

From language to autonomous robot action: An integrated system

Dan Brooks, Constantine
Lignos, Abe Shultz

The known world scenario

Scenario: a badguy has taken hostages and placed bombs in a building. The robot must:

- Locate a goodguy that has the bomb defuser and retrieve it from them
- Use the defuser to defuse bombs
- Report what is seen along the way

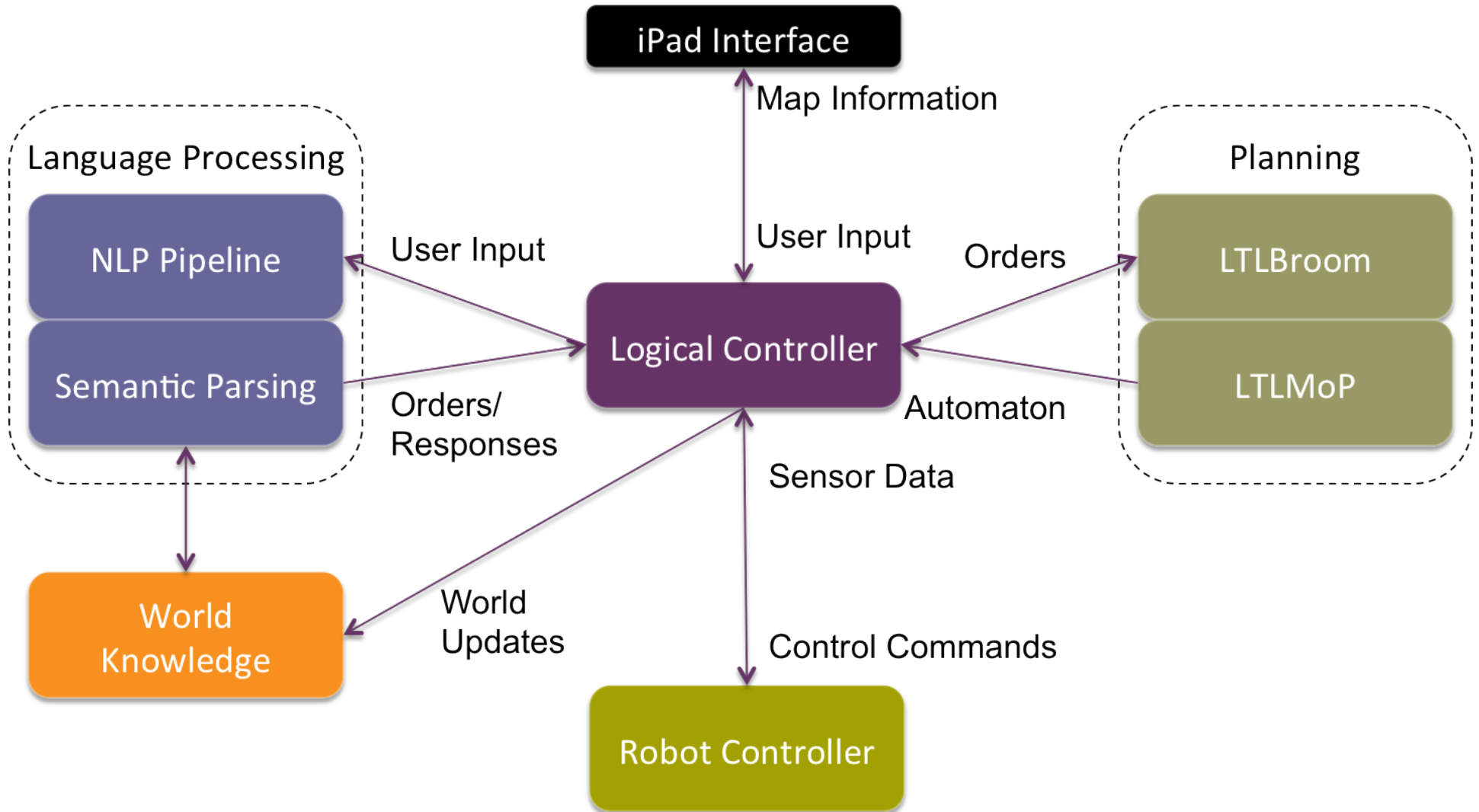
The system should provide:

- Rich communication between commander and JR
- Ability to understand simple natural commands

