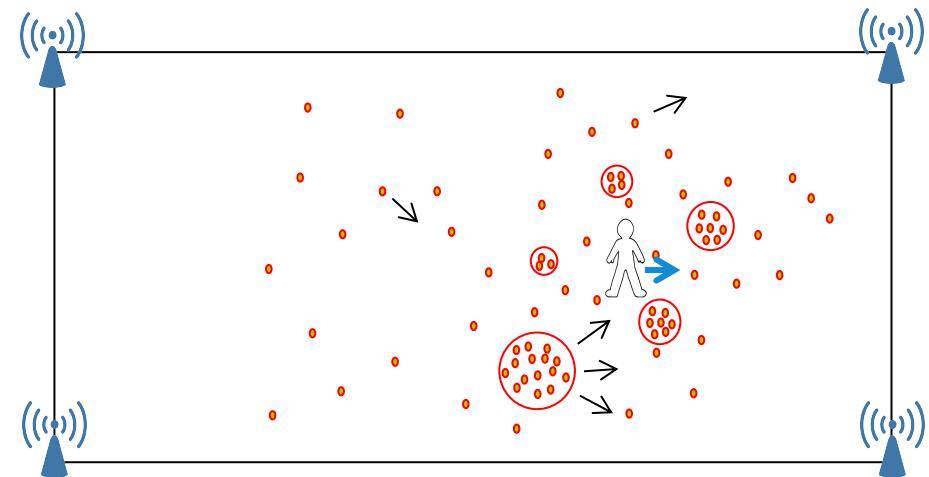


Lecture 05

SmartPhone Sensing

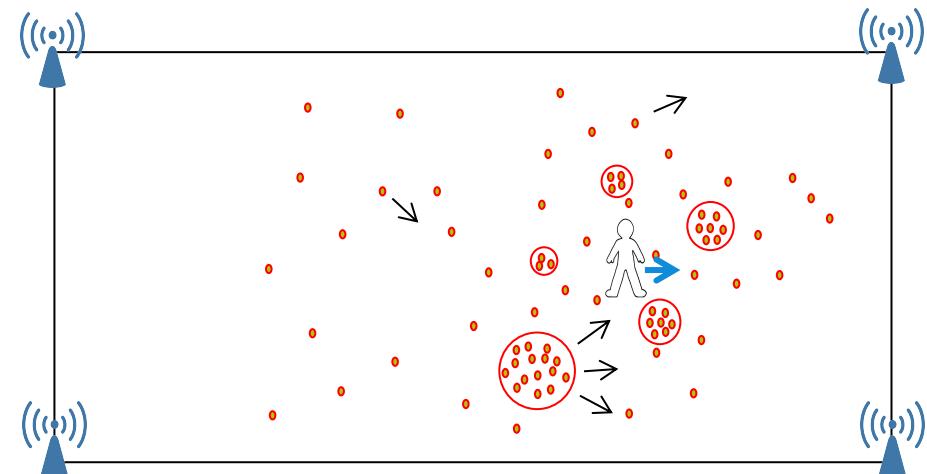
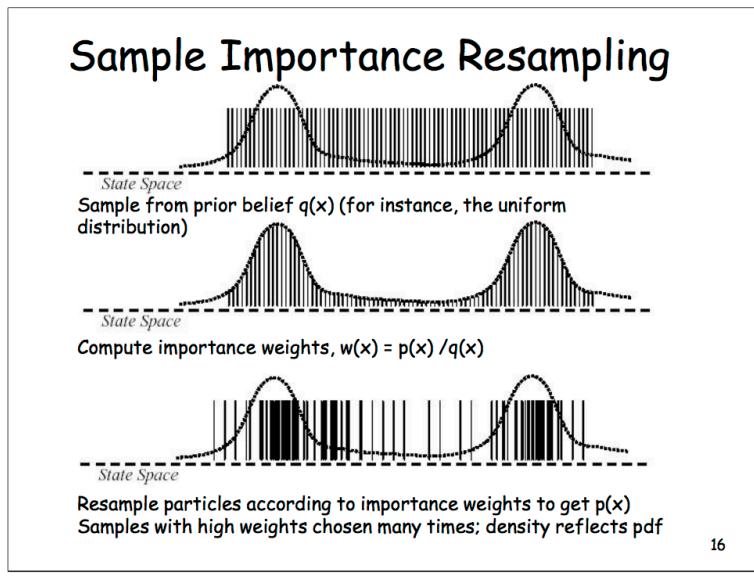
Why do I need to resample?

- Why not just keep weights?

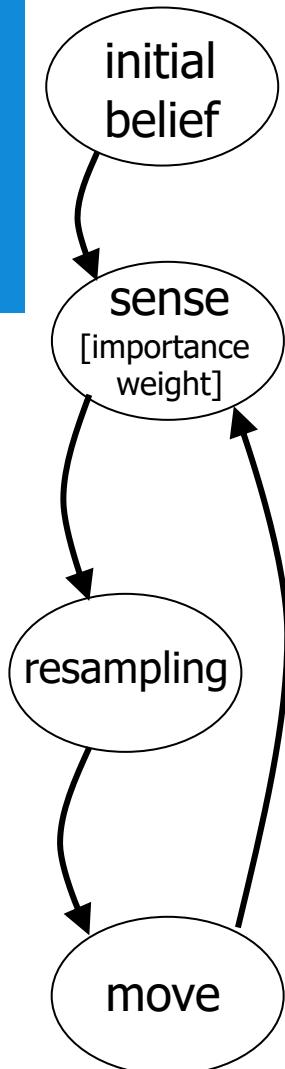


Why do I need to resample?

- Why not just keep weights?
 - Particle depletion
 - Density of particles should represent the pdf
 - Areas with high probability in posterior not represented well.
 - You won't be able to recover fast from errors



Back to the math, we need to add one more step: resampling



If you have no idea where you are, throw particles everywhere

$$\text{current pdf (posterior)} \quad \text{perception model (sense)} \quad \text{pdf from last time step (prior)}$$
$$p(X_k | Z_{1:k}) = \frac{p(Z_k | X_k) p(X_k | Z_{1:k-1})}{p(Z_k | Z_{1:k-1})} \text{ normalization}$$

take N new particles (with replacement) from new posterior

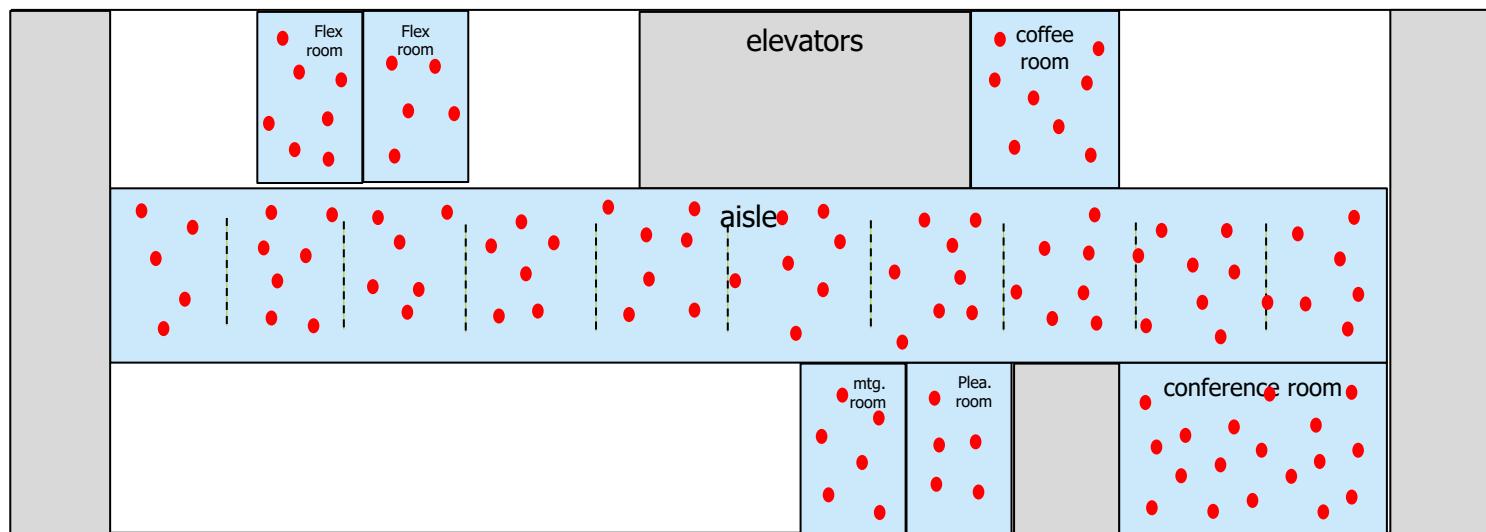
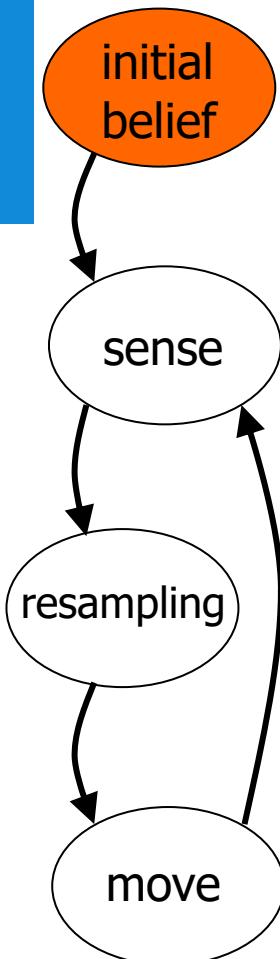
$$p(X_k | Z_{1:k})$$

$$\text{current pdf (posterior)} \quad \text{motion model (move)} \quad \text{pdf from last time step (prior)}$$

$$p(X_k | Z_{1:k-1}) = \int p(X_k | X_{k-1}) p(X_{k-1} | Z_{1:k-1}) dX_{k-1}$$

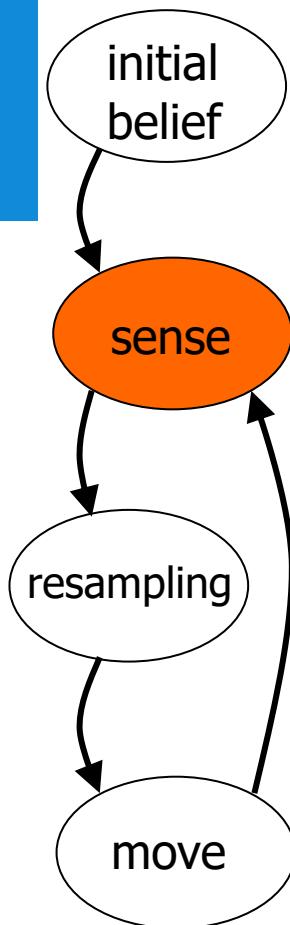
Paper: "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking", IEEE Trans. Signal Processing, 2002.

A toy example

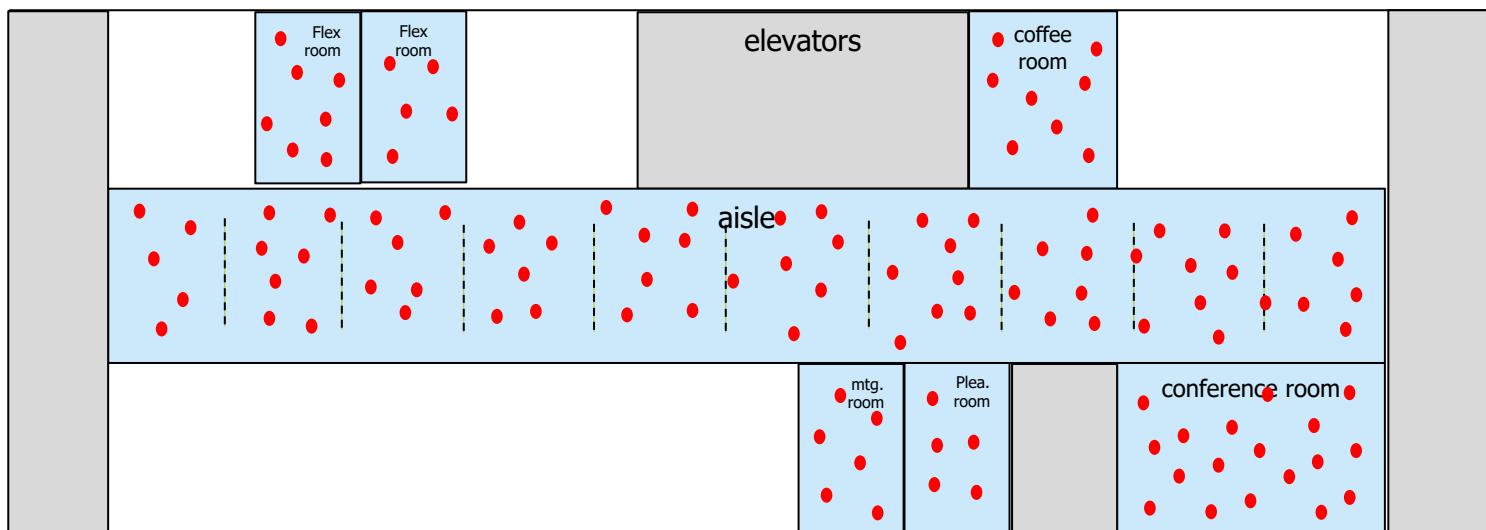


In your phone, you should have a table with (x,y) information for each particle. Put thousands of particles: 5k to 10k.

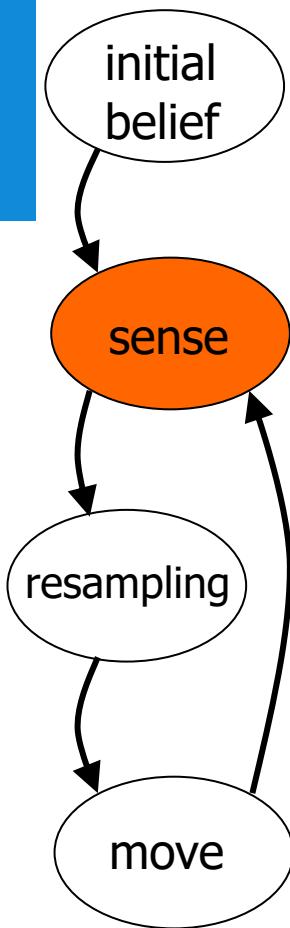
Sense



- We need to get the importance weight.
- What do we sense?
- How do we update the weight?



