



Habitat: A Platform for Embodied AI Research

Authors: M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijnmans, B. Jain, J. Straub, J. Liu, V Koltun, J Malik, D. Parikh, D. Batra

Presented by: Shuqing Li, Arunpreet Sandhu, Ryan Stager



Team



Manolis Savva^{1,4*}



Abhishek Kadian^{1*}



Oleksandr Maksymets^{1*}



Yili Zhao¹



Erik Wijmans^{2,3}



Bhavana Jain¹



Julian Straub²



Jia Liu¹



Vladlen Koltun⁵



Jitendra Malik^{1,6}



Devi Parikh^{1,3}



Dhruv Batra^{1,3}

facebook Artificial Intelligence

1

facebook Reality Labs

2

Georgia Tech

3

SFU

4

intel

5

Berkeley UNIVERSITY OF CALIFORNIA

6



Navigation of unknown 3D spaces is a difficult problem.



Where am I?

Where am I going?

Where did I start?

Have I been here before?

How distant is A from B?
What is the shortest path from A to B?

Real Life applications



Yandex self-driving car. Hyundai.



Air delivery drone for Amazon PrimeAir

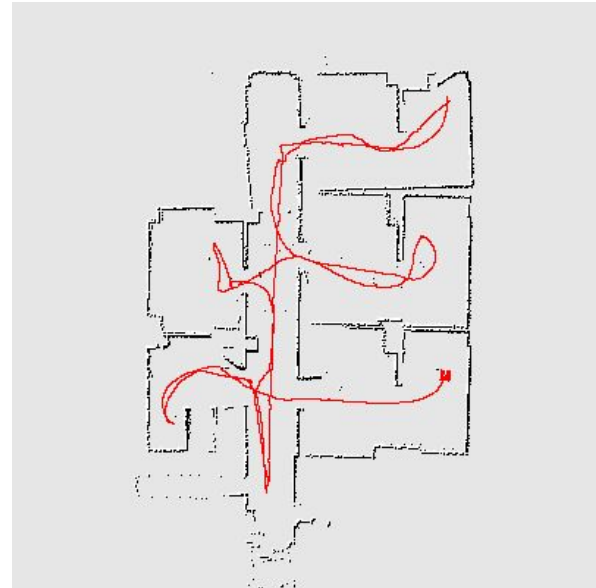
Simultaneous Localization and Mapping (SLAM)

Inputs

- No external coordinate reference
- Time sequence of sensor measurements made as robot moves through the unknown environment
 - Typical visual sensors used

Outputs

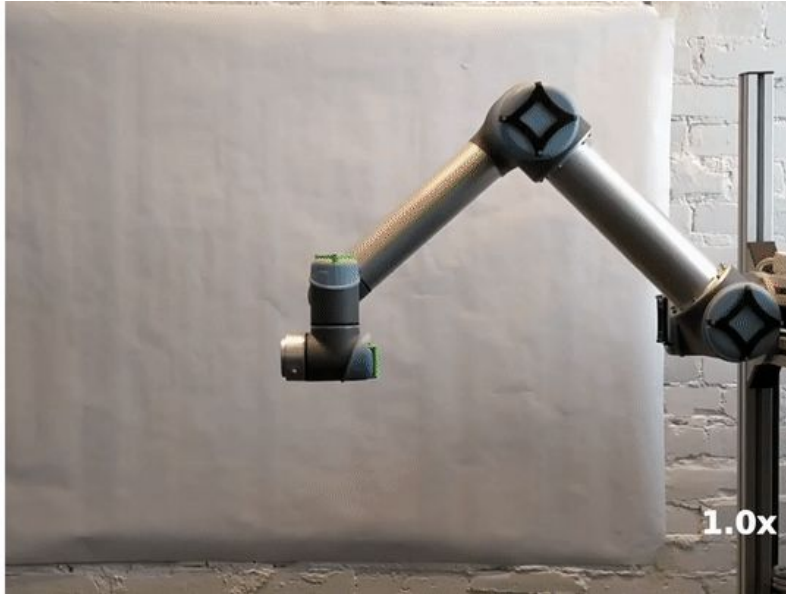
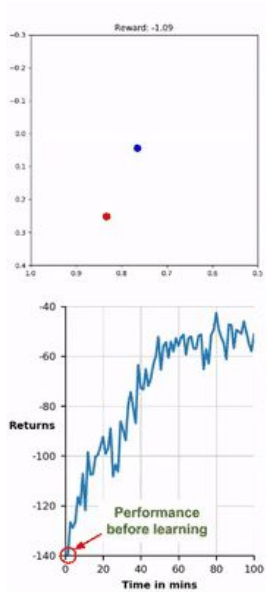
- A map of the environment
- Pose estimate of robot w.r.t the defined map





Nirbot Team- Negev Inspection Robot SLAM Navigation Task

Real World Embodied AI



- Limitations
 - Slow: Only as fast as real time
 - Dangerous: Collisions lead to expensive repairs
 - Expensive: Cannot parallelize without high cost
 - Difficult to control: Higher entropy environments with limited training
- Benefits
 - Can encode semantic data from training set and leverage past experiences

