Supplementary Materials for *V-IRL*: Grounding Virtual Intelligence in Real Life

Jihan Yang[1]* Runyu Ding[1] Ellis Brown[2] Xiaojuan Qi[1] Saining Xie[2]
[1]The University of Hong Kong [2]New York University
https://virl-platform.github.io

A Appendix Outline

In these supplementary materials, we provide additional details for our V-*IRL* platform, including:

- Additional V-*IRL* agents results (Appendix B);
- Technical designs behind V-*IRL* Agents (Appendix C);
- Technical details and challenges in the V-*IRL* environment (Appendix D).
- A low-level case study of Intentional Explorer agent Hiro, delving into implementation details of our system such as LLM prompts (Appendix F);
- More detailed setups and results for our V-*IRL* benchmarks (Appendix G).
- Discussion on ethical and data privacy issues for V-*IRL* (Appendix H).
- Conclusion of V-*IRL* (Appendix I).

B Additional V-IRL Agents

Here we present additional agents that further illustrate the flexibility of the V-*IRL* platform beyond those displayed in Sec. 3.

B.1 RX-399: Urban Assistance Robot

Urban Assistance Robot ENV Map CV Vision

Name: RX-399 **Age:** Unk. **Loc:** HK/NYC
Bio: This urban robot's advanced object detection, localization, and navigational telemetry systems allow it to perform perceptive tasks in busy city streets.
Intention: Report the locations of trash bins to the sanitation dept.

Task: Travel along a specified route and detect instances of a specified object (e.g., trash bins, hydrants, benches, etc.).

Takeaway: V-*IRL* agents can use perceptive input to understand and interact with their environment.

RX-399 is a state-of-the-art robot agent with advanced navigation and sensing capabilities. Its manufacturer is running a pilot program with sanitation departments in Hong Kong and New York City to assess its readiness for garbage duty...

* Work conducted during a visit to NYU.

RX-399 navigates along pre-defined city routes, tagging all trash bins using its open-world detector and geolocation module as depicted in Fig. 1. RX-399 can actively adjust its camera pose to the optimal view for each potential object thanks to our interactive embodied environment and the sensor-rich visual input. *During the pilot in Hong Kong, RX-399 locates eight trash bins, correctly identifying five but overlooking one. In New York, it accurately detects all five trash bins but mistakenly reports two mailboxes.*

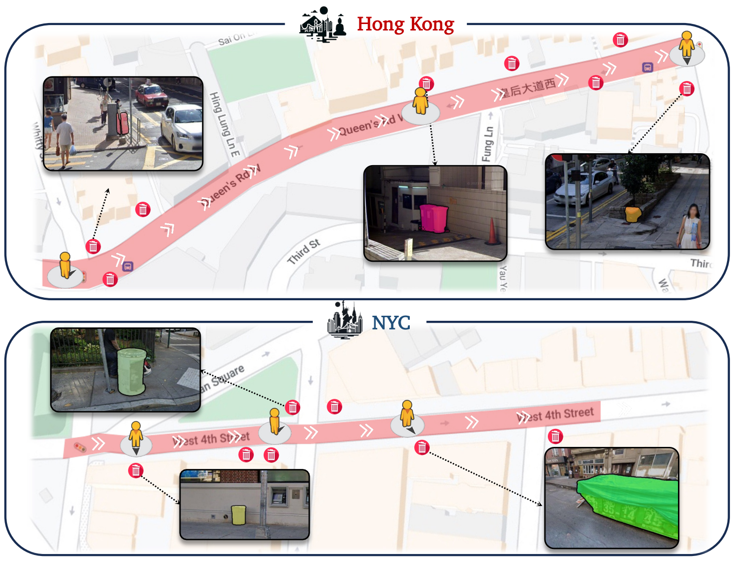


Fig. 1: Portions of RX-399's system records in HK and NYC.

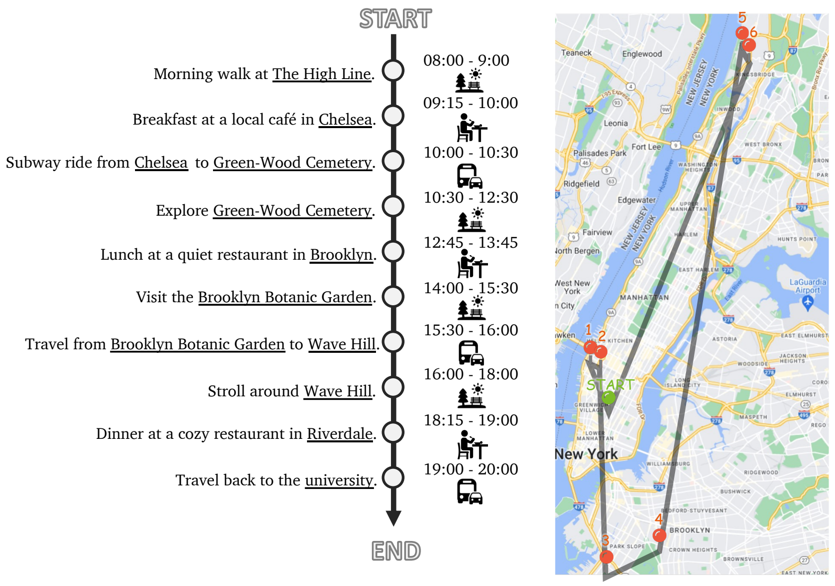
B.2 Ungrounded LLM-only Concierge

In contrast to the expert concierge exemplar agent "Diego" in Sec. 3, Fig. 2 shows that a simpler "ungrounded" LLM-only concierge agent is unable to consider the real distance and travel time between locations without access to V-*IRL*, resulting in an impractical itinerary. For example, lacking real geospatial information, the ungrounded concierge allocates only *30 minutes* for travel between the "Brooklyn Botanic Garden" and "Wave Hill" in the Bronx, which actually requires *60–100 minutes*[1]. The hallucinated travel times overlook geospatial realities and result in a plan with excessively distant destinations.

C Technical Details of V-*IRL* Agents

In Sec. 3, our discussion mainly focuses on the innovative capabilities and behaviors of V-*IRL* agents empowered by our platform. We avoid in-depth discussions about technical details in the main paper due to the concern of readability. In

[1] (per Google Maps https://maps.app.goo.gl/SW1r5GSx3ZVo7BTr7).



START
Morning walk at The High Line.
08:00 - 9:00
Breakfast at a local café in Chelsea.
09:15 - 10:00
Subway ride from Chelsea to Green-Wood Cemetery.
10:00 - 10:30
Explore Green-Wood Cemetery.
10:30 - 12:30
Lunch at a quiet restaurant in Brooklyn.
12:45 - 13:45
Visit the Brooklyn Botanic Garden.
14:00 - 15:30
Travel from Brooklyn Botanic Garden to Wave Hill.
15:30 - 16:00
Stroll around Wave Hill.
16:00 - 18:00
Dinner at a cozy restaurant in Riverdale.
18:15 - 19:00
Travel back to the university.
19:00 - 20:00
END
New York

Fig. 2: An ungrounded LLM-only concierge agent's itinerary.

this section, we go through our main technical designs for each agent. More comprehensive technical implementations are available in our released code.

