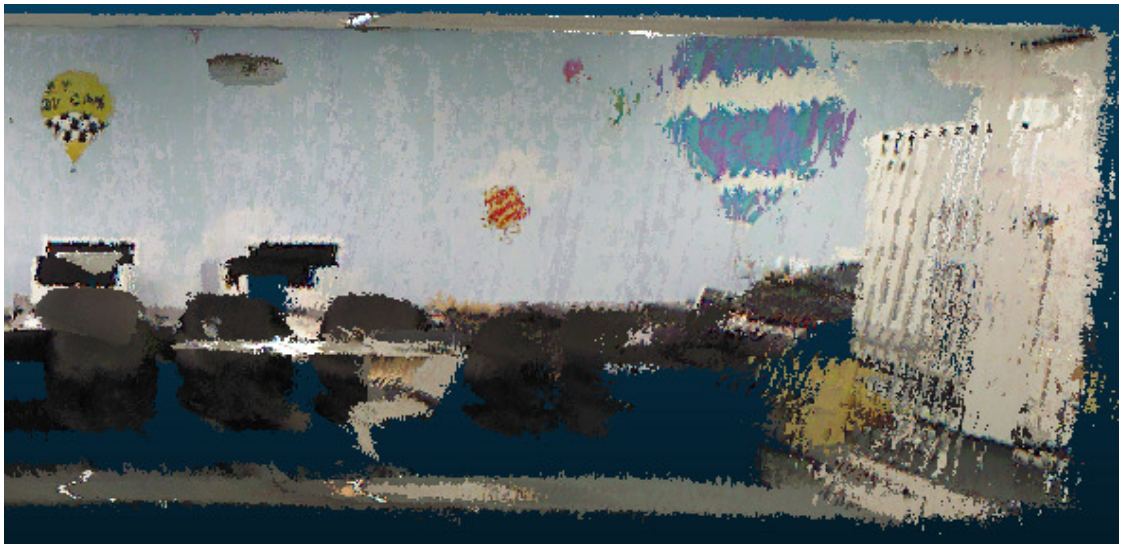
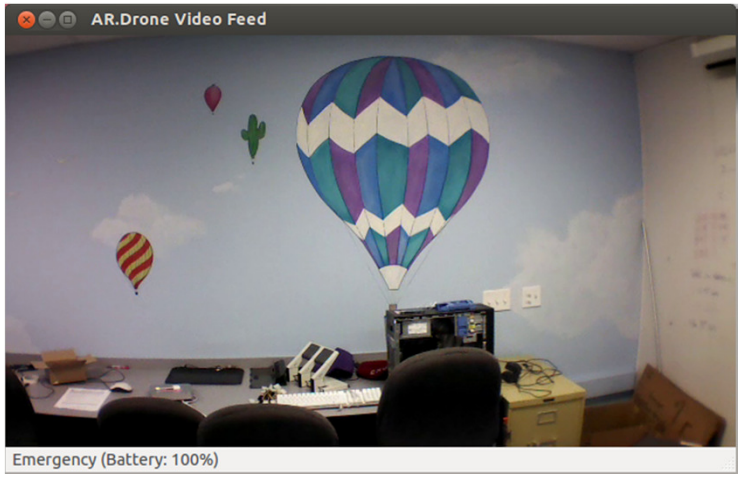
Interdimensional Robot Cooperation

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?!

2d sensors

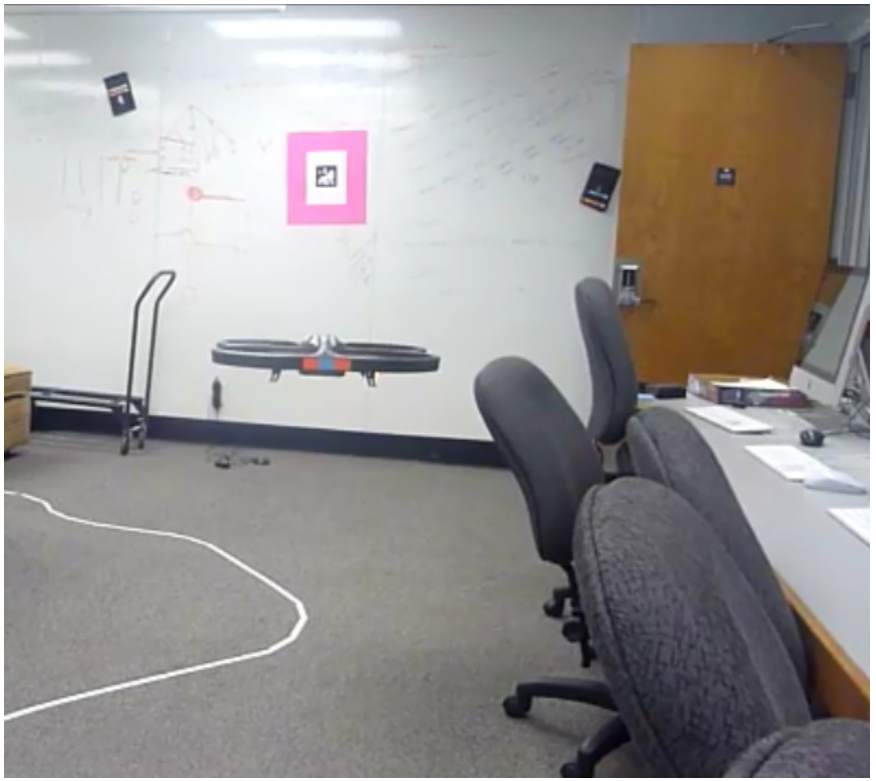
3d models

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

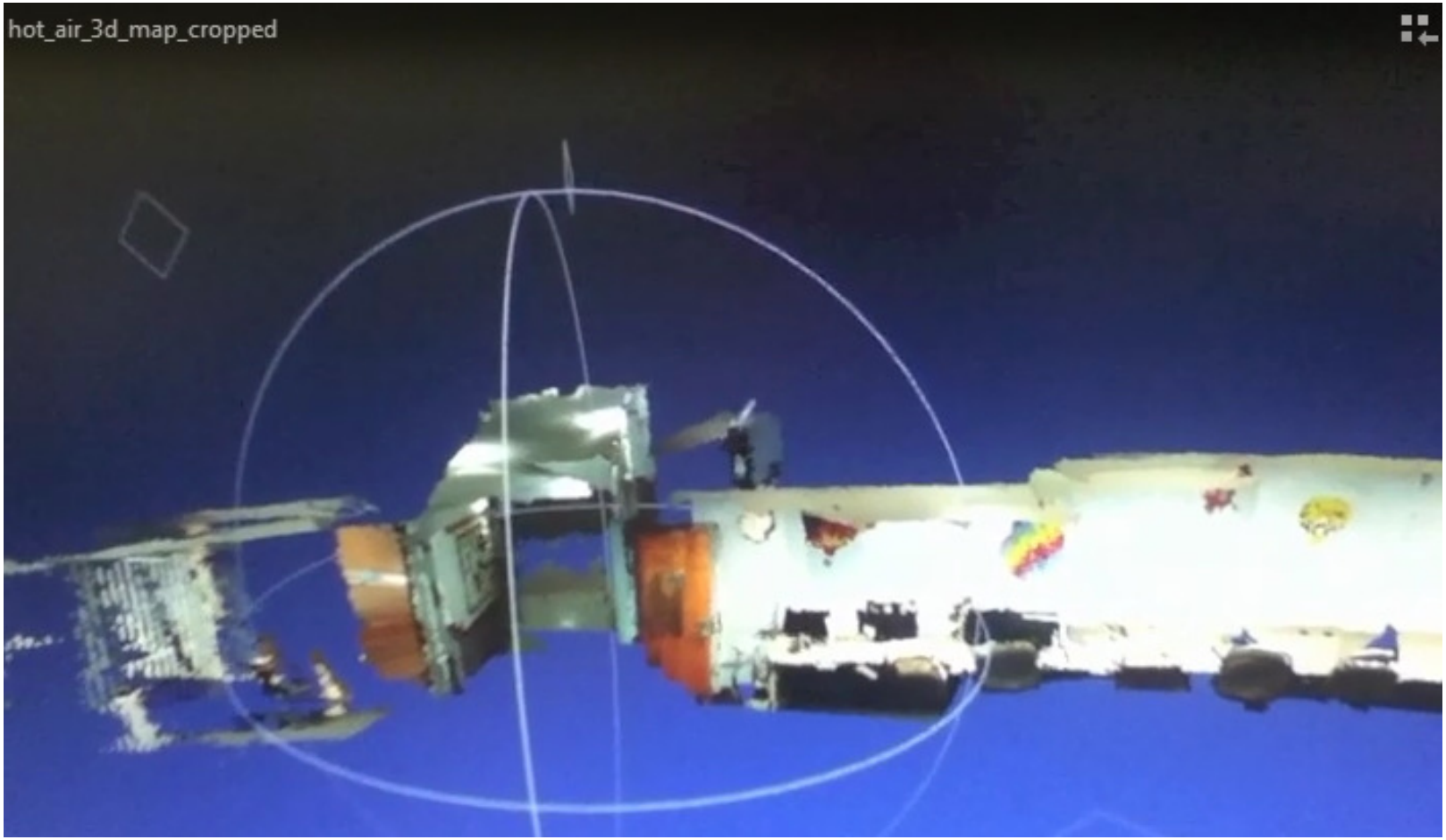
Kristina + Chris

Q: *Why 3d?* A: The Kinect.

Giant shoulders: *RGBD-SLAM*

SLAM ~ Simultaneous Localization And Mapping

Hot-air lab map



Example of a hot-air-lab-and-hallway map created by RGBD-SLAM

Our task: *Escape!*

Localization-based loop:

(given a known goal)

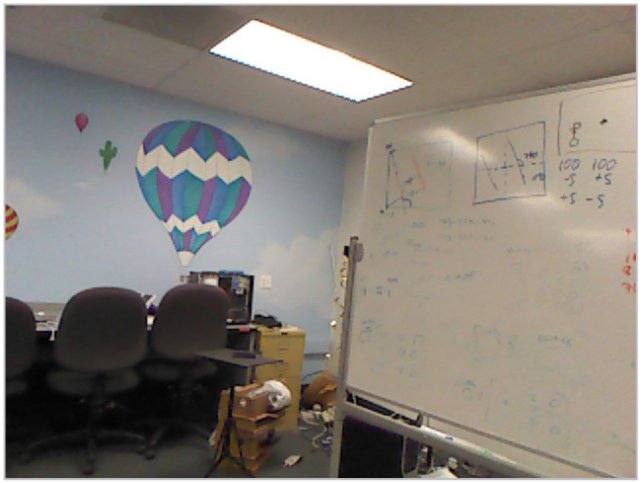
(1) the drone grabs a 2d image

(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



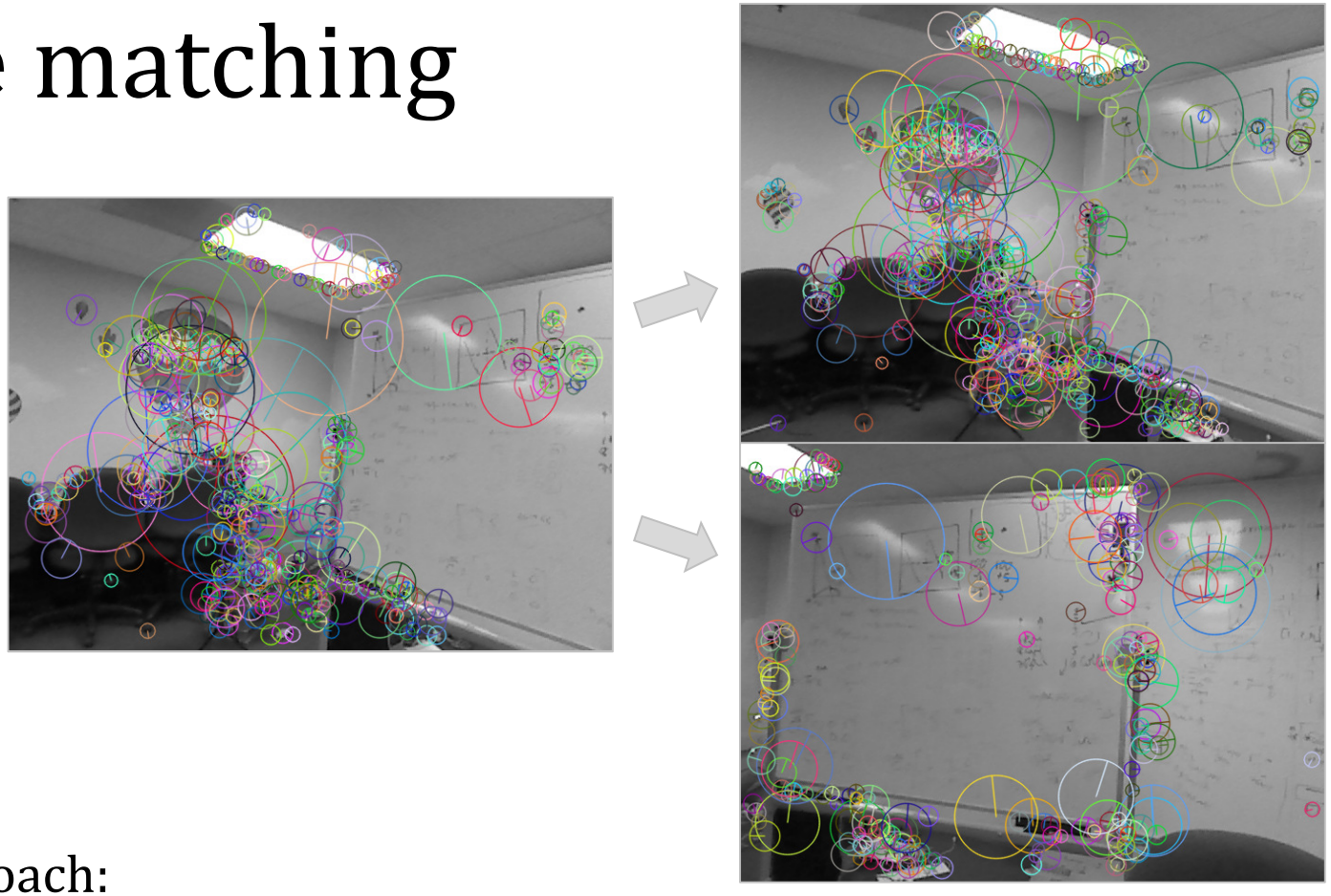
new image (live)

many original images (stored)

Our approach:

- 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



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:

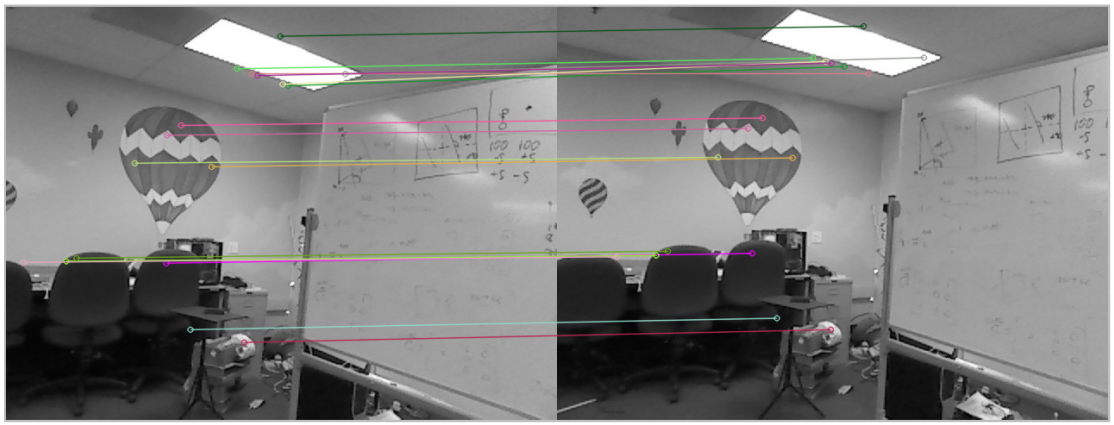
- Find features in each (SIFT)
- Consider all matches (FLANN)
- **Ensure bidirectional constraints**
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

Keep only *two-way* best matches

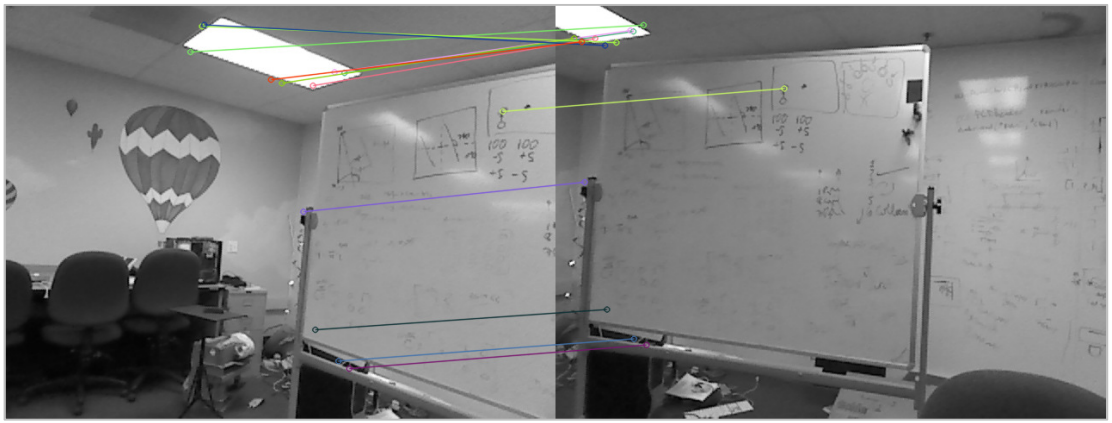
Our approach:

- 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



1st-best match

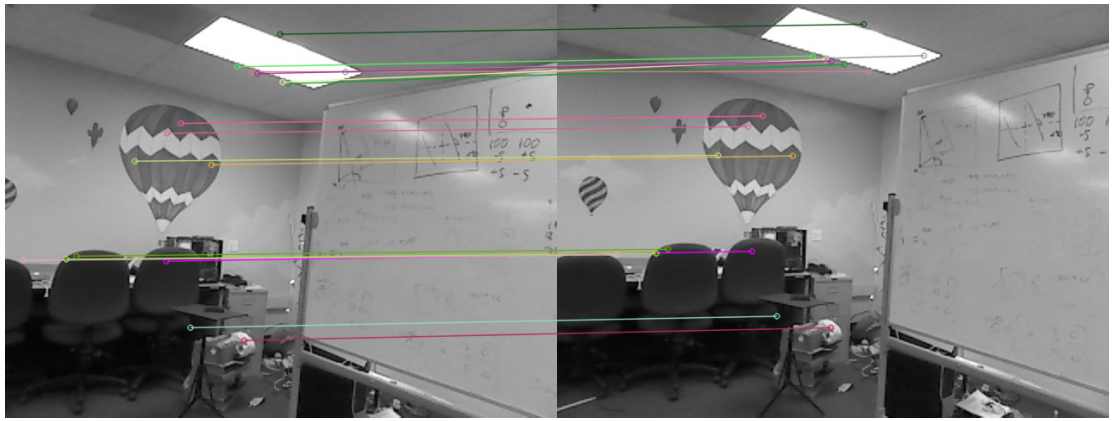


10th-best match

Our approach:

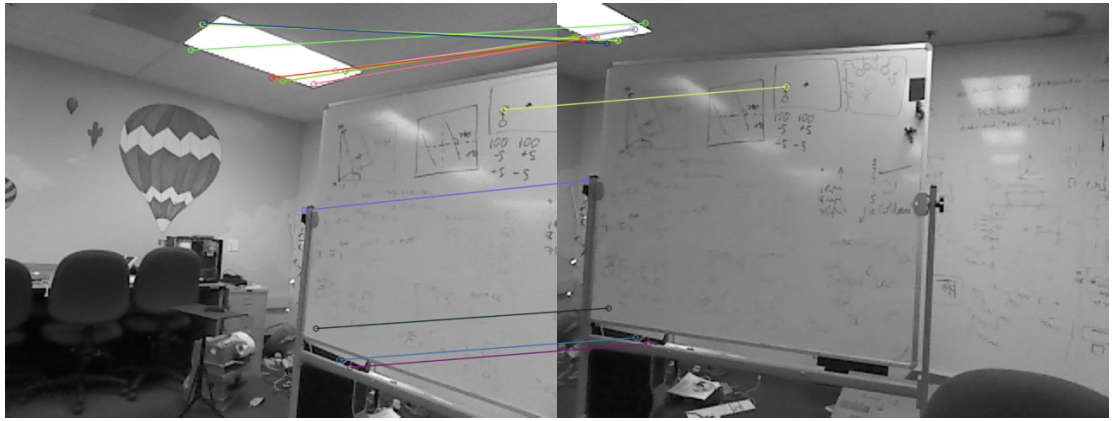
- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (via homography)
- **Best-matching by visual similarity metric**
- Tiebreaking by pixel distances

Visual “closeness”



1st-best match

Average feature distance **~ 40 px**



10th-best match

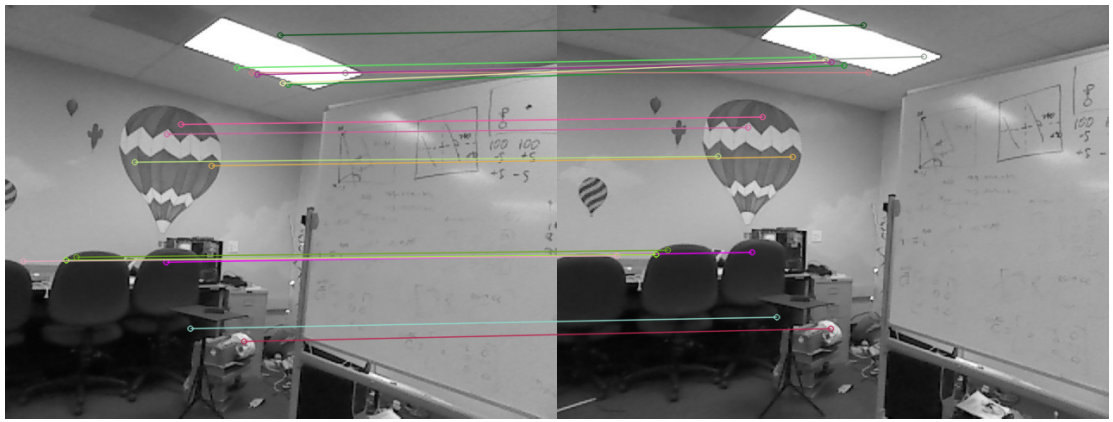
Average feature distance **~ 270 px**

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**

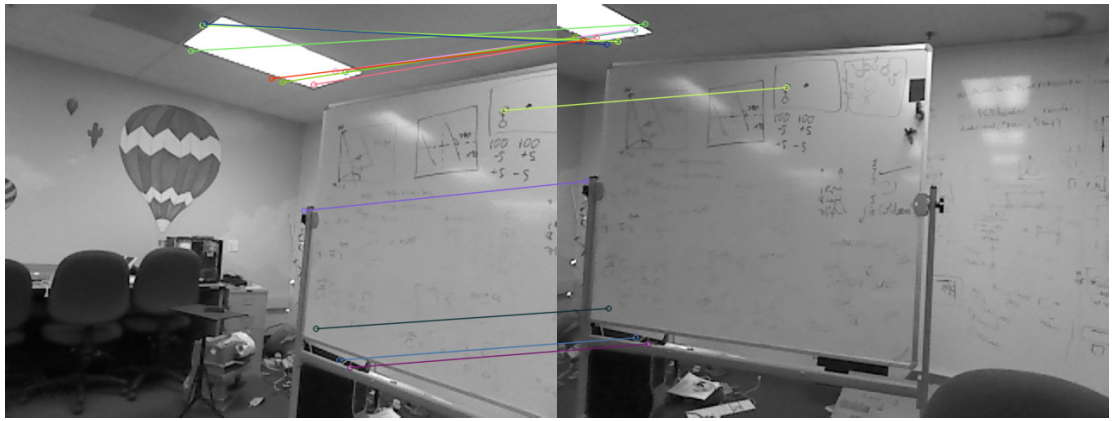
Pixel "closeness"

Image matching



new image (live)

many original images (stored)

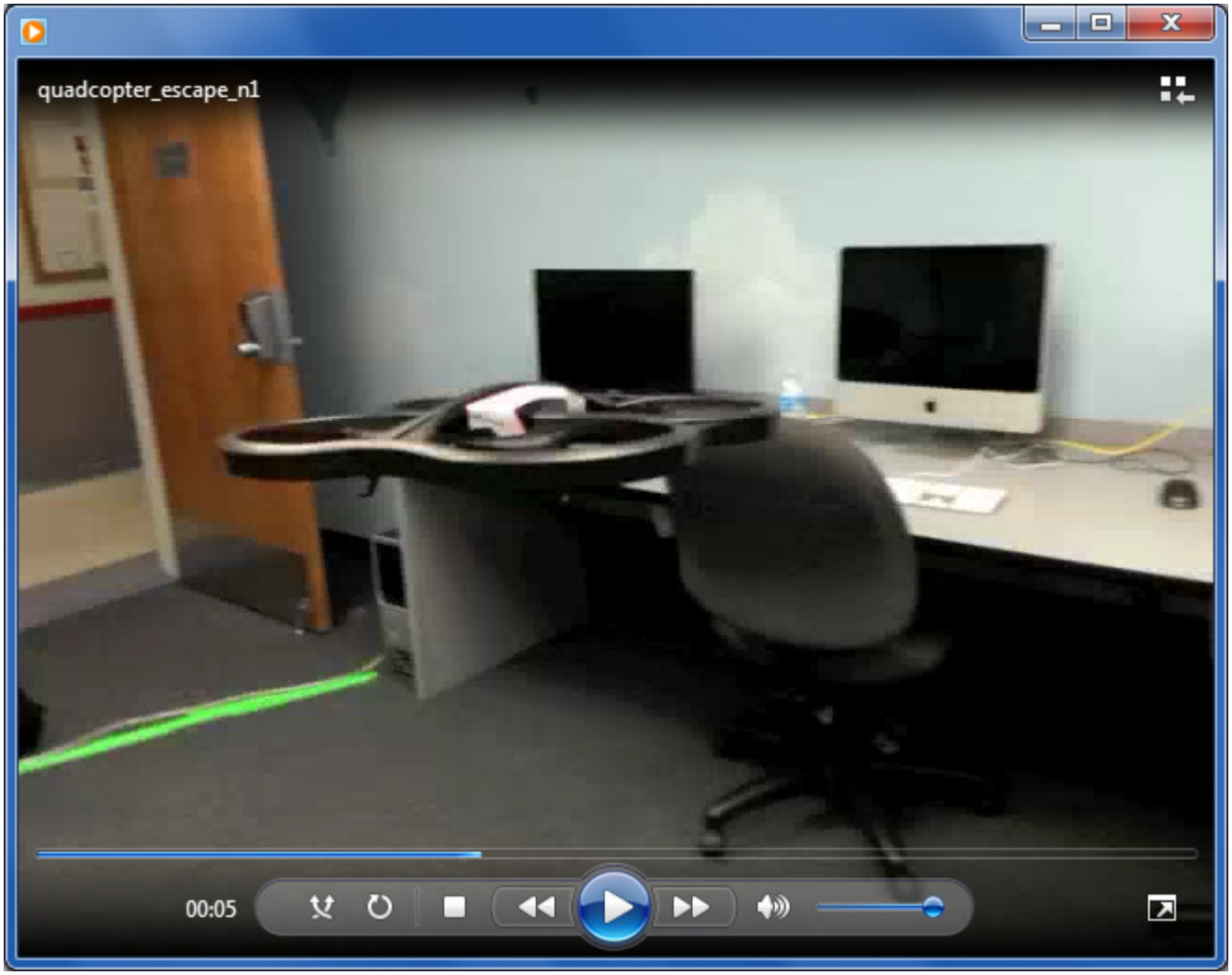


1st-best match

10th-best match

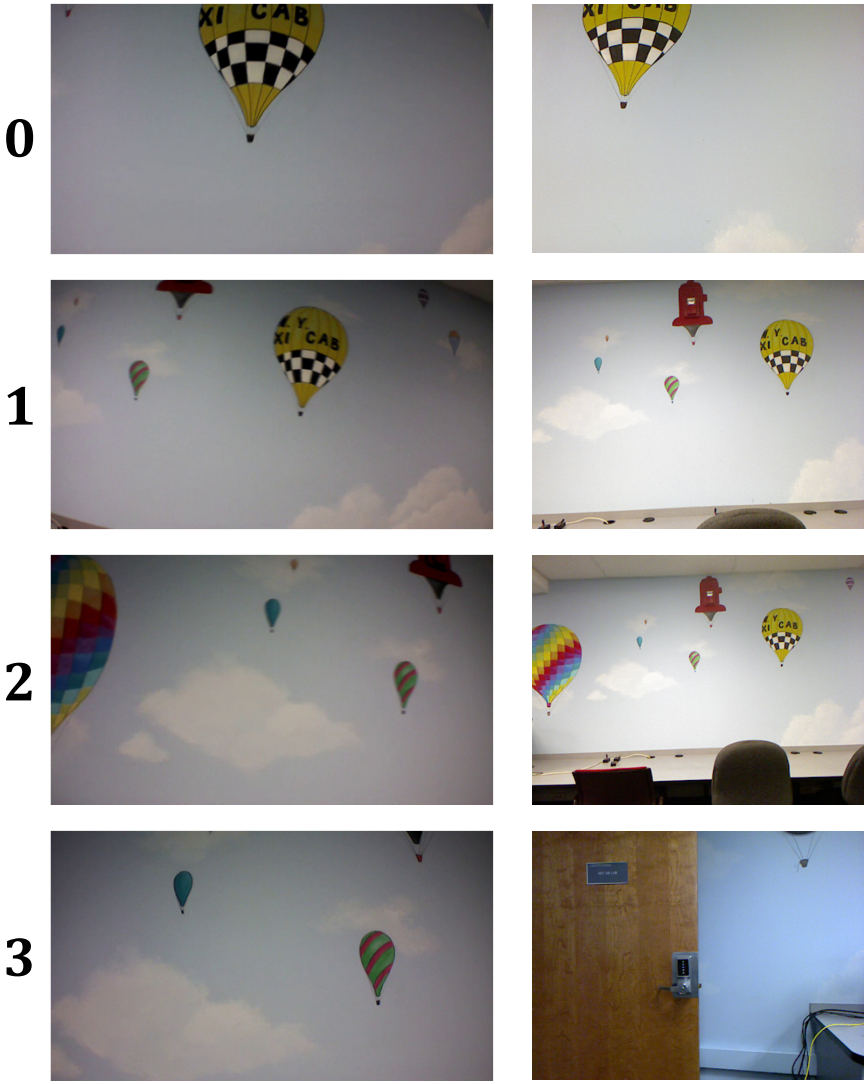
for now, a linear search taking **~6 seconds** for the hot-air map's 42 images

Results

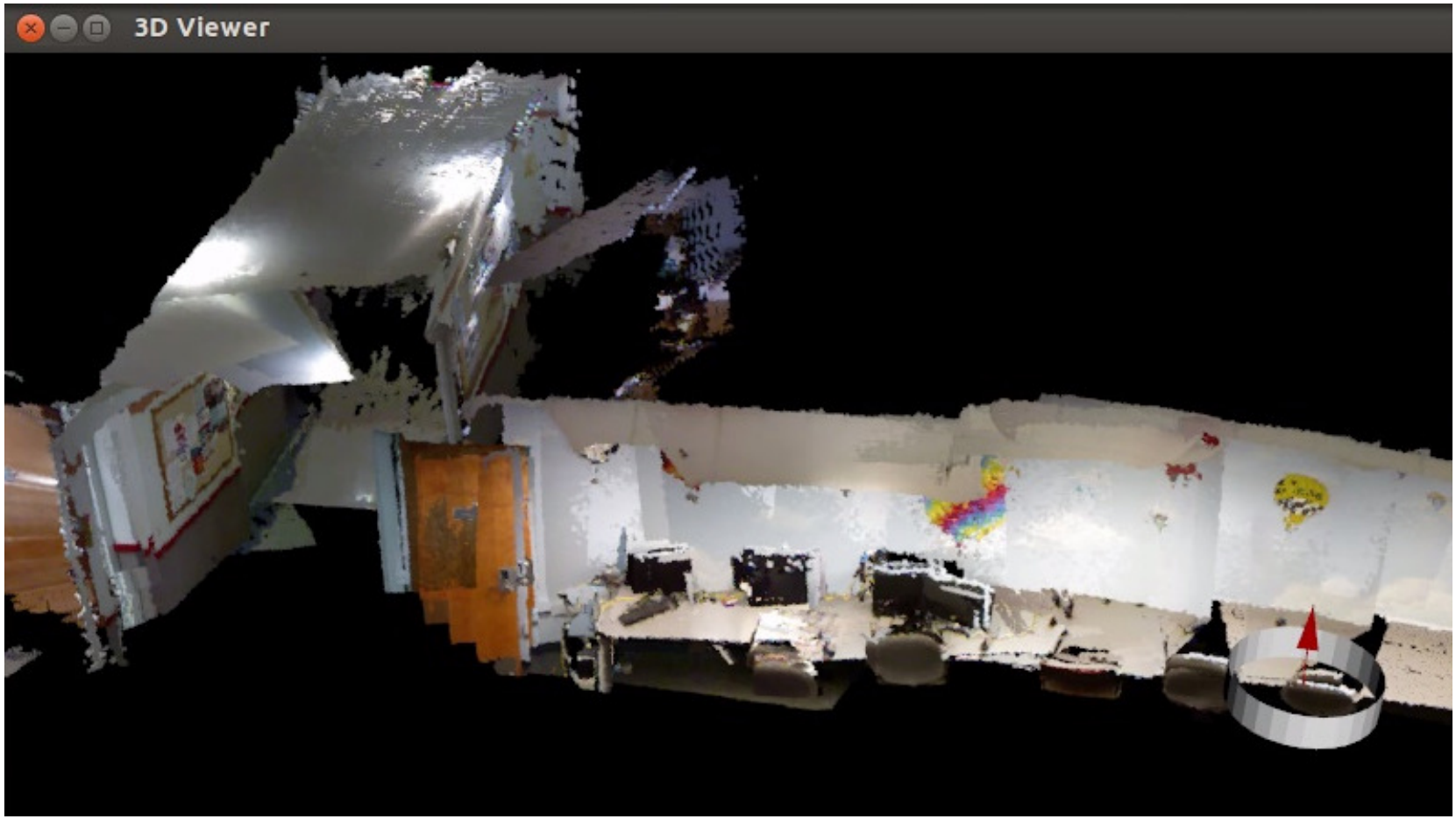


One of the drone's escapes ~ 150 sec.

