

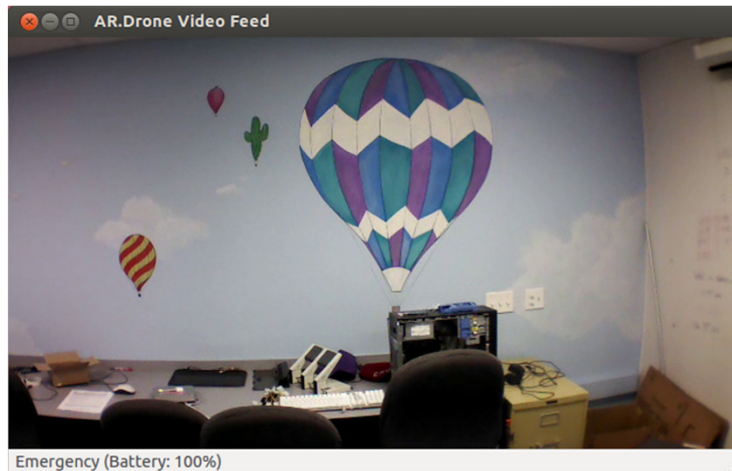
# *Interdimensional* Robot Cooperation



Kristina Ming, Chris Eriksen, *Graham Montgomery*, Jerry Hsiung, Cyrus Huang, Zakkai Davidson, and Zach Dodds

# Now...

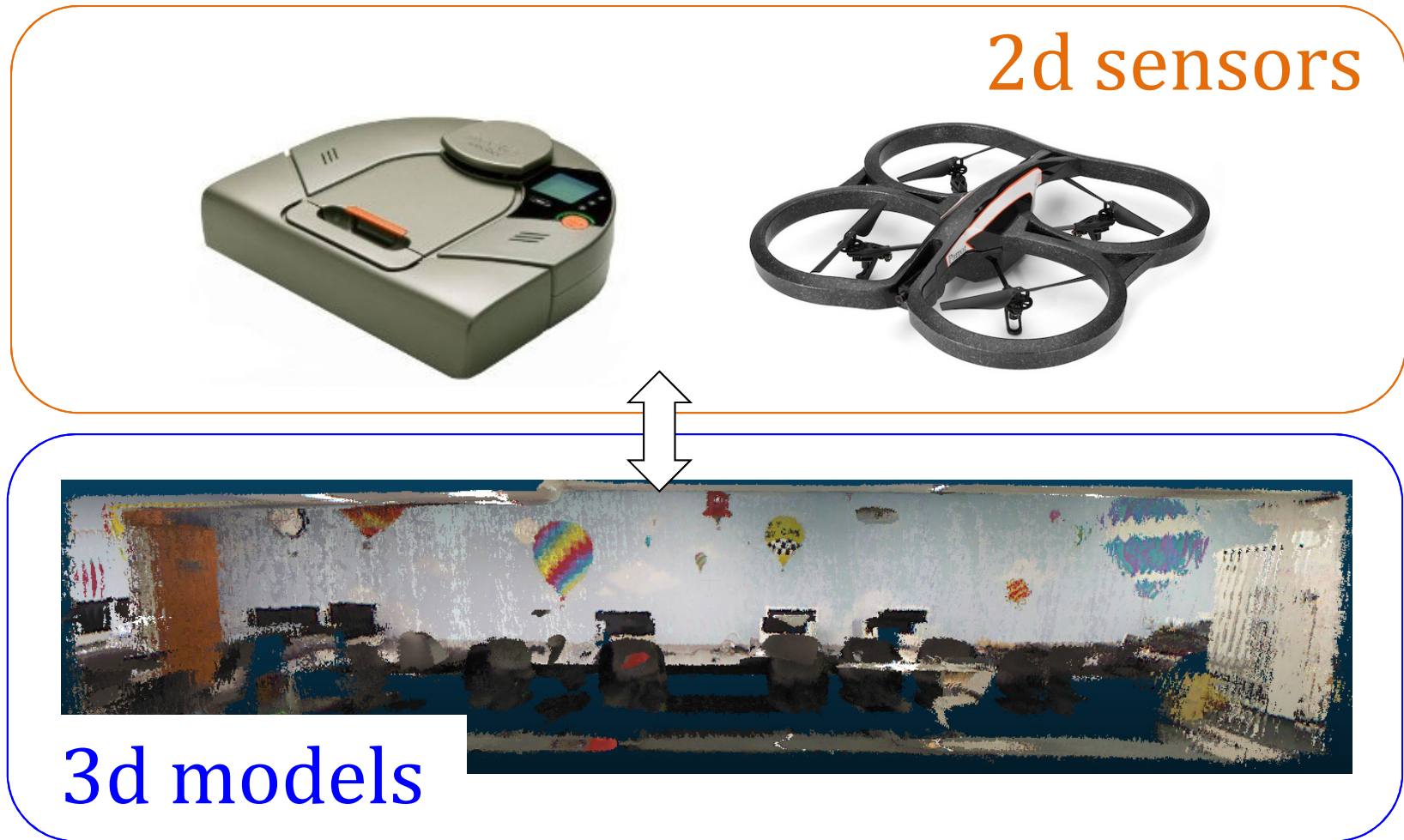
3d representations  
are a new standard  
for robot spatial  
reasoning, ...



... yet 2d image sensing  
is still common – *and*  
*will certainly continue to*  
*be for a long time.*

## ... vs. Then

# Cooperating?!



***Our goal:*** to investigate robot platforms, software, and algorithms for ***using 2d sensors amid 3d models*** of the world.

# Two concrete challenges...

more *robust* navigation



more *general* "lab escape"



bridging 2d sensors with 3d spatial representations

Jerry, Cyrus, Zakkai

# New platform: *Neato*



## Initial tasks

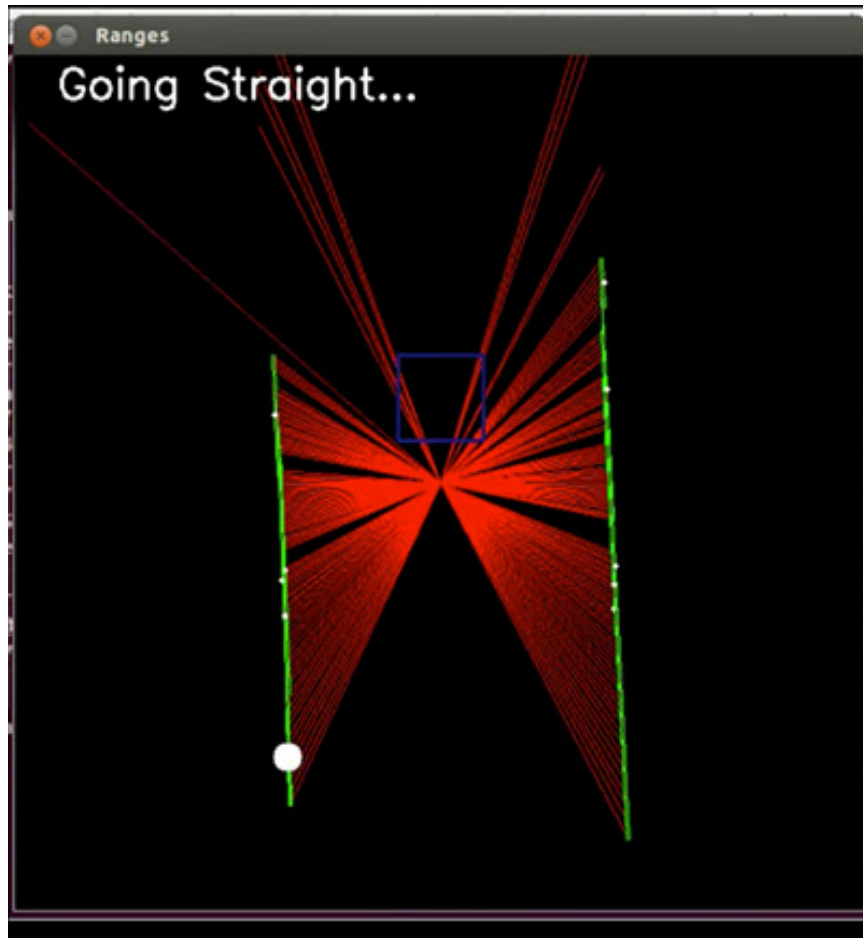
- **learn ASCII API**
- **write device drivers**
- interpret sensor data
- follow corridors
- detect intersections

## AI Tasks

- pt-to-pt navigating
- localize and navigate
- minimize ambiguity
- navigate predictively
- build 2d and 3d maps

**Objective:** to get this robot **running** and **reasoning**

# New platform: *Neato*



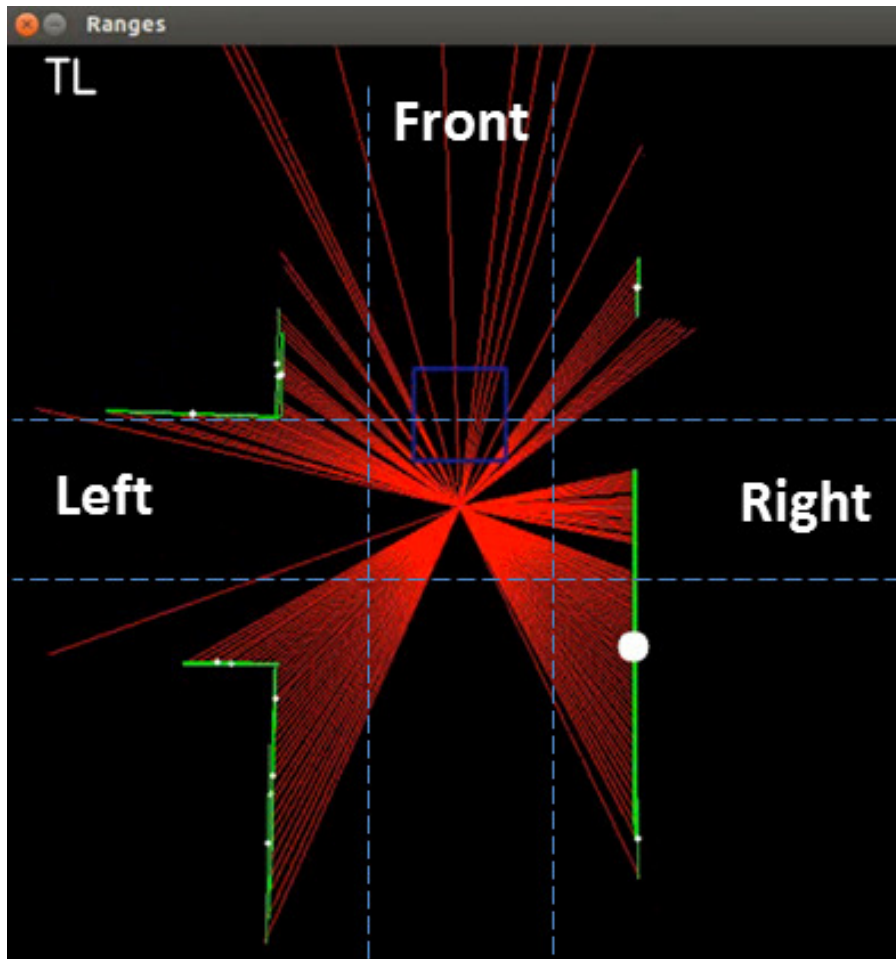
## Initial tasks

- learn ASCII API
- write device drivers
- **interpret sensor data**
- **follow corridors**
- detect intersections



laser-scan data, estimated walls, "*aligning*" wall

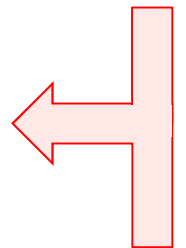
# New platform: *Neato*



## Initial tasks

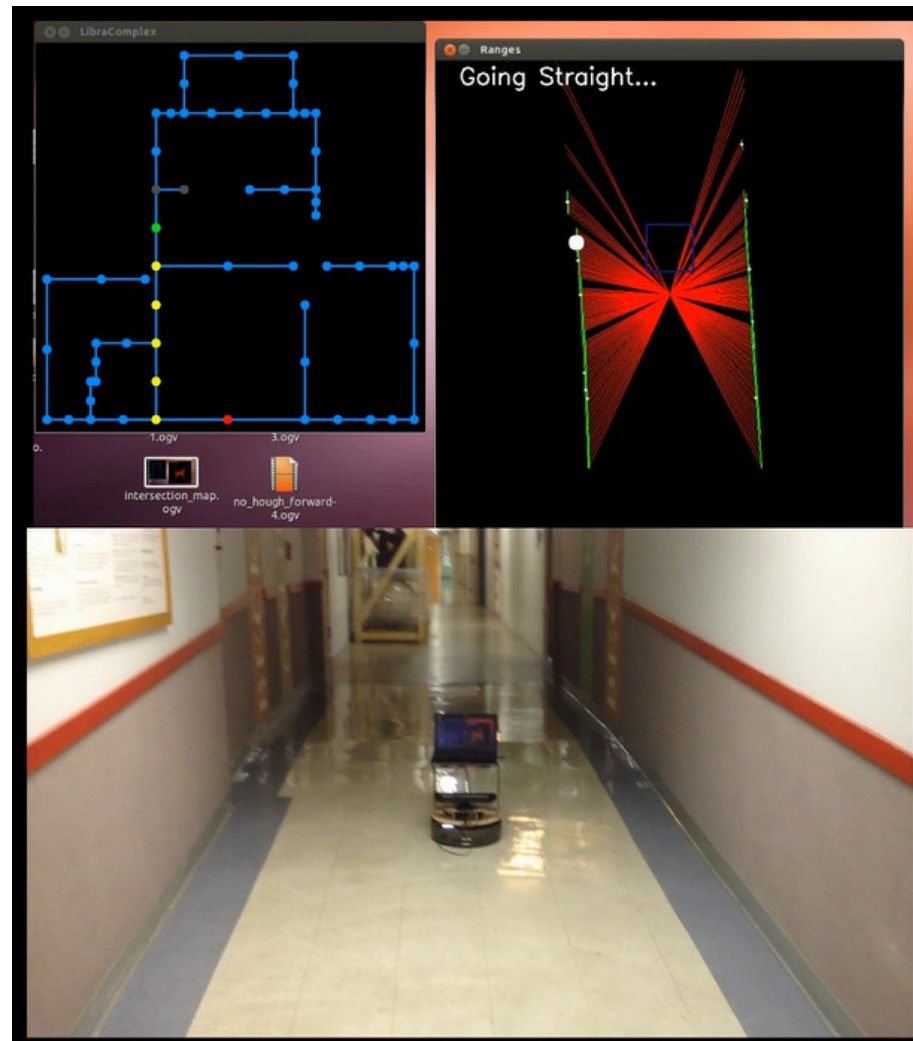
- learn ASCII API
- write device drivers
- interpret sensor data
- follow corridors
- **detect intersections**

This scan shows a  
"TL" intersection



wall-placement determines intersection type

# Neato *navigation*

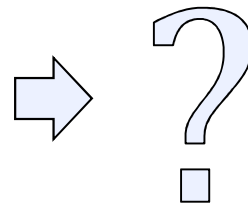
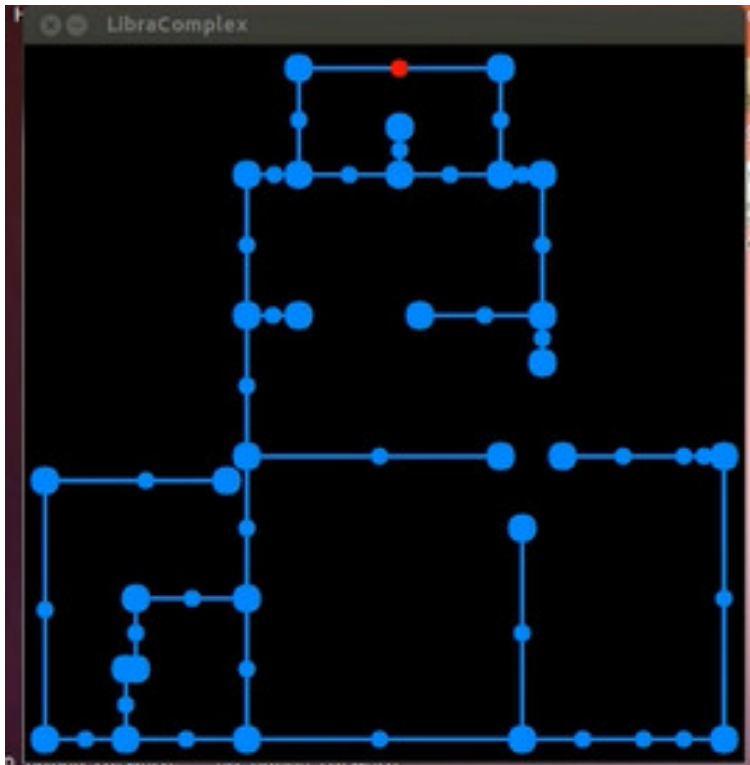


robot\_navigation\_known\_start

navigating the Libra complex, *with the start node given*

# Localization

*What if the robot doesn't know its starting pose?*



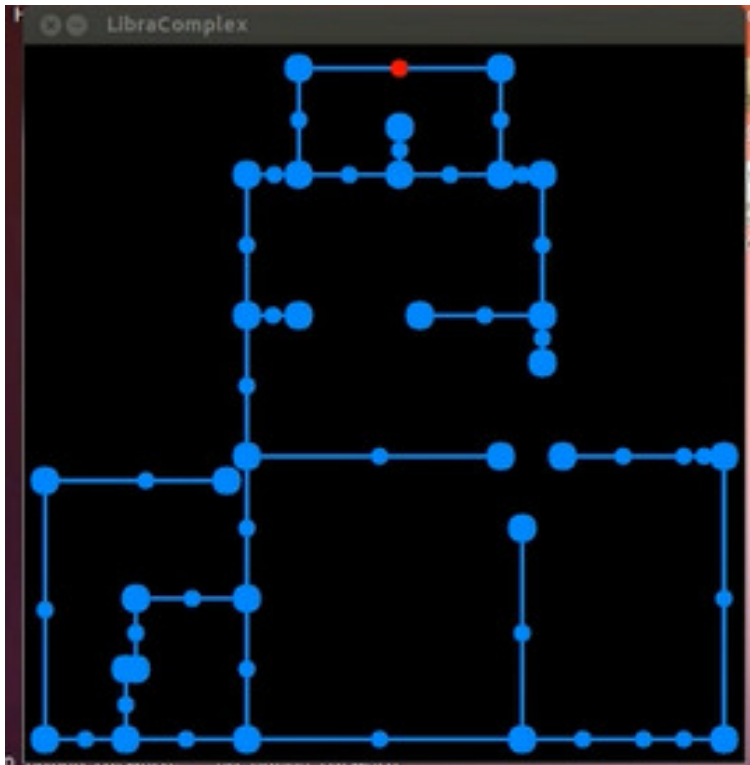
Suppose the robot  
moves and sees  
**a "TL"**

*Which nodes remain?  
Which are culled?*

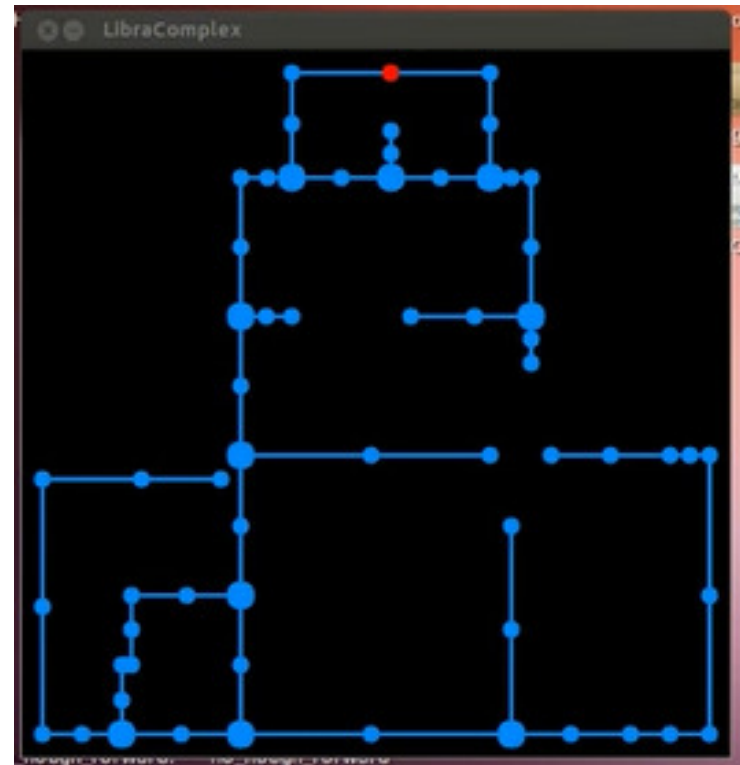
*Don't worry about the goal.  
Instead, make default motions...*

# Localization

*What if the robot doesn't know its starting pose?*



**TL**



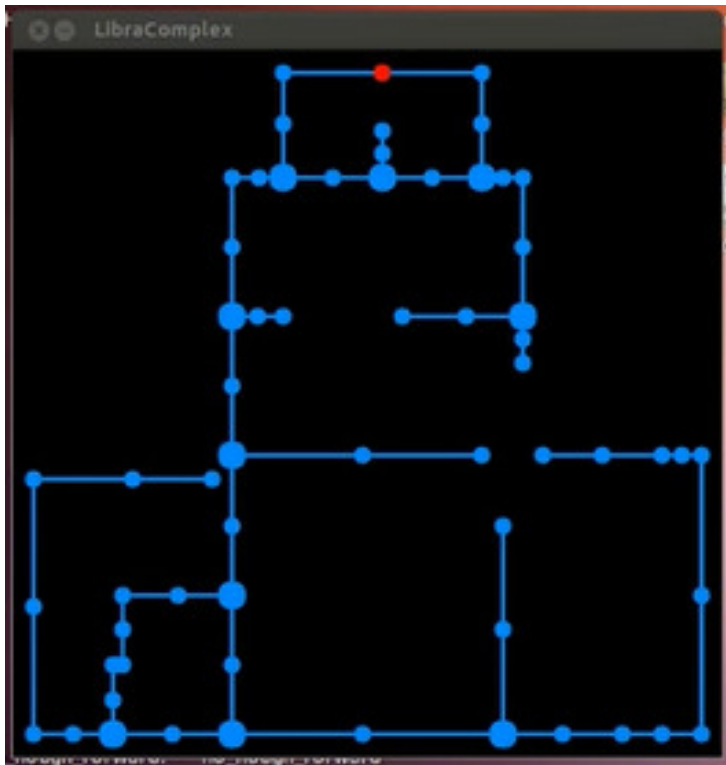
*Don't* worry about the goal.  
Instead, make default motions...

With each observation, we  
cull the robot's possible poses.

**TL**

# Localization

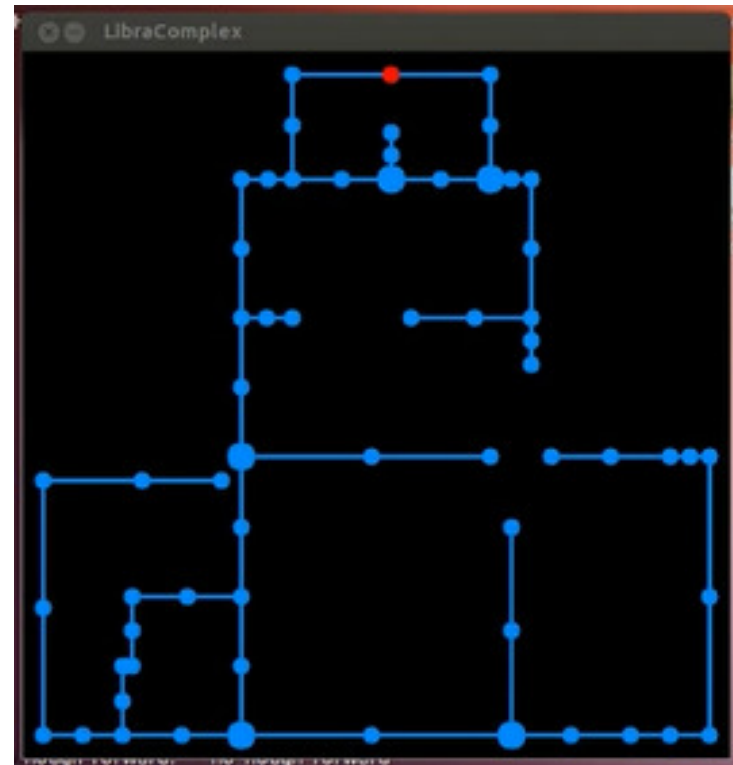
*Keep going...*



previous distribution of poses



**TL**

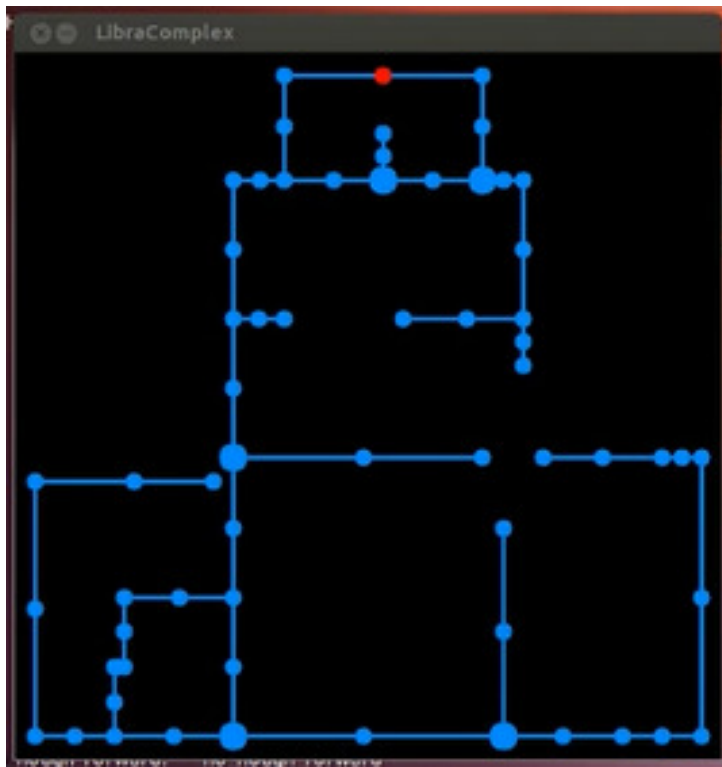


... cull again

**TL**

# Localization

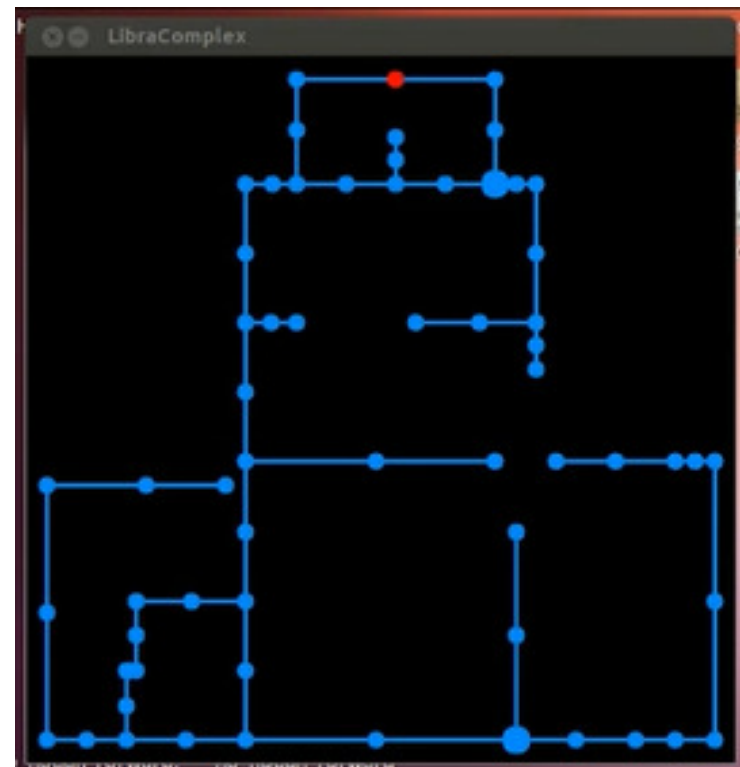
*When it knows its location, head to the goal!*



previous distribution of poses



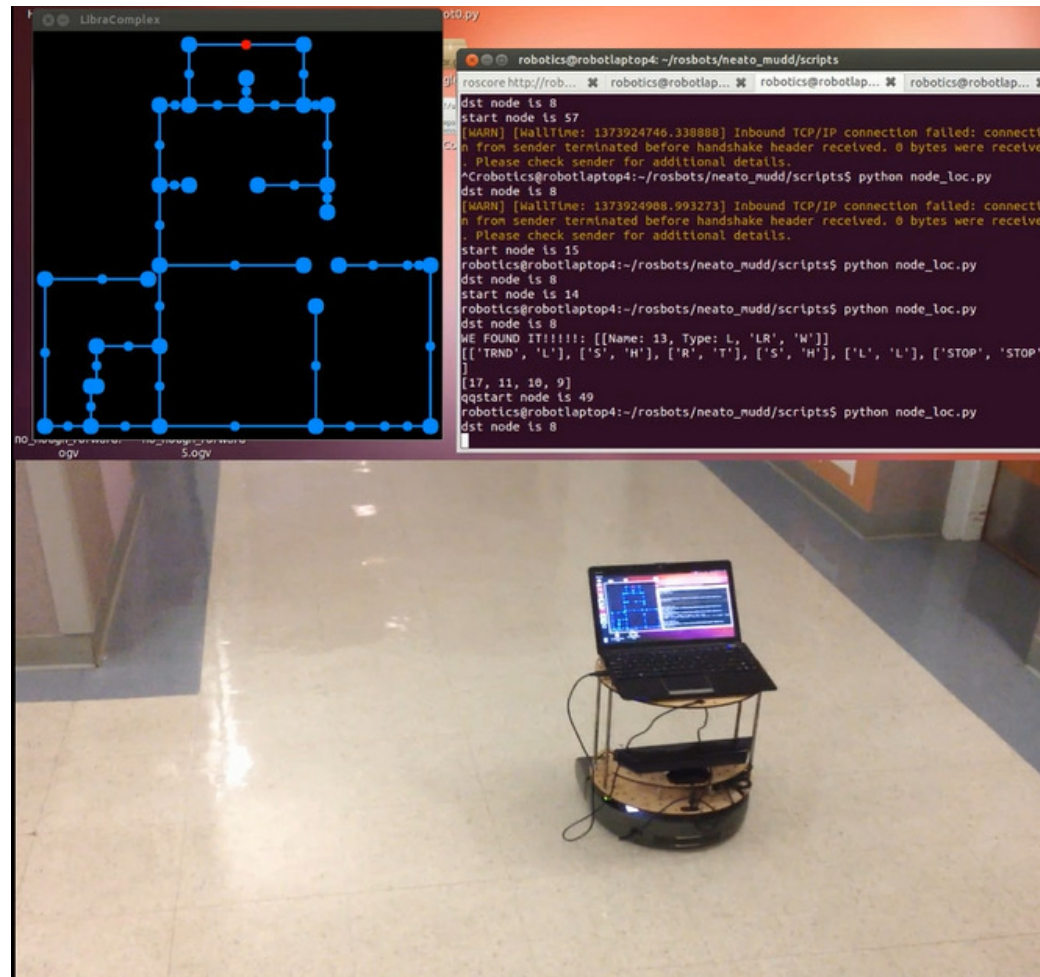
**TL**



... cull again

**TL**

# Markov localization



loc\_short.mp4

navigating to a goal without the start node given

# Can we be more *deliberate*?

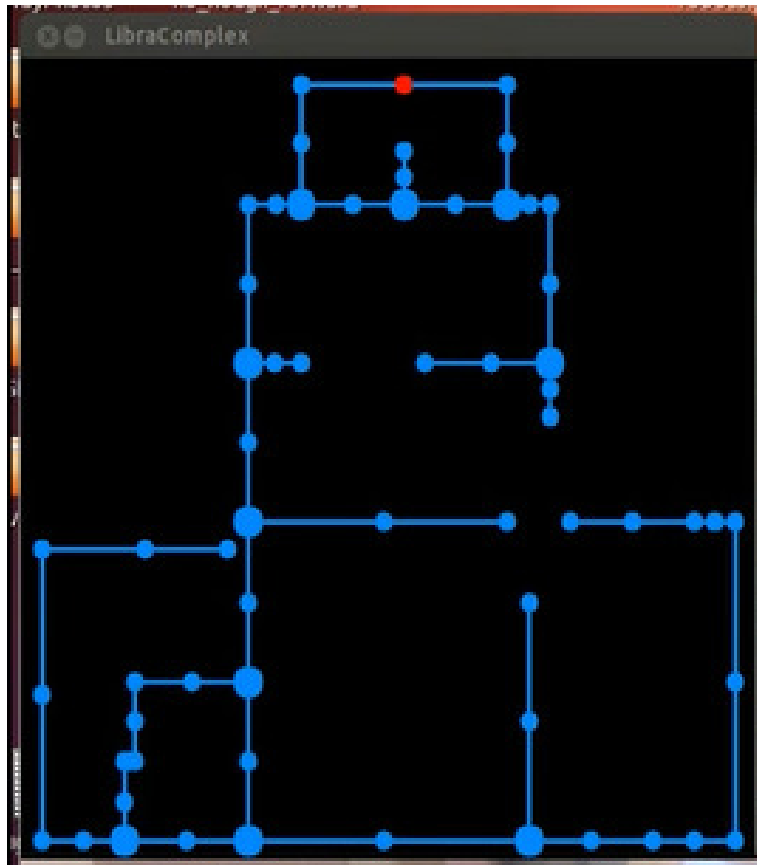
Each intersection offers a choice:

*Which turn to make, even if the robot isn't sure where it is?*

# Can we be more *deliberate*?

Each intersection offers a choice:

*Which turn to make, even if the robot isn't sure where it is?*

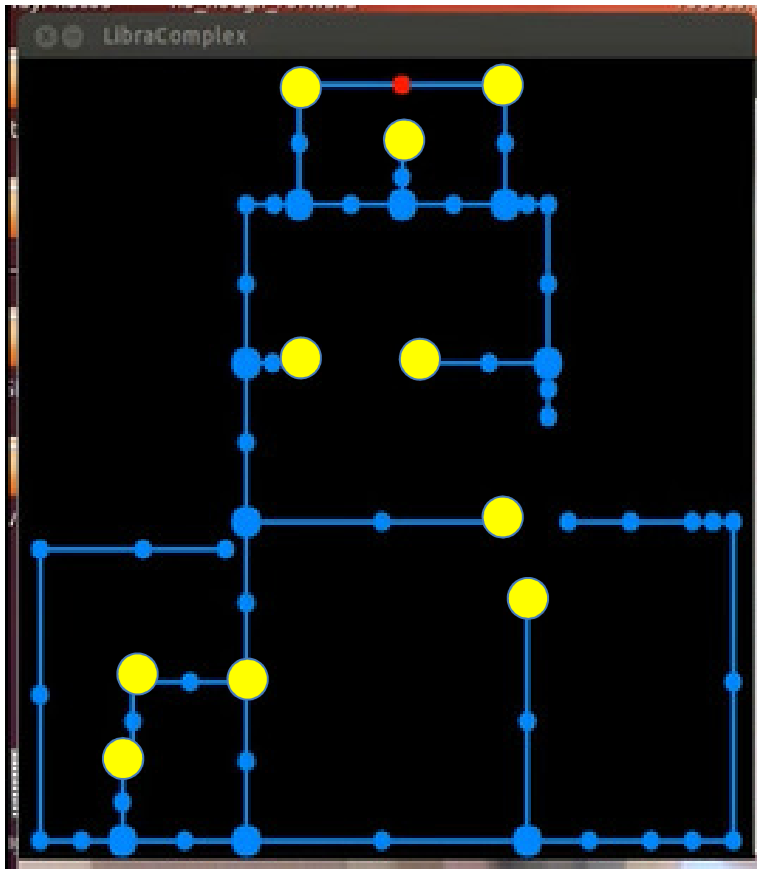


We're at a TL...