C.1 Peng: Route Optimizer

Peng is designed to showcase the utilization of real geographic coordinates within our platform. By processing a sequence of real addresses, Peng calculates the *shortest path* for traversing them using various modes of transportation, such as walking, driving, and bicycling, among others. This capability is powered by the mapping module described in Appendix D.3. After that, Peng proceeds to navigate through the destinations along the predetermined path, employing the point navigation procedure outlined in Appendix D.2.

C.2 Aria: Place Recommender

Aria leverages the realistic place information provided by our Place Info & Search module (see Appendix D.4) to enhance LLMs' reasoning capability in the geographic aspect. Specifically, Aria evaluates Peng's intention to determine the suitable type of place and searches all possible places in the vicinity. For each searched place, Aria considers its reviews and user ratings from Google to summarize a place overview. Subsequently, we customize prompts for Aria to amalgamate Peng's biography, intentions, and the summarized place overviews to rate each place between 0 and 10, accompanied by justifications.

Without such technical designs, LLMs could recommend some places that are either too distant or permanently closed. This issue arises because LLMs struggle to accurately understand geospatial relationships and often depend on outdated databases.

C.3 Vivek: Estate Agent

The process employed by Vivek is similar to that of Aria, as both are designed to recommend places. However, Vivek showcases the versatility of our V-*IRL* platform by demonstrating how it can seamlessly integrate a wide range of realistic information beyond the Google Maps Platform, with a standardized definition of geographic coordinates. This capability enables the creation of even more sophisticated and intriguing agents.

C.4 RX-399: Urban Assistance Robot

Different from previous example agents, RX-399 introduces visual perception capabilities such as open-world detection and feature matching. There are two fundamental systems inside it – navigation and perception. In terms of navigation, RX-399 can automatically navigate from the current position to the pre-defined destination step by step. This navigation process is elaborated in Appendix D.2, and thus, will not be extensively discussed here.

When it comes to its perception system, RX-399 is designed to simulate human visual perception by capturing street views within a 90-degree horizontal angle to both its left and right. For each captured view, an open-world detection process is conducted. Leveraging the interactive capabilities of our environment, we further propose an *active detection* strategy to dynamically adjust the agent's ego-pose and focal length according to the scale and position of potential objects. This significantly improves its performance as illustrated in Tab. 1. In the future, more advanced approaches such as visual search [17] could also be considered. In the subsequent de-duplication procedure, which aims to avoid double-counting objects across different viewpoints, we have tried a few strategies including measuring with multi-view geometry, object tracking, and feature matching. We choose feature matching because of its accuracy and efficiency.

City	Hong Kong	New York City
w/ active detection	**0.63** / **0.83**	**0.71** / **1.00**
w/o active detection	0.10 / 0.33	0.30 / 0.60

Table 1: RX-399 detection performance with or without active detection manner. Metrics are accuracy / recall.

C.5 Imani: Urban Planner

The visual perception system of Imani mirrors that of RX-399. The primary distinction between them lies in their navigation systems. Imani possesses the capability to plan a navigation route in the given polygonal region, enabling RX-399 to traverse that region. This functionality is named "region navigation" and elaborated in Appendix D.2. Additionally, within the Imani agent, we develop a heatmap visualization tool to visualize and verify the data collected by RX-399 (see Fig.3 of the main paper).

C.6 Hiro: Intentional Explorer

Hiro is a representative agent equipped with geographical, perceptual, and reasoning abilities, to address a daily human task: randomly exploring to find a suitable restaurant. In this regard, we have dedicated a separate section to offer an in-depth case study, including the detailed methodology and prompts in Appendix F.

C.7 Ling: Tourist

Our vision language navigation pipeline of Ling is similar to [16], leveraging vision models, the map, and LLMs. At each position, we start by capturing eight street views around the agent, corresponding to `front`, `left front`, `left`, `left behind`, `behind`, `right behind`, `right` and `right front`. Vision models use these street views to identify landmarks mentioned in route descriptions, which are then verbalized as *landmark observations*. Also, intersection information is retrieved from the mover to formulate an *intersection observation*. LLMs play a crucial role in processing landmark & intersection observations along with the agent's previous working history to determine the next action. After each action, current observations and actions are stored into the agent's working history. This auto-regressive process continues until the agent decides to `stop`.

C.8 Local Agent

The primary mission of the Local agent is to generate human-like and easily followable navigation instructions on a global scale (refer to Sec. 3.4). This task is known as navigation instruction generation [15]. Contrary to most existing research, which depends on human-annotated data for limited geographic areas, our "Local" agent automatically selects suitable landmarks taking account into real-world places and generates human-like route descriptions using LLMs across the globe. Remarkably, it achieves this without the need for any training data, relying solely on our tailored prompts and a few in-context examples. The effectiveness of its generated instructions has been verified through collaboration with "Ling". To the best of our knowledge, this is a first in the field. There are massive technical details on selecting easily noticeable landmarks and prompt engineering, which are available in our released code.

C.9 Diego: Interactive Concierge

In Appendix E, we have already presented the technical designs of Diego's itinerary. Here, we detail how Diego can find scenic locations as shown in Fig. 8 of the main paper. For any given destination, such as "Fort Tryon Park", Diego will sample a rectangle region around it and traverse all navigable positions within it. At each position, he will capture a photograph (*i.e.* street view imagery) using pre-defined headings, pitches, and FOVs. Each photograph will then be evaluated using GPT-4(V) [2], where it receives a rating between 0 and 10 along with explanatory reasons.

D Technical Details of Environment

In Sec. 4.2, we provide an overview of our system's environment, which grounds agents in real life. Here, we delve into the technical designs beyond mere leveraging Google Map Platform system calls. Concrete implementations can be found in our open-sourced code.

D.1 Geolocation & Street View Imagery

At the core of V-*IRL* lies its innovative use of sensor-rich environment, including street view imagery and geolocations. They enable agents to gather surrounding place and vision information.

Geolocation. Agents in the V-*IRL* platform inhabit virtual representations of real cities around the globe. At the core of this representation are geographic coordinates (*i.e.* geolocation) corresponding to points on the Earth's surface. The initial geolocation of each agent is specified by its "Location" configuration, as illustrated in Fig. 9 of the main paper. Whenever agents require access to surrounding information (*e.g.* street views, places or maps), geolocation serves as a crucial parameter for querying the related Google Map APIs.

Street view imagery. Google Map Platform specifies each street view imagery with multiple key parameters: geolocation, heading (the horizontal angle ranging from $0°$ to $360°$), pitch (a vertical angle ranging from $-90°$ to $90°$), and Field of View (FOV, ranging from $20 \sim 120$). It's noteworthy that adjusting the FOV here is similar to changing the camera's focal length, rather than simply zooming in on an image, which ensures that image resolution remains high, even as the FOV decreases to a low value. By modifying the heading, pitch, and FOV, we can simulate the human sensory process of adjusting one's pose and concentrating on a specific area.