The matches


0
1
2
3
live, novel images
from drone
best-matching
images from map
4
5
6
7
8

Model



Map-based localization of those 9 estimated positions

Jerry, Cyrus, Zakkai

New platform: *Neato*

Initial tasks

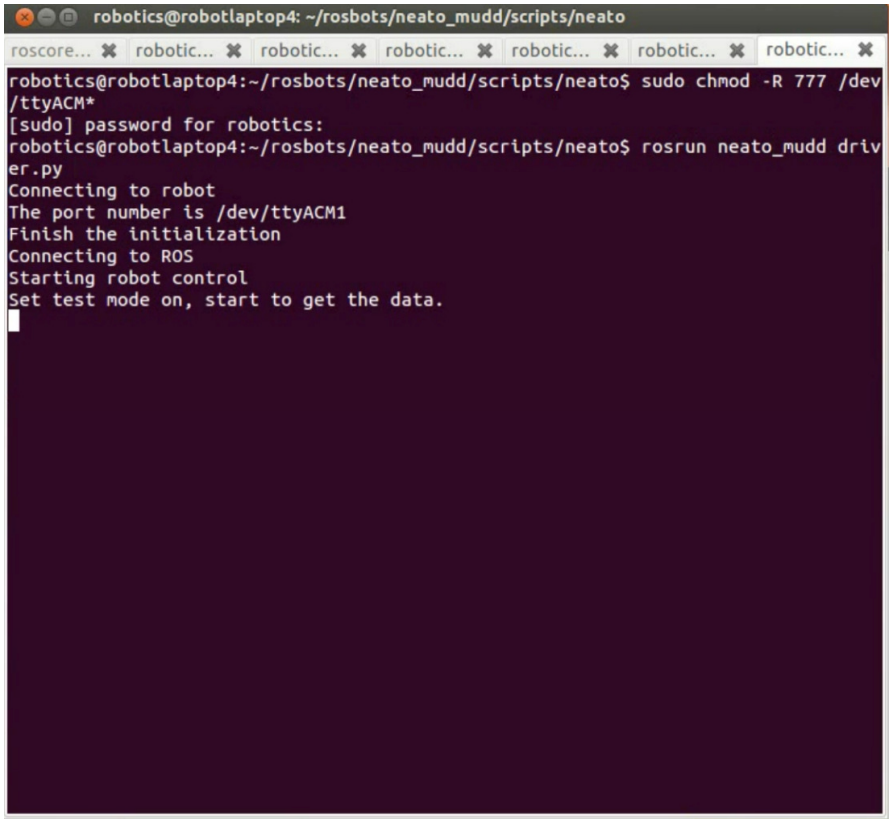
- **learn ASCII API**
- **write device drivers**
- interpret sensor data
- follow corridors
- detect intersections

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

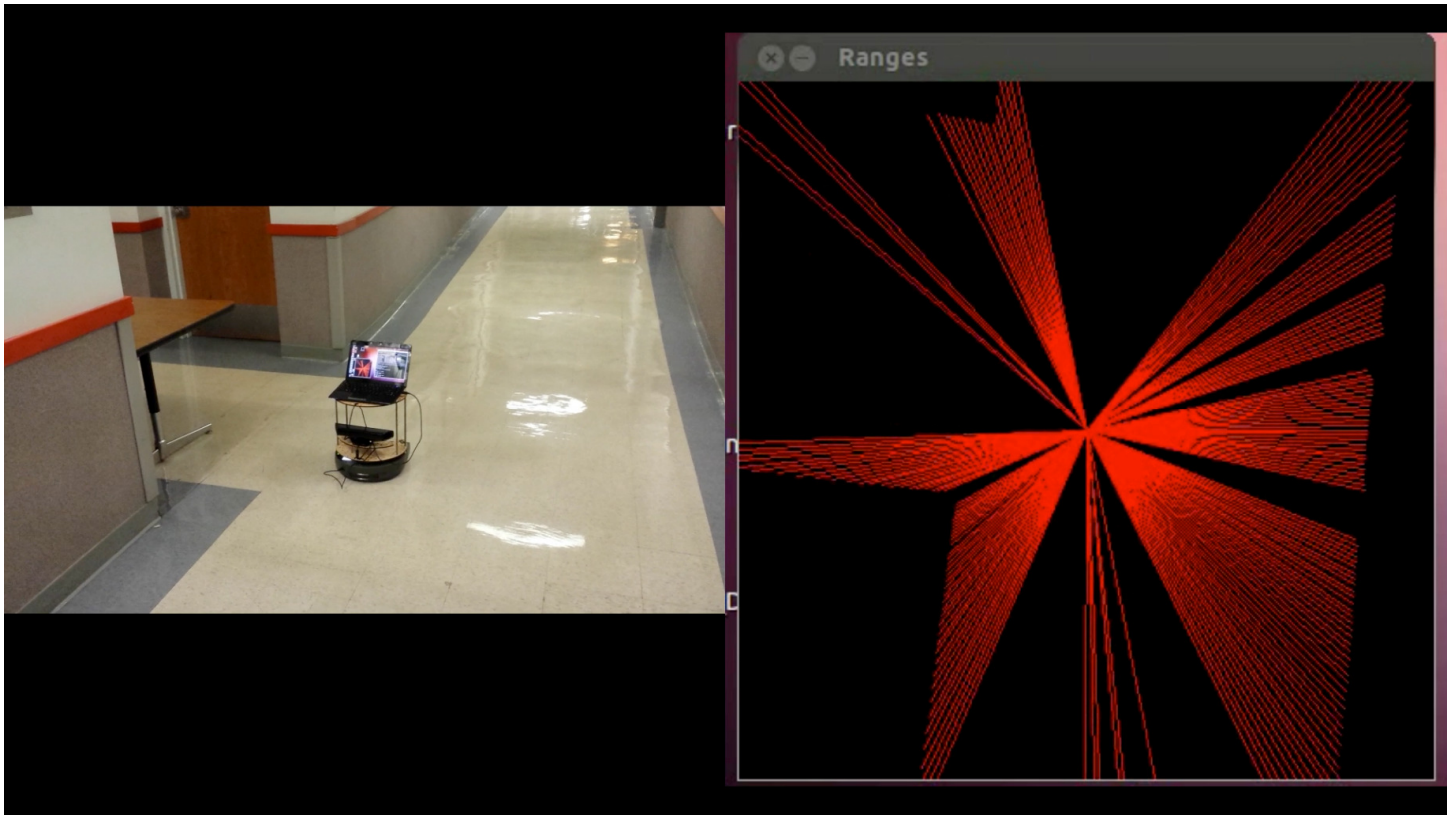
AI Tasks

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

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

Visualizing

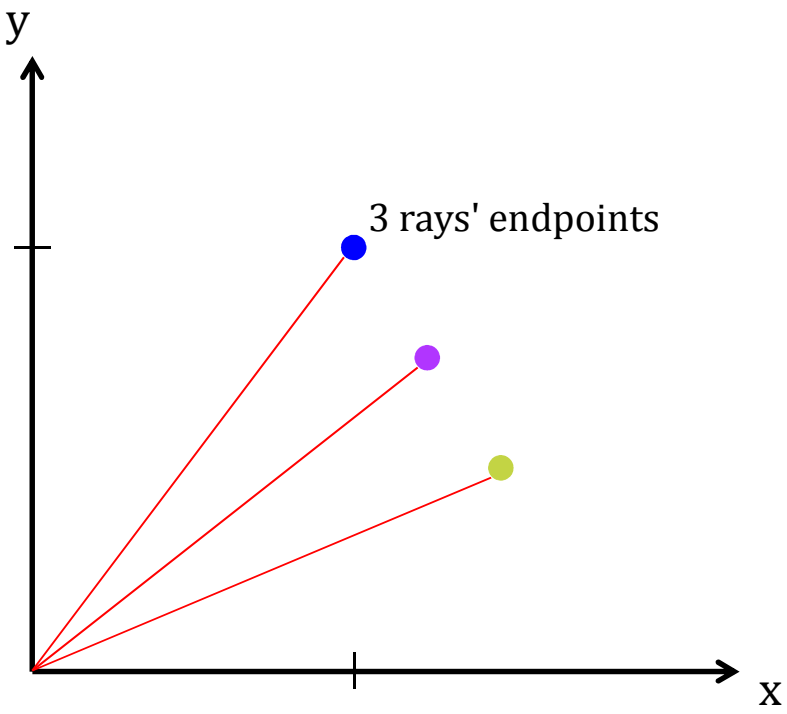
- OpenCV interface
- beautiful data!
- *how to navigate smoothly?*



Objective: Find walls and navigate safely among them

Hough transform

Point space



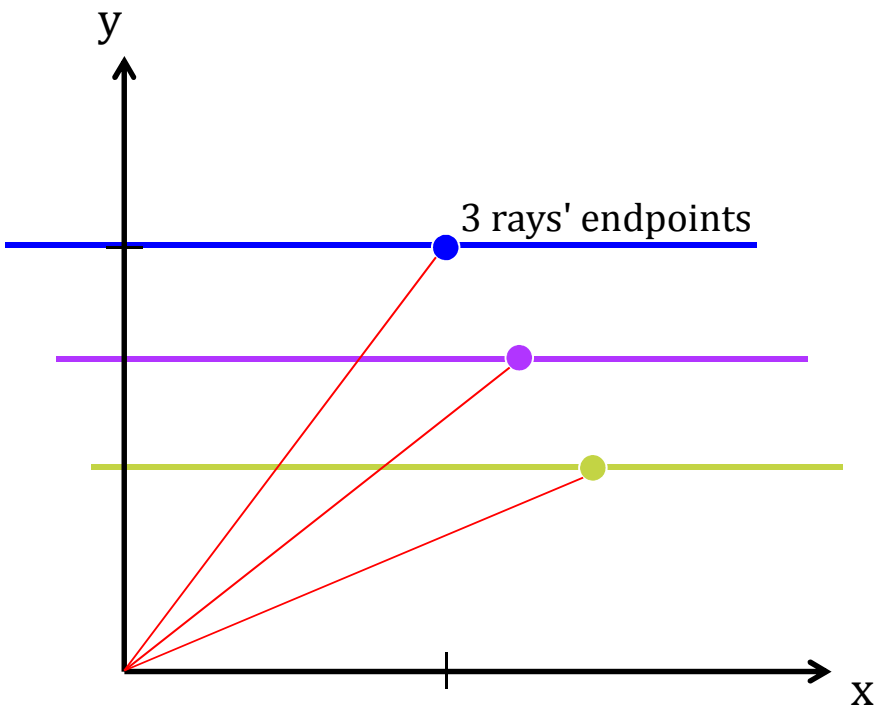
Line space

yield lines with substantial support from the original data (in green)

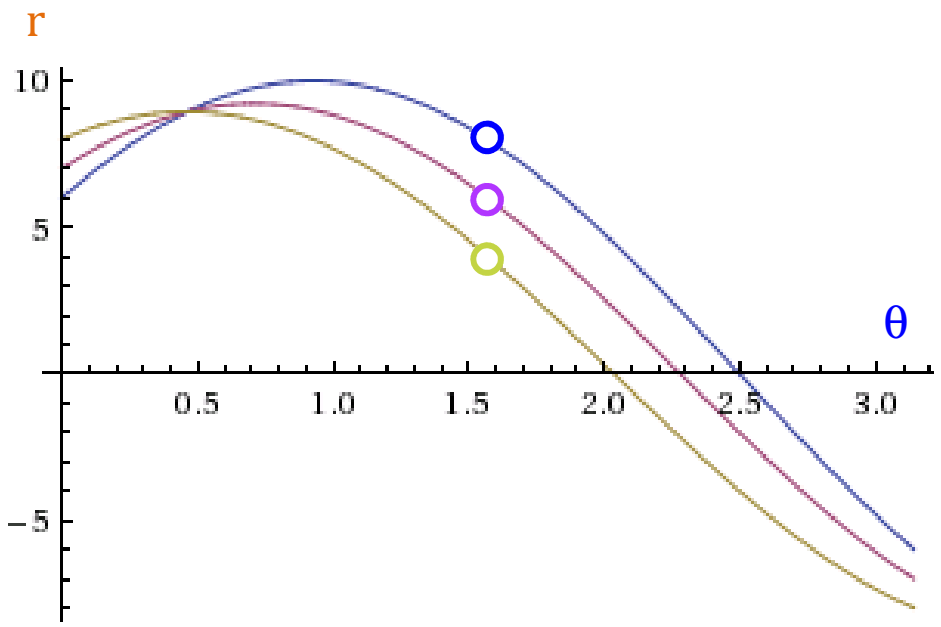
peaks here in line space

Hough transform

Point space



Line space



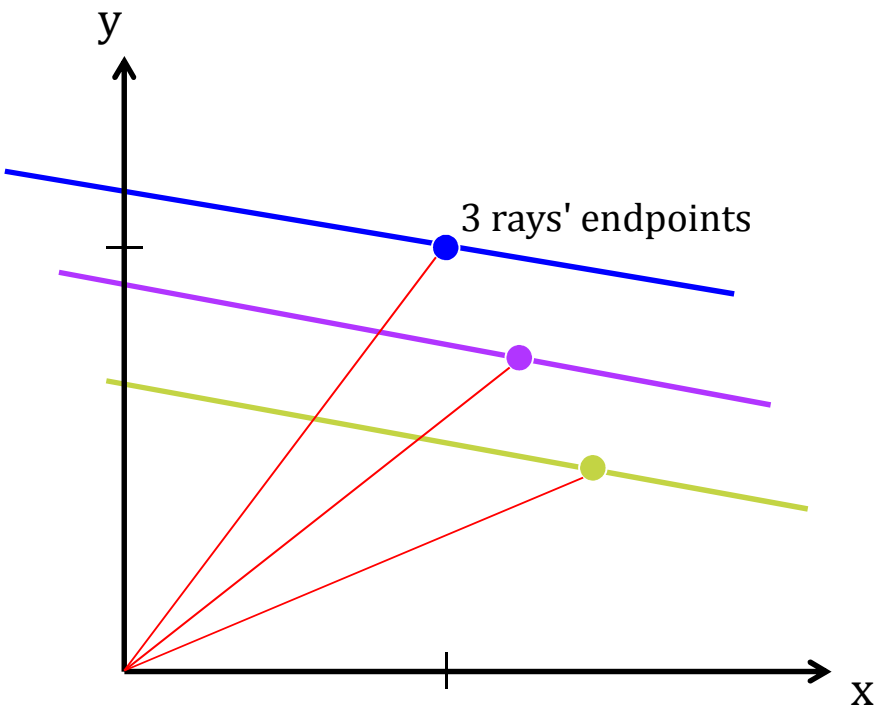
yield lines with substantial support from the original data (in green)

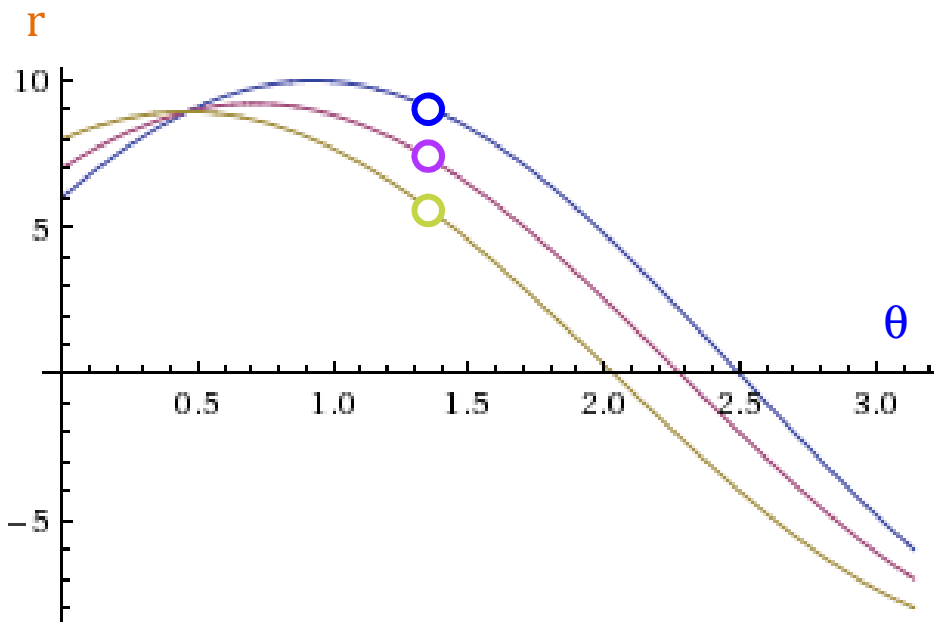
peaks here in line space

Hough transform

Point space

Line space



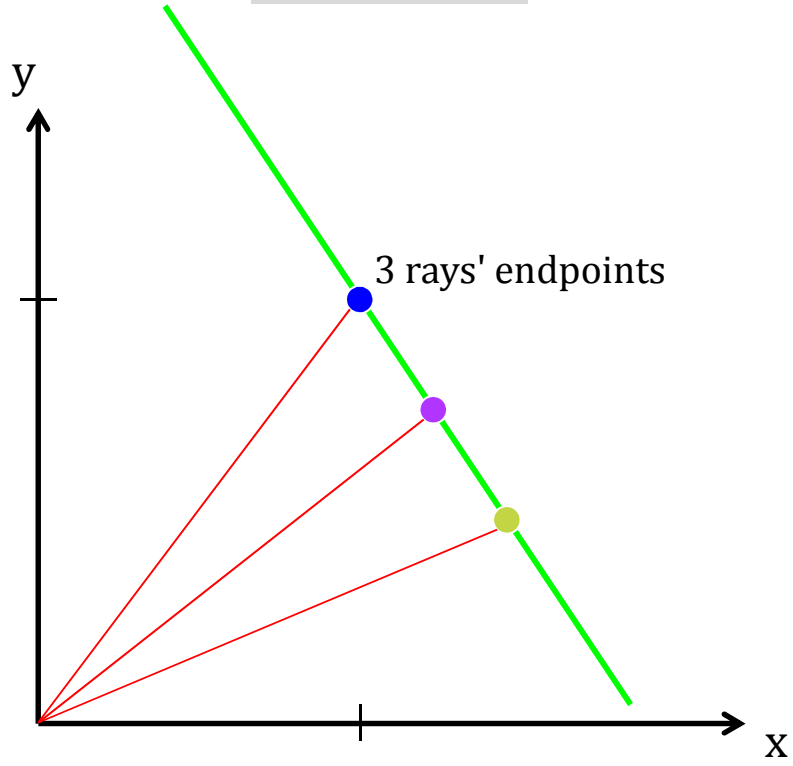


yield lines with substantial support from the original data (in green)

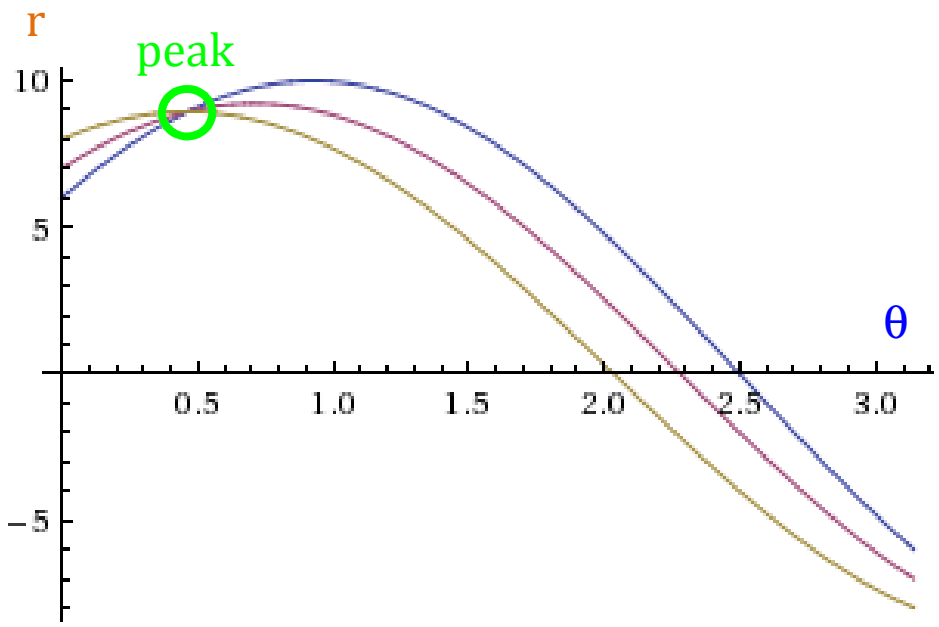
peaks here in line space

Hough transform

Point space



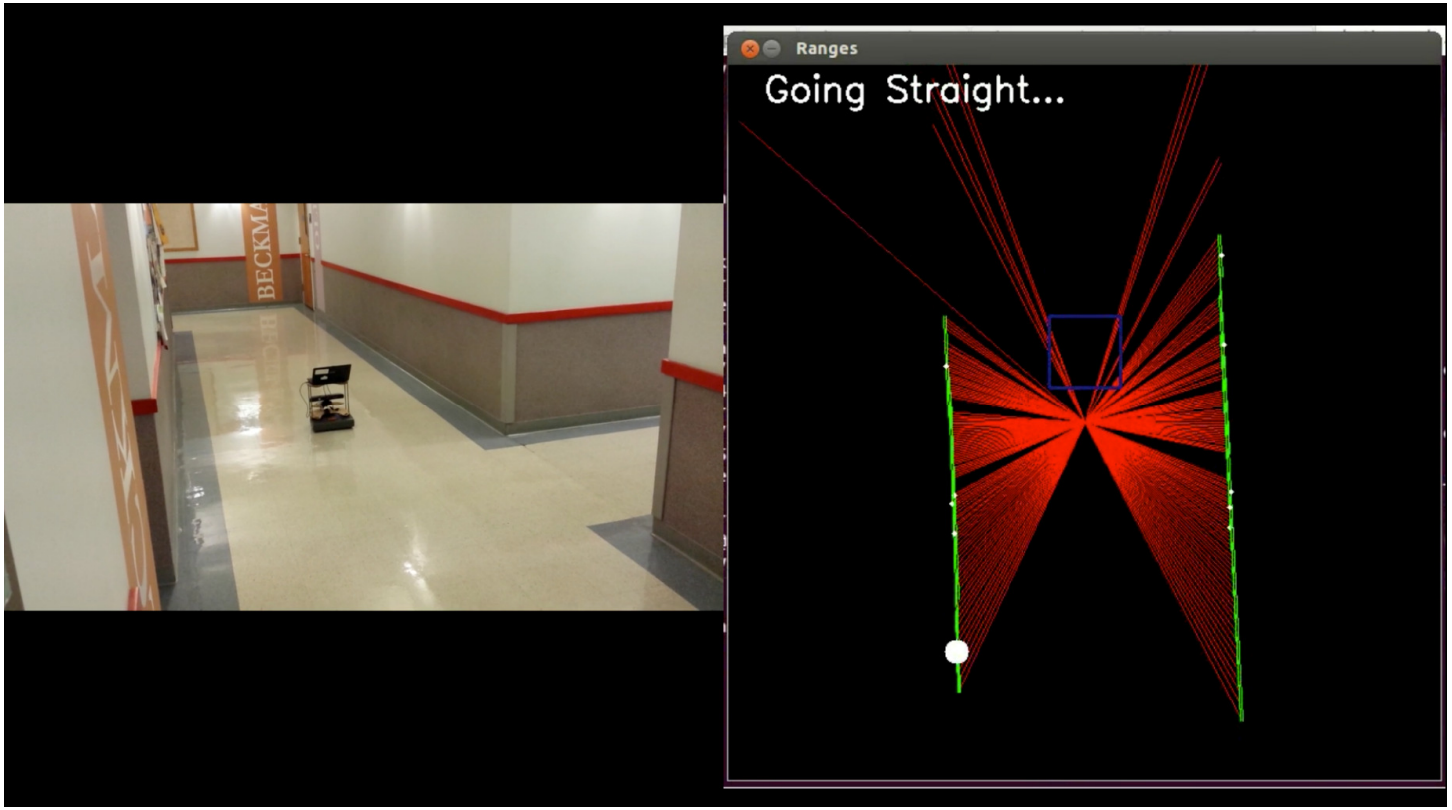
Line space



yield lines with substantial support from the original data (in green)

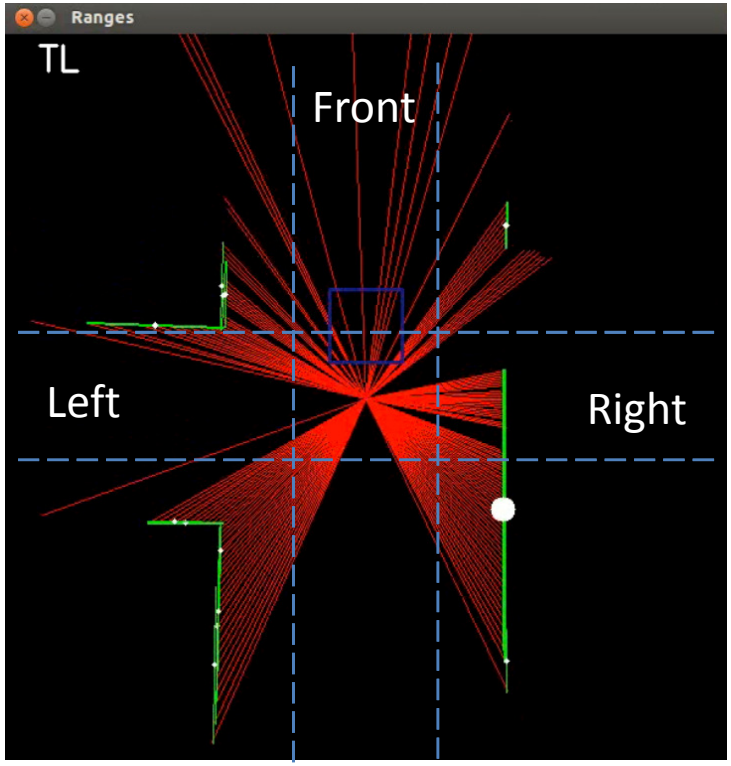
peaks here in line space

Hough-based navigation



Idea: follow the wall *most parallel* to the current heading

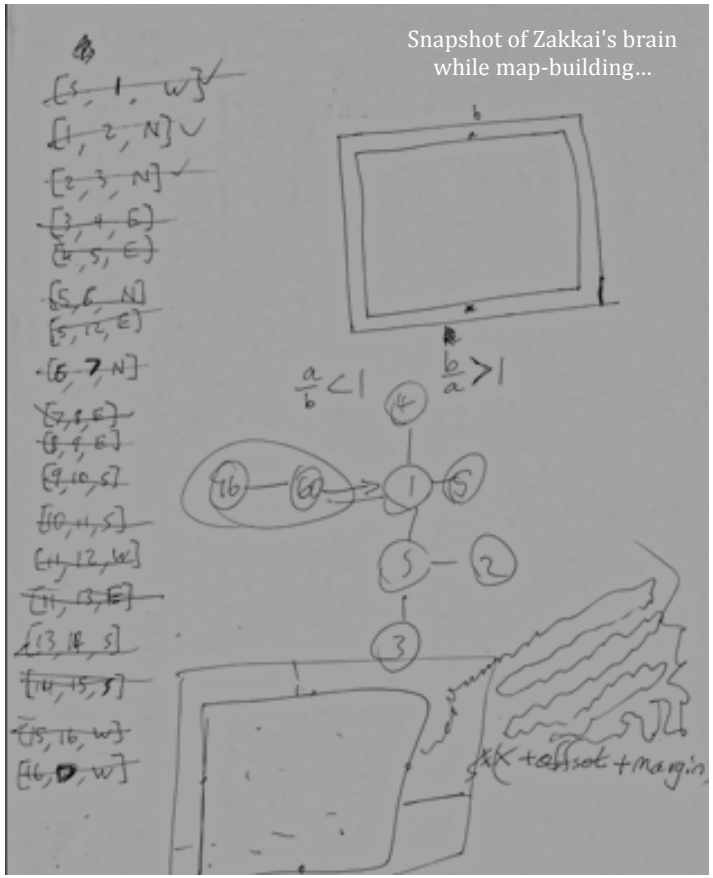
Intersection recognition



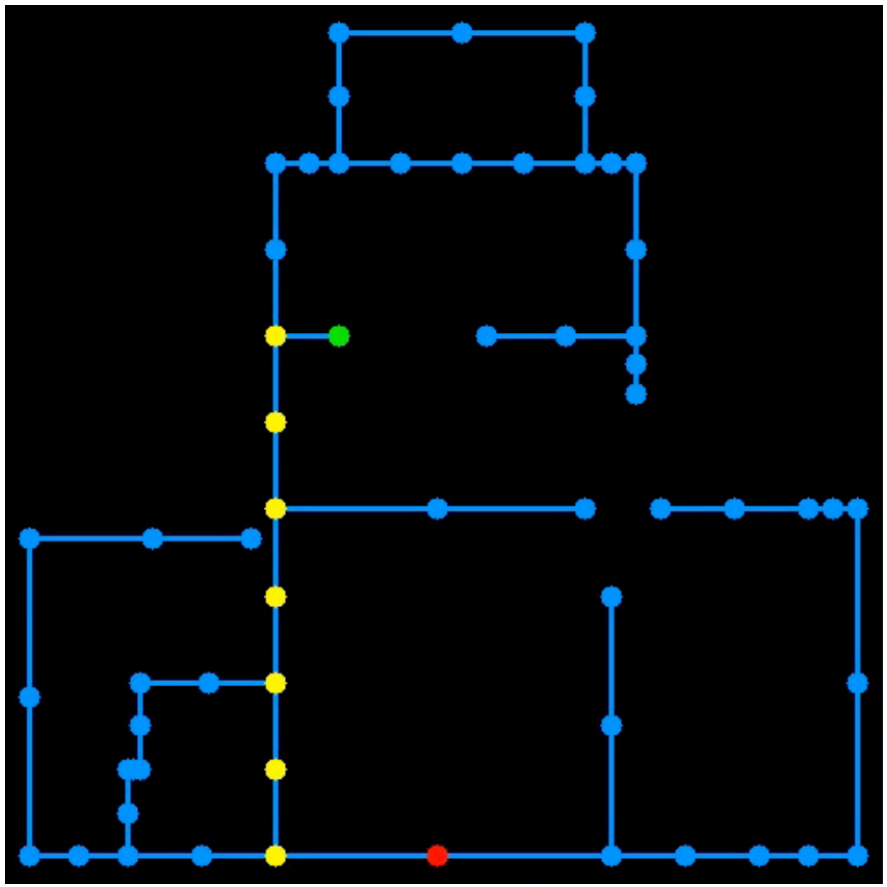
For each scan:

- Check **Front**, **Left**, and **Right**
- Intersection type shows the # of obstacles: 3~**D** 2~**L** 1~**T** 0~**+**
- E.g., **TL** ~ **T**, ***open to the left***

Map-building

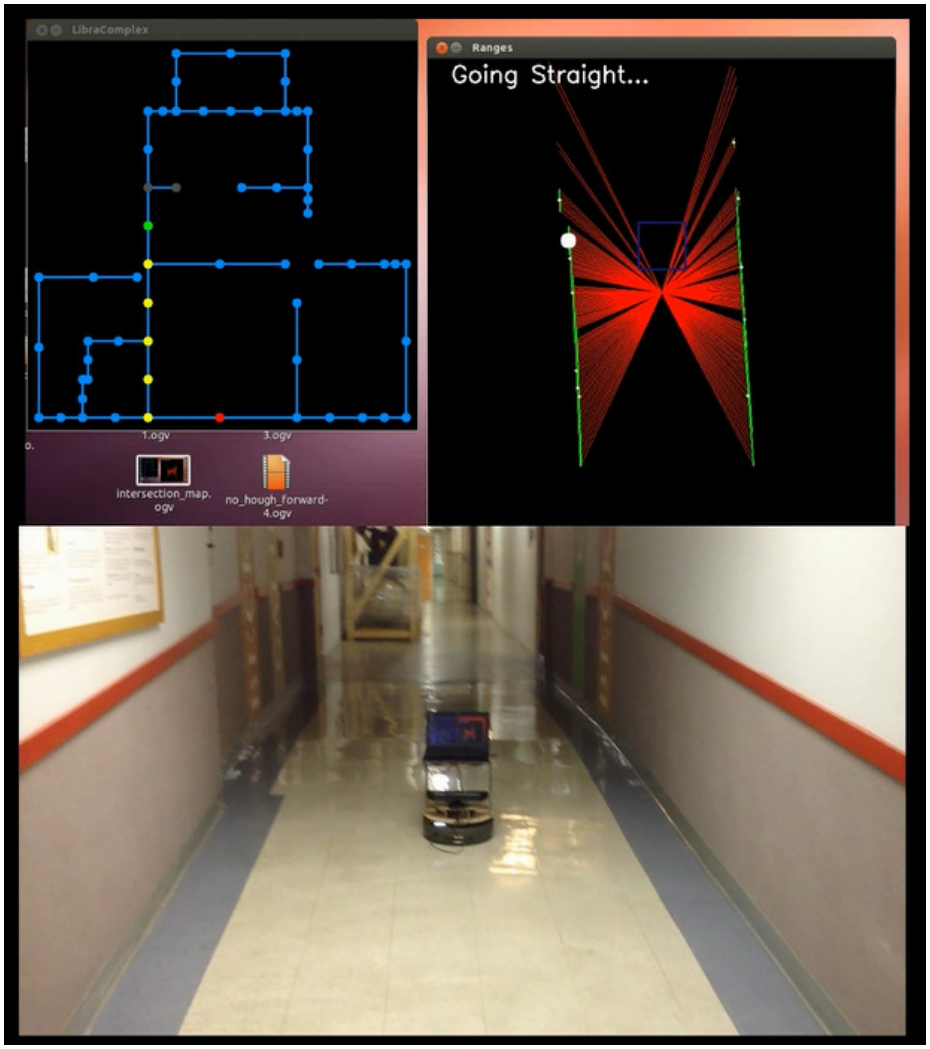


hand-built topological map



Intersections and hallways make up
the nodes and edges of the map

Navigation

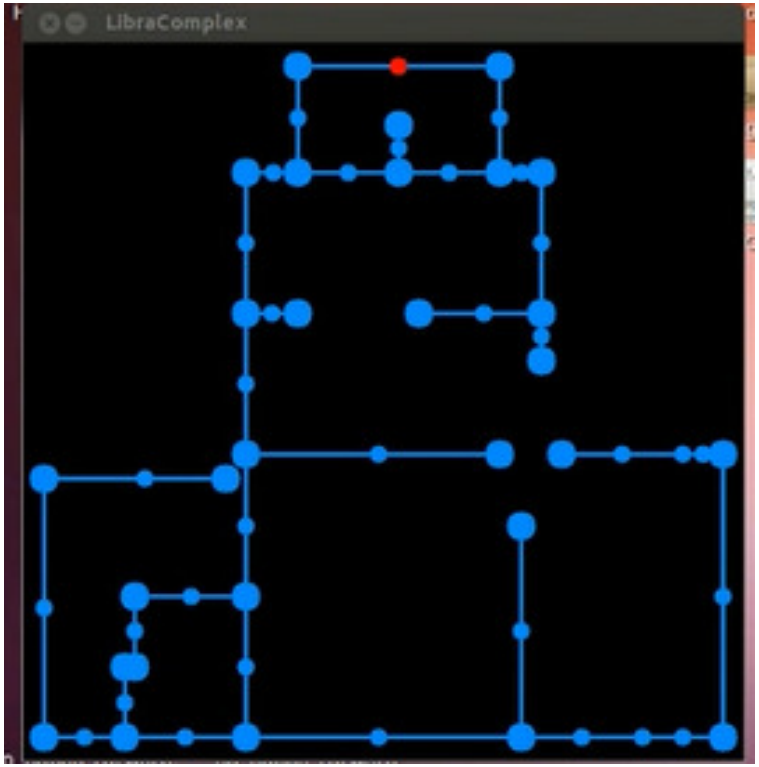


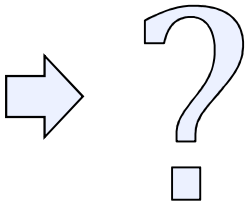
robot_navigation_known_start

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

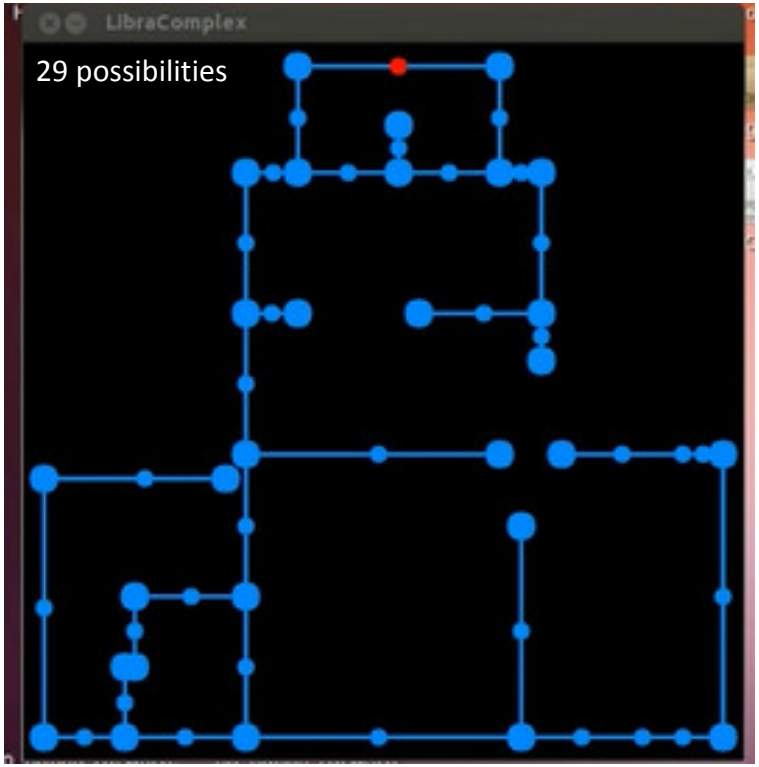
a "TL"

Which nodes remain?
Which are culled?

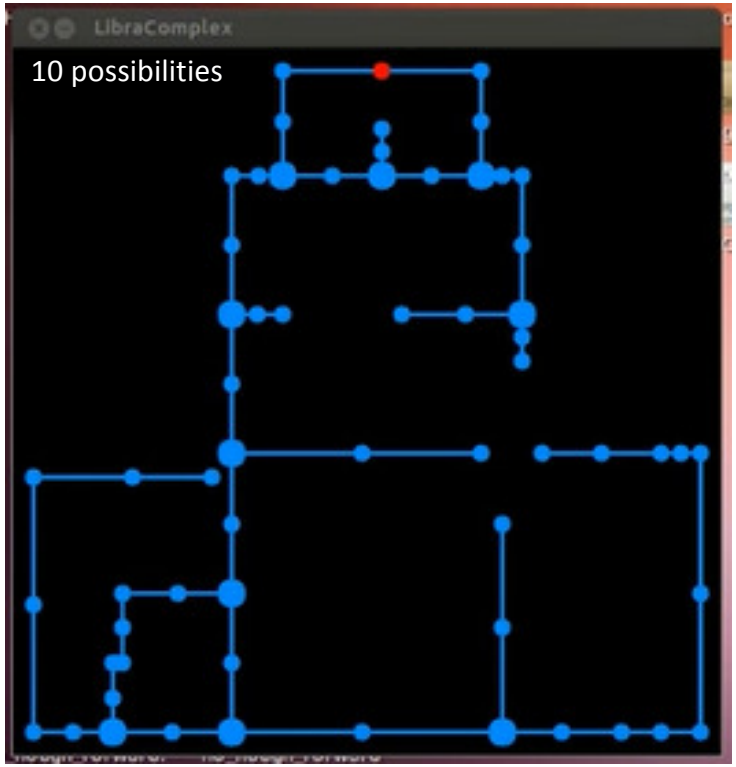
Don't worry about the goal.
Instead, make default motions…

Localization

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



TL



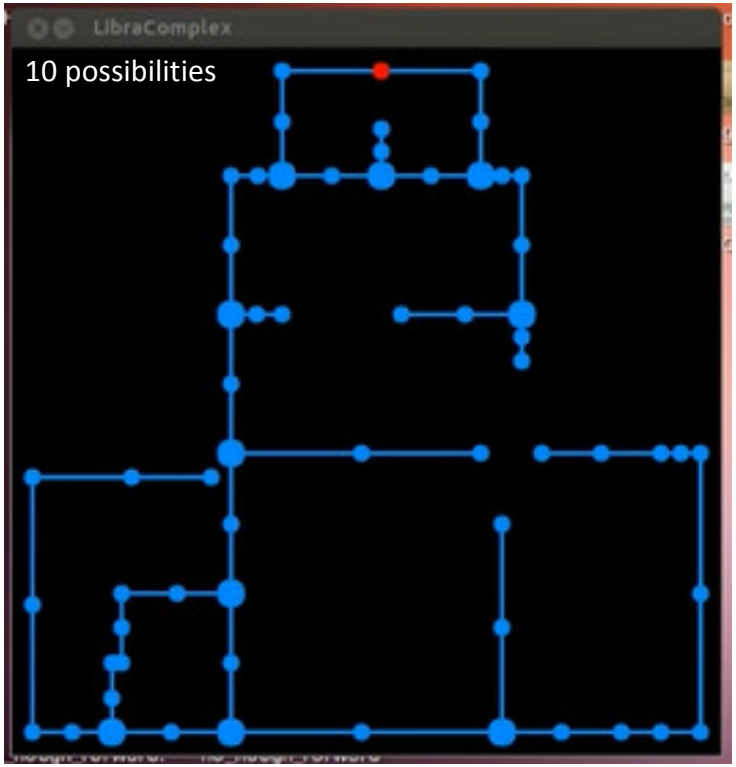
We could be anywhere…

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

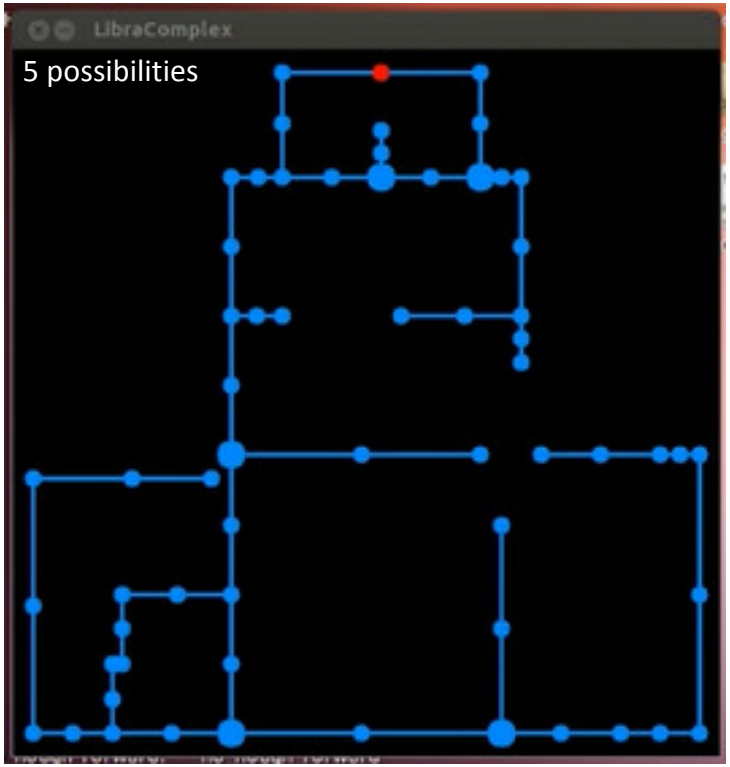
TL

Localization

Keep going, *now in the specified direction…*



TL



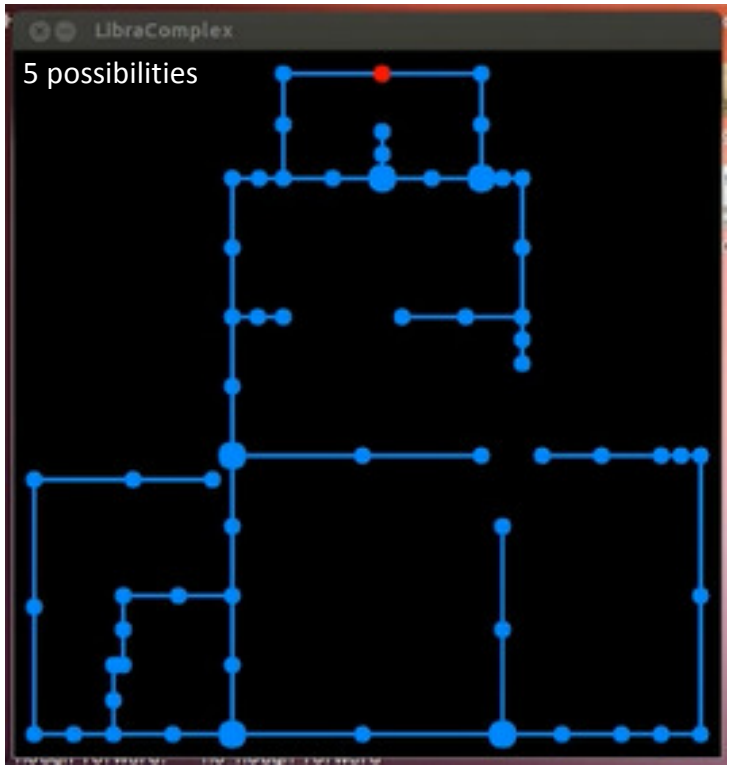
previous distribution of poses

… cull again

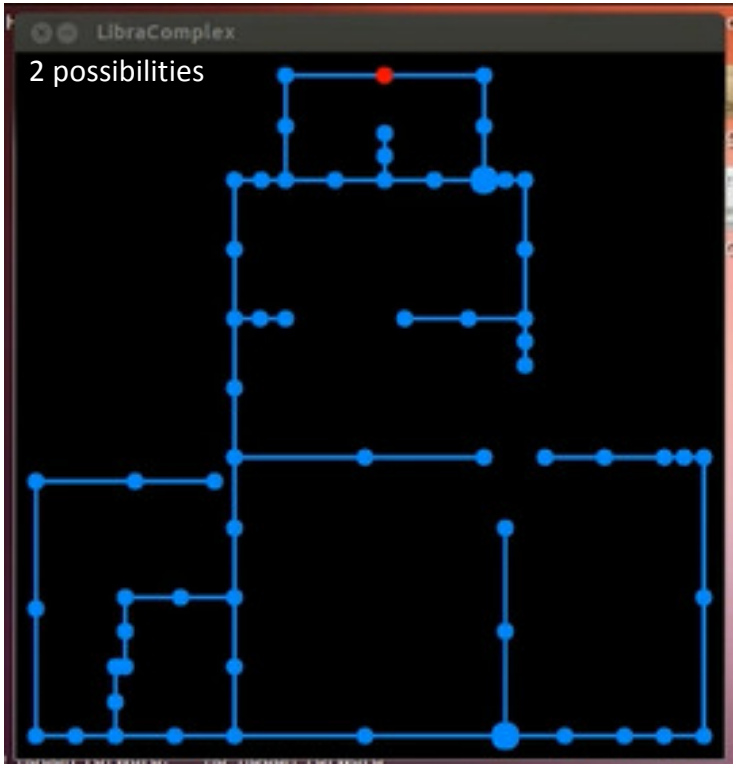
TL

Localization

When it knows its location, plan and head to the goal…



TL

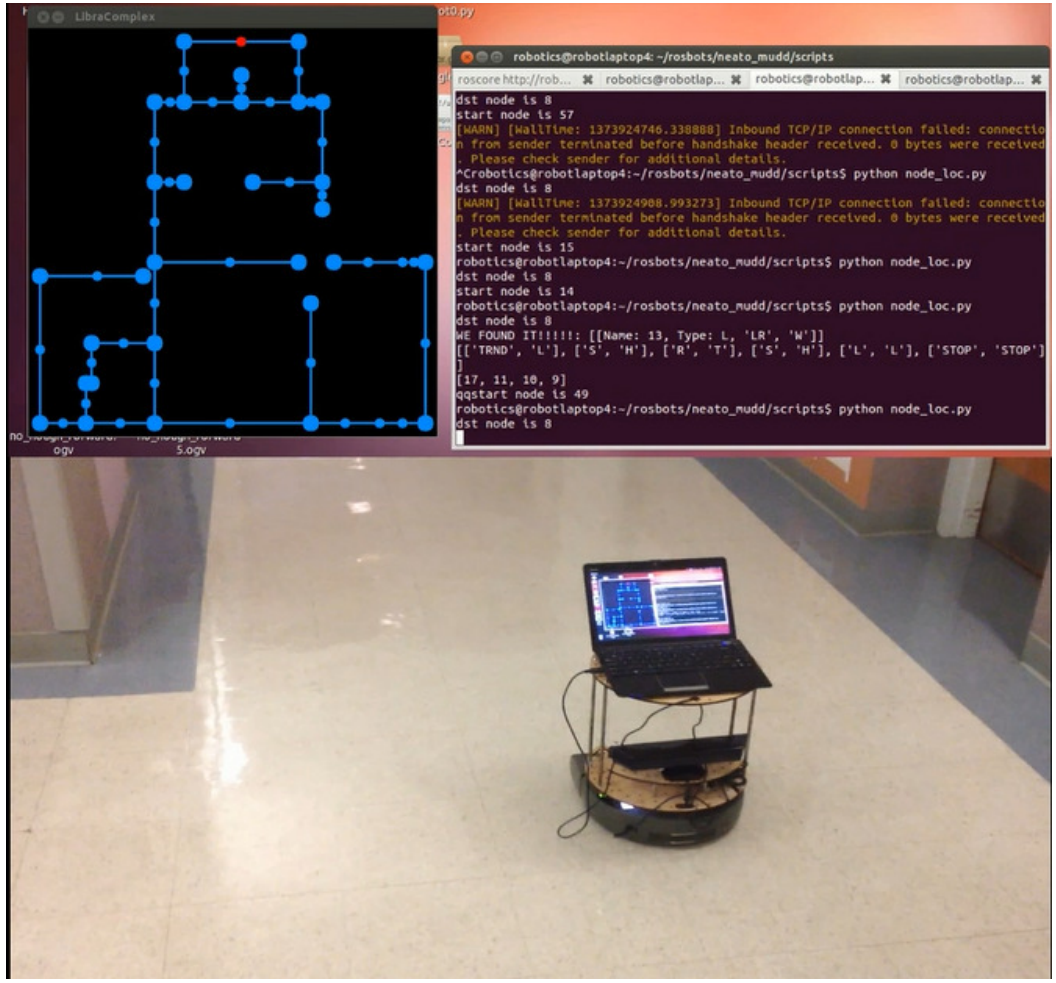


previous distribution of poses

… cull again

LR

Markov/Monte Carlo localization



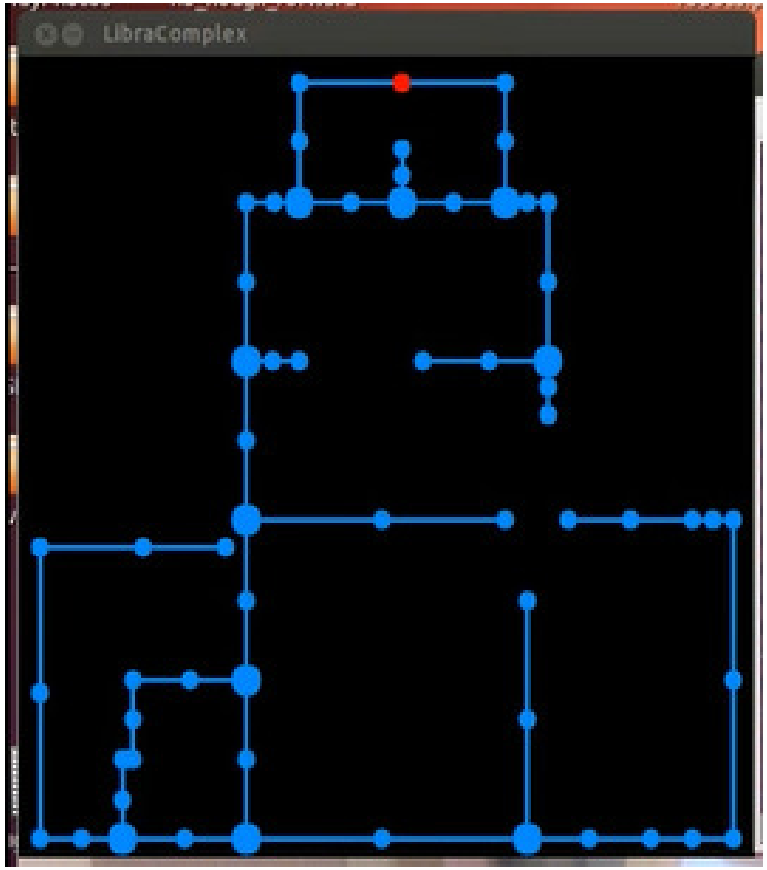
loc_short.mp4

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*?

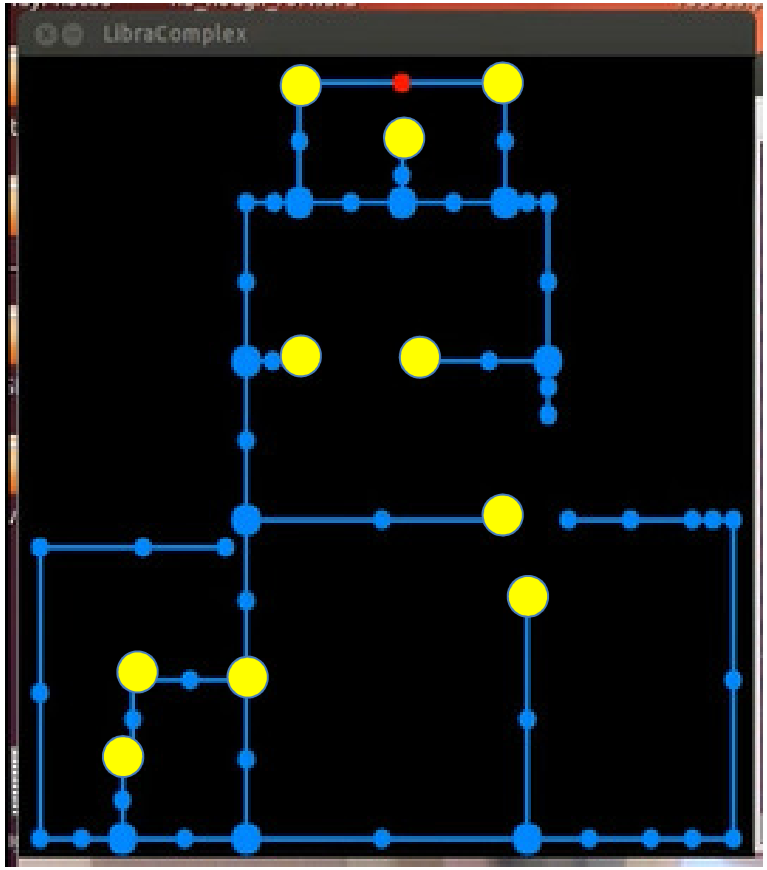


We're at a TL…

Which way
should we go?

Can we be more *deliberate*?

Suppose the robot goes ***left***...



2 LR

2 LL

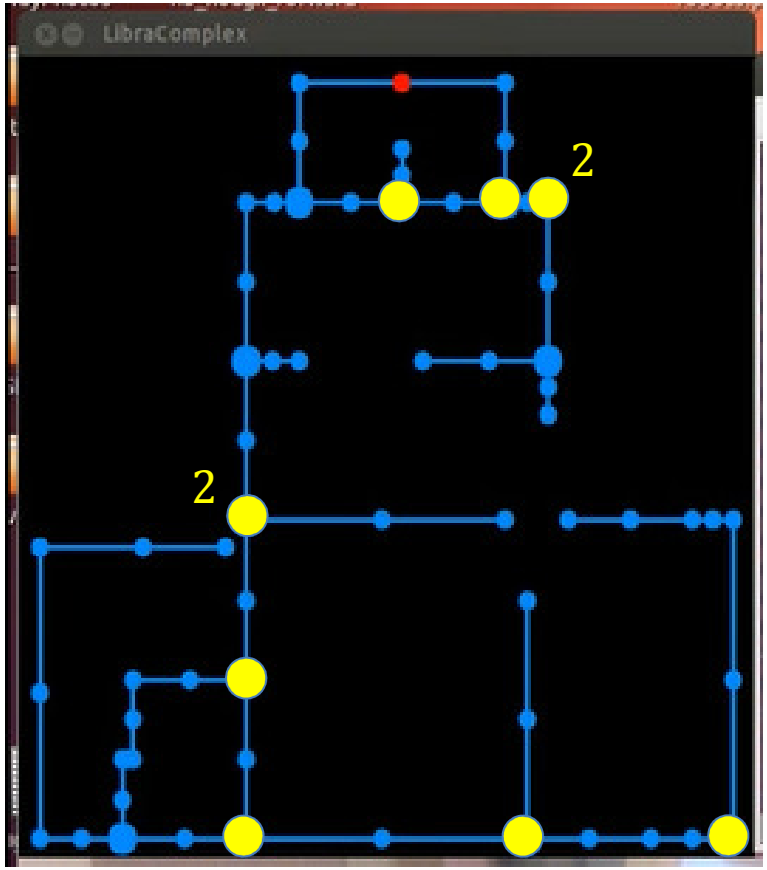
5 DE

1 TL

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

Can we be more *deliberate*?

Suppose the robot goes ***straight***...



1 LR

2 LL

1 TR

6 TL

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

Which possibilities are *better*?

left	*straight*
2 LR	**1 LR**
2 LL	**2 LL**
5 DE	**1 TR**
1 TL	**6 TL**

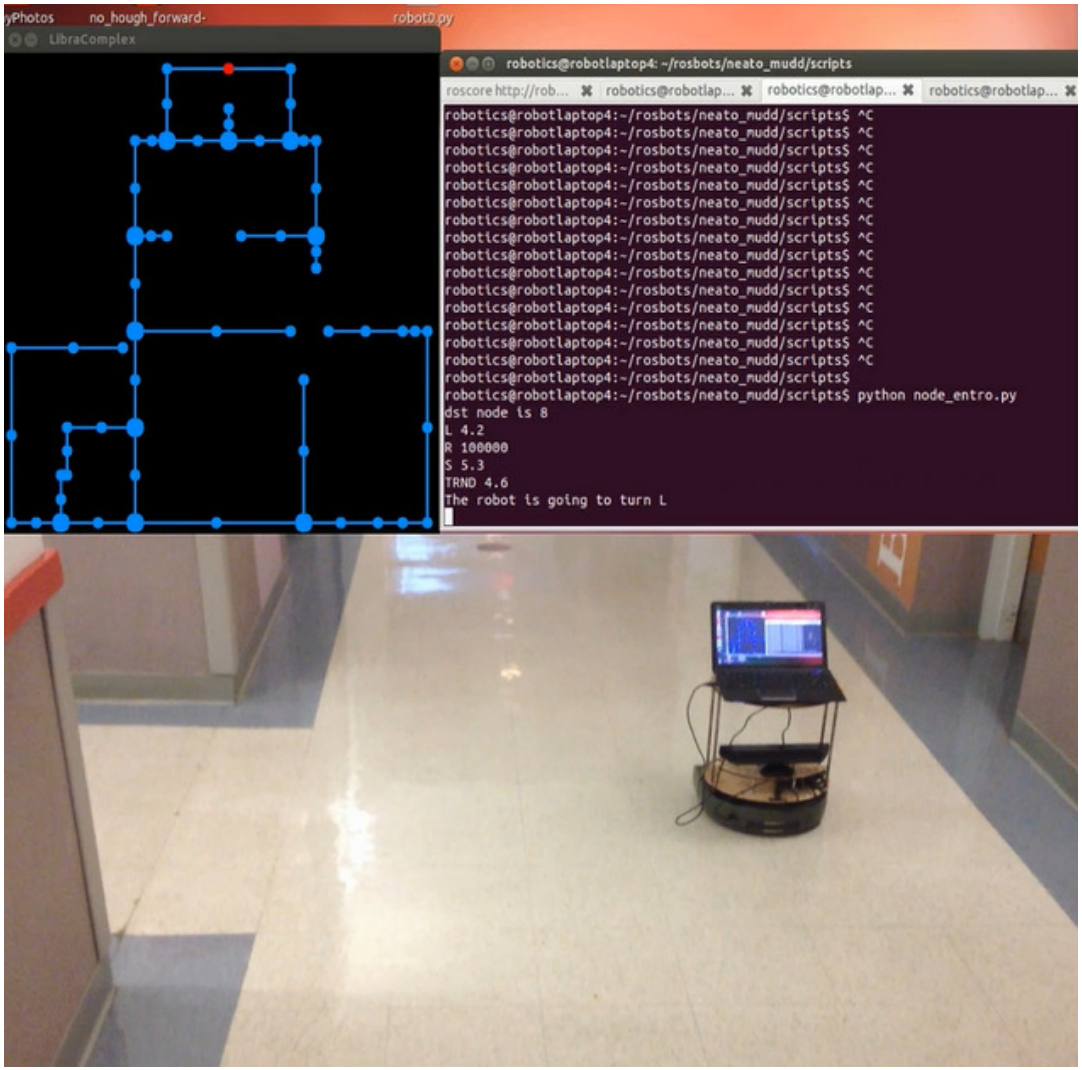
Which of these two possible actions (and their results) should we prefer?