Sense



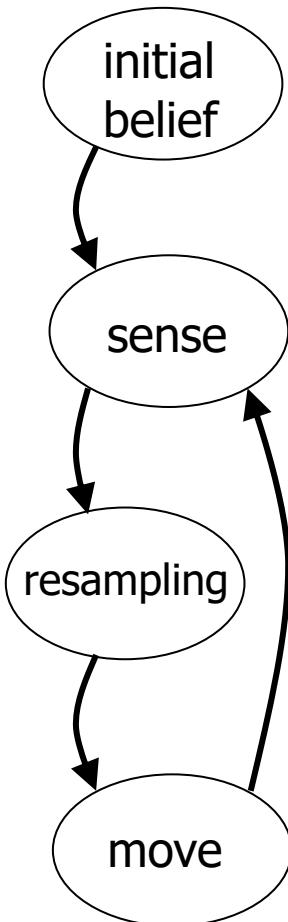
- We need to get the importance weight.
- What do we sense?
- How do we update the weight?

But hold on, we can't update much.
We don't know where we are, we have no
training data, no model, nothing.

We can't calculate weights or do resampling.

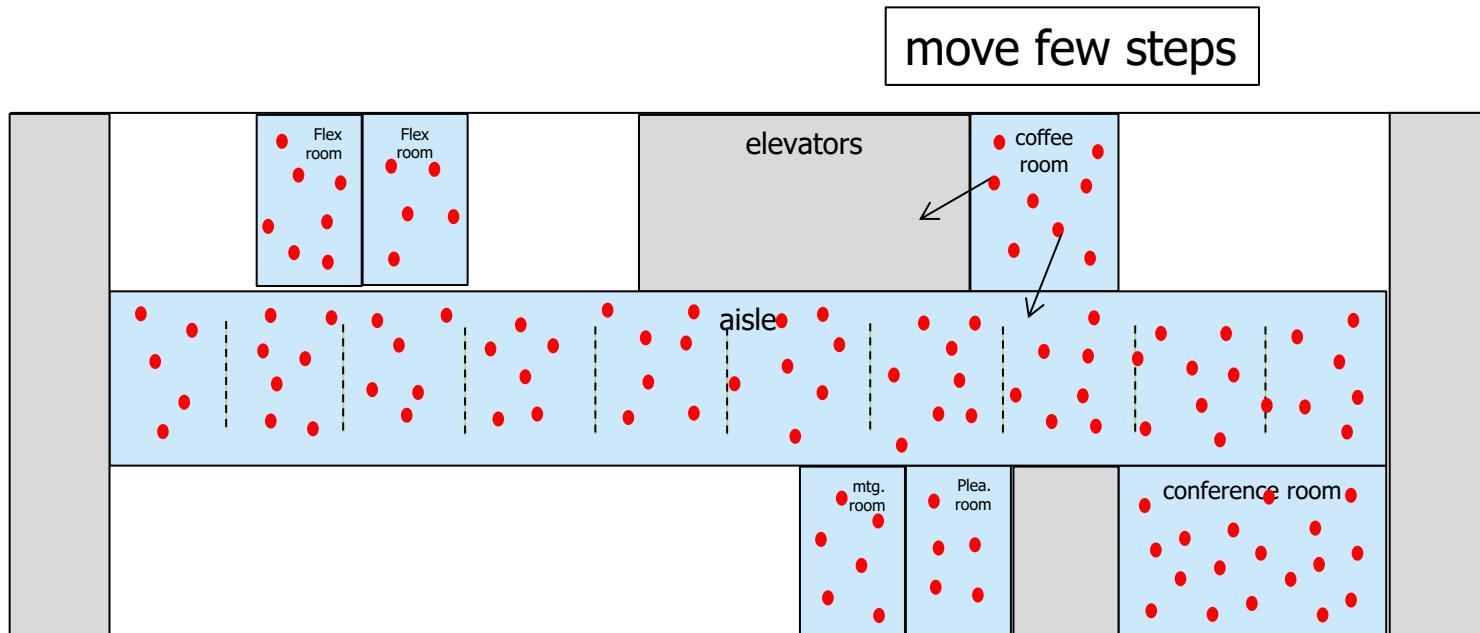
What do we do now? How do we update our prior?

Move



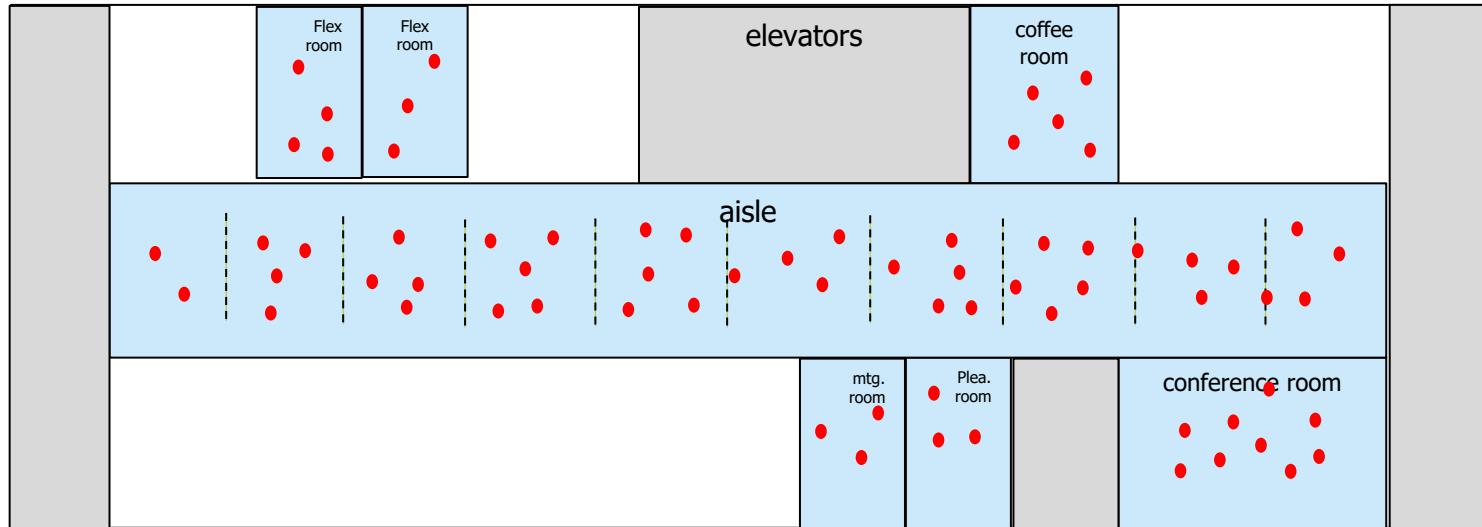
- Let's skip 'sense' and 'resampling' states ... and 'move'
- Assume you can count your steps
 - STEP_COUNTER in Android
- Initially with random direction
 - ROTATION_VECTOR in Android (to get direction)
 - Zee's contributions are in the motion model

Move & sense



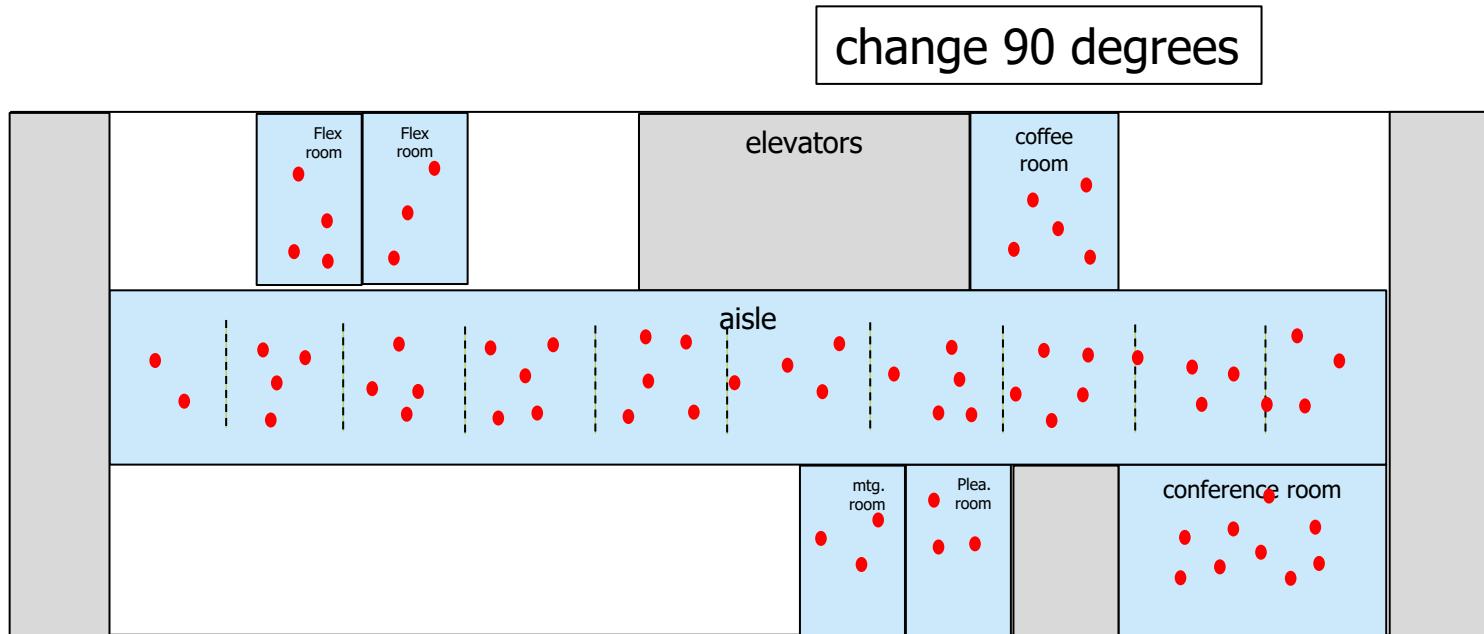
Now we can **sense** physical constraints.
If a path intersects a wall, the particle is eliminated.

Resample



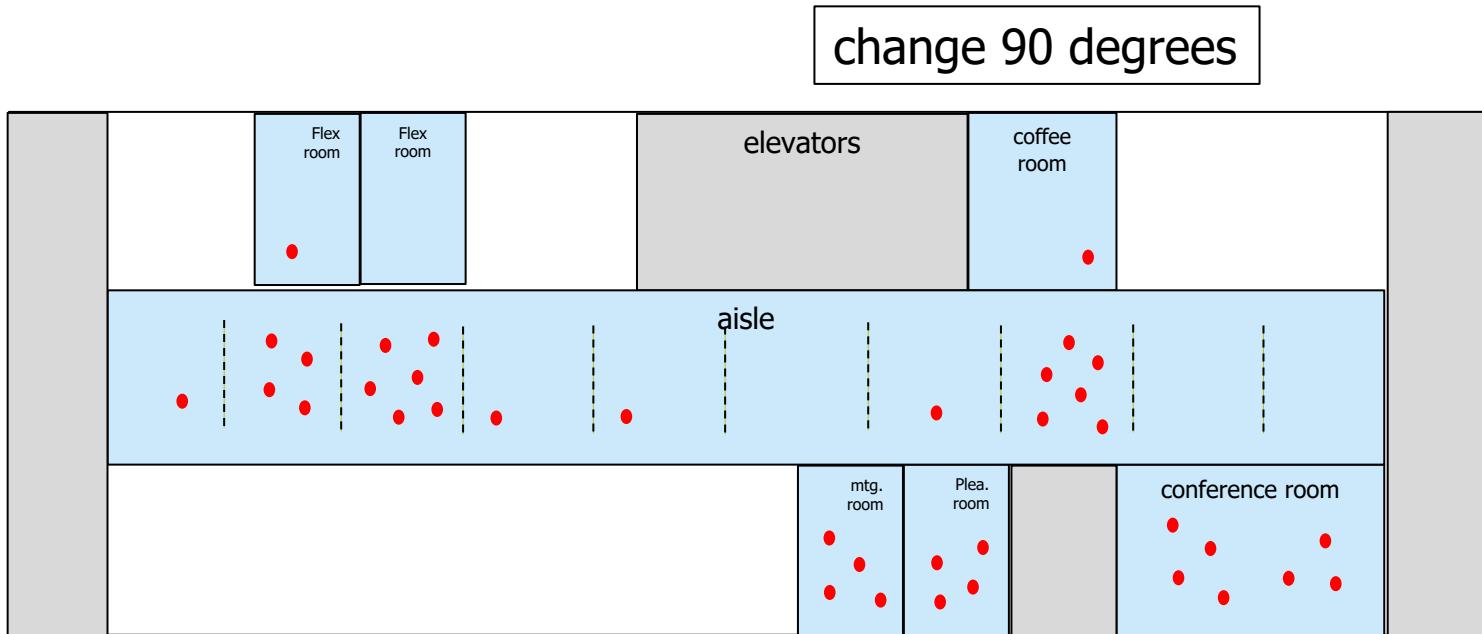
Particles close to walls should start disappearing.
To replace a particle, choose randomly among the set in the previous step and update.

Importance of capturing motion accurately (1)



What cells would end up with more particles (concentration)?

Importance of capturing motion accurately (2)



We are left with a dense (particle) areas!

A good model for distance and direction would accelerate
'convergence'. **We don't want random walks.**

Particle update in real scenario



Turns accelerate the convergence process significantly

IMPORTANT:

Only use these methods if you CANNOT use STEP_COUNTER and ROTATION_VECTOR.