Alignment between street view imagery and geolocation. Within our sensor-rich platform, a fundamental challenge is to ensure agents are positioned at geolocations where street view imagery is available. To address this issue, we design a custom operation named "***relocate***". Specifically, when an agent is initialized at a location lacking street view imagery, the "relocate" operation will identify and transition the agent to the nearest viable geolocation where street view data is available. Notice that, this operation is indispensable to our platform, as the positions with available street views are relatively sparse in comparison to the vast continuous space of all possible coordinates.

D.2 Movement

Enabling agents to move along city streets is a core functionality of our platform, allowing interaction between agents and the real world. Whenever an agent needs to move, this module powers all related processes, from route planning and direction selection to the continuous update of the agent's geolocation during its moving. Since Google Maps Platform does not provide APIs to access nearby navigable positions and directions, the design of this movement module is a significant technical challenge and a substantial contribution from our team. We discuss its low-level implementations in Appendix D.2 and the enabled high-level actions in Appendix D.2.

Mover Move by controlling the web interface. A straightforward solution is to let the agent control the web front-end Google Street View to select moving directions and move. Nevertheless, there are three key challenges for this solution:

(*i*) ***How can Python-implemented agents control the movement via the interaction to the webpage?*** We use a Python package Selenium[2] to locate web elements responsible for movement. After determining a movement direction, the agent uses Selenium to simulate a click action on the web element corresponding to the chosen direction.

(*ii*) ***How can the agent acquire the necessary information to decide moving direction?*** Although agents can access all potential movement directions from web elements, they cannot identify these directions without prior knowledge of what each represents. We find that the "transform" attribute in the web element corresponding to each direction can be leveraged to calculate their represented heading angles. The heading angle also allows us to collect street view imagery for each movement direction. Agent's movement decision-making is then based on these heading angles and the visual data from street view imagery.

(*iii*) ***How to track the agent's geolocation along its movement?*** To accomplish this, we customize a webpage element to display the geolocation of the current street view panorama. As the agents move and trigger updates to the street view panorama, this customized element concurrently refreshes to reflect the new geolocation. By using Selenium, we can then extract this updated geolocation data, enabling continuous tracking of the agent's geolocation changes.

Move by grid-based relocating. In our test of the above web-based mover, a critical limitation emerged: the web-embedded Google Street View panoramas display only a subset of navigable directions. This constraint significantly restricts our agents' mobility, often preventing them from successfully navigating to their intended destinations due to the incomplete coverage of potential routes.

To overcome this obstacle, we develop an alternative method: a grid-based relocating mover. This approach involves performing a grid search for geolocations in the vicinity of the agent and employing the "relocate" operation to sift through these locations, identifying those that are navigable. While this method

[2] https://www.selenium.dev/

offers a more comprehensive view of navigable positions, it is markedly more time-consuming than the web-based approach due to the extensive number of Google Maps API calls required.

In our practical applications, we design a heuristic strategy that combines web-based controlling and grid-based relocation. This hybrid approach aims to balance the trade-offs between the speed and the completeness of navigable position data, optimizing our agents' capabilities and efficiency in real-world scenarios.

Navigator Here, we introduce the high-level action of agents powered by the mover – navigation. Unlike the mover, which concentrates on enabling agent mobility in the environment, the focus here shifts to determining the direction of movement. In our platform, we group different navigators according to their usages into four types:

(*i*) **Point navigator** is designed to tackle navigation tasks that clearly define single or multiple destinations (represented in addresses or geolocations). This navigator employs the route planning function described in Appendix D.3 to obtain a series of key positions for navigation. At each location, the agent utilizes a greedy algorithm to select the most optimal direction towards the next key position that has not yet been reached. Exemplary agents, such as "Peng", "RX-399" and "Local", use this type of navigator in their implementation.

(*ii*) **Region navigator** is tailored for agents like "Imani" and "Diego", who need to traverse every position within a polygonal region. This navigator first employs a grid search combined with our "relocate" operation to identify all navigable positions within the specified region. Subsequently, it adopts a heuristic algorithm designed to solve the traveling salesman problem, planning an efficient order for visiting these positions. The agents' task is to simply follow this predetermined route, visiting each navigable position in the planned order.

(*iii*) **Vision-language navigator** is specifically developed for the tourist agent "Ling", as well as for tasks within the V-*IRL* vision-language navigation benchmark. Its primary function is to guide the agent in selecting a proper direction based on navigation instructions. The detailed pipeline is presented in Appendix C.7.

(*iv*) **Intention navigator** is utilized in intentional explorer agent "Hiro" to select the most suitable direction that aligns with the agent's specific intentions. The detailed methodology and prompt are detailed in Appendix F.2.

D.3 Mapping

The mapping module in our environment is designed to equip agents with functionalities such as route planning and transportation time estimation. It mainly utilizes the "Directions API"[3] from the Google Map Platform to facilitate these capabilities. Given the complex nature of this API's interface, our principal focus has been on parsing its output and adapting it into various user-friendly interfaces for agents.

[3] https://developers.google.com/maps/documentation/directions

D.4 Place Info & Search

Place Info & Search module hosts another important information source in our platform beyond the visual street view imagery, enabling agents to interact with real-world "places". It provides various attributes of places, including type, name, location, imagery, reviews, etc. In this module, our technical efforts are primarily focused on understanding, comparing, and integrating the most suitable functions from the vast array of Google Maps Platform APIs related to place information and nearby place searches. Additionally, we devise some post-processing strategies to identify and eliminate invalid or conflicting data sources from the Google Maps Platform.

Another essential capability enabled by this module is to associate object proposals in street view imagery and their corresponding places in the real city. This function is vital to enhance the reality of our platform by connecting street view and geolocation. It also powers the "Hiro" agent and the evaluation of the *V-IRL Place* detection benchmark. The implementation is detailed in Sec. 5.2.

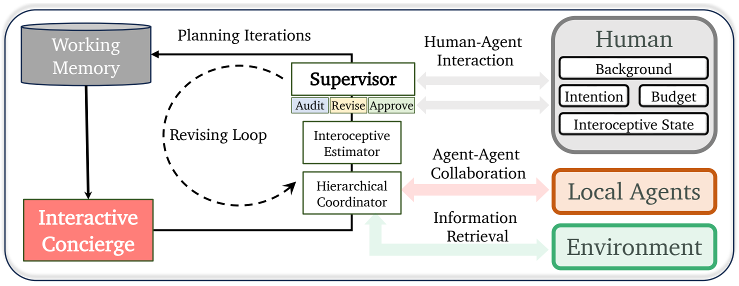


Fig. 3: Architecture overview of interactive concierge agent Diego (Sec. 3.4). See pipeline description in Appendix E.

E High-Level System Case Study: Interactive Concierge "Diego"

By studying Diego (Sec. 3.4), we illustrate how our platform's components are combined to create complex agents.

Behind Diego's proficiency in developing itineraries is his iterative planning pipeline (depicted in Fig. 3). The process begins with Diego creating an initial draft plan for the first activity using GPT-4, taking into account the user's biography, requirements, and previous activities in working memory. This draft is then meticulously refined. First, a `hierarchical coordination` module retrieves real transportation time and asks a recommendation agent for dining recommendations. Subsequently, an `interoceptive estimation` module evaluates the effect of the proposed activity on the user's mental/physical state and budget.