Setting up a Reinforcement Learning Task with a Real-World Robot

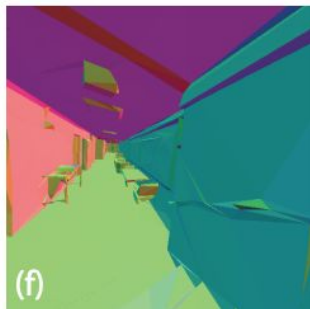
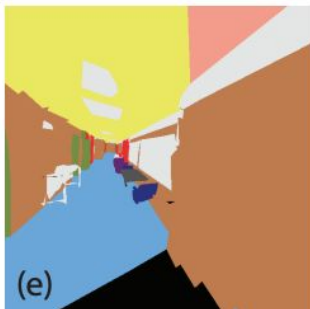
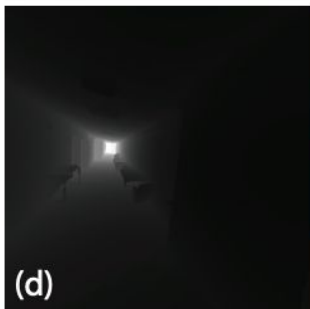
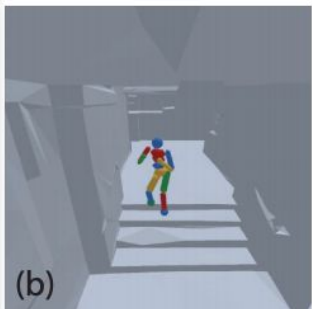
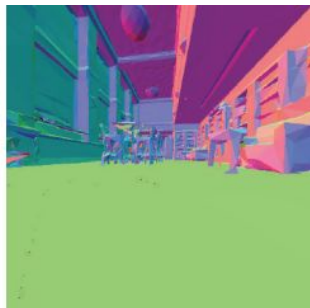
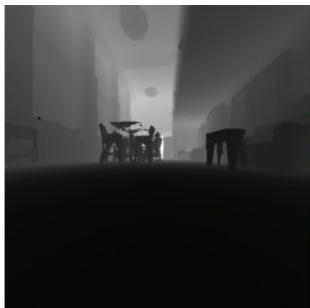
Related Works: AI2-Thor



Related Works: MINOS



Related Works: Gibson



(a)

(b)

(c)

(d)

(e)

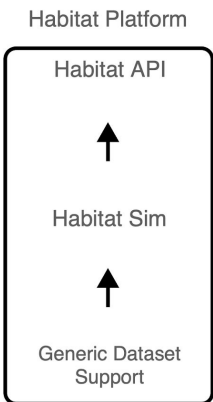
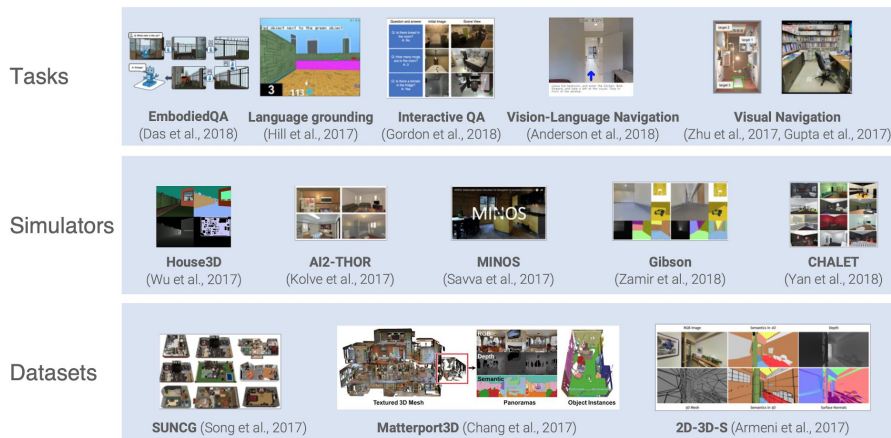
(f)

... And Plenty More!

Simulator	Agent	Modalities	Framerate	Environment	Dataset size
Gazebo (Koenig and Howard, 2014)	articulated	sensor plugins	10s+ FPS	indoor + outdoor	few environments
Project Malmö (Johnson et al., 2016)	continuous /discrete	color	10s+ FPS	Minecraft	few environments
ViZDoom (Kempka et al., 2016)	continuous	color, depth, segm	1000s+ FPS	stylized mazes	few mazes
DeepMind Lab (Beattie et al., 2016)	continuous	color, depth	100s+ FPS	stylized mazes	few mazes + procedural
AI2-THOR (Zhu et al., 2017)	continuous /discrete	color	100s+ FPS	indoor (synthetic)	32 rooms
CMP (Gupta et al., 2017)	discrete	color, depth	10s+ FPS	indoor (reconstructed)	6 floors
CAD2RL (Sadeghi and Levine, 2017)	continuous	color, depth	100s+ FPS	indoor (synthetic)	12 corridors + variations
MINOS	continuous /discrete	reconfigurable multimodal	100s+ FPS	indoor (synthetic+reconstructed)	45K houses + variations

MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments

Unifying Task, Simulator, and Datasets



- Aims to solve
 - Tight coupling of task, Simulation Platform, and Datasets
 - Hard-coded agent configuration
 - Limited control of environment state
 - Poor simulation performance



Key Contribution

Habitat



Datasets

- Datasets such as ImageNet, COCO, and VQA.
- Focus on datasets which are images for training AI that is focused on pattern recognition in images, videos, and text.

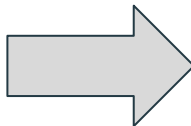
Environments

- Simulations more akin to the Habitat demo.
- Continuous environments which give contextual framing to embodied AI.
- Is more conducive to environment task based training such as point goal.

Habitat Design

Requirements

- Highly performant rendering engine
- Scene dataset ingestion API
- Agent API
 - Sensor suite API
- Scenario and task API
- Humans-as-agents
- Environment state manipulation
- Implementation in python and C++
- Containerization of Habitat



Software Solutions

- Generic 3D dataset using scene graphs
- Rendering Engine
- Efficient GPU usage for certain sensors
- Habitat API

Agent API & Sensor Suite

- Habitat gives developers the an Agent API to test their embodied AI.
- The API gives developers a suite of sensors such as:
- **RGB sensors** to get color information about the scene
- **Depth sensors** to get information about how far something is within the scene
- **Semantic instance mask** to break apart a scene into its different instances.
- **GPS sensor** for knowing where the agent is within the environment.
- **Compass sensor** for directions



