Following High-level Navigation Instructions on a Simulated Quadcopter with Imitation Learning

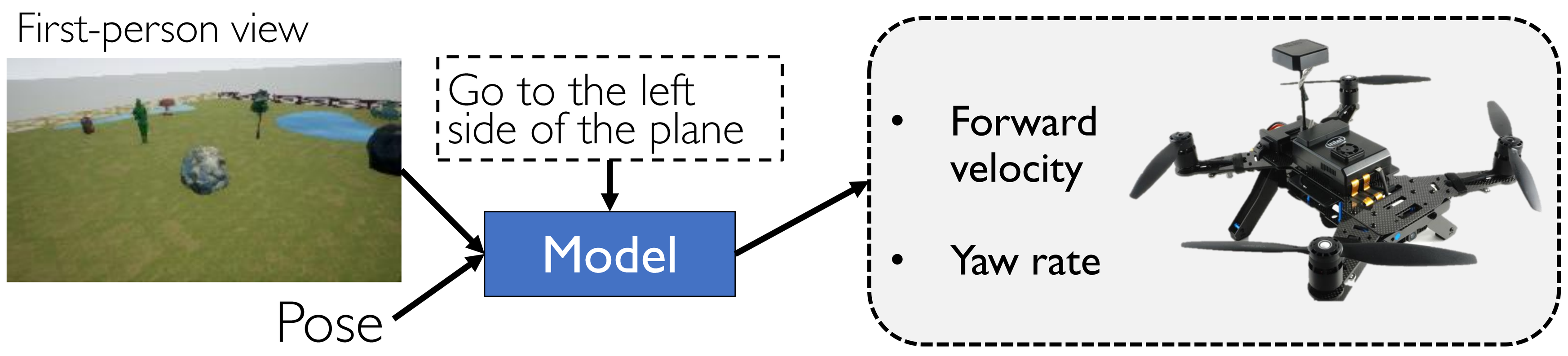
Valts Blukis, Nataly Brukhim, Andrew Bennet, Ross A. Knepper and Yoav Artzi

Problem Statement

Goal: map instructions and visual observations to actions

Control a quadcopter to execute the instruction and stop at the goal location

Agent observes: first-person camera images and pose estimates.
Output: continuous velocity commands.



