Intelligent robotics

The RAGI project

Summary

- What is RAGI?
- The Loomo robot
- Loomo’s navigation system
  - I) Localization
  - II) Path computation
  - III) Path following
  - IV) Obstacle avoidance

What is RAGI?

Système de

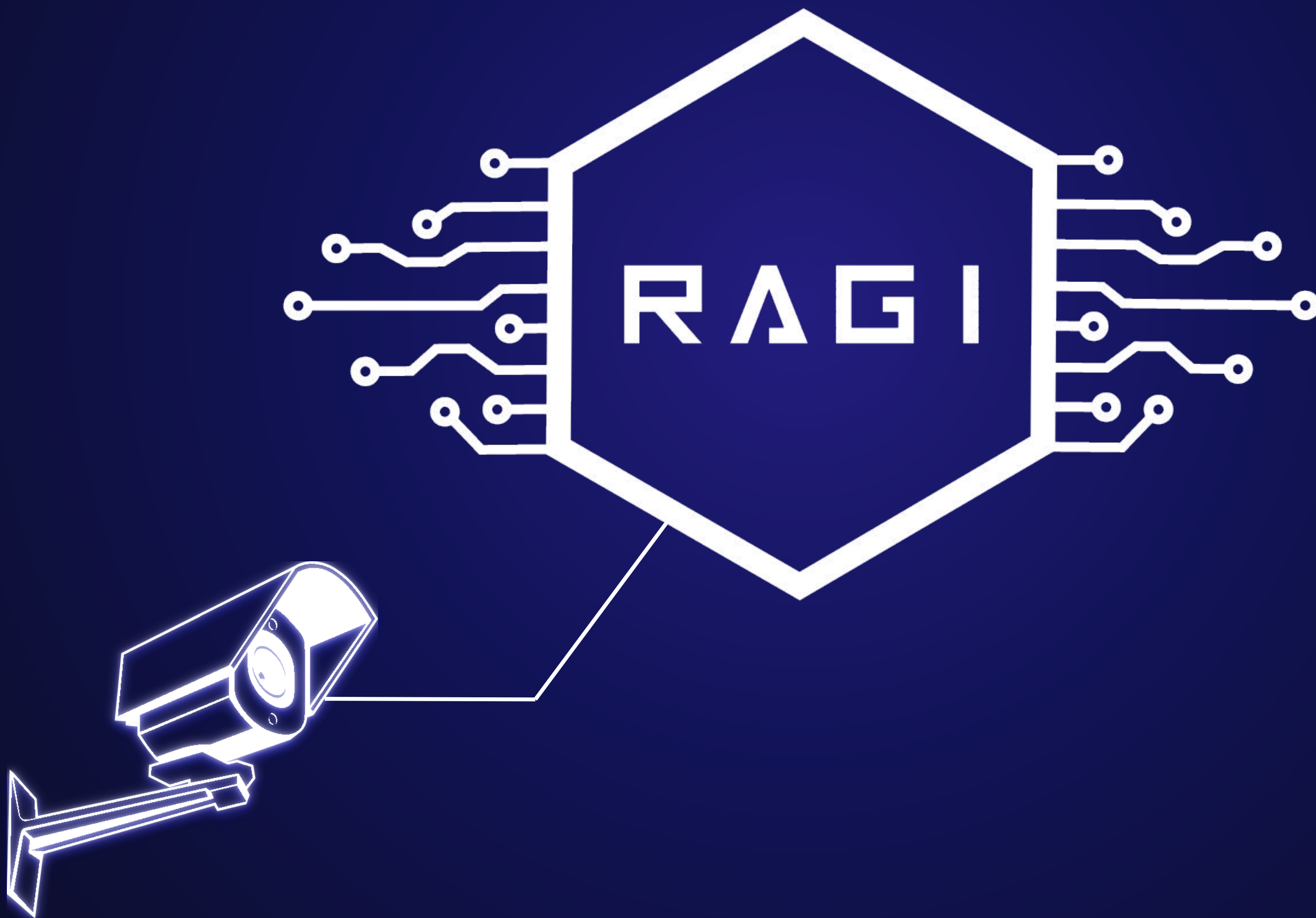
Reconnaissance, d'**Accueil** et de **Guidance Intelligent**

What is RAGI?

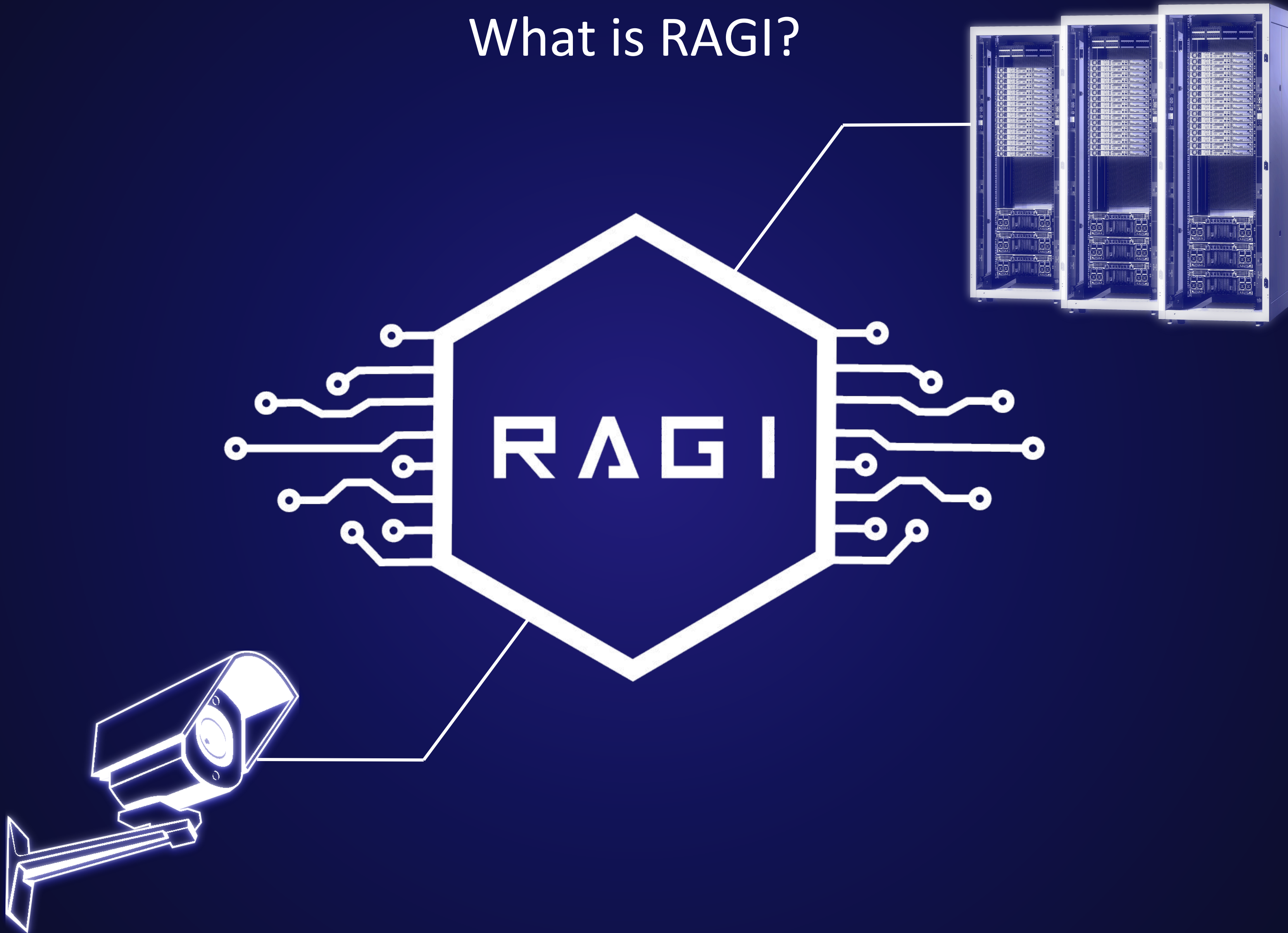
Main goals:

Localizing people **guiding visitors**

What is RAGI?



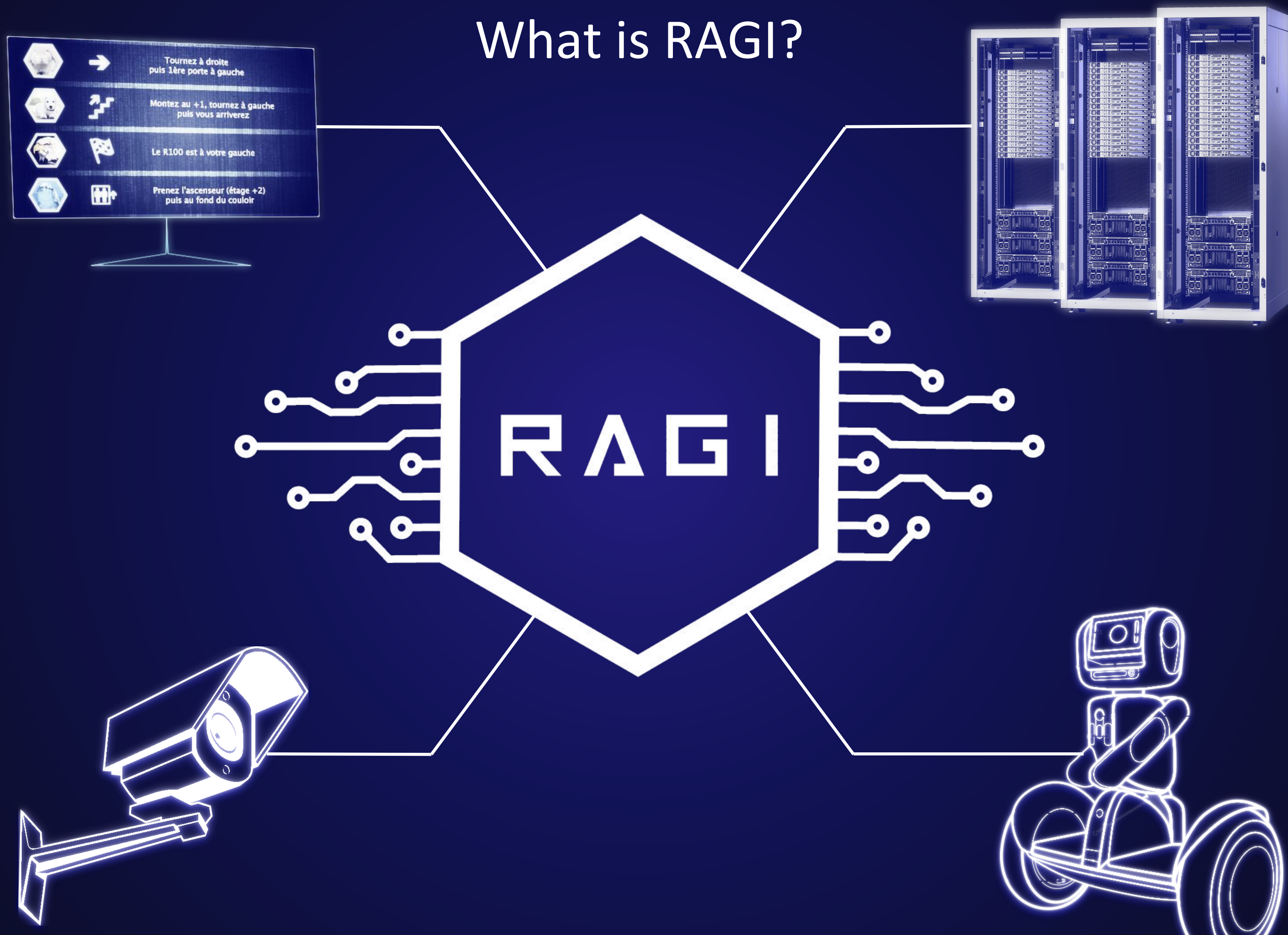
What is RAGI?