Entropy says, *Go left!*

left	*straight*
2 LR	**1 LR**
2 LL	**2 LL**
5 DE	**1 TR**
1 TL	**6 TL**
1.76 bits	***1.57 bits***

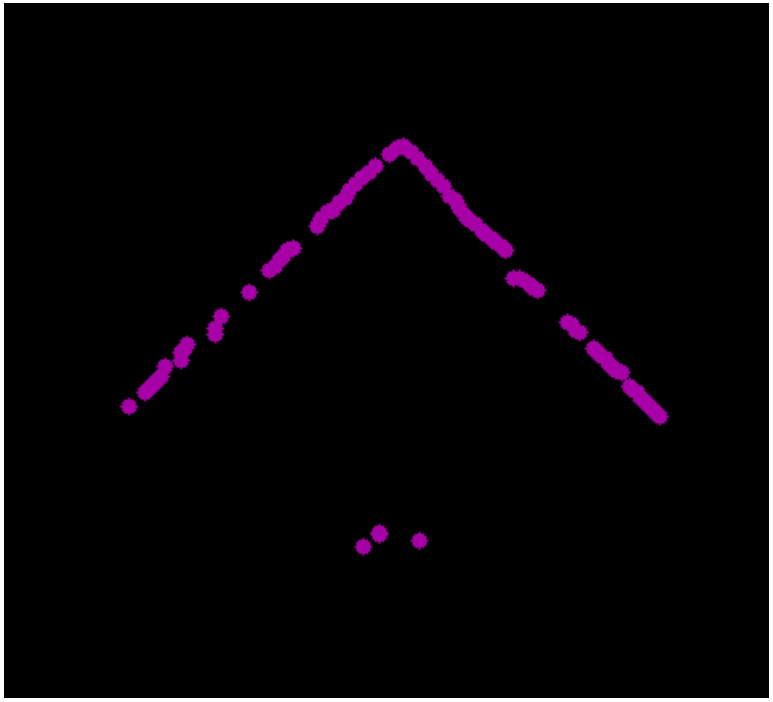
Flatter is better

Entropy-based localization



entro_short.mp4

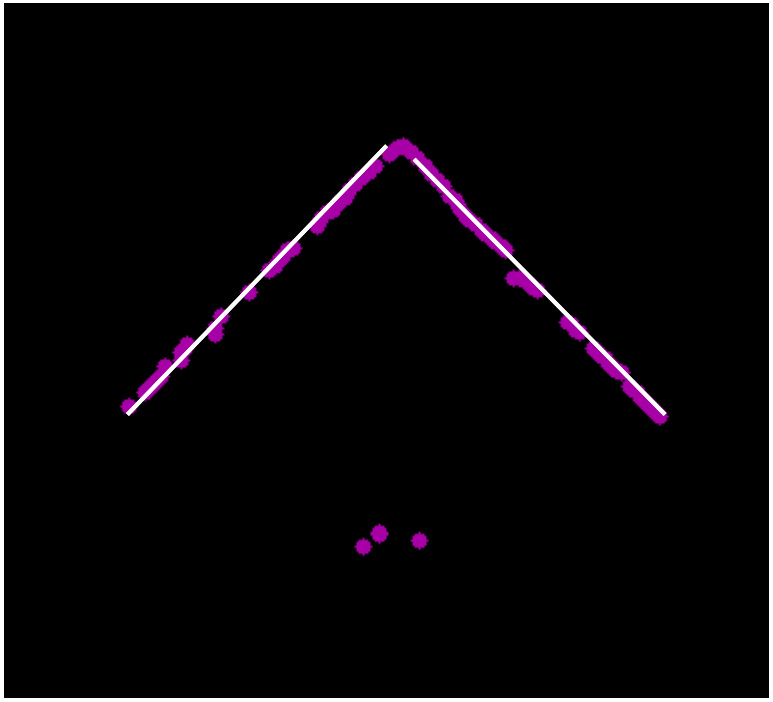
Next? Autonomous 3d mapping…



linked to raw_corner_cropped.m4v

"Verticalized" laser scans lead to artifacts ~ walls vs. ribbons

Better!



linked to hough_corner_3d_cropped.m4v

First simplify the structures, *then* paint them …

Verdicts & insights

2d matching? not fast *(yet)*
not used on Neatos *(yet)*
more geometry to exploit

surprisingly good!

slowness? don't use linear search!

integration? any interest?

opportunities!

accuracy? find a more *"exact"* pose from 3d

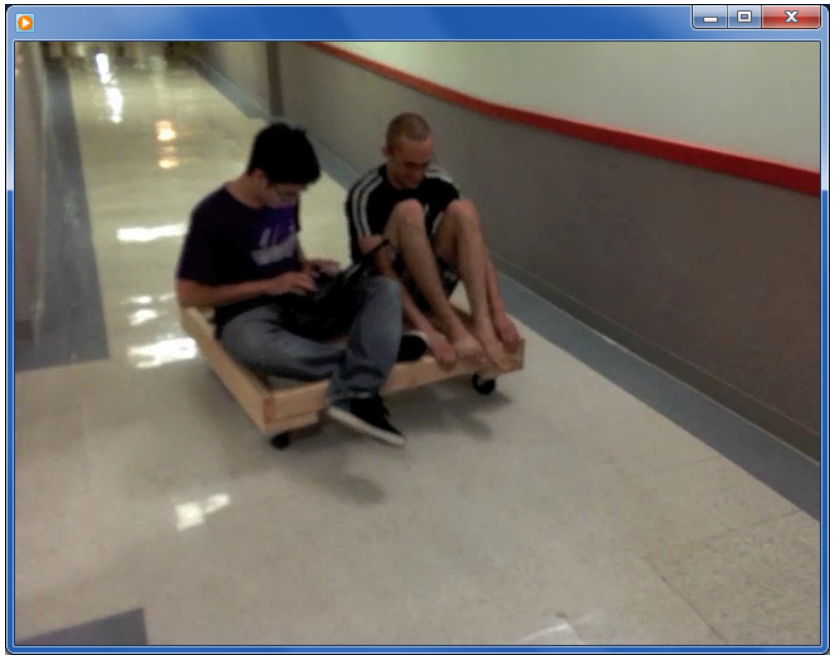
3d overall? **Yes!**

This is next summer's robotics agenda...

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

New platforms…

Microdrone



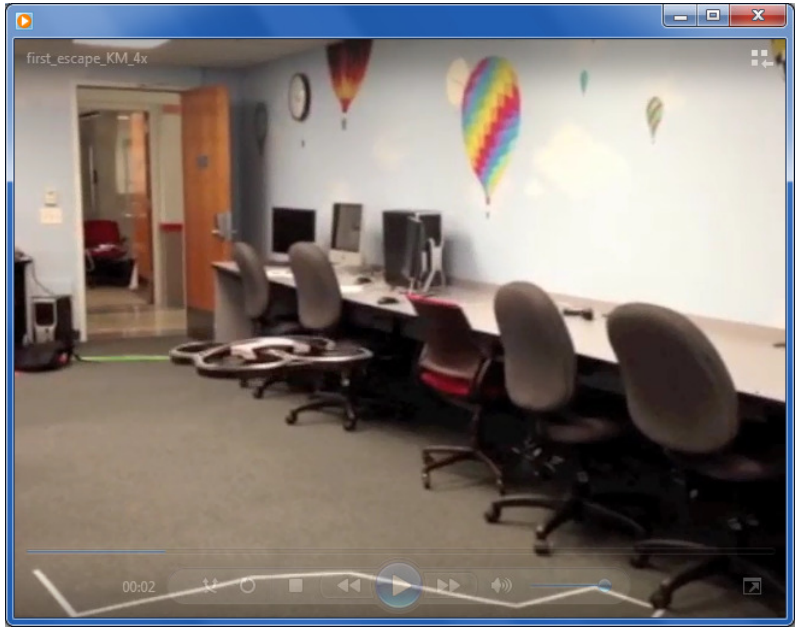
"The Sofa"

LeapMotion

Thoughts?
Questions?

Other runs

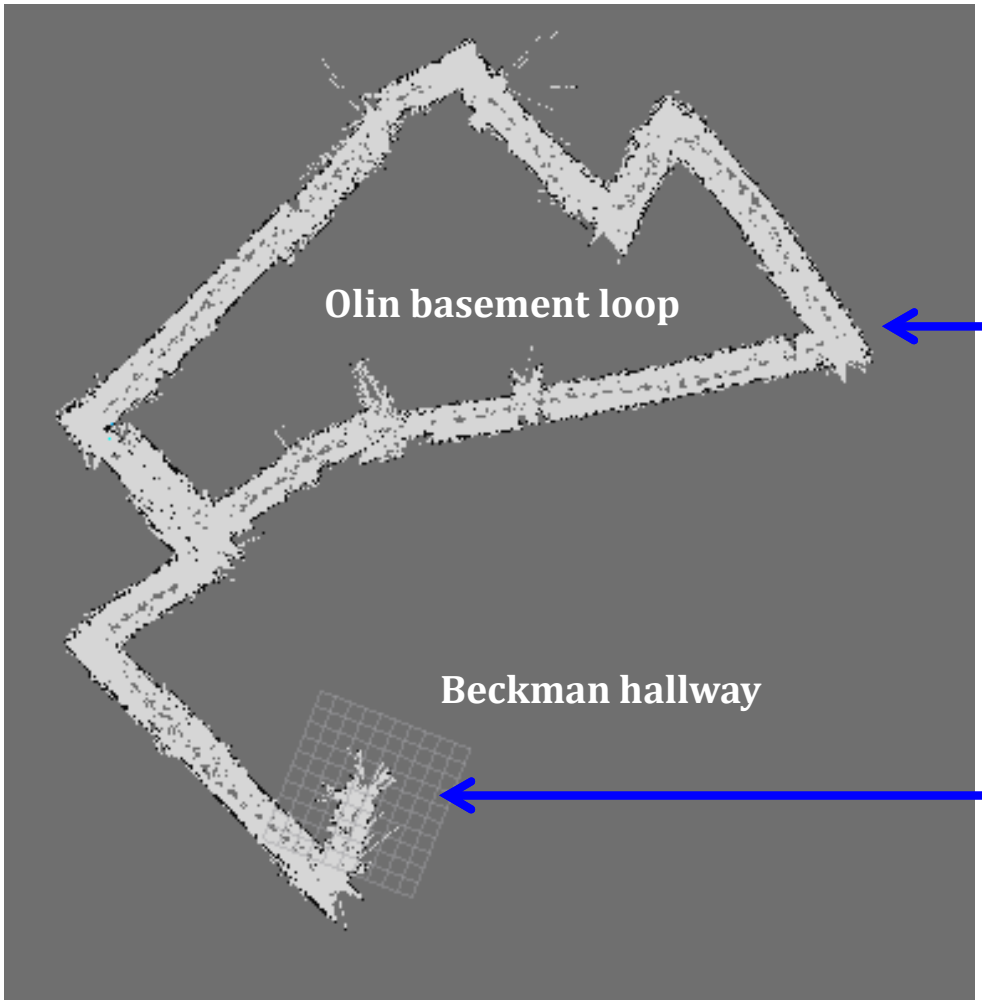
Why not localize once and just go!?



A view from the back of the lab

`gmapping`

an algorithm for autonomous laser-based mapping



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

straightens corridors

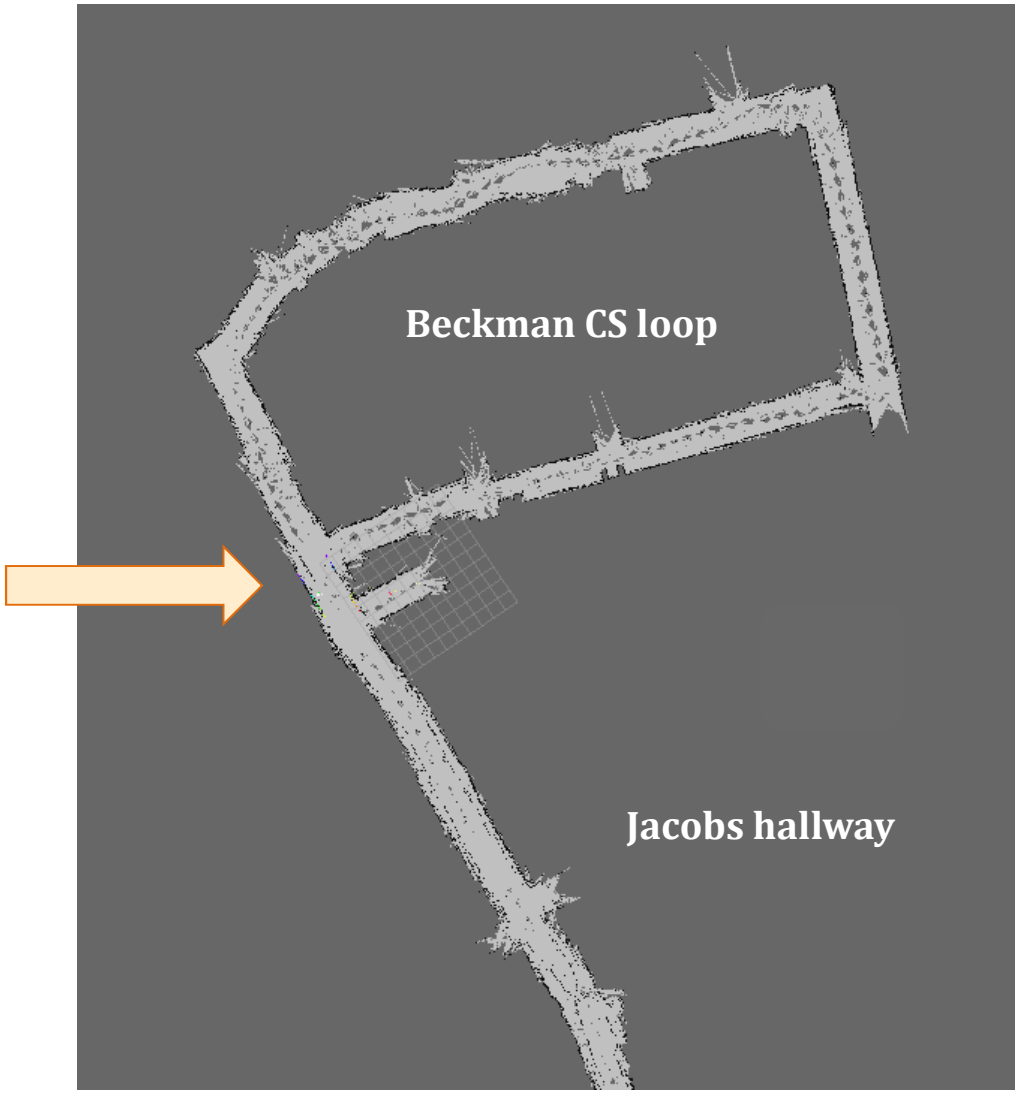
(2) search for distant-past similarities to "close loops"

corrects slippage?

gmapping

... can *miss* "obvious" loops

Laser scans aren't very ***distinctive***

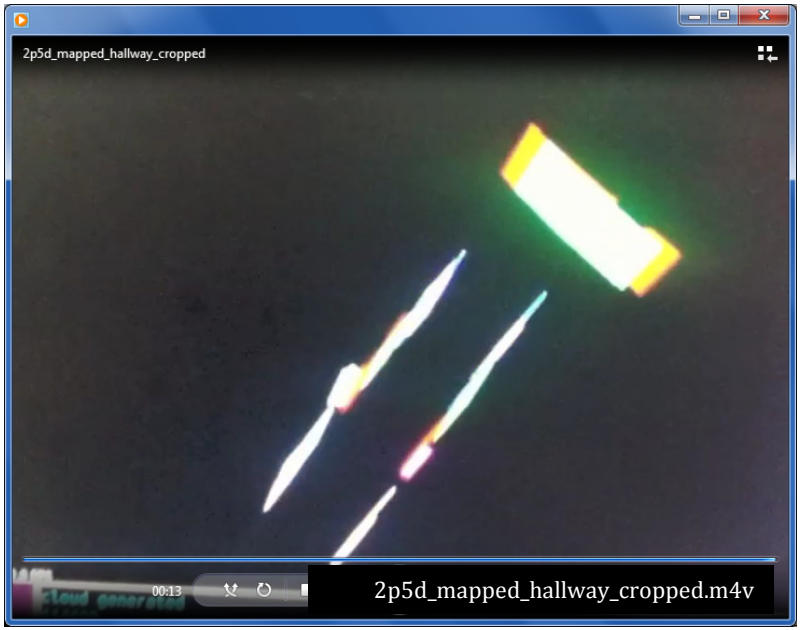


Better landmarks?

... probably texture in 3d

Building the map…

the robot's final position…



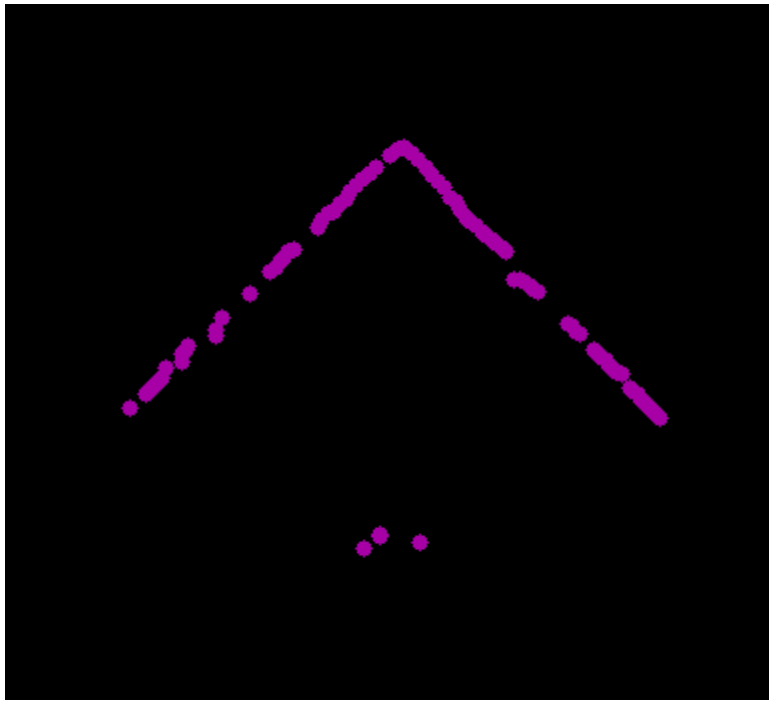
… after creating its 2½-d map

Delegated to 2014: painting these walls in real time…

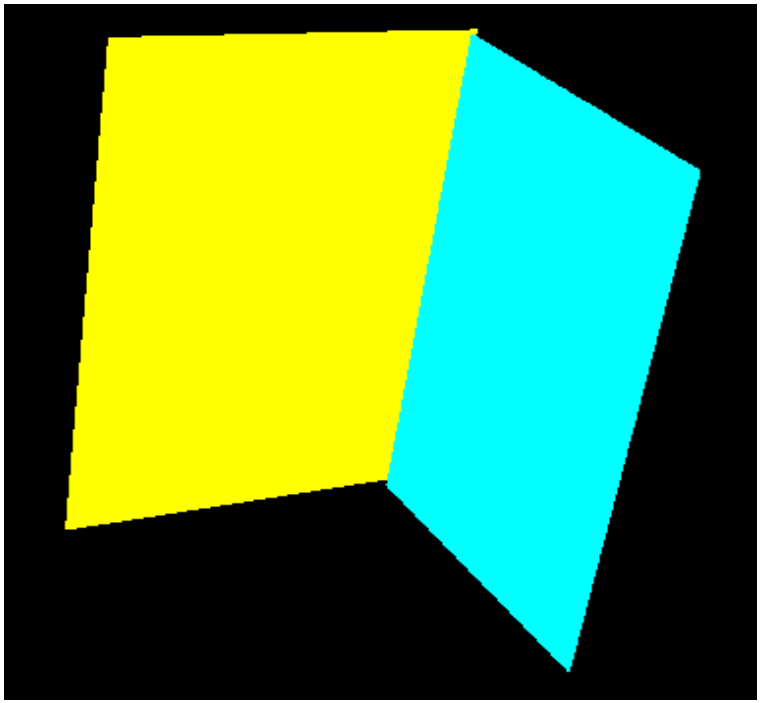
2½-d mapping

Idea:

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

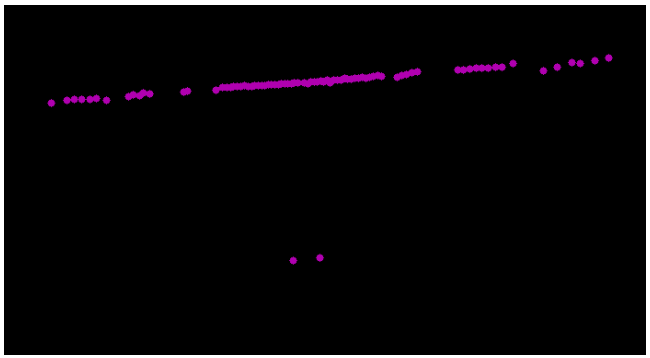


Laser-scan of a corner



Now, as vertical planes

Painting the walls…



Laser-scan of a wall

textured model

Image of the wall

Painting the walls…

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

Contributions…


"LUKE, I AM YOUR
PARENT FUNCTION."


17
IS
14.3 MPH
TTL
HARVEY
MUDD
IS
THE
BEST
0.3
18
HI
BYE
RAPTOR
CAT
INDIA
Contact the Control Desk to extend
0.0

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

This afternoon…

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

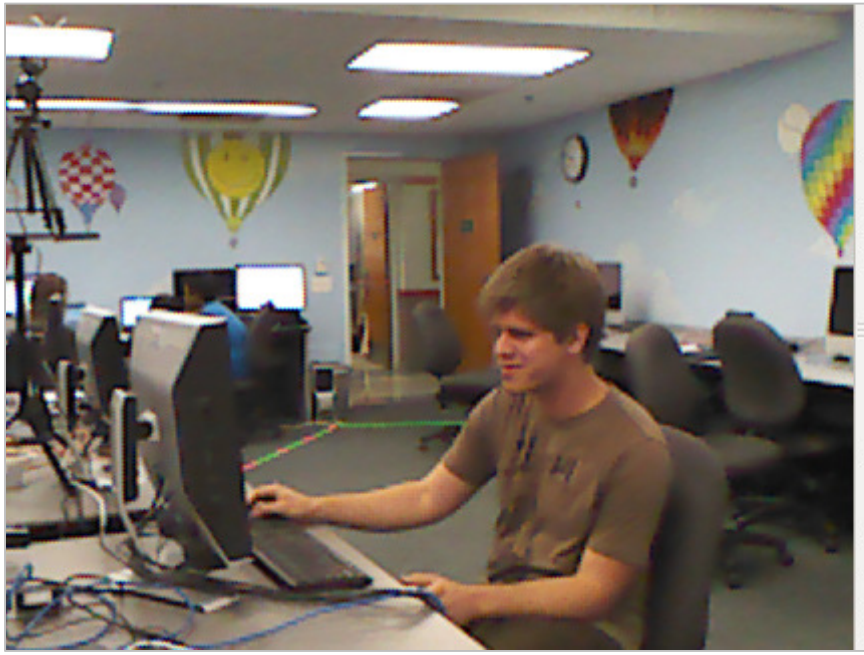
** ***Except in the games cubicle:***

(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

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

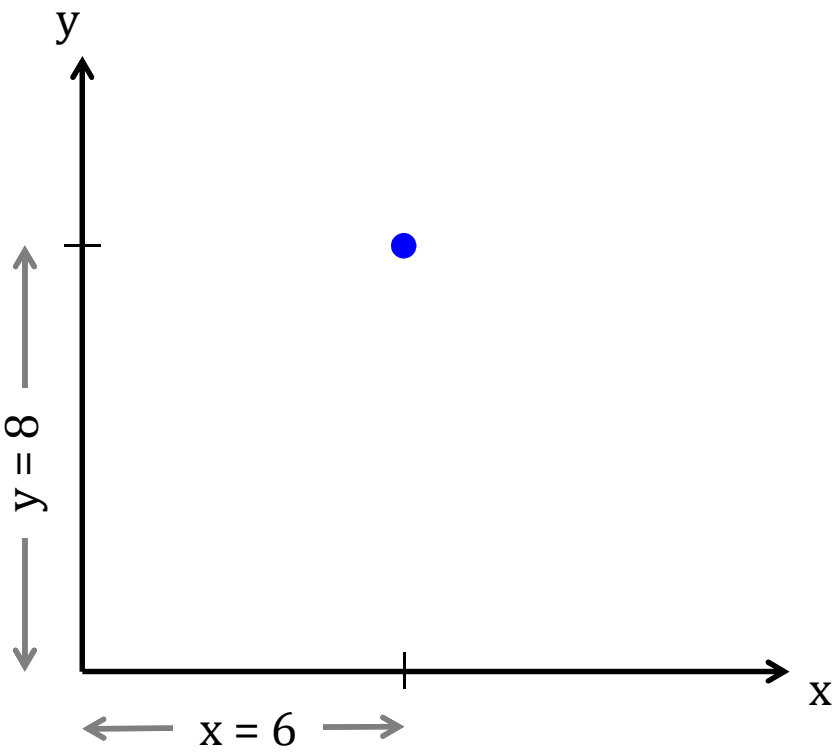
AI Tasks

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

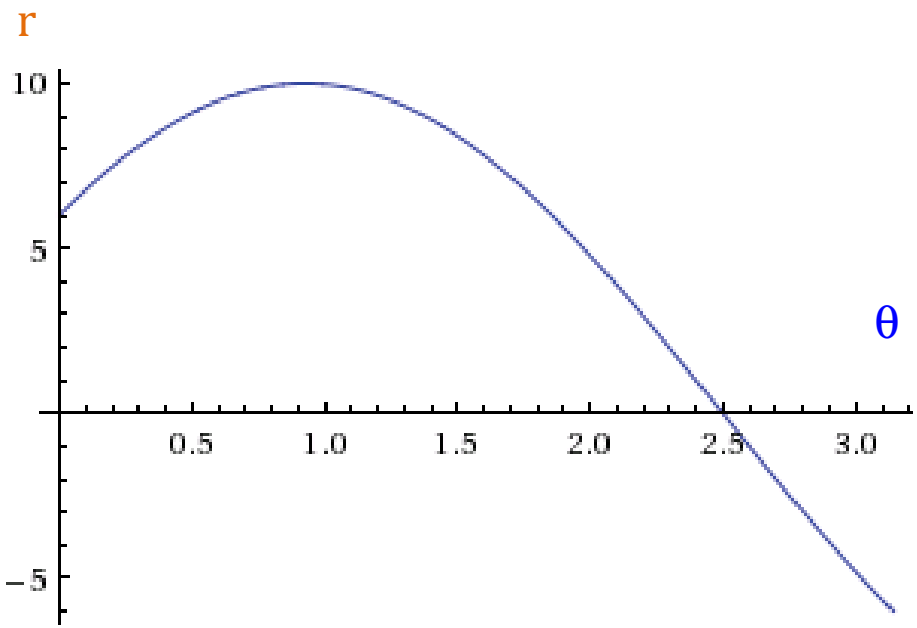
wall placement determines each intersection ***type***

Hough transform

Point space



Line space

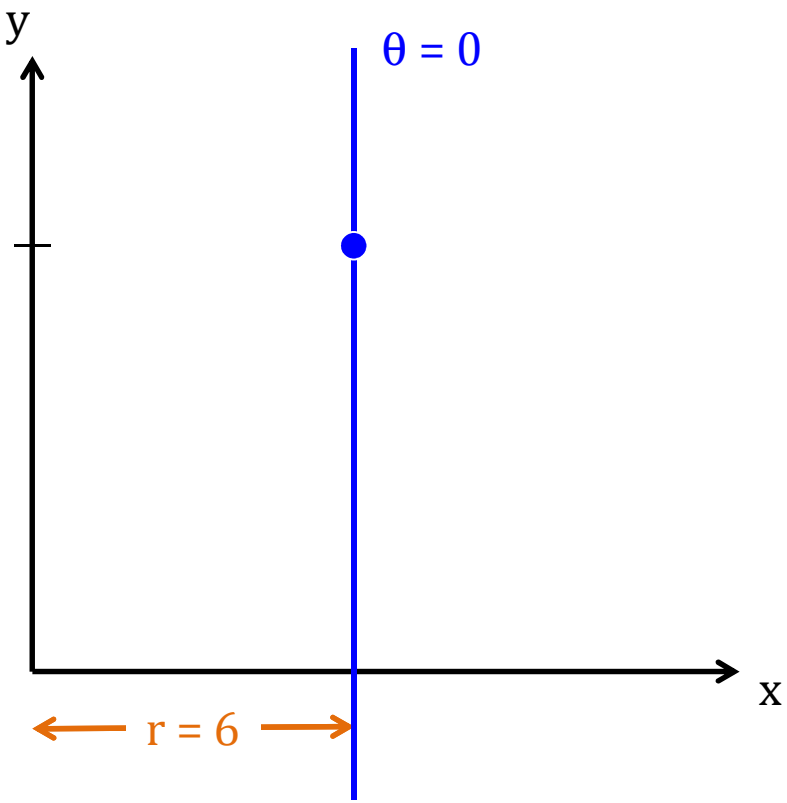


Consider ***one point*** of a laser scan

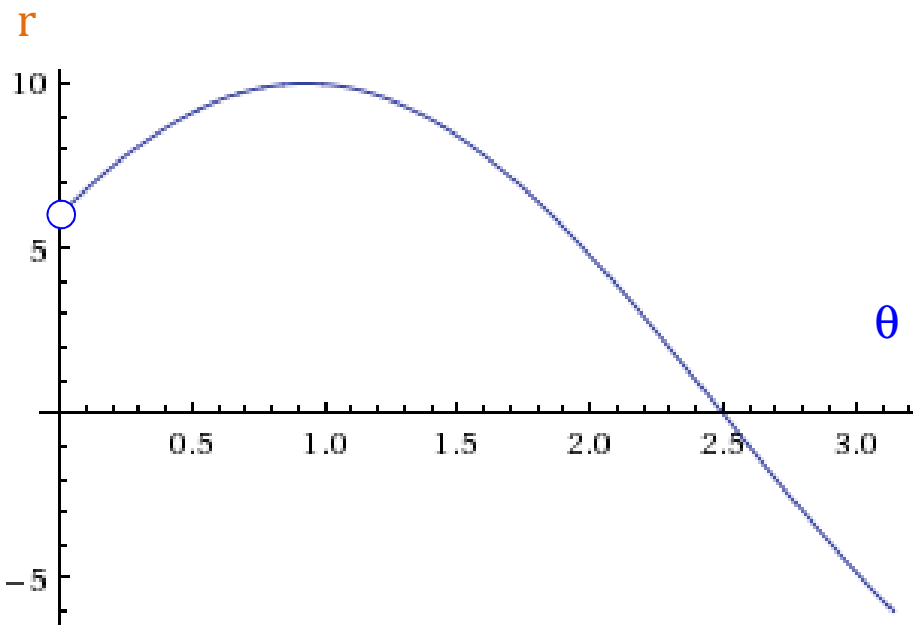
These are ***all*** of the lines through it