Which way  
***should*** we go?

Can we be more *deliberate*?

Suppose the robot goes *left*...



**2 LR**

**2 LL**

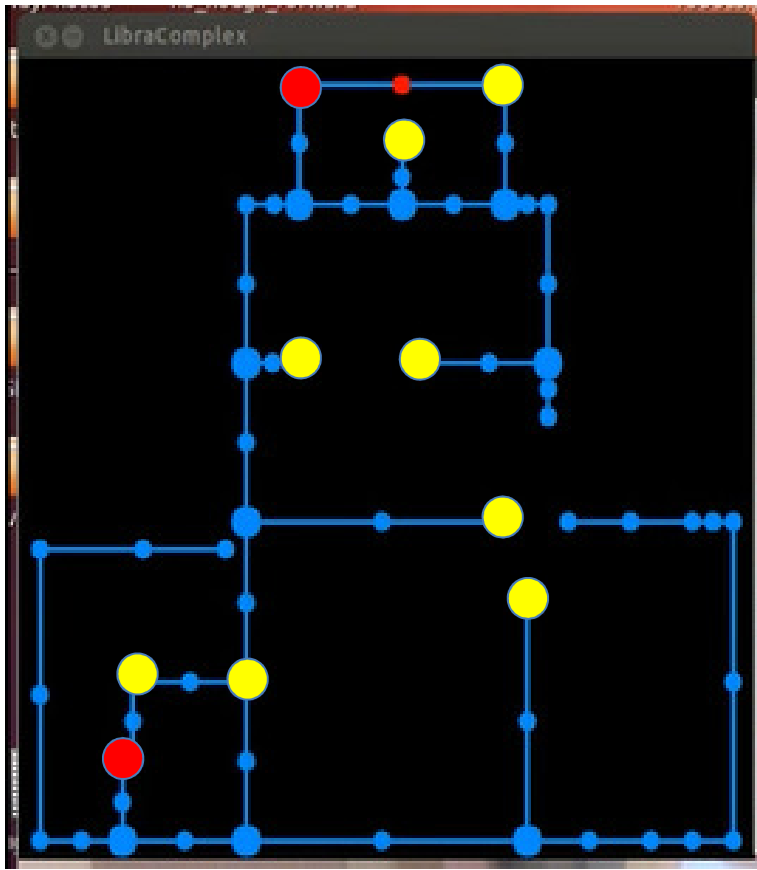
**5 DE**

**1 TL**

This is the distribution of possible  
*next* corners it could see.

Can we be more *deliberate*?

Suppose the robot goes *left*...

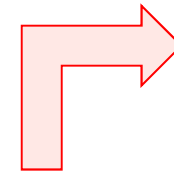


**2 LR**

**2 LL**

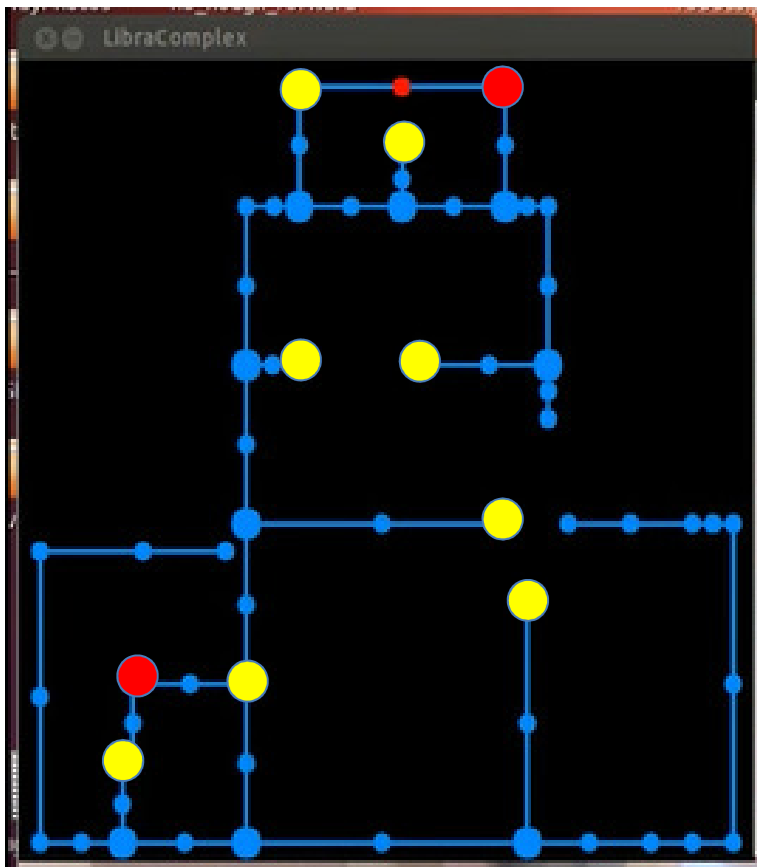
**5 DE**

**1 TL**



Can we be more *deliberate*?

Suppose the robot goes *left*...

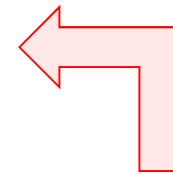


2 LR

2 LL

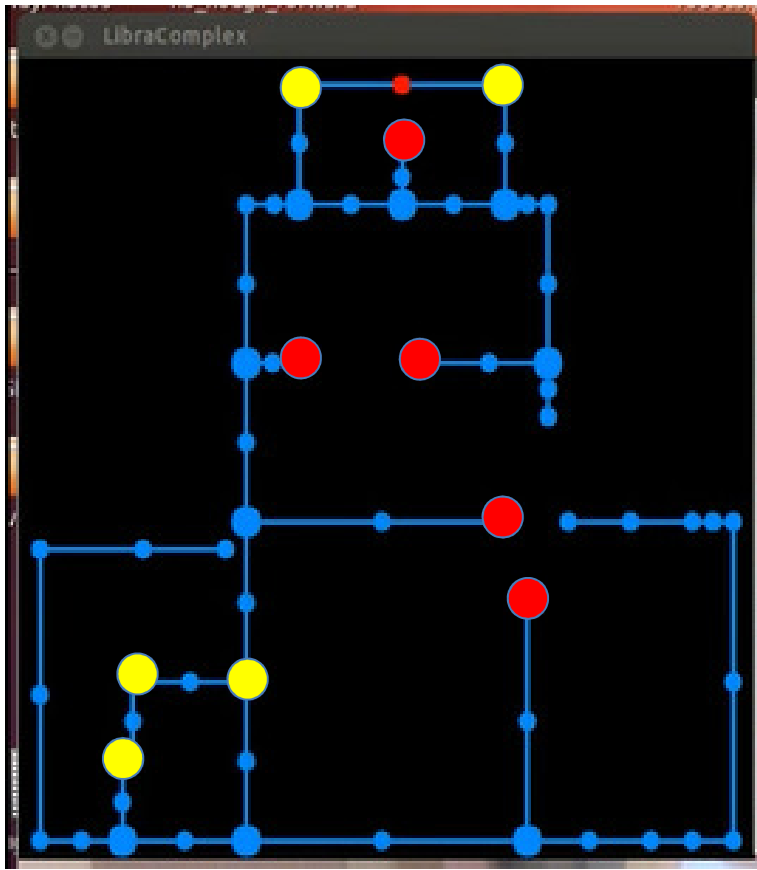
5 DE

1 TL



Can we be more *deliberate*?

Suppose the robot goes *left*...



**2 LR**

**2 LL**

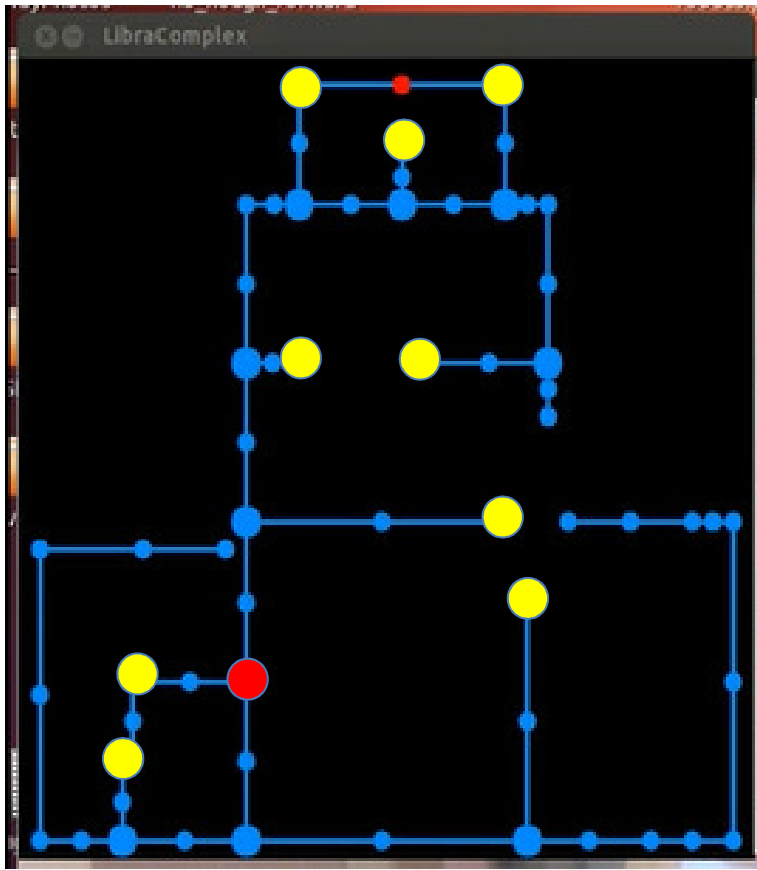
**5 DE**

**1 TL**



Can we be more *deliberate*?

Suppose the robot goes *left*...

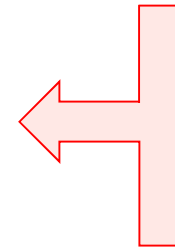


2 LR

2 LL

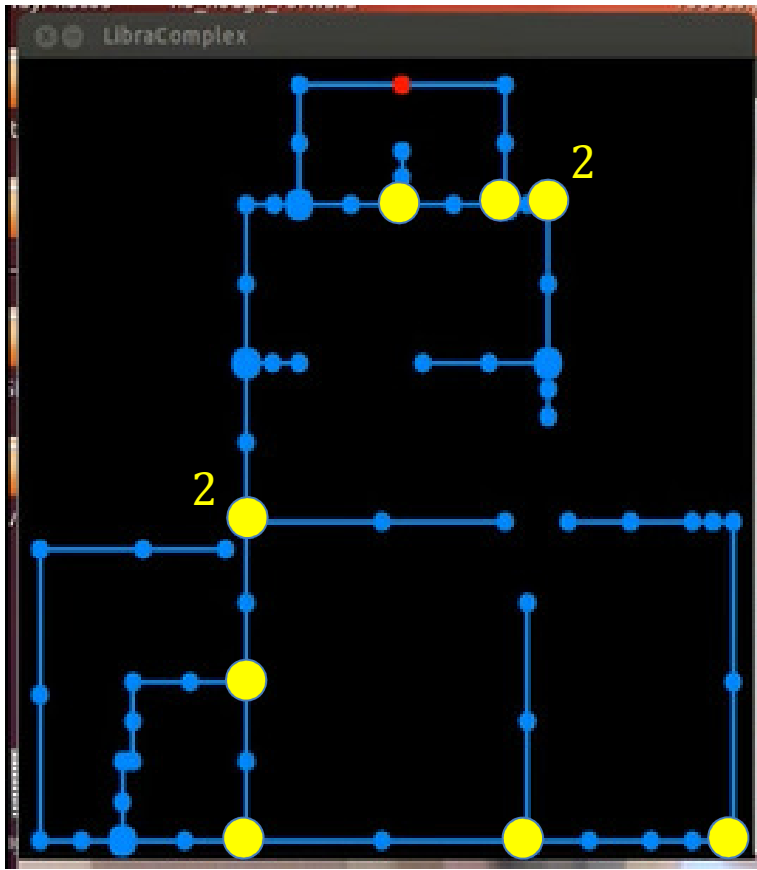
5 DE

1 TL



Can we be more *deliberate*?

Suppose the robot goes *straight*...

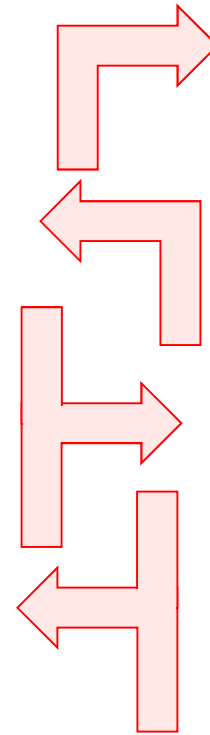


**1 LR**

**2 LL**

**1 TR**

**6 TL**



the possibilities

Which possibilities are *better*?

*left*

**2 LR**

**2 LL**

**5 DE**

**1 TL**

*straight*

**1 LR**

**2 LL**

**1 TR**

**6 TL**

Which of these two possible actions (and their results) should we prefer?

# Entropy

Suppose there were 8 possibilities to begin with...

<b>2</b>	<b>A</b>
<b>2</b>	<b>B</b>
<b>2</b>	<b>C</b>
<b>2</b>	<b>D</b>

2 possibilities  
will remain!

<b>4</b>	<b>A</b>
<b>4</b>	<b>B</b>
<b>0</b>	<b>C</b>
<b>0</b>	<b>D</b>

4 possibilities  
will remain!

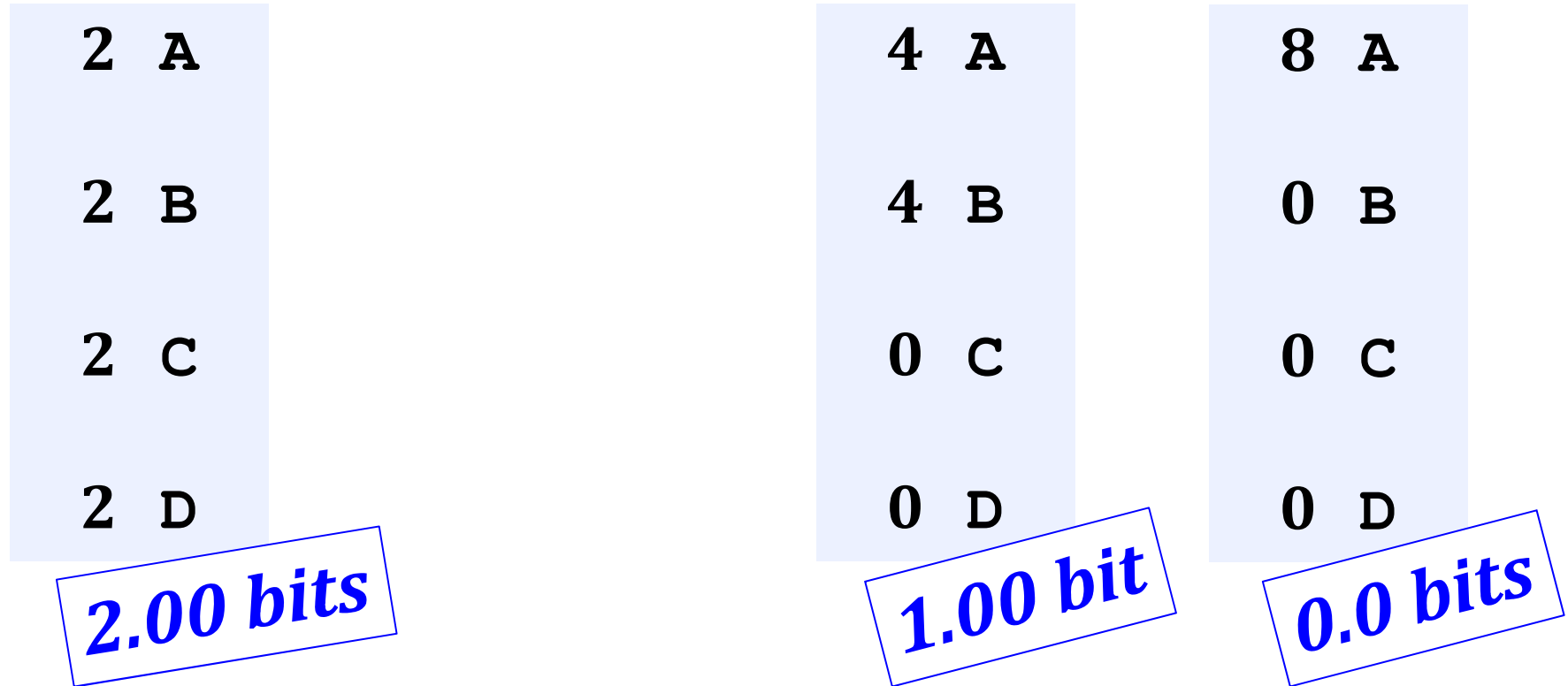
<b>8</b>	<b>A</b>
<b>0</b>	<b>B</b>
<b>0</b>	<b>C</b>
<b>0</b>	<b>D</b>

8 possibilities  
will remain!

← Flatter is better →

# Entropy

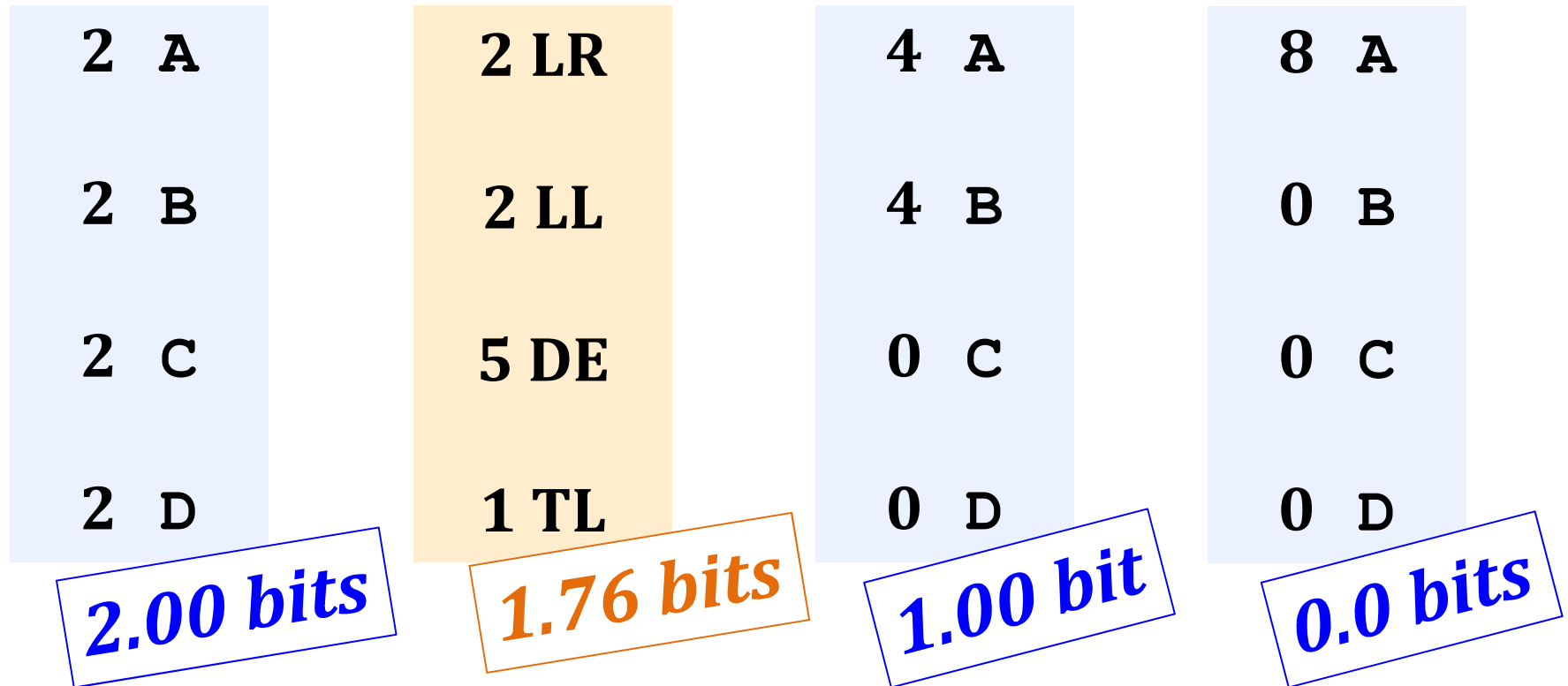
Suppose there were 8 possibilities to begin with...



Entropy ~ *expected information*.

Entropy is the amount of uncertainty you expect to resolve by sampling a distribution.

# Entropy



← Flatter is better →

Go left!

*left*

2 LR

2 LL

5 DE

1 TL

*1.76 bits*

*straight*

1 LR

2 LL

1 TR

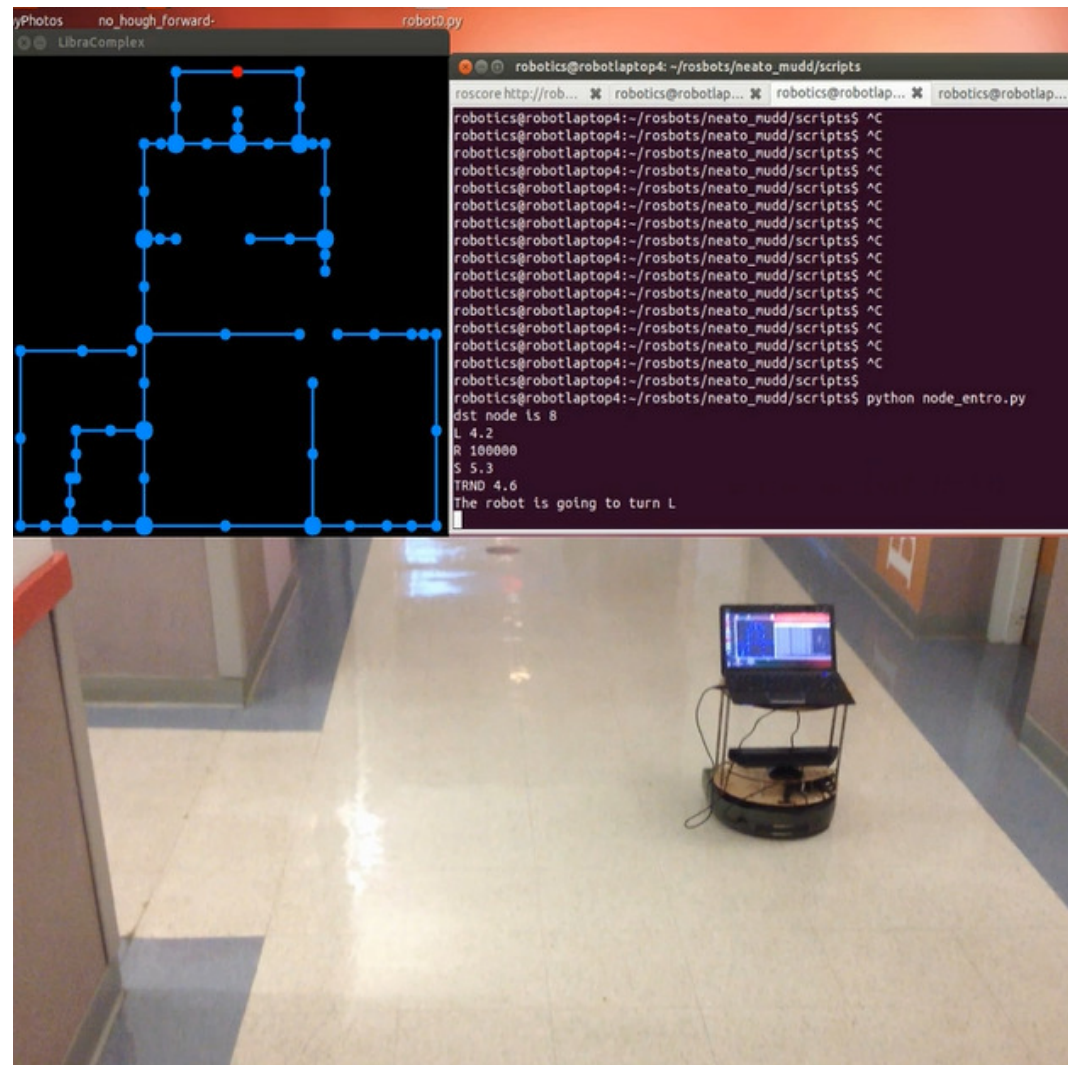
6 TL

*1.57 bits*



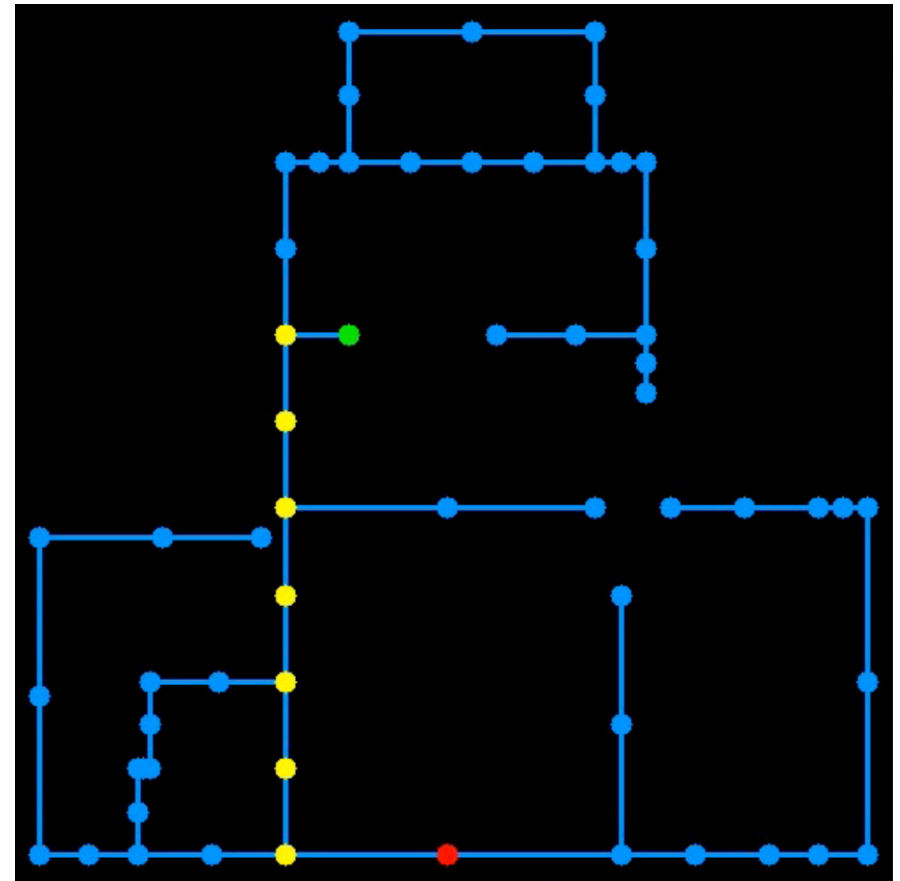
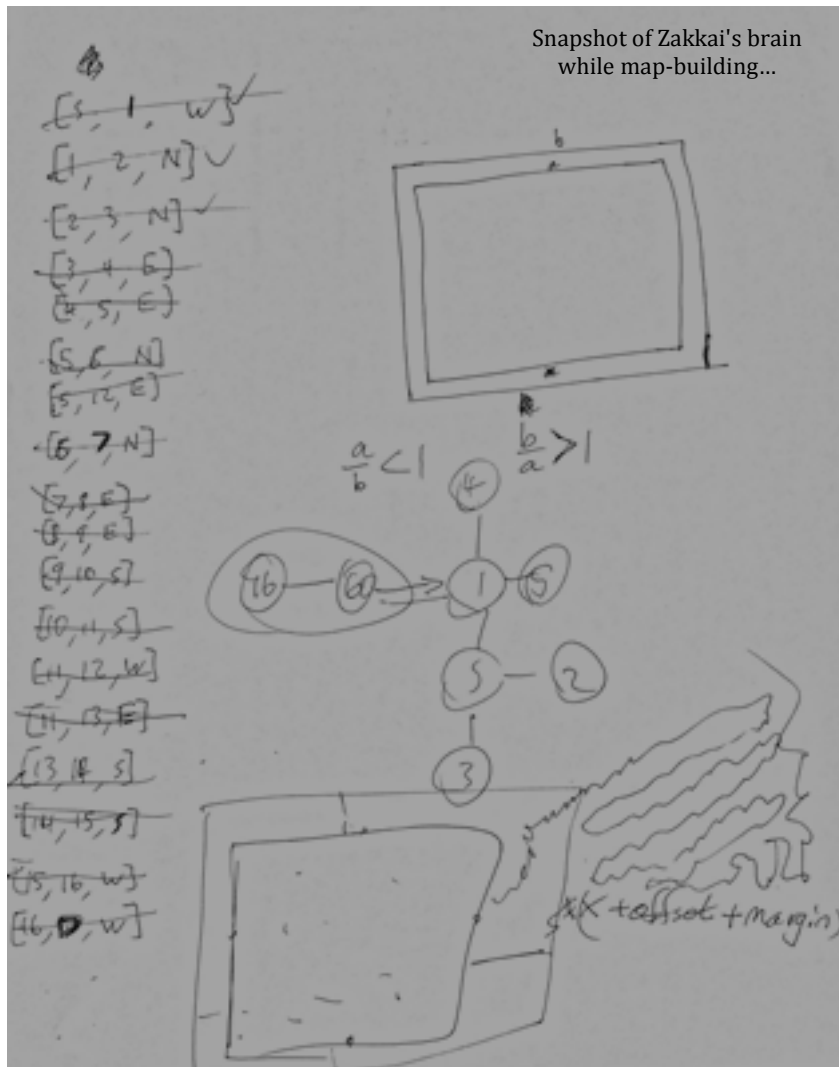
Flatter is better