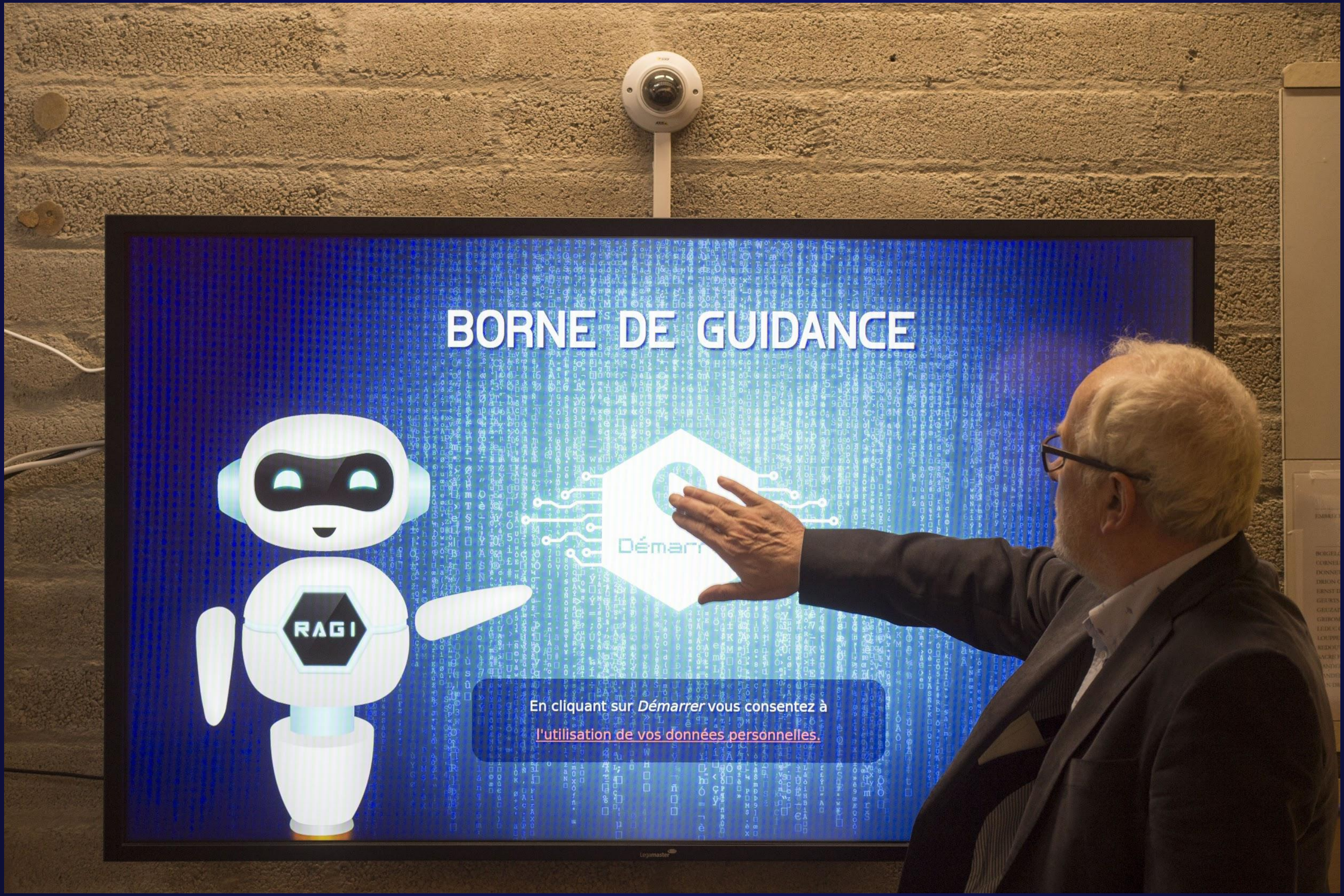
What is RAGI?


Tournez à droite
puis 1ère porte à gauche
Montez au +1, tournez à gauche
puis vous arriverez
Le R100 est à votre gauche
Prenez l'ascenseur (étage +2)
puis au fond du couloir
RAGI

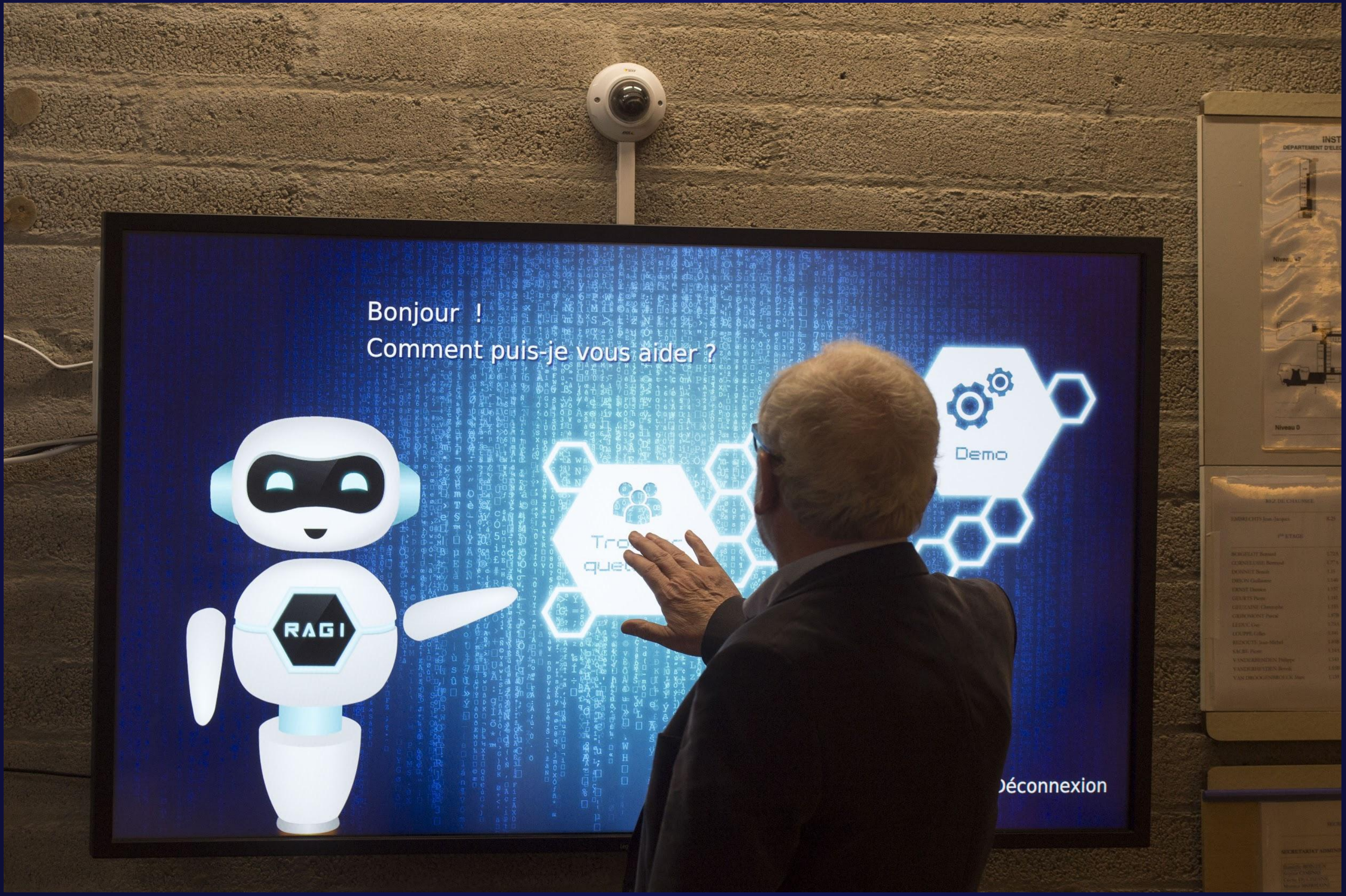
What is RAGI?


Tournez à droite
puis 1ère porte à gauche
Montez au +1, tournez à gauche
puis vous arriverez
Le R100 est à votre gauche
Prenez l'ascenseur (étage +2)
puis au fond du couloir
RAGI

What is RAGI?


BORNE DE GUIDANCE
RAGI
En cliquant sur *Démarrer* vous consentez à
l'utilisation de vos données personnelles.

What is RAGI?



What is RAGI?


Qui cherchez-vous?
Guillaume Drion
Julien Dular
Jonathan Dumas
François Dupont
J. Embrechts
Retour au menu

What is RAGI?

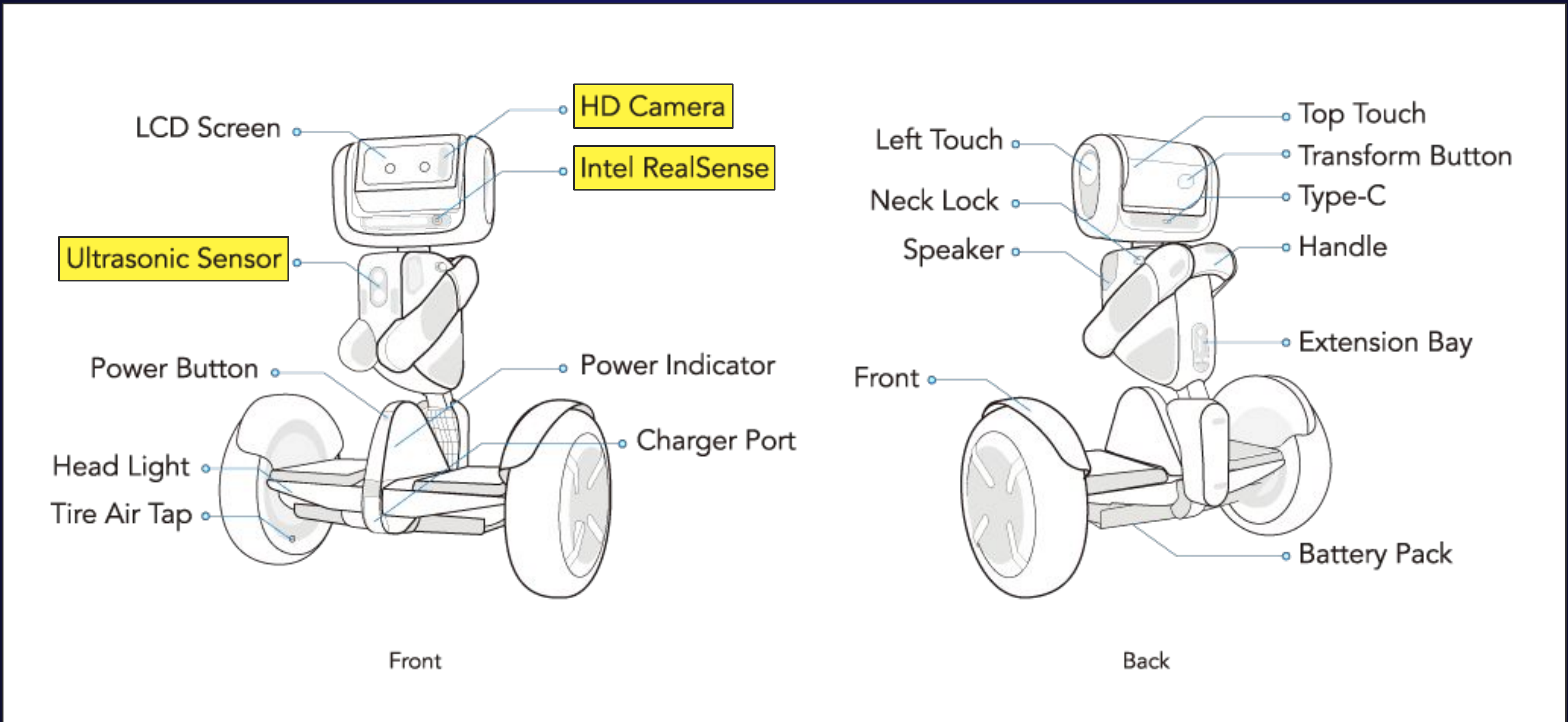
What is RAGI?


TELLIGENCE ARTIFICIELLE
AU SERVICE DE LA GUIDANCE
Reconnaissance faciale
RAGI est basé sur des techniques de reconnaissance faciale dernier cri pour localiser les utilisateurs d'un bâtiment.
RAGI
Comment souhaitez-vous être guidé ?
Par robot
Un robot va vous guider jusqu'à votre destination
Par écrans
Vous serez reconnu par reconnaissance faciale à chaque intersection et votre direction s'affichera automatiquement à l'écran
Retour au menu

What is RAGI?


Faites demi-tour
puis allez à l'écran suivant
Montez, allez à gauche
puis vous arriverez
Tournez à gauche
puis allez à l'écran suivant
Faites demi-tour
puis allez à l'écran suivant
SAMSUNG

Loomo robot



LCD Screen
HD Camera
Intel RealSense
Ultrasonic Sensor
Power Button
Power Indicator
Charger Port
Head Light
Tire Air Tap
Front
Left Touch
Top Touch
Transform Button
Neck Lock
Type-C
Speaker
Handle
Extension Bay
Front
Battery Pack
Back

Why Loomo?