Agent API: Sensor Input

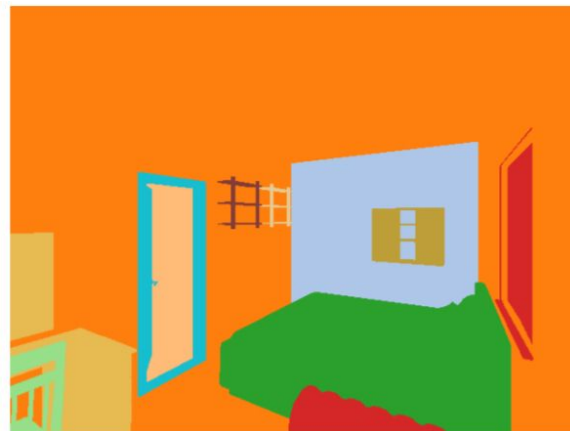
RGB Frames



Depth Frames



Instance



High performance rendering engine

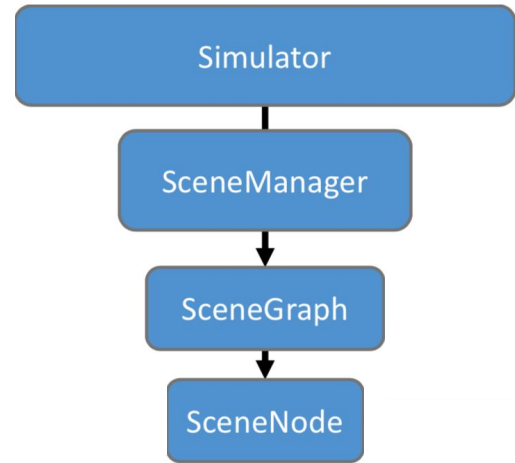
- Allows for thousands of FPS for training time.
- Allows for different sensors to receive information after every action.
- Allows for multiple processes of Habitat to run in parallel on the GPU

Sensors / Resolution	1 proc			5 procs		
	128	256	512	128	256	512
RGB	4,093	1,987	848	10,592	3,574	2,629
RGB + depth	2,050	1,042	423	5,223	1,774	1,348

Sensors / Resolution	1 proc			3 procs		
	128	256	512	128	256	512
RGB	4,093	1,987	848	10,638	3,428	2,068
RGB + depth	2,050	1,042	423	5,024	1,715	1,042
RGB + depth + semantics ⁵	439	346	185	502	385	336

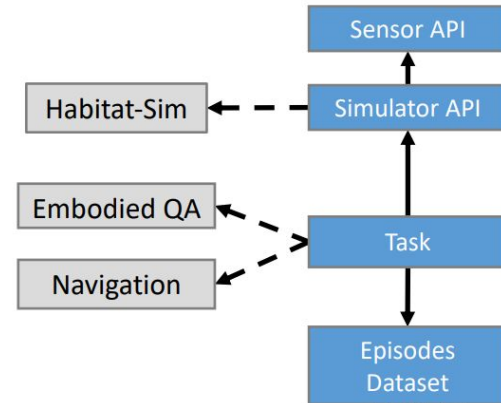
Scene dataset ingestion API

- Habitat introduces the notion of Scene Graphs.
- Scene Graphs are a hierarchical abstraction to ensure consistency.
- Used for composing the 3D environments within Habitat.
- Ensures that there is consistent information from scene to scene, dataset to dataset.
- Solves problem of differences between datasets such as Gibson and Matterport3D.
- Allows for procedural generation of scenes, editing, and other programmatic manipulation of scenes.



Scenario and task API

- Habitat gives developers a suite of embodied AI tasks such as:
 - Point Goal
 - Object Goal
 - Room Navigation
 - Embodied Question Answering
- **Goal specification** is **static** when defining tasks.
 - As in the task is only given once rather than being given every action.
 - GPS sensor is used to help the Agent move through the area



PointGoal Navigation



Scenario and task API: PointGoal and ObjectGoal

Room Navigation



Scenario and task API: Room Navigation

Instruction Following



Scenario and task API: Instruction Following

Embodied Question Answering



"Q: What color is the TV stand"

Scenario and task API: Embodied Question Answering

Habitat Definition: Task & Episode

- **Tasks** are an embodied AI action that has a termination goal.
- A task also supplies an **evaluation** of the certain embodied AI action.
- **Episodes** are a specification that includes
 - Agents initial position and orientation
 - Scene ID
 - Goal position
 - Shortest path to goal (Optional)
- An **episode** is a **description** of an **instance** of a task.
- An episode is a **single training sequence** akin to a singular image in a resnet training