# Entropy-based localization



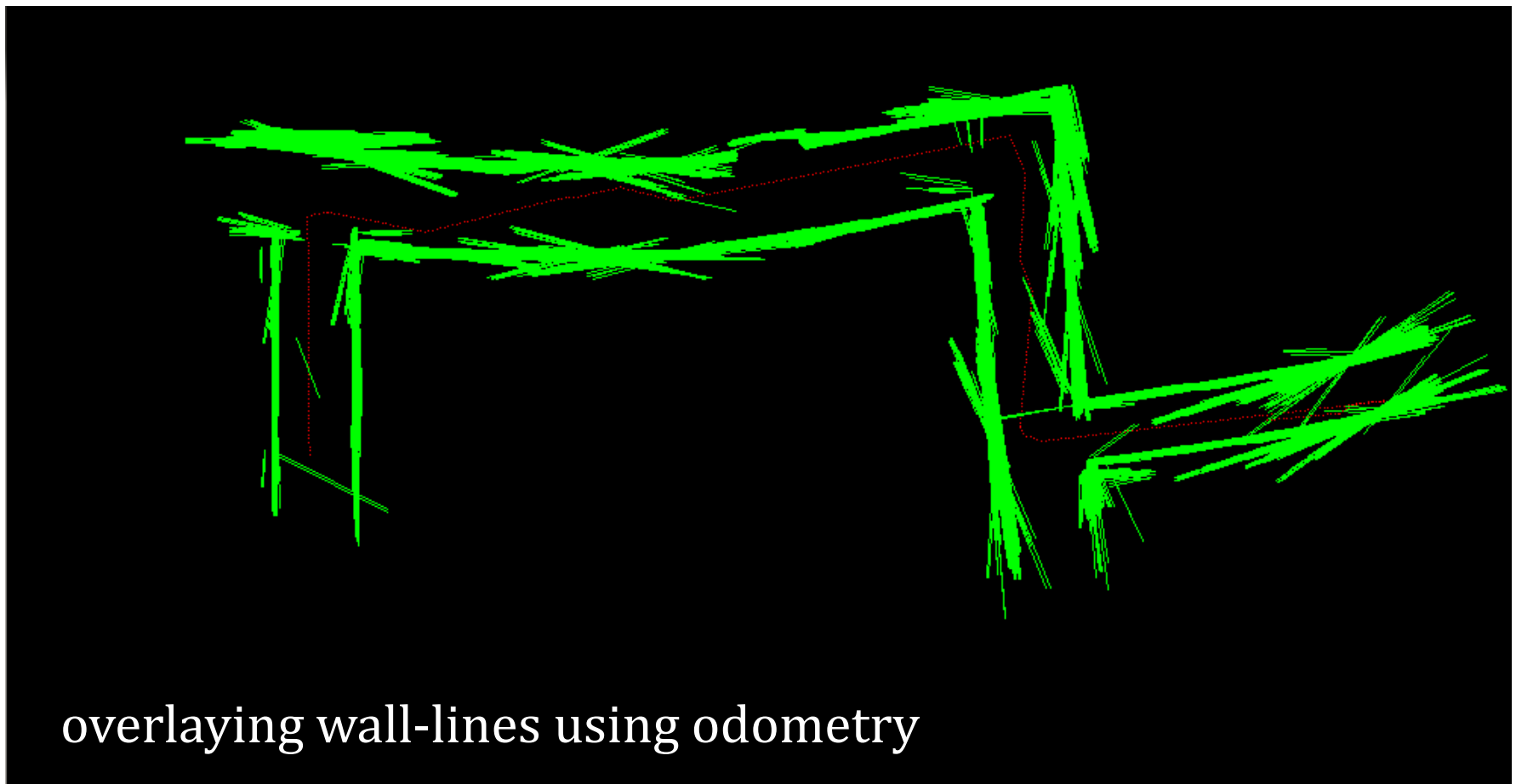
entro\_short.mp4

# Map-building



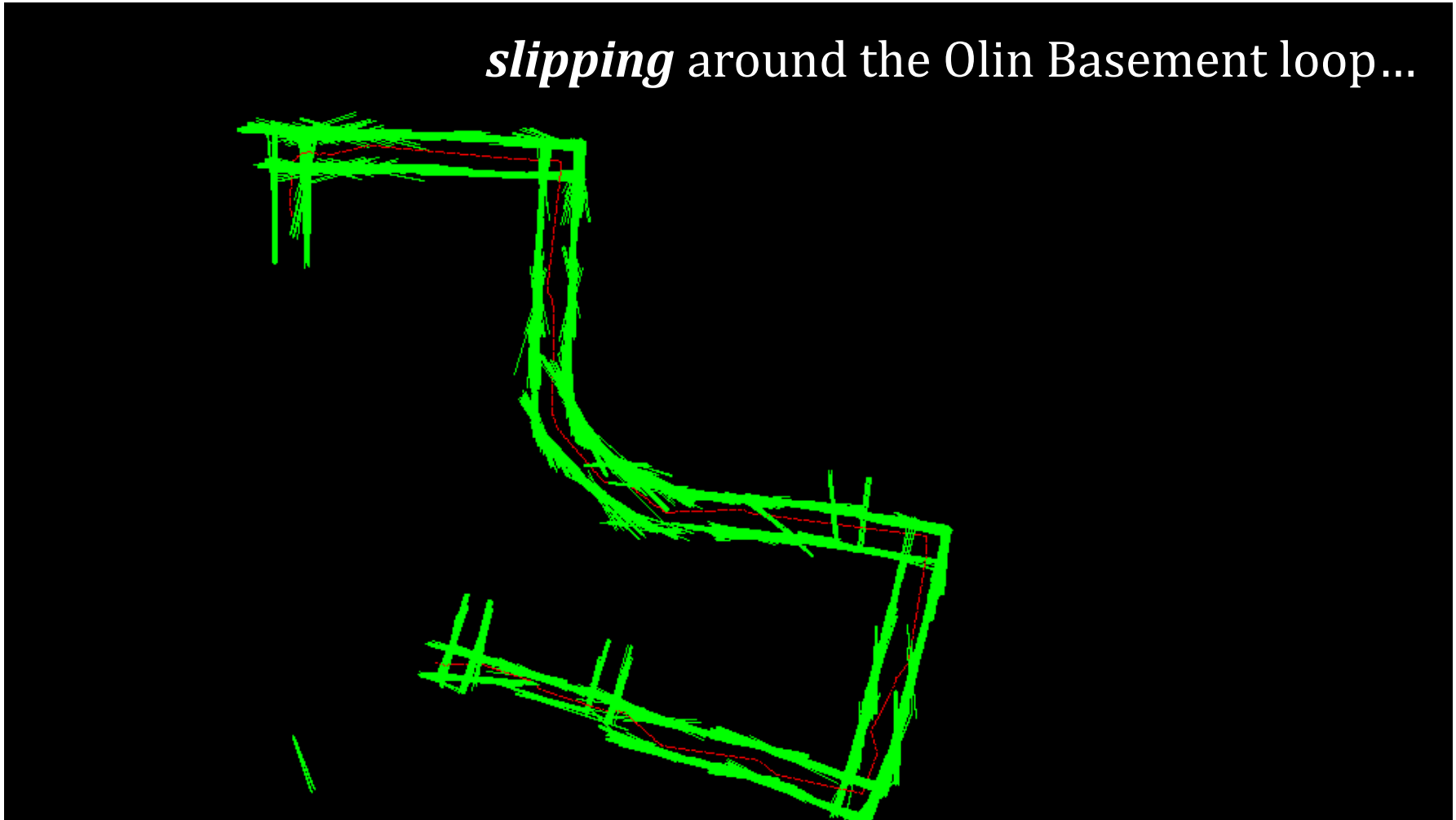
Ideally, mapping would be ***automated***, as well...

# Autonomous Mapping



# Mapping

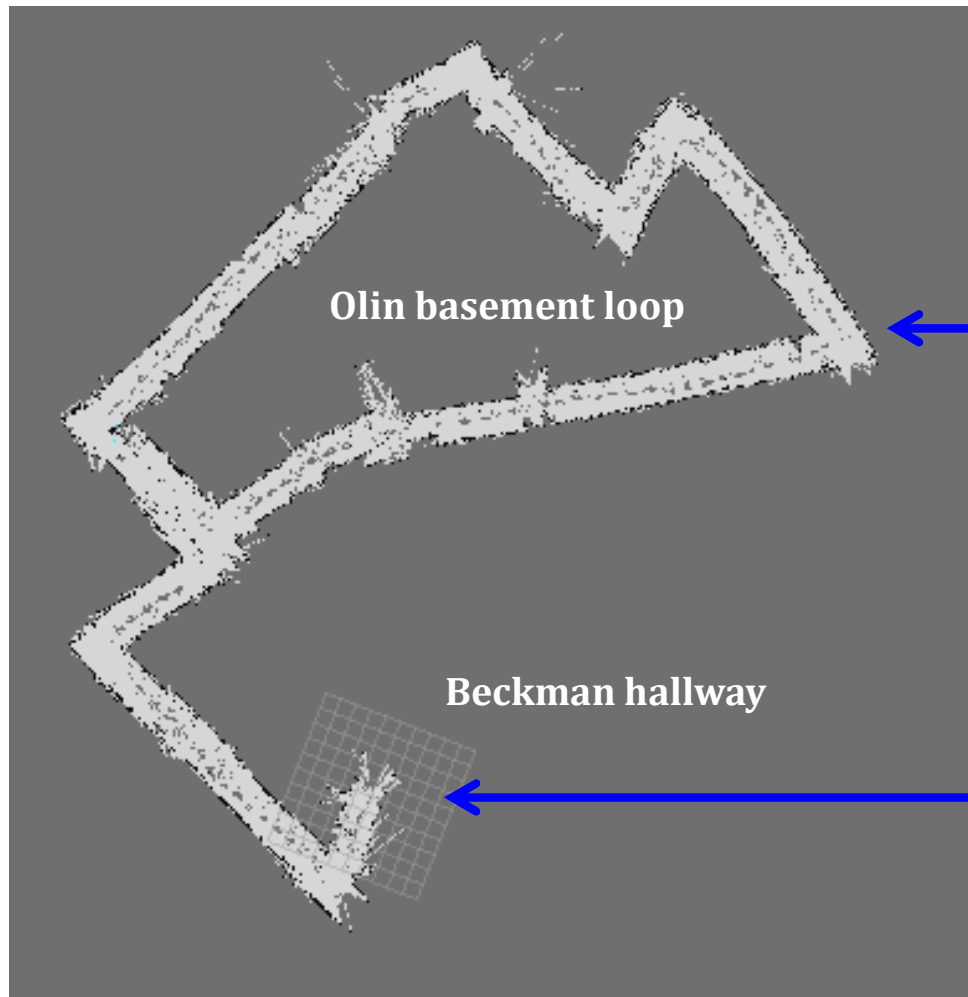
*slipping* around the Olin Basement loop...



Graham + Jerry

# gmapping

an algorithm for autonomous laser-based mapping



(1) adjust each scan's pose to best-fit previous scans

*straightens corridors*

(2) search for distant-past similarities to "close loops"

*corrects slippage?*

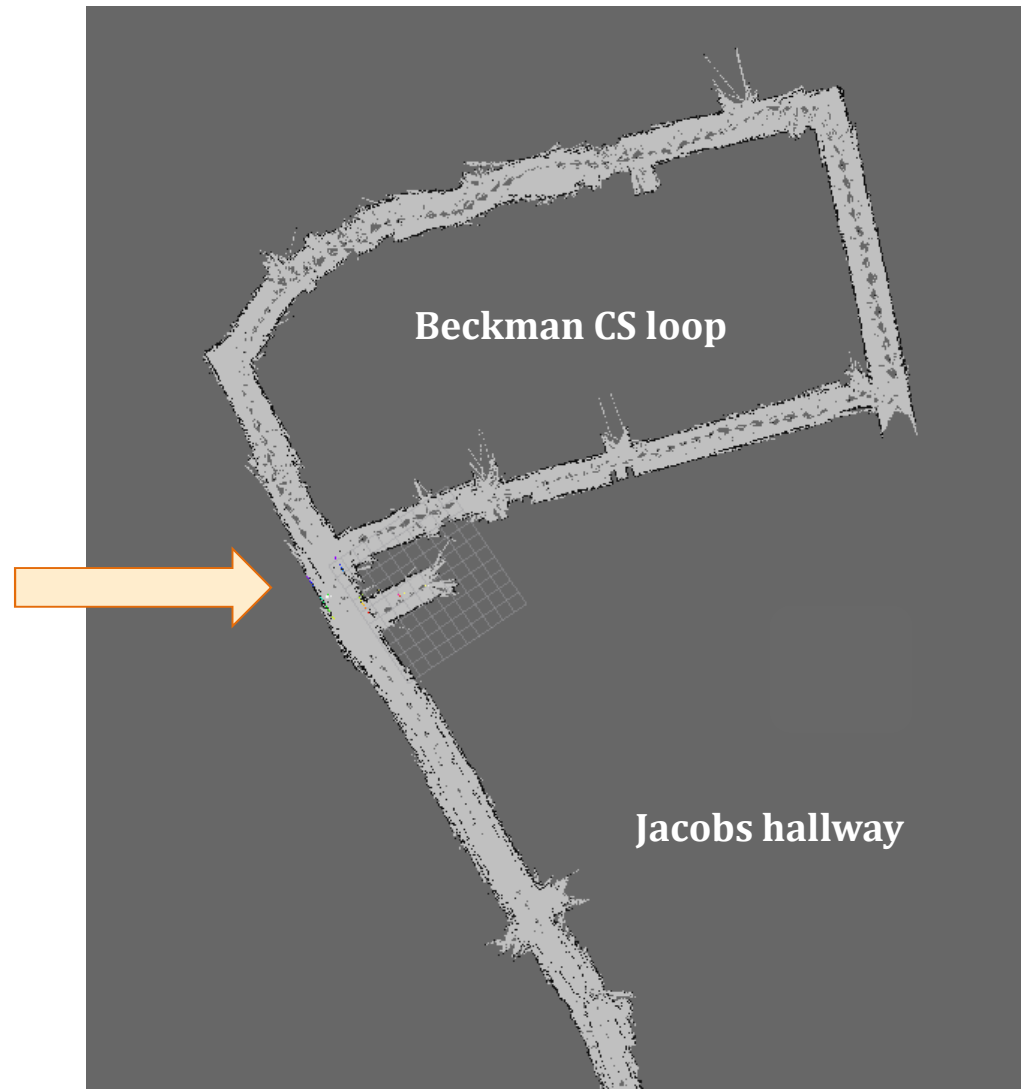
# gmapping

... can *miss* "obvious" loops

Laser scans aren't  
very *distinctive*

*Better landmarks?*

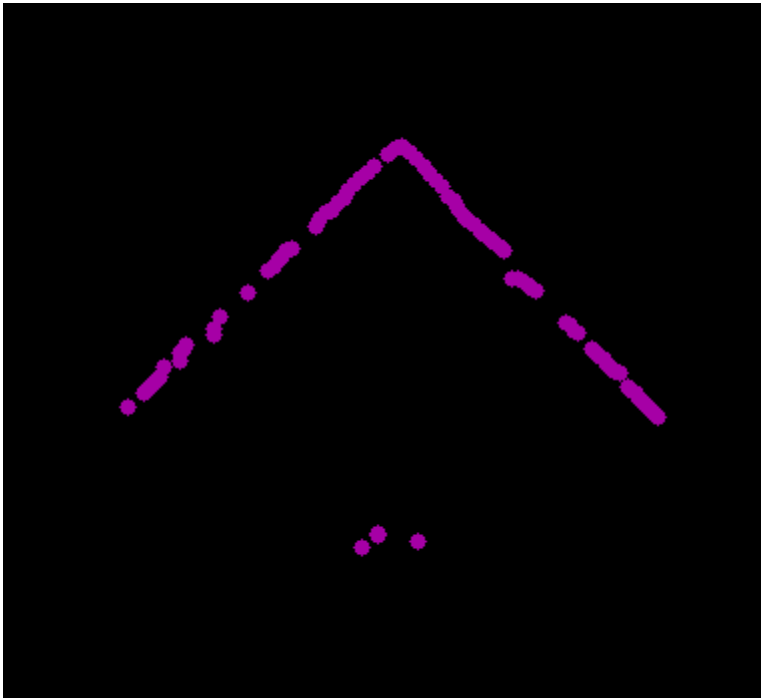
*... probably texture in 3d*



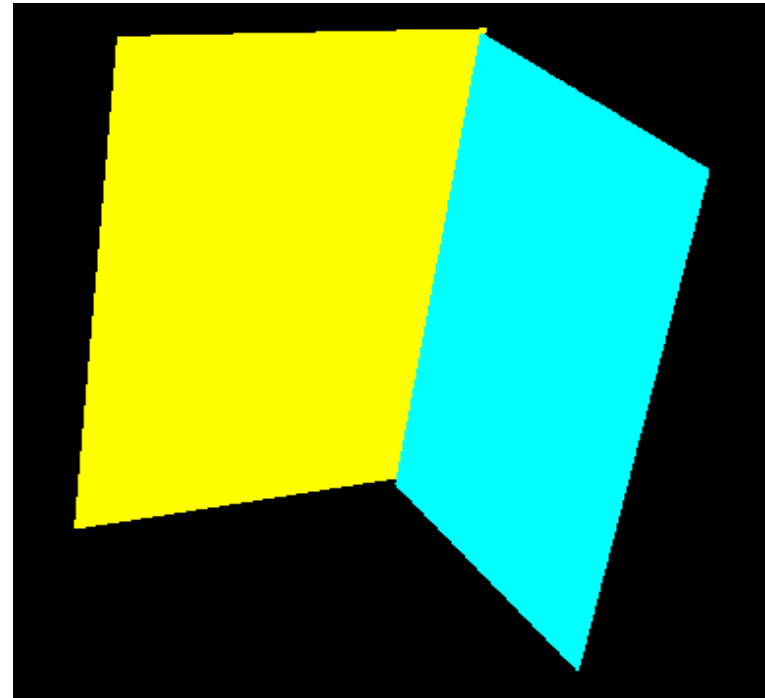
# 2<sup>1</sup>/<sub>2</sub>-d mapping

## Idea:

- Assume **wall-world**: surfaces are vertical planes.
- Create that "3d" representation
- Texture it with the pixel colors ~ *landmarks*.

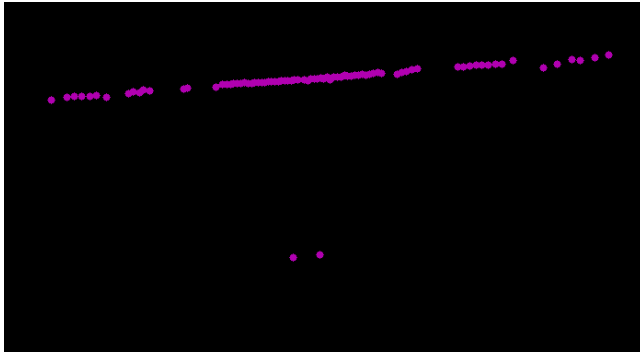


Laser-scan of a corner



Now, as vertical planes

# *Painting the walls...*



Laser-scan of a wall

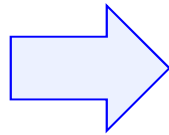


Image of the wall



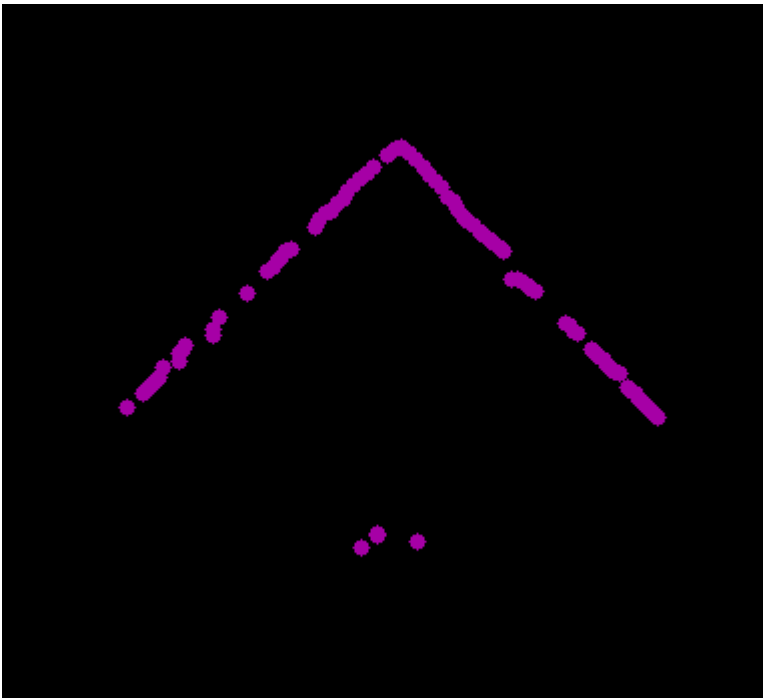
textured model

# *Painting the walls...*



We can apply any geometry we like ~ here, a cylindrical panorama

# Where are the walls?

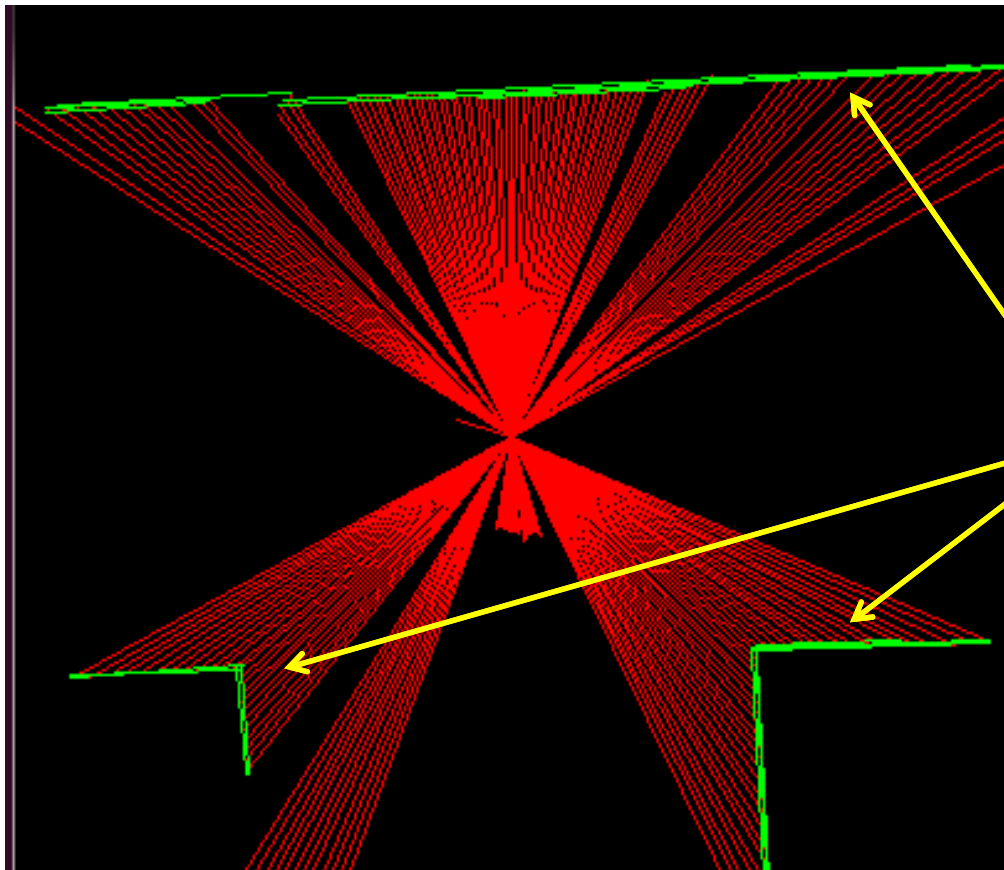


linked to raw\_corner\_cropped.m4v

Raw laser scans lead to artifacts ~ walls vs. ribbons

# Finding the walls

via the Hough transform



~ **24 lines**  
extracted

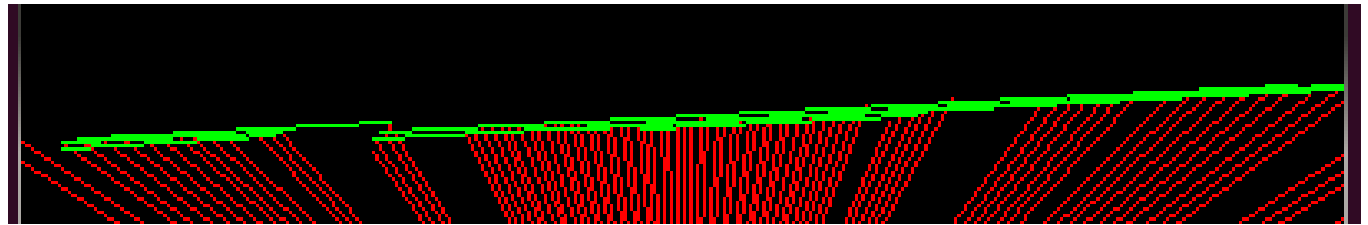
Extracting straight lines from laser scans yields *too many* of them

# Selecting the best walls...

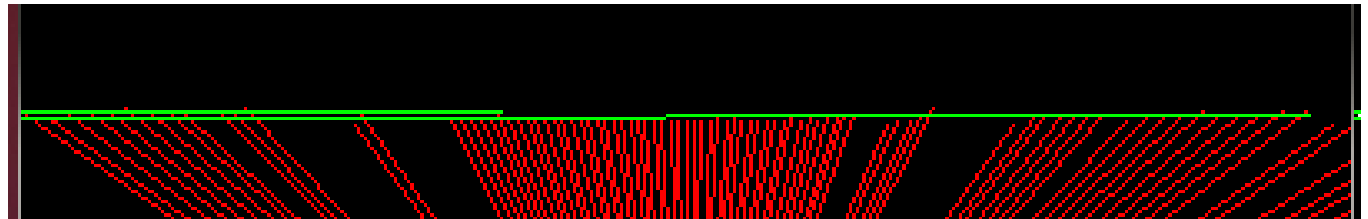
Use the Hough transform to find an initial set of wall segments  
**do**

- Compute the **slope and position** of each segment
  - Find a "close" pair of segments; **remove them from set**
  - Merge the pair into a single segment and **add to set**
- until** no more "close" pairs exist.

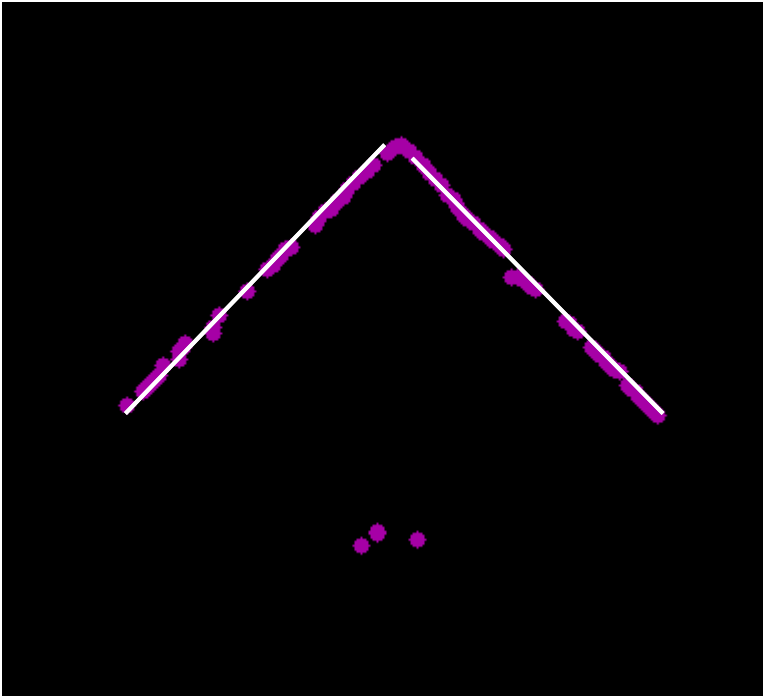
~12 Hough lines



with all but 2  
merged



# Better!



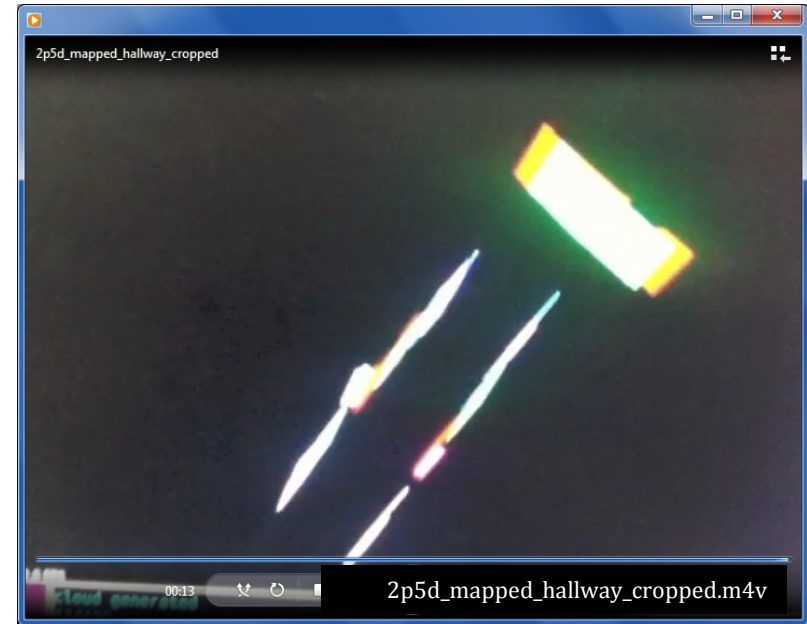
linked to [hough\\_corner\\_3d\\_cropped.m4v](#)

First, simplify the 3d structures, *then* paint them ...