- Cheap
- Powerful & reliable locomotion
- 2D & 3D cameras
- API for developers

Navigation system

4 main problems:

I) Localization

II) Path computation

III) Path following

IV) Obstacle avoidance

Navigation system

Our solutions:

I) Localization

→ Particle filter : **C**orrective **G**radient **R**efinement [1]

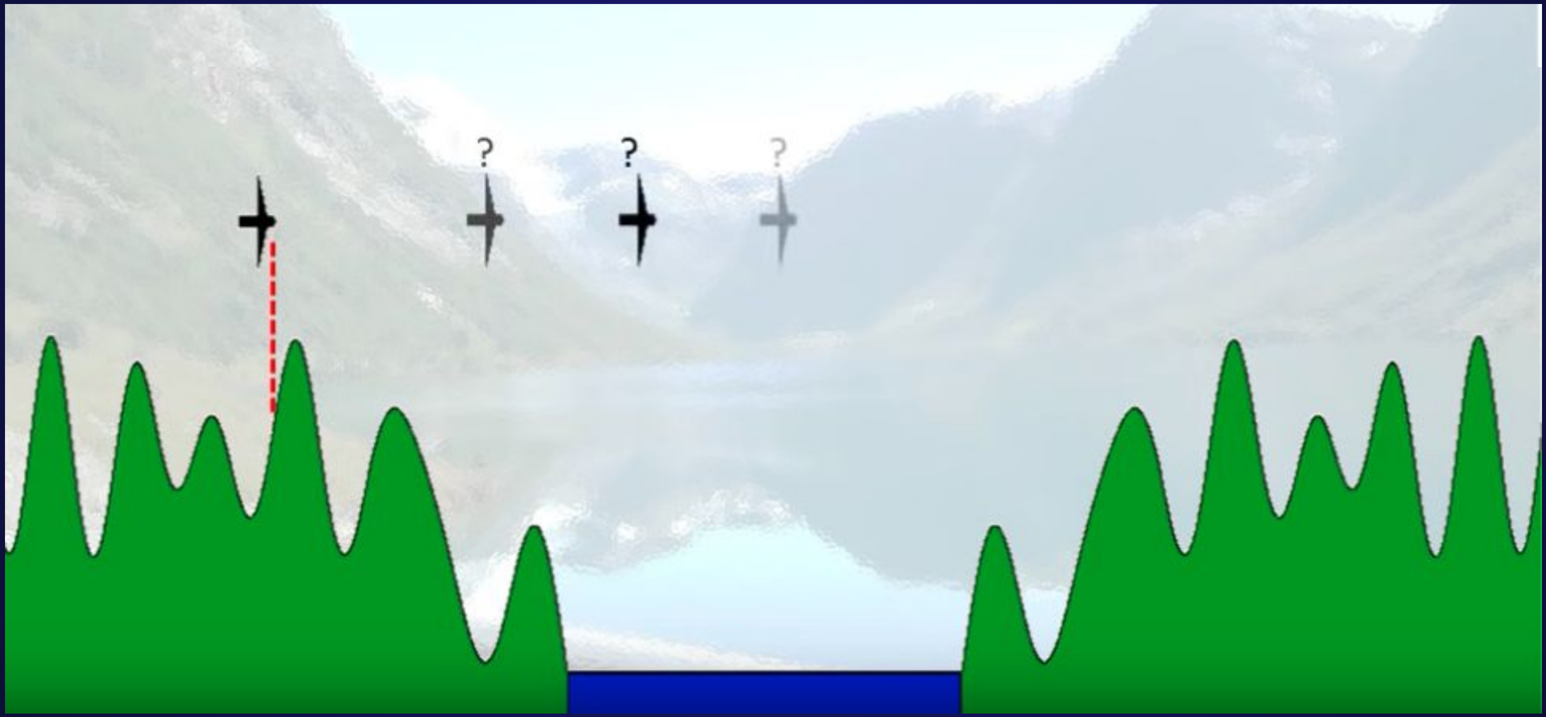
II) Path computation

III) Path following

IV) Obstacle avoidance

I) Localization

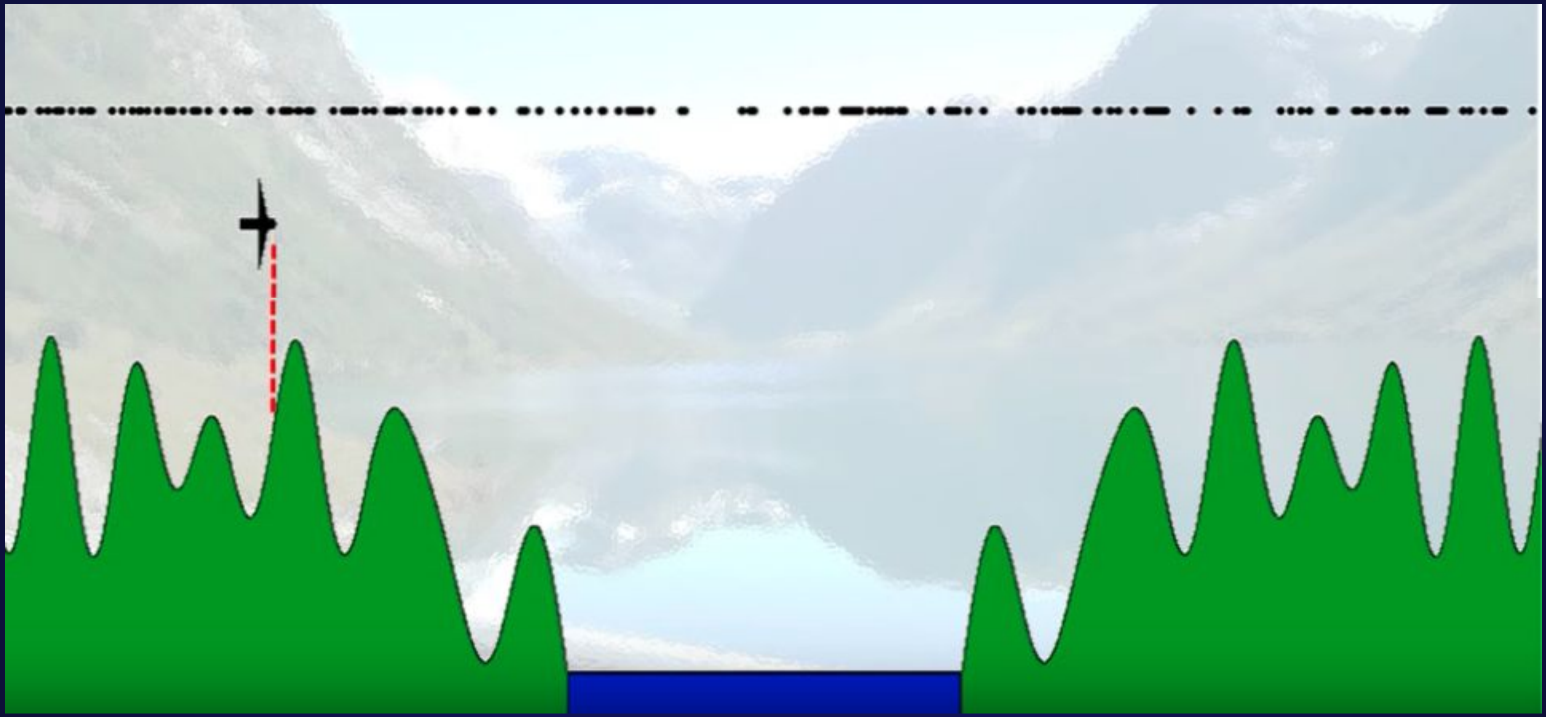
Particle filter principle



Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