The crucial final step involves a `supervisor` module, which reviews ("audits") the incoming activity in light of the current user status, remaining budget, and potential interactions (exemplified in Fig. 7 of the main paper). If the `supervisor` deems the plan unsuitable, it initiates revisions. The revised plan is then looped back to the `hierarchical coordinator` and `interoceptive estimator` for reliability, followed by another review from the `supervisor` (see the revising loop in Fig. 3). This iterative process between the `hierarchical coordinator`, the `interoceptive estimator`, and the `supervisor` continues until the `supervisor` approves the activity and adds it to its `working memory`.

After finalizing an activity, Diego proceeds to plan the subsequent activity by repeating this process until the day's itinerary is complete.

F Low-Level System Case Study: Intentional Explorer "Hiro"

This section delves deeper into the low-level implementation details of the Intentional Explorer agent "Hiro" (Sec. 3.3), focusing on the prompts utilized to interact with various parts of our system. Concretely, we present the prompts in four subparts: *identifying a type of place to search using the user-defined intention* (Appendix F.1), *selecting appropriate roads* (Appendix F.2), *summarizing reviews of places* (Appendix F.3), and *making action decisions* (Appendix F.4). These four components jointly enable Hiro to explore in our interactive embodied environment driven by his initial intention.

F.1 Intention to Place Type

Starting with a user-defined agent intention, Hiro first determines the type of place that could fulfill this intention using GPT-4 and the following prompt:

```
[Role]
You are PlaceSuggesterGPT, an expert in recommending types of places
based on user-specified intentions.

[Task Description]
Given a user-specified intention, determine the type of "place" one
should seek to fulfill the intention. Your response should be in the
following JSON format:
{"place": "Desired Place Type"}

[Example]
Input: "Intention: <buy a book>"
Output: {"place": "bookstore"}

[Input]
Intention: <{agent_intention}>
[Output]
Your recommended place type based on the user-specified intention, in
the required JSON format:
```

Using this prompt with the intention

> *Hiro is hungry and looking for a place where he can try some good local food. He cannot handle spicy food.*

returns the result

```
{"place": "restaurant"}.
```

The identified place type (here, `restaurant`) is extracted and set as the target category for Hiro's open-world detector during his exploration.

F.2 Road Selection

Whenever Hiro is at a crossroads, he determines the best road to follow using his multi-modal LLM and GPT-4. The primary goal of the road selection process is to identify the road most likely to lead to the desired place type that aligns with Hiro's intention. First, Hiro fetches the street view towards each potential road using the V-*IRL* environment. Then he utilizes his multi-modal LLM (such as InstructBLIP [6] or LLaVA [13]) to generate captions for each road using the following prompt:

```
I am looking for a {place_type}. Please detail information that might
be helpful for me along this road:
```

Captions for each road are then formatted in the style of

```
{road_idx}: {road_description}
```

and concatenated to form `all_road_descriptions`. These road captions, along with Hiro's user-defined intention, are then fed into GPT-4 to determine the most promising road to follow using the following prompt:

```
[Role]
You are PathSelectorGPT, an expert in choosing the optimal road from
multiple candidates based on a user-specified intention.

[Task Description]
Given an intention, the road previously traveled, and descriptions of
available candidate roads, select the best road from the crossroad.
Your response must be in the following JSON format:
{"idx": "Selected road index", "reason": "Justification for your selection"}
[Example]
For the intention "find a grocery store", the road previously traveled
as "1", and with candidates "2: Leads to residential area, 3: Leads
to a shopping district", the output might be: {"idx": "3", "reason":
"Road 3 leads to a shopping district which is more likely to have a
grocery store."}
[Input]
User Intention: <{agent_intention}>
Road Descriptions: {all_road_descriptions}
Previously Traveled Road: Road {from_road_idx}
[Output]
Your chosen road index and the reasoning behind your selection, in the
required JSON format:
```

We design such a two-stage captioning and decision-making pipeline for road selection because Multi-modal LLMs cannot process multiple images simultaneously. However, with the recent advancements of GPT-4V, it may be possible to perform road selection using several road images with a single prompt at once. Empirical findings suggest that GPT-4V yields more reasonable choices with the following prompt:

```
[Role]
You are PathSelectorGPT, an expert in choosing the optimal road from
multiple road images according to a user-specified intention.

[Task Description]
Given a set of road images, select the best road from the crossroad.
Your answer must be in the following JSON format:
{"idx": "Selected road index (start by 0)", "reason": "Justification
for your selection"}

[Input]
User Intention: <{agent_intention}>

[Output]
Please answer with the road index and the reasoning behind your selection,
in the required JSON format:
```

An example road selection response for the first crossroad selection in Fig. 4 of the main paper is as follows:

⭐ *"idx": "0", "reason": "Choosing road 0 promises a genuine taste of local cuisine in a less commercialized setting. It's likely to have family-run eateries where I can request non-spicy dishes and savor authentic flavors. This road offers a tranquil dining atmosphere, allowing for a more engaged and leisurely culinary exploration among the locals."*

F.3 Summarize Place Reviews

When Hiro discovers a place in the street view imagery, he retrieves its corresponding name and Google reviews from the V-*IRL* environment. There is a complex algorithm behind projecting the 2D box on street view imagery to a concrete place in the real world, which is detailed in "matching between object proposal and places" of Sec. 5.2. After Hiro obtains these place reviews, he summarizes them into a place overview (to aid in decision-making) using the following prompt:

```
[Role]
You are SummarizeGPT, skilled at condensing multiple reviews into a
concise overview of a location.

[Task Description]
Given multiple reviews with ratings, craft a brief overview of the place.
```

```
Your response should be in the following JSON format:
{"summarization": "Concise description (limited to 80 words)"}

[Example]
For reviews "Great ambiance but average food (Rating: 3)" and "Loved
the decor, food could be better (Rating: 3.5)", the output might be:
{"summarization": "The place boasts great ambiance and decor, but the
food quality receives mixed reviews."}

[Input]
Reviews: {all_reviews}

[Output]
Your concise overview (max 80 words) based on the provided reviews,
in the prescribed JSON format:
```

F.4 Action Decision

After obtaining the overview of the identified place, Hiro decides to visit the place or keep exploration using GPT-4 and the following prompt:

```
[Role]
You are ActionSelectorGPT, proficient in choosing the most appropriate
action based on a user's background, intention, and an overview of a
place.

[Task Description]
Evaluate the provided user background, intention, and place overview
to select the most suitable action from the list. Your response should
be in the following JSON format:
{"action": "Selected Action", "reason": "Justification for your choice"}

Possible actions:
- enter_place(): Enter the designated place.
- continue(): Continue searching for another appropriate place.

[Example]
For the background "loves historical sites", intention "discover local
history", and place overview "This is a 200-year-old preserved mansion",
the output might be:
"action": "enter_place()", "reason": "The historical mansion aligns
with the user's interest in historical sites."

[Input]
User Background: <{background}>
User Intention: <{intention}>
Place Overview: <{place_intro}>

[Output]
Your chosen action and the rationale behind your decision in the prescribed
JSON format:
```

Hiro's exploration will continue if he decides to `continue()` and will terminate if he opts for `enter_place()`.