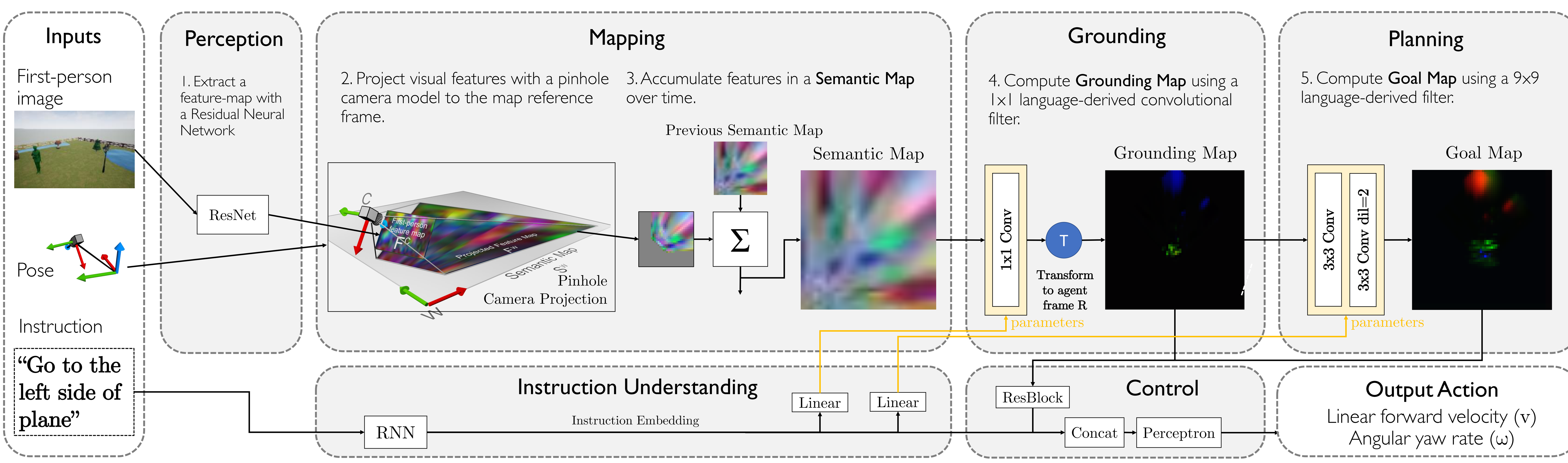
Two Main Types of Existing Approaches

1. **Decompose problem** in perception, instruction understanding, mapping, planning and control modules. Requires design of intermediate representations.
2. **End-to-end neural network** with recurrent and convolutional layers. Lack interpretability. Require large amount of training data. Difficult to handle constantly changing first-person observations.

Our Approach: combine the best of both approaches

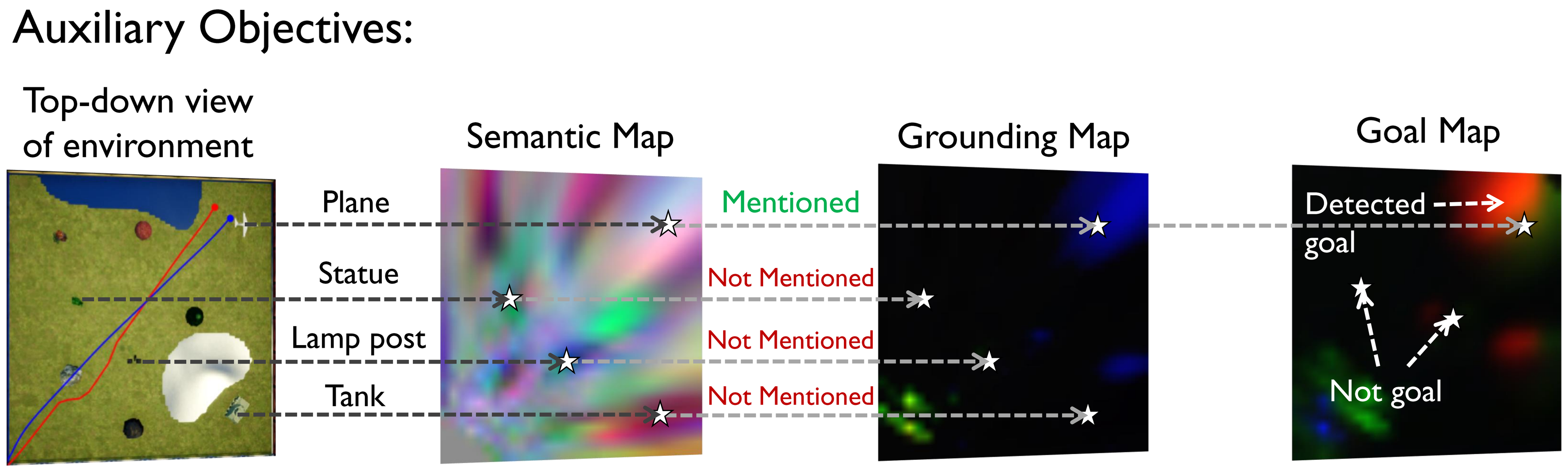
- **Neural network** architecture with mapping.
- **Mapping module** that explicitly builds an environment map.
- **Modular architecture** where each module has a specific function.
- **End-to-end training** prevents compounding of errors.

Grounded Semantic Mapping Network (GSMN)



Objective Function

Auxiliary Objectives:



$J_{\text{percept}}(\Theta)$: object classification in the **semantic map**.