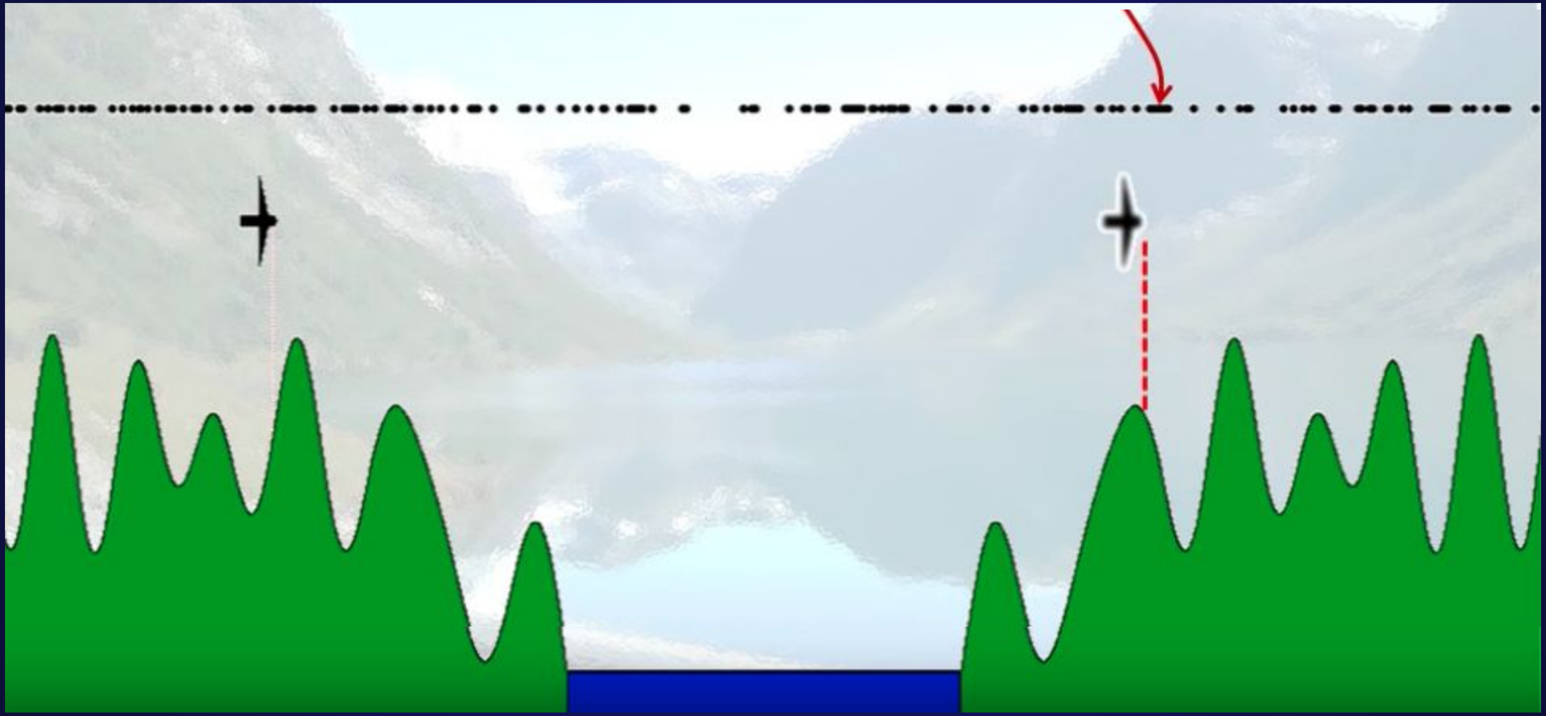


Step 1: particles generation

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

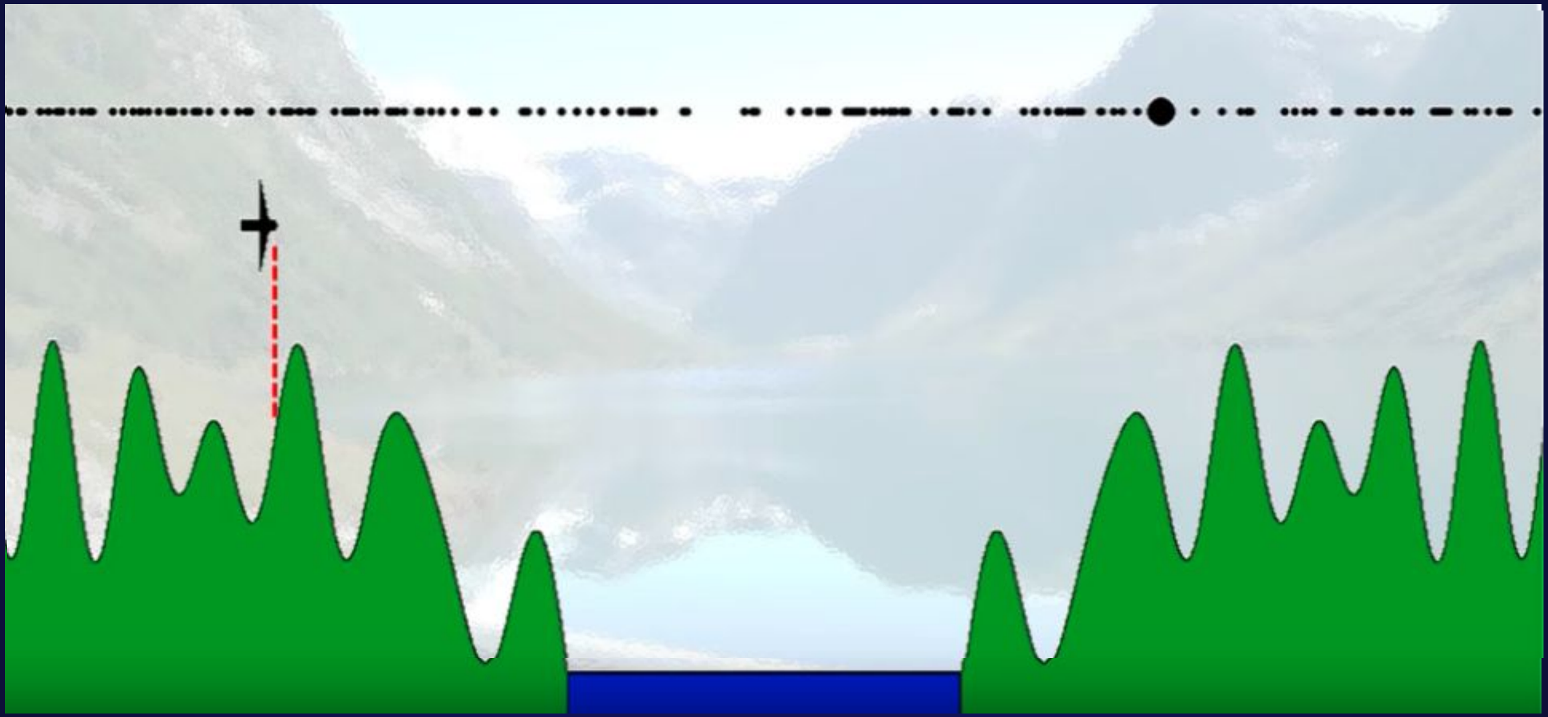


Step 2: particles weight update

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

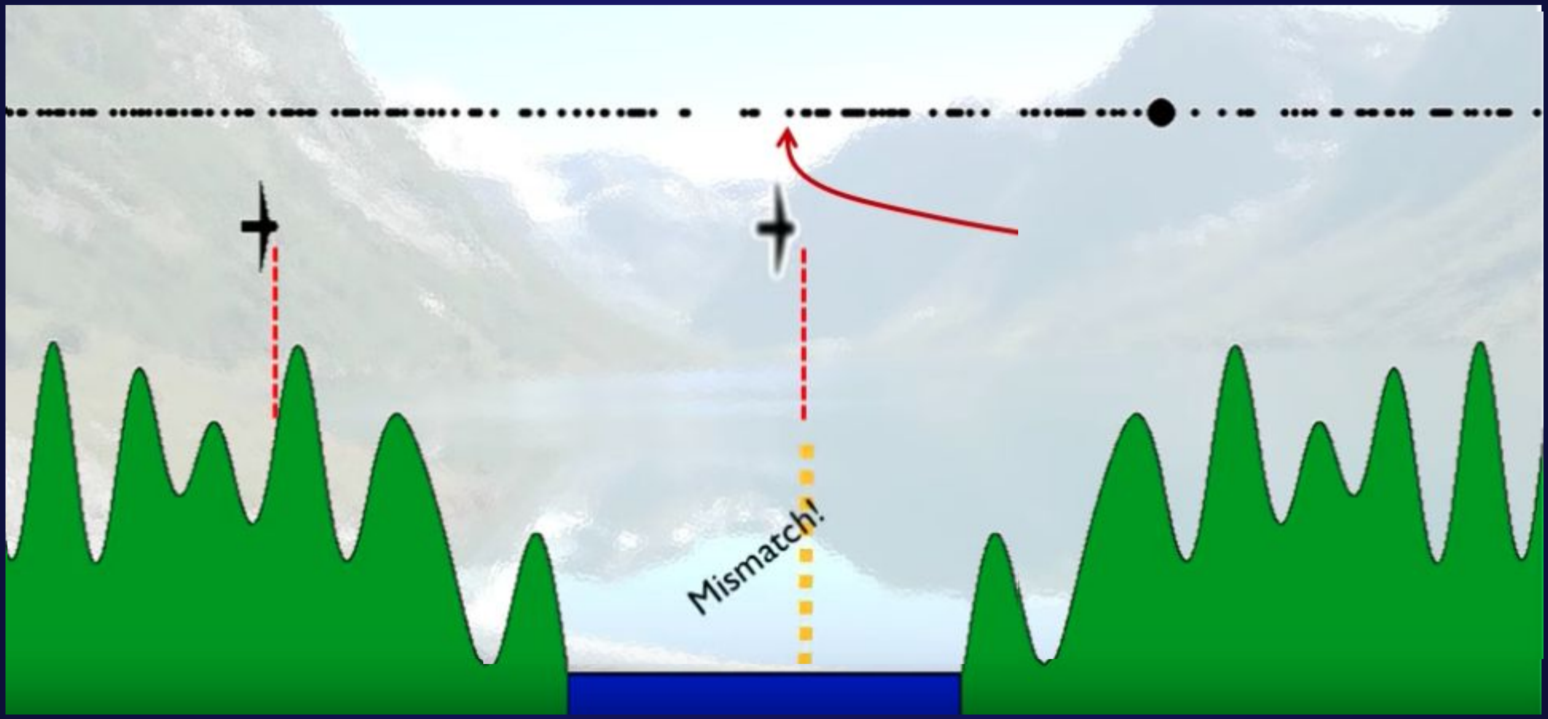


Step 2: particles weight update

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

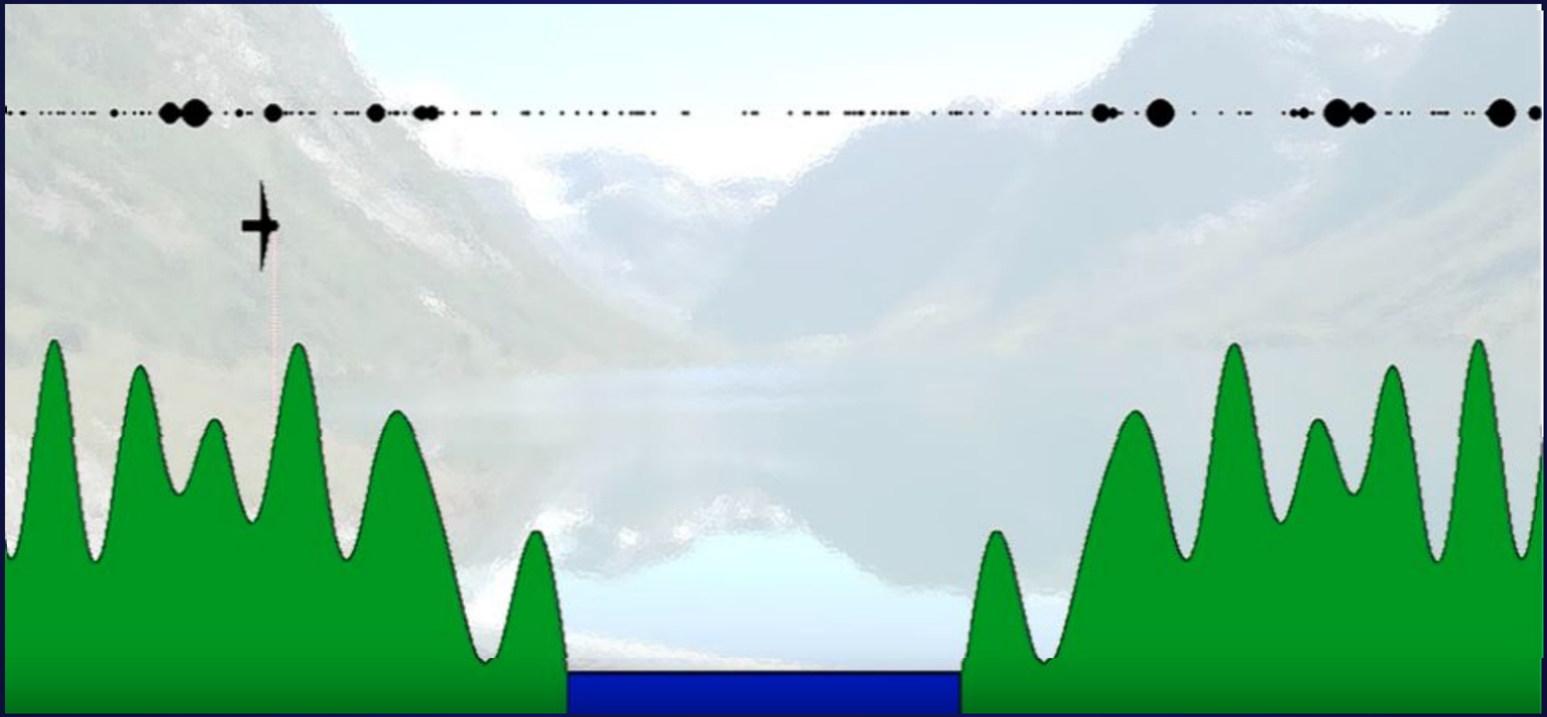


Step 2: particles weight update

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