Hough transform

Point space



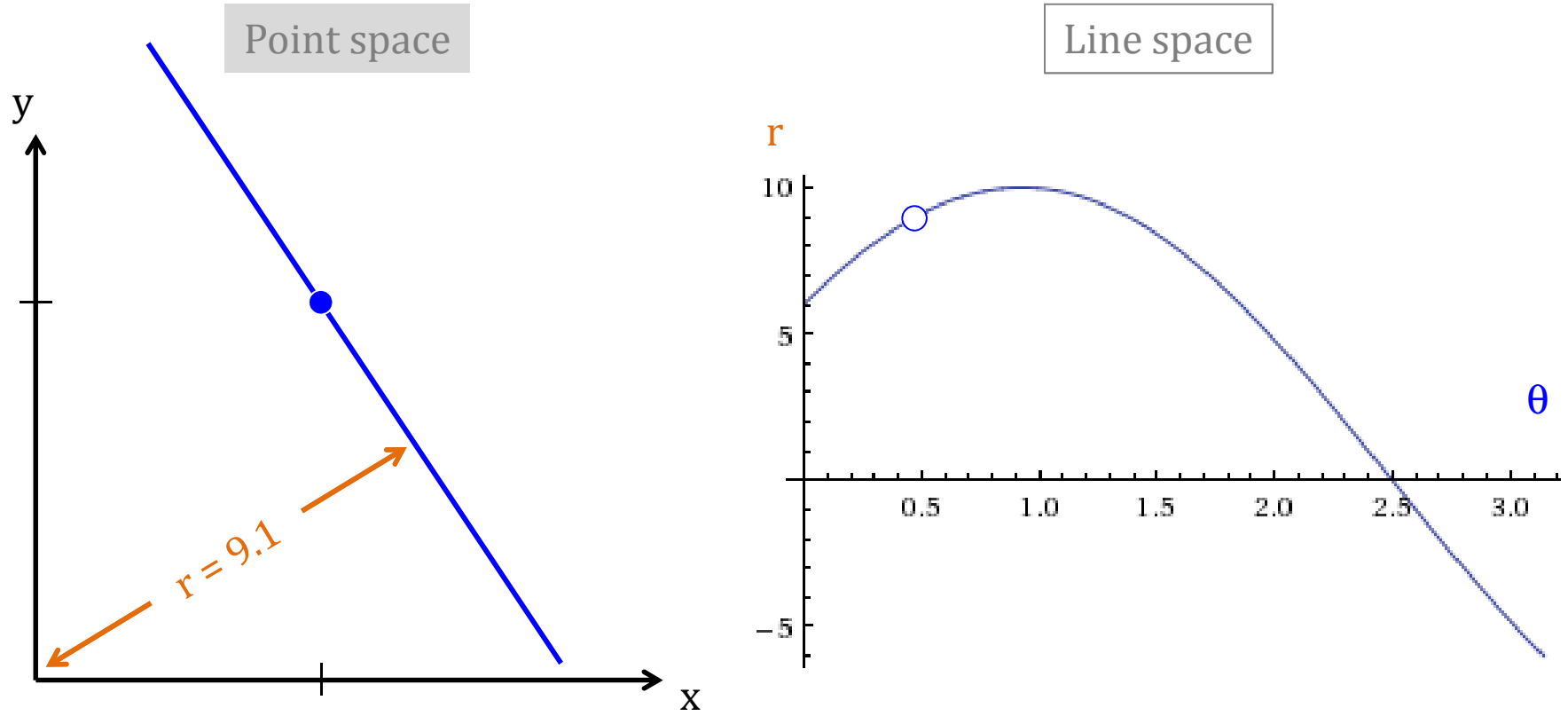
Line space



vertical line ~ θ = 0

The dot is the line r = 6, θ = 0

Hough transform

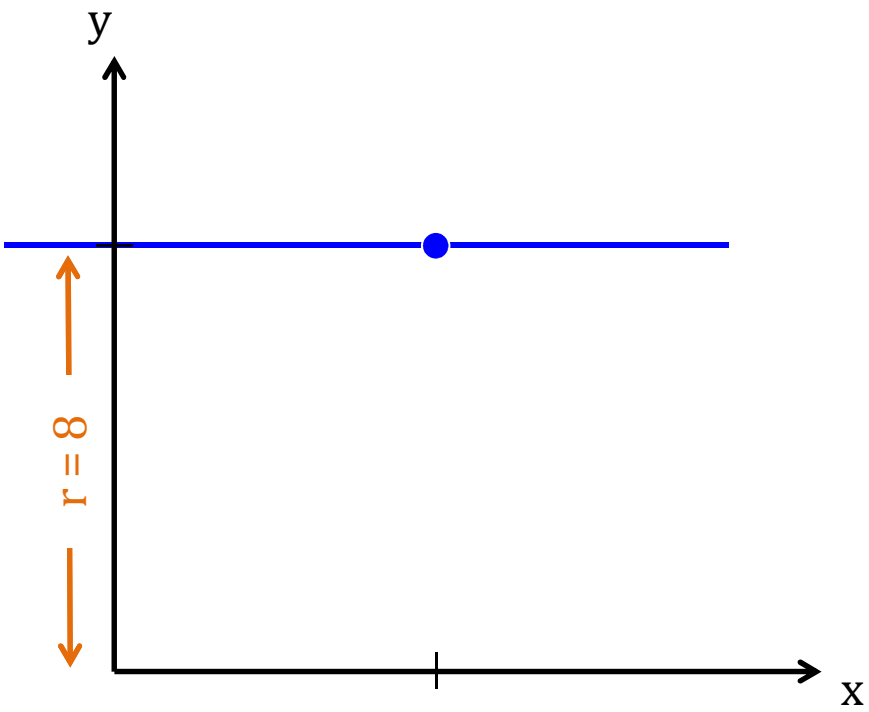


angled line ~ θ = 32°

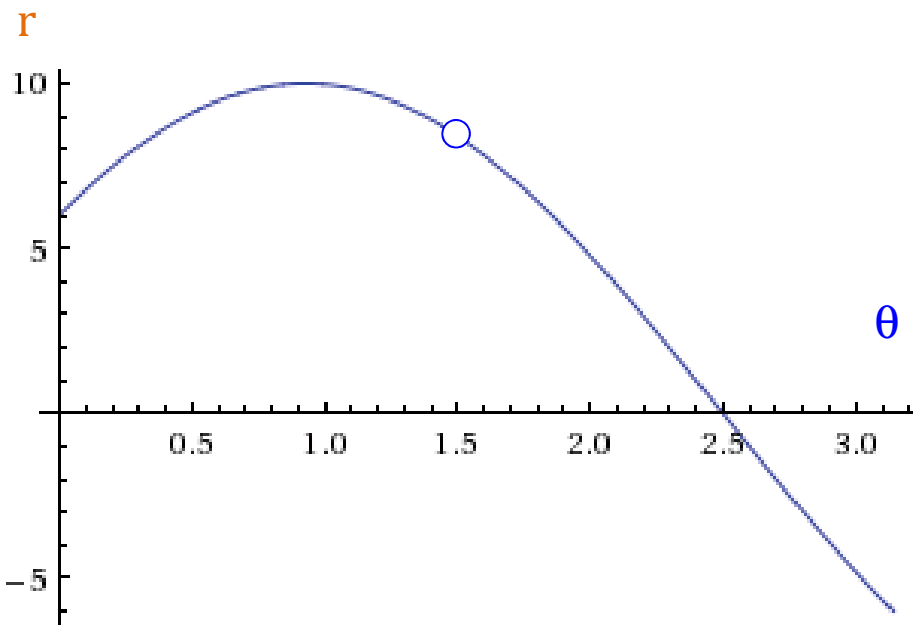
The dot is the line r = 9.1, θ = 32°

Hough transform

Point space



Line space

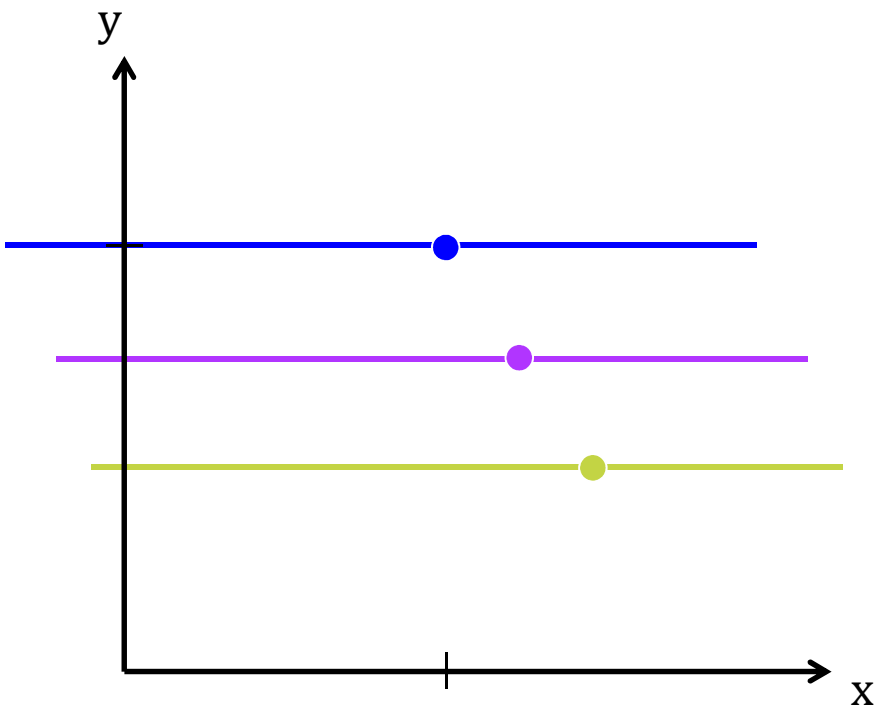


angled line ~ θ = 90°

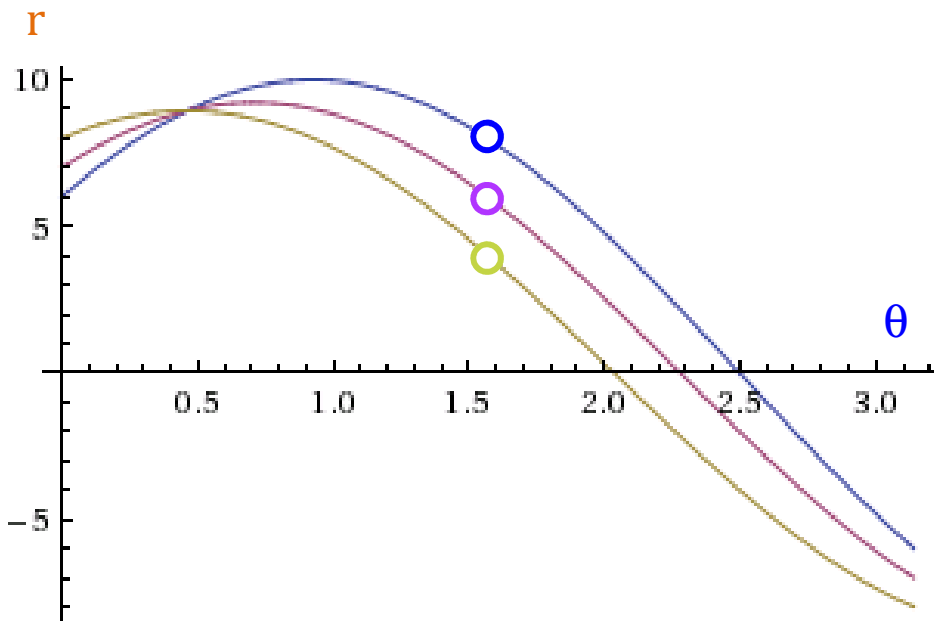
The dot is the line r = 8, θ = 90°

Hough transform

Point space



Line space



three points of a laser scan

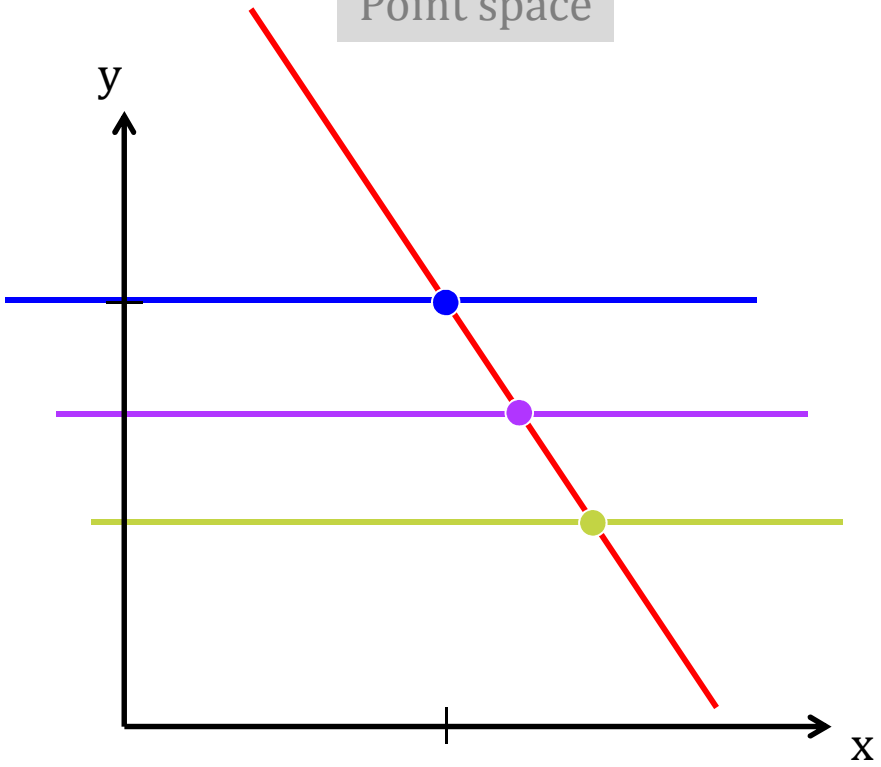
(horizontal lines are shown)

all the lines through all three

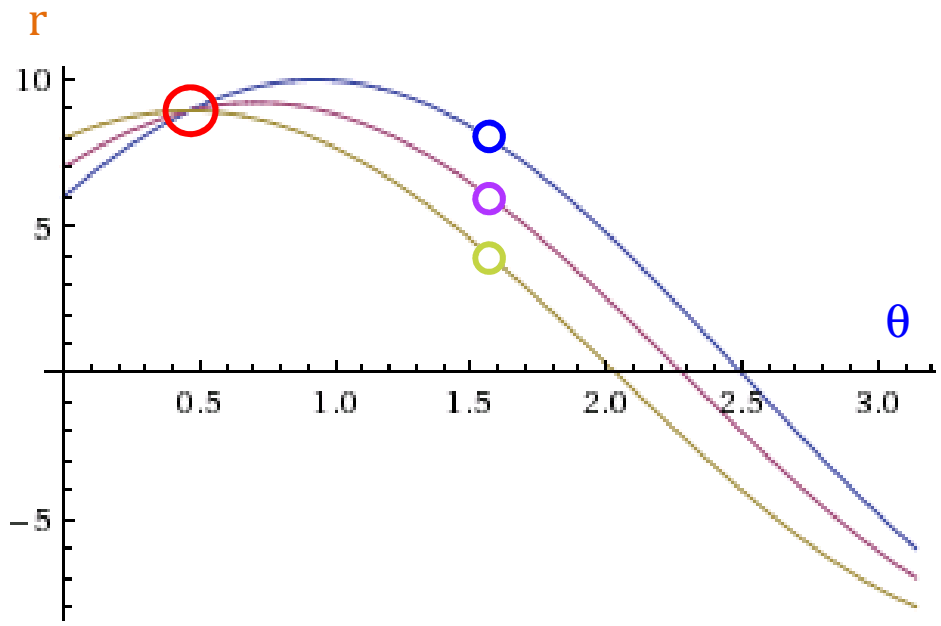
(dots ~ the shown horizontal lines)

Hough transform

Point space



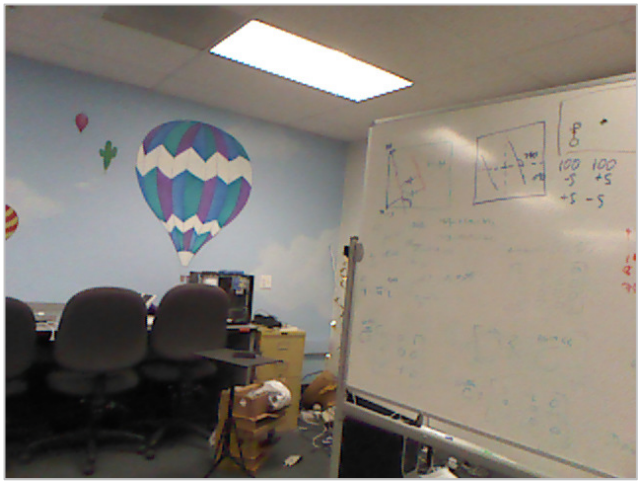
Line space



yield lines with substantial support in the original data

peaks here in line space

Image matching?



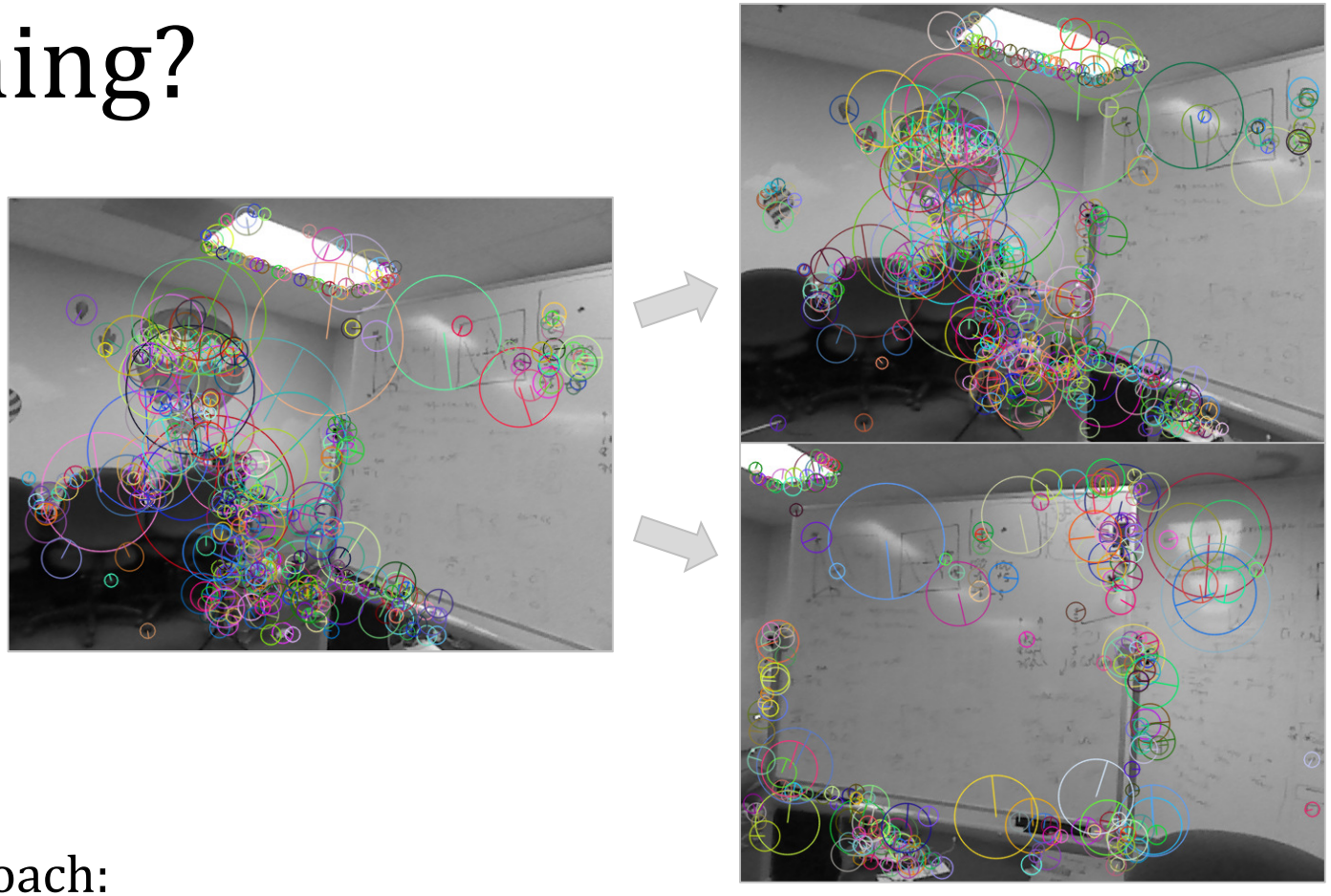
new image

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:

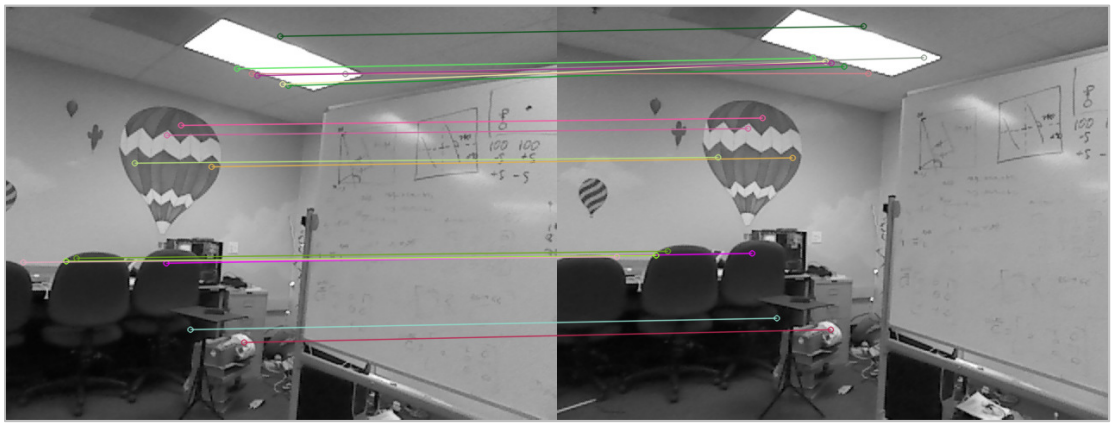
- Find features in each (SIFT)
- Consider all matches (FLANN)
- **Ensure bidirectional constraints**
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

Keep only *two-way* best matches

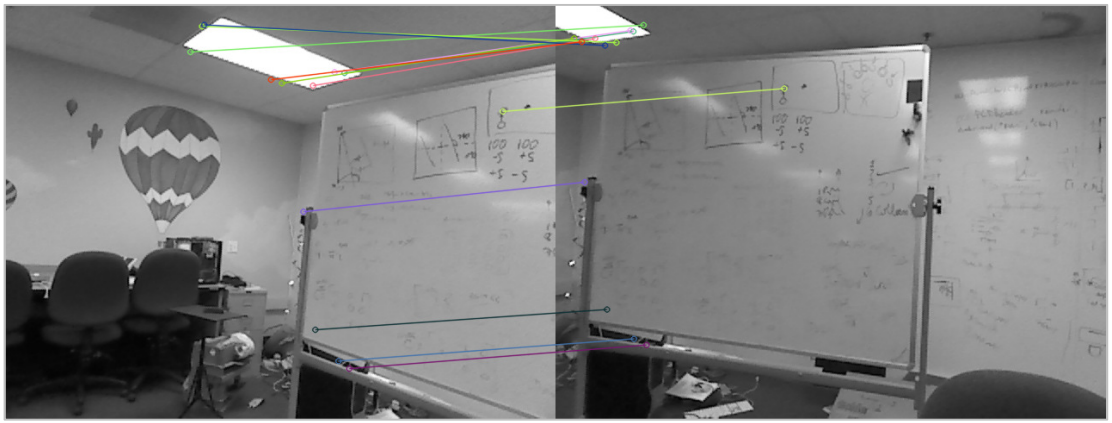
Our approach:

- 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



1st-best match

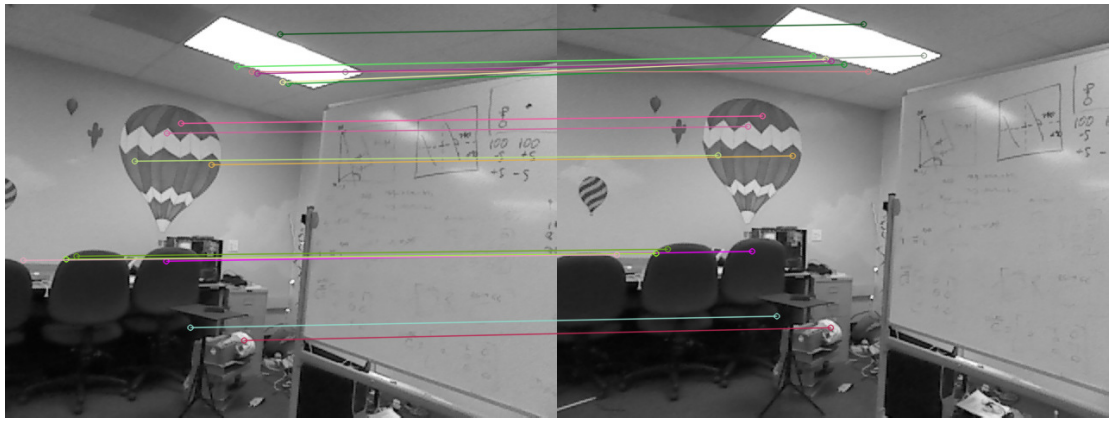


10th-best match

Our approach:

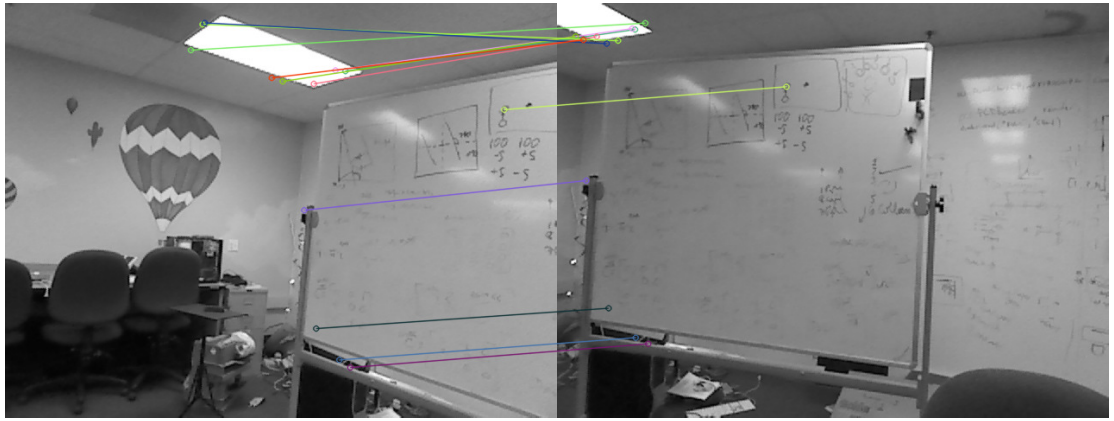
- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (via homography)
- **Best-matching by visual similarity metric**
- Tiebreaking by pixel distances

Visual “closeness”



1st-best match

Average feature distance **~ 40 px**



10th-best match

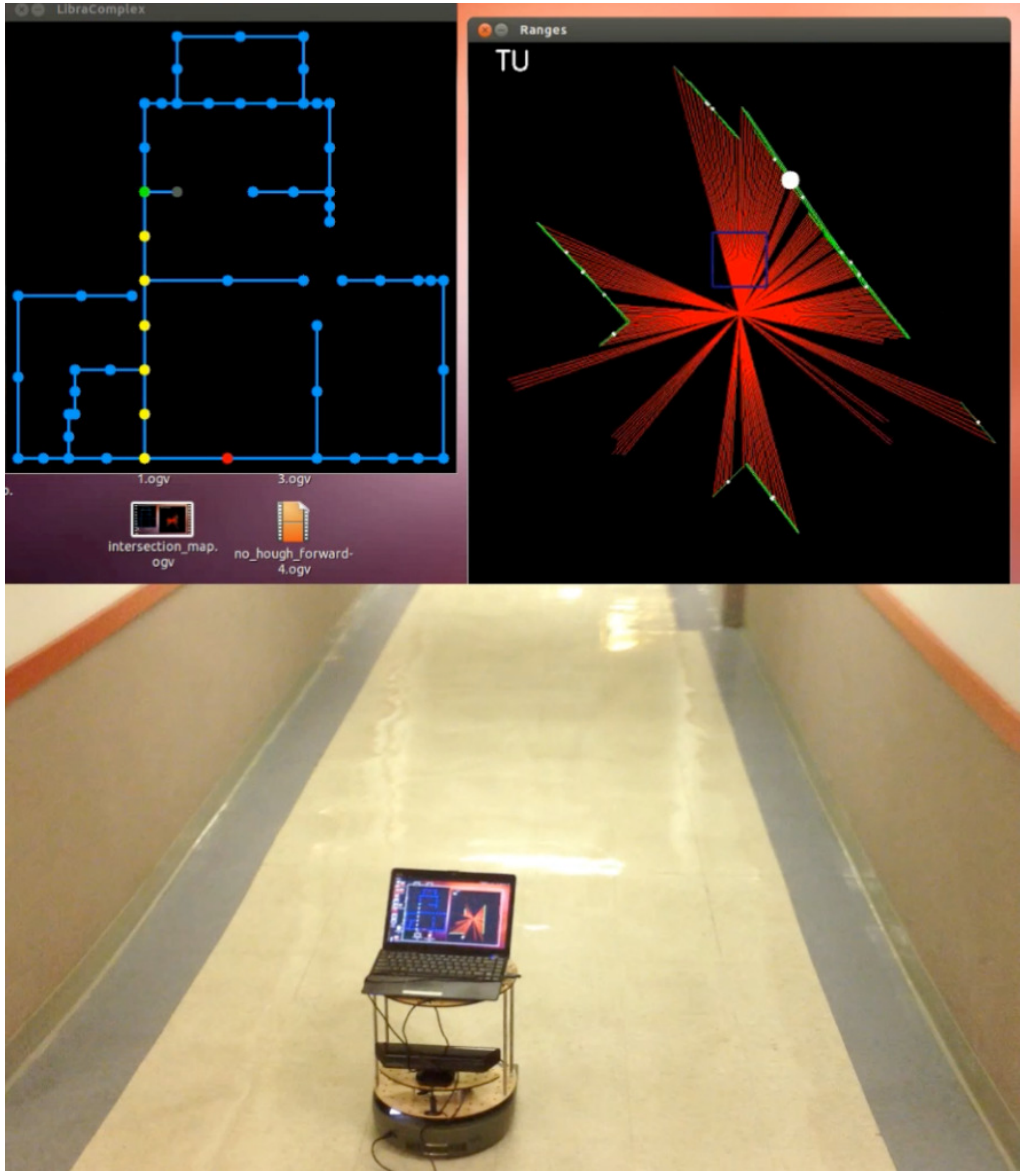
Average feature distance **~ 270 px**

Our approach:

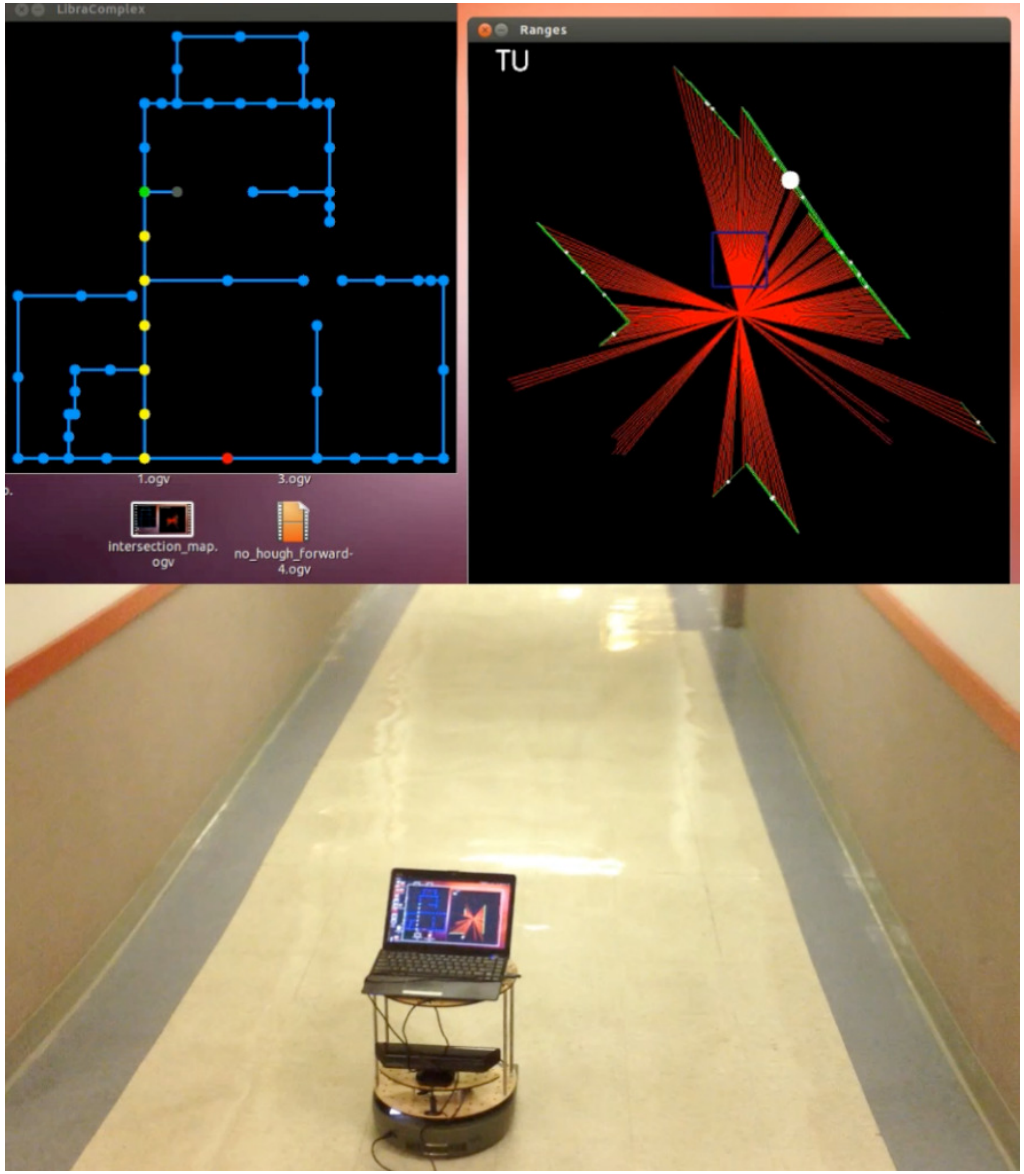
- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (via homography)
- Best-matching by visual similarity metric
- **Tiebreaking by pixel distances**