# Building the map...



the robot's final position...



... after creating its 2½-d map

*Delegated to 2014:* painting these walls in real time...

Chris + Kristina

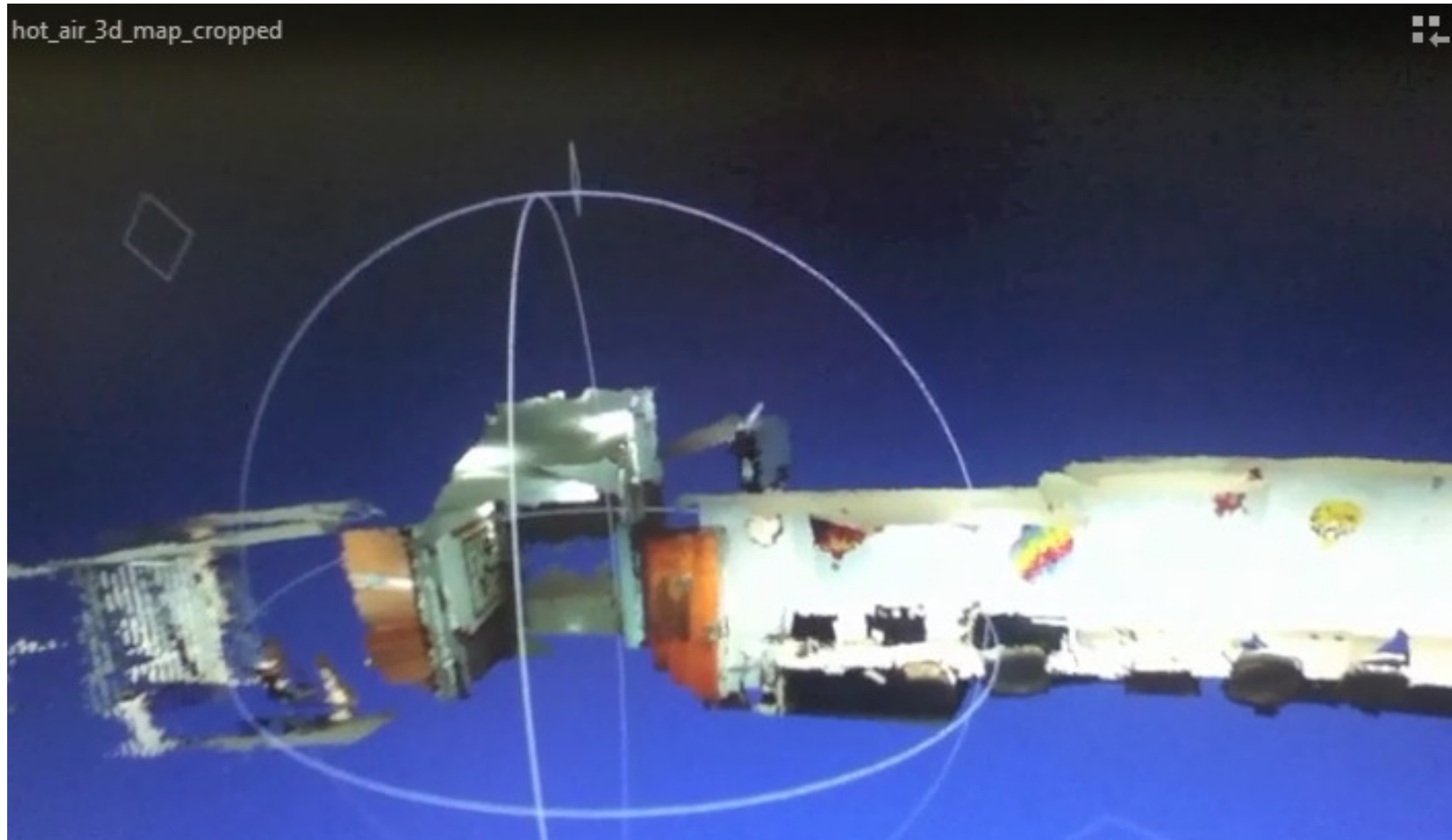
Or, use the Kinect: *3d mapping*



Demonstration of RGBD-SLAM

SLAM ~ simultaneous localization and mapping

# The hot-air lab's map



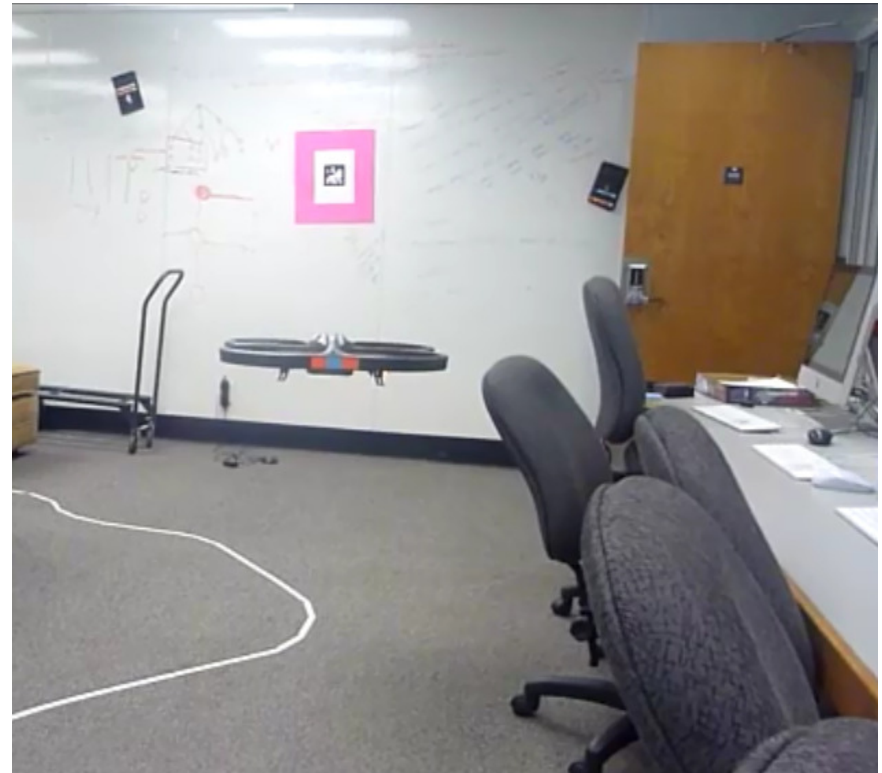
Example of lab and hallway map created by RGBD-SLAM

# Two concrete challenges...

more *robust* navigation



more *general* "lab escape"



bridging 2d sensors with 3d spatial representations

# *Escape the lab!*

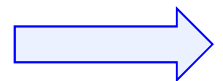


## Localization-based loop:

(given a known goal)

(1) the drone grabs a 2d image

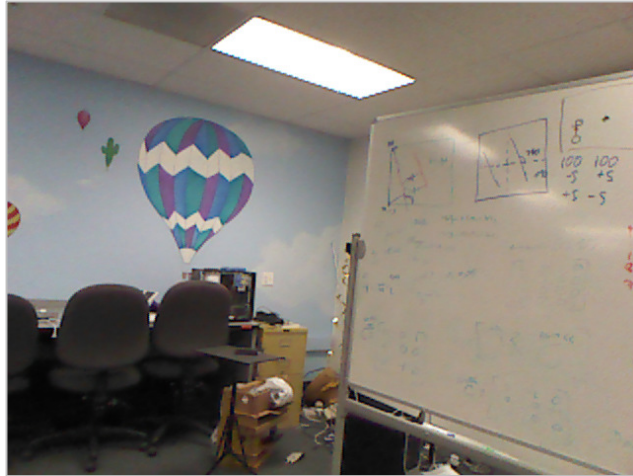
(2) the system *matches* that image into the 3d map to locate the drone



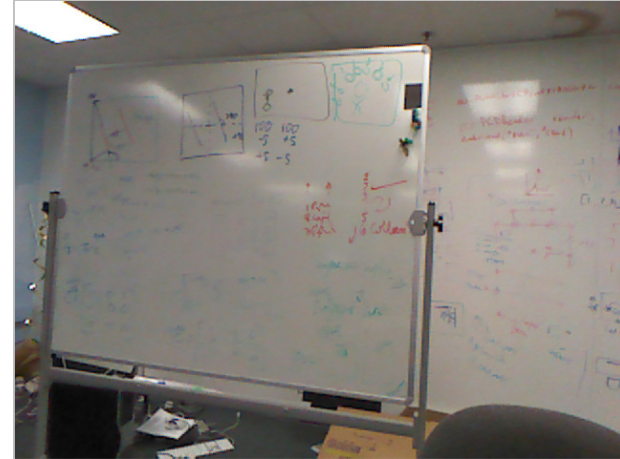
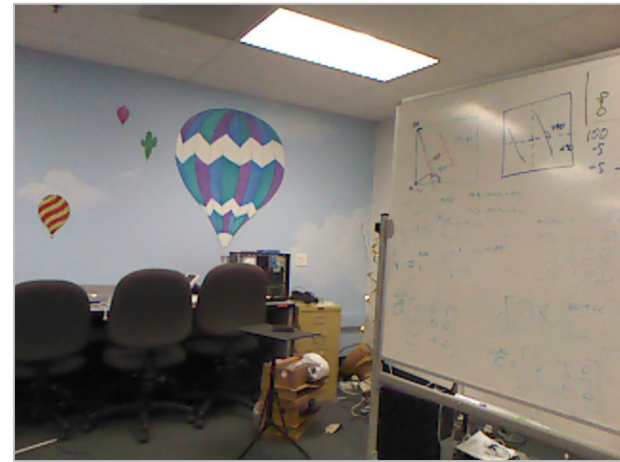
(3) once located, the drone computes its desired velocity and *applies* it

(4) after two seconds, stop and goto (1)

# Image matching



new image (live)

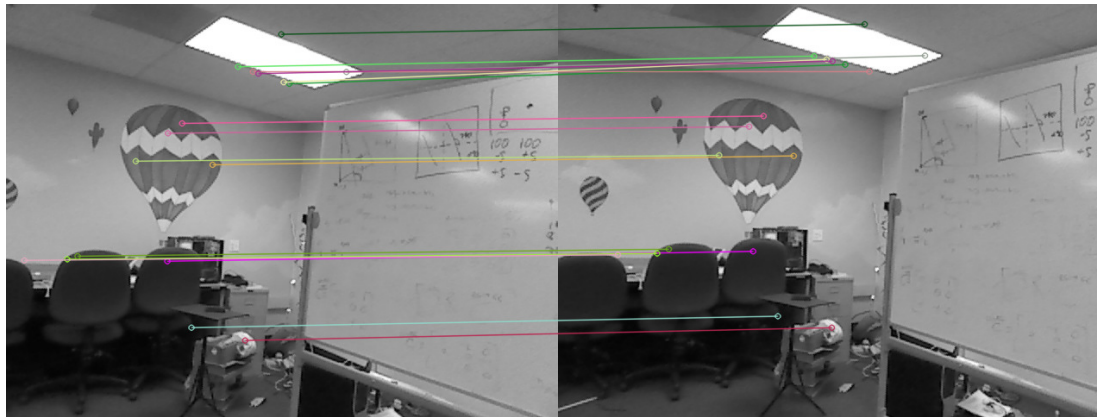


many original images (stored)

## Our approach (well, really Kristina's):

- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

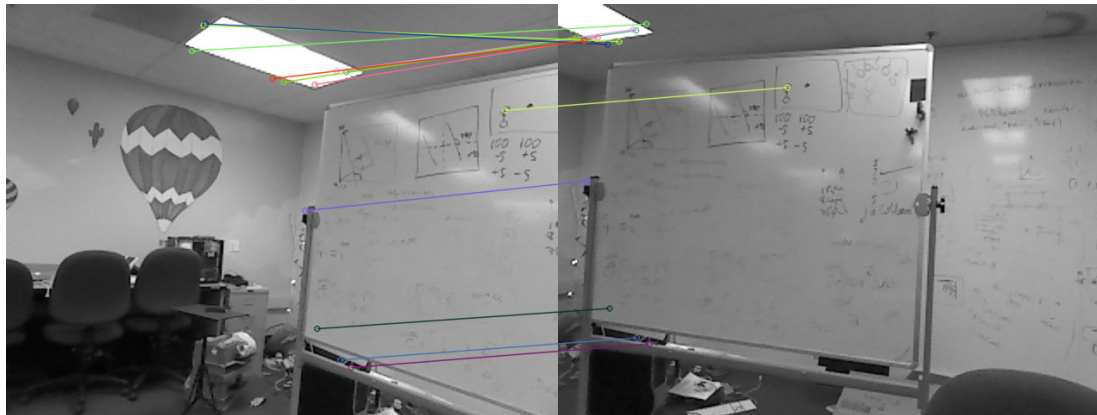
# Image matching



new image (live)

many original images (stored)

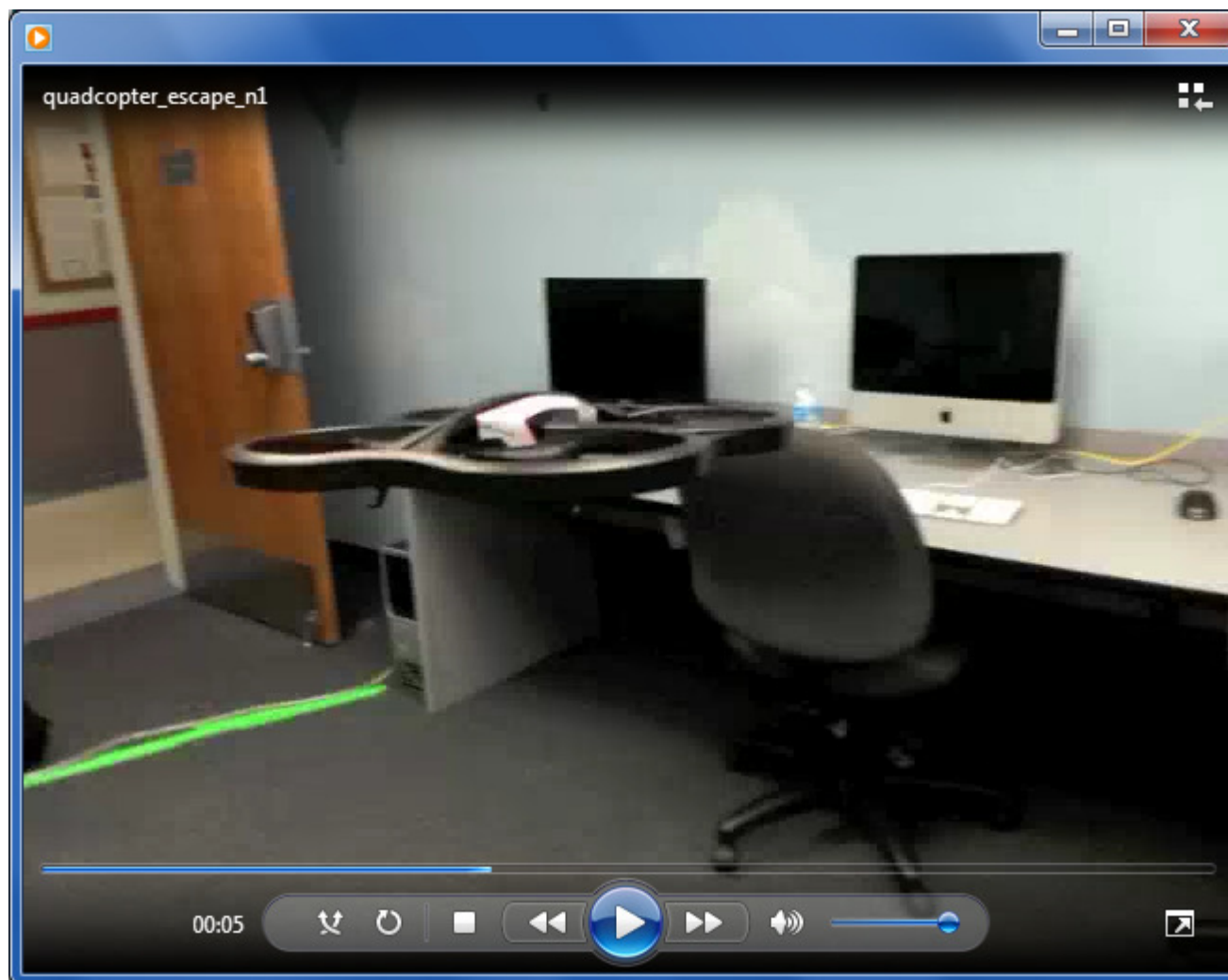
1<sup>st</sup>-best match



10<sup>th</sup>-best match

for now, a linear search taking **~10 seconds** for the hot-air map's 42 images

# Results



One of the drone's escapes ~ 150 sec.

# 9 "landmarks"

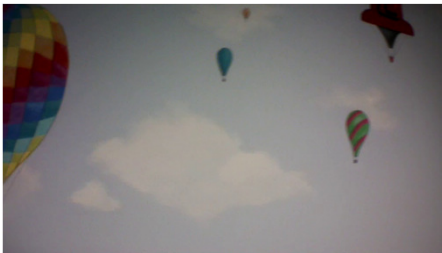
0



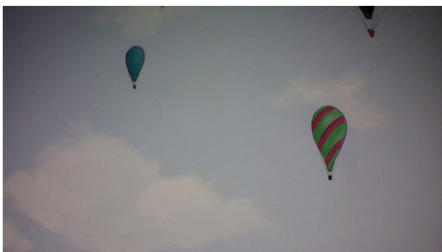
1



2



3

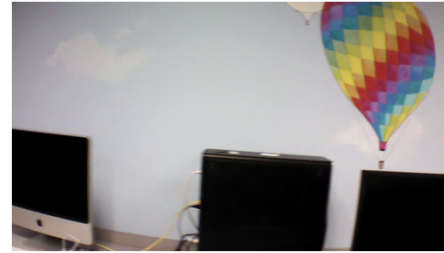


live, novel images

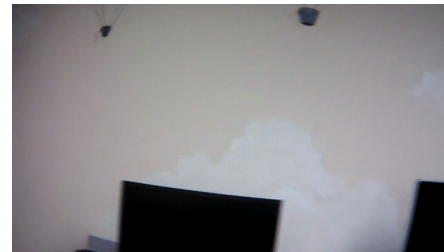
4



5



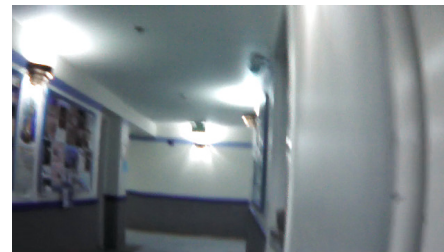
6



7



8



# Their matches

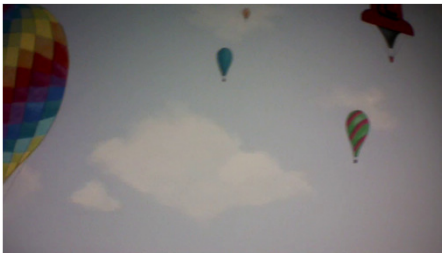
0



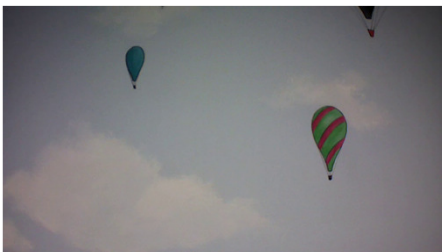
1



2



3



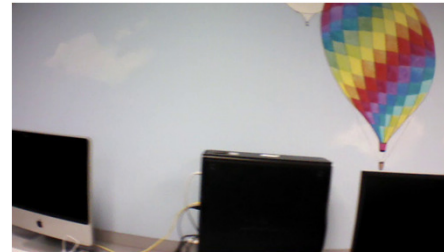
live, novel images

map matches

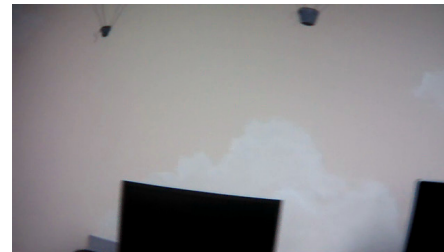
4



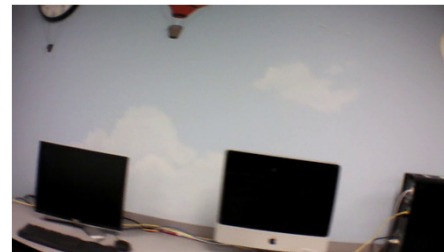
5



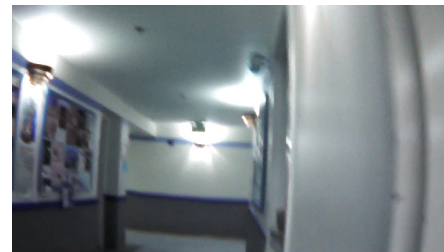
6



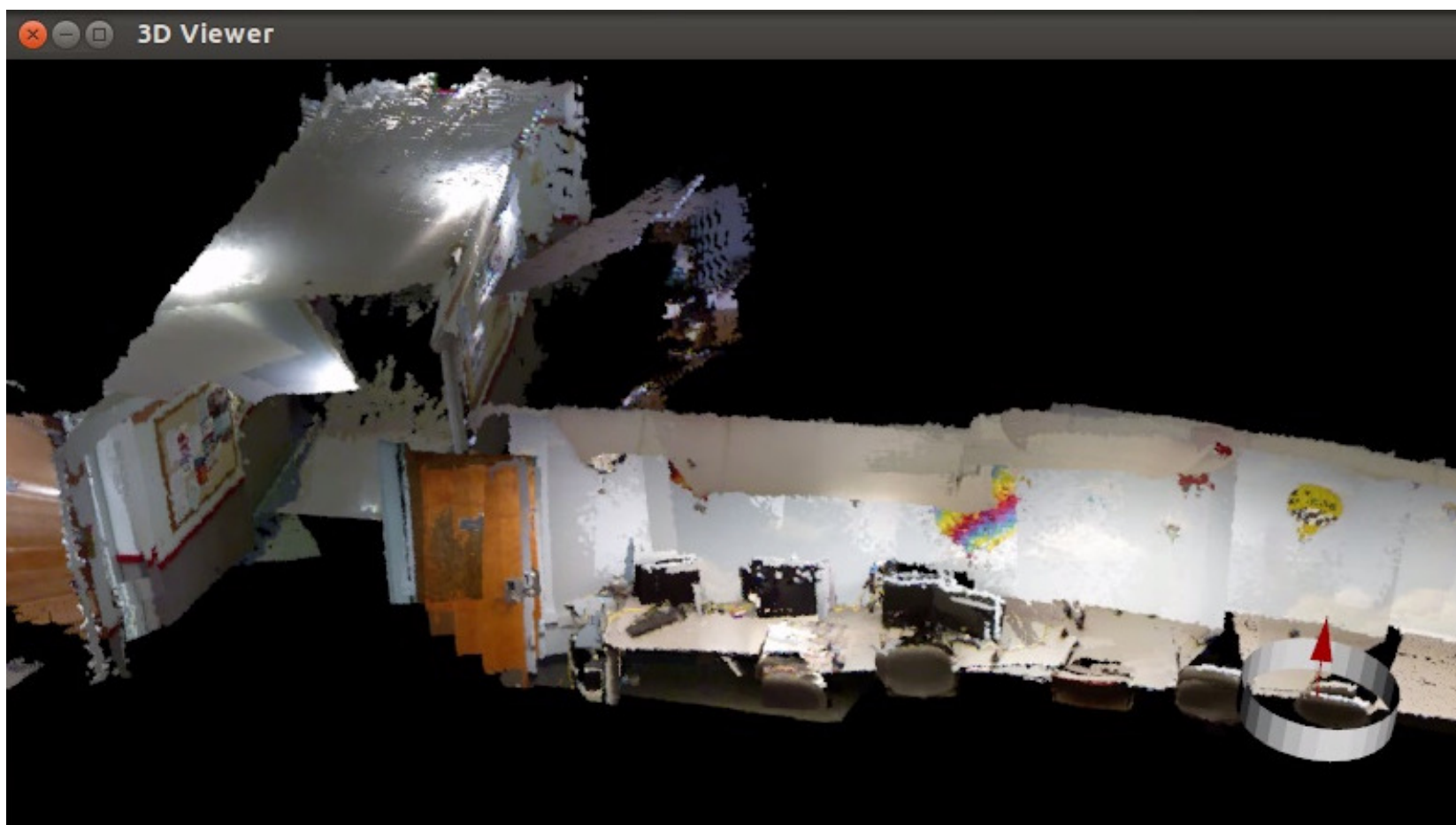
7



8



# Model

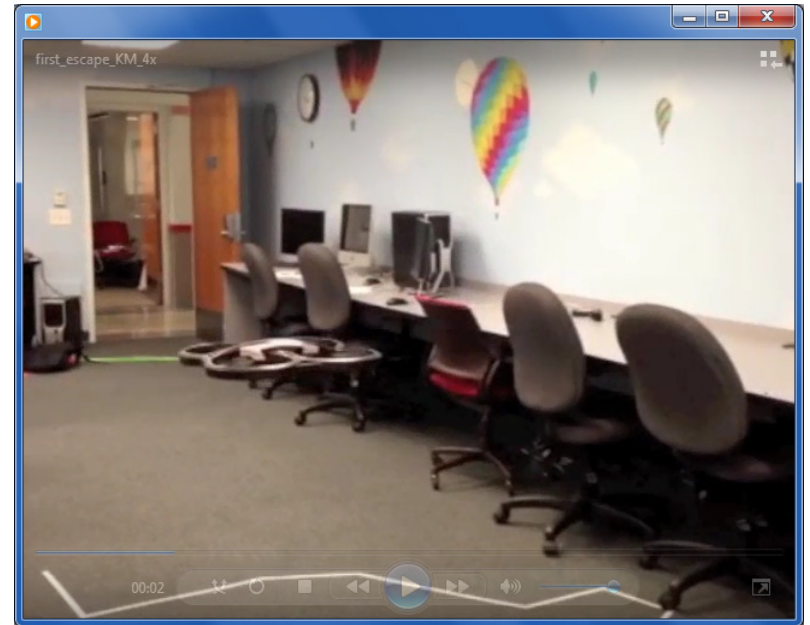


Map-based localization of those 9 estimated positions

# Other runs



Why not localize once  
and just go!?



A view from the back of  
the lab

# Verdicts & insights

**2d matching?**

not perfect  
not fast... *yet*  
more geometry to exploit

surprisingly  
good!

**slowness?**

don't use linear search!  
do use multi-resolution search!!  
use *lack of features* somehow

opportunities!

**accuracy?**

use 3d model to get "*exact*" location

**3d overall?**

**Yes!**

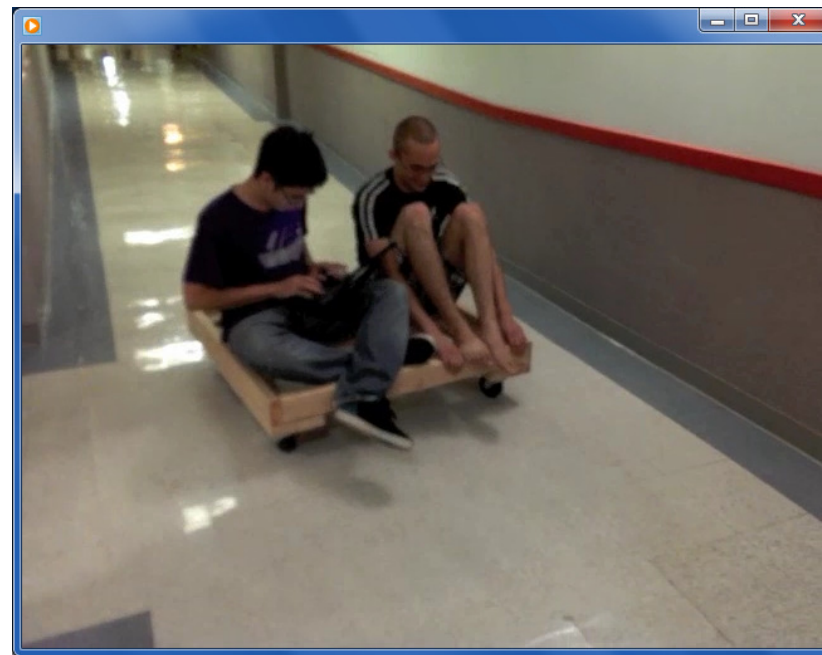
Skeleton of next summer's  
robotics agenda...

What about next  
summer's **robots**?

# New platforms...



Microdrone



"The Sofa"

LeapMotion

Thoughts?  
Questions?

# This afternoon...

[www.cs.hmc.edu/~cs5grad/survey/](http://www.cs.hmc.edu/~cs5grad/survey/)



(1) **[anytime]** Take our NSF research-finale survey!

(2) **[afternoon]** Create a slideshow of best summer images -- it's great to have *all* of the images, but it's equally or more useful to have a slideshow of the highlights

(3) Let's show @ 4:30...

(4) **[starting ~ 3:30]** Save/archive your work

(5) Shutdown the HMC-owned computers

\*\* ***Except in the games cubicle:***

(6) Create lines of monitors, mice, cables, surge protectors, computers, etc. along the back walls

(7) Clean/wipe off the workspace tabletops

(8) For kitchen, remove trash + clean surfaces/dishes

# Contributions...







17	IS					TTL
	6	7	8	9	10	
HARVEY	31	63	57	81	-9	119
	68	77	95	104	113	113
MUDD	41	18	-9	71	-6	117
	23	32	41	49	55	55
IS	-7	6	7	-9		116
	42	59	66	75		75
THE	7	77	87	101		127
	69	87	105	114		114
BEST	16	-1	15	67		120
	40	41	47			57
0.3						

18	IS				TTL
	6	7	8	9	
HH	42	87	7	-2	
	55	72	79	8	
BYE	8	8	6	3	
	42	50	56	6	
RAPTOR	53	51	3	8	
	61	67	85		
CAT					
INDIA					
0.0					

Contact the Control Desk to extend ...