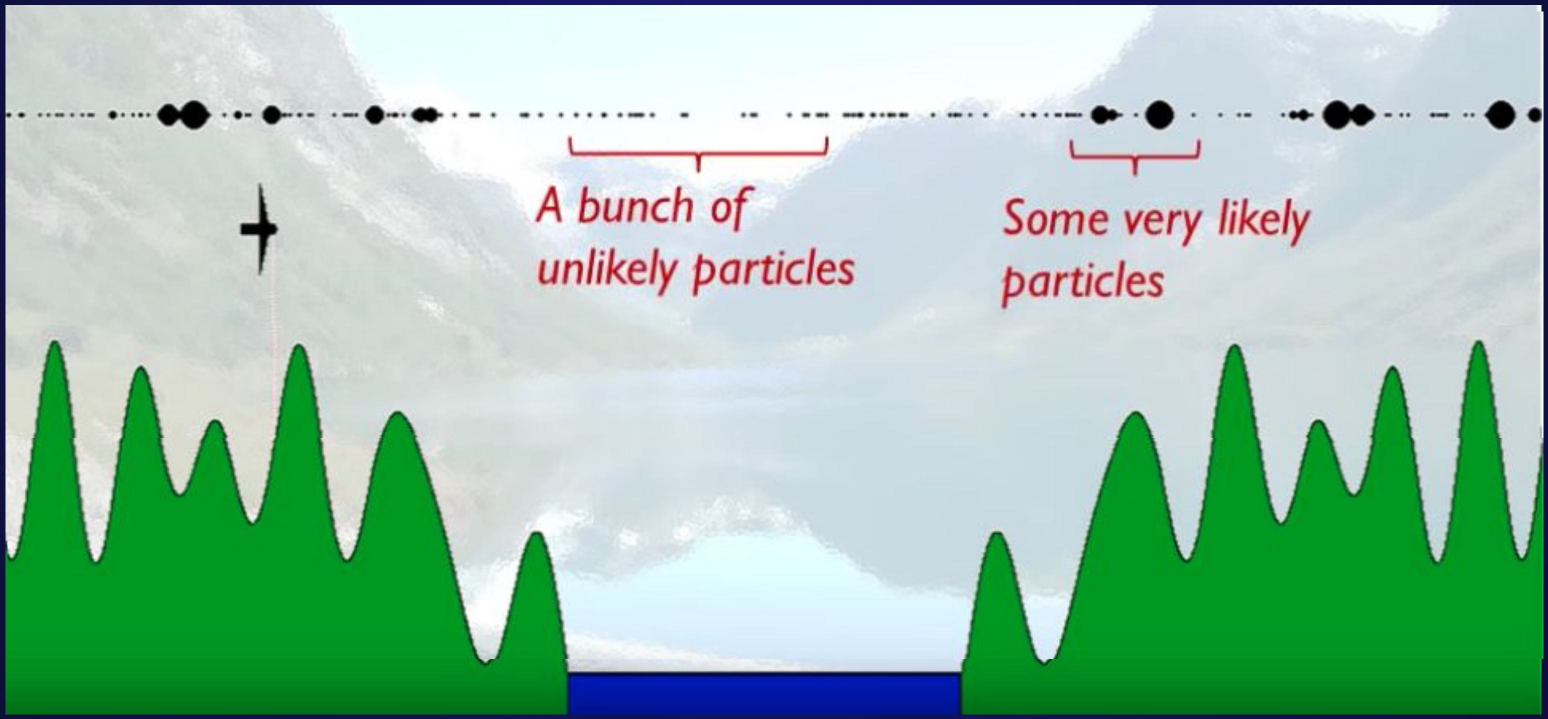


Step 2: particles weight update

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

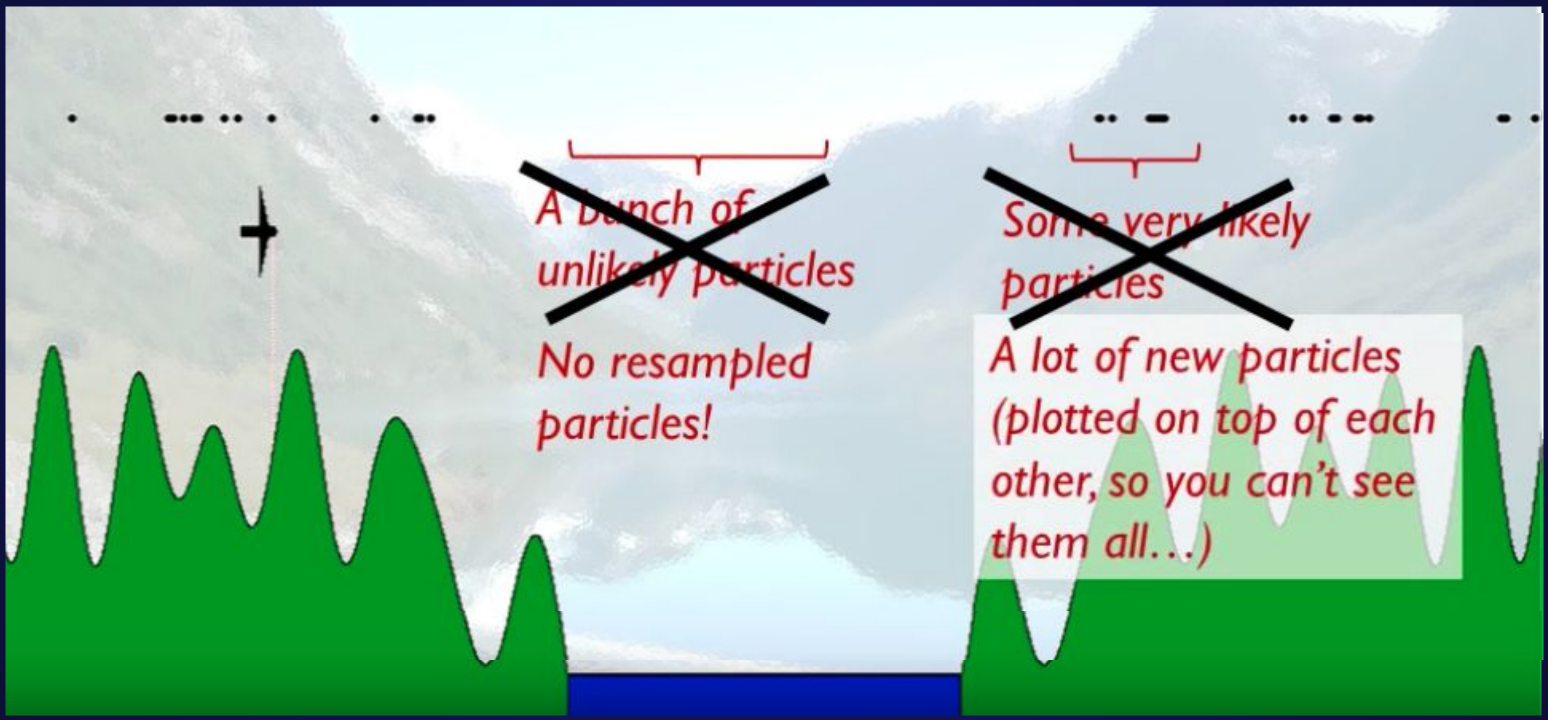


Step 2: particles weight update

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

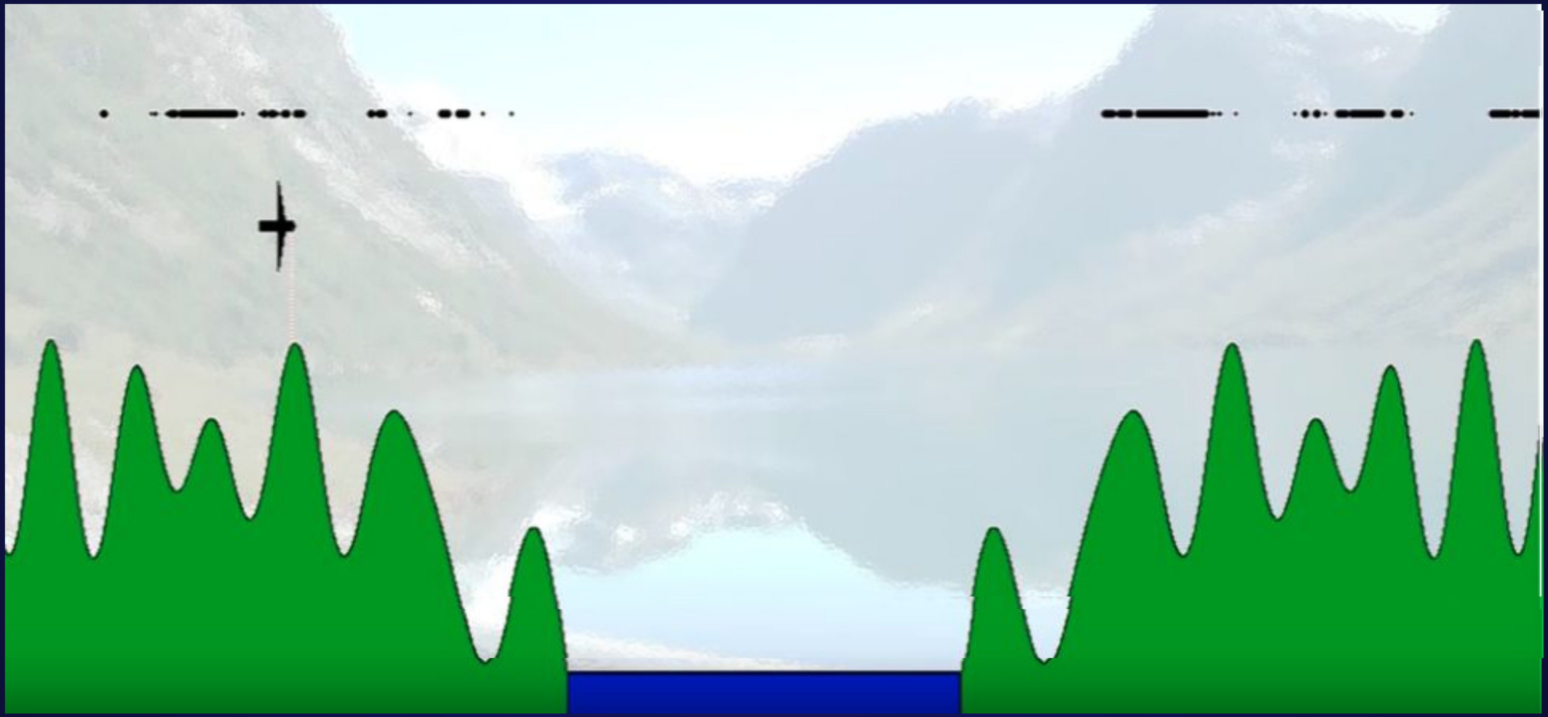


Step 3: resampling

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

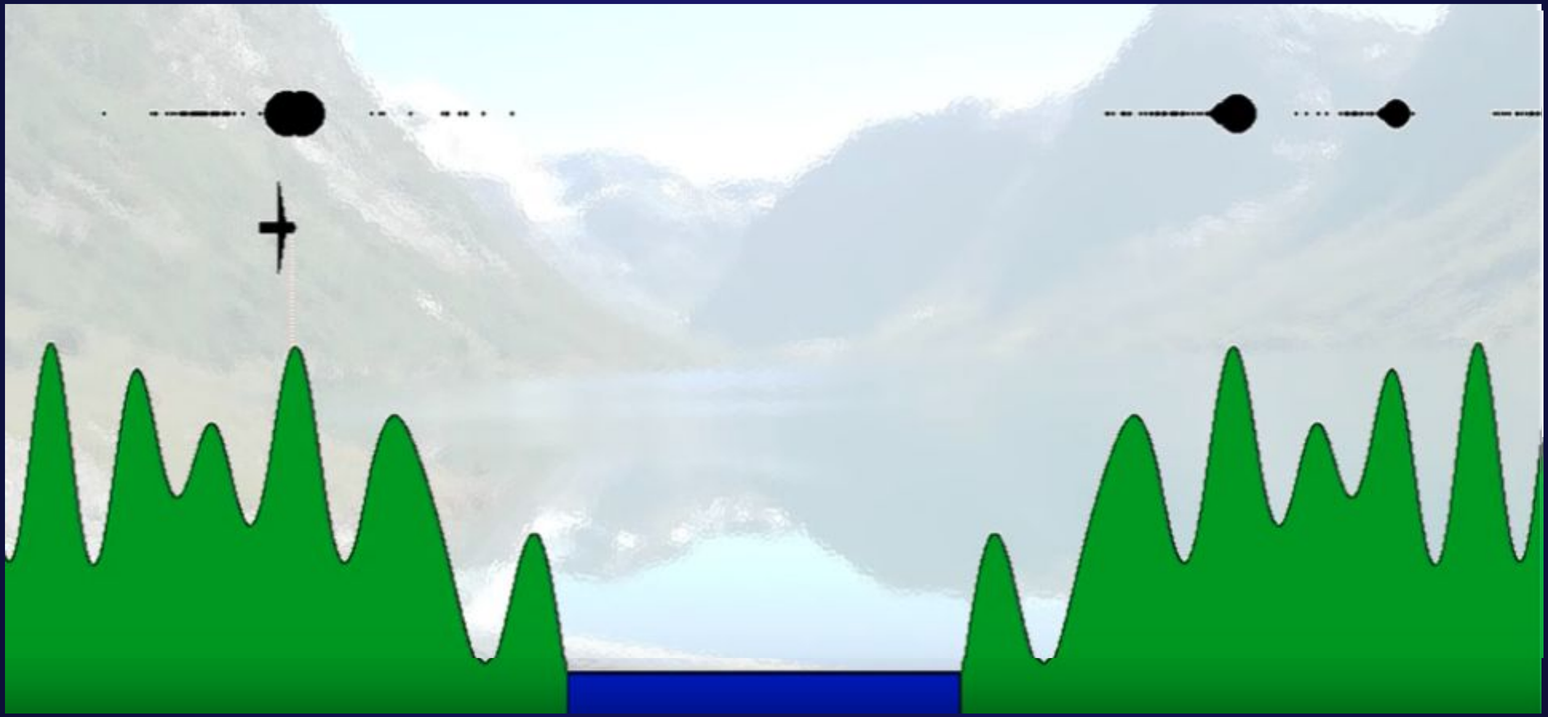


Step 4: move the particles as the vehicle moves

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