If you have to develop the motion model on your own, you must notify me, to adjust your evaluation accordingly

MOTION MODEL

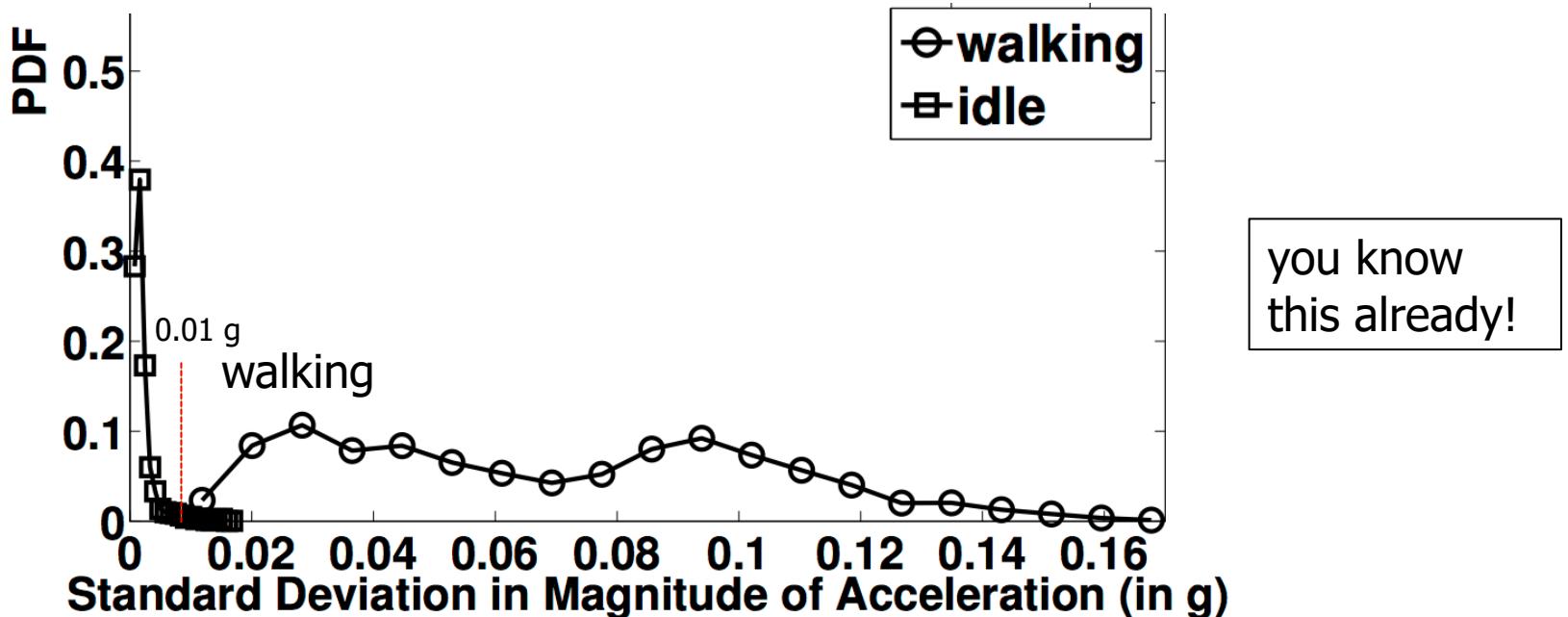
Capturing people's motion

- Each particle has (x, y, distance, direction)
- Given the importance of capturing motion accurately, the paper focuses on:
 - 1) Counting steps (distance)
 - Standard deviation
 - Autocorrelation
 - 2) Heading Offset (direction)
 - Fourier transforms
 - Geometry

Counting steps

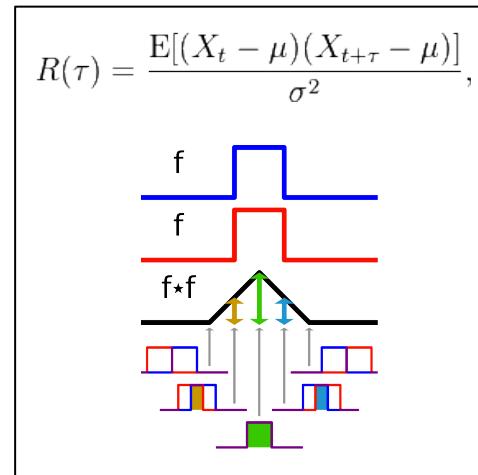
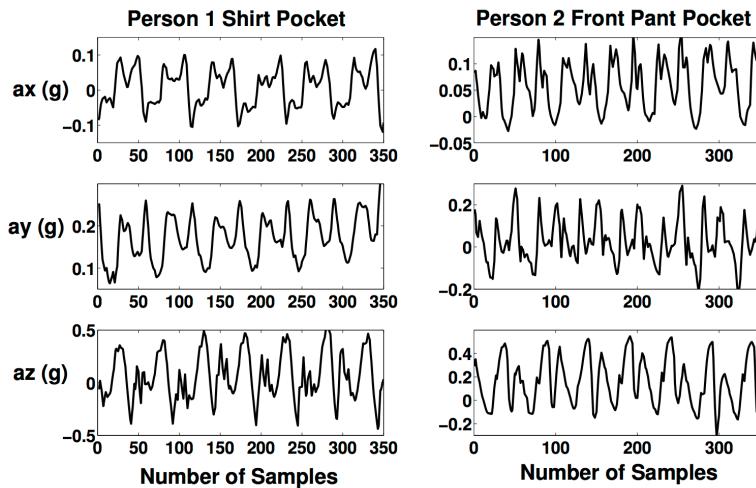
- Method works irrespective of device placement.
 - Tested in pockets, bags, belt, hands.
- Method relies on
 - Standard deviation of acceleration ☺
 - Autocorrelation

Standard deviation



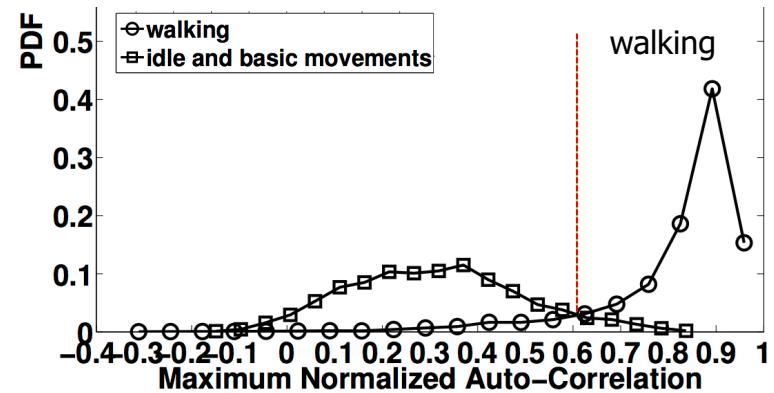
- 6 people (man/women) with phone in different positions.
- Feature: **magnitude** of acceleration. Window: 1 sec
- But acceleration is not sufficient, sudden movements such as hand waving, chair rotation, standing up, also cause acceleration

Autocorrelation



We need to take advantage of repetitive nature of walk

The period also provides the time taken per step ($\tau/2$)



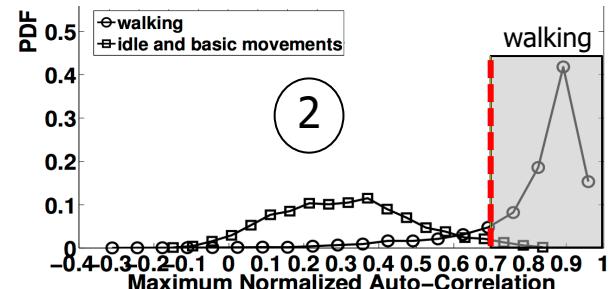
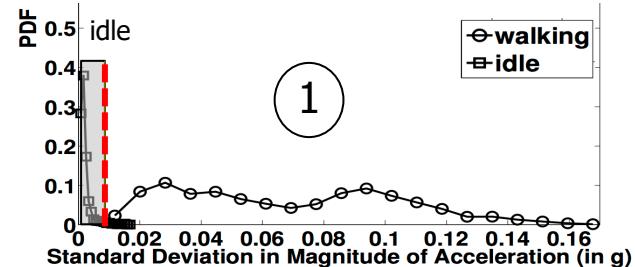
Final algorithms and results

Algorithm

If $\sigma_{\text{all}} < 0.01$ then *state* = IDLE.
Else If $\psi > 0.7$ then *state* = WALKING.
Else no change in current value of *state*.

	Hand While Using	Pant Front Pocket	Pant Back Pocket	Hand Not Using	Shirt Pocket	Hand bag	Over (all)
False +ive	0%	0%	0%	0%	0%	0%	0%
False -ive	2%	0%	0%	0%	0%	0%	0.6%
True +ive	100%	100%	100%	100%	100%	100%	100%
True -ive	98%	100%	100%	100%	100%	100%	99.4%

Great results for various placements!
**The method is a combination of what
(some of) you guys are doing!**



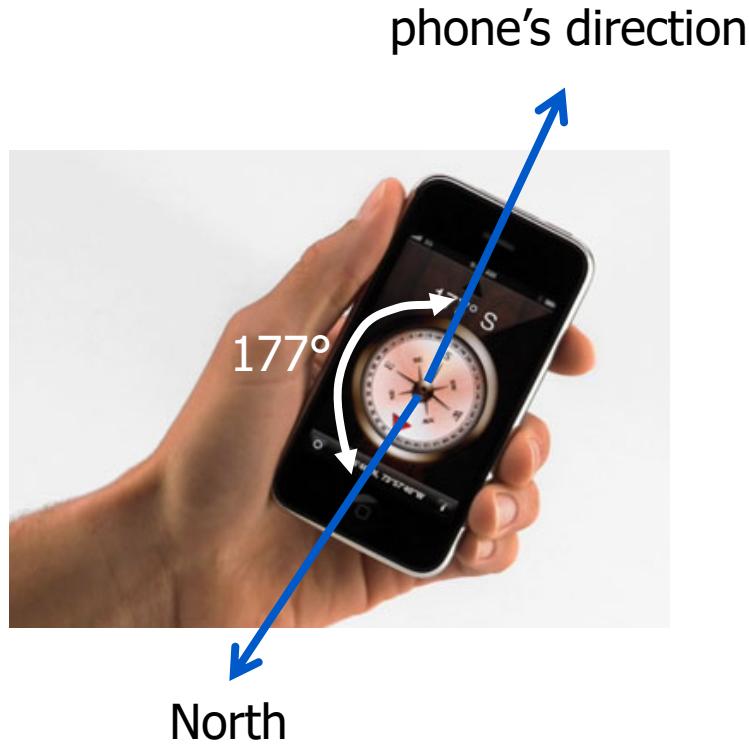
keep previous state

Recall that each particle has four parameters (x, y, distance, direction)

We now have **distance**

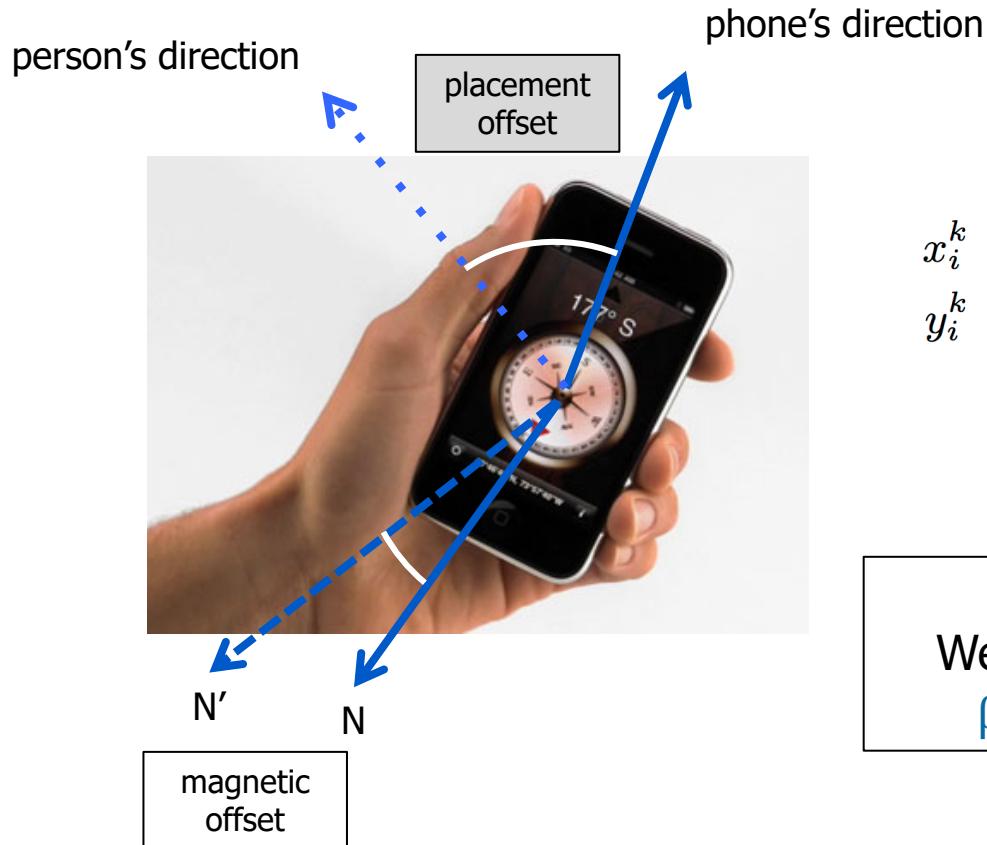
But we still need **direction**

Heading offset



The phone gives you a relative direction with respect to the cardinal North (like any compass would do)

Placement & magnetic offsets



$$\begin{aligned}x_i^k &= x_i^{k-1} + \underbrace{(s_i + \delta_i)}_{\text{distance}} \cos(\alpha_i + \theta + \beta_i) \\y_i^k &= y_i^{k-1} + \underbrace{(s_i + \delta_i)}_{\text{distance}} \sin(\alpha_i + \theta + \beta_i)\end{aligned}$$

We have Θ (phone direction).
We need α (placement offset) and
 β (magnetic offset plus errors).

Placement offset

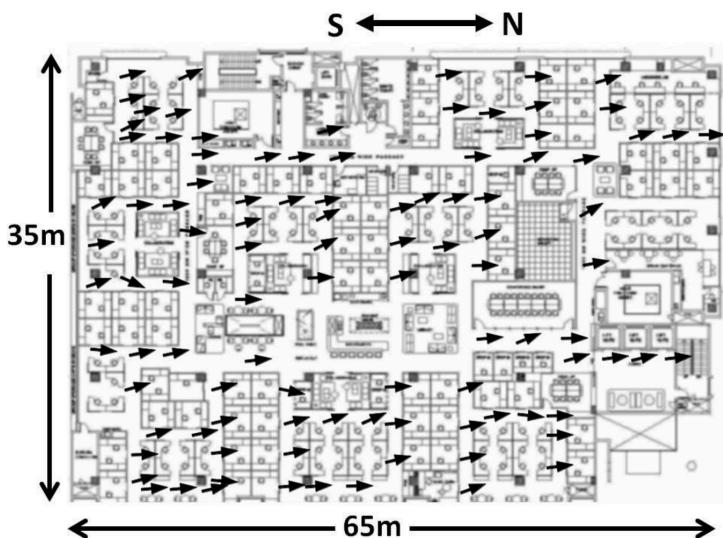
- The phone very likely points to a direction different from our moving direction.