G V-*IRL* Benchmarks: Details

G.1 Data Curation

Place Types. We collect place information in each region for all 96 places types annotated by GMP[4]. Our V-*IRL* place: detection, recognition and VQA benchmarks are built upon all or part of these place types.

Data Cleaning. Though scalable, automated data collection can introduce noise due to the absence of human supervision. To this end, we design three automatic data cleaning strategies: *i*) *distance-based filtering* to exclude places not easily visible from any street views due to their distance; *ii*) *human-review filtering* to remove "zombie" places with no reviews which might no longer be valid or relevant; and *iii*) *CLIP-based filtering* to retain only *place-centric images* with a high CLIP likelihood of being `storefronts`.

Continent	**City**	**District**
Africa	Johannesburg	Rosebank
	Lagos	Surulere
Asia	Mumbai	Khar
	New Delhi	Lajpat Nagar
	Hong Kong	Prince Edward
	Tokyo	Shinjuku
Australia	Melbourne	CBD
	Melbourne	SouthBank
Europe	Milan	Brera
	London	Oxford St
North America	New York City	Chinatown, Manhattan
	New York City	SoHo, Manhattan
	San Francisco	Union Square
South America	Buenos Aires	Monserrat

Table 2: Region list for global V-*IRL* benchmarks.

Data Quality Verification. We randomly sample over 10% data from each benchmark and manually validate the quality. As shown in Tab. 3, only about 6% of the samples contain errors. This confirms the high quality of our data, particularly given the real-world sources. This is attributed to the three data-cleaning strategies for benchmark curation.

[4] https://developers.google.com/maps/documentation/places/web-service/supported_types/#table1

V-*IRL* benchmark	Rec & VQA	Loc	VLN
Sample error rate	3.8%	6.3%	7.1%

Table 3: Data quality verification for V-*IRL* benchmarks.

G.2 V-*IRL* Places: Detection (Details)

Matching between Object Proposals and Places. As mentioned in Sec. 5.1, we do not annotate bounding boxes for places on each potential street view image. Such human annotation diverges from our initial motivation of providing plug-and-play and sensor-rich (V-*IRL*) benchmarks. To assign ground truth for each object proposal in this scenario, we develop a simple matching strategy to assign object proposals from street view object detections to nearby places.

As illustrated in Fig. 4, we first project the bounding box of each object proposal onto a frustum in the 3D space, subject to a radius. We then determine if any nearby places fall within this frustum and radius. If any nearby place is found, the closest one is assigned as the *ground truth* for the object proposal. Otherwise, the object proposal is regarded as a *false positive*. When multiple places are inside the frustum, we consider the nearest one as the ground truth since it would likely block the others in the image. *This process is also used in Intentional Explorer agent Hiro to parse object proposals on images to place information.*

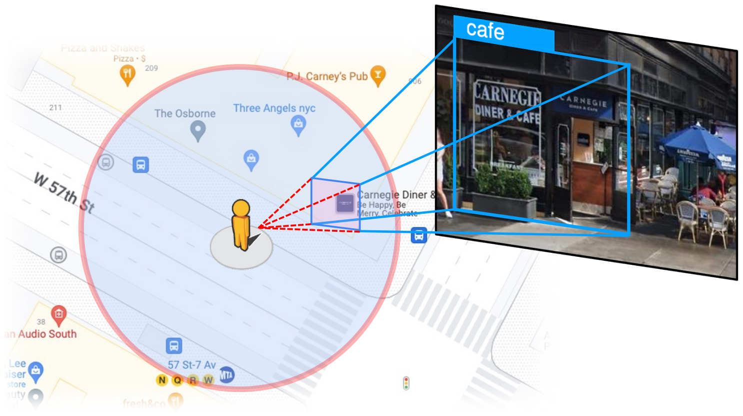


Fig. 4: Matching between 2D object proposal and street place.

All Category Results. Due to the page limit of the main paper, we only present the results of 10 categories in Tab. 1 of the main paper. Here, we present the place recall for all 20 categories in Fig. 5.

Example Illustrations. To facilitate the understanding of *V-IRL Place* Detection benchmark, we present some examples of CLIP (w/ GLIP proposals) in Fig. 12.

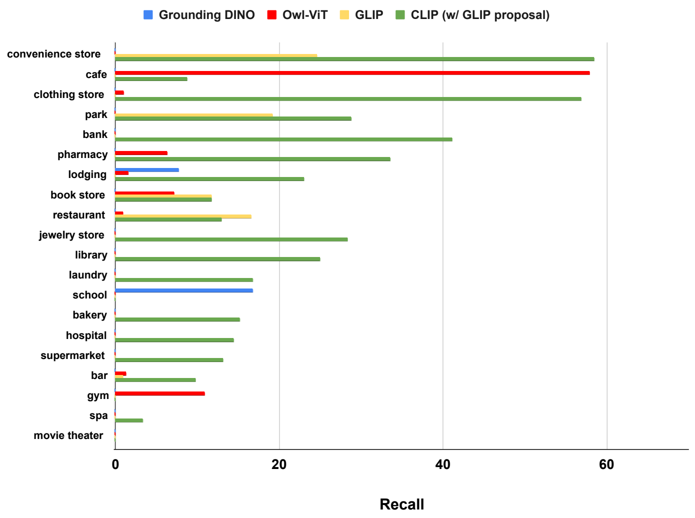


Fig. 5: Recalls in *V-IRL Place* Detection

Error Diagnosis of Detectors. Here, we conduct error diagnosis for detectors in the *V-IRL Place* detection benchmark. We examine two error types: localization error and classification error. As depicted in Fig. 6, the primary challenge arises in classification, where detectors struggle to assign correct labels, despite having accurate object proposals.

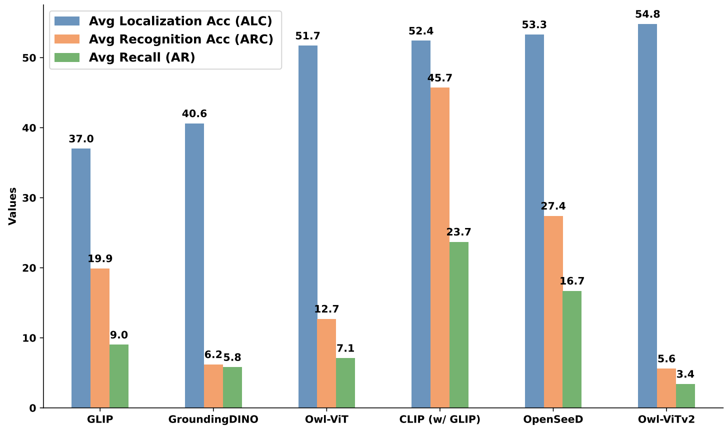


Fig. 6: Error diagnosis in *V-IRL Place* Detection

G.3 V-*IRL* Places: Recognition and VQA (Details)