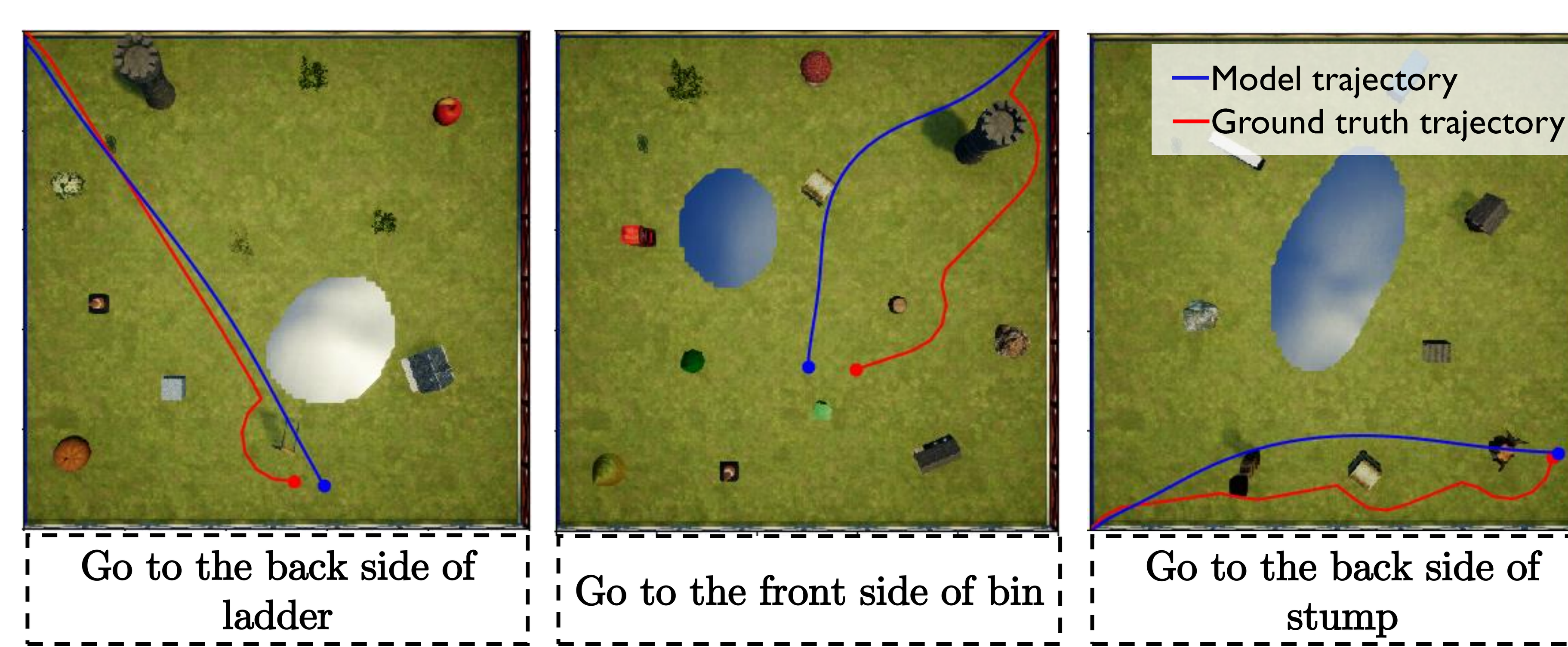
$J_{\text{ground}}(\Theta)$: instruction mention classification in the **grounding map**.

$J_{\text{plan}}(\Theta)$: performs goal location detection in the **goal map**.

Full Objective Function:

$$J(\Theta) = \underbrace{J_{\text{act}}(\Theta)}_{\text{Task}} + \underbrace{\lambda_v J_{\text{percept}}(\Theta) + \lambda_l J_{\text{lang}}(\Theta) + \lambda_g J_{\text{ground}}(\Theta) + \lambda_p J_{\text{plan}}(\Theta)}_{\text{Auxiliary Objectives}}$$

Experimental Setup



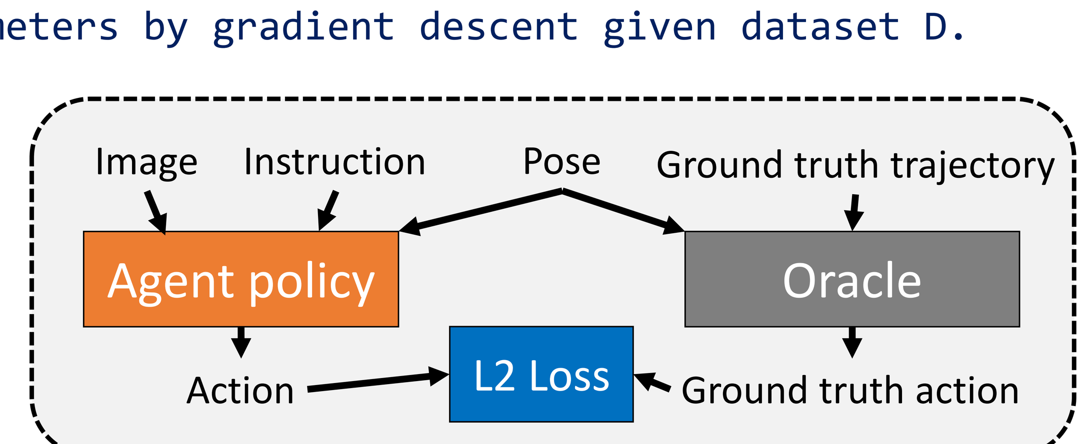
- 3500/750/750 environments-instruction pairs for training/development/testing.
- 63 different objects, 6 to 13 per environment.
- Realistic-dynamics simulator based on Microsoft Airsim and Unreal Engine.

