation, page 2]

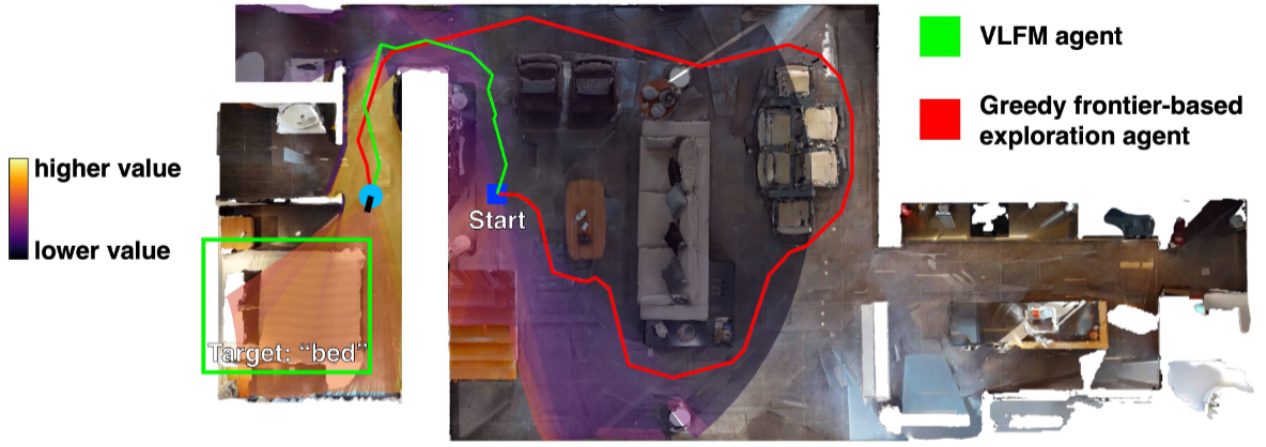


Fig. 1: VLFM achieves state-of-the-art semantic Object Goal Navigation performance in unfamiliar environments, without task-specific training, pre-built maps, or prior knowledge of the surroundings. It utilizes a vision-language model to explore the environment, capitalizing on visual semantic cues that are likely to guide the agent towards the goal to explore the environment more efficiently than a greedy frontier-based exploration agent.

# Methodology 🛠️

## Problem Formulation 🤔

### Task

**ObjectNav** - Robot is tasked with searching an instance of a *target object category* in a *previously unseen environment*.

### High-Level Understanding 🧠

This approach encourages the robot to understand the environment on a higher level, semantically - *type of room?* based on the objects seen; rather than solely relying on geometric cues.

## Sensors 📷

1. Egocentric RGB-D camera.
2. Odometry sensor providing current `(x, y)` and `heading` relative to starting pose.

## Action Space

1. `move_forward(0.25)`
2. `turn_left(30)`
3. `turn_right(30)`
4. `look_up(30)`
5. `look_down(30)`
6. `stop()`

> ✓ **Success**
>
> An episode is defined to be completed sucessfully ✅ if `stop()` is called within $1\,m$ of the target object in $\leq 500$ steps.