Agent Action Space

- Turn left (10 degrees)
- Turn left (10 degrees)
- Move Forward (0.25m)
- Stop

Habitat Definition: Environment

- An environment is what Habitat is at its core.
- An abstraction of all important information needed to run a simulation.
- This includes:
 - 3D space rendering
 - Collision detection
 - Metadata needed for embodied AI testing.

```
import habitat

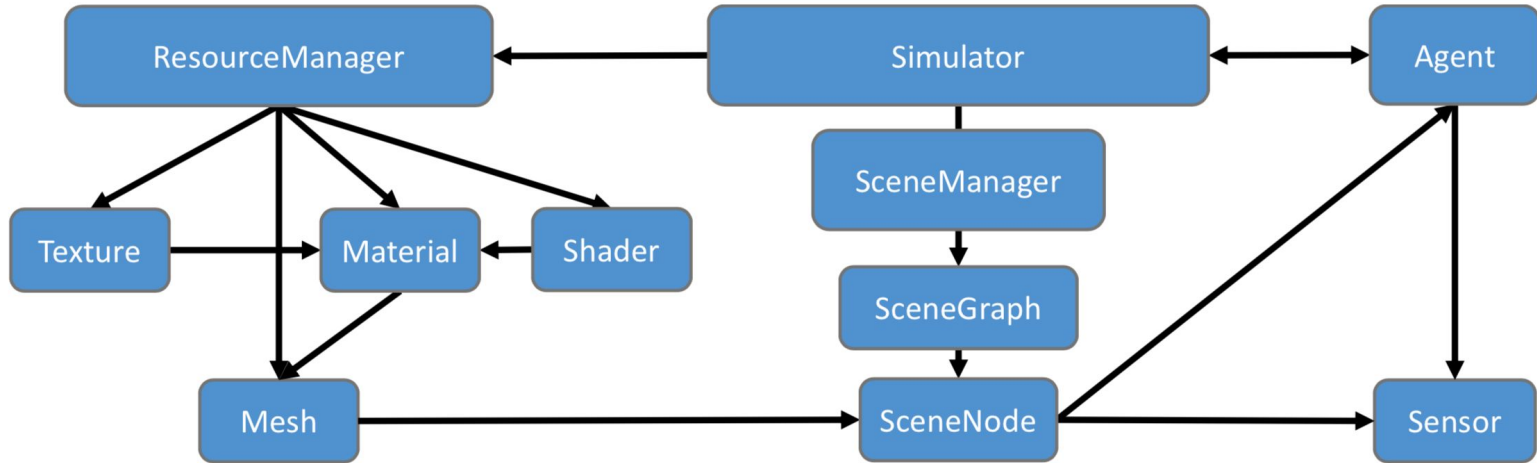
# Load embodied AI task (PointNav)
# and a pre-specified virtual robot
config = habitat.get_config(config_file=
                             "pointnav.yaml")

env = habitat.Env(config)

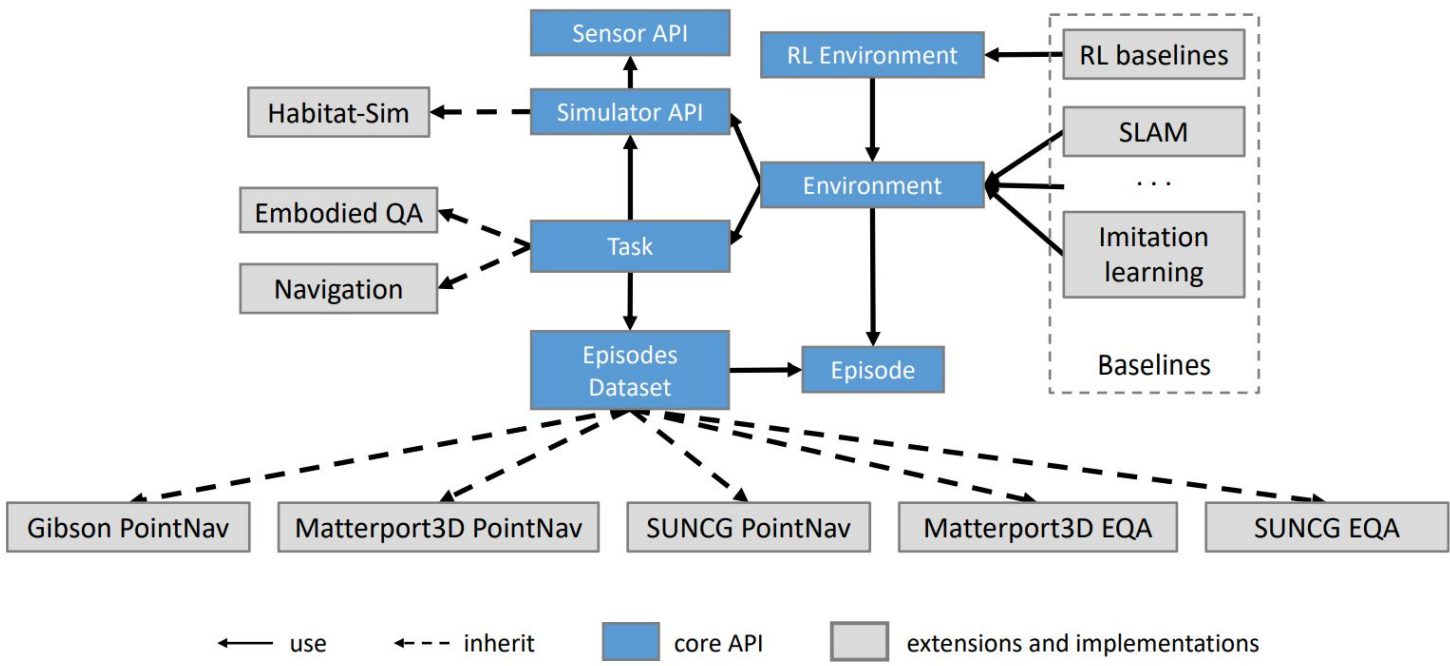
observations = env.reset()

# Step through environment with random actions
while not env.episode_over:
    observations = \
        env.step(env.action_space.sample())
```


Habitat-Sim Design

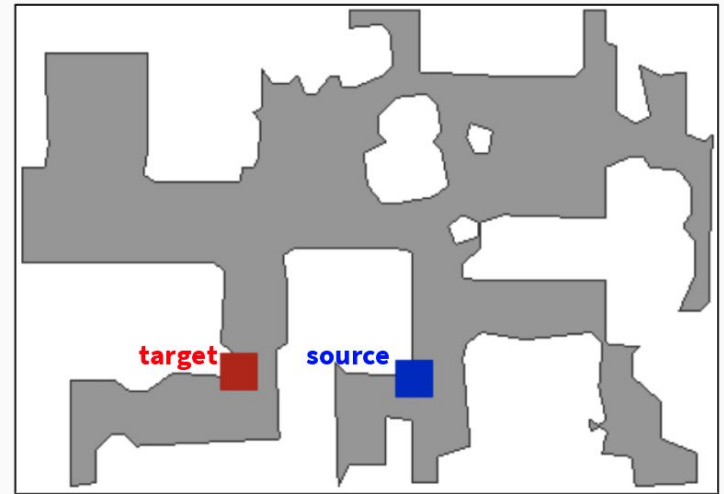


Habitat-API Design



PointGoal Navigation

- Agent initialize at a random start position and orientation in an environment
- A target coordinate w.r.t the agent's position
- Agent use sensor inputs to navigate without ground truth map