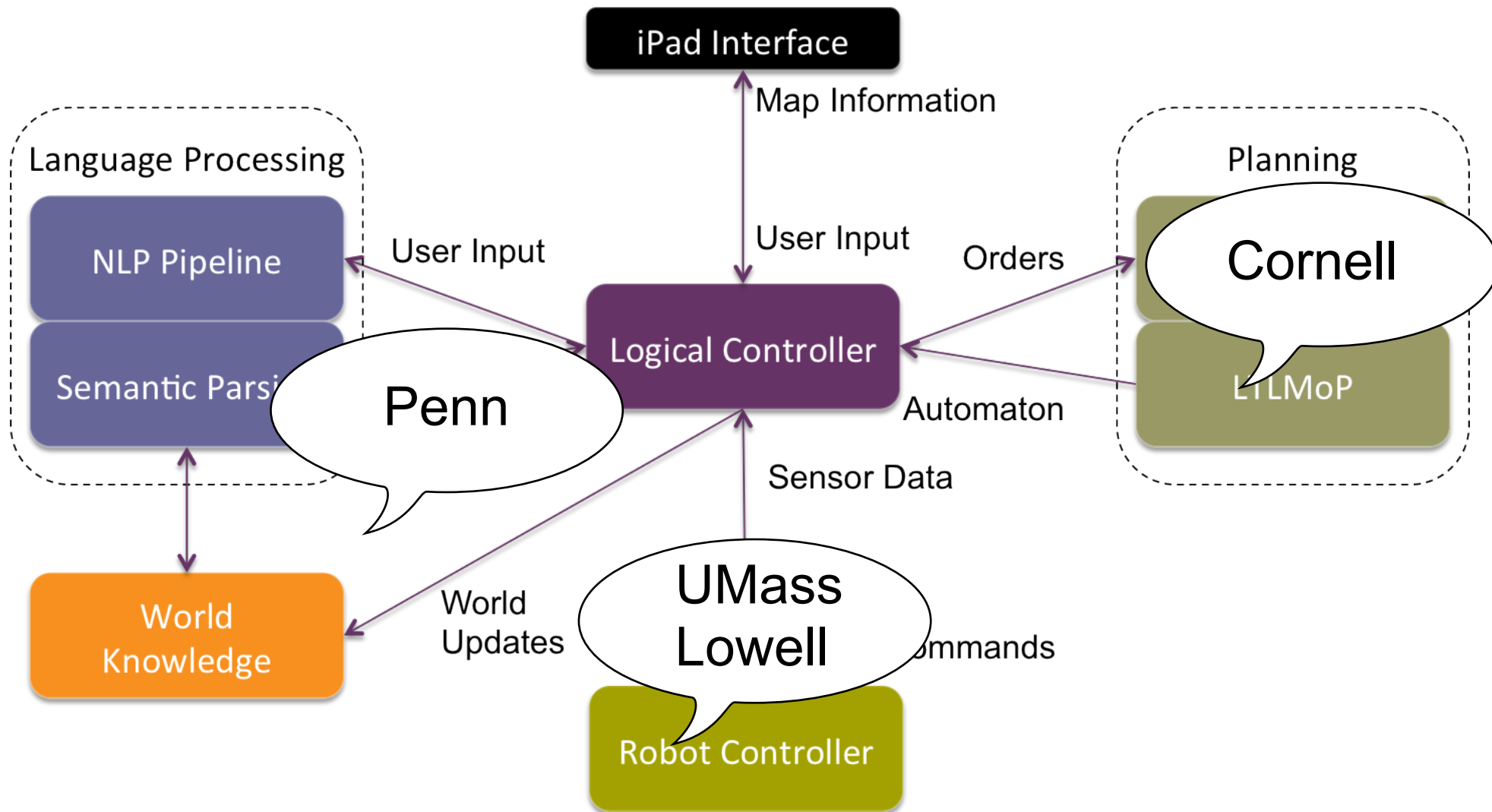
Assumptions:

- The map is known
- Actuators and sensing are simplified

Architecture, as seen from above



It takes a village to raise a robot!



A quick demonstration

- The commander gives instructions to JR by typing them into the iPad
 - iPad provides text input and shows updates from the logical controller
- When the commander is done giving orders:
 - Orders are used to generate an LTL specification
 - Specification is compiled into a finite state automaton
 - Automaton is executed
- Actions you'll see
 - Asking the robot to notify if something is seen
 - Following a target
 - Specifying a plan

Language to action

Commander's instructions

```
graph TD; A[Commander's instructions] --> B[Parsed sentences]; B --> C[Semantic representation]; C --> D[MetaPAR/LTL]; D --> E[Automaton];
```

The diagram illustrates a five-step process for converting natural language into an automaton. It consists of five horizontal bars arranged in a descending staircase pattern from top-left to bottom-right. Each bar is connected to the next by a grey downward-pointing arrow. The bars are colored in a gradient from dark purple at the top to grey at the bottom. The text inside each bar is white.

Parsed sentences

Semantic representation

MetaPAR/LTL

Automaton

Processing commands from semantics

- Semantic outputs actions and their arguments:
 - *Go*(Location: Hallway)
 - *Defuse*(Theme: Bomb)
 - *Eat*(Patient: Sandwich)
- Filter down to commands that match the robot's abilities
 - Going to rooms, getting objects, defusing bombs, searching
 - ARL Cognitive Robotics: research regarding using long-term memory to work with unknown actions
- Some fixed pragmatics is involved:
 - If you have orders that require you to act on objects you detect, you should search rooms you go to

Specifying a plan using LTL

Linear Temporal Logic (LTL)

- Use three temporal concepts: *next*, *always*, *eventually*.
- Use logic to control the state space and produce a verifiably correct controller

Examples:

- Room 1 is adjacent to rooms 2 and 3: *If you're in room 1, next you will be in rooms 1, 2, or 3.*
- If you see a bomb, defuse it: *Always, if and only if you see a bomb, defuse it.*
- Search all the rooms and come back to room 1: *Always, if and only if you've searched rooms 1, 2, and 3 and you're in room 1, you're done.*

LTL generation (LTLBroom)

- Plans may vary in complexity:
 - Go to the hallway.
 - Search the lab.
 - Defuse any bombs you see.
 - Look for User 1 in the library and the classroom. Tell me if you see User 1. Get the bomb defuser from User 1 and come back here.
- We translate these into goals that include:
 - Objects to get
 - Rooms to search
 - Rooms to go to
 - Standing orders based on sensors
- LTLBroom generates an LTL specification using these

Example: Action of getting an object

1. Orders: Get the bomb defuser from User 1.
2. Semantics: *Get*(Theme: Bomb defuser, Source: User 1)
3. Transform into LTL:

$\square \text{ (!(s.have_defuser} \mid \text{next(s.have_defuser))} \ \& \ \text{next(e.user)}) \leftrightarrow \text{next(s.get_defuser)}$

Always, get the defuser if and only if you don't have it and the user is in the room.

$\square \text{ ((s.get_defuser} \ \& \ \text{next(e.get_defuser_done))} \mid \text{s.have_defuser}) \leftrightarrow \text{next(s.have_defuser)}$

Always, you have the defuser if and only if either you have it or you were getting it and now you're done getting it.

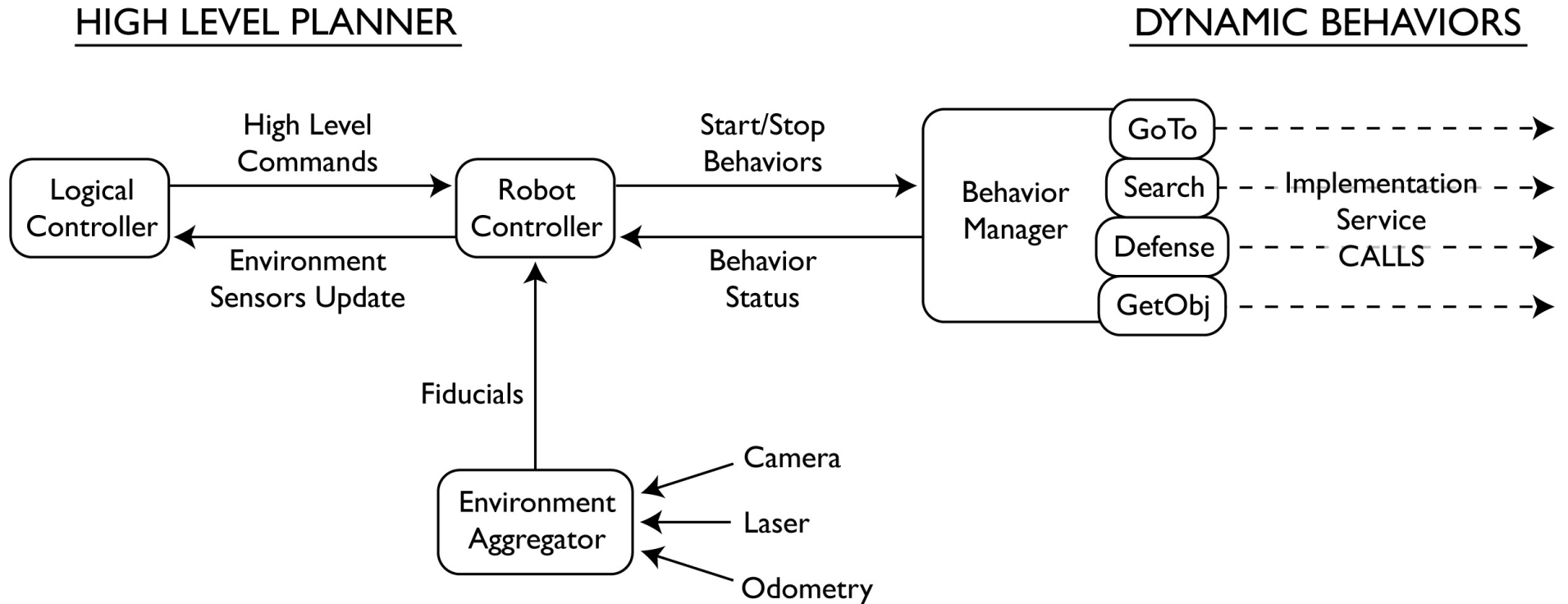
$\square \text{ s.have_defuser} \leftrightarrow \text{s.done}$

Always, you are done if and only if you have the defuser.