Place-centric Images *vs.* Street View Images. In contrast to the street view imagery utilized in the *V-IRL Place* detection benchmark, the *V-IRL Place* recognition and VQA benchmarks use place-centric images. To illustrate the

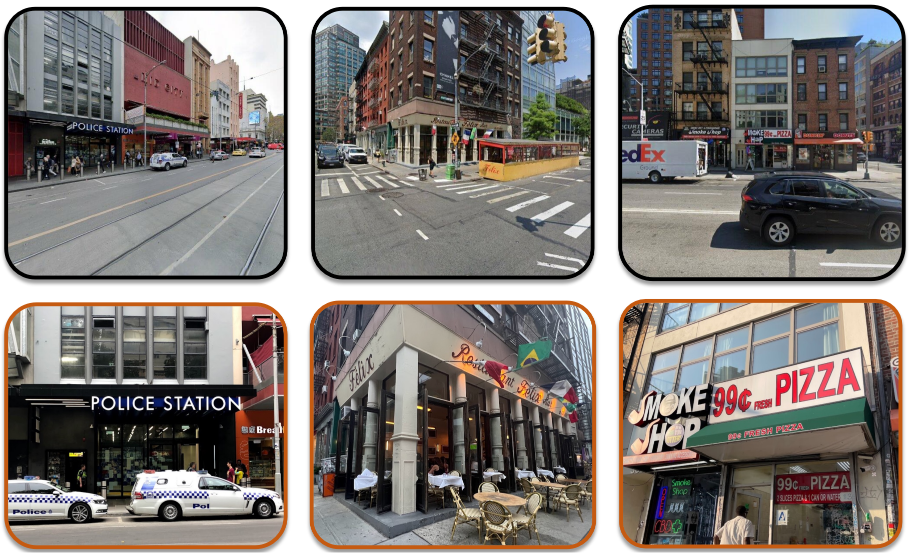


Fig. 7: Top row: examples of street view imagery. Bottom row: corresponding place-centric images.

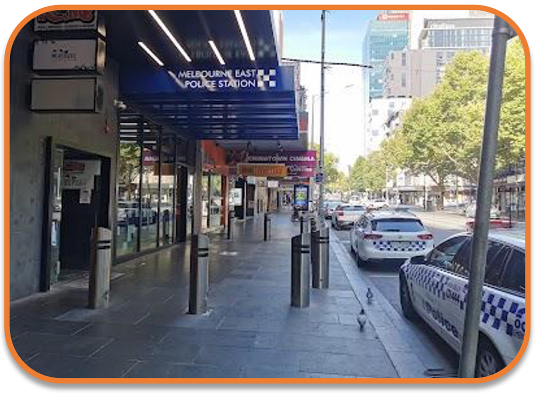
distinction between these image types, we present examples in Fig. 7. The figure shows that street view images, sourced from the Google Street View database[5], are taken from the street and encompass a broader view of the surroundings, including multiple buildings and possible occluding objects/vehicles. In contrast, place-centric images, drawn from the Google Place database[6], are taken on foot and focus more closely on the specific place—providing a more concentrated view.

***V-IRL Place* VQA Process.** The *V-IRL Place* VQA process is illustrated in Fig. 8, where the candidate and true choices are generated by GPT-4 [2] given the place types and place names corresponding to the image.

Place Types Performance for Recognition. In Figure 9, we present the averaged accuracy for each place type across 10 benchmarked vision models. The size and the x-axis position of each bubble correspond to the number of places within each type. A clear trend emerges: accuracy tends to correlate with the frequency. Common categories such as `clothing store`, `cafe` exhibit higher accuracy, whereas vision models often struggle with infrequent place types like `bowling alley` or `mosque`.

Place Types Performance for VQA. The place types performance of the V-*IRL* place VQA in Fig. 10 further verifies the correlation between accuracy and frequency from a human intention perspective. The top-10 categories are closely aligned with the most common human activities, purchasing and dining.

[5] https://developers.google.com/maps/documentation/streetview/request-streetview
[6] https://developers.google.com/maps/documentation/places/web-service/photos



Question: Which human intentions can be accomplished here?
Choices: A. Learning how to cook authentic Australian food.
B. Applying for a reduction on parking fines.
C. Reporting a crime or lost property.
D. Attending a yoga session.

Fig. 8: Example of *V-IRL Place* VQA process.

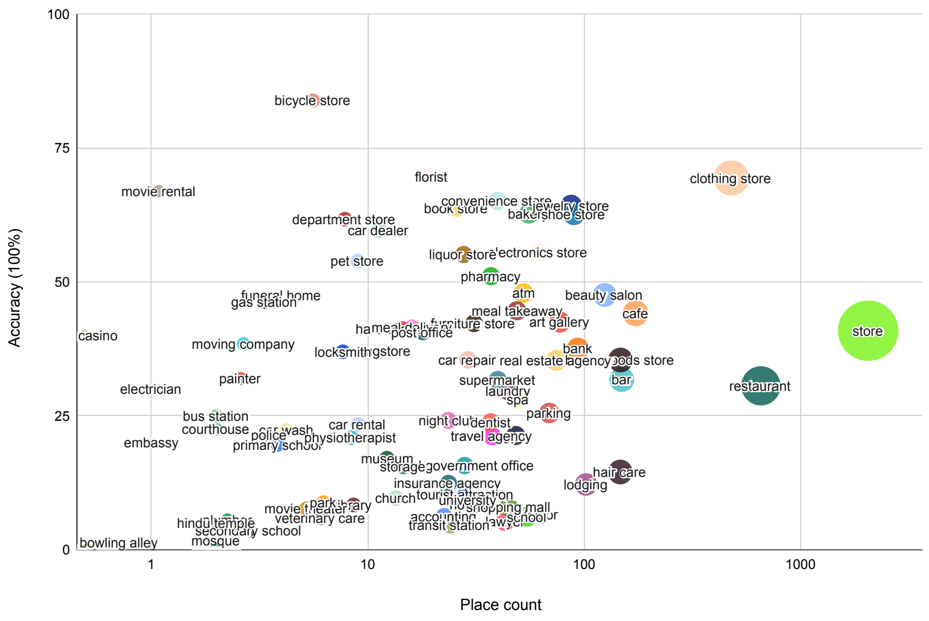


Fig. 9: Category-wise accuracy and numbers for V-*IRL* Place Recognition benchmark.

In contrast, the bottom-10 place types relate to places that are less frequently encountered and serve a more diverse purpose, such as `mosque`, `plumber` and `embassy`.

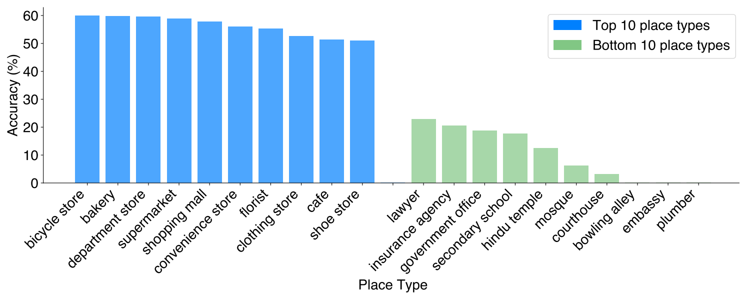


Fig. 10: Top-10 and bottom-10 place types averaged on four vision models of V-*IRL* Place VQA.