Pixel "closeness"

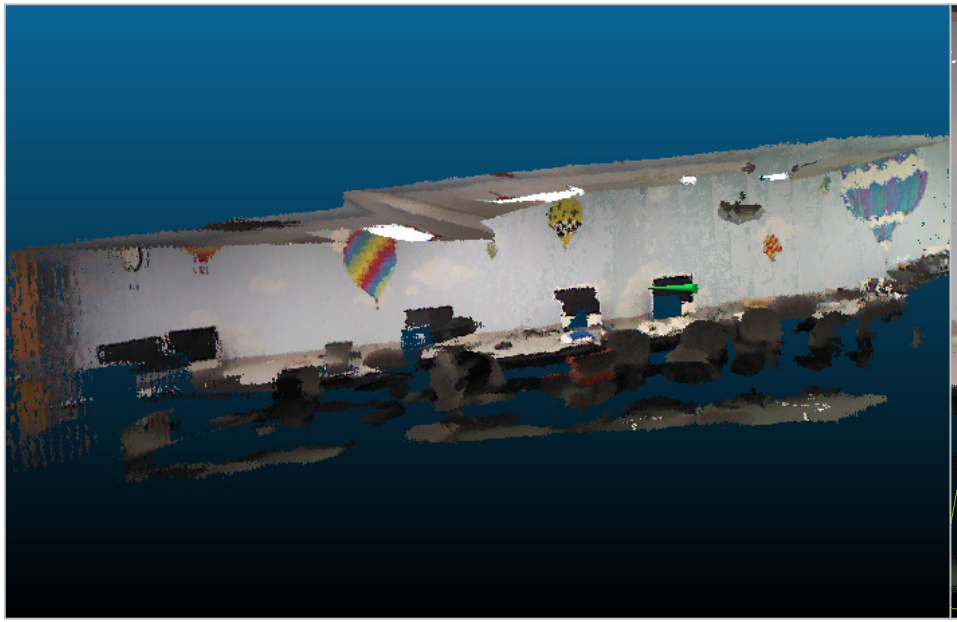
Map-based navigation…



Next: localization



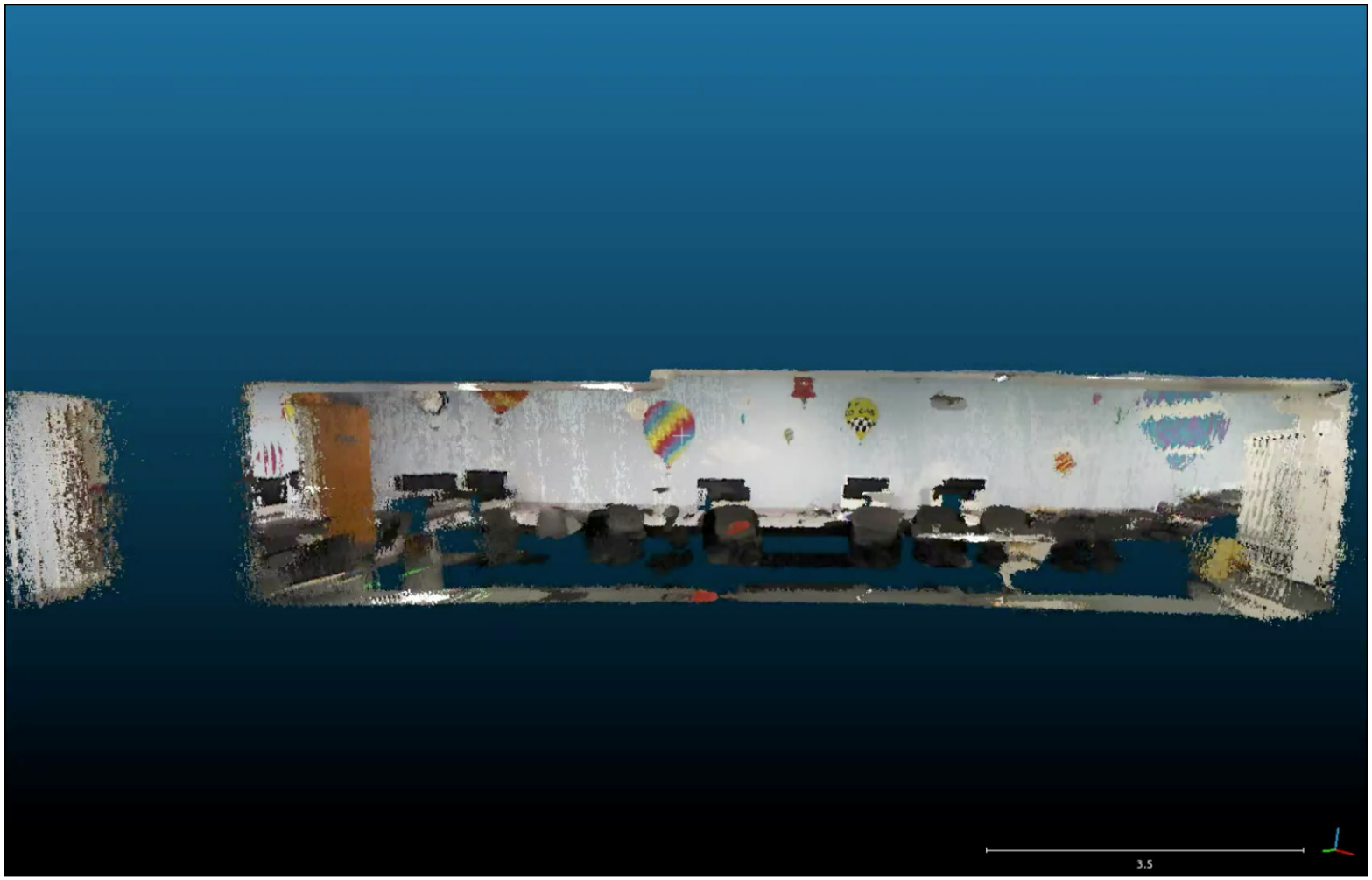
Accessing these models



3d lab model ~ with a surface added

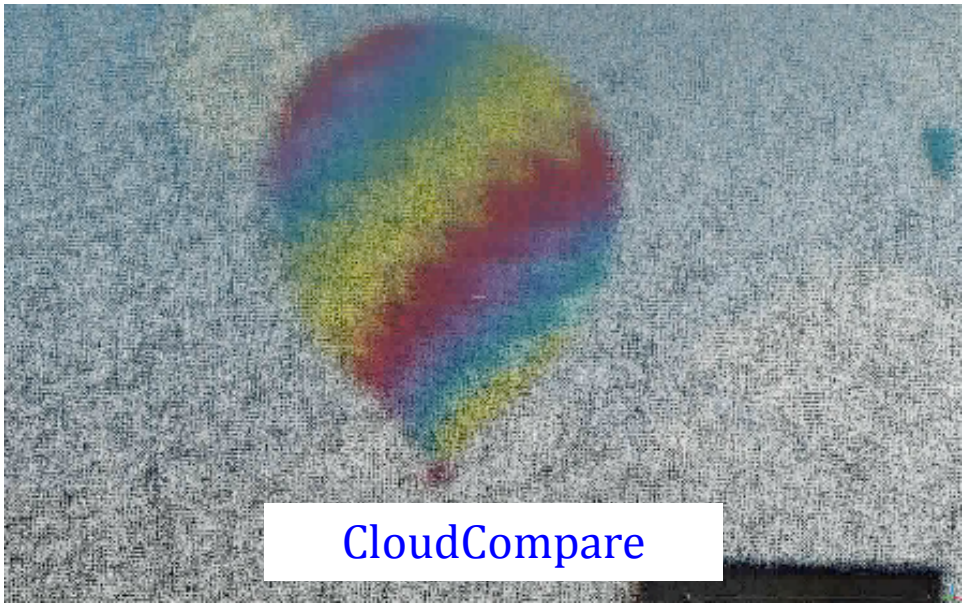
the rendered view from that green cone

Accessing these models



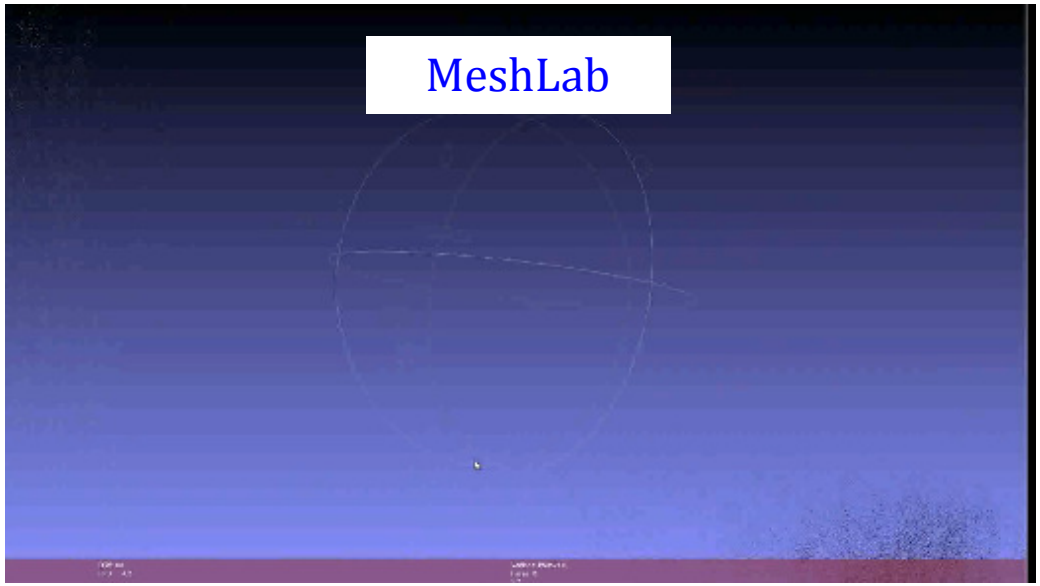
Cloud Compare is a visualizer that can include our own “annotations” …

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?

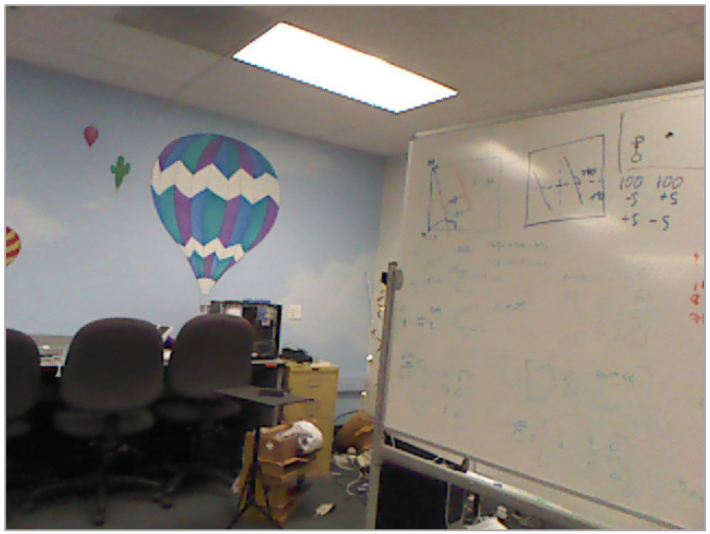
2d – 2d matching

3d – 3d matching

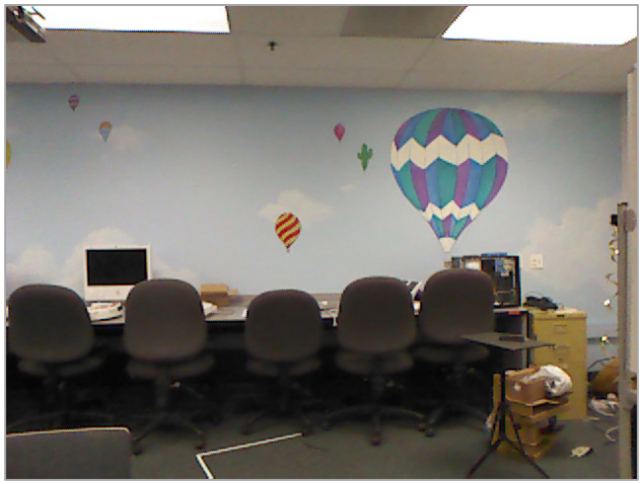
Ideally, a 2d sensing system could localize – *and reason* – within a 3d model

2d – 2d matching

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



we could compare



vs. original images

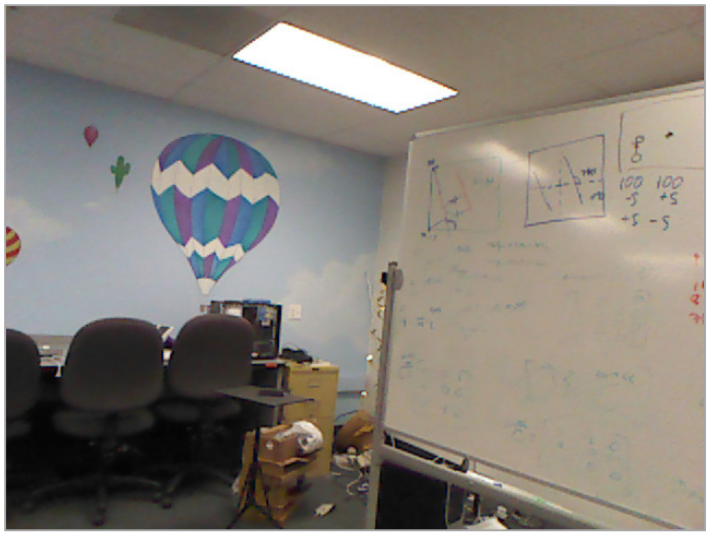
These are the images used to create the 3d model in the first place: fewer, higher-fidelity.

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

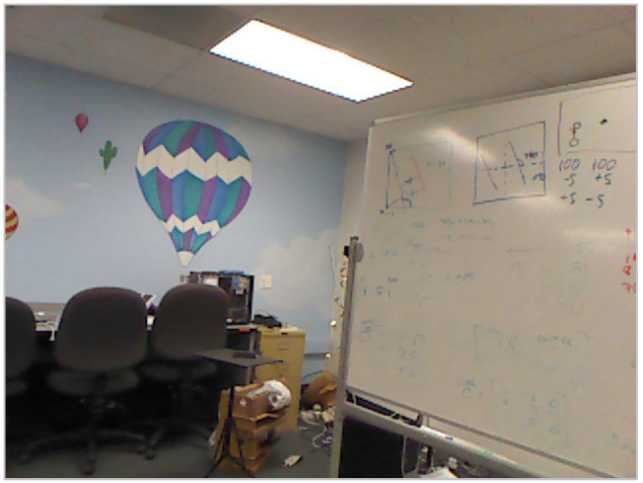
vs. original images

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?



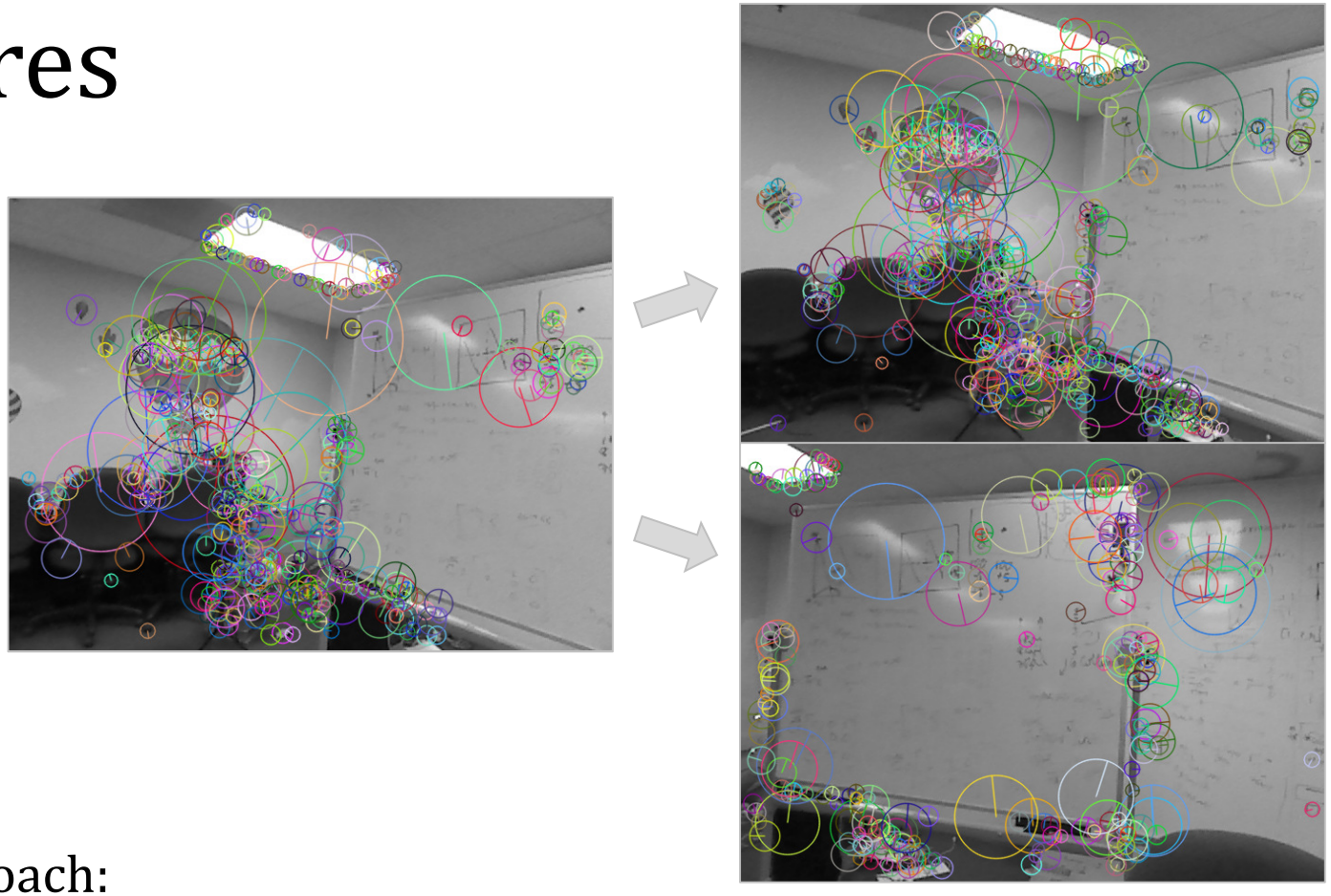
new image

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:

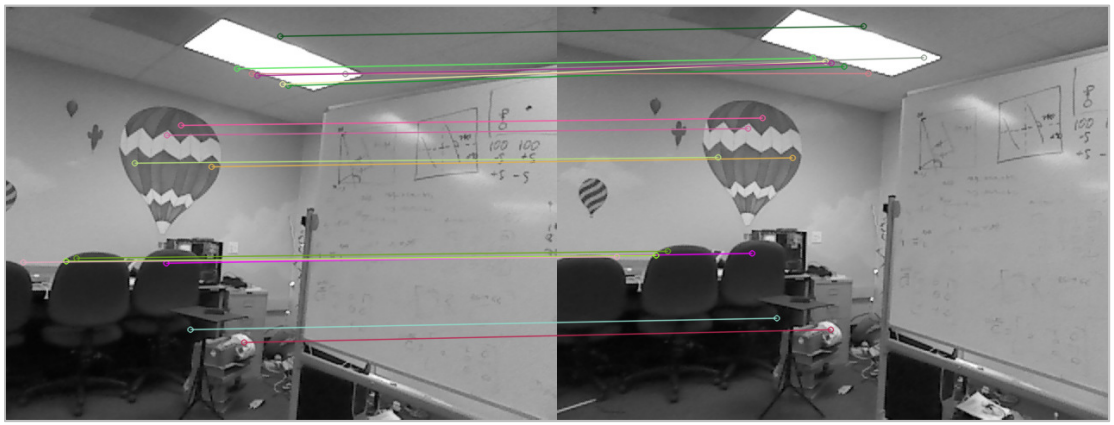
- Find features in each (SIFT)
- Consider all matches (FLANN)
- **Ensure bidirectional constraints**
- Consistency filtering (homography)
- Best-matching by visual similarity metric
- Tiebreaking by pixel distances

Keep only *two-way* best matches

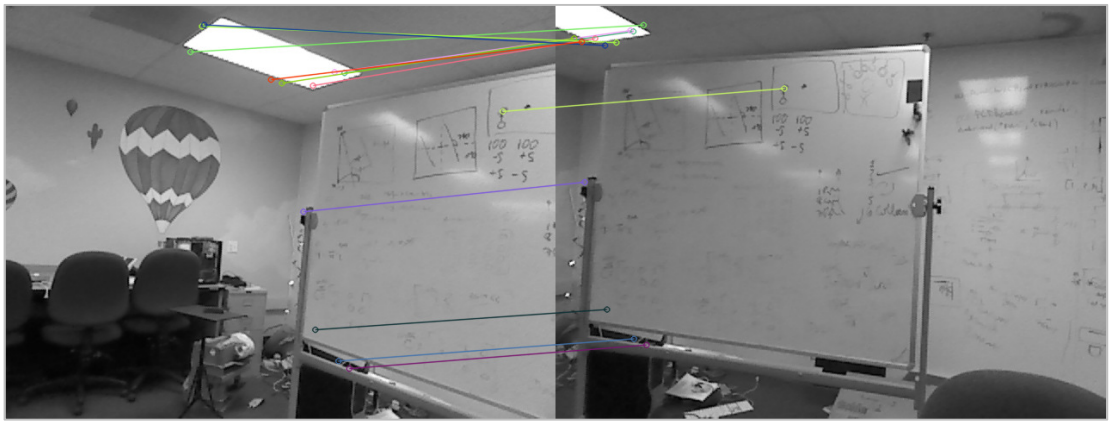
Our approach:

- 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



1st-best match

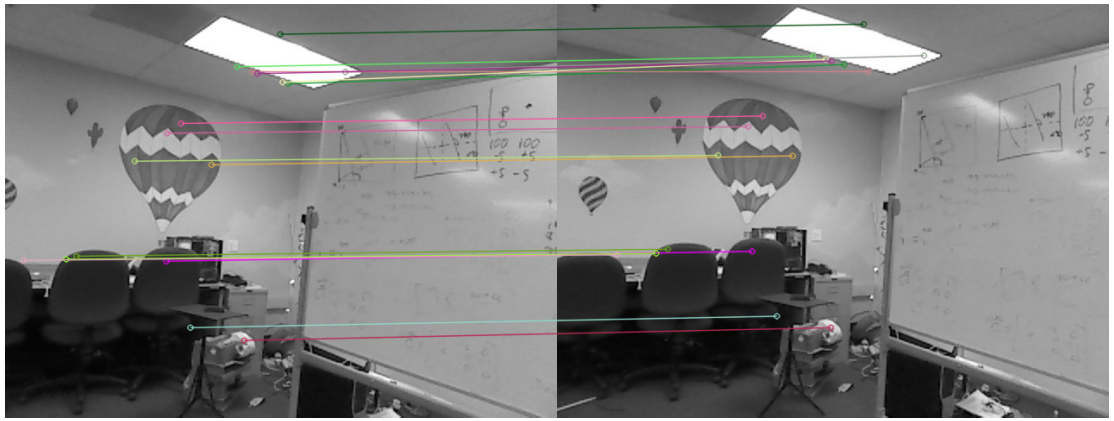


10th-best match

Our approach:

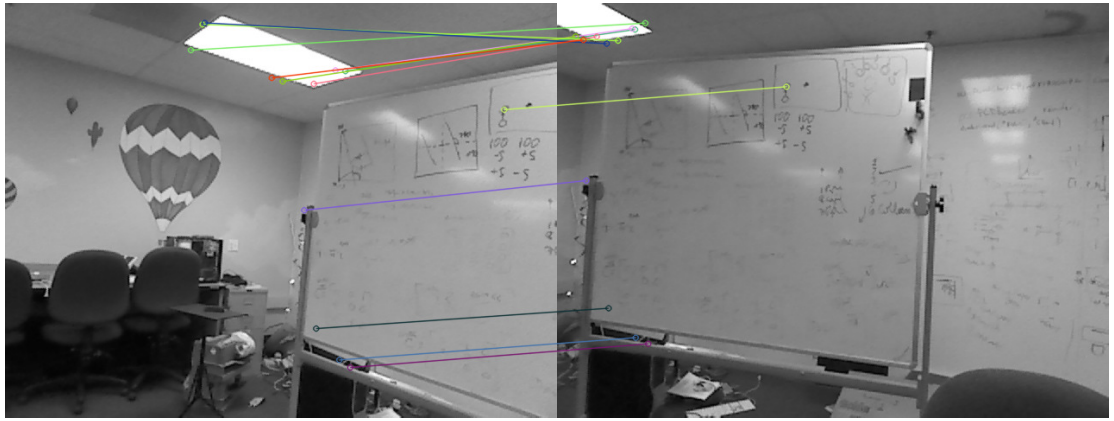
- Find features in each (SIFT)
- Consider all matches (FLANN)
- Ensure bidirectional constraints
- Consistency filtering (via homography)
- **Best-matching by visual similarity metric**
- Tiebreaking by pixel distances

Visual “closeness”



1st-best match

Average feature distance **~ 40 px**



10th-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

pcl About News Blog Downloads

Documentation

How to extract NARF keypoint from a range image

This tutorial demonstrates how to extract NARF key points from a range image. The executable enables us to load a point cloud from disc (or create it if not gi
both in an image and a 3D viewer.

The code

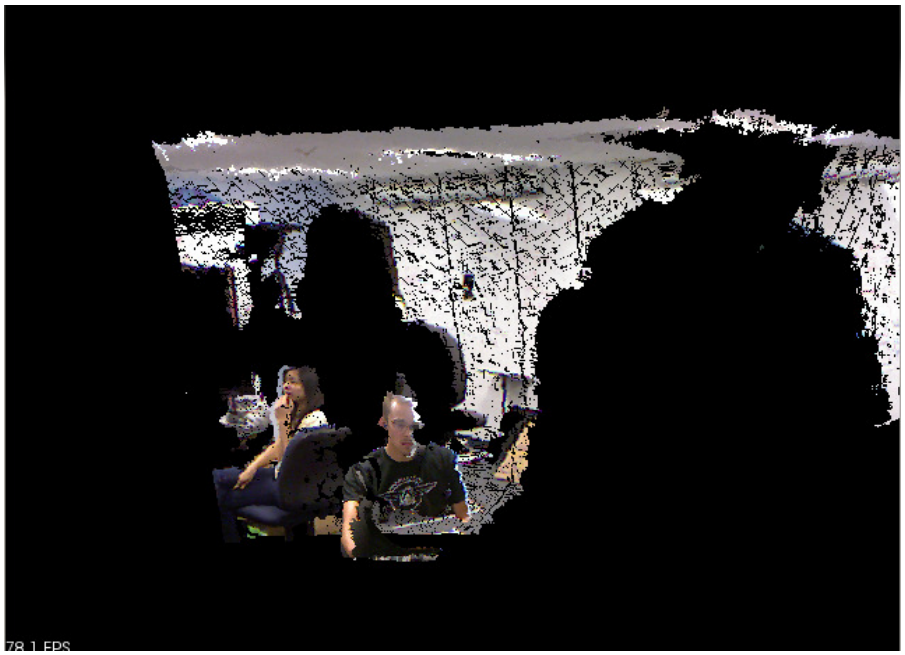
First, create a file called, let's say, `narf_keypoint_extraction.cpp` in your favorite editor, and place the following code inside it:

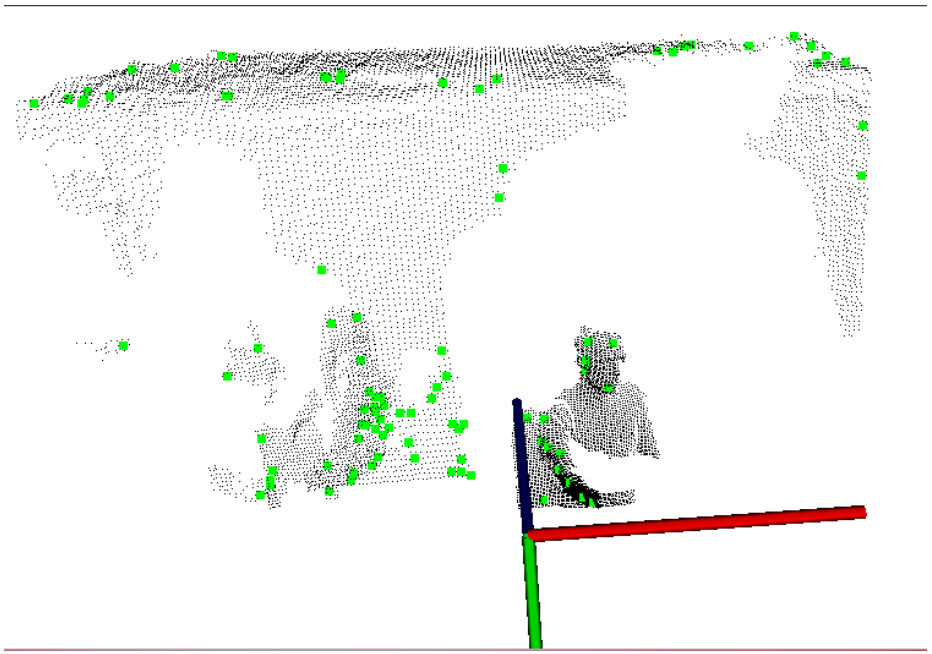
```
/* \author Bastian Steder */

#include <iostream>

#include <boost/thread/thread.hpp>
#include <pcl/range_image/range_image.h>
#include <pcl/io/pcd_io.h>
#include <pcl/visualization/range_image_visualizer.h>
#include <pcl/visualization/pcl_visualizer.h>
#include <pcl/features/range_image_border_extractor.h>
#include <pcl/keypoints/narf_keypoint.h>
#include <pcl/console/parse.h>

typedef pcl::PointXYZ PointType;

// --------------------
// -----Parameters-----
// --------------------
float angular_resolution = 0.5f;
float support_size = 0.2f;
pcl::RangeImage::CoordinateFrame coordinate_frame = pcl::RangeImage::CAMERA_FRAME;
bool setUnseenToMaxRange = false;

// --------------
// -----Help-----
// --------------
void
printUsage (const char* progName)
{
  std::cout << "\n\nUsage: "<<progName<<" [options] <scene.pcd>\n\n"
            << "Options:\n"
            << "-------------------------------------------\n"
            << "-r <float>   angular resolution in degrees (default "<<angular_resolution<<")\n"
            << "-c <int>     coordinate frame (default "<< (int)coordinate_frame<<")\n"
            << "-m           Treat all unseen points as maximum range readings\n"
            << "-s <float>   support size for the interest points (diameter of the used sphere - "
            <<                                                    "default "<<support_size<<")\n"
```

- 3D-Reasoning
- Easy point cloud access

NARFs!