The placement offset (α) is an important factor, but it is not mandatory to deal with it in this course.

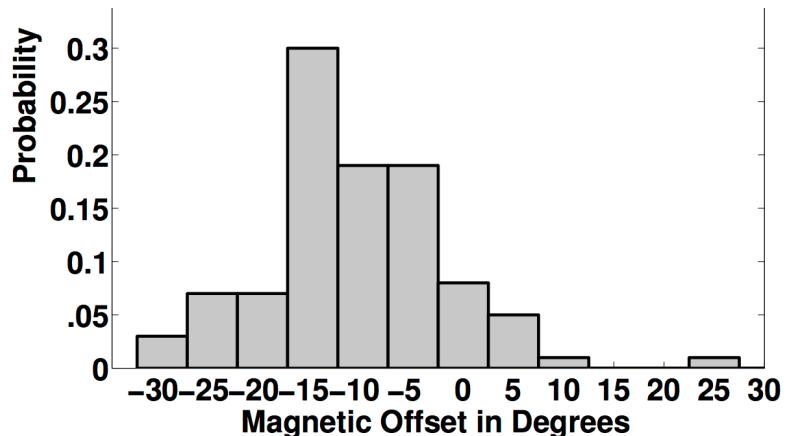


- Typical offsets
 - watching a movie with phone held laterally: 90
 - phone in shirt/pants pockets: +/- 45
 - bags: anywhere

Magnetic offset



$$x_i^k = x_i^{k-1} + (s_i + \delta_i) \cos(\alpha_i + \theta + \beta_i)$$
$$y_i^k = y_i^{k-1} + (s_i + \delta_i) \sin(\alpha_i + \theta + \beta_i)$$

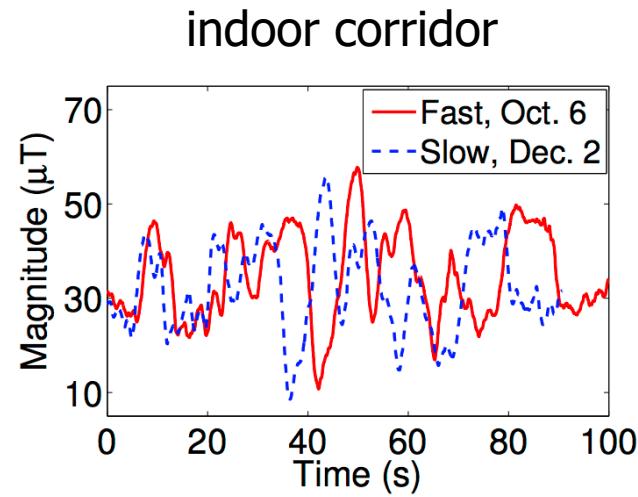
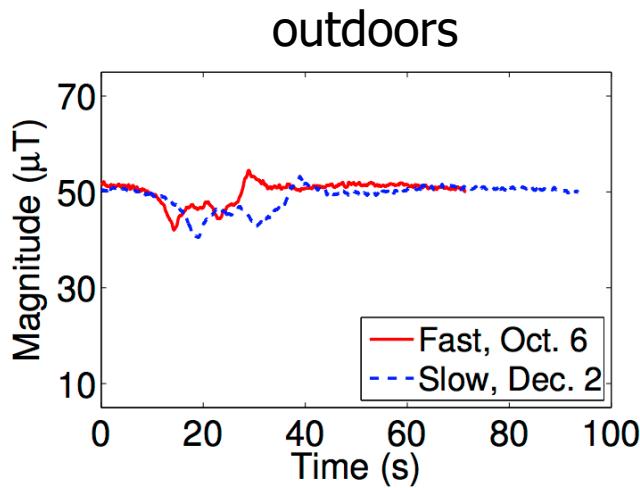


The magnetic offset depends on the building materials and surrounding metallic elements.

This may not be the most important factor, but we will only test this factor in your app.

The problem with compass (1)

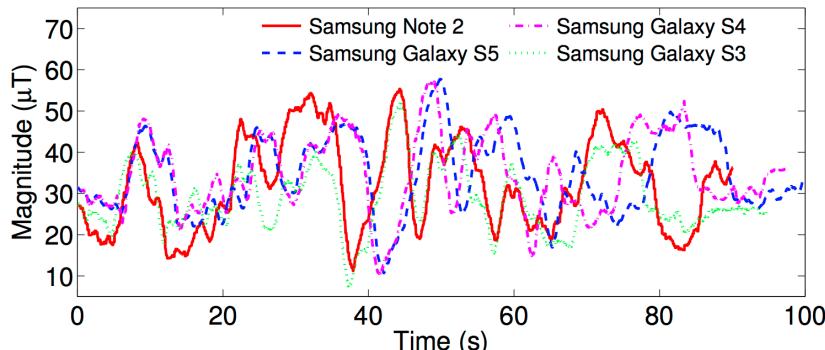
- Compass is very sensitive in indoor environments



Images taken from: "Last-Mile Navigation Using Smartphones", ACM Mobicom 2015

The problem with compass (2)

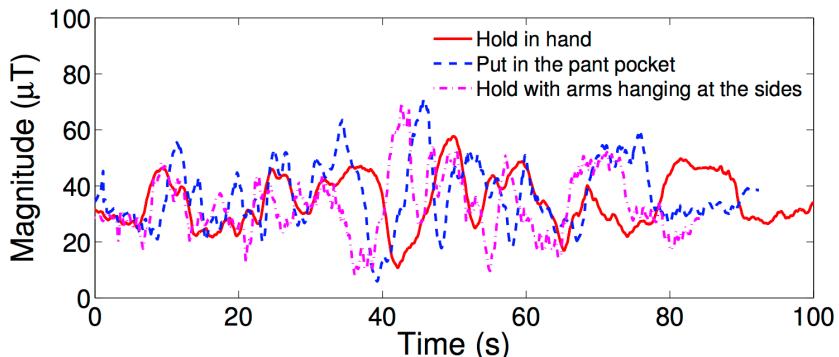
- The noise is phone-dependent and location-dependent.



(a) Device diversity

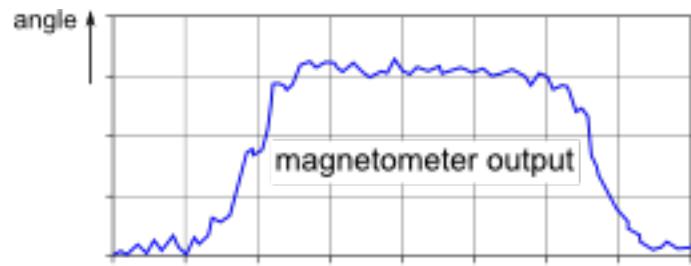
different phones will have
different values
(sometimes it gets weird)

different body locations
will have different values



(b) Usage diversity

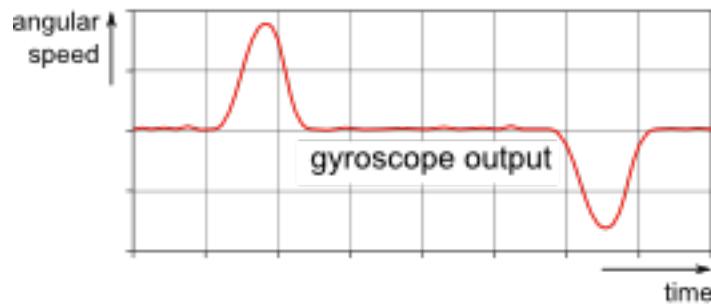
Combine Compass with Gyroscope



Pro: good for macro direction

Con: bad for micro direction

- magnetic field is easily affected in modern buildings



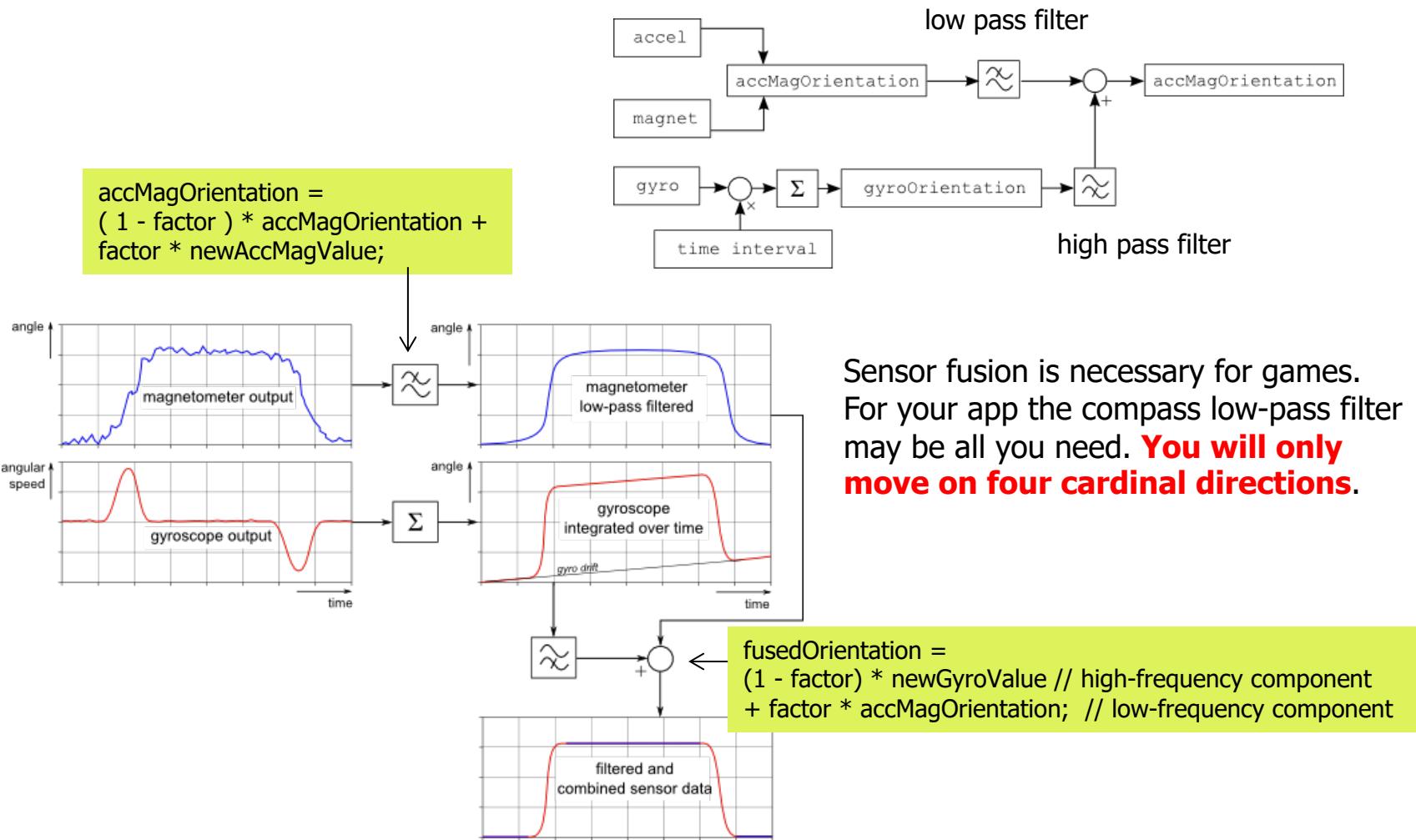
Pro: good for micro direction

Con: bad for macro direction

- accumulated error grows in time

<http://www.codeproject.com/Articles/729759/Android-Sensor-Fusion-Tutorial>

Sensor Fusion: combine acc, gyro and magnetometer



Sensor fusion is necessary for games.
For your app the compass low-pass filter
may be all you need. **You will only
move on four cardinal directions.**

Directionality: not an easy problem in general

- Do it incrementally
 - Hold phone in hand in same direction of walk
 - distance = step count + noise
 - direction = phone direction + magnetic offset + noise
- Related paper
 - “I am a Smartphone and I can Tell my User’s Walking direction”, ACM Mobicys 2014
 - <http://synrg.csl.illinois.edu/papers/walkcompass.pdf>

GETTING RSSI DATA FOR FREE

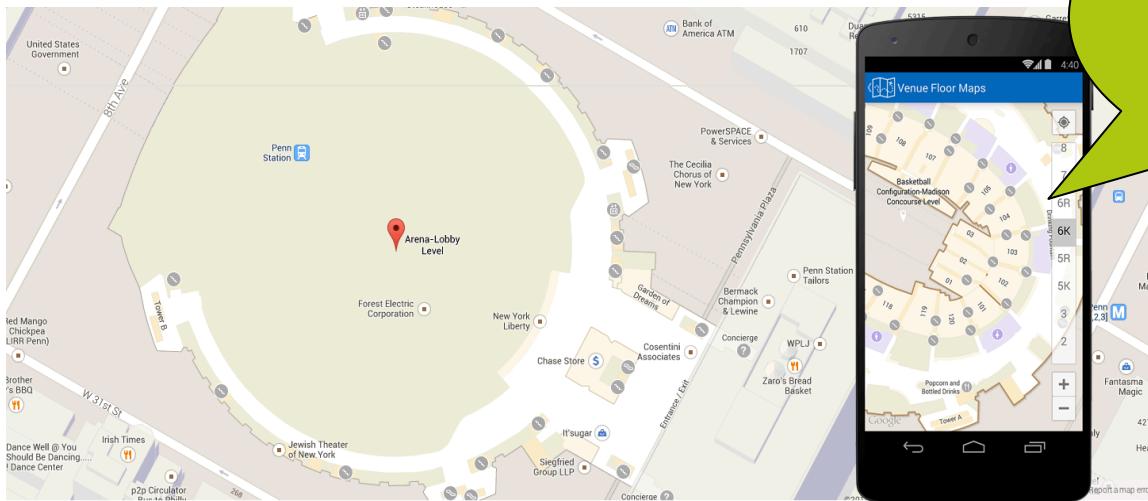
Paper:

"Zee: Zero-Effort Crowdsourcing for Indoor Localization", ACM mobicom 2012

<http://research.microsoft.com/apps/pubs/default.aspx?id=166309>

ZEE: ZERO-EFFORT LOCALIZATION

Indoor maps



Where am I?
I wish somebody
would have
collected rssi data
for me

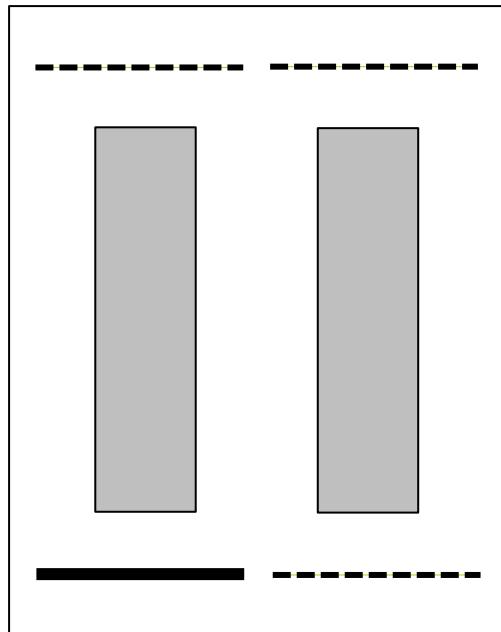
This Lecture has 3 parts

- 1) Particle Filters
- 2) Motion model: distance and direction
- 3) Building RSSI database for “free”

Zee: Zero Effort Crowdsourcing

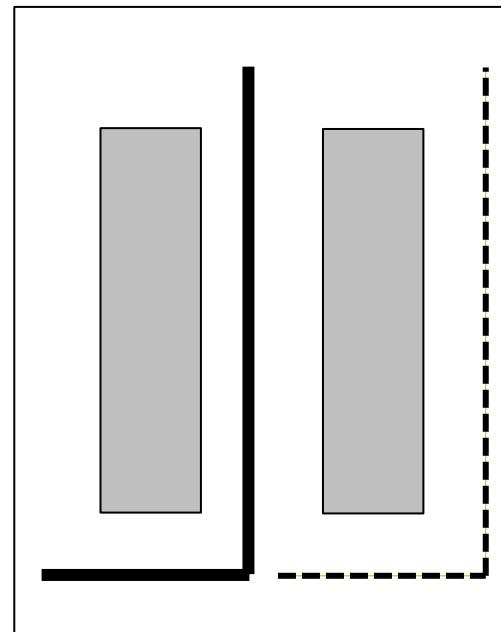
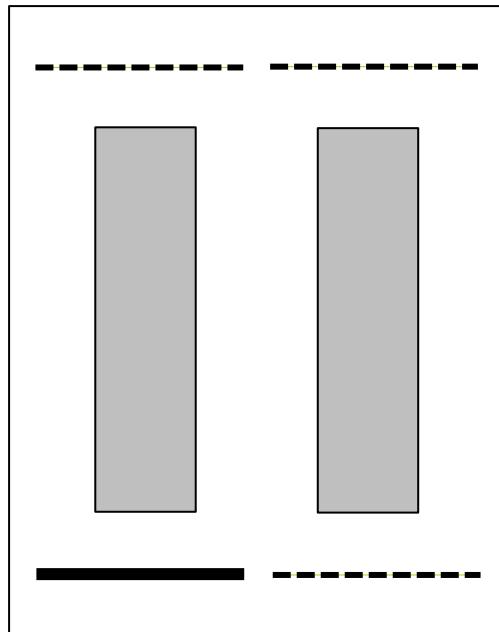
- An ‘easier’ way to build radio maps with particle filters.
- Zee relies on very reasonable assumptions:
 - Inertial sensors
 - WiFi modems
 - Indoor map
 - Methods:
 - autocorrelation, Fourier Transforms, Basic Geometry and particle Filters!

Basic idea: (1) constrained maps



many possible locations

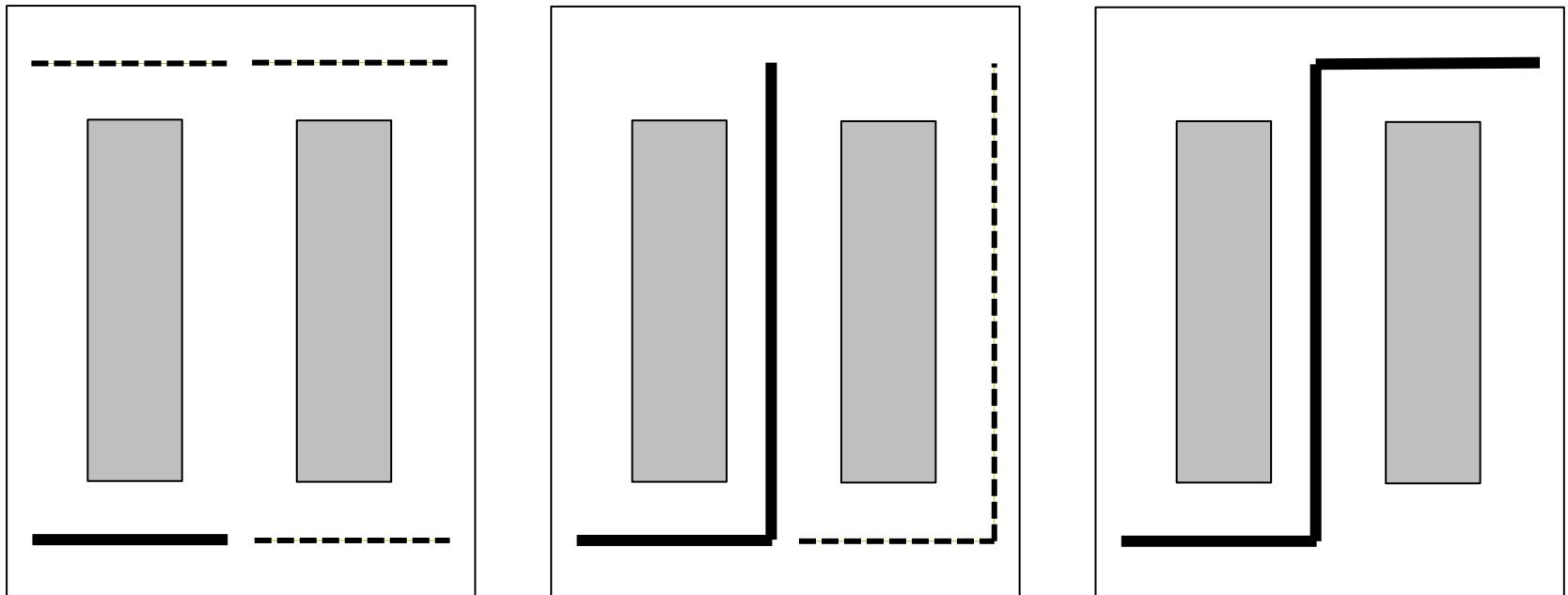
Basic idea: (1) constrained maps



two possible locations

Basic idea: (1) constrained maps

how do we do this with particle filters?



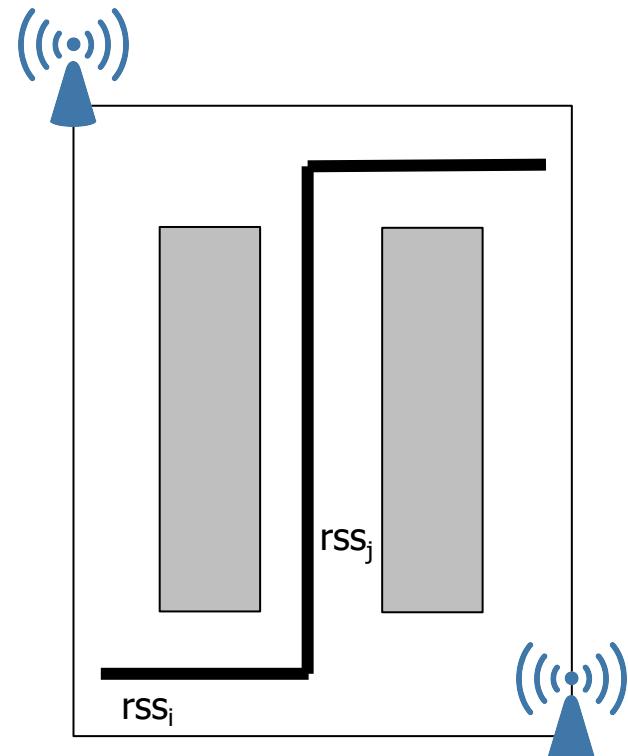
one possible location

Basic idea: (2) back propagation

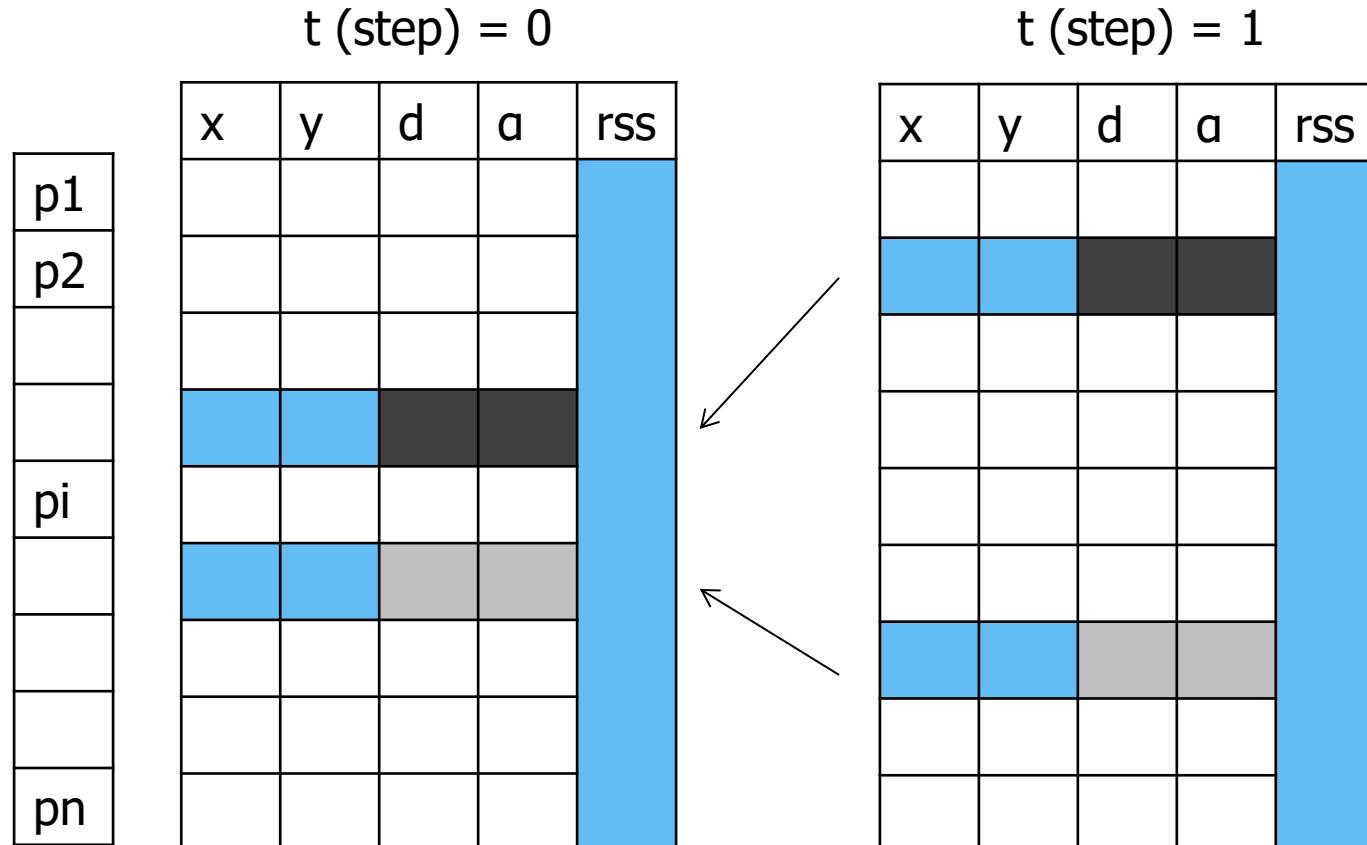
During this time, we have been recording RSS data.

As users cover unvisited areas, we can collect the entire radio map.

For subsequent users the (partial) radio map can be used to improve the initial belief.

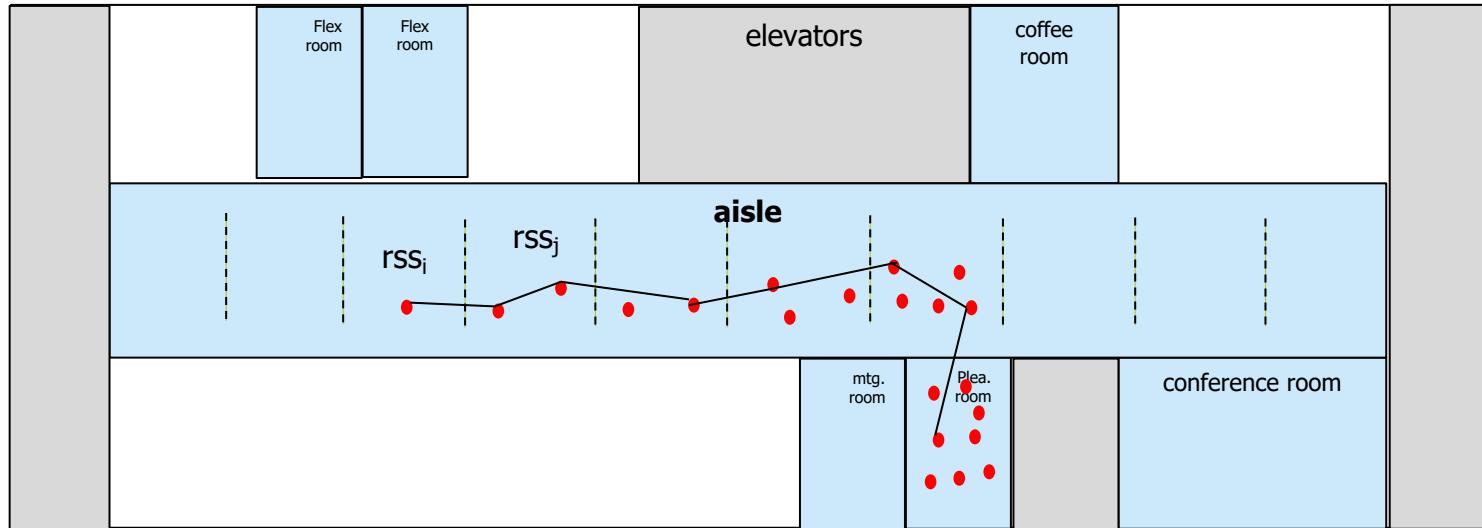


Back propagation



Keep track of parents at each step.
Prune the rest (to save memory).