## The Algorithm

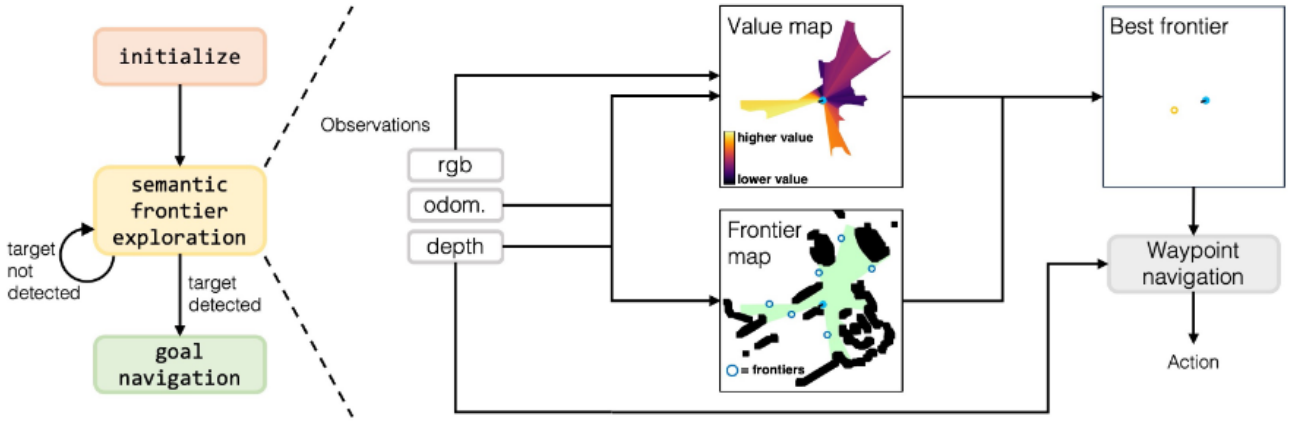


Fig. 2: VLFM constructs an occupancy map of the scene identifying frontiers of explored space as well as a value map of the likelihood of each region to lead toward the out-of-view target object. In a navigation episode, the robot first spins in a circle to initialize these maps and then begins executing frontier-based exploration by selecting waypoints from the current map frontier using the value map. Navigation to each waypoint is executed with a PointNav policy trained with Variable Experience Rollout (VER) [9]. The policy is also used to navigate to the target object once it is detected ('goal navigation').

## Frontier Waypoint Generation

1. Convert depth image to point cloud
2. Classify points as obstacle or not obstacle
3. Transform all points to global frame (based on current odometry), then project onto a 2D grid
4. Identify each boundary separating the *explored* and *unexplored* areas, identify its **midpoint** as a potential frontier waypoint.

### Frontiers and Exploration

- As the robot explores the area, the positions and quantity of these frontier wayopints will vary until the *entire environment* has been explored, at which points there will be *no more frontiers*.
- If the target object *has not been found* at this point, the robot simply triggers `stop()`, ending the episode unsuccessfully ☹️

## Value Map Generation 🗺️

- Value map is a 2D grid similar to the frontier map.
- This map assigns a *value to each cell within the explored area*, quantifying the **semantic relevance/similarity** to the target object.
- The value map is used to *evaluate each frontier*, and the *highest value* is chosed as the next waypoint to explore.

### Comparison to Frontier Map

| Frontier Map | Value Map |
| --- | --- |
| Uses depth and odometry values to build a top-down map iteratively | Same |
| Has a single channel representing the "obstacle"-ness of the cell in the grid | Has **two channels**, representing the *semantic value score* and *confidence score* |

## Achieveing Human-Like Reasoning 🚶

- Humans derive cues directly from 👁 *visual observation* (lighting, room type, room size, navigability to other rooms, etc.), rather than attempting to first convert current visuals to text 📄
- Here, a pre-trained vision language model (BLIP-2) is used to obtain a **cosine similarity score** from the robot's *current RGB observation* and a *text-prompt* containing information about the *target object*.

## Semantic Value Channel

> 🖼 + 📄 (*"Seems like there is a* `target_object` *ahead"*)
> ↓
> BLIP-2
> ↓
> *Similarity (to 📄) score* for each pixel in 🖼

## Confidence Channel

> The confidence channel aims to determine how a pixel's value in the *semantic value channel should be updated* if it has a value assigned from a *previous time step* and is within the robot's field-of-view (FOV) at the current time step

VLFM-Zero-Shot-Semantic-Navigation, page 3

- Value of pixel in semantic value channel not affect if the given pixel was not seen until the current time step
- Confidence of pixels along the optical axis have a value of $1$ and those at the left and right edges of the image have $0$.

$$c_{\text{pixel}} = \cos^2\left(\frac{\theta}{\theta_{\text{fov}}/2} \cdot \frac{\pi}{2}\right)$$

where

- $\theta$ is the angle between the *pixel and the optical axis*

- $\theta_{\text{fov}}$ is the *horizontal FOV of the robot's camera*


Fig. 3: *Left:* Visualization of how the confidence score of a pixel within the robot's FOV is determined based on its location relative to the optical axis. *Right:* The confidence scores are used when the robot's current FOV overlaps with the previously seen area; the new semantic values within this region become an average of the previous and current values, weighted by their confidence scores.