Executing the automaton

- LTLMoP synthesizes an automation from the specification
- When orders change, or the topology changes, a new automaton is synthesized
 - We support "hot" resynthesis in the middle of actions
- The logical controller executes it
 - Uses sensor input to select next states
 - Sends commands to the robot as needed:
 - Move to room X
 - Search
 - Defuse
 - Get object X
- These commands are carried out by the robot controller

Architecture, as seen from below



Robot Controller - High Level Commands

- Drive To Region
- Search Current Room
- Follow Me
- Get Object
- Defuse Bomb
- Explore *

Robot Controller - Environment Sensors

Environment Sensor Rules

- Report objects you "see" in your current region
- Continue reporting the object until you leave the region
- Report if the object changes it's location
- Stop reporting if the object leaves the region

Environment "Scenes"

- Appeared
- Visible
- Moved
- Disappeared

Robot Controller - Actions

Actions

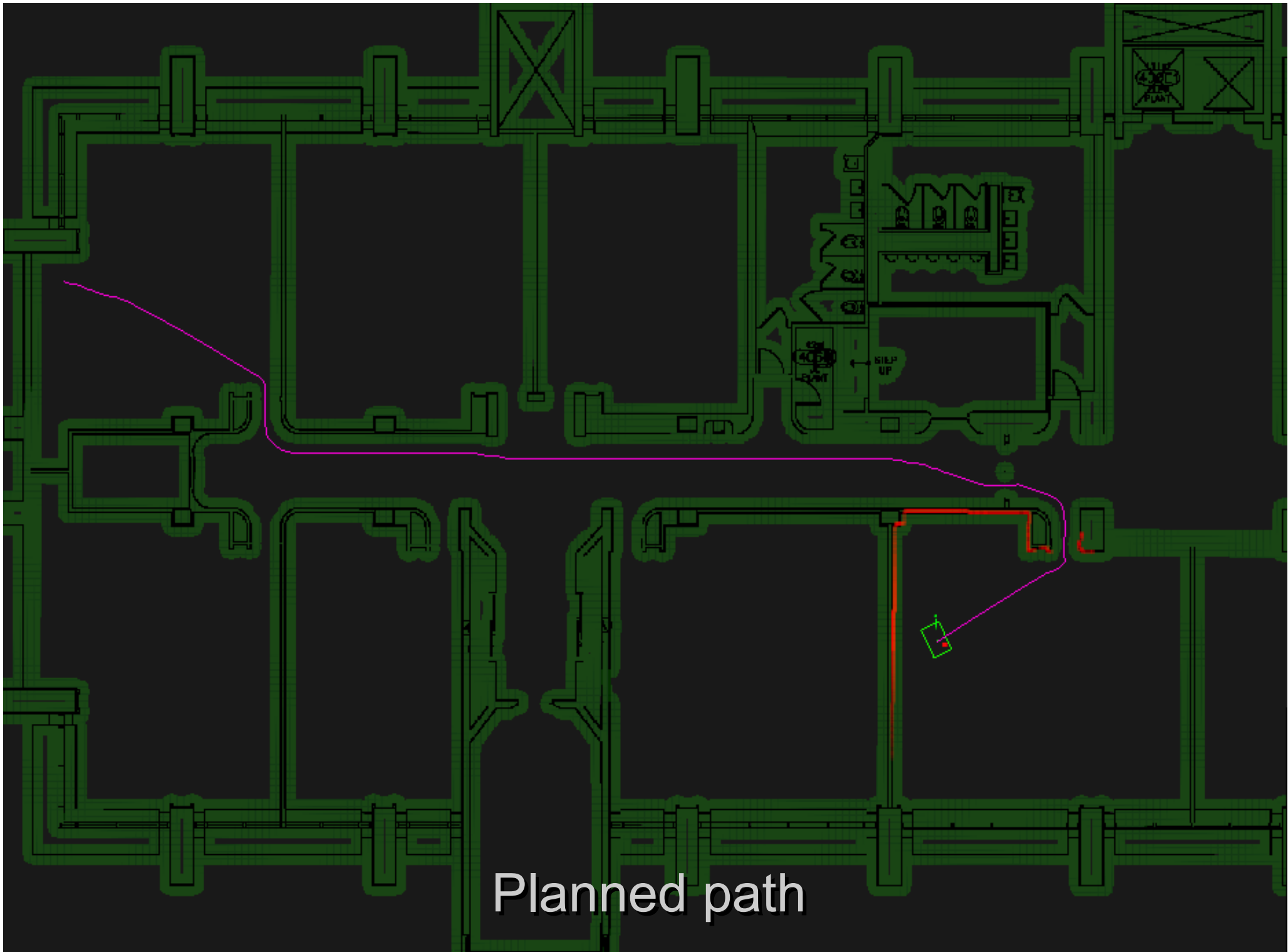
- Basic activities the robot is capable of performing
- Implemented using ROS ActionLib (Client/Server)
- Could be run in parallel

Implemented Actions

- DriveTo(Location)
- AreaSweep
- Follow

Robot Controller - DriveTo

- Destination specified by name of room or object
- Translate name into coordinates
- Generate waypoint path to coordinates
- Use local obstacle avoidance to get between waypoints



Robot Controller - Area Sweep

- Goal is to "see" the entire room
- Drops a "check point" at starting location
- Performs an intelligent "wall follow"
- Robot stays inside current region (does not go through doors)
- A "timeout" can be set to cancel the sweep