## GO! FIGHT! WIN!

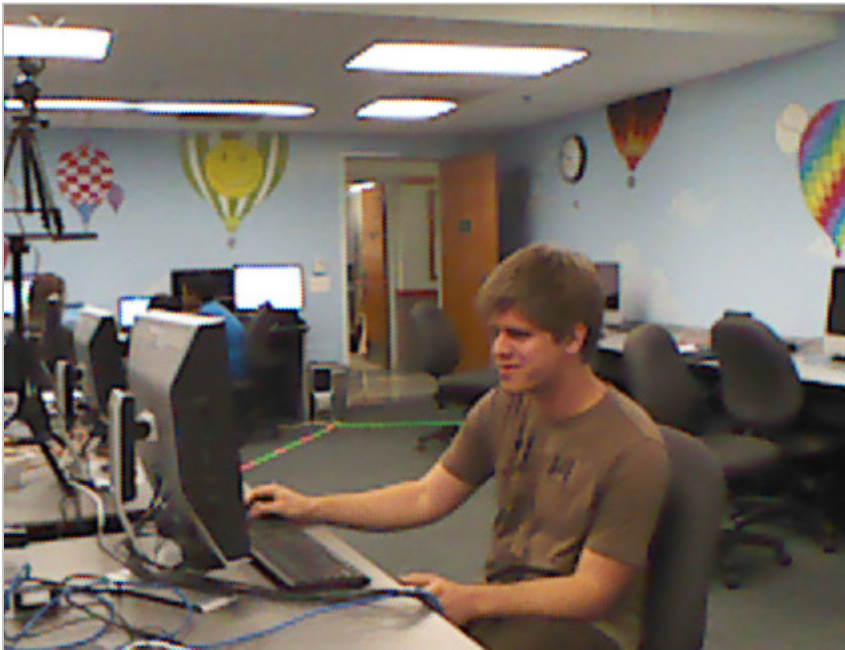
See which schools are rallying the most pep!

TOTAL VOTES!  
**329,427**

01	Harvey Mudd College	100277	11	Texas Christian University	9697
02	Mount Holyoke College	17621	12	Elon University	9099
03	Claremont McKenna College	16692	13	Colorado School of Mines	8675
04	Wartburg Theological Seminary	15763	14	Harvard University	7800
05	Our Lady of the Lake College	14809	15	Northwestern College	7277
06	Massachusetts Institute of Technology	13216	16	Oglethorpe University	6681
07	Erskine College	12955	17	Liberty University	6117
08	Nazarene Bible College	12129	18	Oberlin College	5606
09	Immaculata University	11102	19	Gonzaga University	4975
10	Nyack College	10322	20	Yeshiva College of the Nations Capital	4498



# *Accessing* these 3d maps



saves each rgb image



and each depth image

*Getting the data:* by splicing into the code to extract the input images

# Neato *reasoning*

## Initial tasks

- learn ASCII API
- write device drivers
- interpret sensor data
- follow corridors
- detect intersections

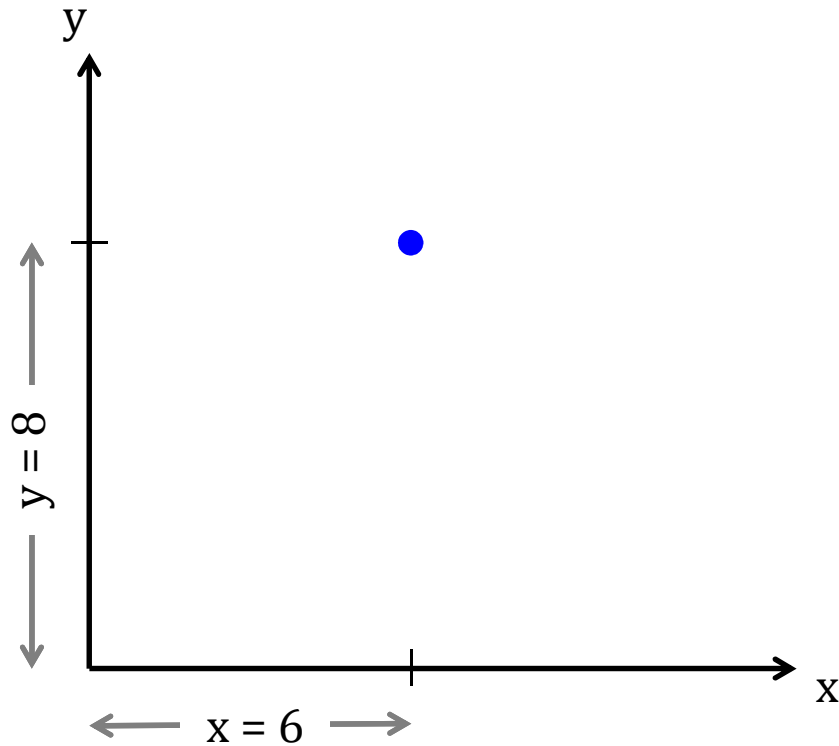
## AI Tasks

- pt-to-pt navigating
- localize and navigate
- minimize ambiguity
- navigate predictively
- build 2d and 3d maps

wall placement determines each intersection *type*

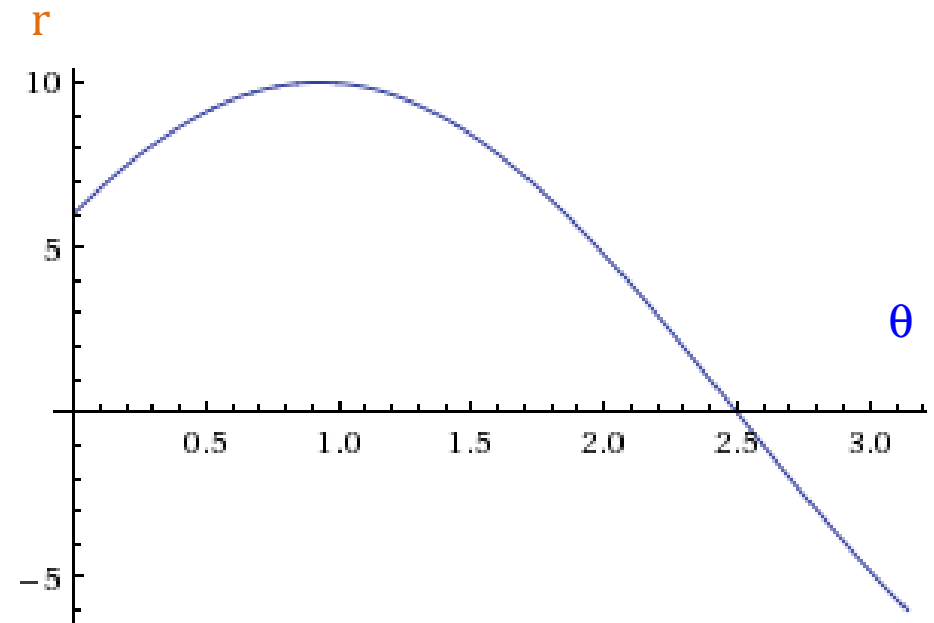
# Hough transform

Point space



Consider *one point* of a laser scan

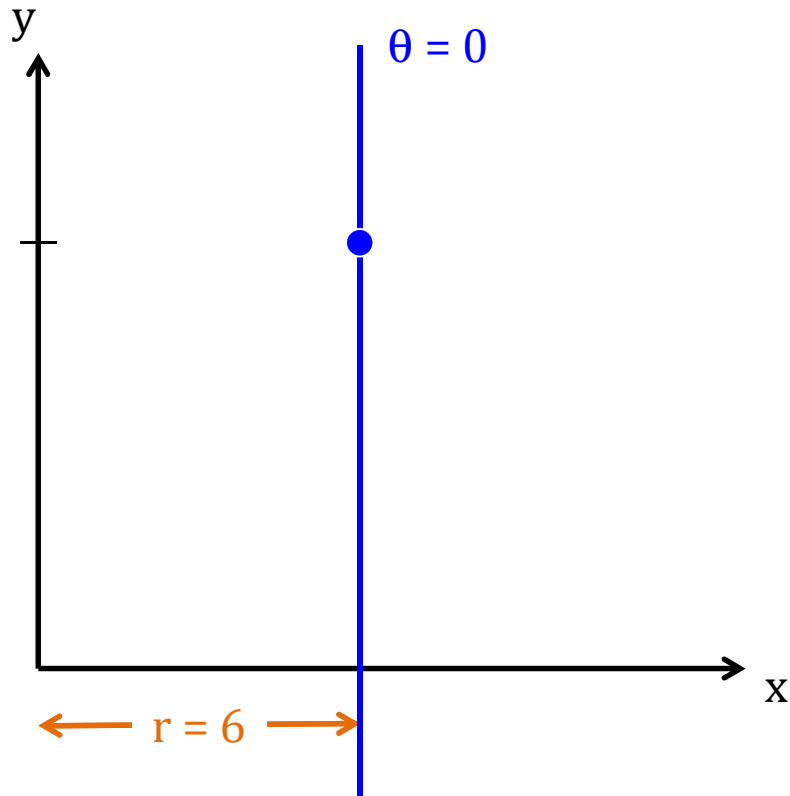
Line space



These are *all* of the lines through it

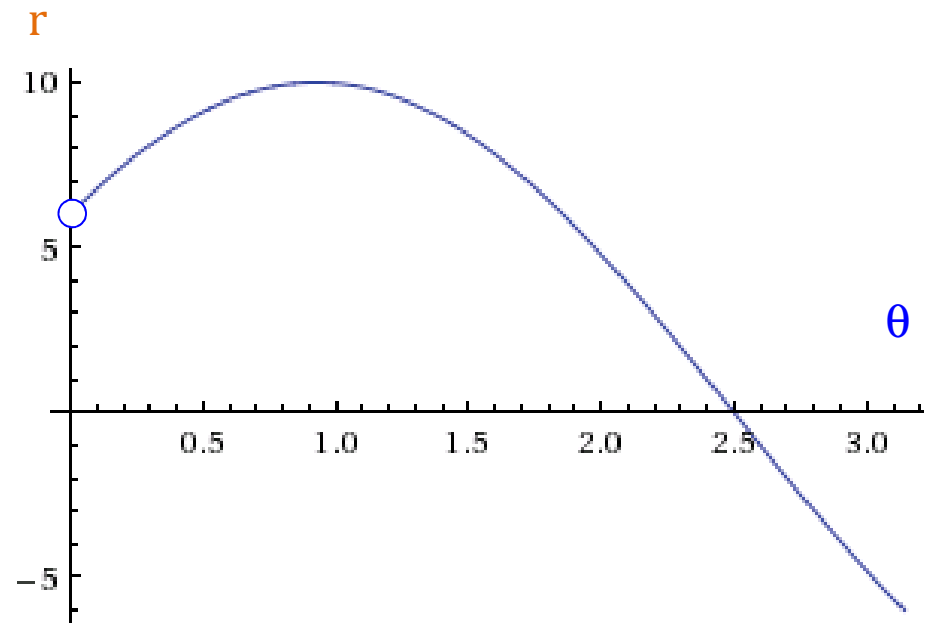
# Hough transform

Point space



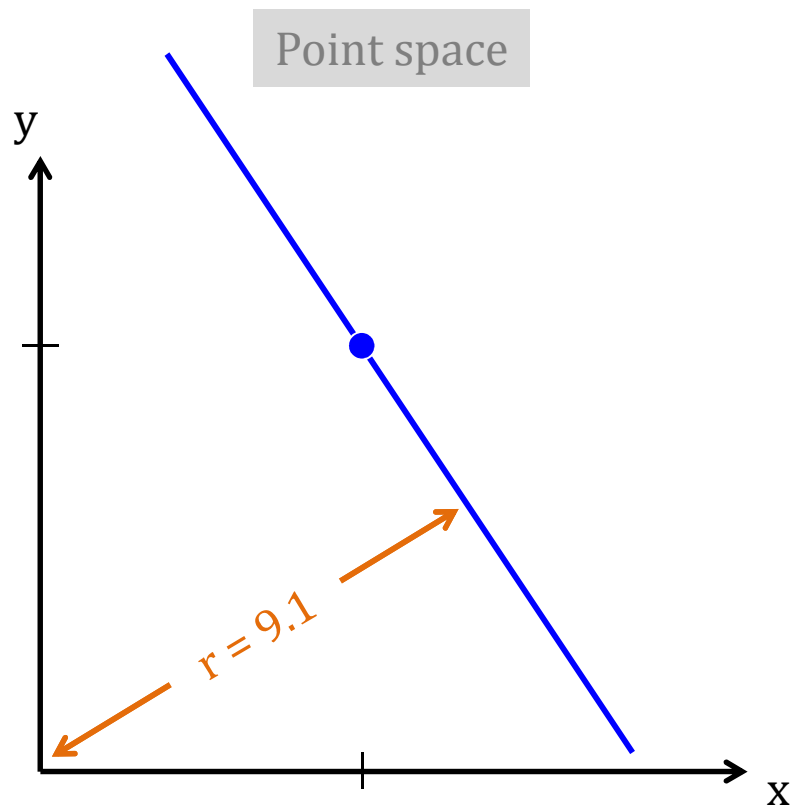
vertical line  $\sim \theta = 0$

Line space

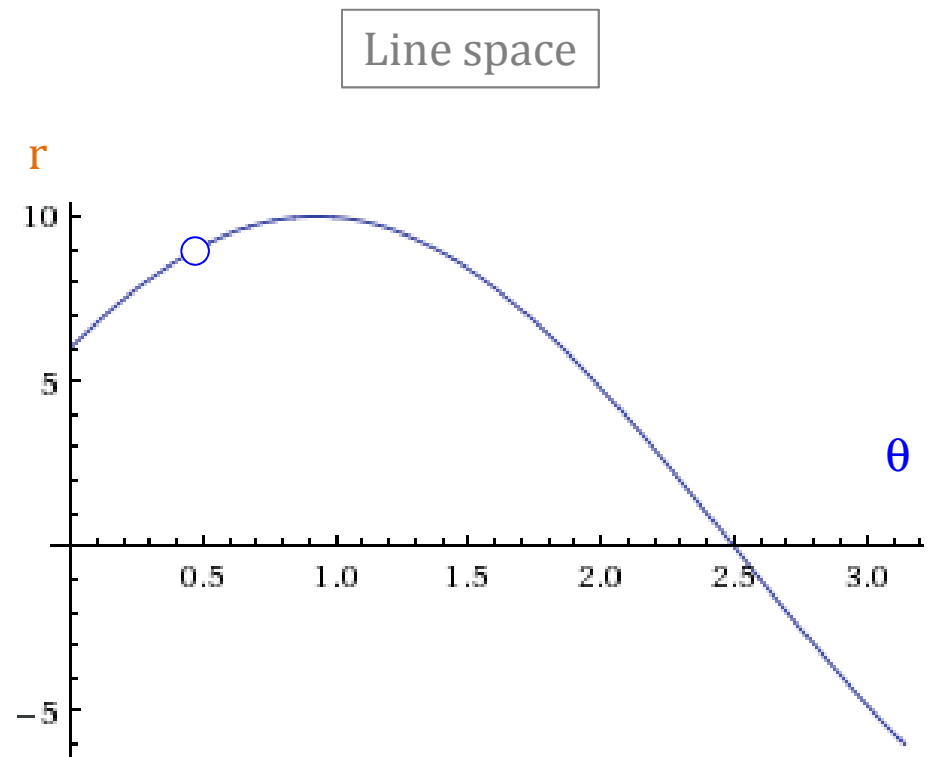


The dot is the line  $r = 6, \theta = 0$

# Hough transform



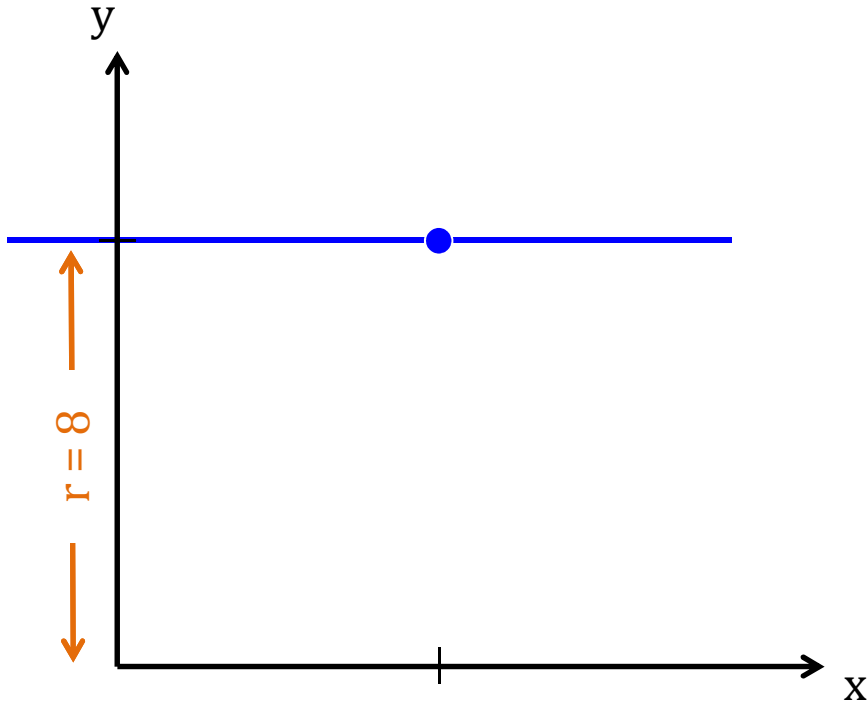
angled line  $\sim \theta = 32^\circ$



The dot is the line  $r = 9.1, \theta = 32^\circ$

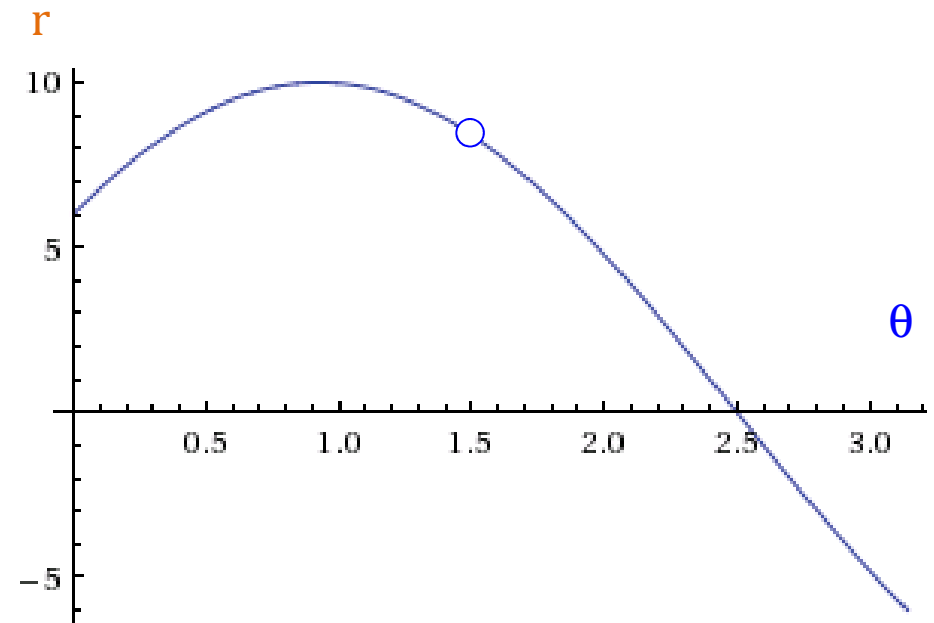
# Hough transform

Point space



angled line  $\sim \theta = 90^\circ$

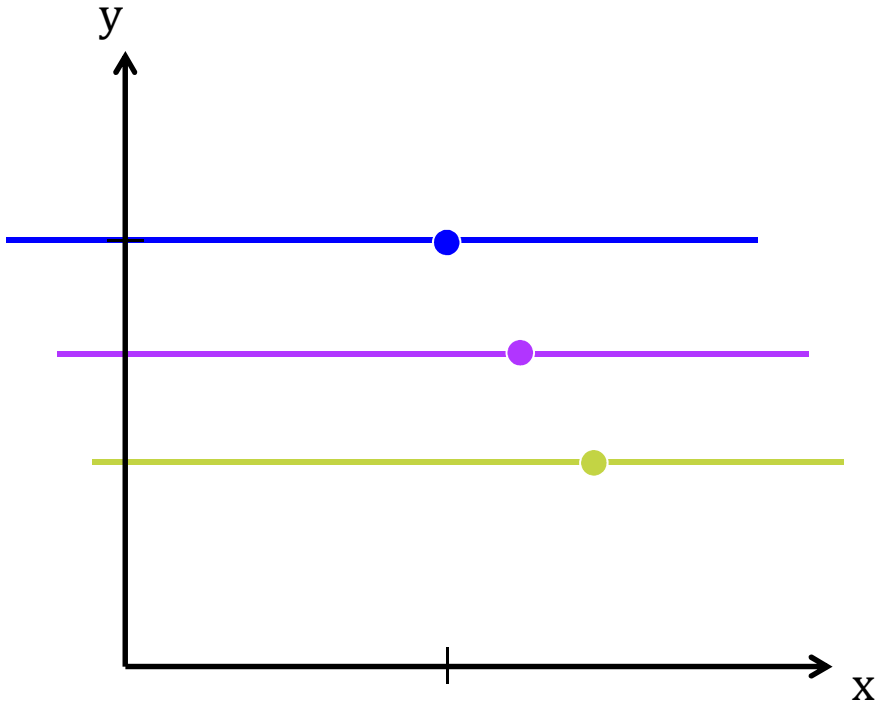
Line space



The dot is the line  $r = 8, \theta = 90^\circ$

# Hough transform

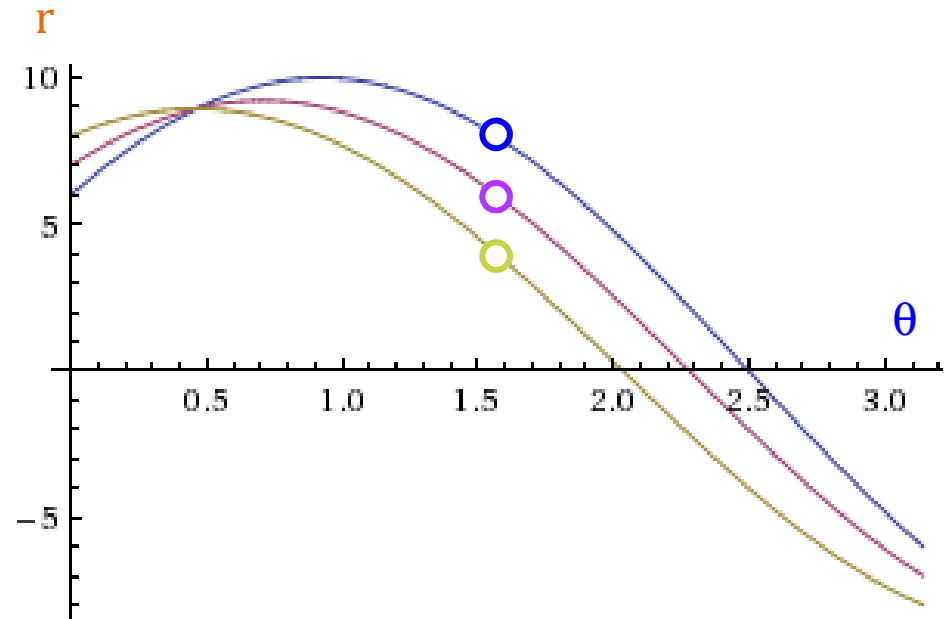
Point space



**three points** of a laser scan

*(horizontal lines are shown)*

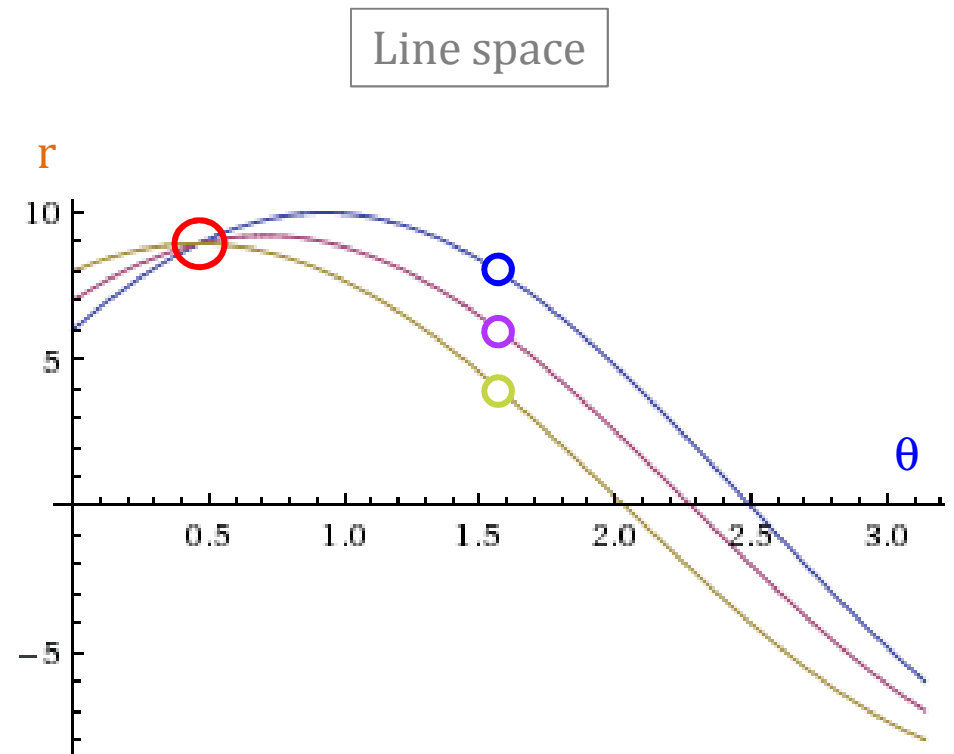
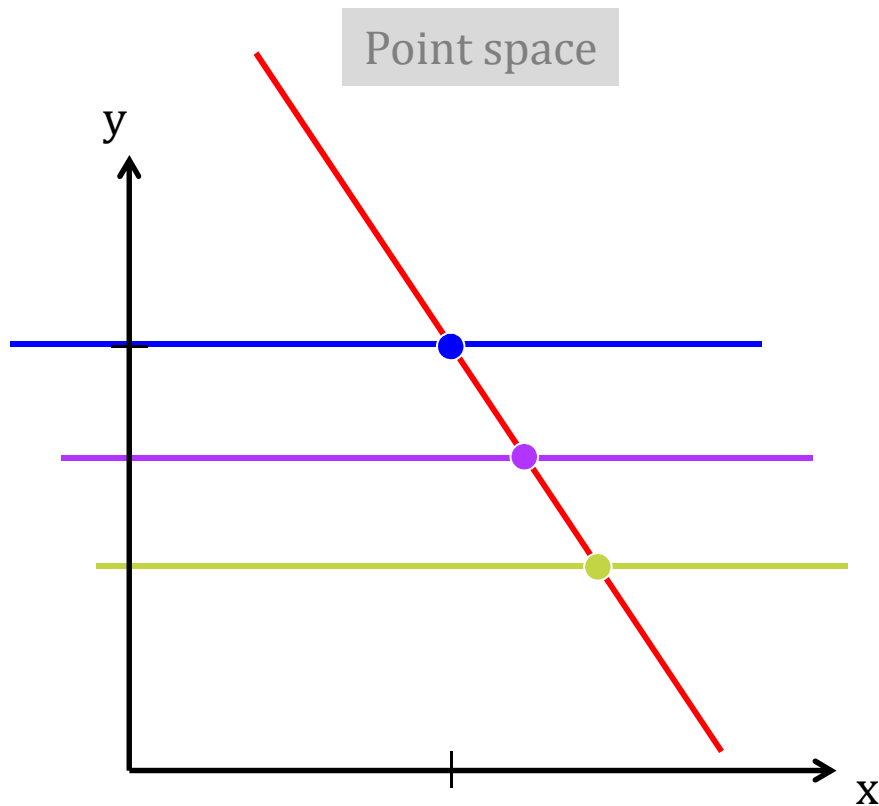
Line space



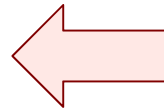
**all** the lines through all three

*(dots ~ the shown horizontal lines)*

# Hough transform

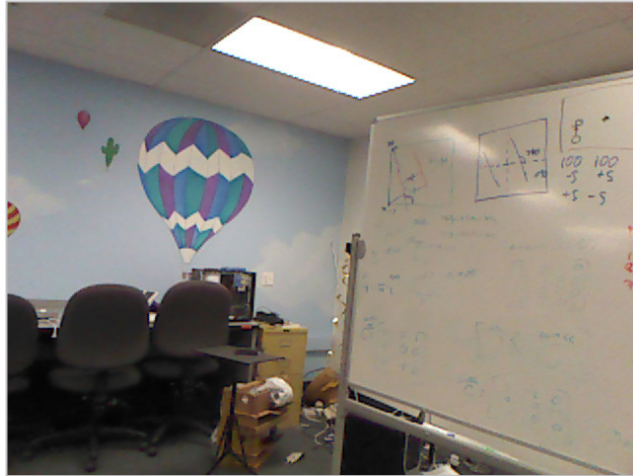


yield lines with substantial support in the original data

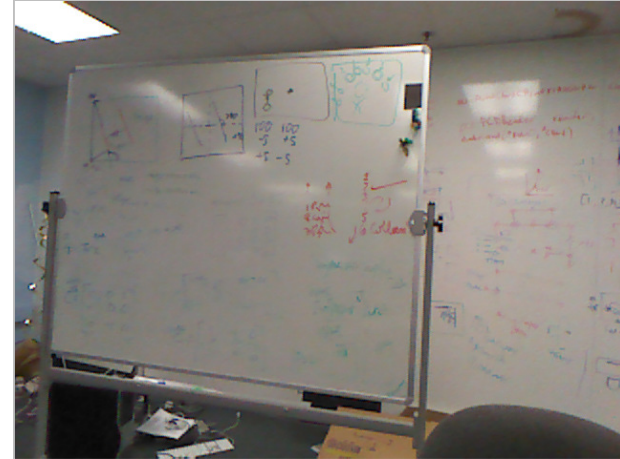
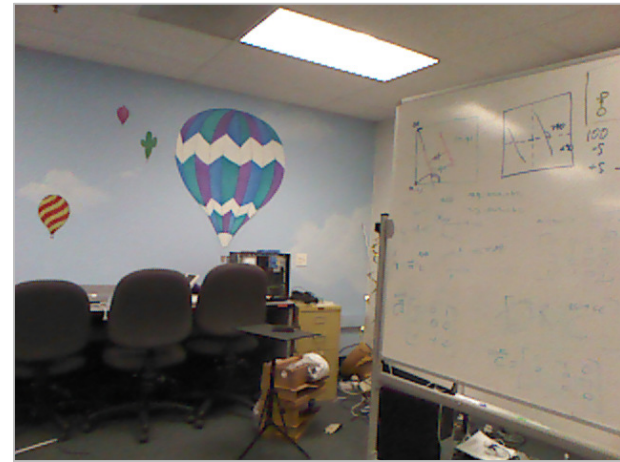


peaks here in line space

# Image matching?



new image



many original images

## Our approach:

- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

# matching?



Our approach:

- **Find features in each (SIFT)**
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