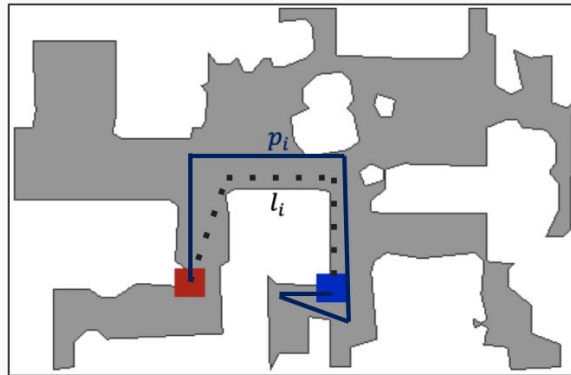
Evaluation: Success Weighted by Path Length

$$SPL = \frac{1}{N} \sum_{i=1}^N S_i \frac{l_i}{\max(p_i, l_i)}$$

l_i : length of shortest path between goal and target for episode

p_i : length of path taken by agent in episode

S_i : binary indicator of success in episode



PointGoal Navigation: Non-learning baselines

- Random: Agent randomly chooses an available action
- Forward only: Only moves forward
- Goal Follower: Moves towards goal
- Simultaneous localization and mapping (SLAM)
 - The main non-training baseline comparison.
 - Constructs a map of the unknown environment through sensors.
 - Then plans how to move through the environment.



PointGoal Navigation: Reinforcement Learning baselines

RL (PPO): Single CNN. Arch: {Conv 8x8, ReLU, Conv 4x4, ReLU, Conv 3x3, ReLU, Linear, ReLU}

- Blind: No sensors
- RGB: Only Red, Green, Blue sensors
- Depth: Only depth sensors
- RGBD: Both RGB and depth.

All RL agents were trained with this reward function:

s - a success reward

d_t - The geodesic distance from goal

λ - a time penalty





$$r_t = \begin{cases} s + d_{t-1} - d_t + \lambda & \text{if goal is reached} \\ d_{t-1} - d_t + \lambda & \text{otherwise} \end{cases}$$

PointGoal Navigation: Training

- **Six concurrently running** threads establish 500 episode blocks for training.
- Trained until each of the blocks have achieved **75 million agent steps** have been accumulated across all worker threads.
- This training procedure is **15 times larger** than previous experiences that investigated SLAM vs learning approaches.
- Overall there was 2267 GPU hours for this experiment:
 - 320 GPU-hours for Blind
 - 566 GPU-hours for RGB
 - 475 GPU-hours for Depth
 - 906 GPU-hours for RGBD



Results and Analysis

- 
- 
- 
- 
- (1) As the amount of training experience increases, how do learning-based agents compare to the classical SLAM and hand-coded baselines?
 - (2) How well do the learned agents generalize across 3D datasets?