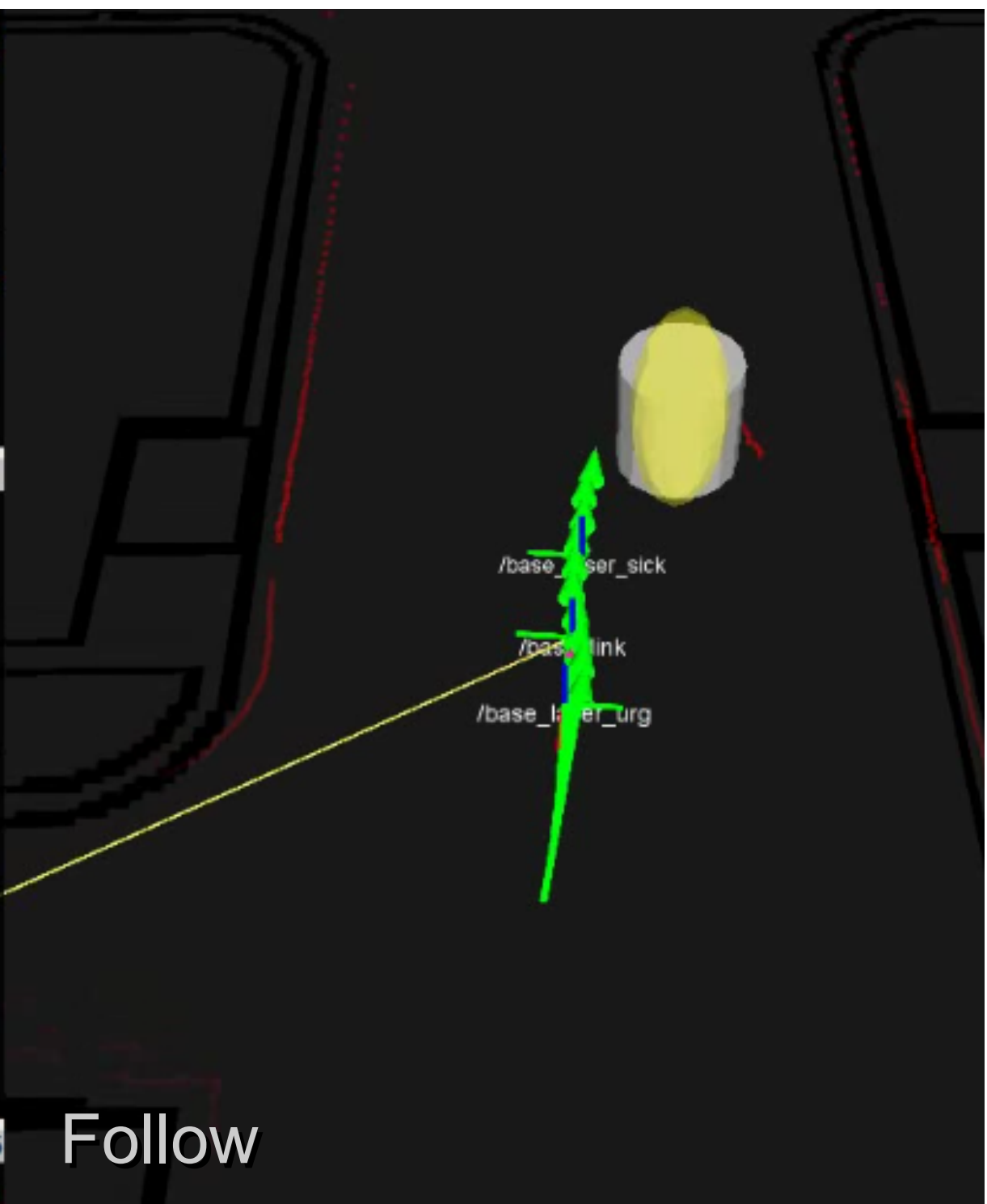
The image displays a floor plan with a central room. A green path, representing an area sweep, starts at the bottom center of the room, moves left along the bottom wall, then up along the left wall, and finally right along the top wall. A red path, representing wall following, starts at the bottom center, moves right along the bottom wall, then up along the right wall, and finally left along the top wall. A small green rectangle is located near the bottom center of the room, and a small red 'L' shape is located near the top center. The floor plan includes various rooms, corridors, and structural elements like walls and doors.