More *V-IRL Place* VQA Results. Here, we present more state-of-the-art MLLMs including LLaVA-NeXT [12], Mini-Gemini [10], InternVL-1.5 [5], GPT-4(V) [2],Qwen-VL-Max [3]. As shown in Fig. 11, LLaVA-NeXT (7B) outperforms its predecessors LLaVA-1.5 and 1.0, but still has over 8% gap to InternVL-1.5 with 26B parameters. Closed-source MLLMs GPT-4V and Qwen-VL-Max yield outstanding performance compared to most open-sourced models.

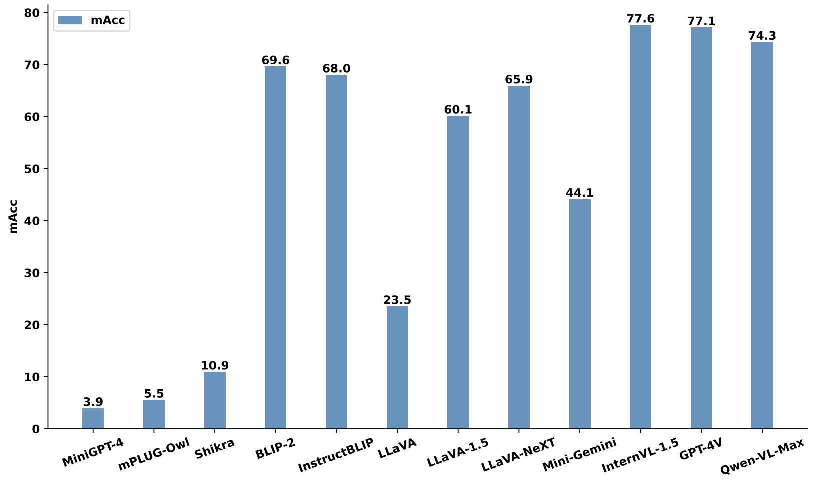


Fig. 11: More state-of-the-art model results in *V-IRL Place* VQA benchmark.

Consistency Analysis of *V-IRL Place* VQA Results. Here, we study the *sensitivity of MLLM to the order of QA options* in VQA. As shown in Tab. 4, advanced MLLMs still exhibit 4.5%-10.6% mAcc drop with circular evaluation. This highlights that MLLMs are still sensitive to the order of QA options presented.

Method	InternVL-1.5 [5]	GPT-4V [2]	Qwen-VL-Max [3]
mAcc (w/ circular)	77.6	77.1	74.3
mAcc (w/o circular)	82.1	83.1	84.9
mAcc drop	**-4.5**	**-6.0**	**-10.6**

Table 4: MLLM consistency analysis on *V-IRL Place* VQA benchmark.

G.4 V-*IRL* Vision-Language Navigation (Details)

Navigation Pipeline. As mentioned in Appendix C.7, our VLN pipeline is similar to [16], however, our benchmark offers greater scalability through the worldwide V-*IRL* platform and an automated data collection pipeline, as opposed to the manual annotation of a specific region. Furthermore, our benchmark emphasizes the analysis of the *vision* component in the VLN pipeline, as opposed to [16], which aims to enhance performance on existing VLN datasets using LLMs.

Implementation Details. Here, we introduce the implementation details for LLaVA-1.5 [11] and PP-OCR [7] (+ GPT-3.5). For LLaVA-1.5 [11], we transform the landmark recognition task to a *multiple choice VQA* problem, asking

```
Which of the following landmarks can be identified with a high
degree of confidence?
```

The VQA options include all potential landmarks mentioned in the route description, along with a "`None of above`" choice. The model's response to this question is then parsed as the landmark observation.

For PP-OCR [7] (+ GPT-3.5), we first extract all recognized text using PP-OCR [7] for each street view image. Then, GPT-3.5 [14] determines the presence of each landmark in this street view image, jointly considering the OCR text and landmark name.

Full Set Results. Apart from the mini-set results presented in Sec. 5.4, we also provide the full set results of Oracle and CLIP (L/14@336px) in Tab. 5. The Oracle results, interestingly, do not achieve a 100% success rate, due to incorrect decisions made by the LLM at stop positions. This is evidenced by the high arrival ratio and low reaction accuracy at stop positions. Empirically, we observe that the LLM occasionally decides to keep moving, despite clear destination indications in the observations.

When we substitute the map in oracle with the CLIP model to gather landmark observations from street view imagery, we observe a significant drop in the success rate, due to the inevitable model prediction errors. To improve the success rate in VLN, we can focus on two important factors: (*i*) designing better vision models; (*ii*) developing LLMs and prompt techniques that are robust to vision-related noise. Especially, our empirical findings suggest that sophisticated prompt designs significantly improve the robustness of LLMs to visual observation noise.

Method		Start	Intersection		Stop	
	Success	**Reac**	**Arr**	**Reac**	**Arr**	**Reac**
Oracle (No Vision)	0.88	1.0	0.95	0.99	0.96	0.88
CLIP (L/14@336px)	0.22	0.84	0.66	0.90	0.61	0.22

Table 5: Results of V-*IRL* VLN-full.

H Discussion: Ethics & Privacy

Our platform serves as a tool for AI development and as a crucible for ethical discourse and preparation. As AI is inevitably being integrated into society—*e.g.*, via augmented reality wearables, spatial computing platforms, or mobile robots navigating city streets—it is imperative to confront and discuss ethical and privacy concerns now. Unlike these impending *real-time* systems, the data accessed by V-*IRL* is "stale" and preprocessed—providing a controlled environment to study these concerns.

Notably, V-*IRL* exclusively utilizes preexisting, readily available APIs; it does not capture or make available any previously inaccessible data. Our primary source of street-view imagery, Google Maps [9], is subject to major privacy-protection measures, including blurring faces and license plates [8]. Moreover, V-*IRL* complies with the Google Maps Platform license[7], similarly to notable existing works that also leverage Google's street views [1, 4].

We believe V-*IRL* is an invaluable tool for researching bias. As discussed in Sec. 5.5, V-*IRL*'s *global scale* provides a lens to study linguistic, cultural, and other geographic biases inherent in models. By using V-*IRL* to study such questions, we aim to preemptively tackle the ethical dilemmas that will arise with deploying real-time systems rather than being blindsided by them. We hope our work helps spur proactive discussion of future challenges throughout the community.

I Conclusion

In this work, we introduce V-*IRL*, an open-source platform designed to bridge the sensory gap between the digital and physical worlds, enabling AI agents to interact with the real world in a virtual yet realistic environment. Through V-*IRL*, agents can develop rich sensory grounding and perception, utilizing real geospatial data and street-view imagery. We demonstrate the platform's versatility by creating diverse exemplar agents and developing benchmarks measuring the performance of foundational language and vision models on open-world visual data from across the globe.