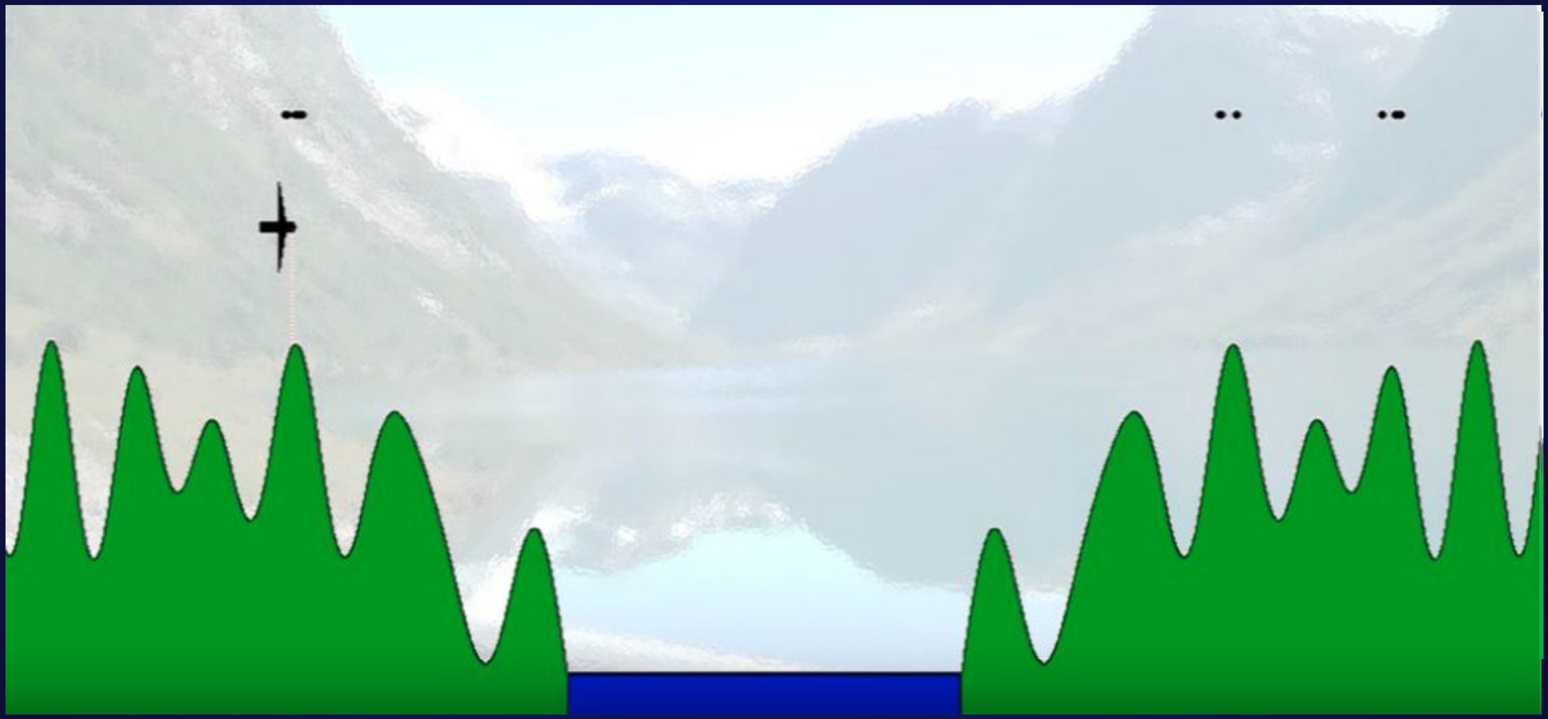


Back to step 2: update the weights

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

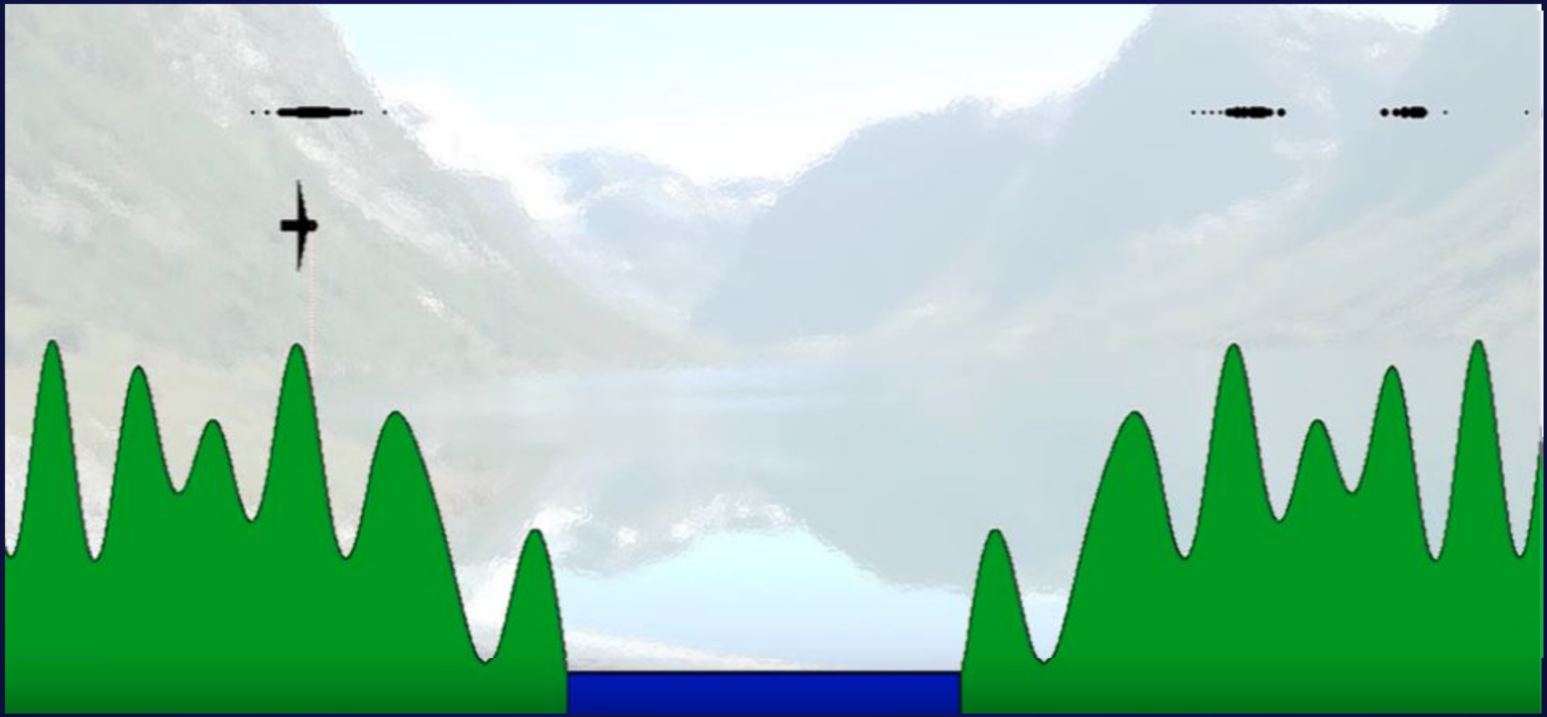


Back to step 3: resampling

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

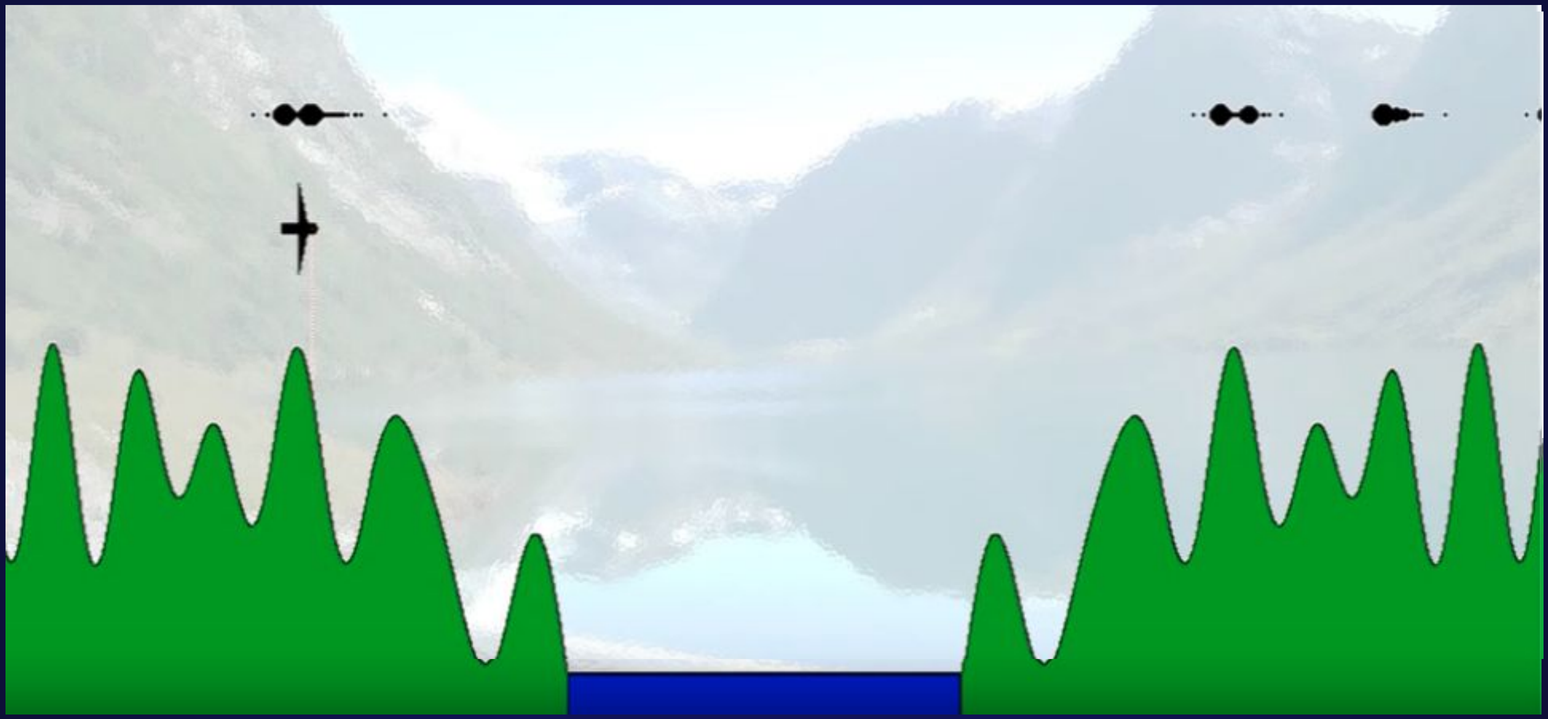


Back to step 4: move the particles

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

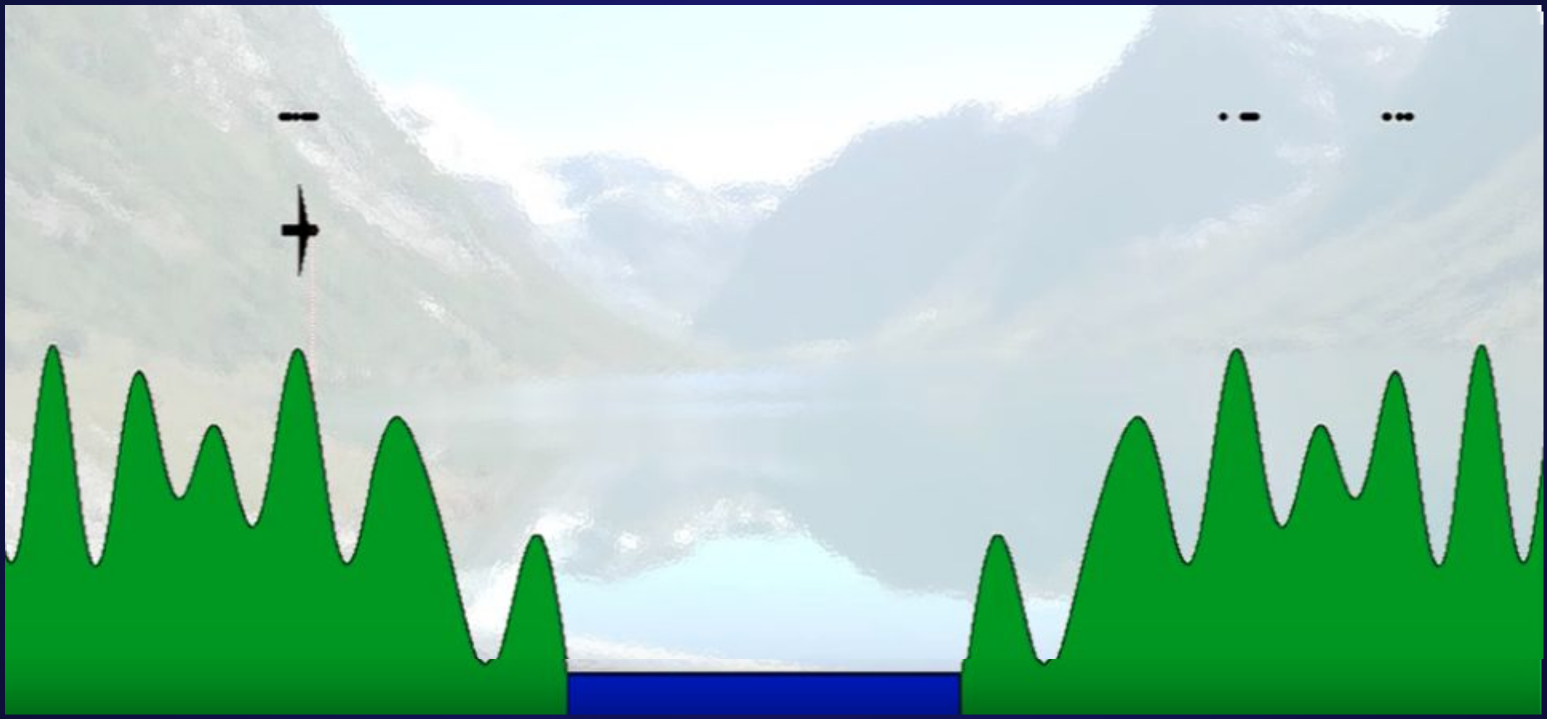


And so on...

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

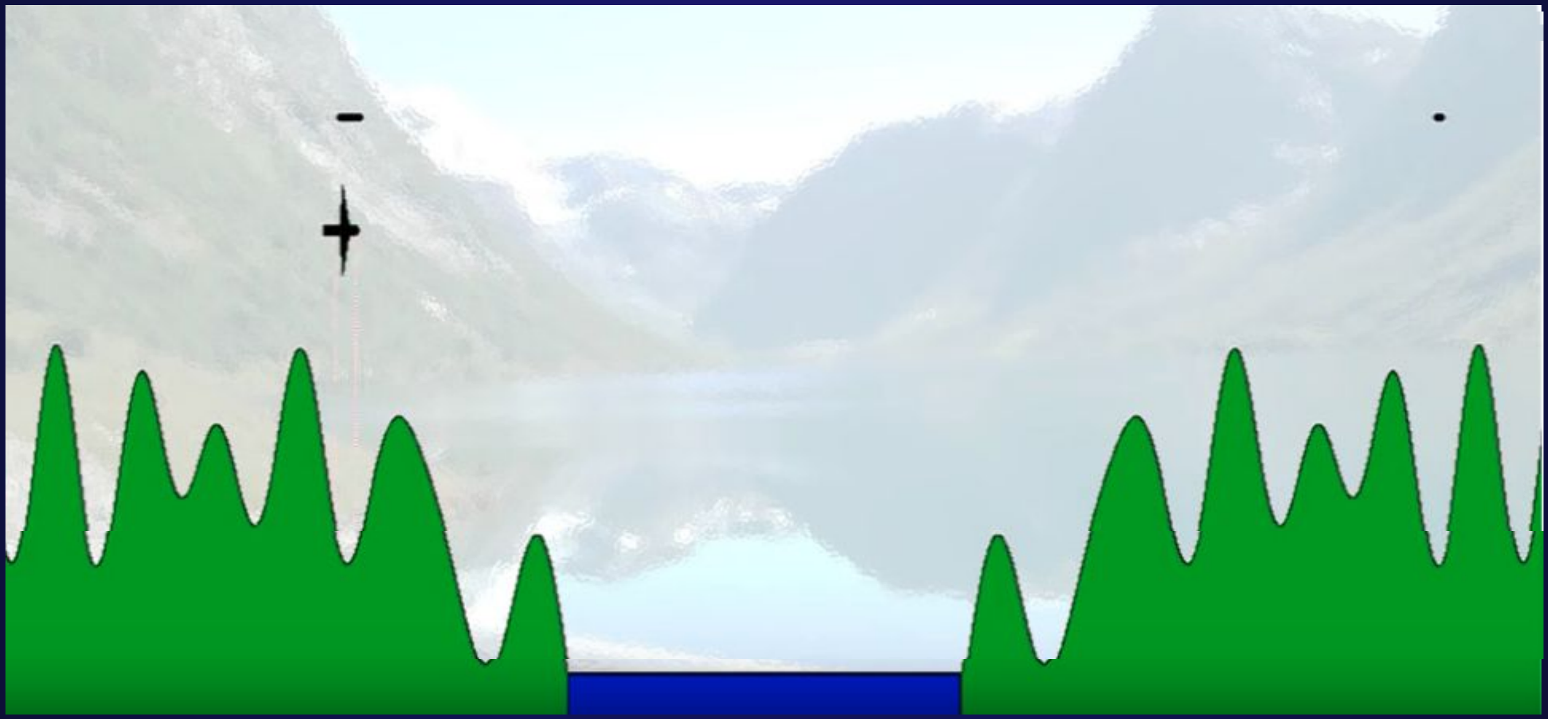


And so on...