Area Sweep as wall follow

Robot Controller - Follow

- Can follow any object detected
- Attempts to maintain a specific distance from target
- Times out after loosing "sight" of the target
- Option to use Blob detection

```
you have set map parameters, but also requested to be
overwritten by those given by the map server [Costm
received a 1312 X 954 map at 0.033400 m/pix
time period is set to 0.05 [TrajectoryPlannerROS::in
all waiting for move_base [GlobalDriver::GlobalDri
map update loop missed its desired rate of 2.0000Hz
[stmap2DRos::mapUpdateLoop]
]: Received a target: yellow-shirt [FollowServer::
]: Target acquired [FollowServer::executeCallback]
```



```
subtle/ros-marco [cmvision+re.../launch) - VIM 34x5
rosrun cmvision cmvision
```

Follow

Demonstration

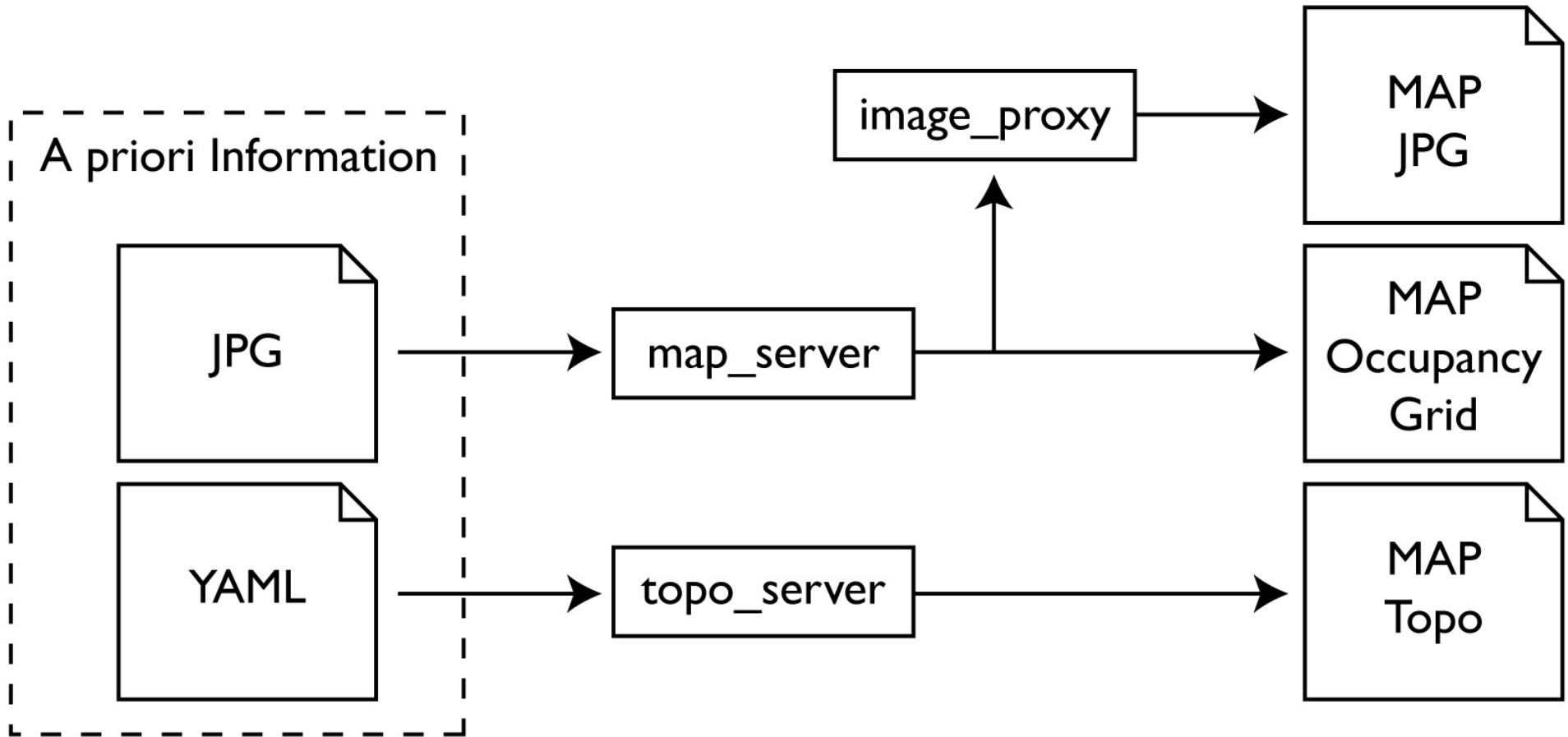
Today in video, but if you fancy a trip to Lowell, you can see it in person!

Frontier Exploration - Challenges

- What does the world look like?
- How do we talk about places only recently discovered?
- How do you answer questions like "How do I get to you?"
- How do we incorporate all these changes into the system?