This platform opens new avenues for advancing AI capabilities in perception, decision-making, and real-world data interaction. As spatial computing and robotic systems become increasingly prevalent, the demand for and possibilities of AI agents will only grow. From personal assistants to practical applications

[7] https://cloud.google.com/maps-platform/terms

like urban planning to life-changing tools for the visually impaired, we hope V-*IRL* helps usher in a new era of perceptually grounded agents.

References

1. Image geo-localization based on multiplenearest neighbor feature matching usinggeneralized graphs. TPAMI (2014) 21
2. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023) 6, 17, 19, 20
3. Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., Zhou, J.: Qwen-vl: A frontier large vision-language model with versatile abilities. arXiv preprint arXiv:2308.12966 (2023) 19, 20
4. Chen, H., Suhr, A., Misra, D., Snavely, N., Artzi, Y.: TOUCHDOWN: Natural language navigation and spatial reasoning in visual street environments. In: CVPR (2019) 21
5. Chen, Z., Wang, W., Tian, H., Ye, S., Gao, Z., Cui, E., Tong, W., Hu, K., Luo, J., Ma, Z., et al.: How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. arXiv preprint arXiv:2404.16821 (2024) 19, 20
6. Dai, W., Li, J., Li, D., Tiong, A.M.H., Zhao, J., Wang, W., Li, B., Fung, P., Hoi, S.: InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning. In: NeurIPS (2023) 11
7. Du, Y., Li, C., Guo, R., Yin, X., Liu, W., Zhou, J., Bai, Y., Yu, Z., Yang, Y., Dang, Q., et al.: PP-OCR: A practical ultra lightweight ocr system. arxiv 2020. arXiv preprint arXiv:2009.09941 (2020) 20
8. Frome, A., Cheung, G., Abdulkader, A., Zennaro, M., Wu, B., Bissacco, A., Adam, H., Neven, H., Vincent, L.: Large-scale privacy protection in google street view. In: ICCV (2009) 21
9. Google Map Team: Google Map Platform. `https://mapsplatform.google.com/` 21
10. Li, Y., Zhang, Y., Wang, C., Zhong, Z., Chen, Y., Chu, R., Liu, S., Jia, J.: Mini-gemini: Mining the potential of multi-modality vision language models. arXiv preprint arXiv:2403.18814 (2024) 19
11. Liu, H., Li, C., Li, Y., Lee, Y.J.: Improved baselines with visual instruction tuning. arXiv:2310.03744 (2023) 20
12. Liu, H., Li, C., Li, Y., Li, B., Zhang, Y., Shen, S., Lee, Y.J.: Llava-next: Improved reasoning, ocr, and world knowledge (January 2024), `https://llava-vl.github.io/blog/2024-01-30-llava-next/` 19
13. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: NeurIPS (2023) 11
14. Schulman, J., Zoph, B., Kim, C., Hilton, J., Menick, J., Weng, J., Uribe, J.F.C., Fedus, L., Metz, L., Pokorny, M., et al.: ChatGPT: Optimizing language models for dialogue. OpenAI blog (2022) 20
15. Schumann, R., Riezler, S.: Generating landmark navigation instructions from maps as a graph-to-text problem. In: ACL (2020) 5
16. Schumann, R., Zhu, W., Feng, W., Fu, T.J., Riezler, S., Wang, W.Y.: VELMA: Verbalization embodiment of llm agents for vision and language navigation in street view. arXiv preprint arXiv:2307.06082 (2023) 5, 20
17. Wu, P., Xie, S.: V*: Guided Visual Search as a Core Mechanism in Multimodal LLMs. arXiv preprint arXiv:2312.14135 (2023) 4

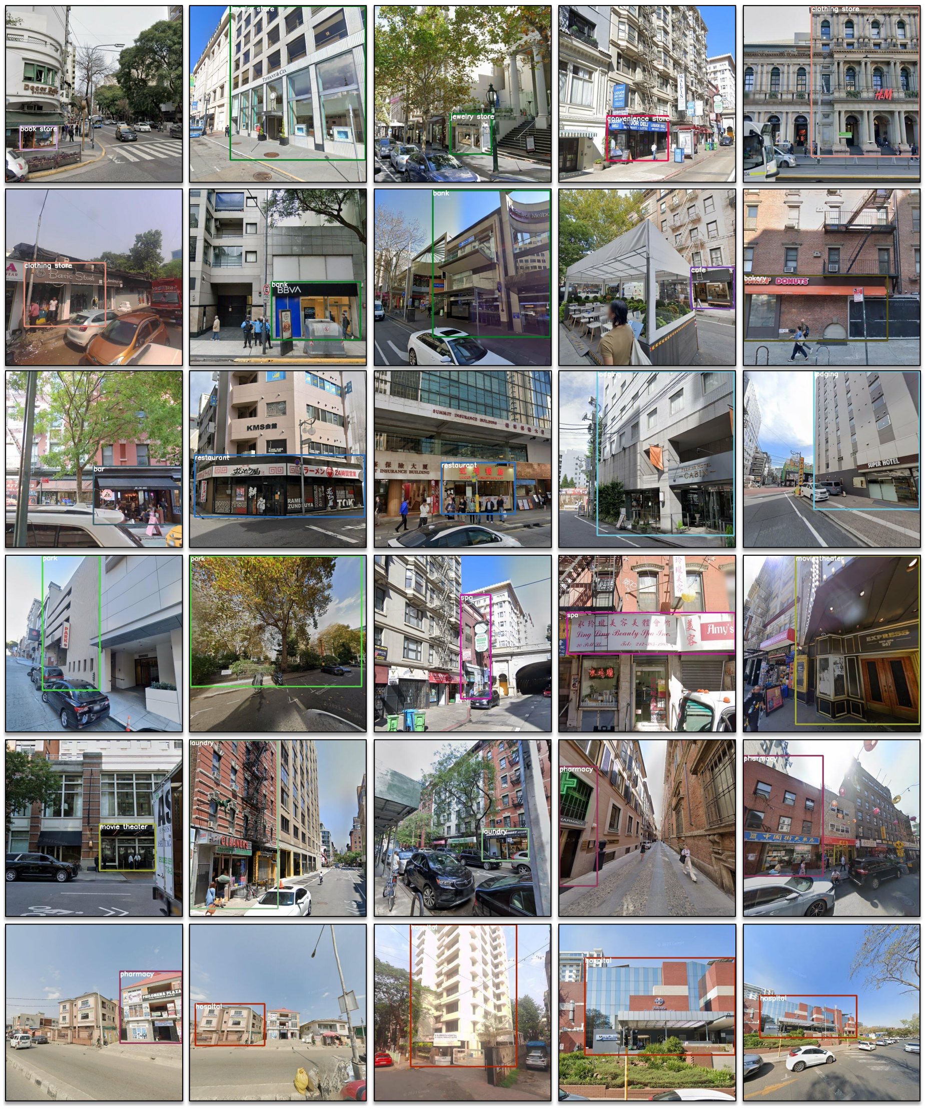


Fig. 12: Samples of *V-IRL Place* Detection using CLIP (w/ GLIP proposals).