Final radio map



In time we collect information for
our radio map.

You don't need to do this for your final App

Particle Filters (Continuous):

at the beginning a blanket represents the pdf, then prior and posteriors lead to peaks and valleys until we have a single peak (hopefully)

Bayesian Filters (Discrete):

similar to particles but with “building heights”

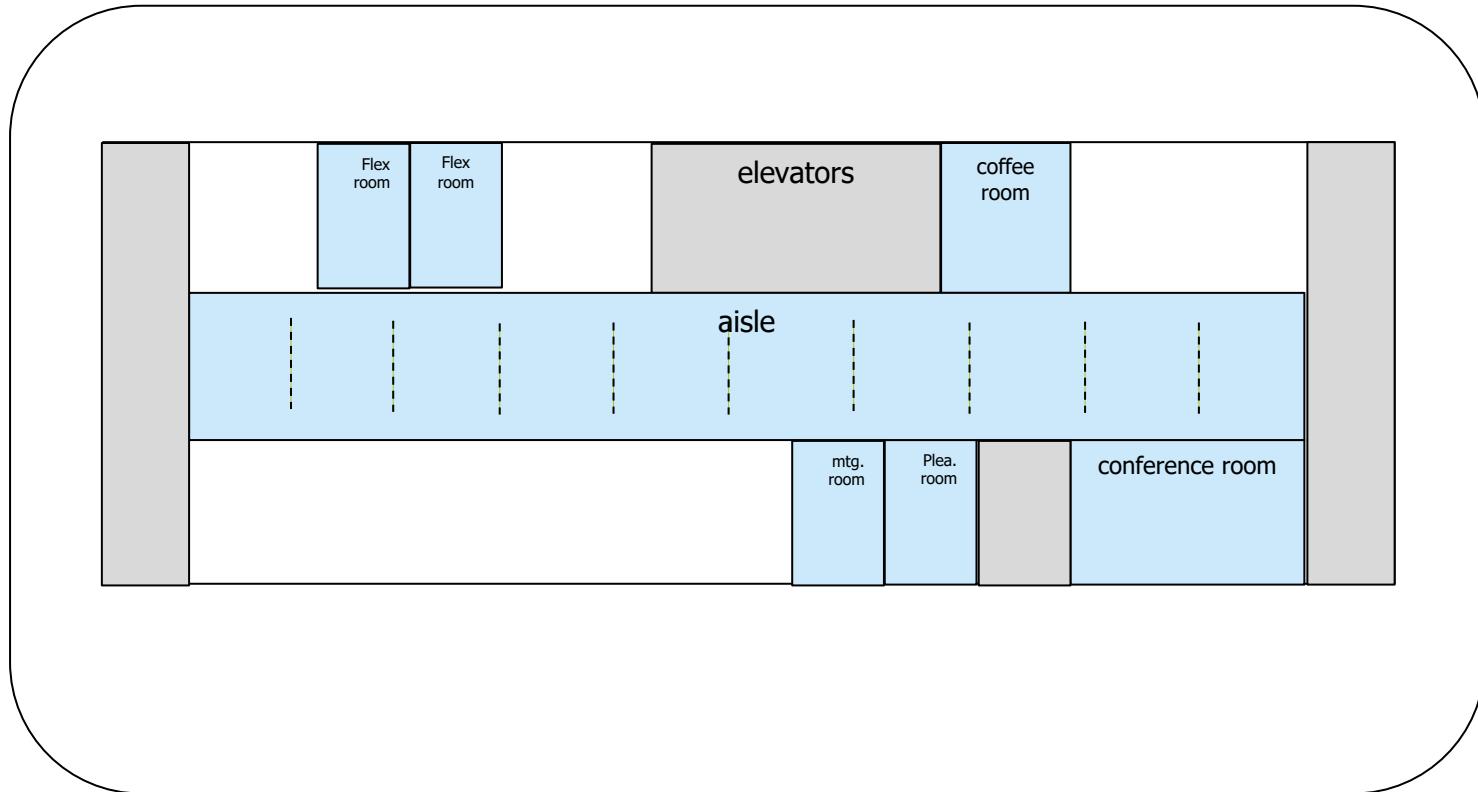
GRANDMA'S EXPLANATION

A cookbook for Particle Filters

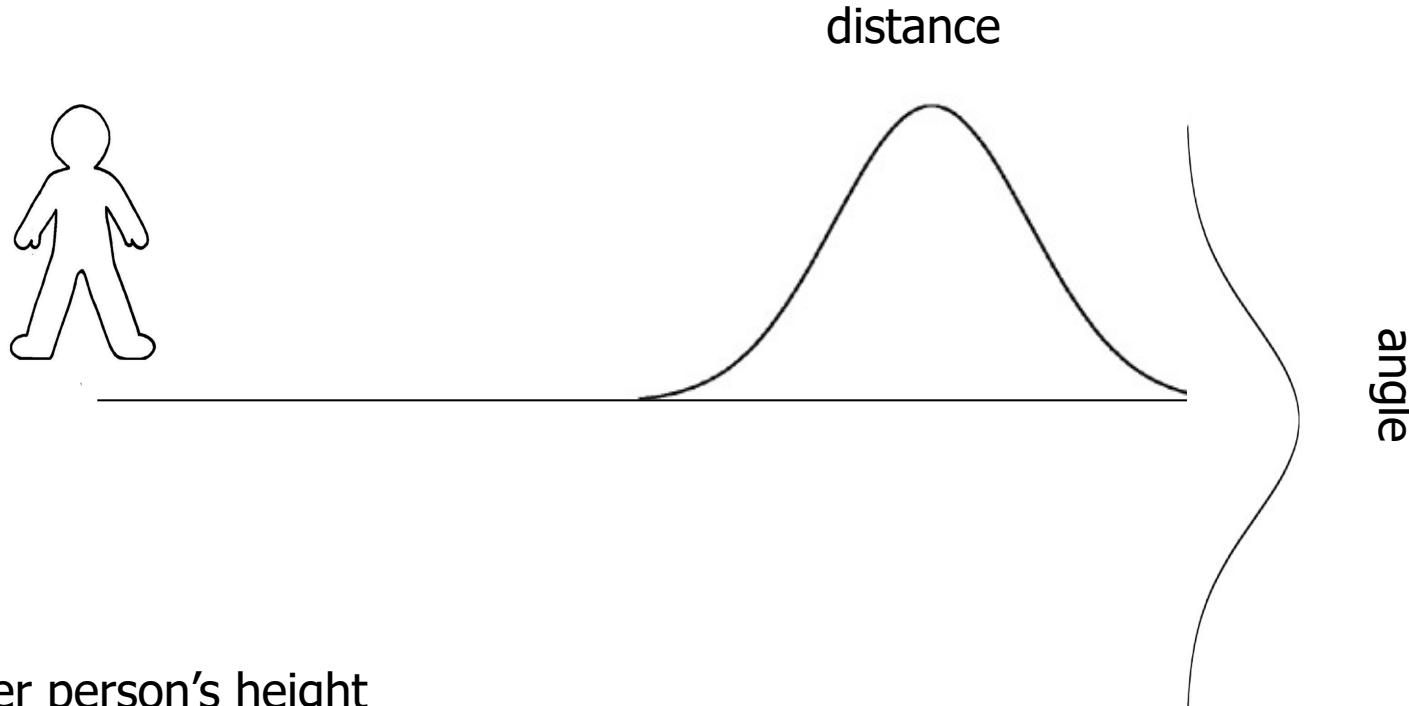
- 1) Create map with boundaries in your phone (**check example**)
- 2) Define motion model: distance & direction (consider person's height)
- 3) Deploy N particles. Big N is good (thousands), but consider phone capabilities.
- 4) Move particles, it can be done in three main ways:
 - Preferred: TYPE_ROTATION_VECTOR, TYPE_STEP_COUNTER, TYPE_STEP_DETECTOR (API 19)
 - Simple but inaccurate. Distance: moving/idle. Direction: Quadrant.
 - Accurate but challenging. Develop your own STEP_COUNTER and ROTATION_VECTOR
- 5) Identify particles that violate constraints (dead particles).
- 6) Resample: place dead particles on top of surviving ones.
- 7) Check for convergence. Go back to step 5
- 8) **Test it online in our floor!**

Step 1) Create Map

- Different ways to do this in Android (Canvas)
- More elaborated than simple Bayesian
- Map with dimensions is posted in FF



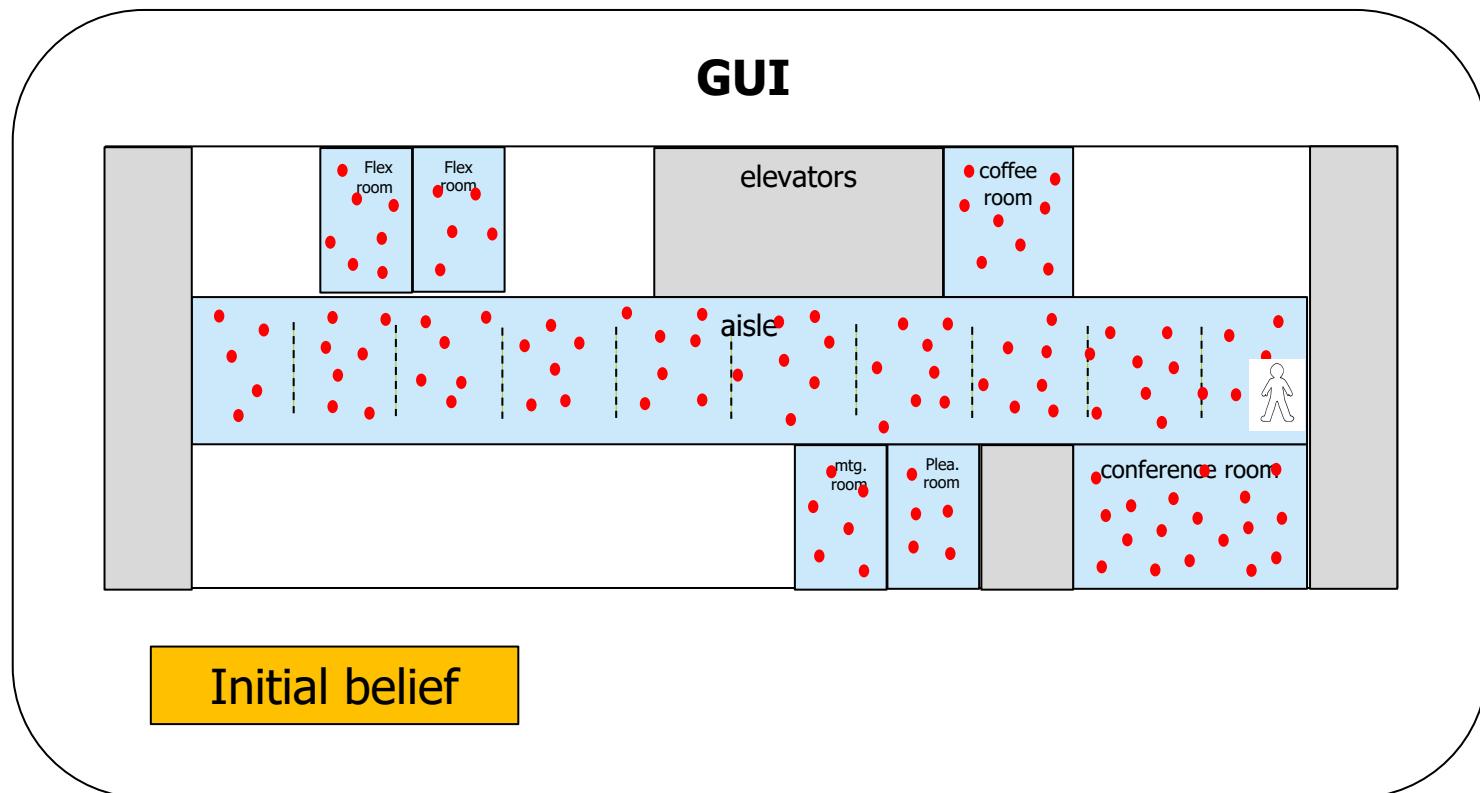
Step 2) Define motion model



- Consider person's height
- **Perform minor training** (walk a known distance and count steps)
- This model depends fundamentally on how you implement particle motion (step 5)
- **You could do this also in an automatic manner by exploiting maps.**
- Tradeoff between accuracy and convergence

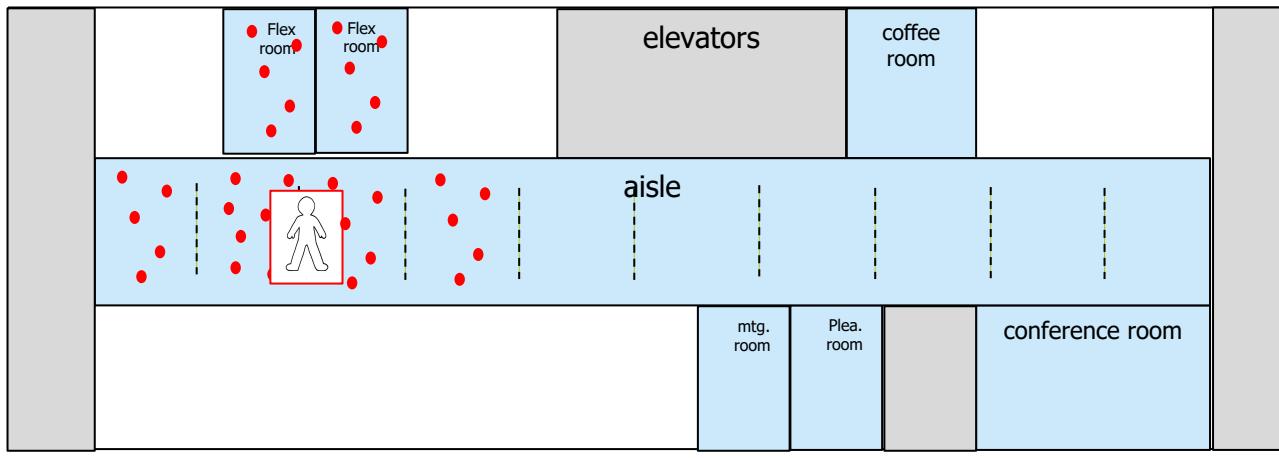
Step 3) Deploy N particles

Big N is good, but consider your phone's processing capabilities!