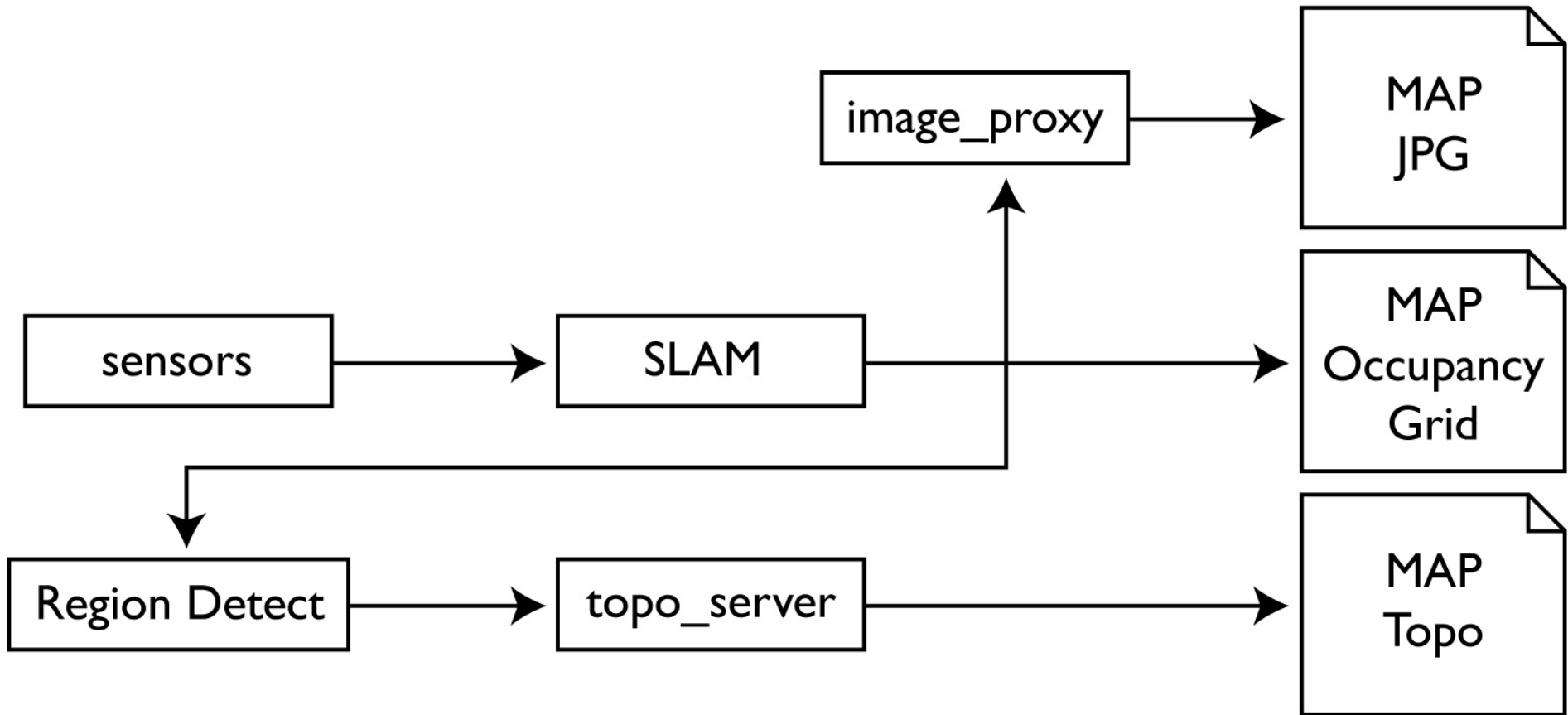
Flexible Design - Map Services Example

Known World Map Services



Flexible Design - Map Services Example

Frontier World Map Services



Frontier Exploration - Map Building

- Uses "area sweep" behavior in starting area
- Remembers open doorways as places to come back to
- When finished, uses "drive to" behavior to go to doors
- Uses "area sweep" on new area.
- Video demo

Room Detection

Uses morphological operators based on computer vision techniques.

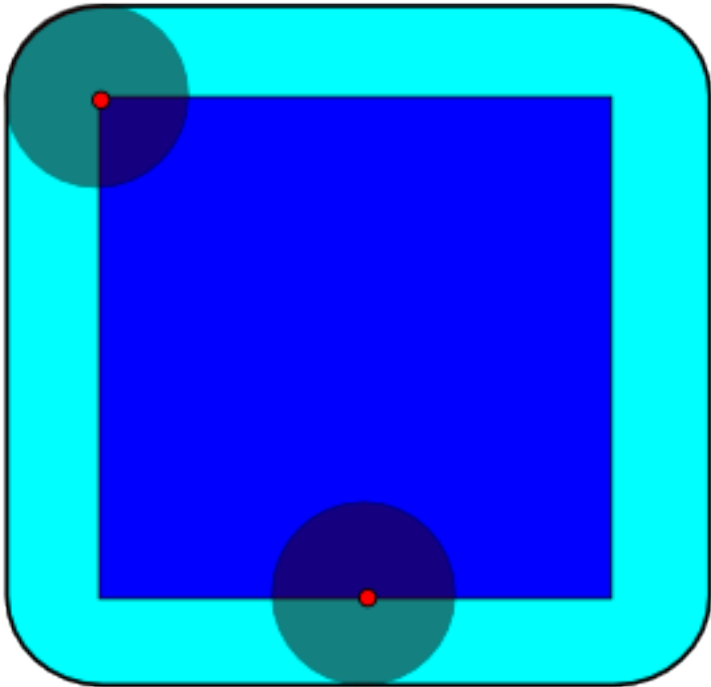
Segments map by detecting regions of open space separated by narrow passages.

Combines regions that are determined to be linked.