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

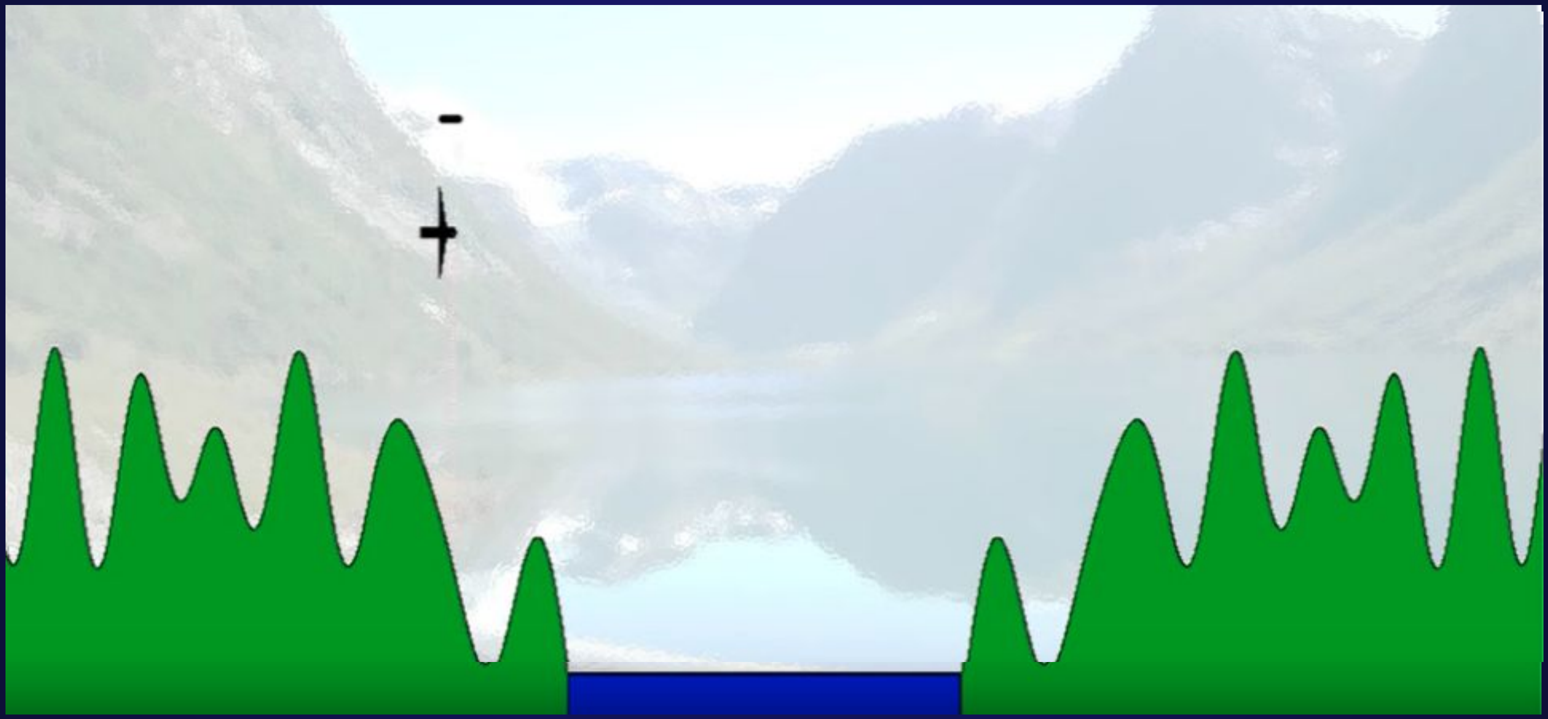


And so on...

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

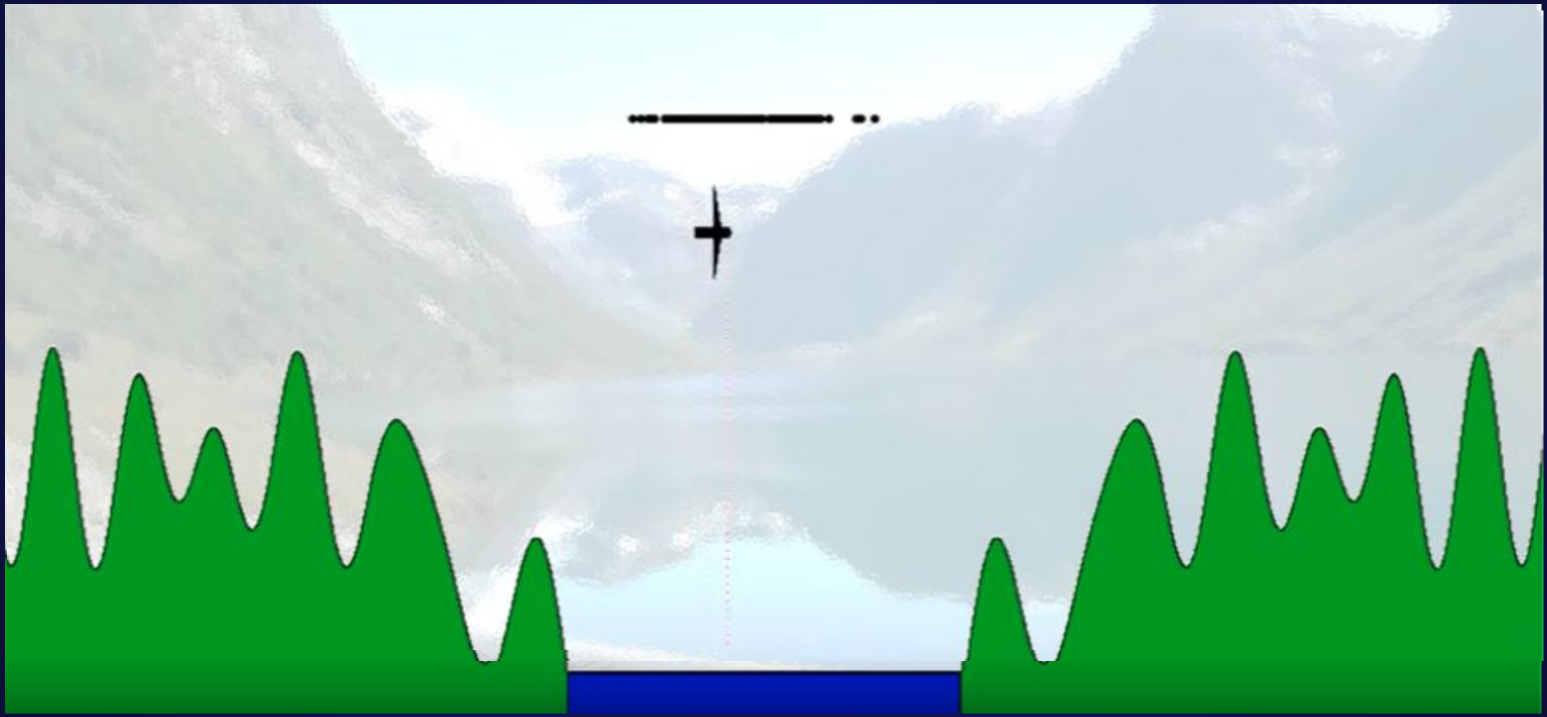


And so on...

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle

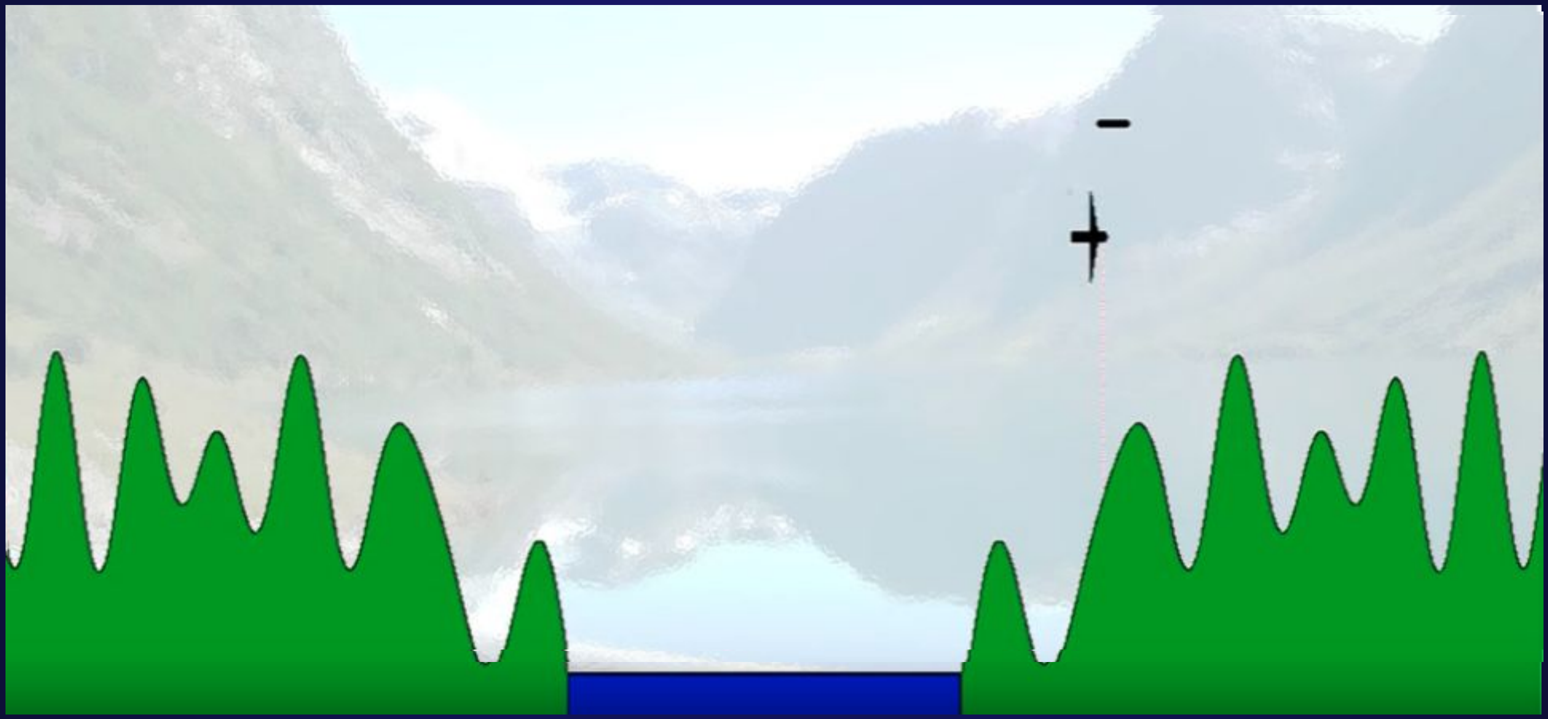


And so on...

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

Particle filter principle



And so on...