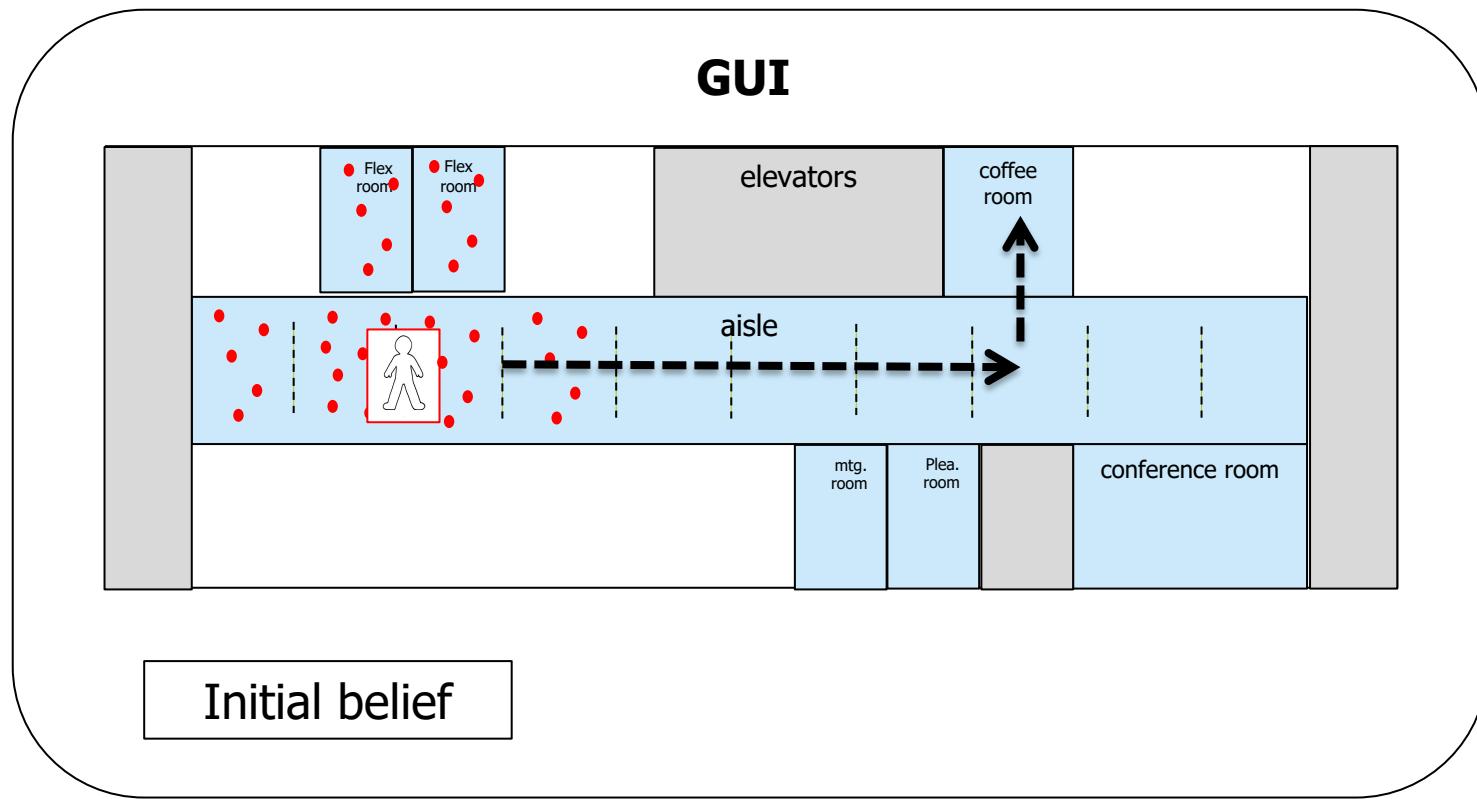
Remember: Keep track of particles in table

GUI



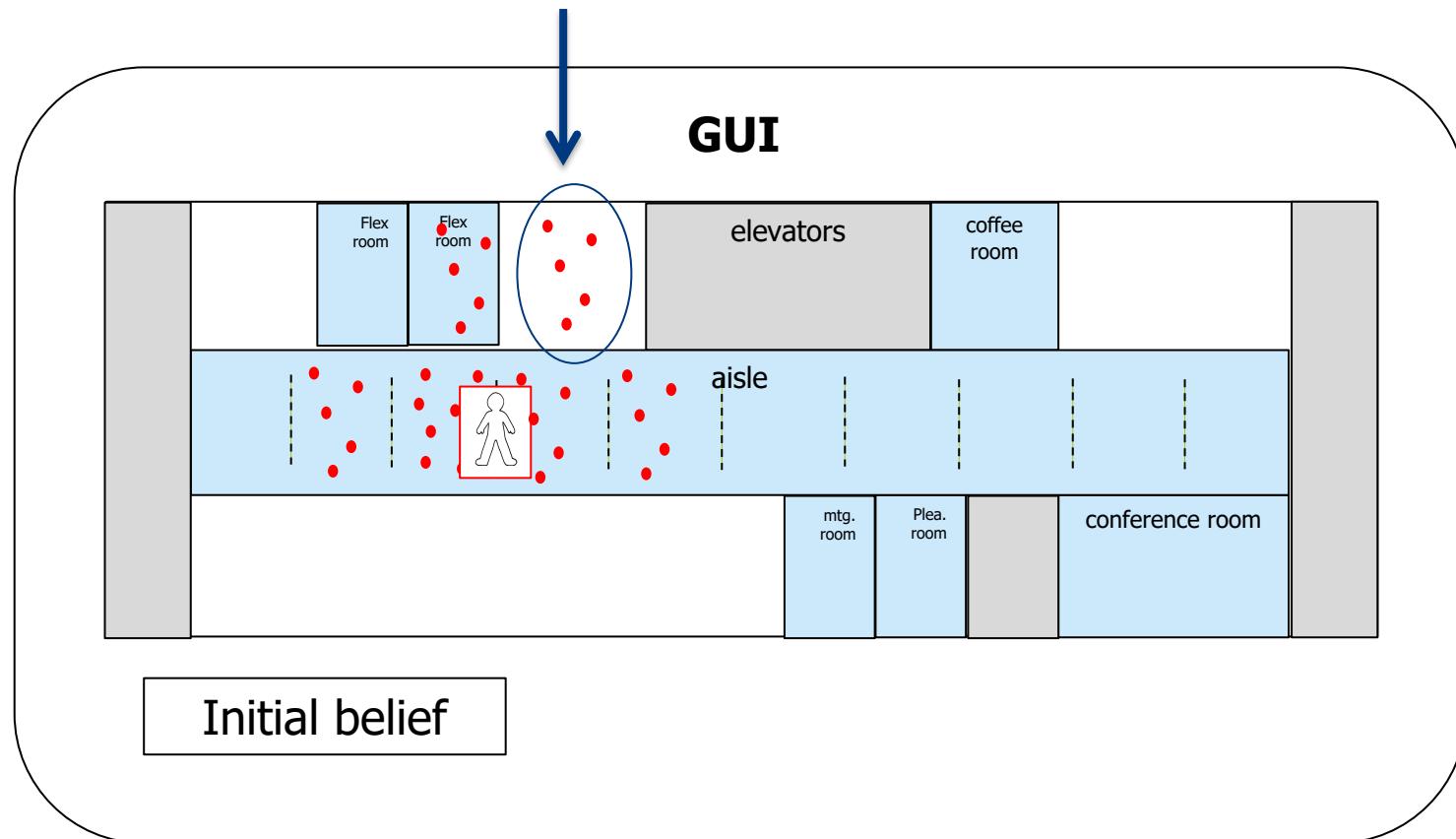
	x	y	d
p1	X1	Y1	D1
p2			
Pn	Xn	Yn	Dn

Step 4) Move particles

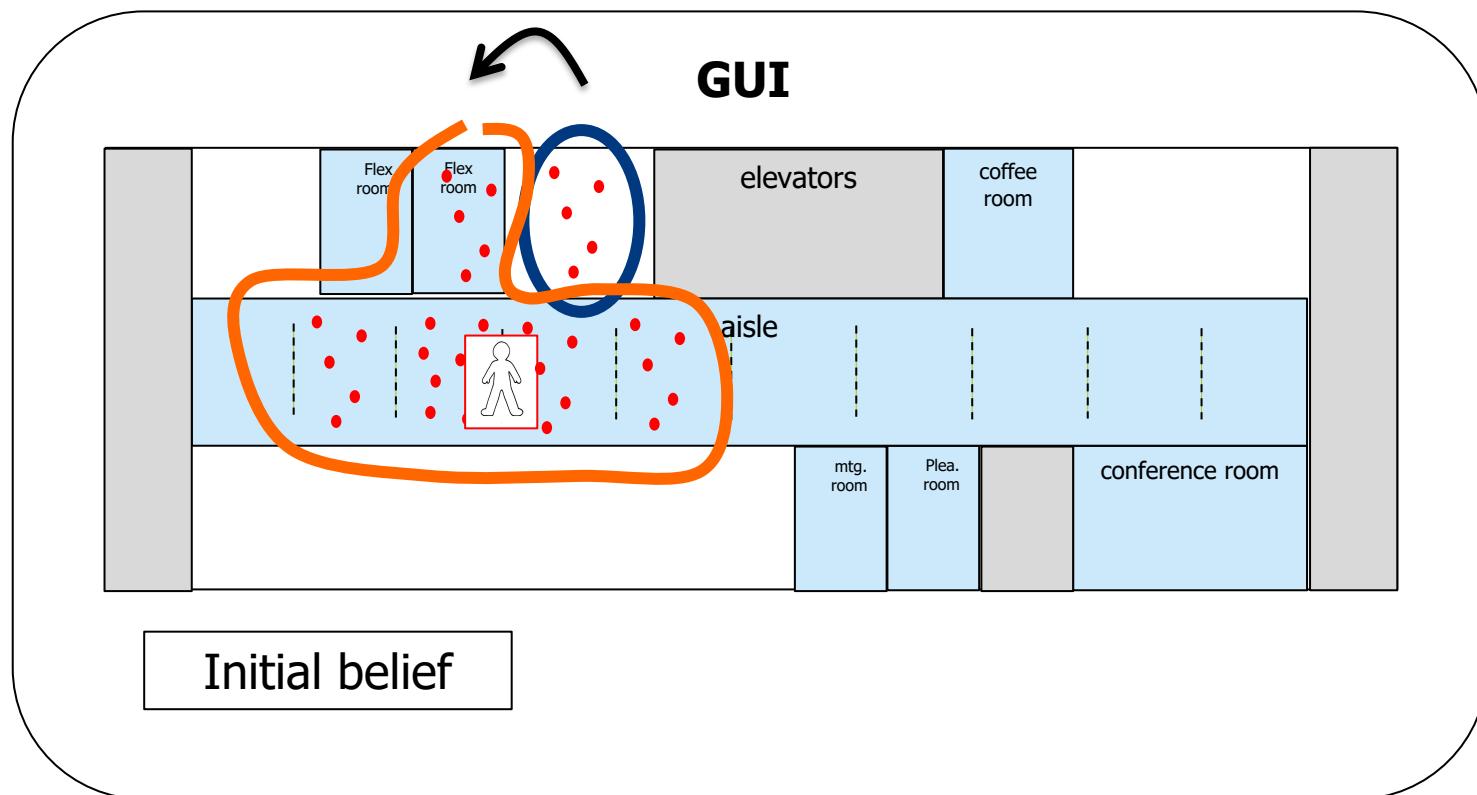


- **Preferred.** TYPE_ROTATION_VECTOR, TYPE_STEP_COUNTER,
- **Simple but inaccurate.** Distance: moving/idle. Direction: Quadrant.
- **Accurate but challenging.** Develop your own STEP_COUNTER and ROTATION_VECTOR