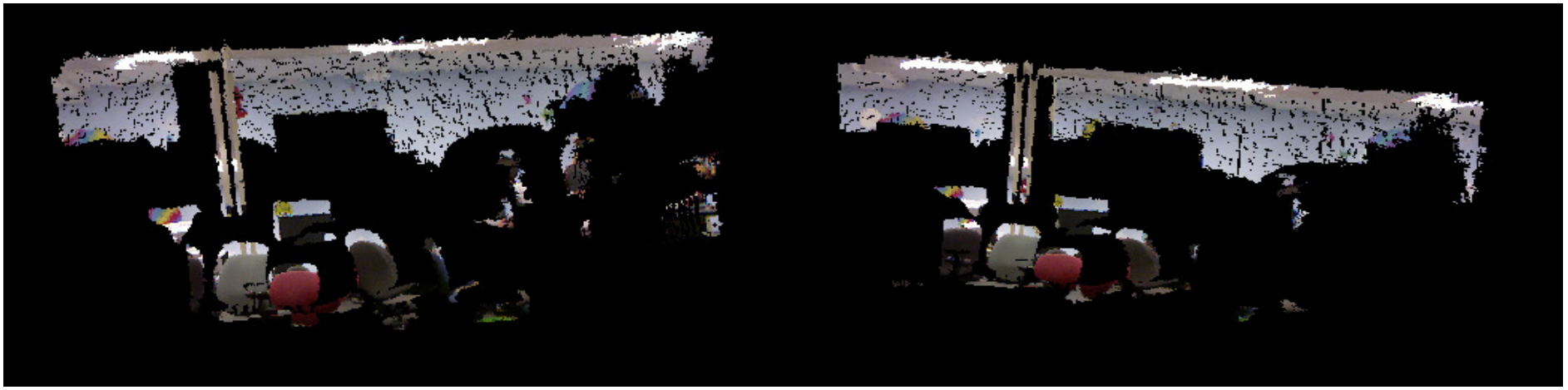
3d features: *Normal Aligned Radial Features*





- 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

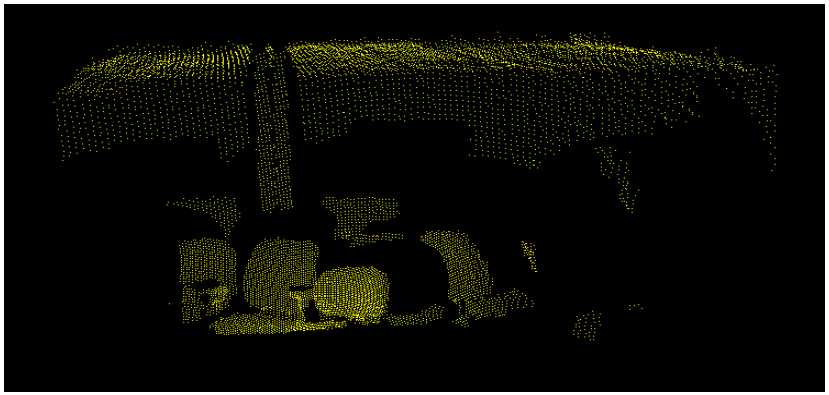


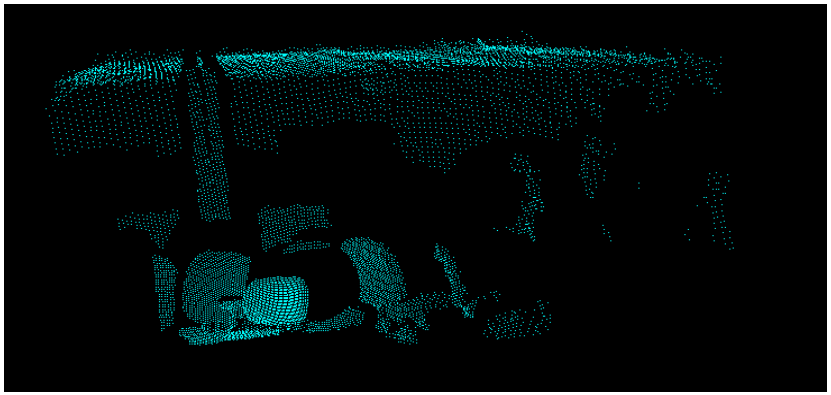
- Take two point clouds
- Find NARFs
- Match features and find transform
- Make transform and combine

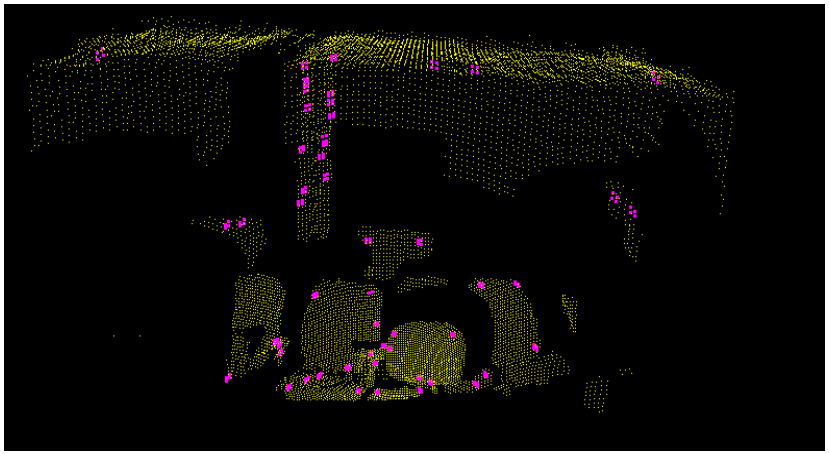
naive merging

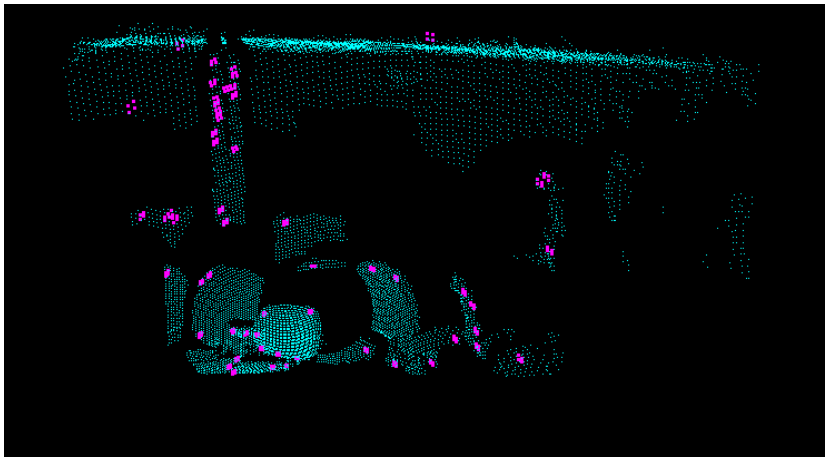
3d SLAM

Simultaneous Localization And Mapping





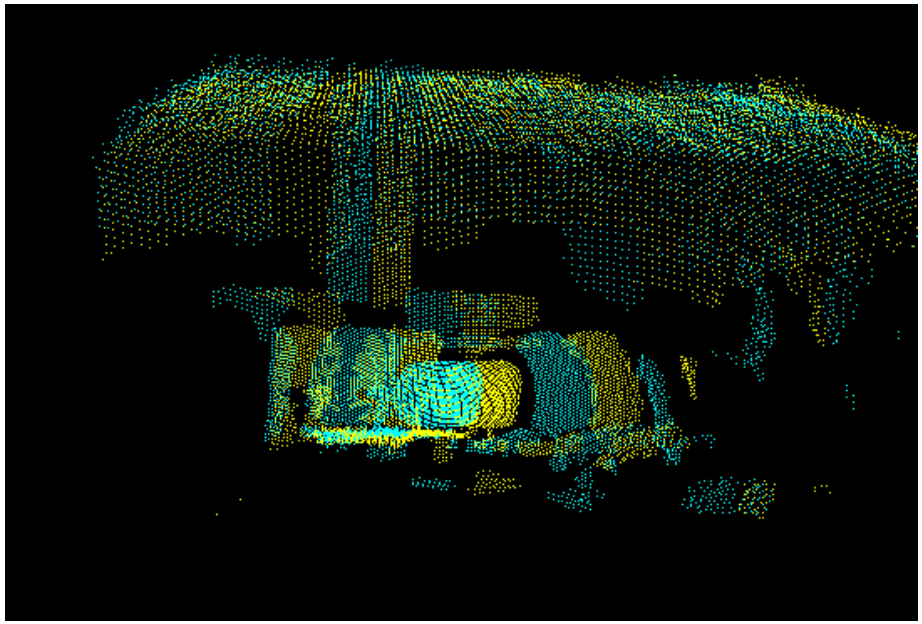


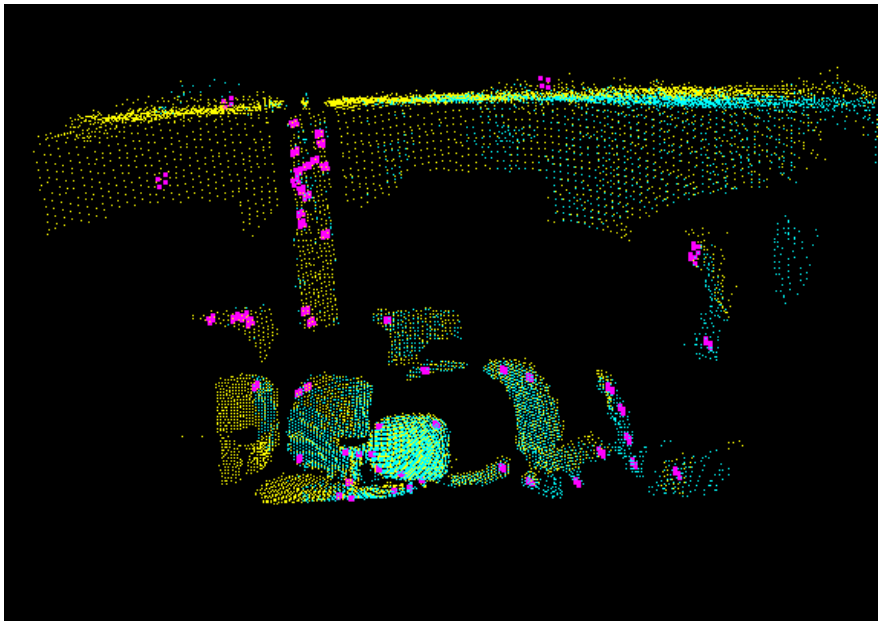


- Take two point clouds
- Find NARFs
- Match features and find transform
- Make transform and combine

3d SLAM

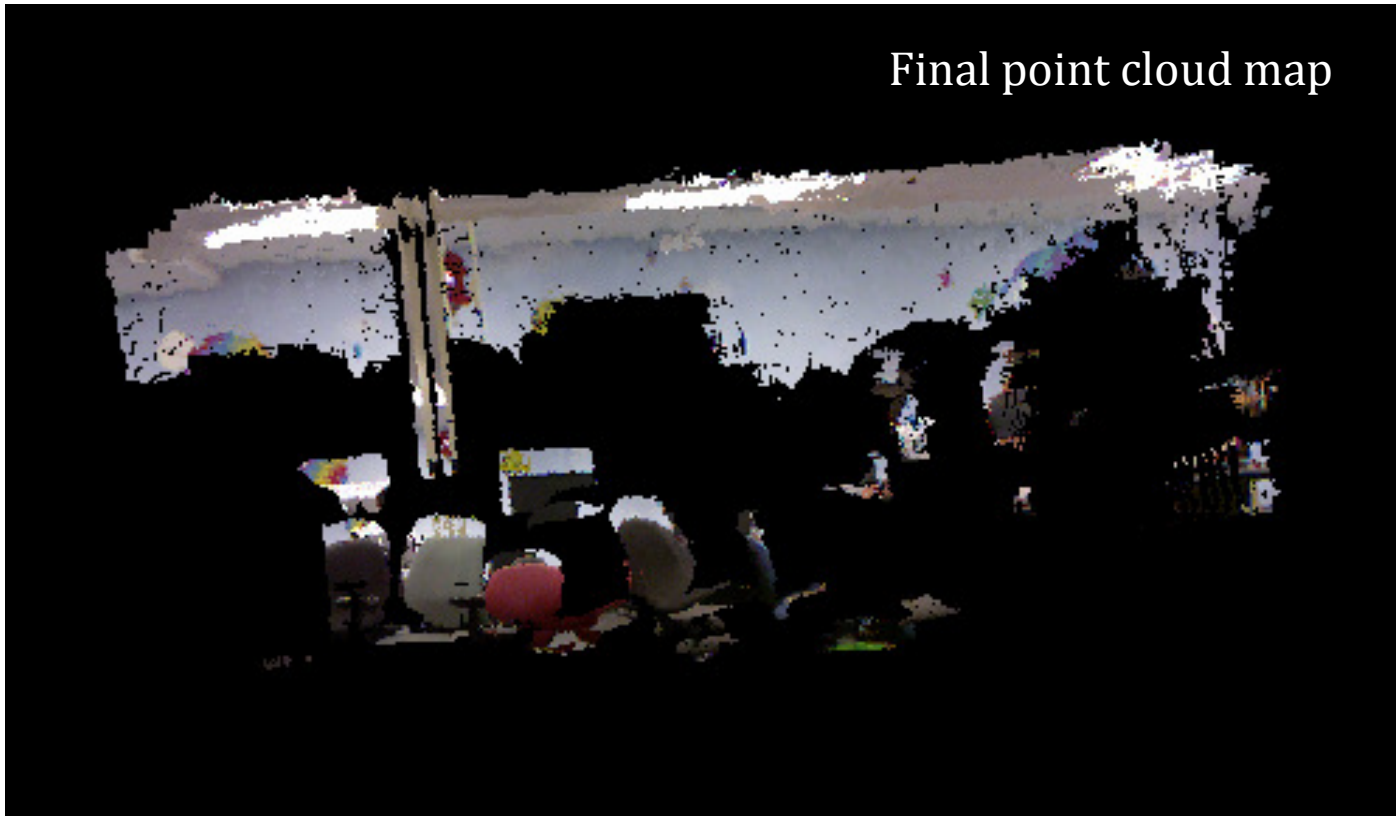
Simultaneous Localization And Mapping





- 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


DAD, HOW COME OLD PHOTOGRAPHS ARE ALWAYS BLACK AND WHITE? DIDN'T THEY HAVE COLOR FILM BACK THEN?
SURE THEY DID. IN FACT, THOSE OLD PHOTOGRAPHS ARE IN COLOR. IT'S JUST THE WORLD WAS BLACK AND WHITE THEN.
REALLY?
YEP. THE WORLD DIDN'T TURN COLOR UNTIL SOMETIME IN THE 1930s, AND IT WAS PRETTY GRAINY COLOR FOR A WHILE, TOO.
THAT'S REALLY WEIRD.
WELL, TRUTH IS STRANGER THAN FICTION.
BUT THEN WHY ARE OLD PAINTINGS IN COLOR?! IF THE WORLD WAS BLACK AND WHITE, WOULDN'T ARTISTS HAVE PAINTED IT THAT WAY?
NOT NECESSARILY. A LOT OF GREAT ARTISTS WERE INSANE.

Title


Calvin and Hobbes
by Watterson
WOW, HONEY, YOU'RE MISSING A BEAUTIFUL SUNSET OUT HERE!
I'LL COUNT TO 10, AND THEN... POW!
DAD, HOW COME OLD PHOTOGRAPHS ARE ALWAYS BLACK AND WHITE? DIDN'T THEY HAVE COLOR FILM BACK THEN?
SURE THEY DID. IN FACT, THOSE OLD PHOTOGRAPHS ARE IN COLOR. IT'S JUST THE WORLD WAS BLACK AND WHITE THEN.
REALLY?
YEP. THE WORLD DIDN'T TURN COLOR UNTIL SOMETIME IN THE 1930s, AND IT WAS PRETTY GRAINY COLOR FOR A WHILE, TOO.
THAT'S REALLY WEIRD.
WELL, TRUTH IS STRANGER THAN FICTION.
BUT THEN WHY ARE OLD PAINTINGS IN COLOR?! IF THE WORLD WAS BLACK AND WHITE, WOULDN'T ARTISTS HAVE PAINTED IT THAT WAY?
NOT NECESSARILY. A LOT OF GREAT ARTISTS WERE INSANE.
BUT...BUT HOW COULD THEY HAVE PAINTED IN COLOR ANYWAY? WOULDN'T THEIR PAINTS HAVE BEEN SHADES OF GRAY BACK THEN?
OF COURSE, BUT THEY TURNED COLORS LIKE EVERYTHING ELSE DID IN THE '30s.
SO WHY DIDN'T OLD BLACK AND WHITE PHOTOS TURN COLOR TOO?
BECAUSE THEY WERE COLOR PICTURES OF BLACK AND WHITE, REMEMBER?
THE WORLD IS A COMPLICATED PLACE, HOBBES.
WHENEVER IT SEEMS THAT WAY, I TAKE A NAP IN A TREE AND WAIT FOR DINNER.

Accessing these models

plane

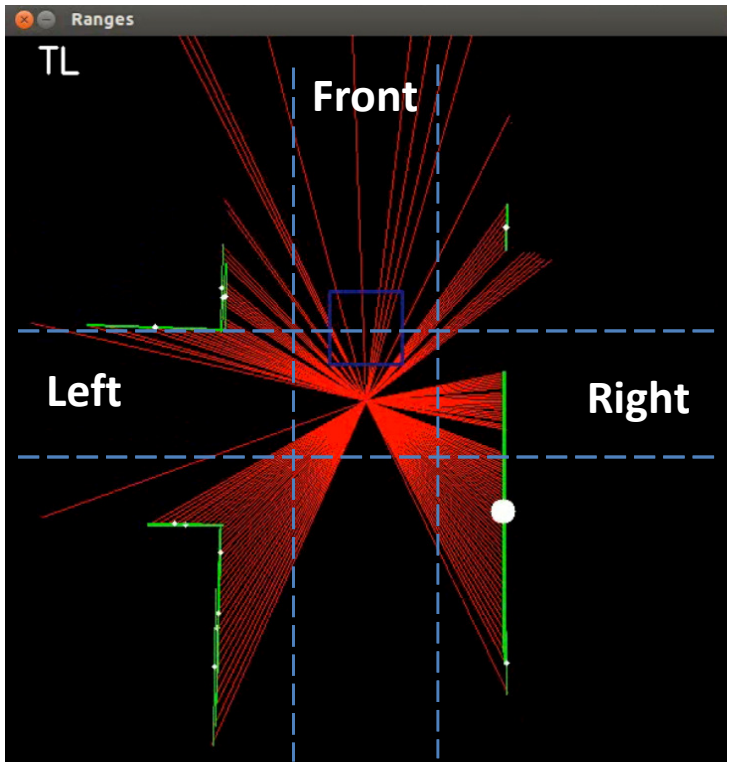
other (sphere, cylinder, wavy,…)
K.M.

allows us to define the depth at each pixel

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

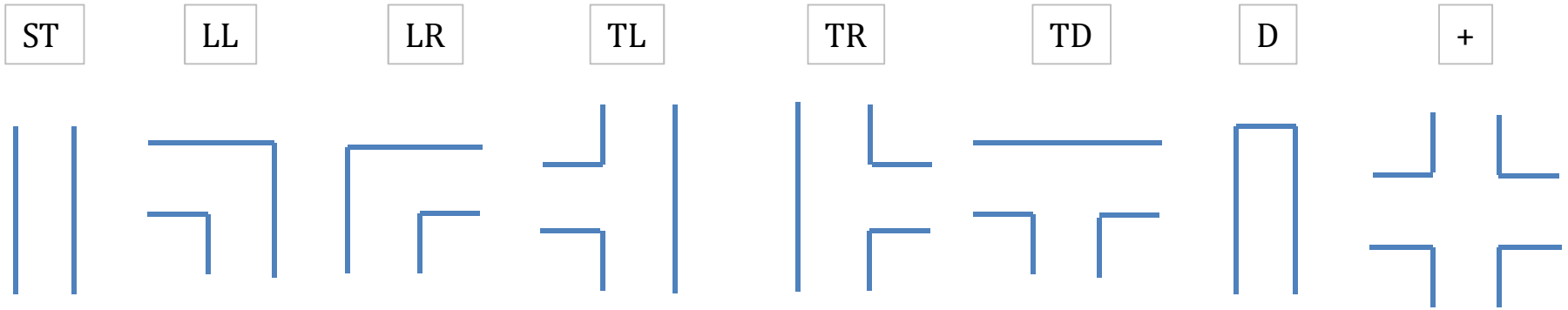
Ideally, a camera-only system could localize – *and reason* – within the 3d model

Intersection recognition

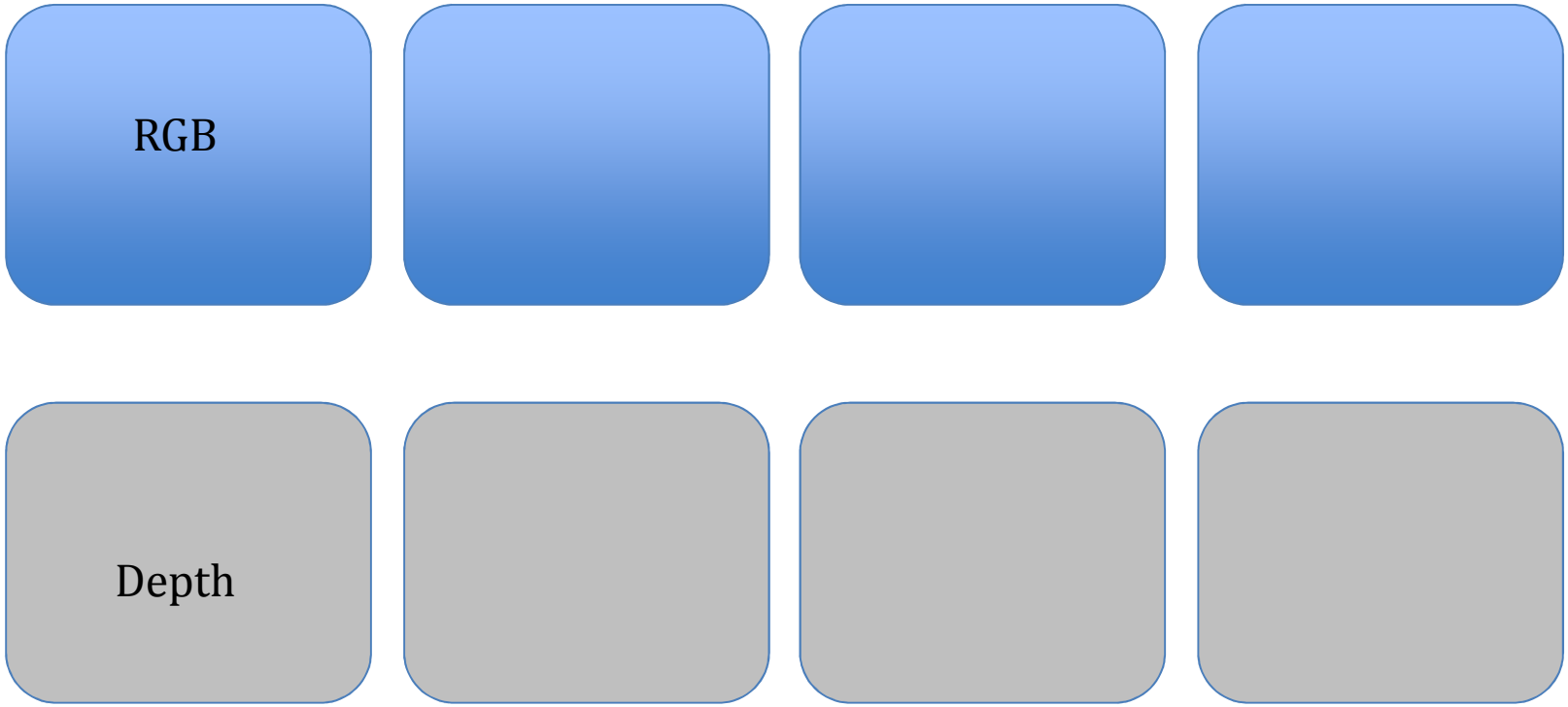


For each scan:

- 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**.

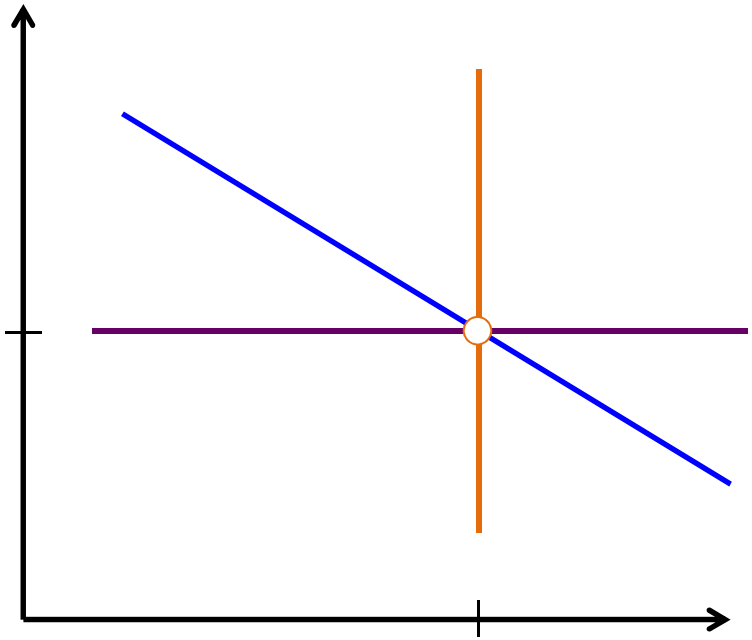


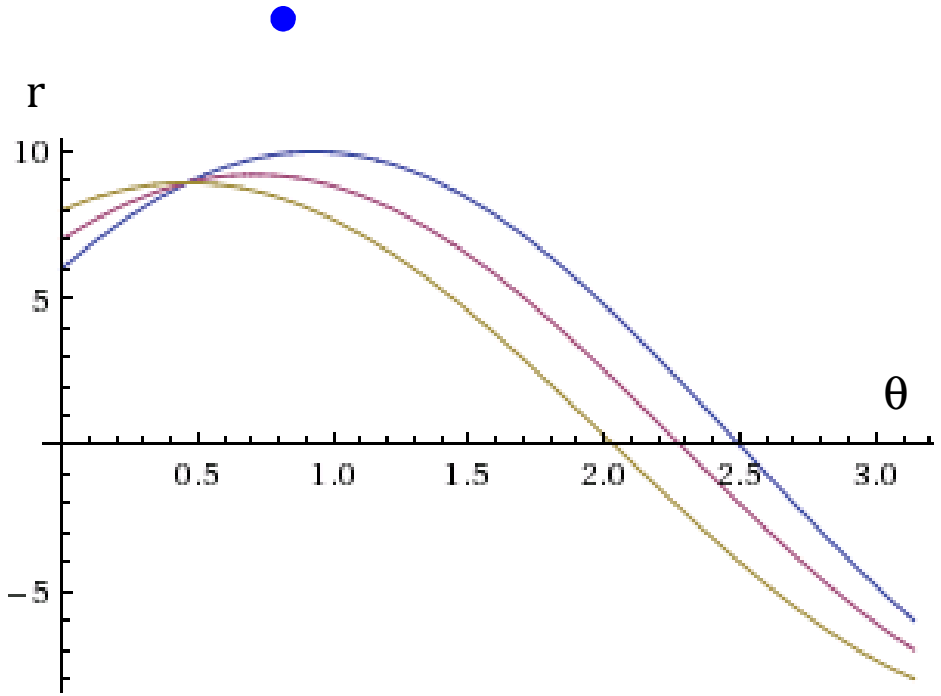
Accessing 3d models



splicing into the code…

Hough transform





Consider ***one point*** of a laser scan

These are ***all*** of the lines through it

http://gmarty.github.io/hough-transform-js/

http://liquify.eu/flash/HoughTransform

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*?

map of current knowledge…

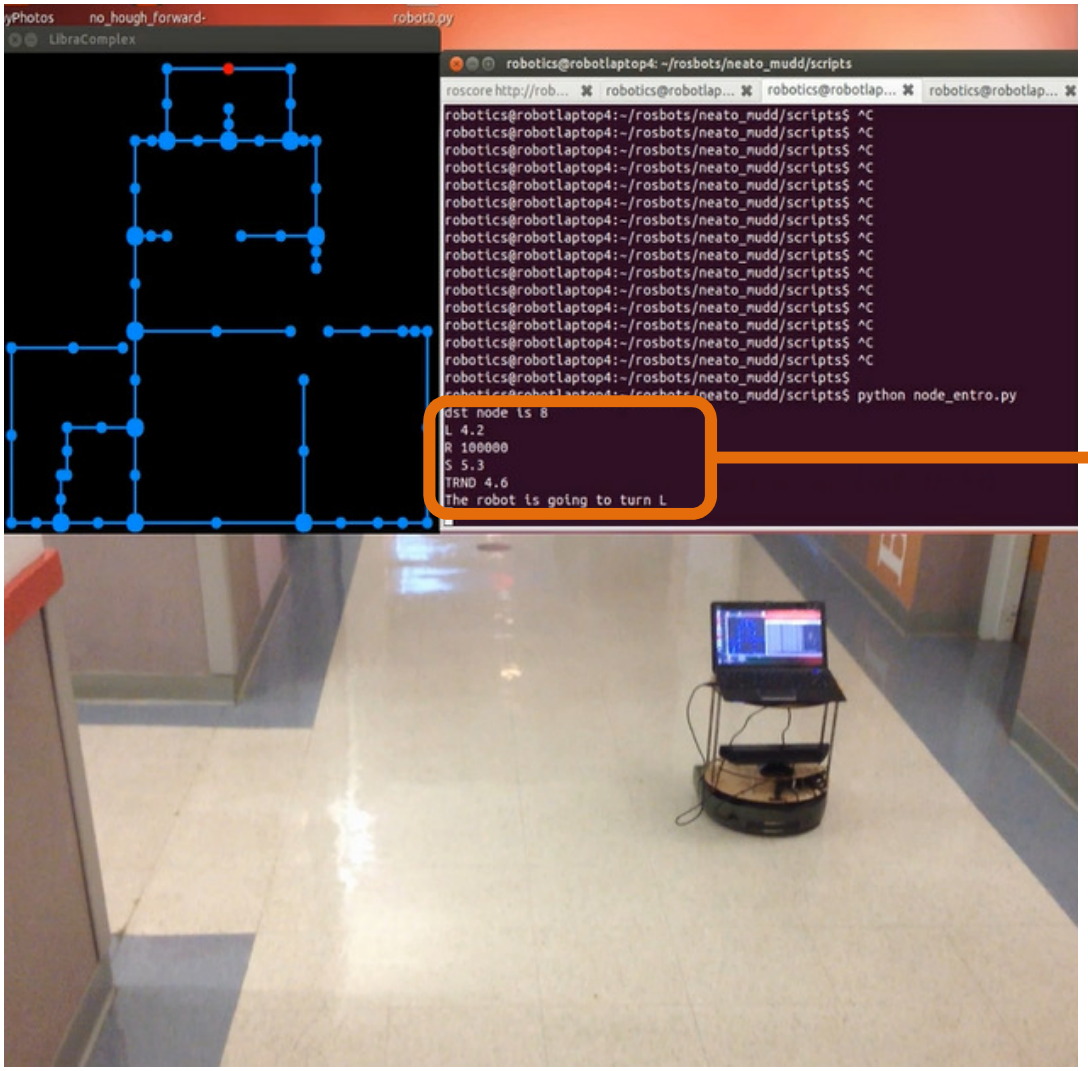
Possible results if the robot heads straight — bits

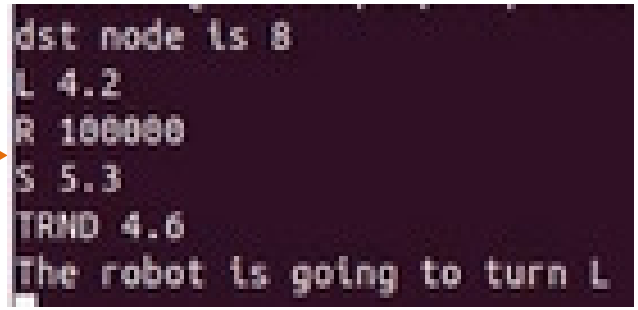
Possible results if the robot turns the corner — bits

Possible results if the robot reverses course — bits

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

Entropy-based localization





entro_short.mp4

Could we be **bolder**?

Leap before you look.