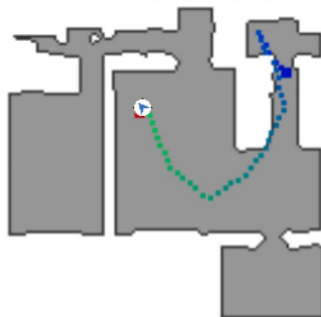
PointGoal Navigation Episodes

Gibson

Blind SPL=0.28



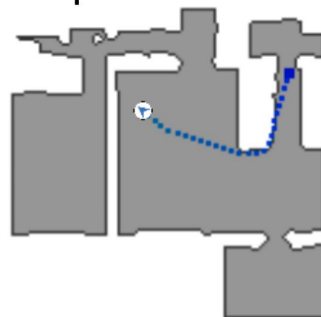
RGB SPL=0.57



RGBD SPL=0.91



Depth SPL=0.98



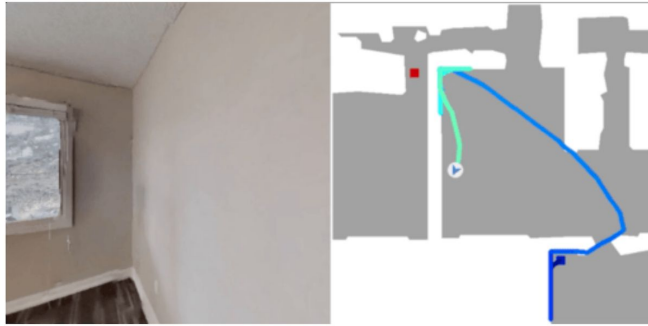
Blue dot: start point

Red dot: goal position

Blue arrow: final position

Blue-green-red line: agent's trajectory

Gibson



Blind SPL = 0.00



RGB SPL = 0.45



RGBD SPL = 0.82

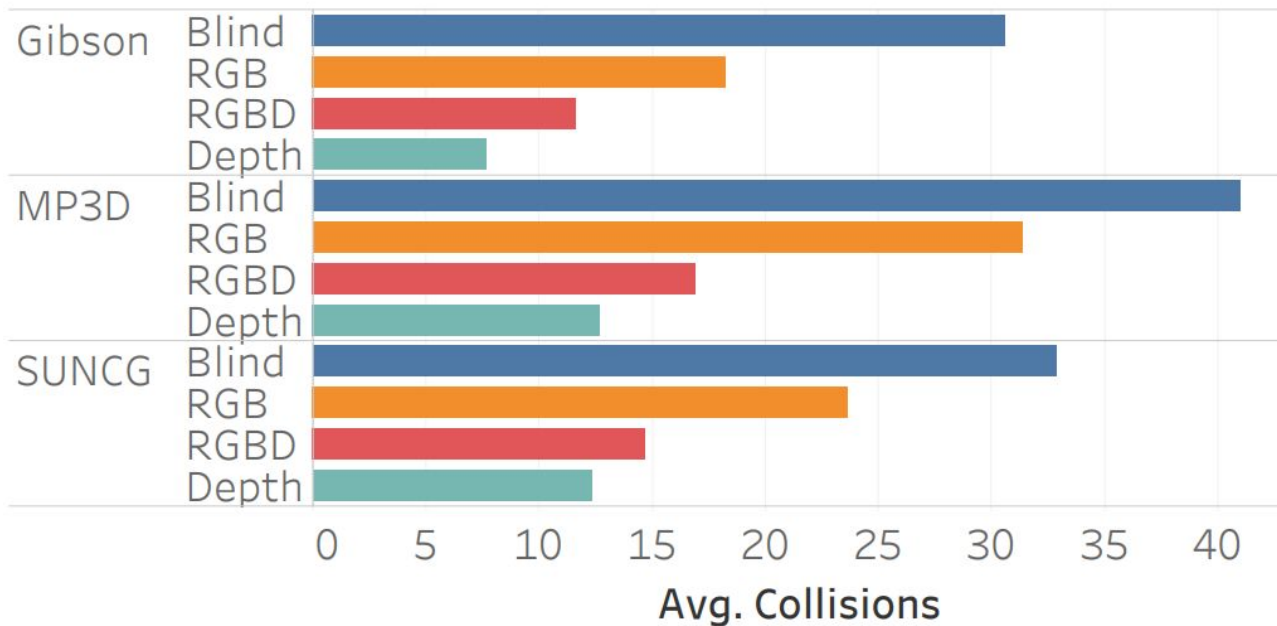


Depth SPL = 0.88

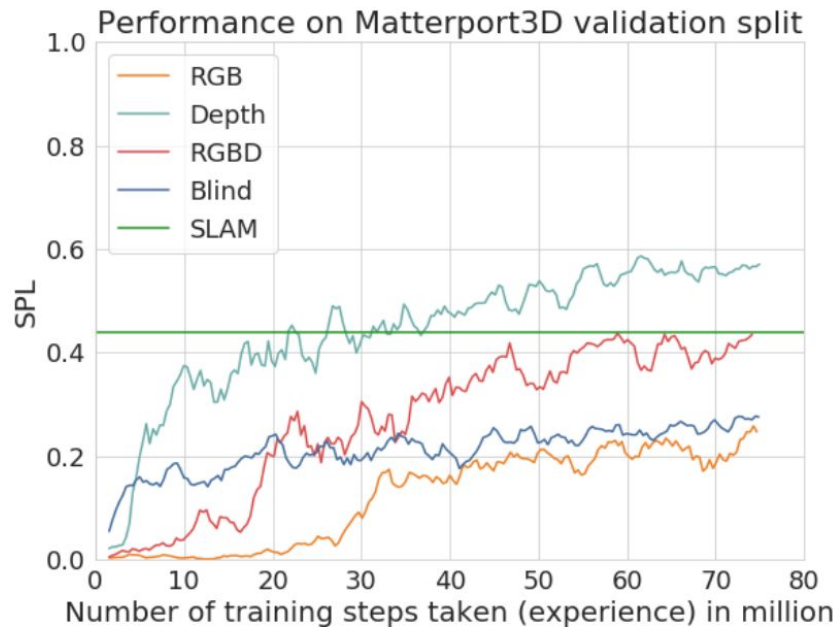
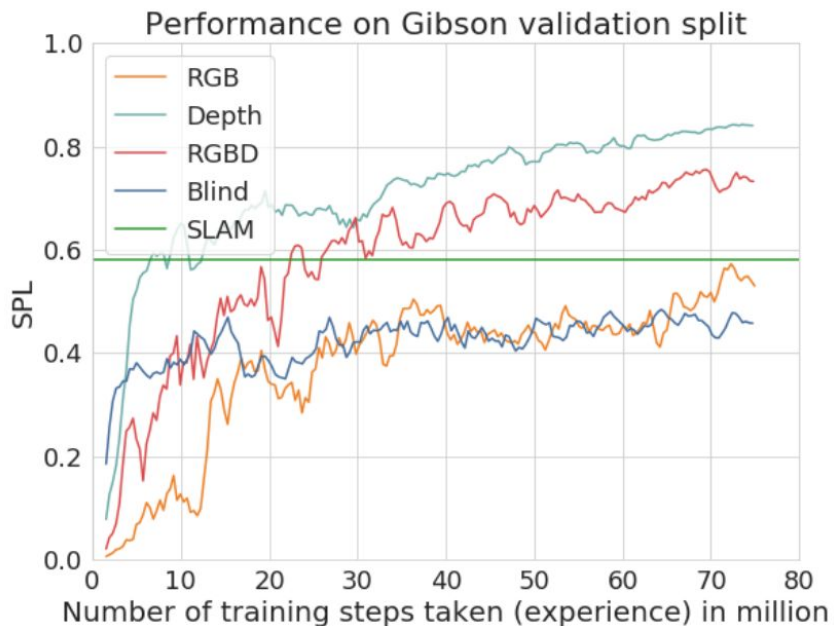
Blind: 'wall-following' behavior