Imitation Learning with DAggerFM

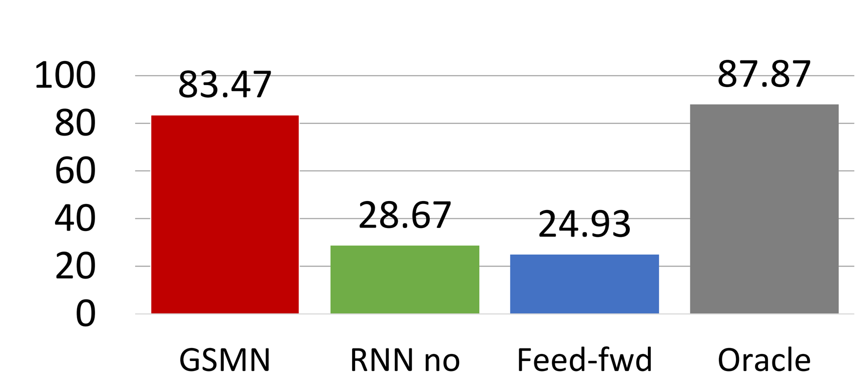
- Learn by imitating actions given by oracle policy:

```
1. Execute oracle and collect dataset D of trajectories, each a sequence
   of observations and ground-truth actions.
2. Loop:
  2.1. Drop N trajectories from D.
  2.2. Execute agent policy to collect N trajectories and add to D.
       Every observation is annotated with ground-truth oracle actions.
  2.3. Update policy parameters by gradient descent given dataset D.
```

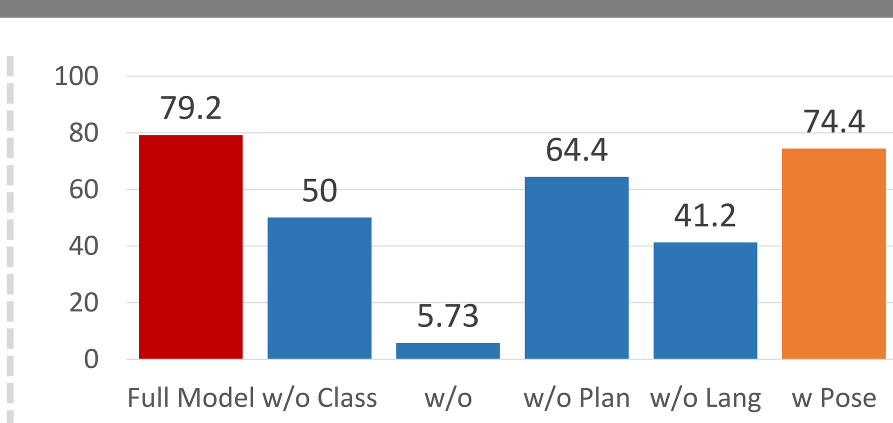
- Dataset D does not grow.
- Oracle is a simple carrot-planner control rule following ground truth trajectories.



Results and Analysis



Sucess Rate (%) on heldout test data



Sucess Rate (%) on development data

- Our model outperforms traditional neural architecture where instead of building a map, a recurrent network is used as memory.
- Almost reaches oracle performance.
- Resilient to noise in position estimates (0.5m std dev, map is 30x30m).
- Grounding auxiliary $J_{\text{ground}}(\Theta)$ is essential in our few-sample regime.
- Foal-prediction auxiliary $J_{\text{plan}}(\Theta)$ is not essential for simple instructions.