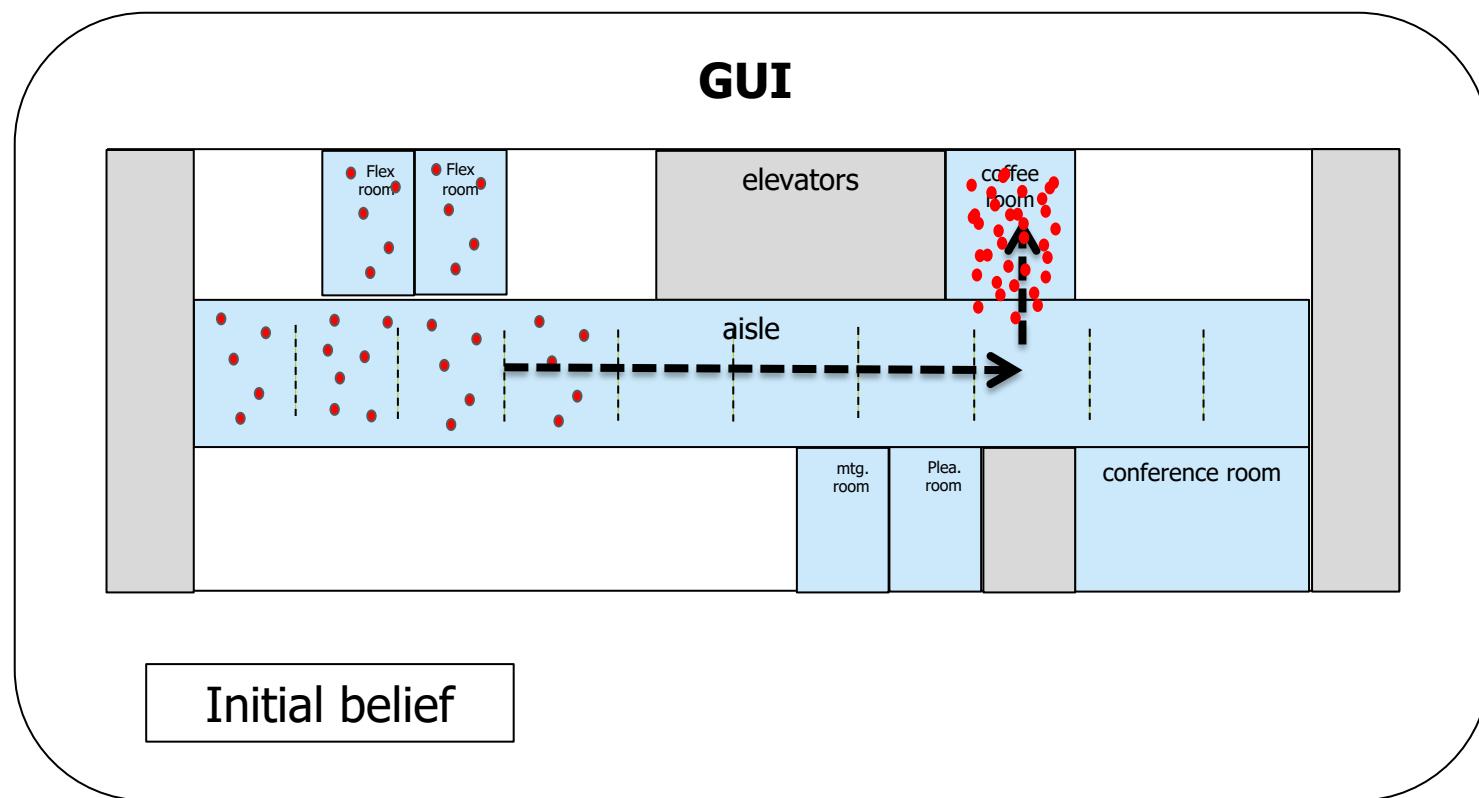
Step 5) Eliminate particles that violate constraints



Step 6) Resample



Step 7) Convergence

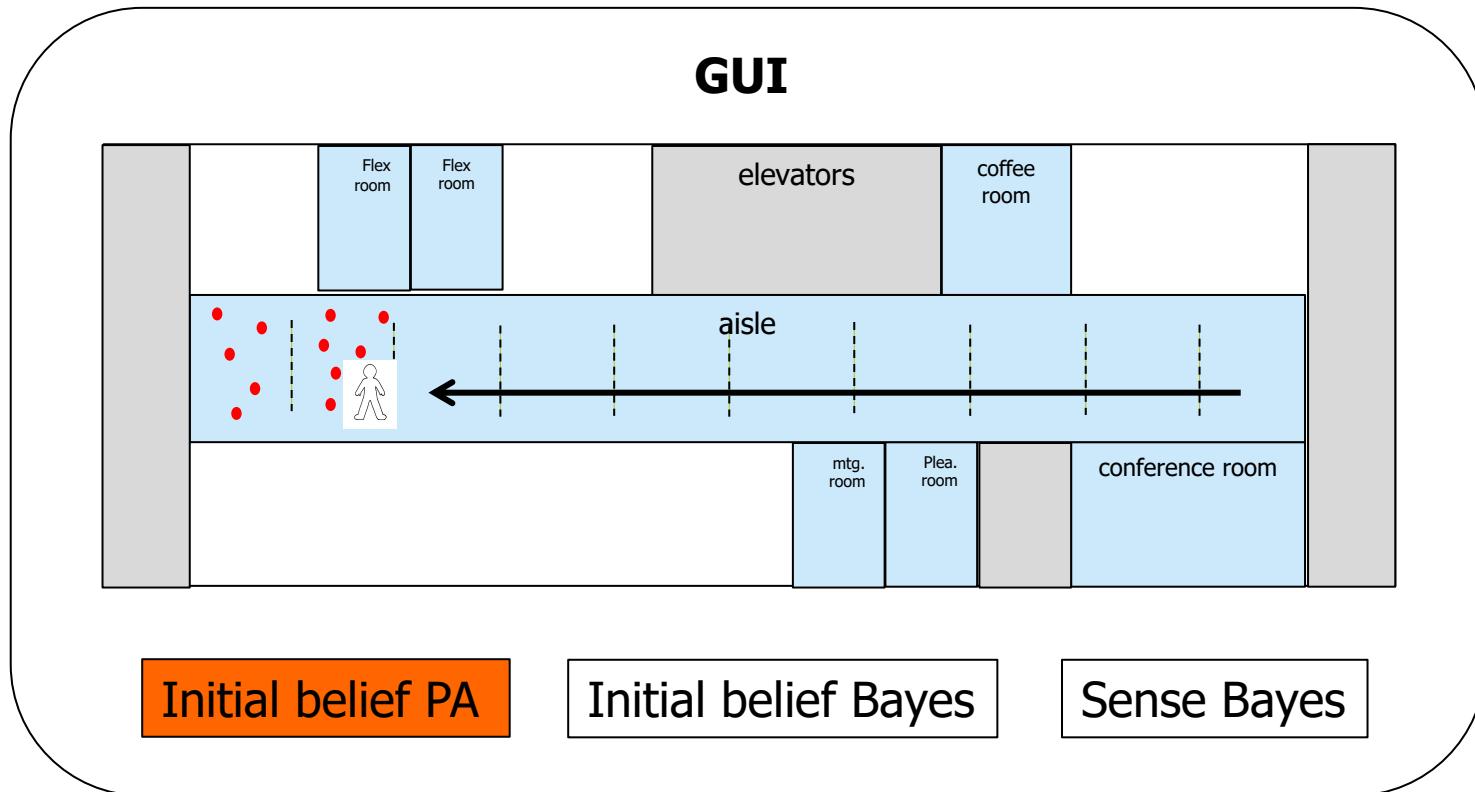


Be careful with particle traps

- If your motion model is NOT accurate. The particles may get stuck at a given point.
- You will have three chances during the evaluation to overcome this problem. After that we won't count any more points.

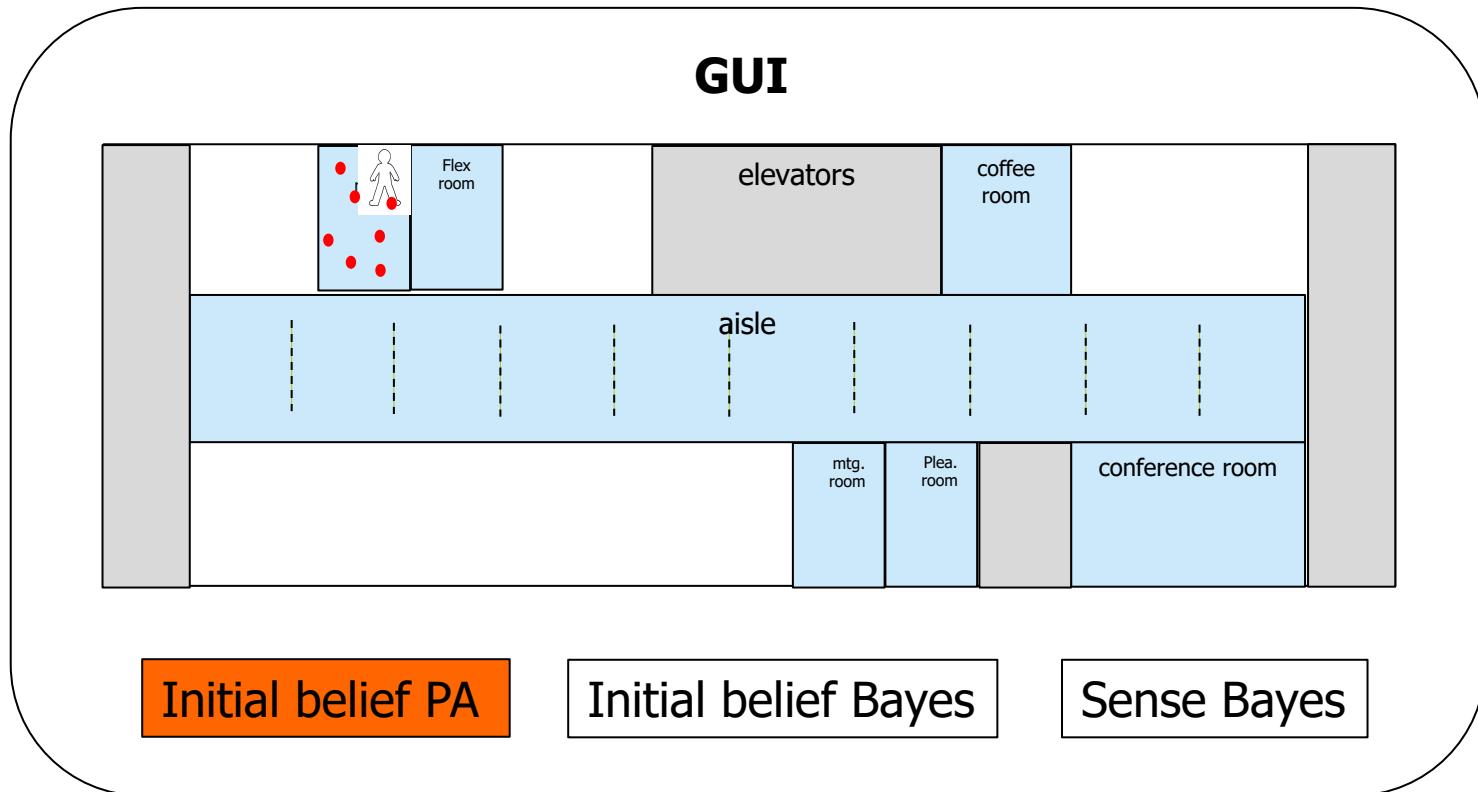
WHAT THE APP SHOULD DO

App: find convergence (1)



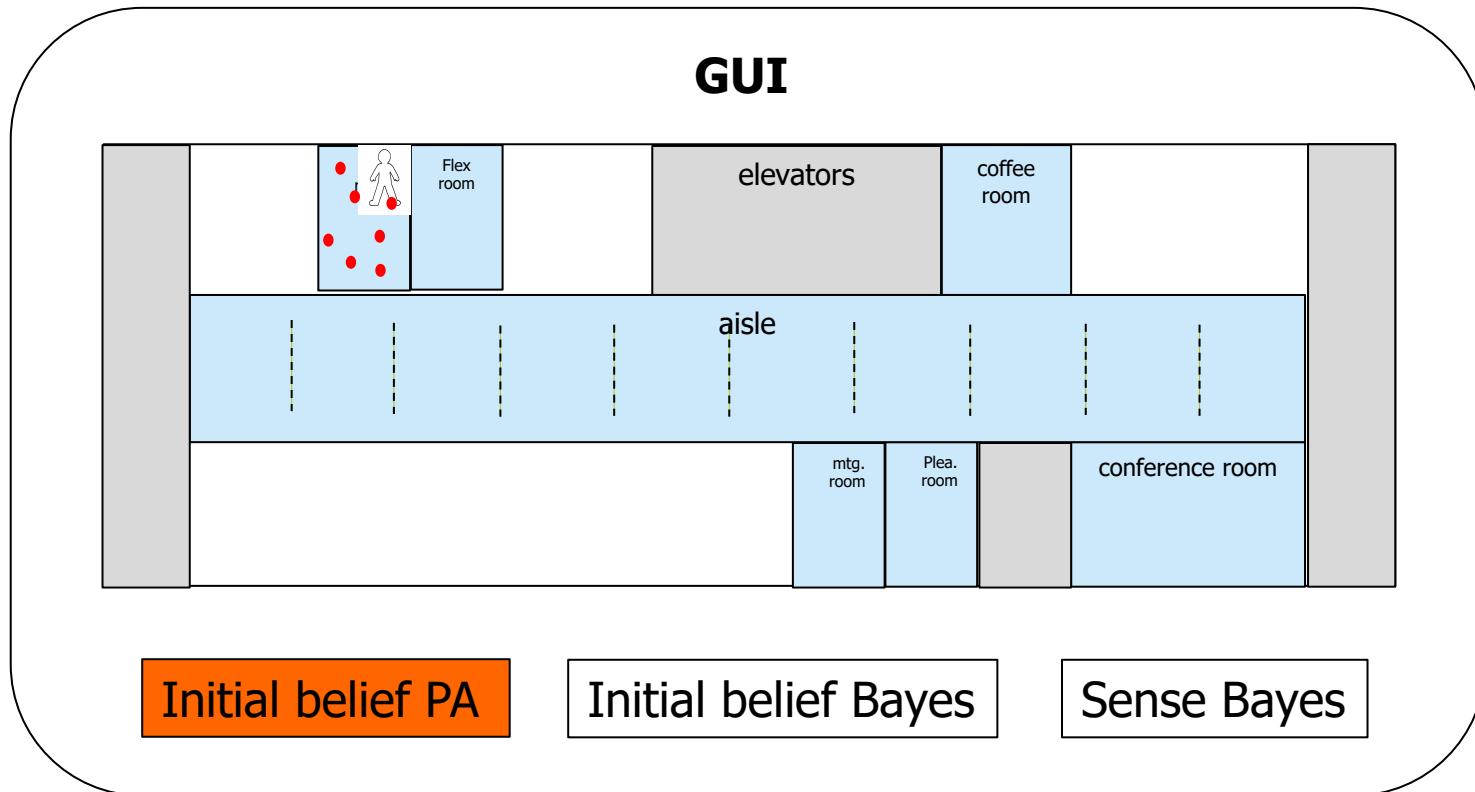
We will perform a ‘uniquely determined’ walk to find convergence.
How you determine convergence is up to you.

App: find convergence (2)



Once you converge we start testing the accuracy of particle filters.

App: testing of cells



We will visit all cells.
Your App should state the cell where you are

Pros and Cons of Particle filters

- Bayesian Filters
 - discrete, multimodal, computationally expensive, easy to program.
- Kalman Filters
 - continuous, unimodal, computationally inexpensive but assumes linearity and Gaussian distributions
- Particle filters
 - continuous, multimodal, computationally expensive, easy to program.
- Notes:
 - discrete: good for room-level localization
 - unimodal: can be a limitation
 - computationally expensive: really? this is the 2010's!