Depth: Navigate efficiently

Average number of collisions during successful navigation episodes

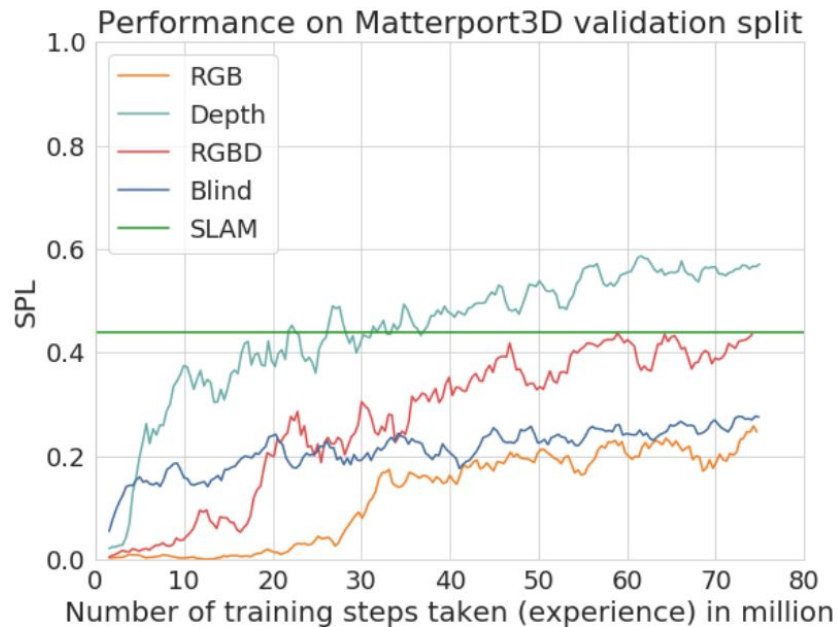
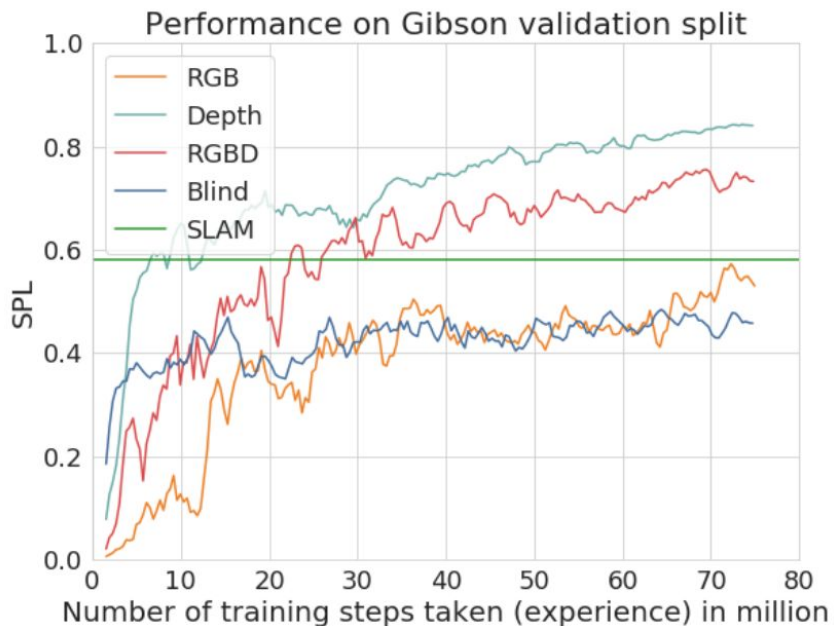


Average SPL of agents on val set



Depth and RGBD learning-agents match the and outperform classical SLAM.

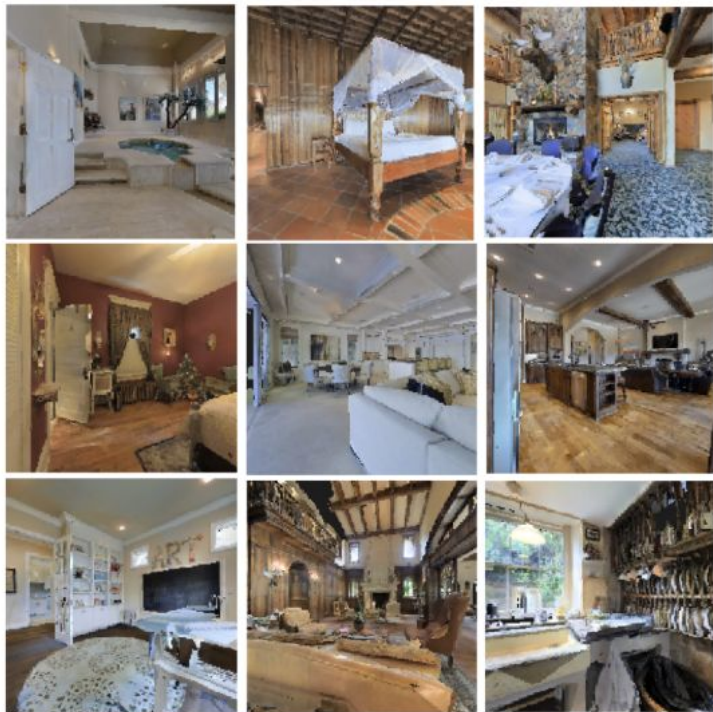
Average SPL of agents on val set



RGB sensors provide a high-dimensional complex signal that may cause overfitting.