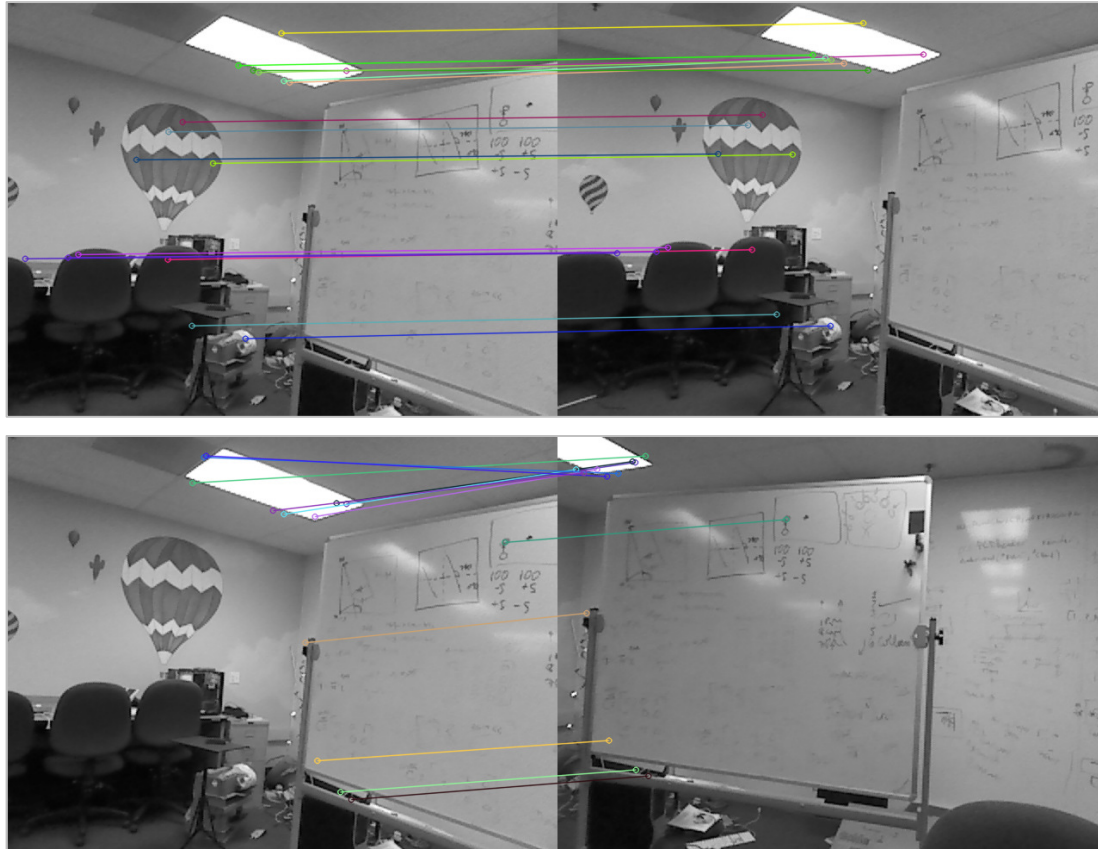
Extract features



## Our approach:

- Find features in each (SIFT)
- **Consider all matches (FLANN)**
- Ensure bidirectional constraints
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

Nearest-neighbor  
matching



## Our approach:

- Find features in each (SIFT)
- Consider all matches (FLANN)
- **Ensure bidirectional constraints**
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

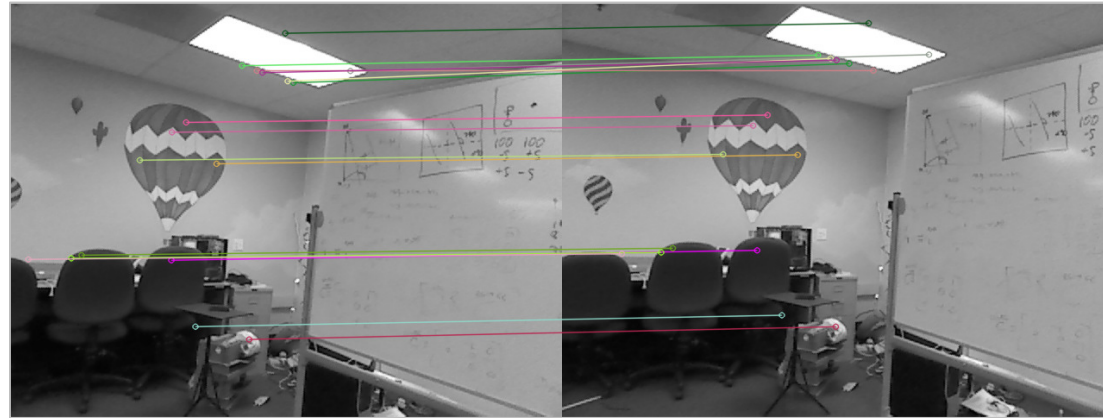
Keep only *two-way*  
best matches



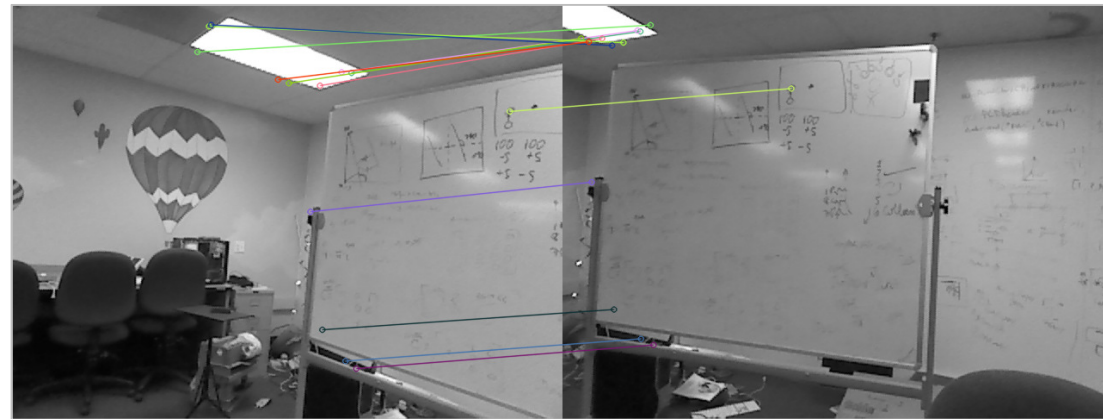
## Our approach:

- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- **Consistency filtering (via homography)**
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

Keep *geometrically realistic* matches



1<sup>st</sup>-best match

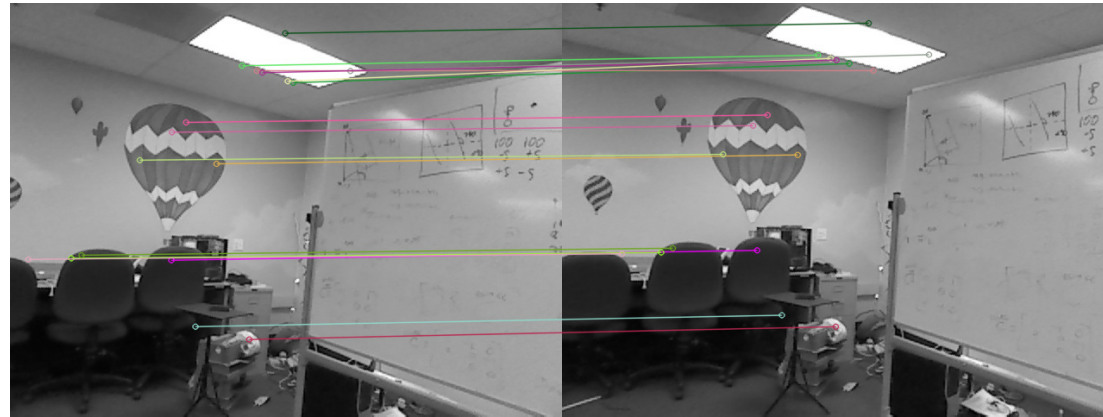


10<sup>th</sup>-best match

Our approach:

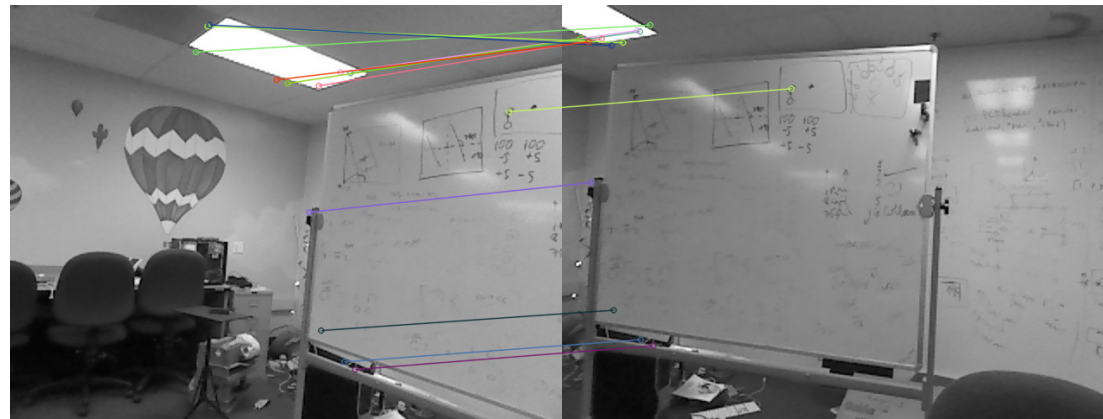
- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (via homography)
- **Best-matching by visual similarity metric**
- Tiebreaking by pixel distances

Visual “closeness”



1<sup>st</sup>-best match

Average feature  
distance ~ **40 px**



10<sup>th</sup>-best match

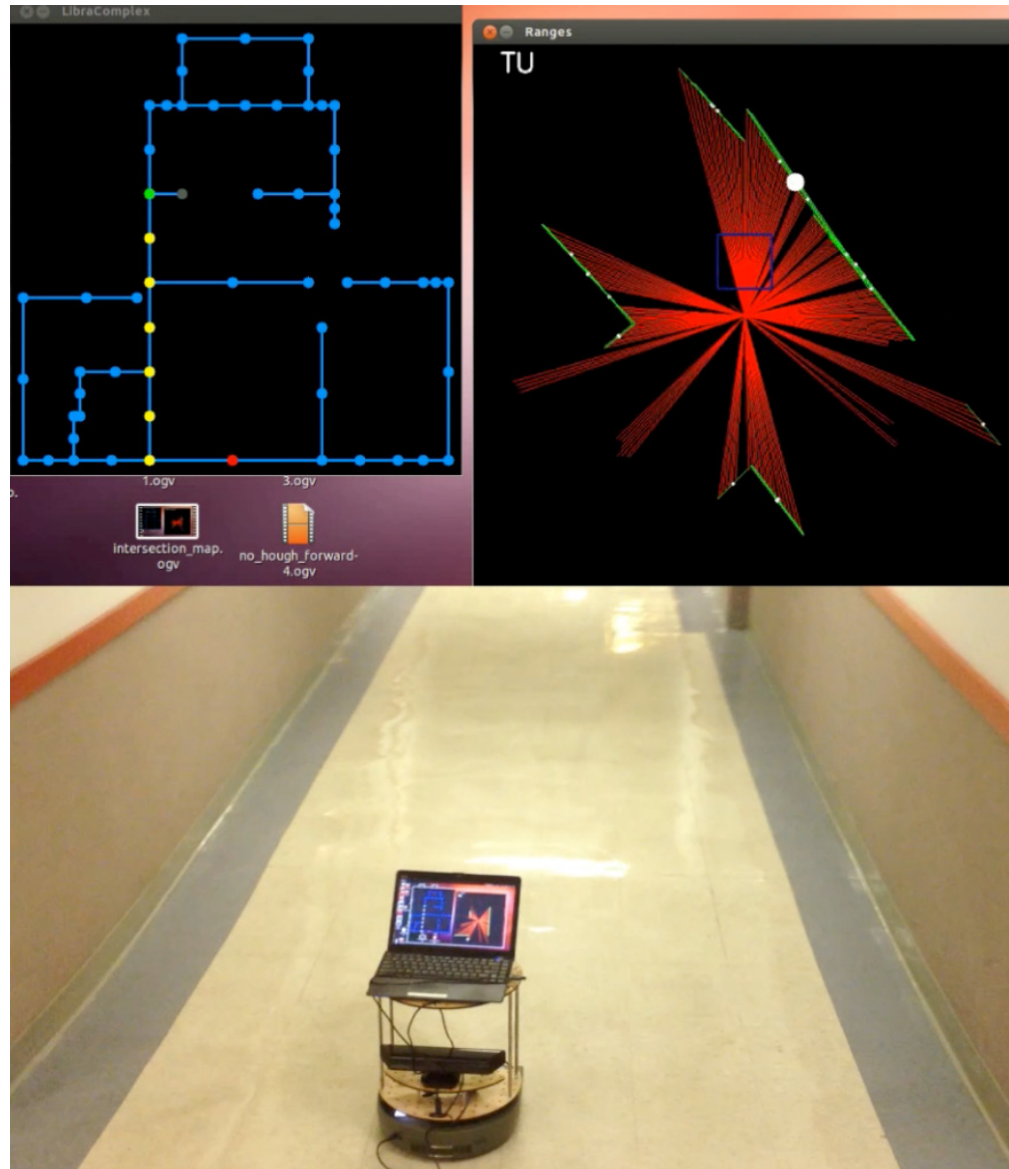
Average feature  
distance ~ **270 px**

Our approach:

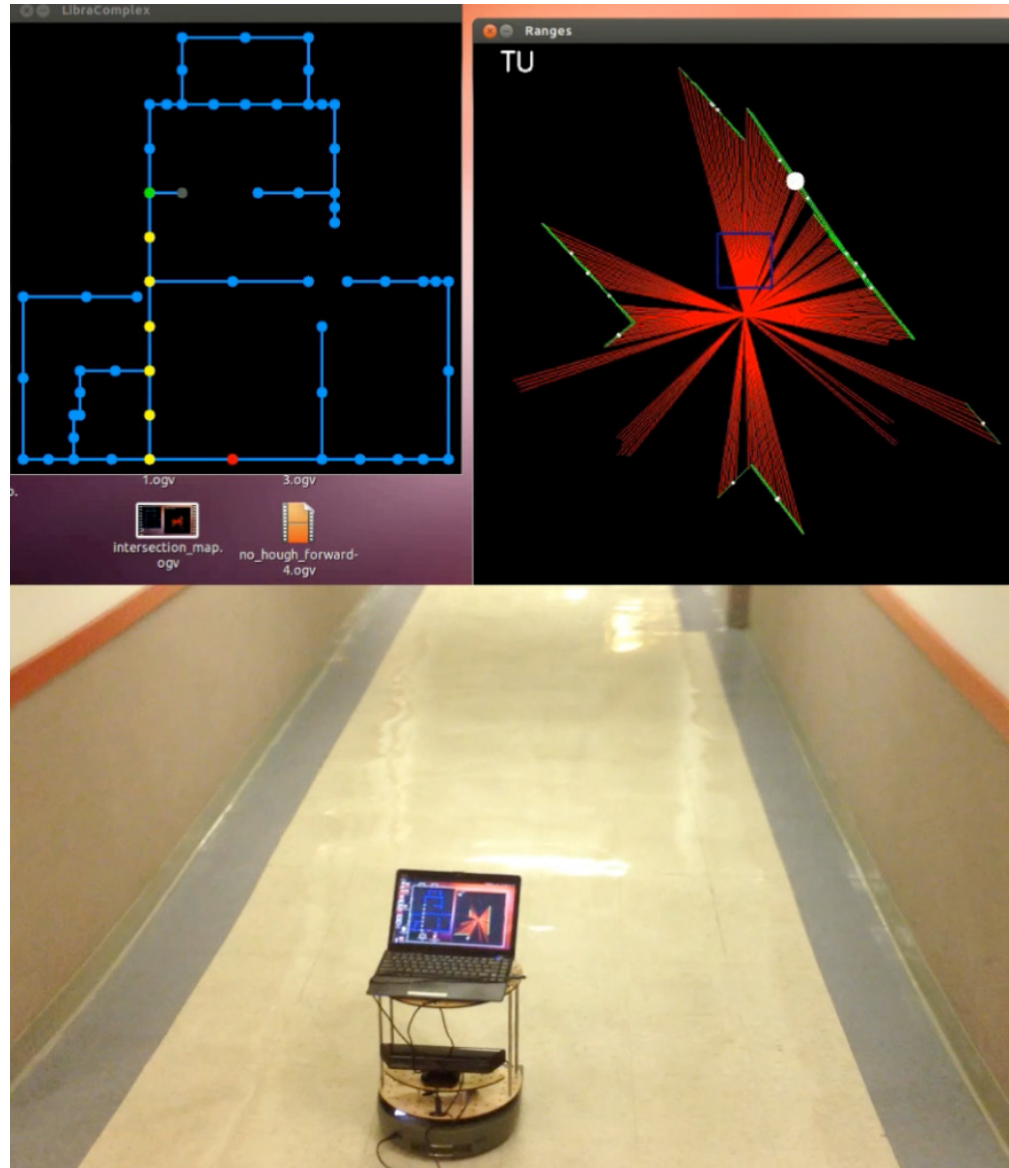
- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (via homography)
- Best-matching by visual similarity metric
- **Tiebreaking by pixel distances**

Pixel “closeness”

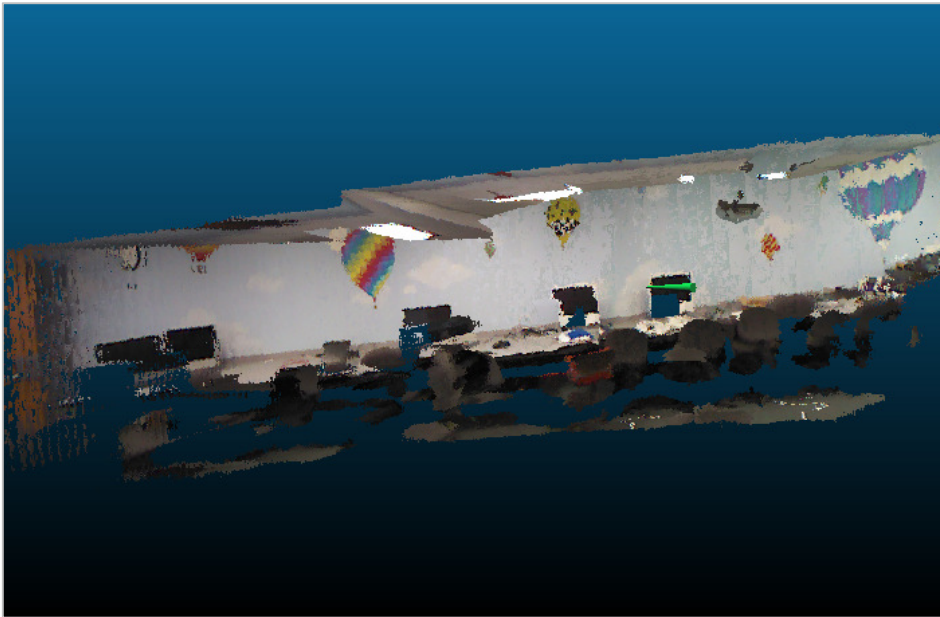
# Map-based navigation...



# Next: localization



# *Accessing* these models

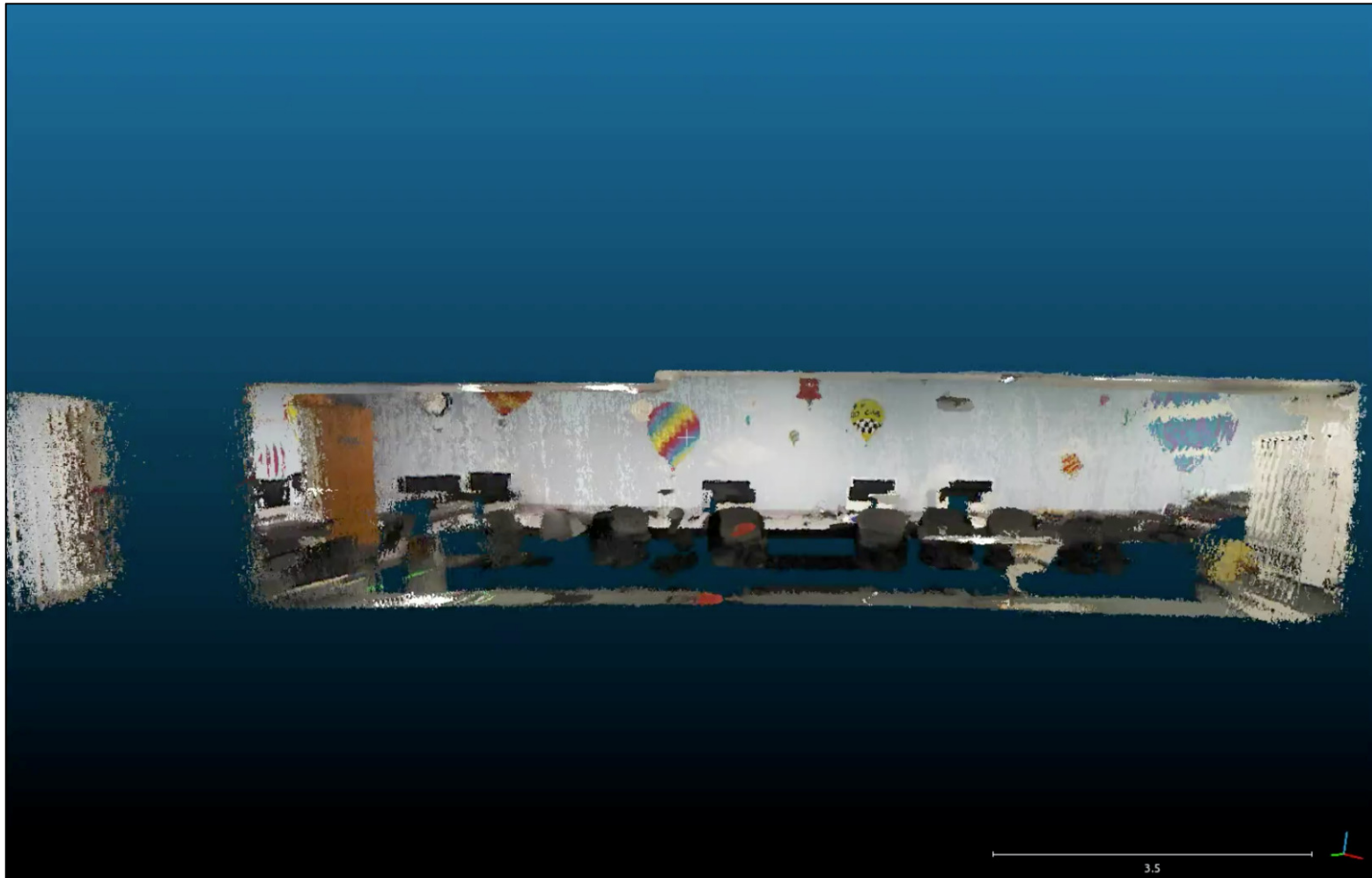


3d lab model ~ with a surface added



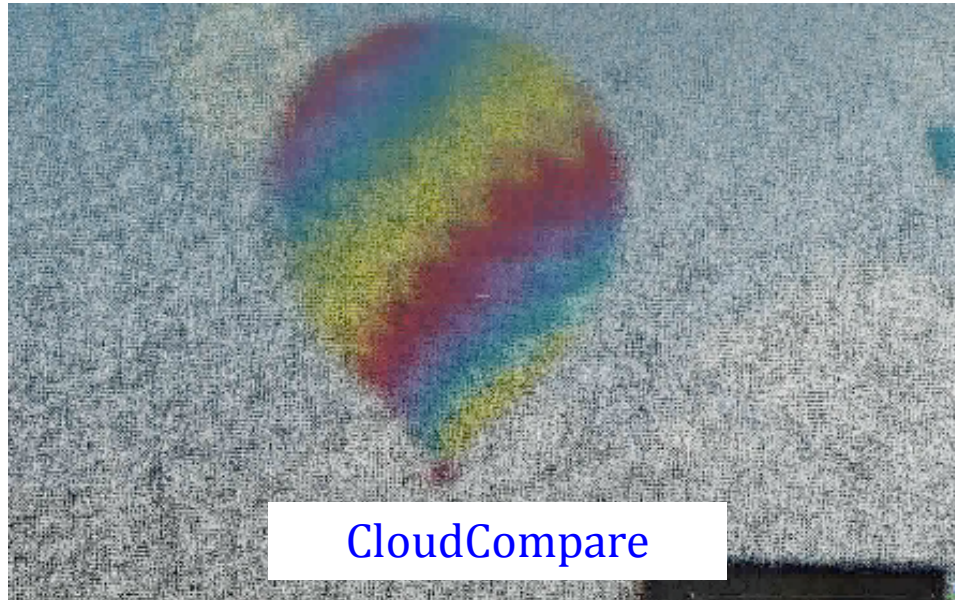
the rendered view from that green cone

# *Accessing* these models



**Cloud Compare** is a visualizer that can include our own “annotations” ...

# Zooming vs. Moving



zooming in to the balloon...



... vs. moving towards/past it

# *Using* these models

How might a robot with an ordinary camera, i.e., only 2d sensing, use these 3d models?

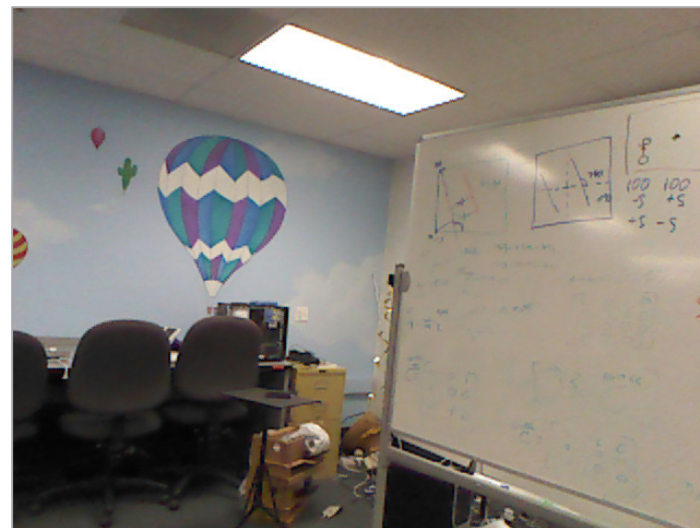
2d – 2d  
matching

3d – 3d  
matching

Ideally, a 2d sensing system could localize –  
*and reason* – within a 3d model

# 2d - 2d matching

with a **new image** from a robot (or other device)



we could compare



vs. original images

These are the images used to create the 3d model in the first place: fewer, higher-fidelity.

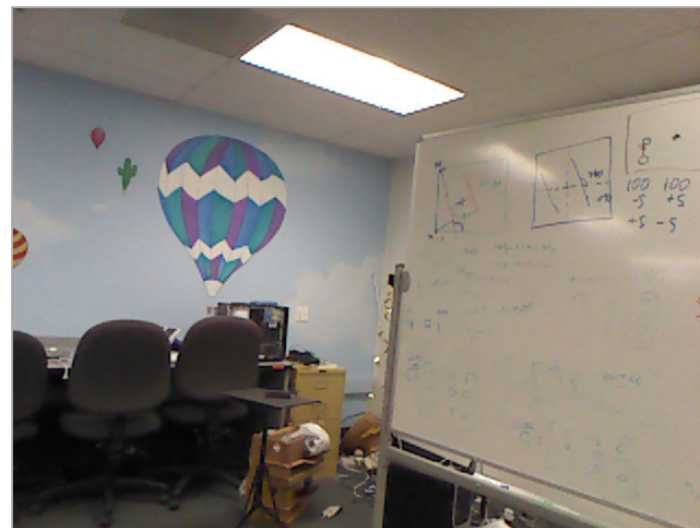


vs. rendered images

Images yielded by the 3d model: arbitrarily many, but lower-quality.

# 2d - 2d matching

with a **new image** from a robot (or other device)



we could compare



vs. original images

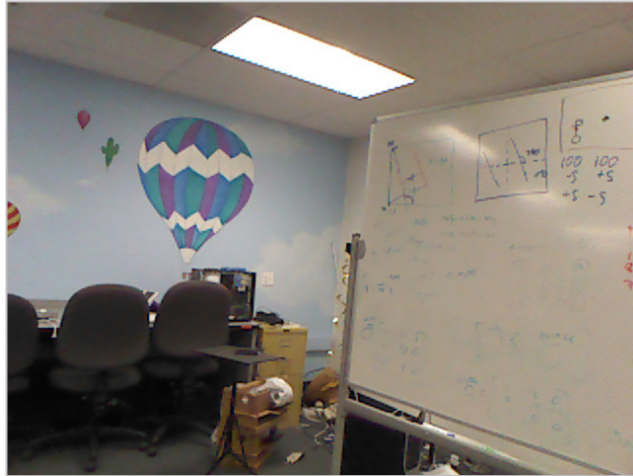
These are the images used to create the 3d model in the first place: fewer, higher-fidelity.



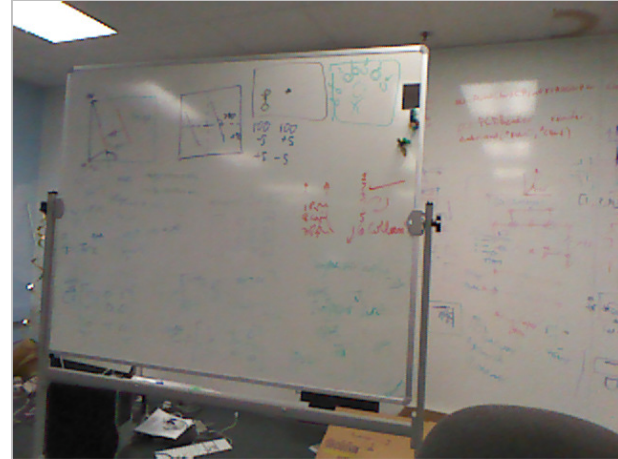
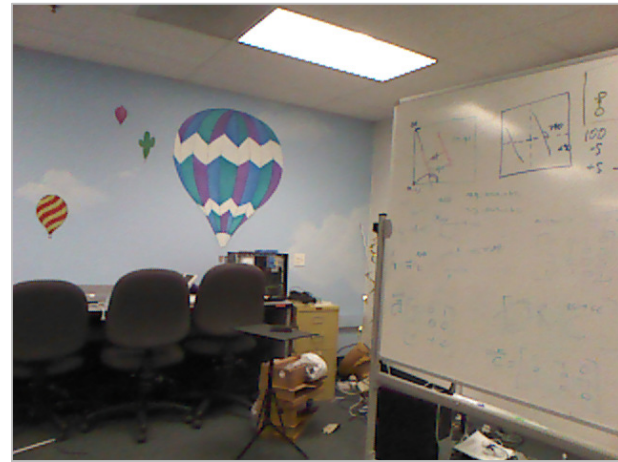
vs. rendered images

Images yielded by the 3d model: arbitrarily many, but lower-quality.