Fabrizi, E.; Saffiotti, A.; , "Extracting topology-based maps from gridmaps," *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on* , vol.3, no., pp.2972-2978 vol.3, 2000

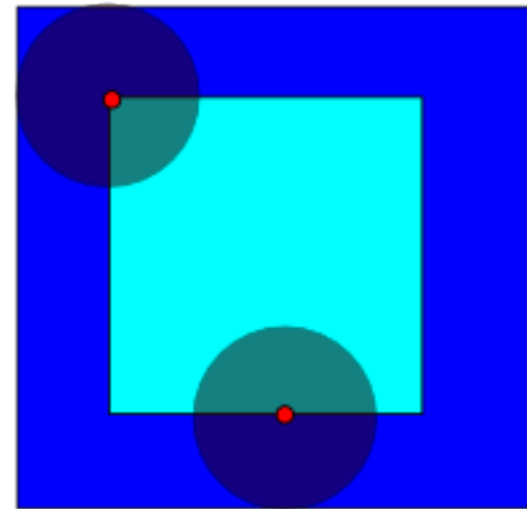
Room Detection

Dilation



The shape, plus all points covered by structuring element.

Erosion



The shape, less all points covered by center of structuring element.

Room Detection

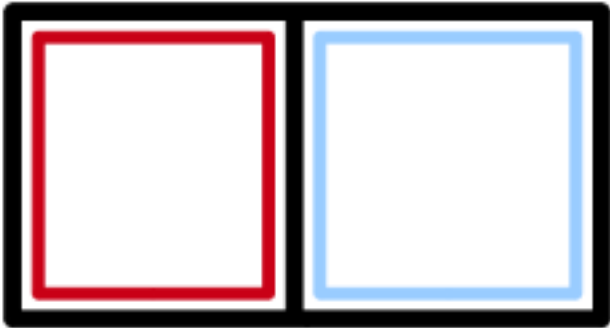


Closure

Dilate to close gaps, then erode to recover room space

Closes all gaps smaller than structuring element width

Room Detection



Detect contours



Use detected contours as seeds for watershed algorithm



Apply original borders and look for area transitions to build adjacency map.

Room Detection

Detected areas are assigned IDs.

User can assign new IDs to make dialog more natural.

List of previously seen regions and IDs is used to check if a new region is an addition to a pre-existing region.

Works best for human-oriented, constructed environments (buildings, not forests).

Direction Generation

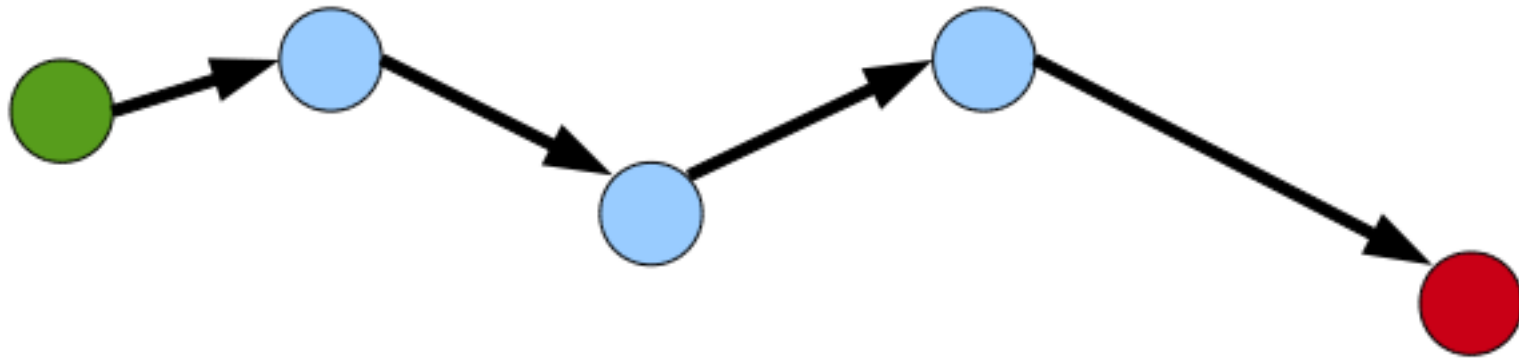
Answering the question "How do I get there?"

"There" can be where the robot is, or some place it has seen.

Good for when human intervention is needed in a location the robot has found.

Direction Generation

Path described as a set of turns interspersed with traveling in a straight line.



From each point, get the angle to the next one.

Angles expressed as "turn left/right" or "bear left/right" depending on magnitude of turn.

Direction Generation

Two methods:

- Pre-existing map of connected points

 - Good if you know the entire map *a priori*

- Wavefront planner

 - Good if you only have a partial map

iPad - Multimodal Communication

iPad 7:33 PM 59%

OPTIONS connect disconnect ENTER / SEND MESSAGE EDIT ROOM NAME

lab office kitchen
classroom library hall lounge

x Cmdr: Get the bomb defuser from User 2 and come back here.
Cmdr: That's all.
Jr: I shall endeavor to make it so.
Jr: I see a bomb.
Jr: I see a user_2.

Ongoing work

- Completing integration and testing of frontier-based exploration
 - Architecture and components are completed
- User interface/"polish" improvements
 - Better feedback from language components

Questions?