RGB is a high entropy, extraneous signal

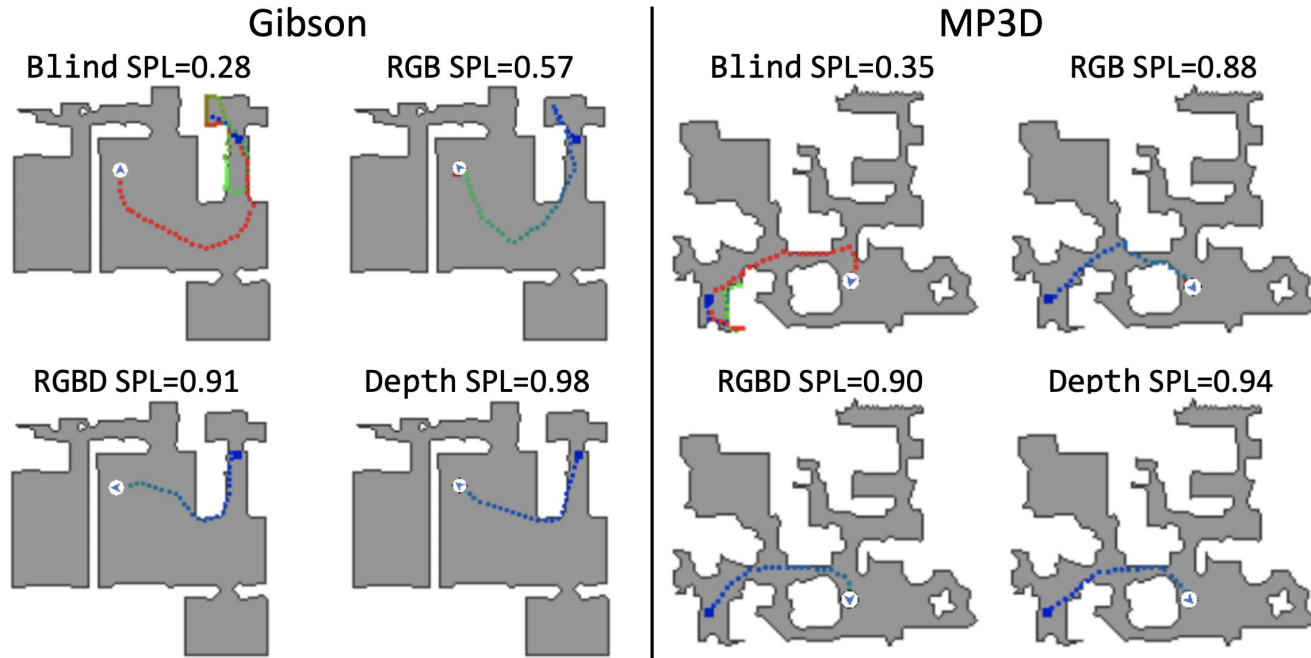
RGB sensor



Depth sensor



Compare performances of different datasets



All Methods Perform Better on Gibson than Matterport 3D

Performance of baseline methods for PointGoal task

		Gibson		MP3D		SUNCG	
Sensors	Baseline	SPL	Succ	SPL	Succ	SPL	Succ
Blind	Random	0.02	0.03	0.01	0.01	0.02	0.03
	Forward only	0.00	0.00	0.00	0.00	0.00	0.00
	Goal follower	0.23	0.23	0.12	0.12	0.23	0.24
	RL (PPO)	0.42	0.62	0.25	0.35	0.35	0.54
RGB	RL (PPO)	0.46	0.64	0.30	0.42	0.42	0.59
Depth	RL (PPO)	0.79	0.89	0.54	0.69	0.55	0.72
RGBD	RL (PPO)	0.70	0.80	0.42	0.53	0.42	0.57
	SLAM [19]	0.51	0.62	0.39	0.47	—	—

Generalization of the learned Agents

		Gibson	MP3D	SUNCG
Blind	Gibson	0.42	0.34	0.38
	MP3D	0.28	0.25	0.25
	SUNCG	0.35	0.29	0.35
RGB	Gibson	0.46	0.40	0.28
	MP3D	0.25	0.30	0.17
	SUNCG	0.25	0.24	0.42
Depth	Gibson	0.79	0.68	0.64
	MP3D	0.56	0.54	0.49
	SUNCG	0.50	0.48	0.55
RGBD	Gibson	0.70	0.53	0.48
	MP3D	0.44	0.42	0.35
	SUNCG	0.42	0.38	0.42

Each row is the average SPL for a model trained on the source dataset and then tested on another.

Generalization of Agents

		Gibson	MP3D	SUNCG
Blind	Gibson	0.42	0.34	0.38
	MP3D	0.28	0.25	0.25
	SUNCG	0.35	0.29	0.35
RGB	Gibson	0.46	0.40	0.28
	MP3D	0.25	0.30	0.17
	SUNCG	0.25	0.24	0.42
Depth	Gibson	0.79	0.68	0.64
	MP3D	0.56	0.54	0.49
	SUNCG	0.50	0.48	0.55
RGBD	Gibson	0.70	0.53	0.48
	MP3D	0.44	0.42	0.35
	SUNCG	0.42	0.38	0.42

- All agents suffer a drop in performance
- Agents trained on Gibson consistently outperform agents trained on other datasets
- Visual navigation agents can benefit from curriculum learning

Strengths

- When rendering a scene from the database, Habitat-Sim achieves several thousand fps running on single-thread and reaches over 10,000 fps multi-process on a single GPU.
- Conclude learning-based agents can match and exceed the performance of classical visual navigation methods when trained for long enough
- Habitat uses dynamic data over static data to train embodied agents.
- Can parallelize training of embodied agents.

Weaknesses

- The paper only focus on Point Goal Navigation without combining with other tasks
- Lack of analysis: Which kind of task is easier to learn? What are the successful cases? what are not successful cases? For what kinds of navigation tasks, you need more training?

Future work

New tasks, datasets and baselines

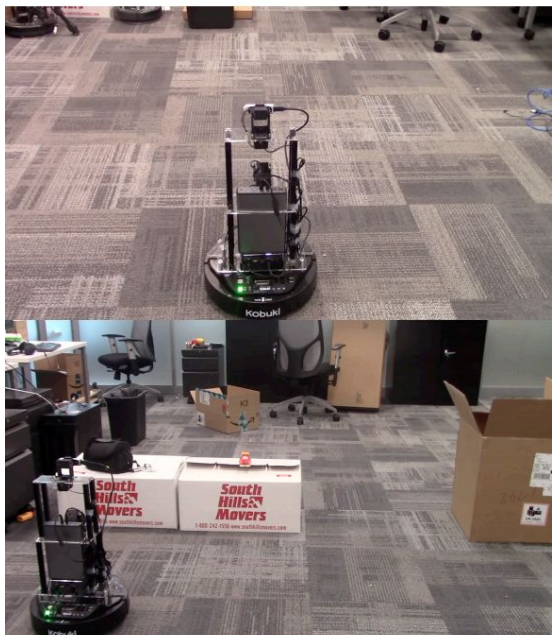
- PointGoal navigation
- Object navigation
- Room navigation
- Instruction navigation
- Embodied question answering



Slide adapted from *Oleksandr Maksymets*

Future work

Articulated robot integration



<https://pyrobot.org/>

- PyRobot provides a common API for different robots and uses Robot Operating System
- Sim2Real: train models in Habitat and evaluate on a real robot
- Incorporate an actuation and sensors noise models from real robots into Habitat
- Articulated agents format URDF & DART integration
- Standardize sim2real evaluation

Slide adapted from *Oleksandr Maksymets*

Future work

Object interactions and physics engine



- Dynamic objects: loading objects that can move
- Modular physics engines integration: Bullet, DART, PhysX
- Objects physical properties: mass distribution, coefficients of friction and restitution
- Interaction API: `apply_force`, `grip`, `twist` and etc.



Slide adapted from *Oleksandr Maksymets*