# Matching?



new image

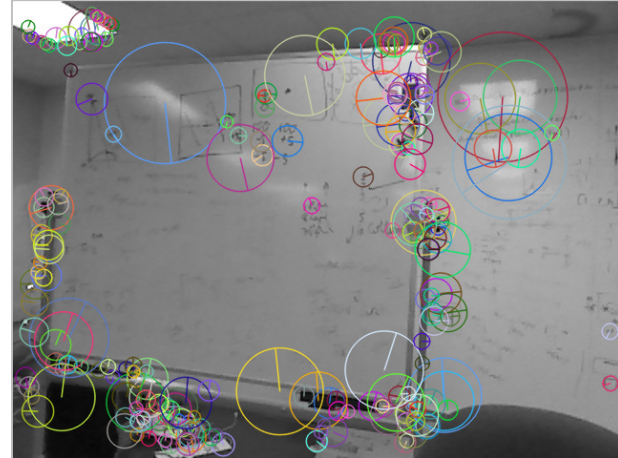
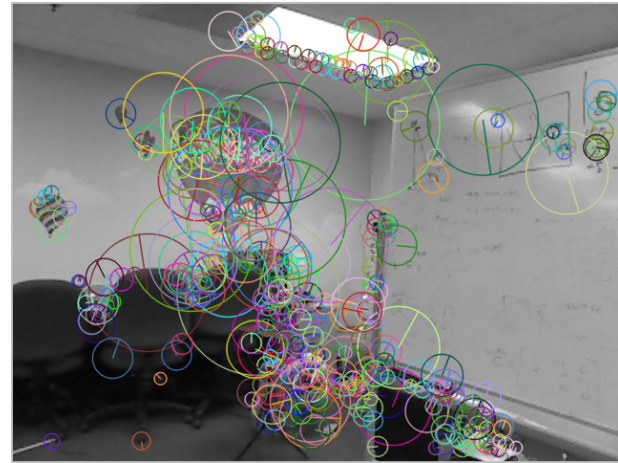
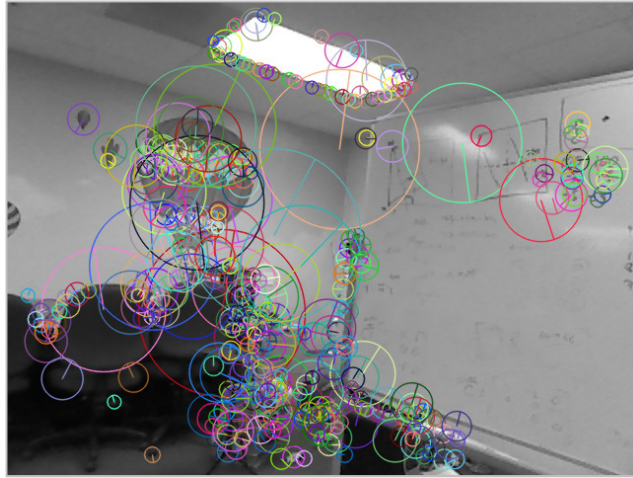


many original images

## Our approach:

- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

# Features



Our approach:

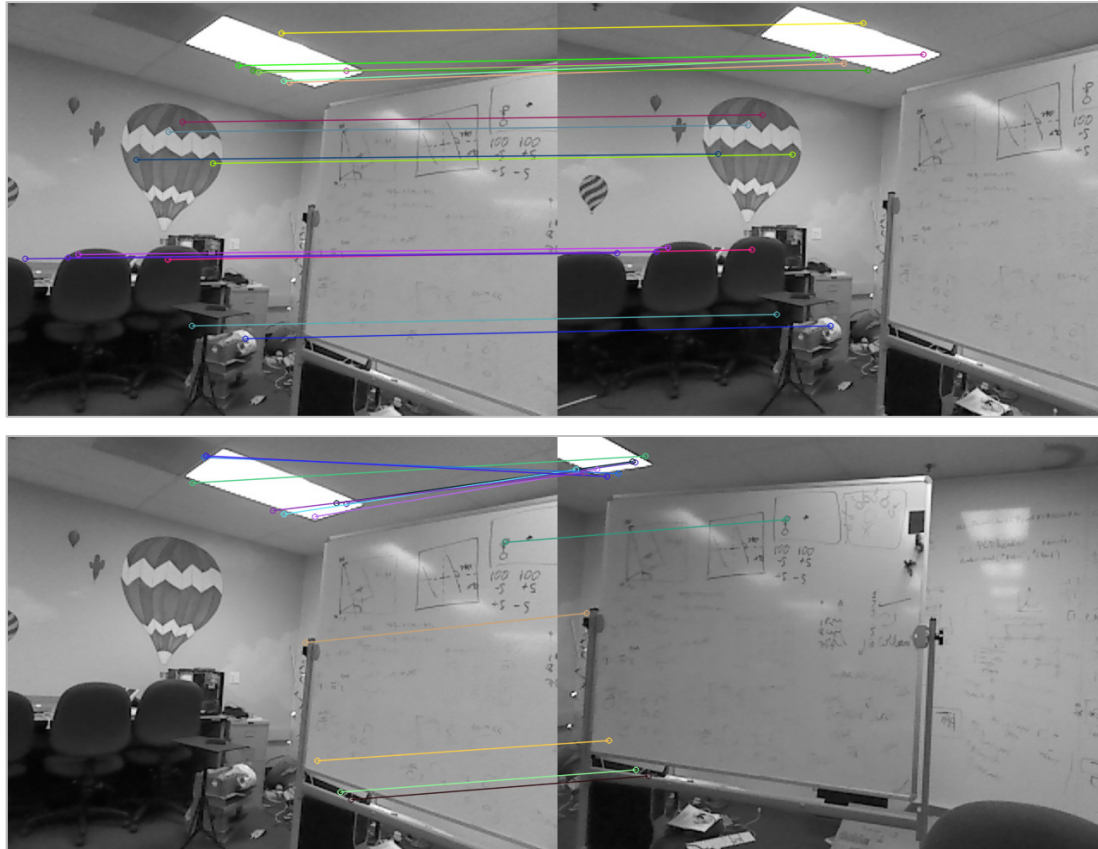
- **Find features in each (SIFT)**
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances



## Our approach:

- Find features in each (SIFT)
- **Consider all matches (FLANN)**
- Ensure bidirectional constraints
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

Nearest-neighbor  
matching



## Our approach:

- Find features in each (SIFT)
- Consider all matches (FLANN)
- **Ensure bidirectional constraints**
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

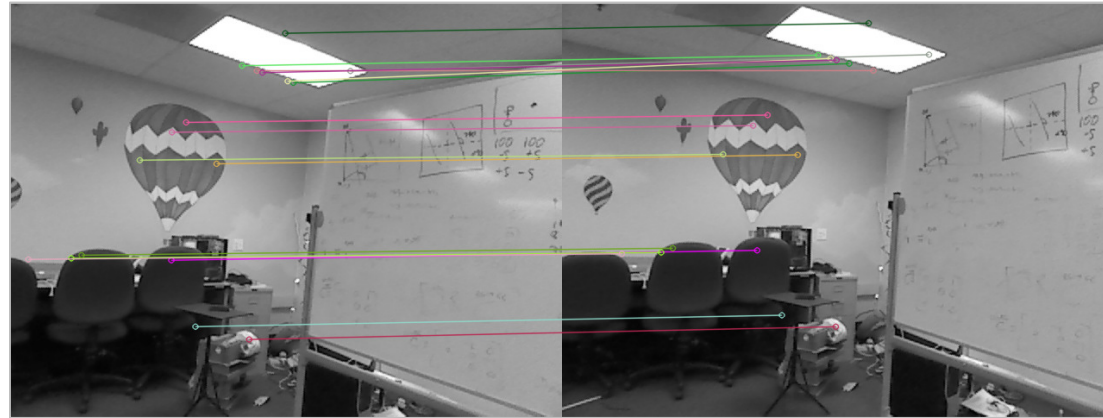
Keep only *two-way*  
best matches



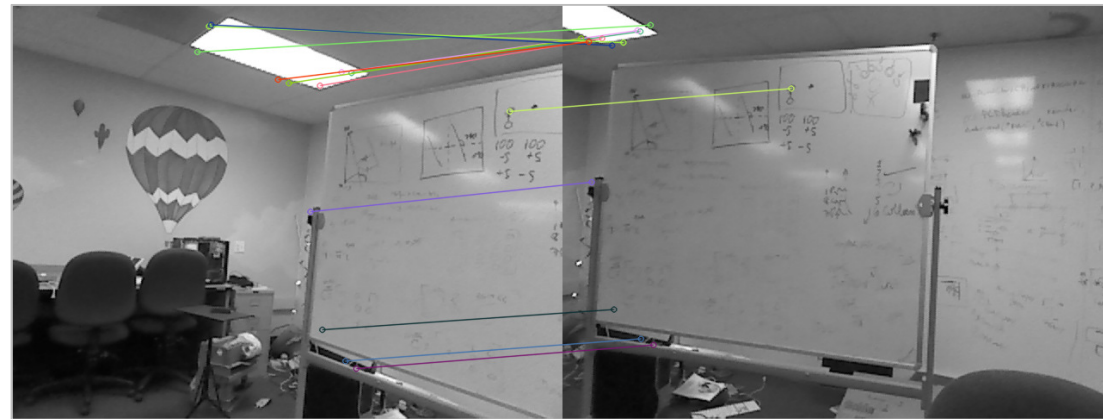
## Our approach:

- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- **Consistency filtering (via homography)**
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

*Keep geometrically realistic matches*



1<sup>st</sup>-best match

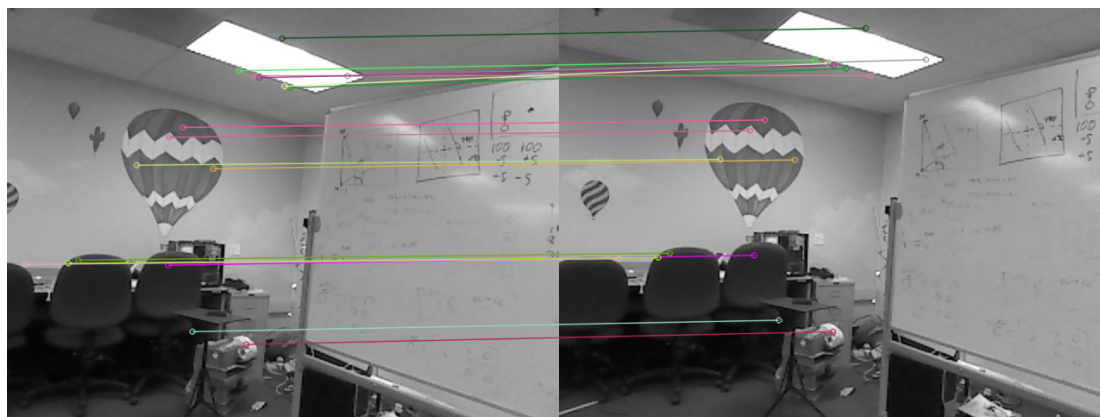


10<sup>th</sup>-best match

Our approach:

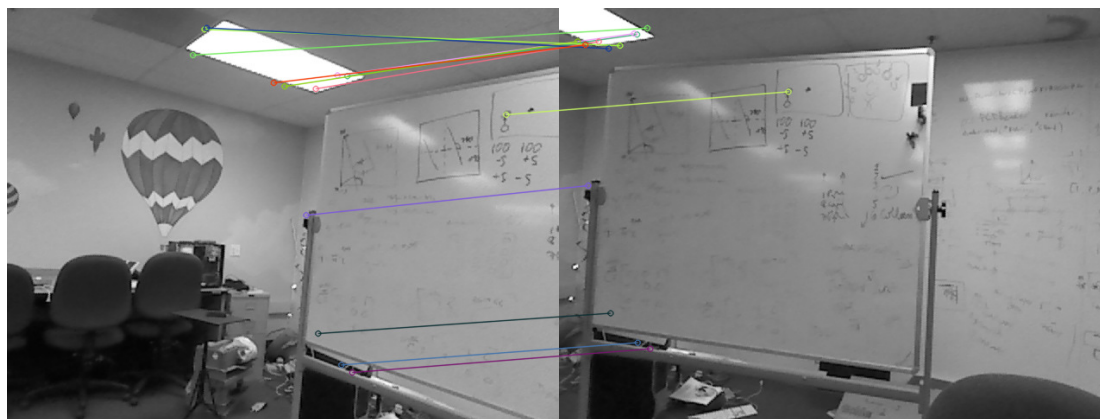
- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (via homography)
- **Best-matching by visual similarity metric**
- Tiebreaking by pixel distances

Visual “closeness”



1<sup>st</sup>-best match

Average feature  
distance ~ **40 px**



10<sup>th</sup>-best match

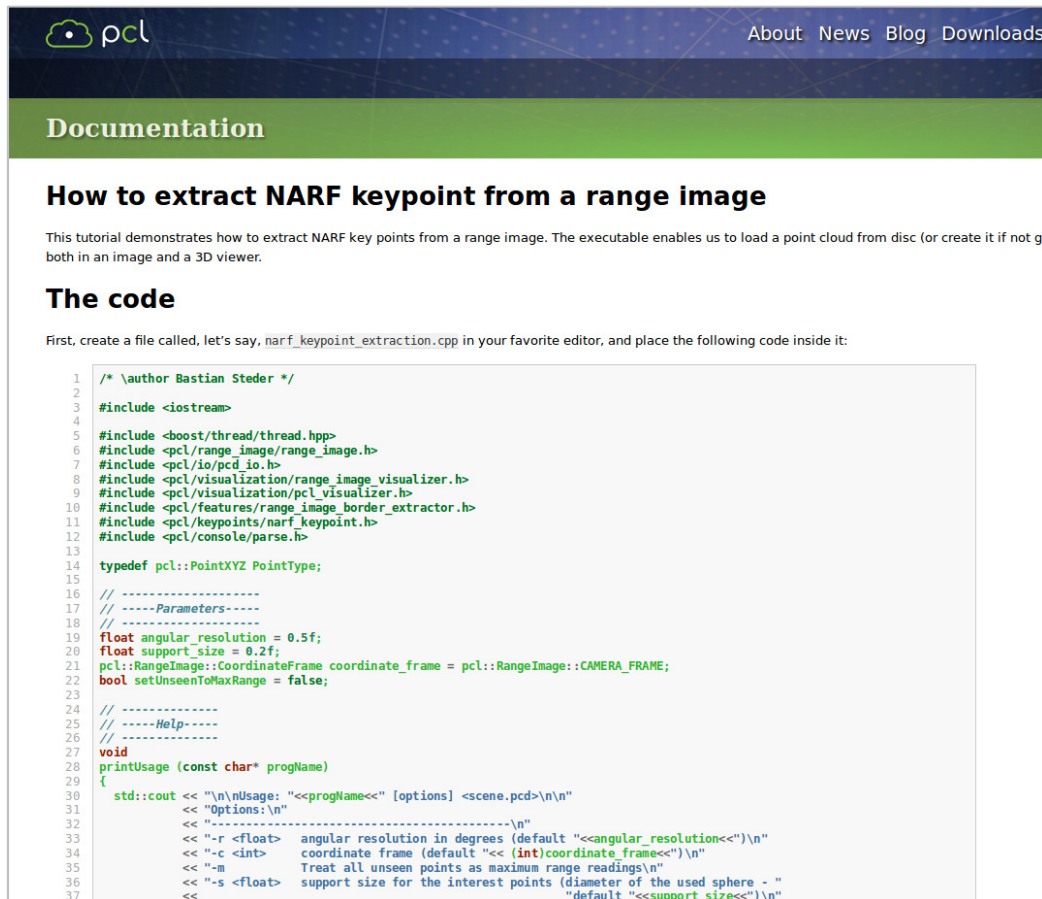
Average feature  
distance ~ **270 px**

Our approach:

- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (via homography)
- Best-matching by visual similarity metric
- **Tiebreaking by pixel distances**

Pixel “closeness”

# PCL: Point Cloud Library



The screenshot shows the PCL website's documentation page. At the top, there is a navigation bar with links for 'About', 'News', 'Blog', and 'Downloads'. Below this is a green header with the word 'Documentation'. The main content area is titled 'How to extract NARF keypoint from a range image'. It includes a brief introduction, a section for 'The code', and a code block containing C++ source code for a program that extracts NARF keypoints from a range image. The code includes various PCL headers, defines a PointXYZ type, sets parameters like angular resolution and support size, and includes a usage function.

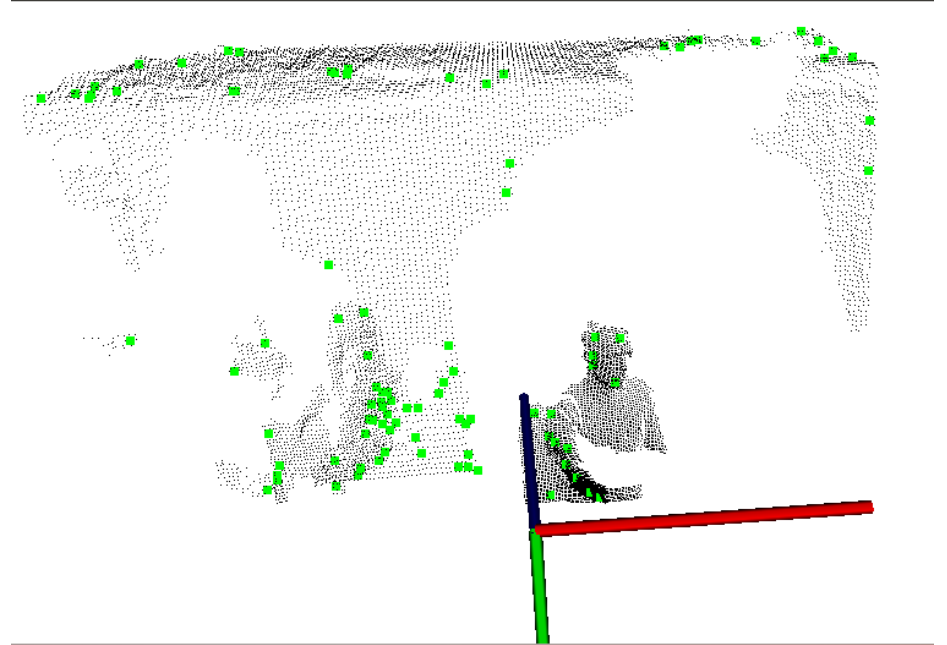
```
1  /* \author Bastian Steder */
2
3  #include <iostream>
4
5  #include <boost/thread/thread.hpp>
6  #include <pcl/range_image/range_image.h>
7  #include <pcl/io/pcd_io.h>
8  #include <pcl/visualization/range_image_visualizer.h>
9  #include <pcl/visualization/pcl_visualizer.h>
10 #include <pcl/features/range_image_border_extractor.h>
11 #include <pcl/keypoints/narf_keypoint.h>
12 #include <pcl/console/parse.h>
13
14 typedef pcl::PointXYZ PointType;
15
16 // -----
17 // -----Parameters-----
18 // -----
19 float angular_resolution = 0.5f;
20 float support_size = 0.2f;
21 pcl::RangeImage::CoordinateFrame coordinate_frame = pcl::RangeImage::CAMERA_FRAME;
22 bool setUnseenToMaxRange = false;
23
24 // -----
25 // -----Help-----
26 // -----
27 void
28 printUsage (const char* progName)
29 {
30     std::cout << "\n\nUsage: "<<progName<<" [options] <scene.pcd>\n\n"
31               << "Options:\n"
32               << "-----\n"
33               << "-r <float>  angular resolution in degrees (default "<<angular_resolution<<")\n"
34               << "-c <int>   coordinate frame (default "<< (int)coordinate_frame<<")\n"
35               << "-m         Treat all unseen points as maximum range readings\n"
36               << "-s <float> support size for the interest points (diameter of the used sphere - "
37               << "default "<<support_size<<")\n"
```

- 3D-Reasoning
- Easy point cloud access



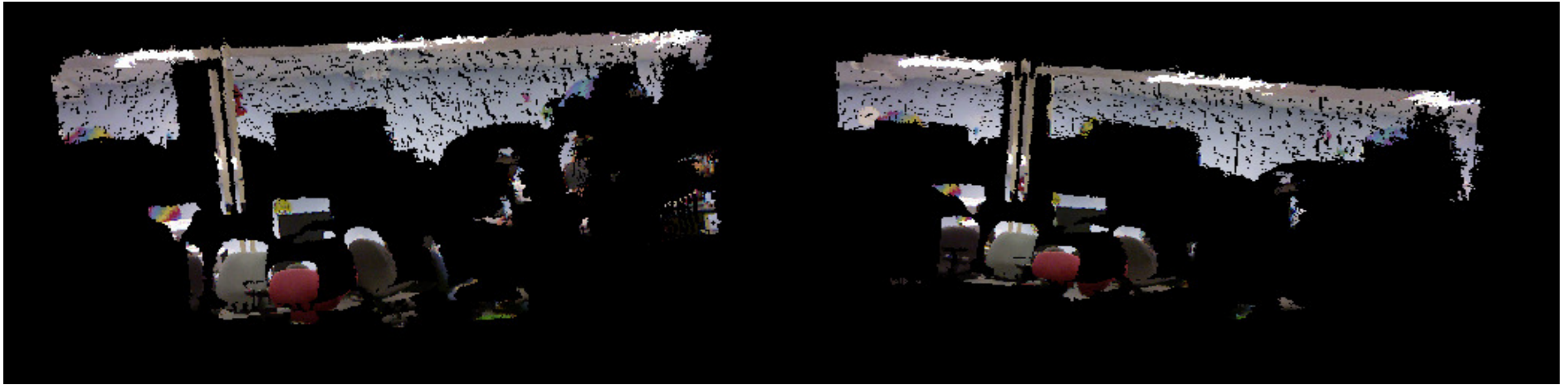
# NARFs!

3d features: *Normal Aligned Radial Features*



- NARF features are purely based on the *range image* of a point cloud: its geometry, not its looks (SIFT)
- They are based on the normals of the point cloud's surfaces
- They are places distinguishable from multiple perspectives.

# 3d matching



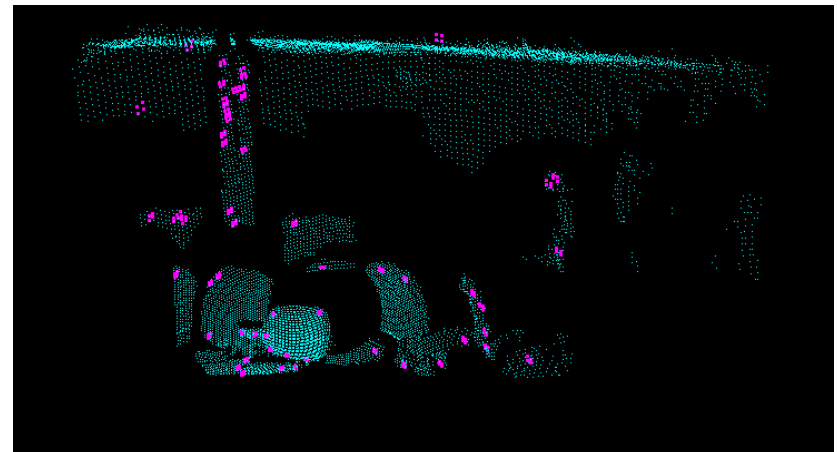
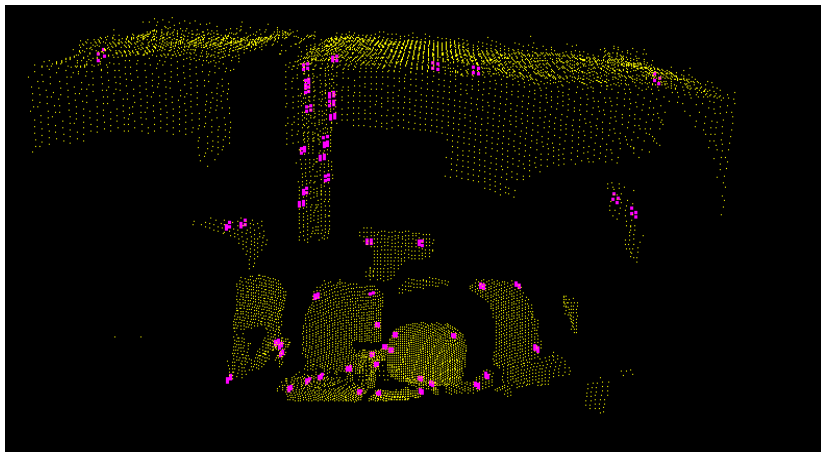
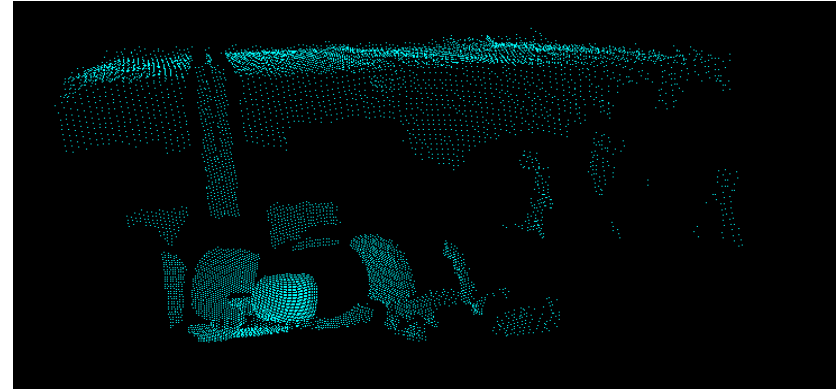
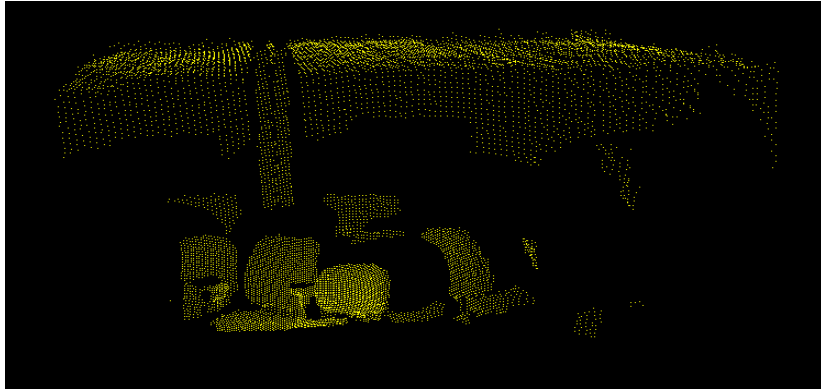
- Take two point clouds
- Find NARFs
- Match features and find transform
- Make transform and combine



naive merging

# 3d SLAM

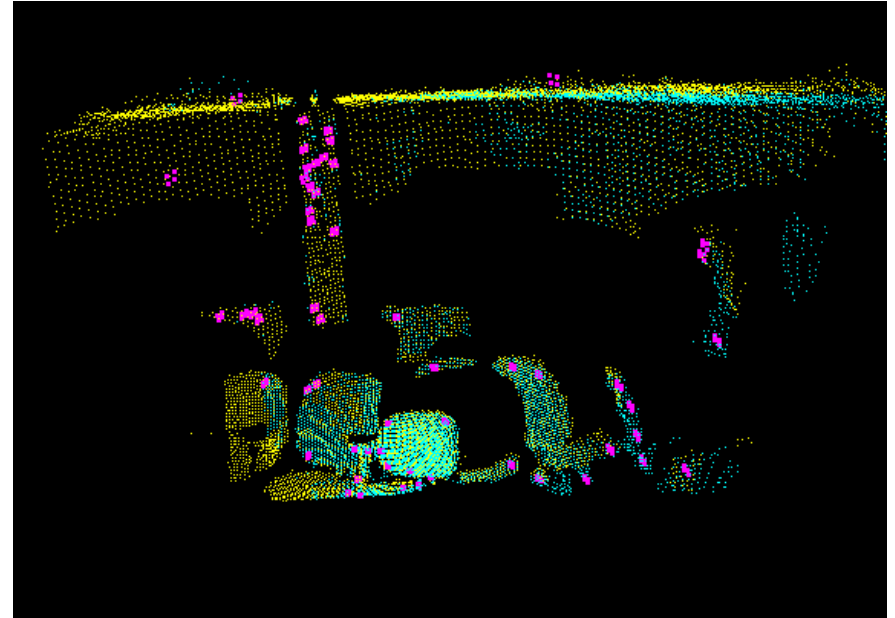
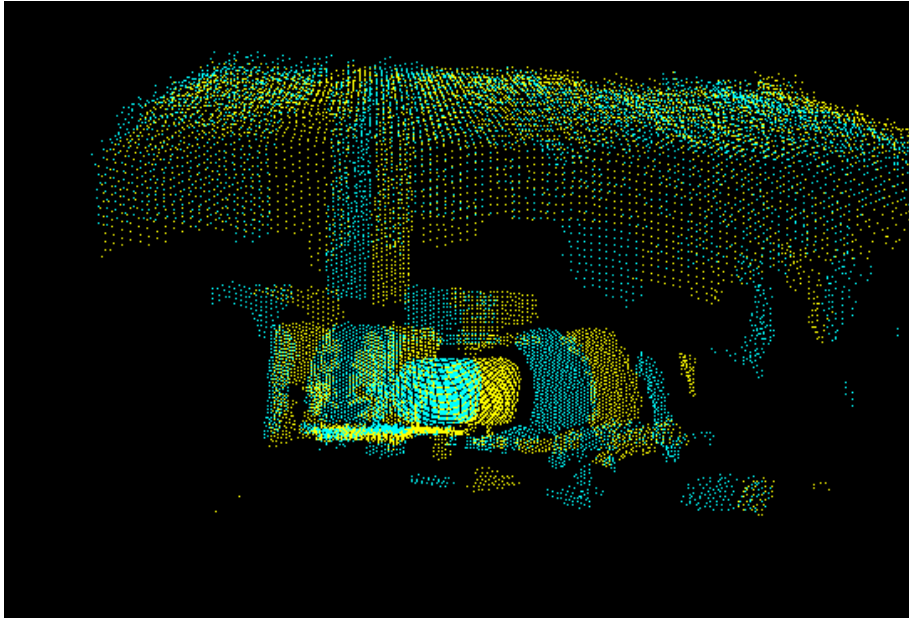
## Simultaneous Localization And Mapping



- Take two point clouds
- Find NARFs
- Match features and find transform
- Make transform and combine

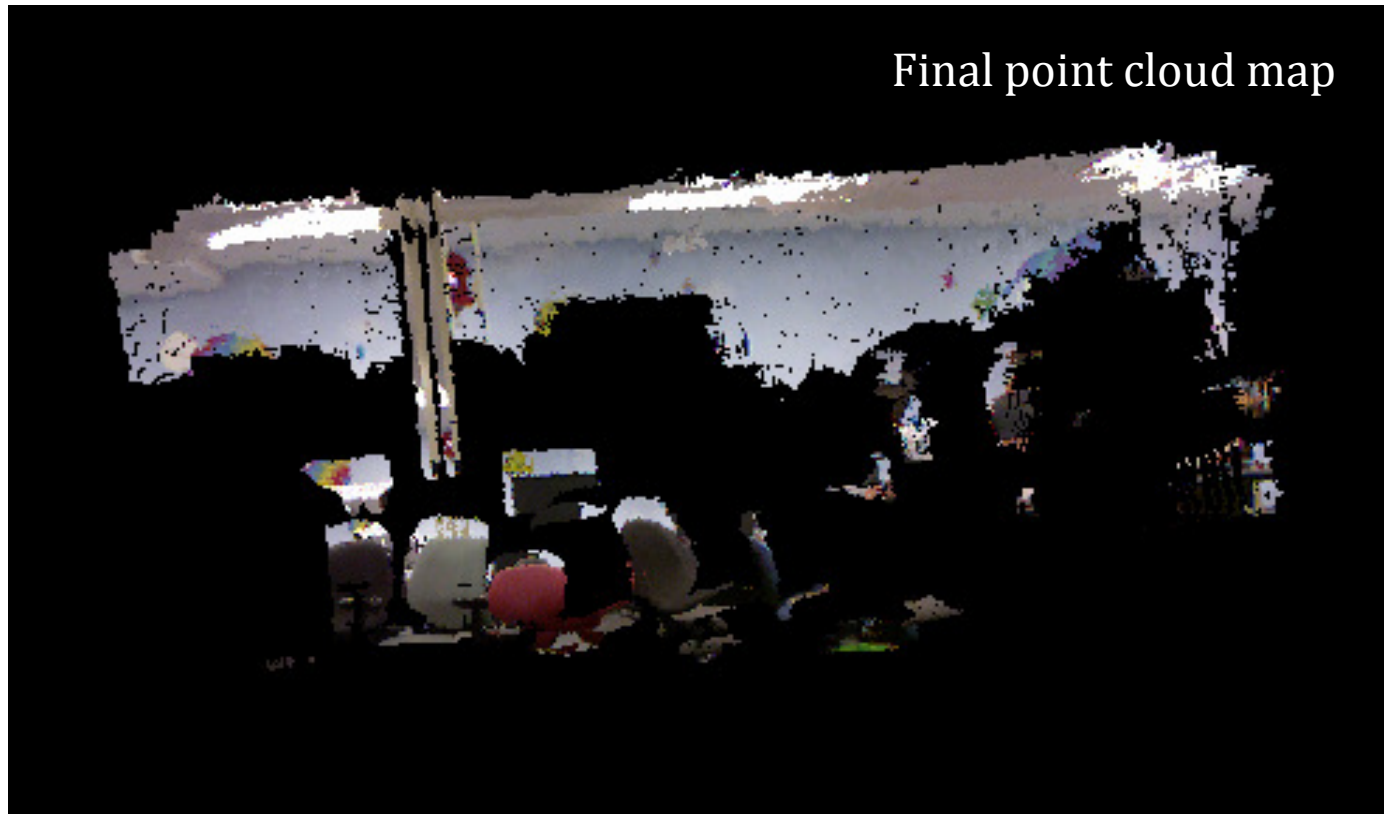
# 3d SLAM

## Simultaneous Localization And Mapping



- Take two point clouds
- Find NARFs
- Match features and find transform (ICP)
- Make transform and combine

# 3d SLAM



Questions?