~ W. H. Auden

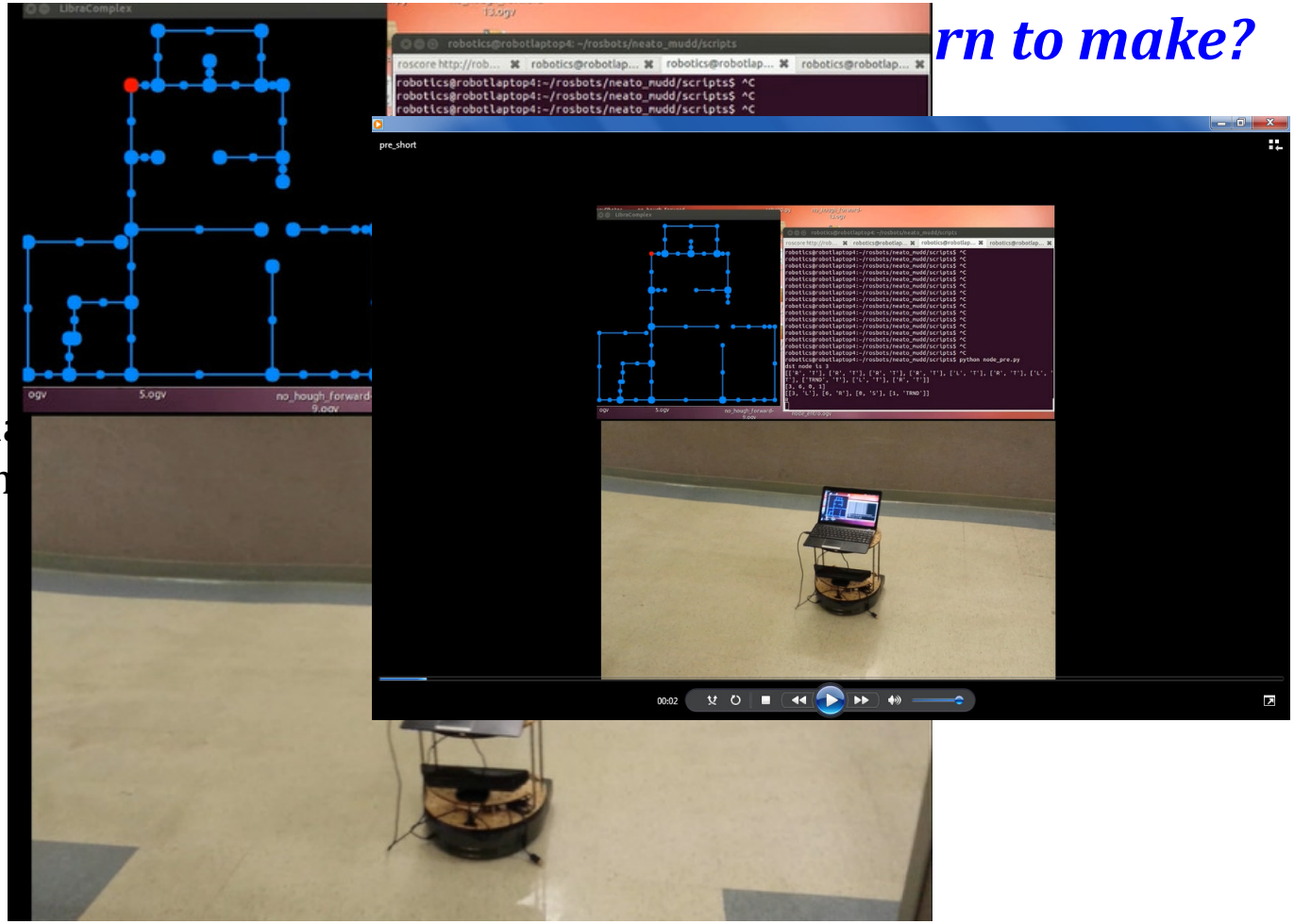
Just do it.

~ Nike

… perhaps if robots felt more contemplative.

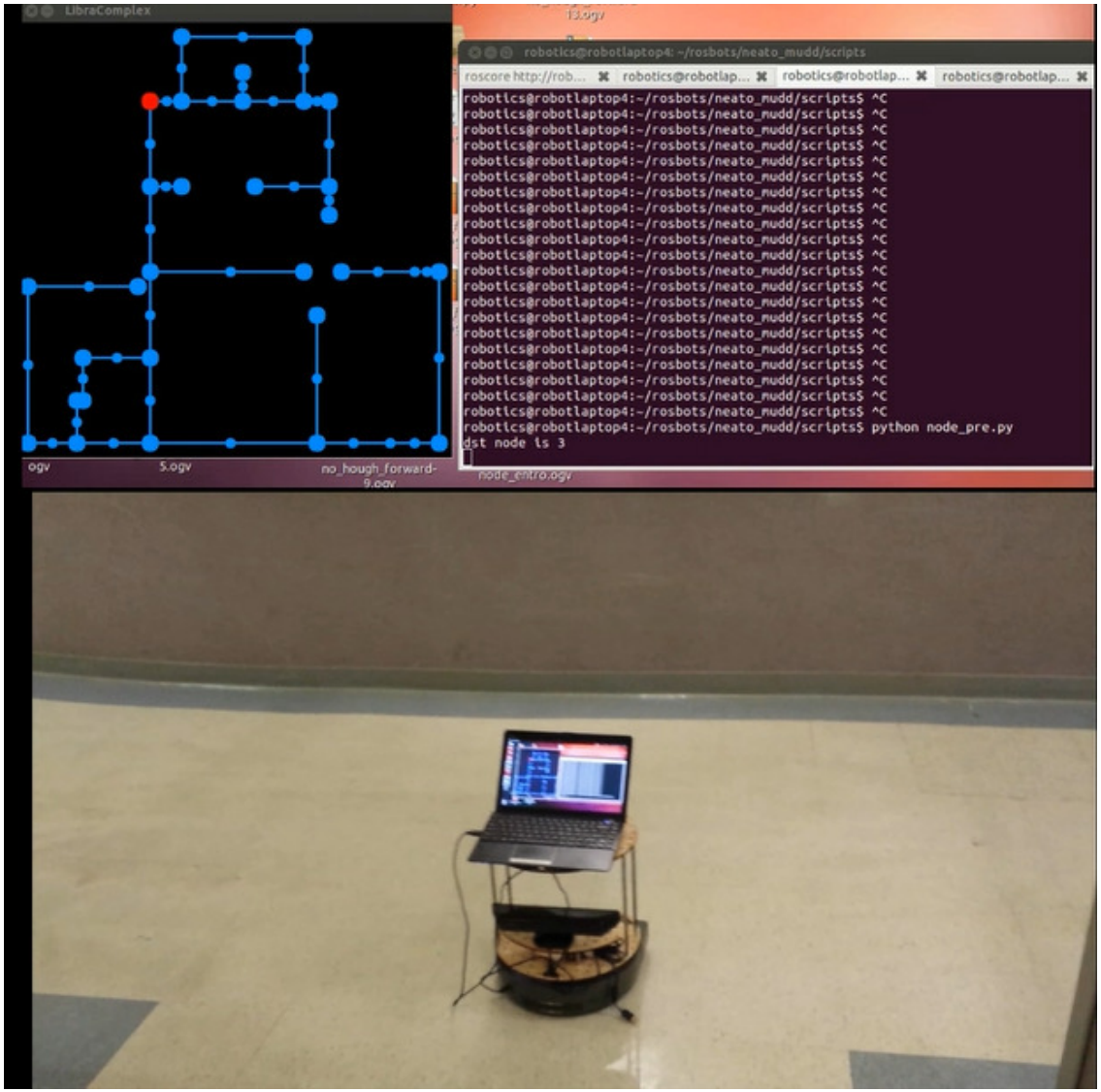
Predictive path planning

rn to make?



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

Predictive path planning



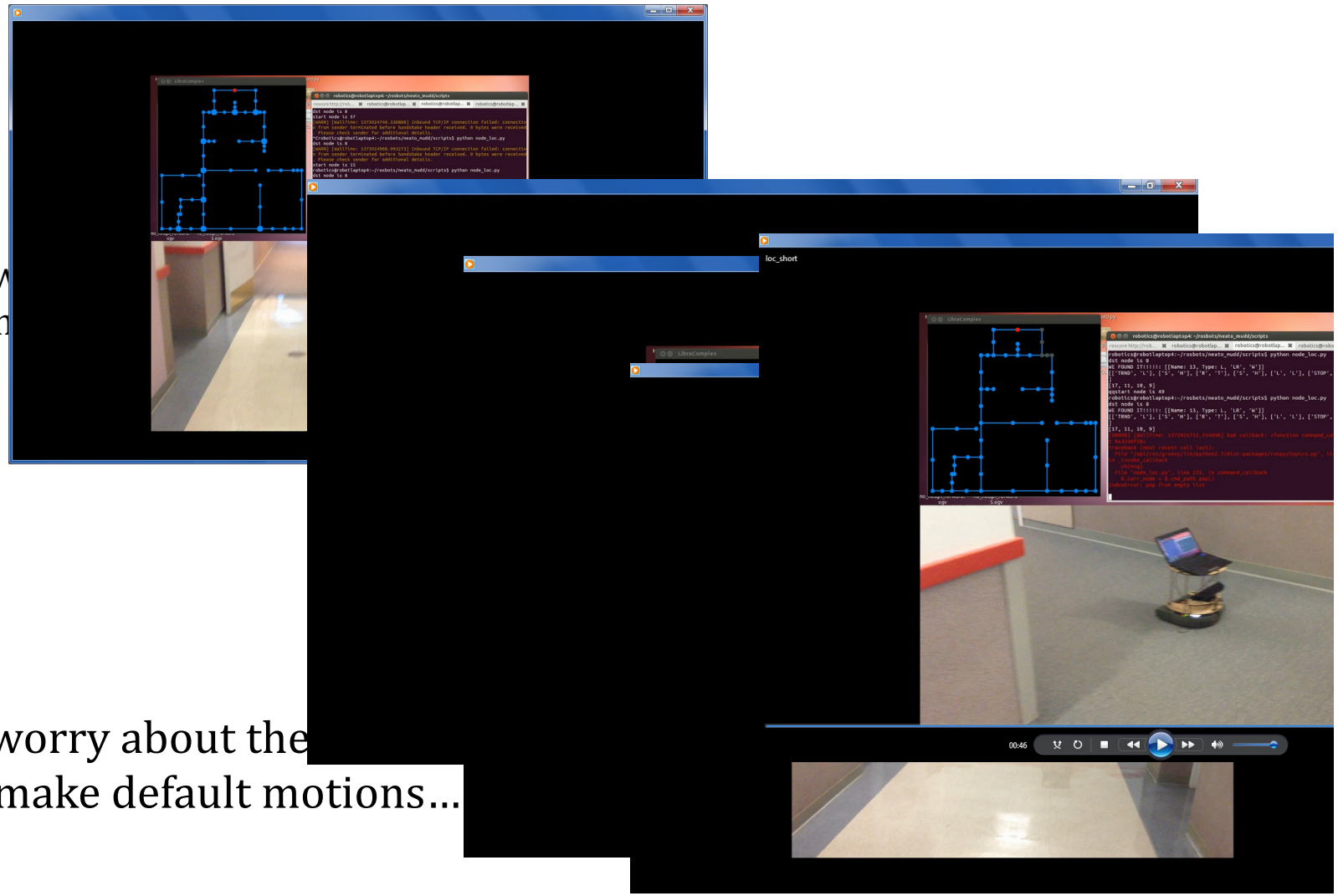
pre_short.mp4

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

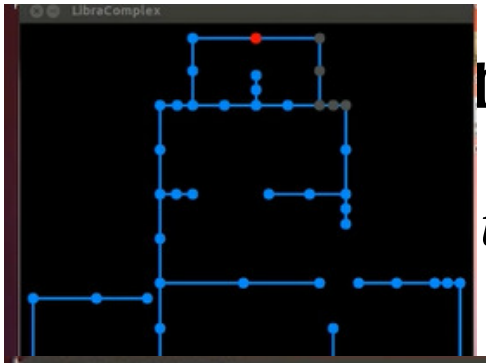
Localization

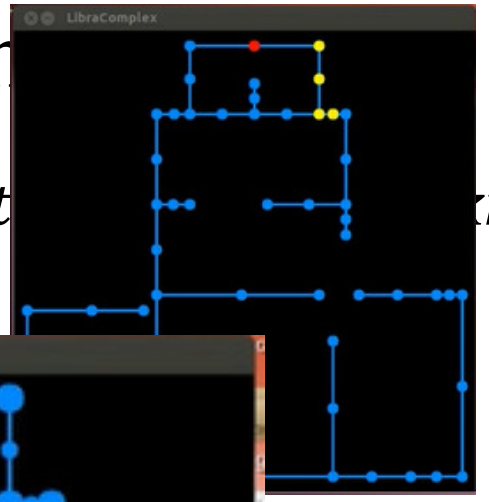
What if the robot doesn't know its starting pose?

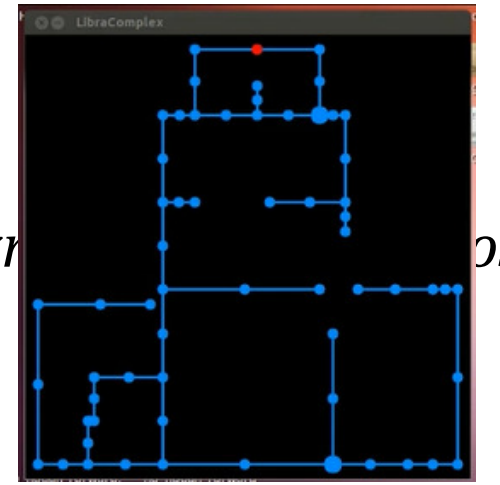
map w
and th

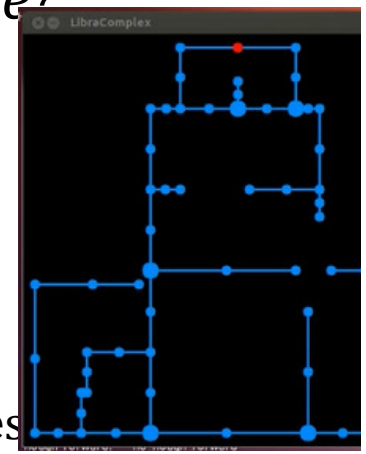


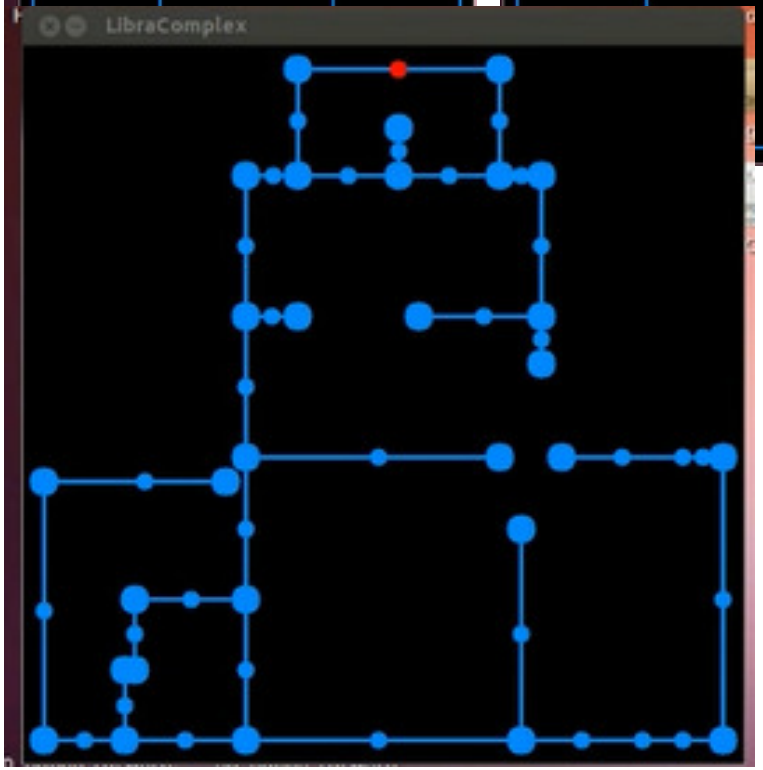
Don't worry about the
Instead, make default motions…





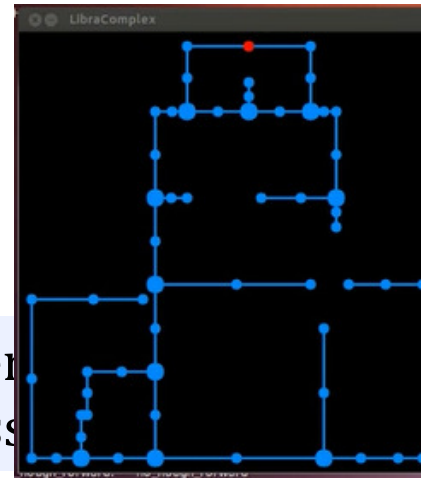






map with fewer nodes
large and the goal node in
red

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

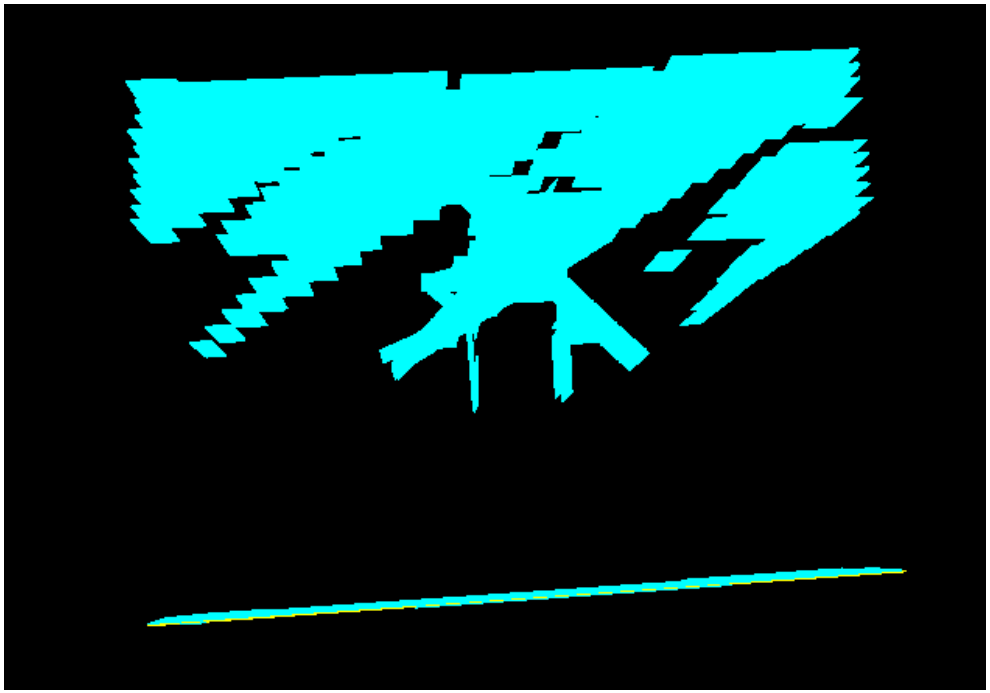


Don't worry about the goal.
Instead, make default motions…

With each obser
cull the robot's poss

Too many walls…

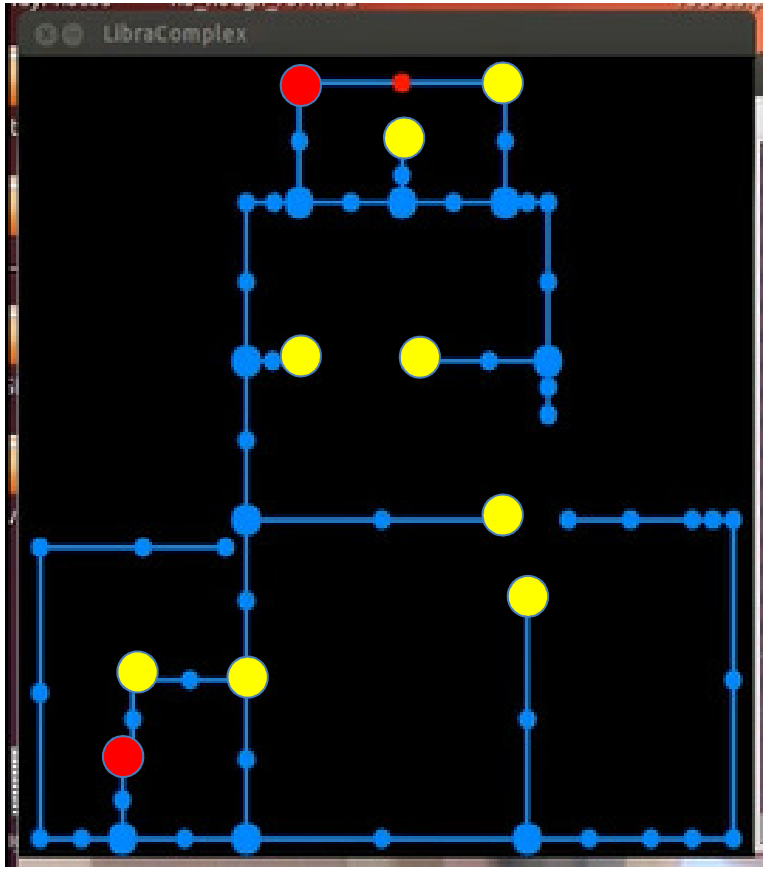
Uninitialized images ~
aargh!



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

Can we be more *deliberate*?

Suppose the robot goes ***left***...

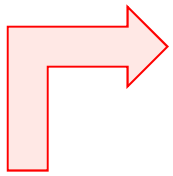


2 LR

2 LL

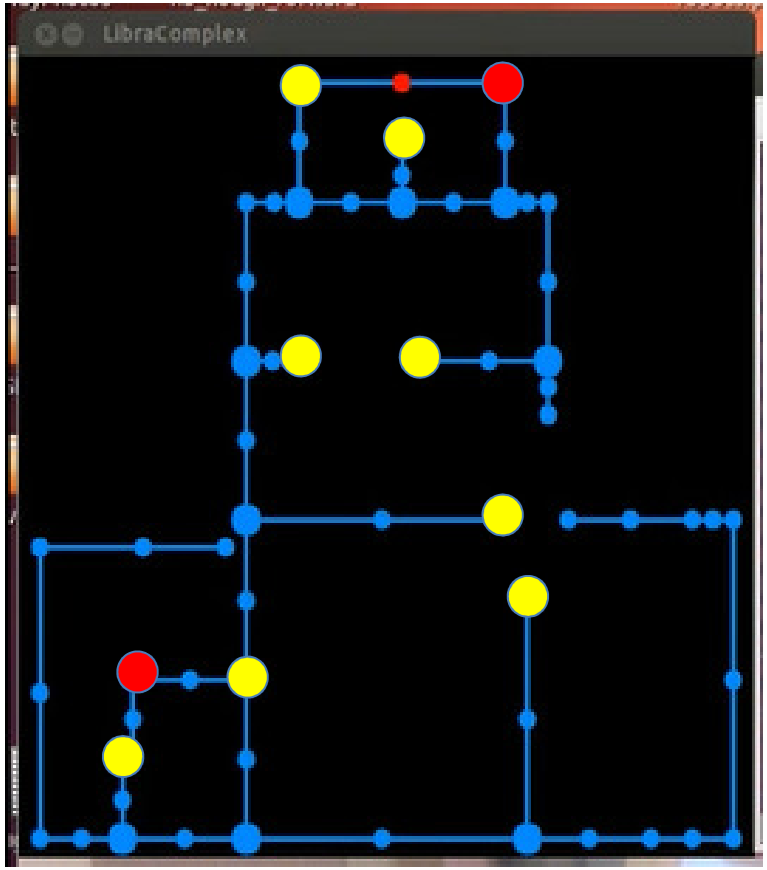
5 DE

1 TL



Can we be more *deliberate*?

Suppose the robot goes ***left***...

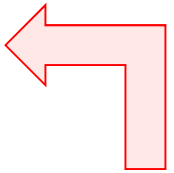


2 LR

2 LL

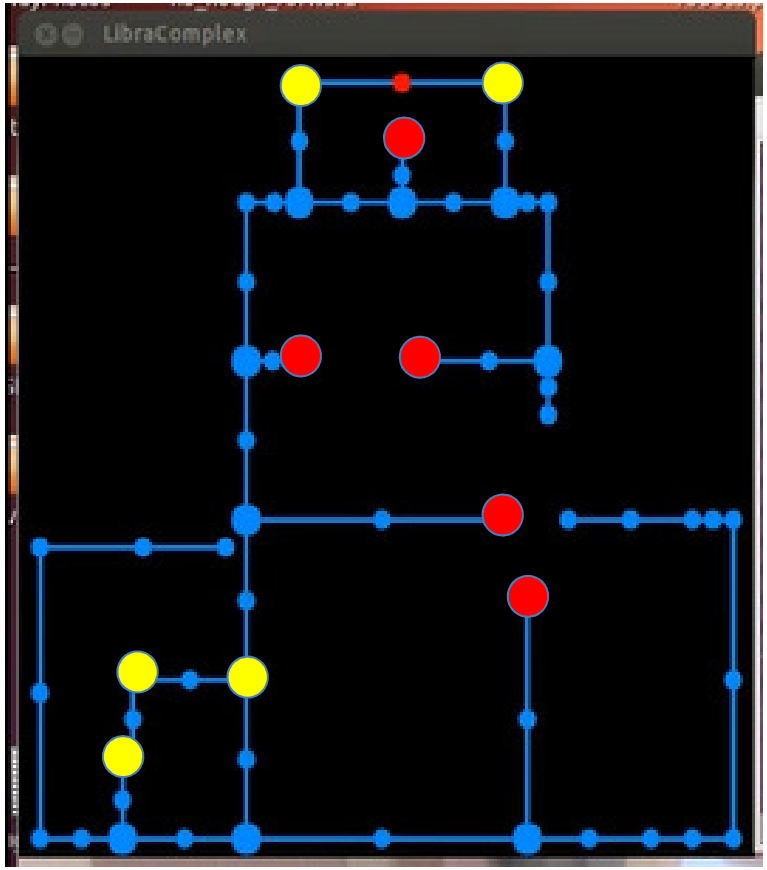
5 DE

1 TL



Can we be more *deliberate*?

Suppose the robot goes ***left***...



2 LR

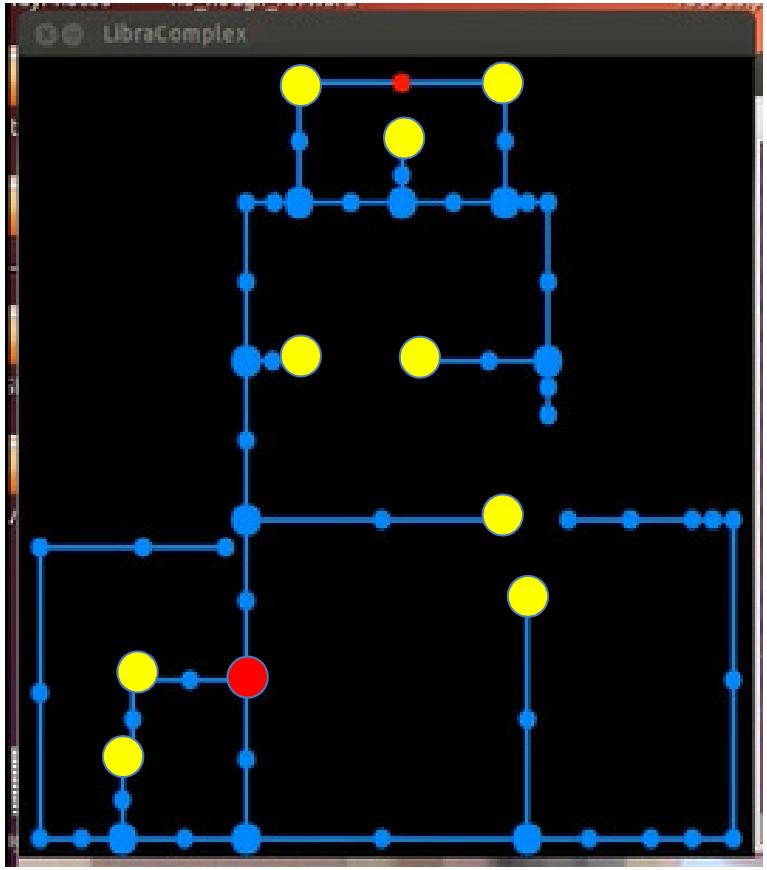
2 LL

5 DE

1 TL

Can we be more *deliberate*?

Suppose the robot goes ***left***...

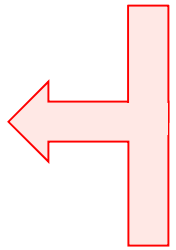


2 LR

2 LL

5 DE

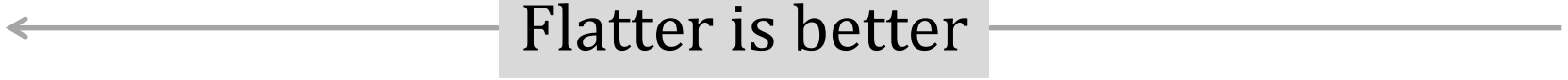
1 TL



Entropy

Suppose there were 8 possibilities to begin with…

2 A	4 A	8 A
2 B	4 B	0 B
2 C	0 C	0 C
2 D	0 D	0 D
2 possibilities will remain!	4 possibilities will remain!	8 possibilities will remain!



Entropy

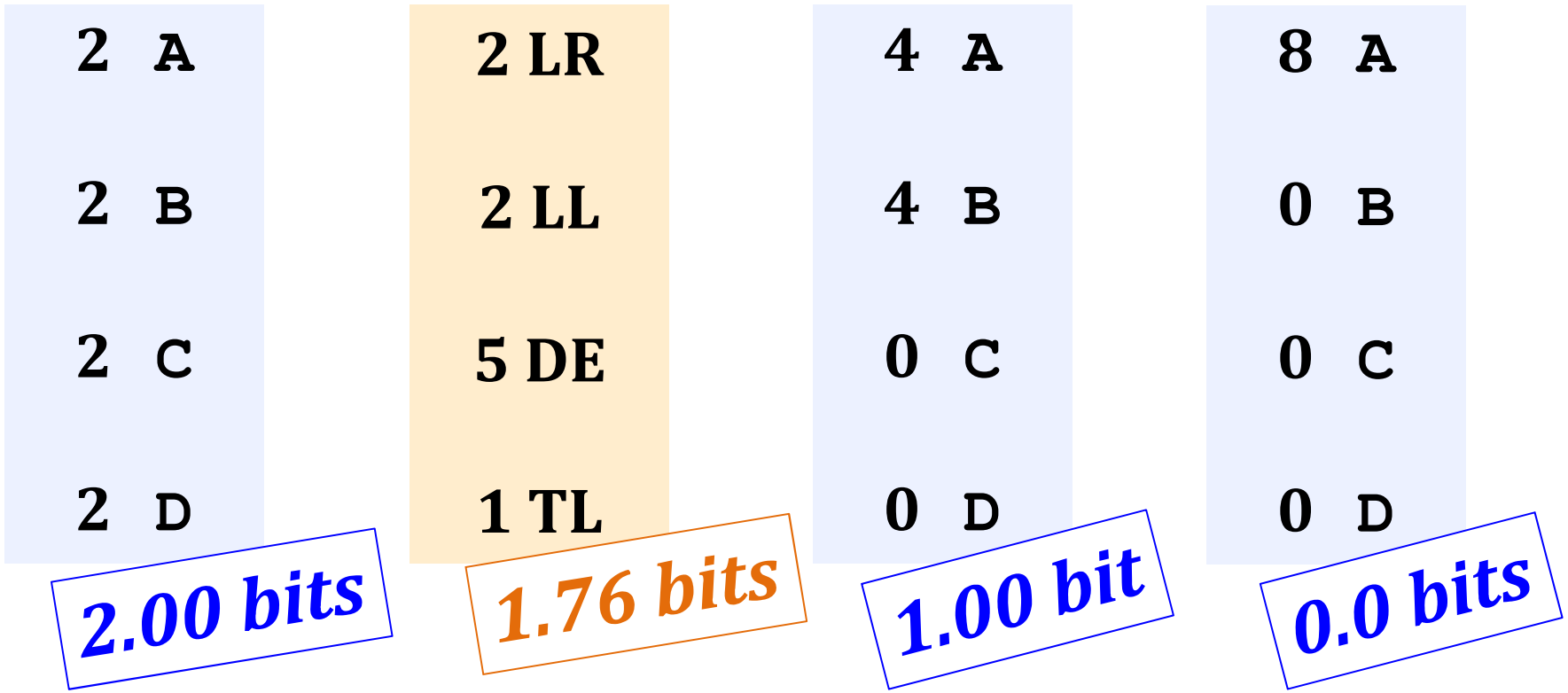
Suppose there were 8 possibilities to begin with…

2 A	4 A	8 A
2 B	4 B	0 B
2 C	0 C	0 C
2 D	0 D	0 D
2.00 bits	**1.00 bit**	**0.0 bits**

Entropy ~ ***expected information***.

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

Entropy


2 A
2 B
2 C
2 D
2.00 bits
2 LR
2 LL
5 DE
1 TL
1.76 bits
4 A
4 B
0 C
0 D
1.00 bit
8 A
0 B
0 C
0 D
0.0 bits
Flatter is better