## Using both the Channels

### Update Semantic Value Channel $v_{i,j}^{\text{new}}$

$$v_{i,j}^{\text{new}} = \frac{c_{i,j}^{\text{curr}} v_{i,j}^{\text{curr}} + c_{i,j}^{\text{prev}} v_{i,j}^{\text{prev}}}{c_{i,j}^{\text{curr}} + c_{i,j}^{\text{prev}}}$$

### Update Confidence Channel $c_{i,j}^{\text{new}}$

$$c_{i,j}^{\text{new}} = \frac{(c_{i,j}^{\text{curr}})^2 + (c_{i,j}^{\text{prev}})^2}{c_{i,j}^{\text{curr}} + c_{i,j}^{\text{prev}}}$$

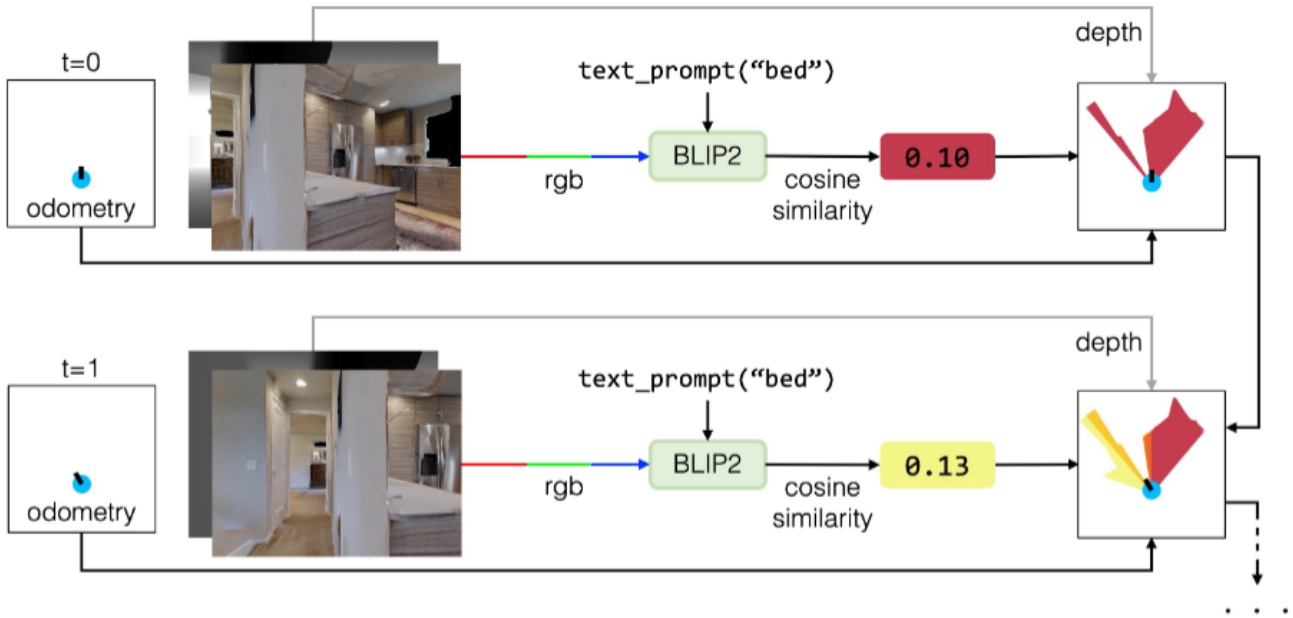


Fig. 4: VLFM iteratively constructs value maps for target-driven navigation by using BLIP-2 to compute the cosine similarity between a text prompt incorporating the target object and an RGB image taken from the robot's current pose. These semantic value scores are projected onto a top-down 2D pixel grid in the shape of the camera's FOV and exclude regions occluded by obstacles captured in the depth image.

## Object Detection ⬚

Different types of pre-trained object detectors which infer bounding boxes with semantic labels are used:

1. YOLO: For target objects that fall within the COCO dataset.
2. Grounding-DINO: For all other categories. IDEA-Research/GroundingDINO
3. Mobile-SAM: Once the target object is detected, the contours are extracted from the bounding box using the Mobile-SAM model.

> ✓ Target Waypoint Navigation
>
> - This contour along with the **depth image** is used to point out the object that is closest to the robot's current position (there may be *many instances* of the target object in the environment).
> - Once the robot closes in to a *distance less than the success radius*, it calls `stop()`, ending the episode successfully ✅

## Waypoint Navigation 🚗

TLDR;

> Our preference for the **PointNav** policy stems from its speed and ease-of- use; it alleviates several concerns associated with traditional mapping-based approaches, particularly when the waypoint resides outside the navigable area (e.g., when the waypoint is on a target object, which itself may also be on a different obstacle), as navigability of the goal does not affect the policy or its observations.

VLFM-Zero-Shot-Semantic-Navigation, page 5