***Future:*** using Neato to create 2.5d maps

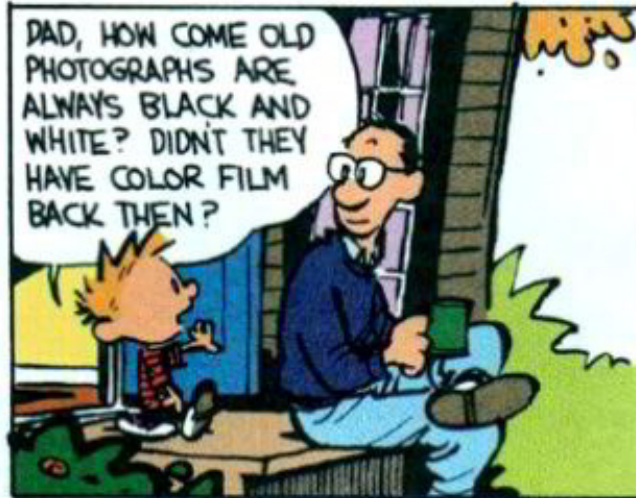


# Kinect

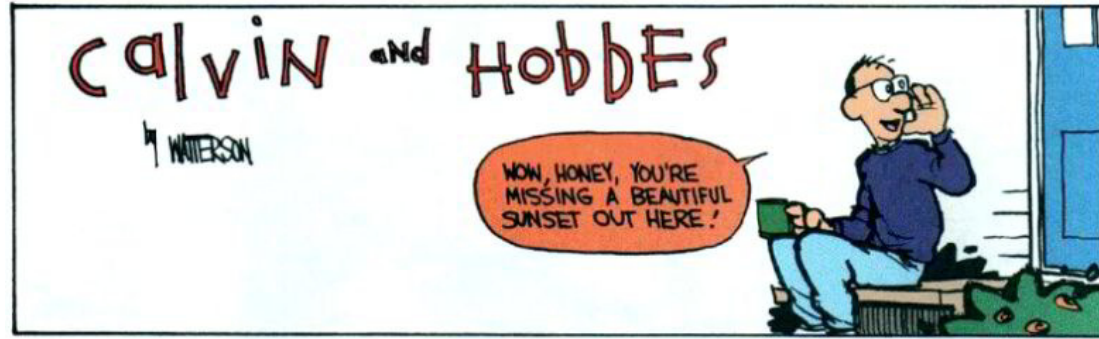
Z.D. video

The Kinect estimates distance-to-camera by projecting a “star field” on the world.

# Now vs. Then



# Title



# *Accessing* these models

plane

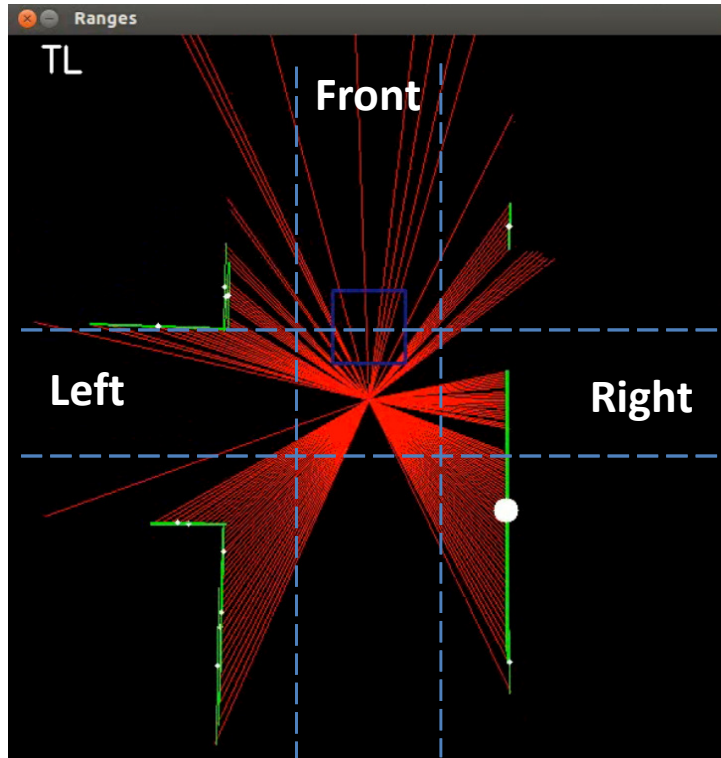
other (sphere, cylinder, wavy,...)  
K.M.

allows us to define the depth at each pixel

Ideally, a camera-only system could localize – *and reason* – within the 3d model

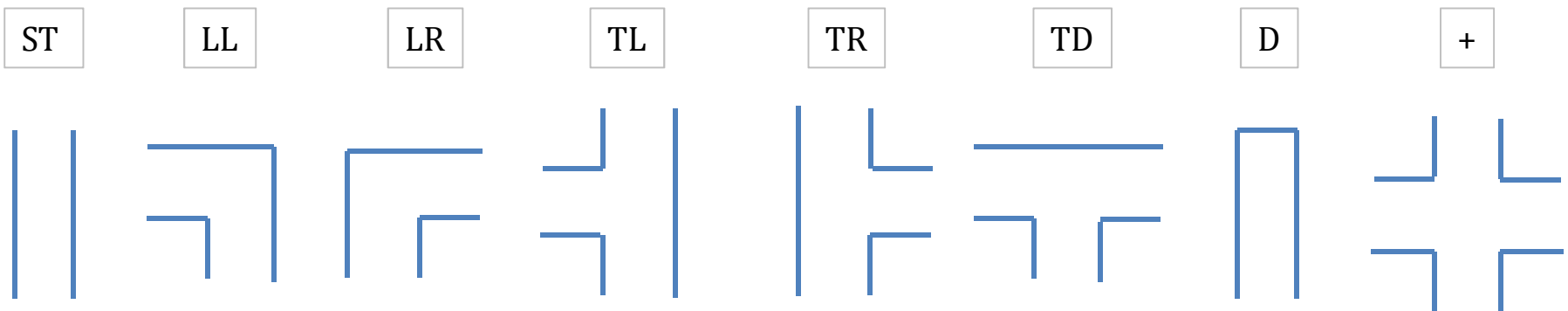
Ideally, a camera-only system could localize – *and reason* – within the 3d model

# Intersection recognition

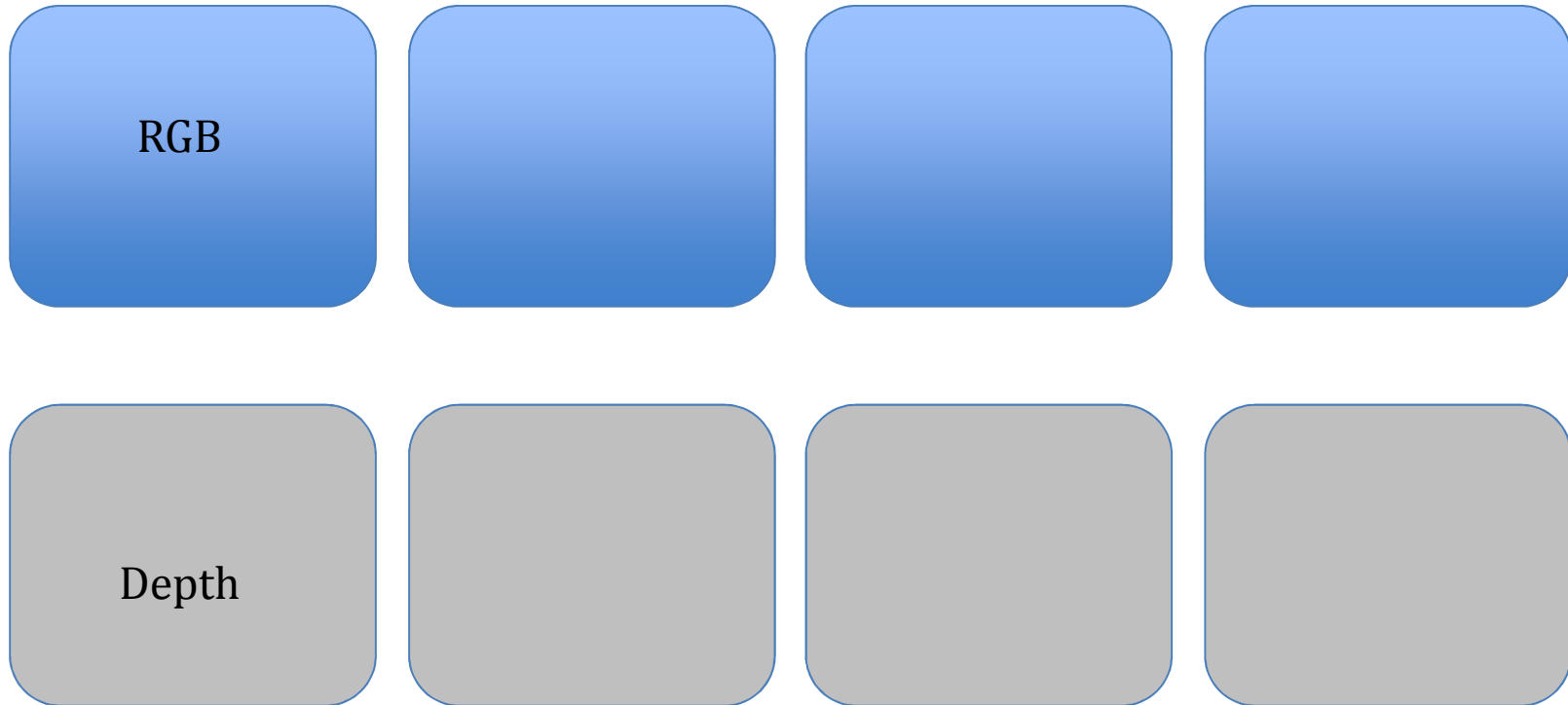


For each scan:

- Check for **Front**, **Left**, and **Right** obstacles
- The number of obstacles determines the intersection type 3:**D** 2:**L** 1:**T** 0:**+**
- Open space determines final label, e.g., **TL**
- After 15 recognitions in a row, the robot checks the map (by broadcasting a message)
- The robot does not broadcast again until it resets in a hallway: **ST**.

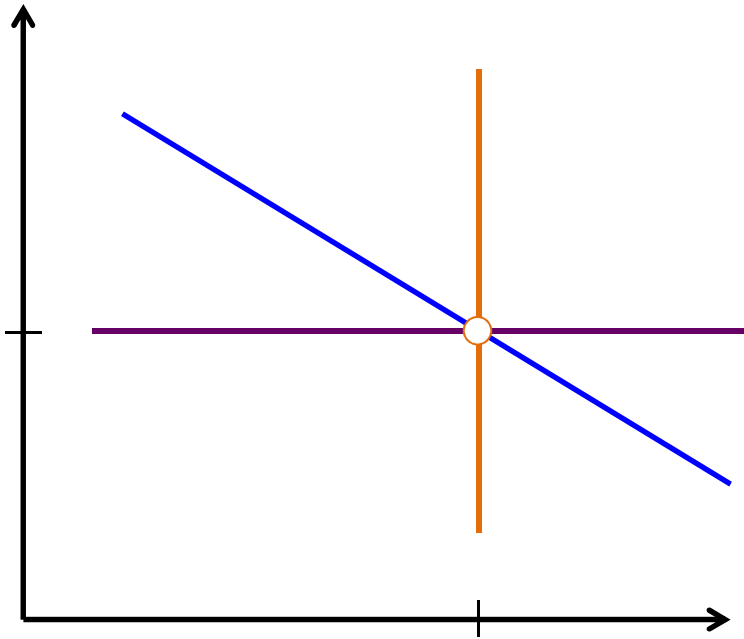


# *Accessing* 3d models

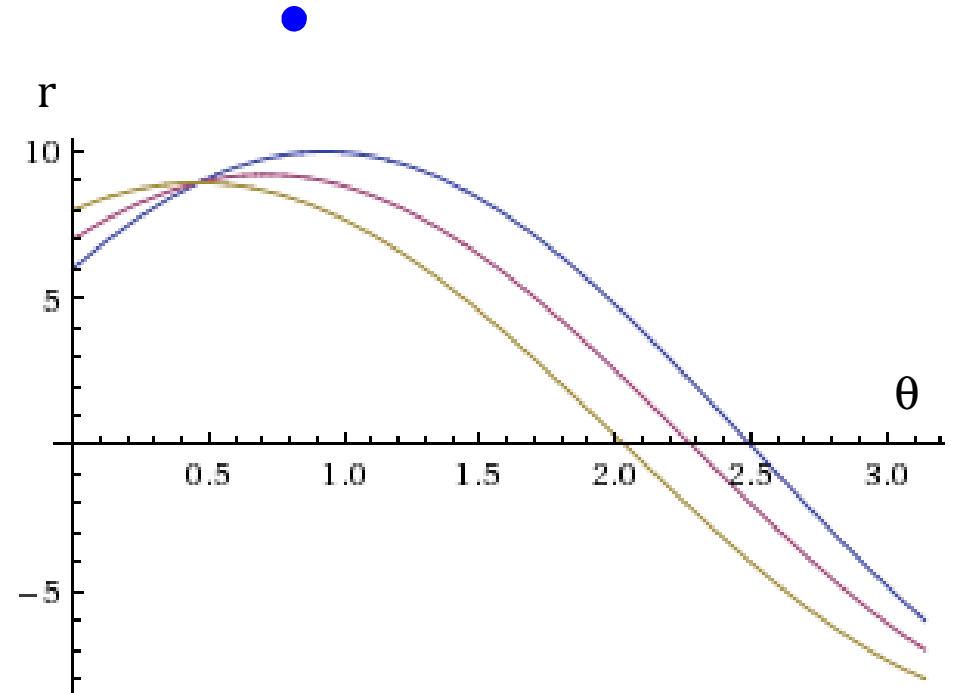


splicing into the code...

# Hough transform



Consider *one point* of a laser scan



These are *all* of the lines through it

# Can we be more *deliberate*?

Each intersection offers a choice:

*Which turn to make, even if the robot isn't sure where it is?*

map of current  
knowledge...

Possible results if the robot  
heads straight

Possible results if the robot  
turns the corner

Possible results if the robot  
reverses course

Consider all of the possible *distributions* of results...

# Entropy

~ the *expected* amount of information available from a distribution of possibilities.

example 1 bit

example of 2 bits

example of 1.05 bits

in general

# Entropy-based localization

Each intersection offers a choice:

*Which turn to make, even if the robot isn't sure where it is?*

Possible results if the robot  
heads straight bits

map of current  
knowledge...

Possible results if the robot  
turns the corner bits

Possible results if the robot  
reverses course bits

We choose the action with the greatest *expected information*.

# Entropy-based localization

```
robotics@robotlaptop4: ~/rosbots/neato_mudd/scripts
roscore http://rob...  robotics@robotlap...  robotics@robotlap...  robotics@robotlap...
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ ^C
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ python node_entrop
robotics@robotlaptop4:~/rosbots/neato_mudd/scripts$ python node_entrop
dst node ls 8
L 4.2
R 100000
S 5.3
TRND 4.6
The robot is going to turn L
```

```
dst node ls 8
L 4.2
R 100000
S 5.3
TRND 4.6
The robot is going to turn L
```



entro\_short.mp4

# Could we be **bolder**?

Leap before you look.

*~ W. H. Auden*

Just do it.

*~ Nike*

... perhaps if robots felt more contemplative.



# Predictive path planning

rn to make?

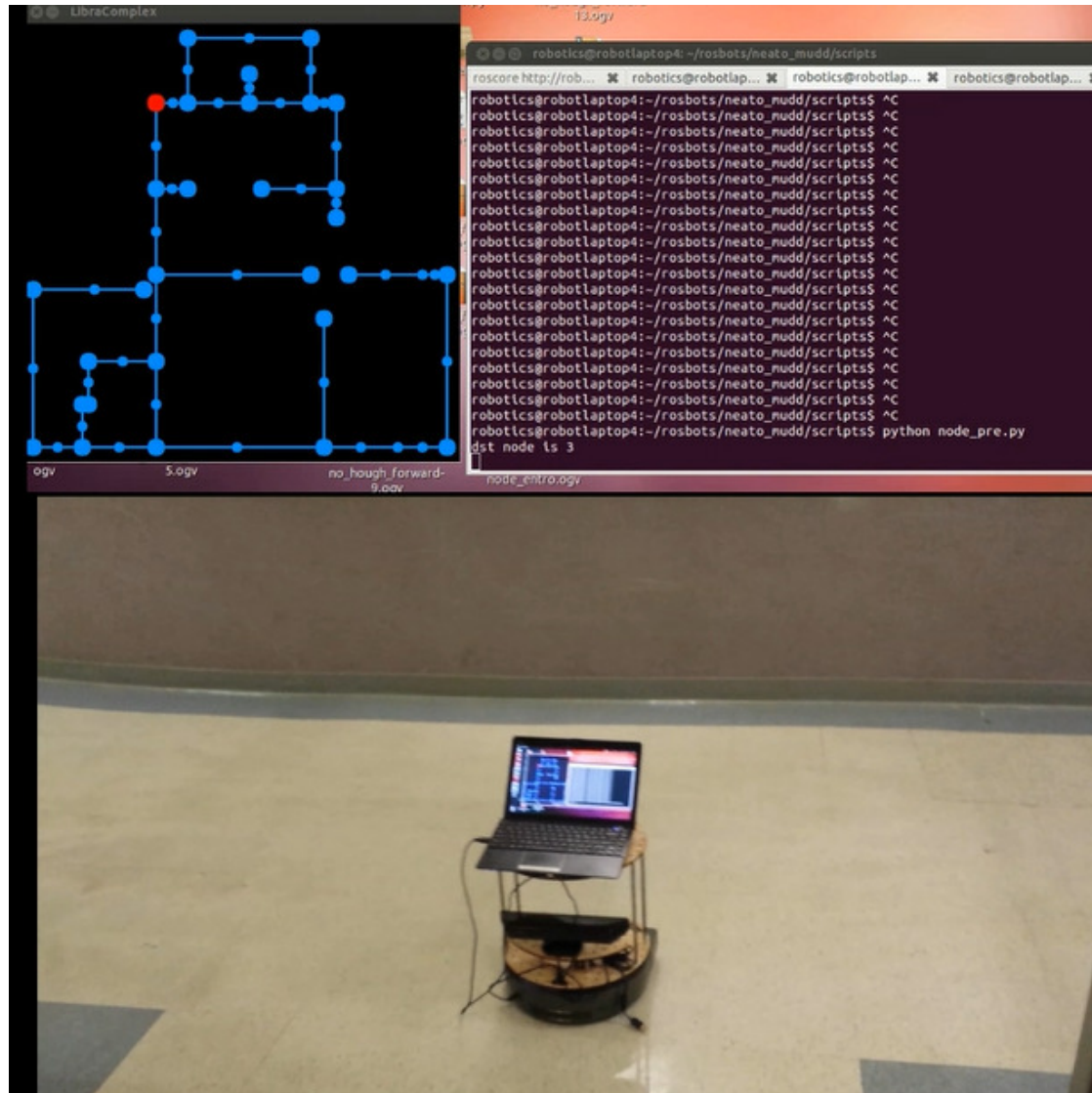
ogv 5.ogv no\_hough\_forward 9.ogv

00:02

ma  
kn

We could opt for the turn *most likely to make progress toward the goal.*

# Predictive path planning



pre\_short.mp4

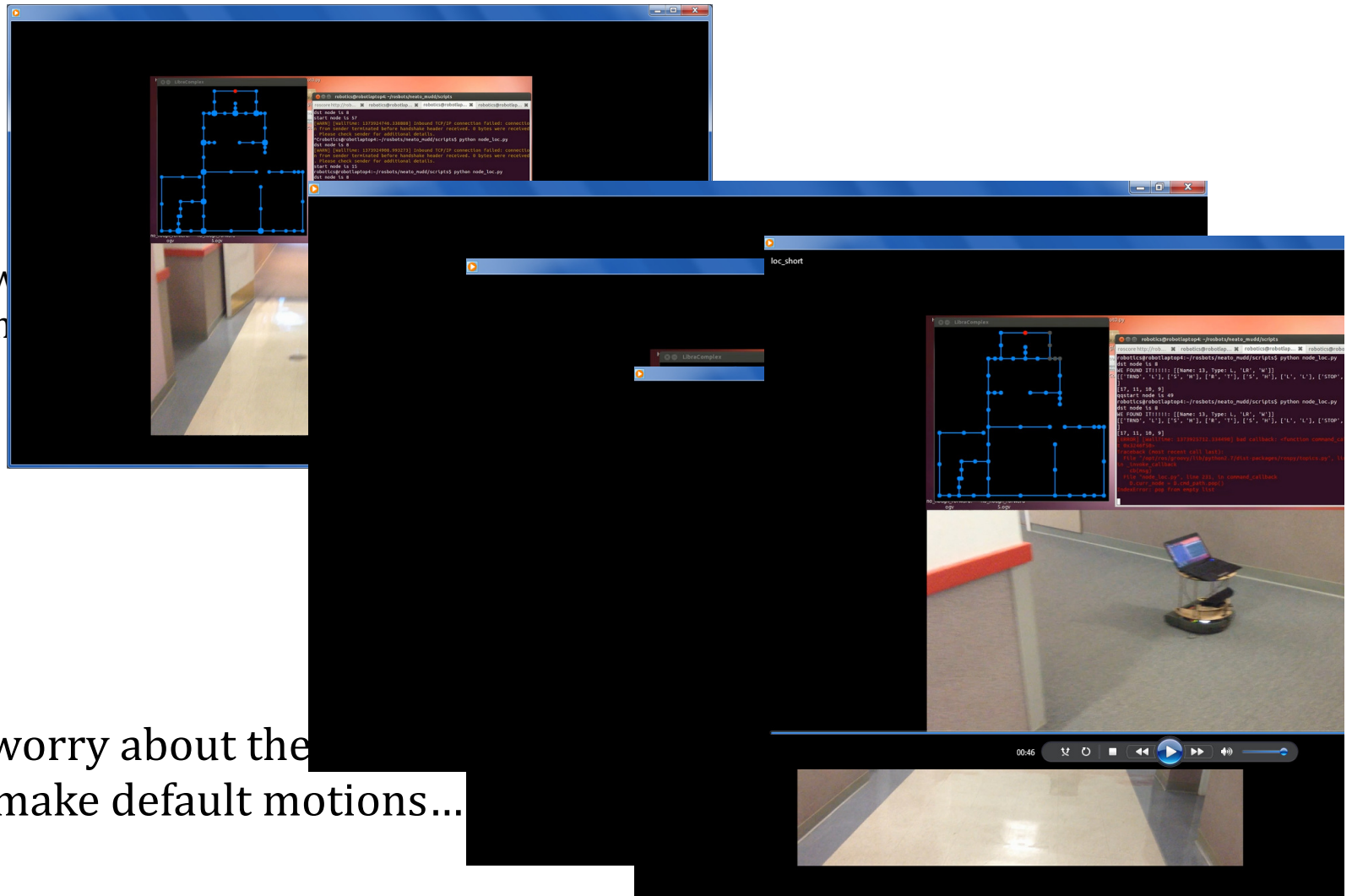
more robust navigation ~ *but all with a hand-built map*

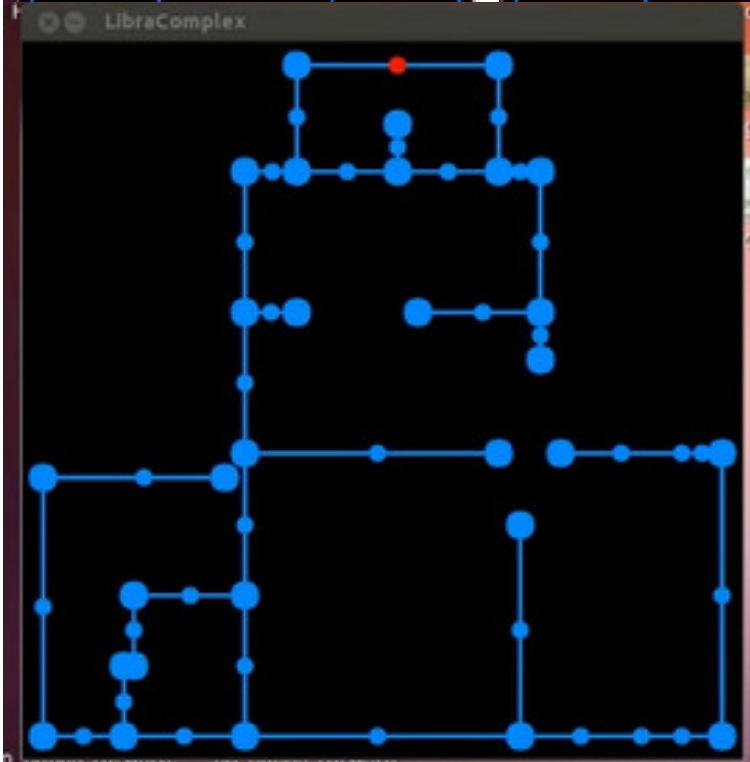
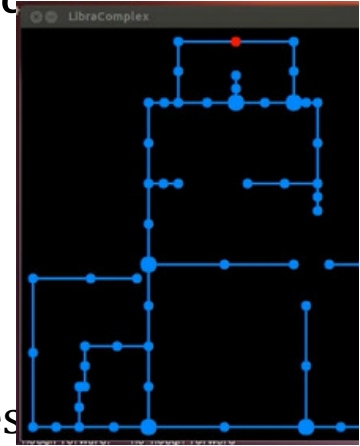
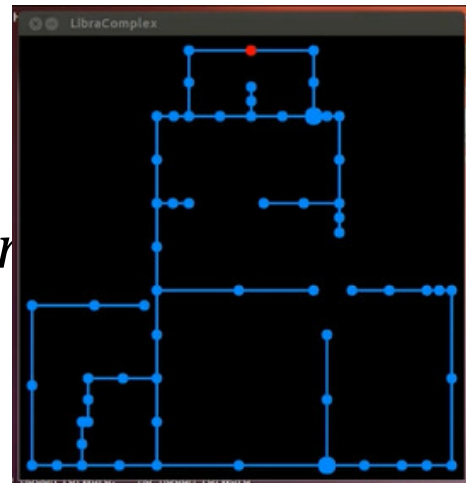
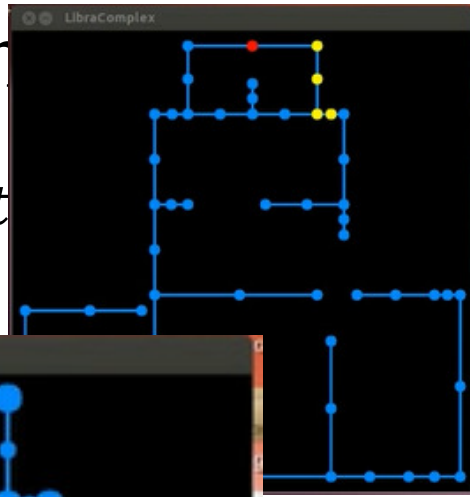
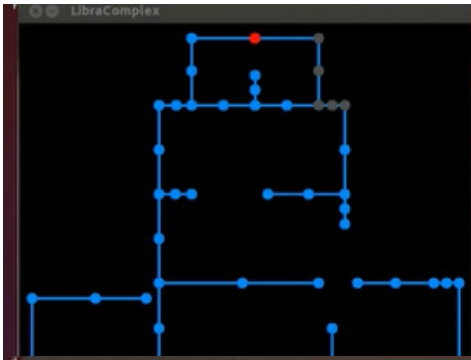
# Localization

*What if the robot doesn't know its starting pose?*

map w  
and th

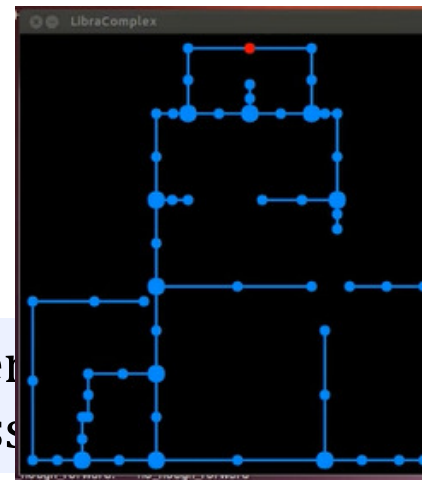
*Don't worry about the*  
Instead, make default motions...





map with fewer nodes  
large and the goal node in  
red

~ sees a TL ~  
a T intersection  
open to the left



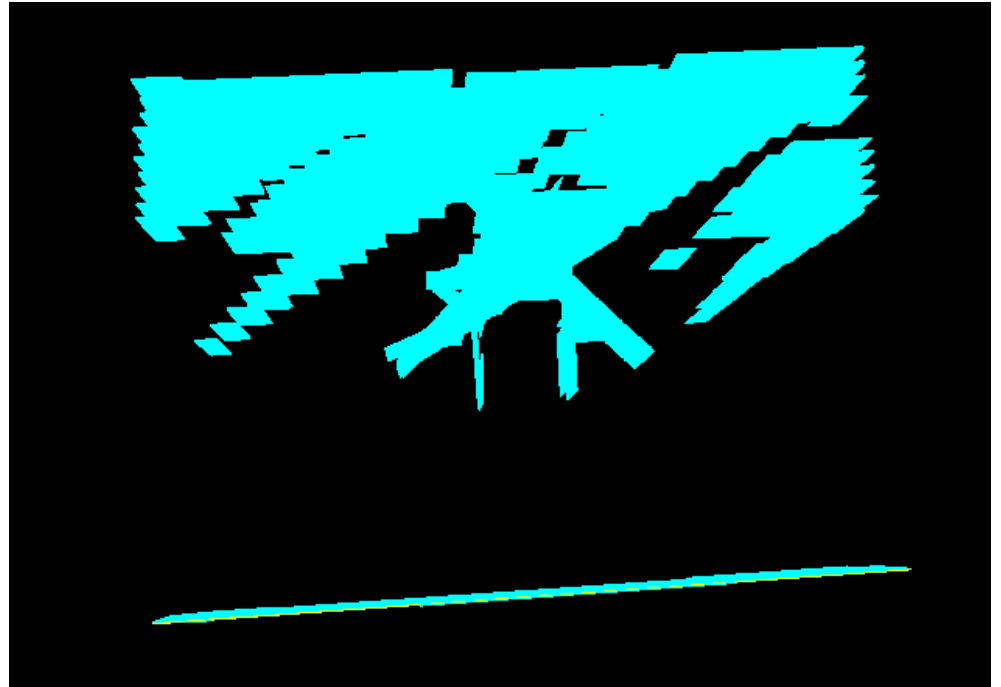
*Don't* worry about the goal.  
Instead, make default motions...

With each obser  
cull the robot's poss

# Too many walls...



Uninitialized images ~  
aargh!



More of a "*point fog*"  
than a point cloud