Source: https://www.youtube.com/watch?v=aUkBa1zMKv4

I) Localization

General algorithm:

1) Particles initialization
2) Weights update based on measurements
3) Resampling
4) Particles propagation Through motion model

I) Localization

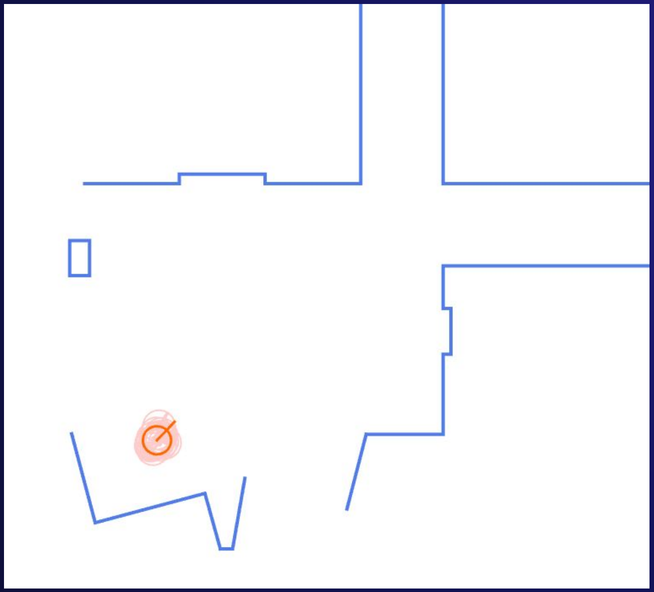
Our case, **corrective gradient refinement (CGR)**:

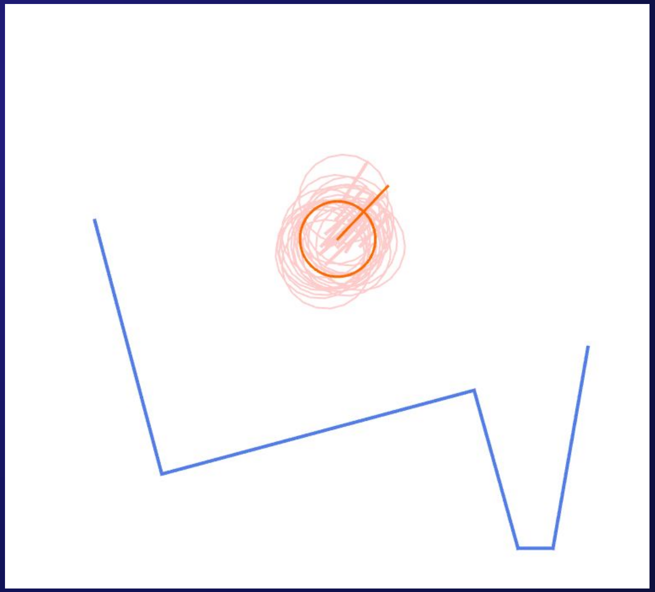
1) Particles initialization
2) Weights update based on measurements
3) Resampling
4) Particles propagation Through motion model
5) Refinement

I) Localization

Our case, **corrective gradient refinement (CGR)** :

1) Particles initialization → [x, y, orientation] , few particles, Startup point known approximately





The less particles, the faster the computation time!

I) Localization

Our case, **corrective gradient refinement (CGR)**:

2) Weights update based
   on measurements

   → a) Wall planes extraction
   b) 2D projection
   c) For each particle,
      probability of pointcloud observation

⚠ Not the only way to go, could add other types of measures (Lidar,...)

I) Localization

Our case, **corrective gradient refinement (CGR)**:

2) Weights update based
   on measurements

   → a) Wall planes extraction
   with **Fast Sampling Plane Filtering** [2] algorithm

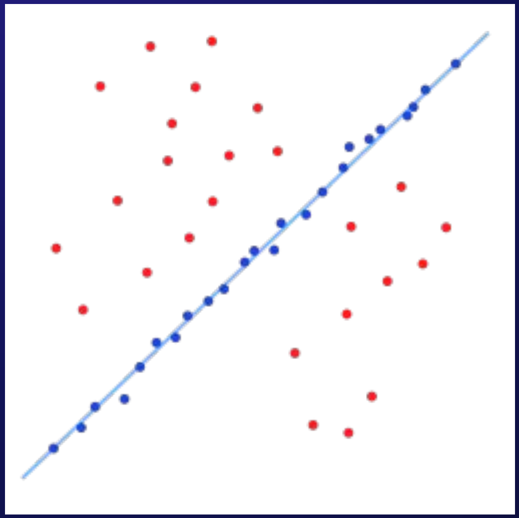
I) Localization

Our case, **corrective gradient refinement (CGR)**:

2) Weights update based on measurements

→ a) Wall planes extraction with **Fast Sampling Plane Filtering** [2] algorithm

RANSAC [3] based



I) Localization

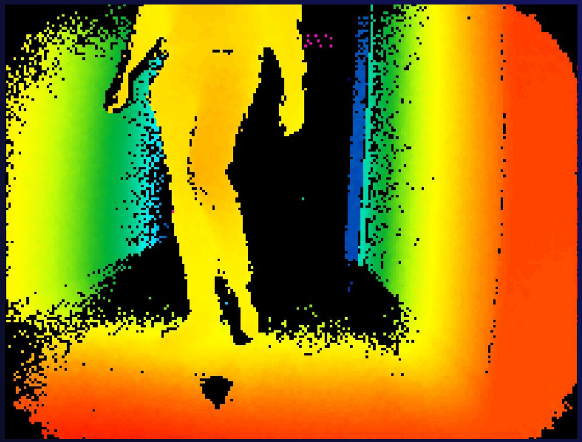
Our case, **corrective gradient refinement (CGR)**:

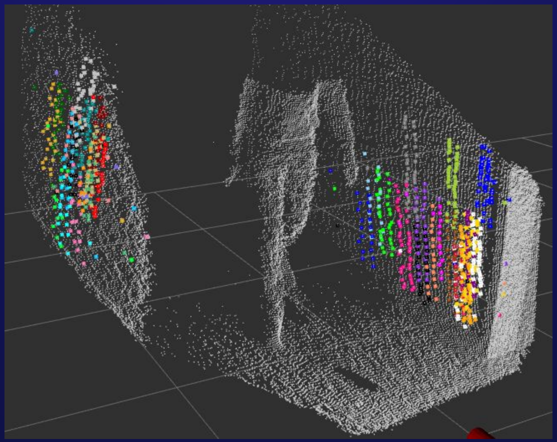
2) Weights update based on measurements

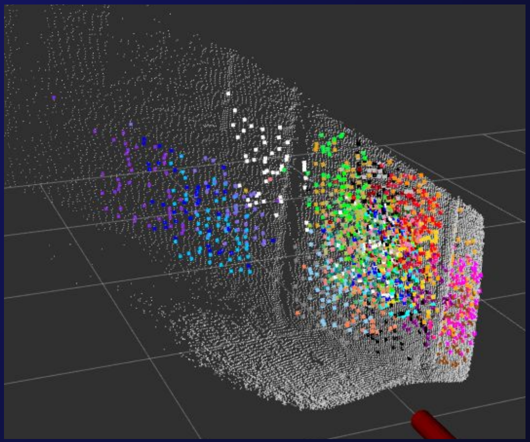
→ a) Wall planes extraction with **Fast Sampling Plane Filtering** [2] algorithm

Extraction directly from depthmap

Reduced pointclouds







I) Localization

Our case, **corrective gradient refinement (CGR)**:

2) Weights update based on measurements

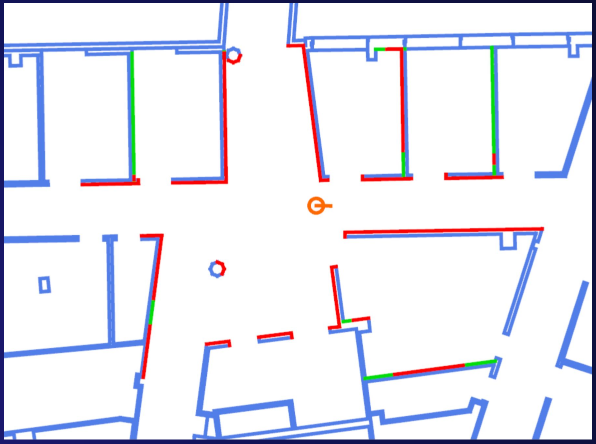
→ a) Wall planes extraction
b) 2D projection → *trivial*
c) For each particle, probability of pointcloud observation

I) Localization

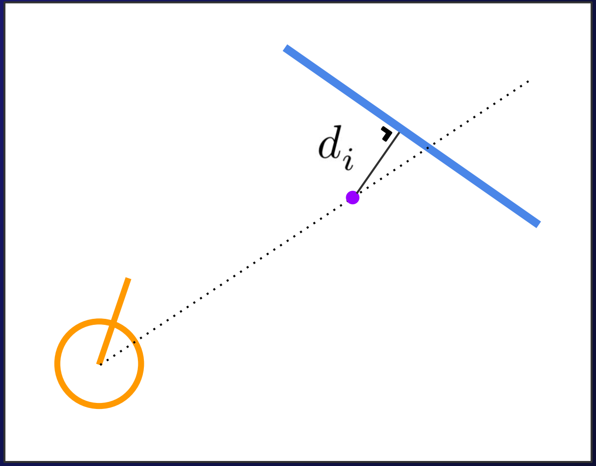
Our case, **corrective gradient refinement**:

2) Weights update based on measurements

c) For each particle, **probability** of pointcloud **observation**



$$p(y|x) = \prod_{i=1}^{n} \exp\left[-\frac{d_i^2}{2f\sigma^2}\right]$$



y = pointcloud observation
x = considered particle
n = number of points in y
$\boldsymbol{\sigma}$ = standart deviation of distance measurement
f = factor to discount for the correlation between rays

Do not consider all **n** points, need outliers rejection!

I) Localization

Our case, **corrective gradient refinement (CGR)**:

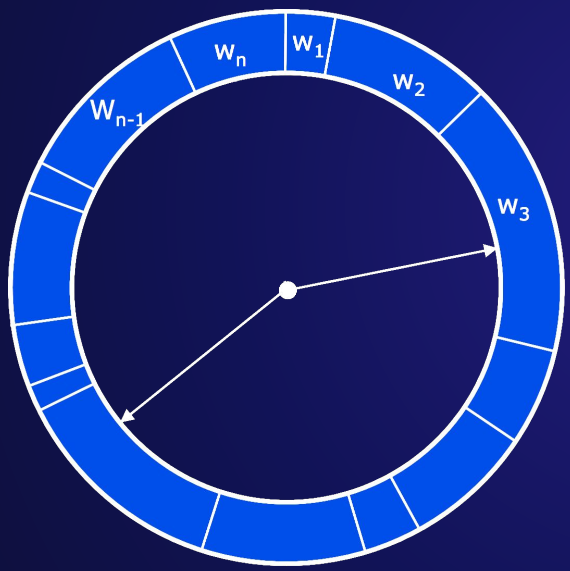
3) Resampling

Source: https://www.youtube.com/watch?v=eAqAFSrTGGY

I) Localization

Our case, **corrective gradient refinement (CGR)**:

3) Resampling

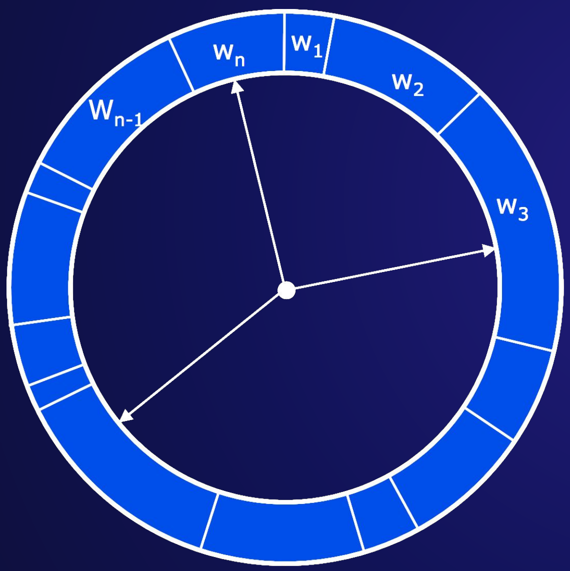


Source: https://www.youtube.com/watch?v=eAqAFSrTGGY

I) Localization

Our case, **corrective gradient refinement (CGR)**:

3) Resampling

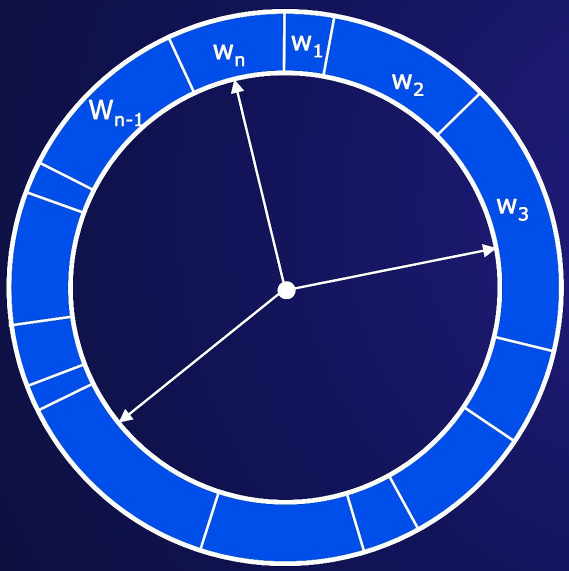


Source: https://www.youtube.com/watch?v=eAqAFSrTGGY

I) Localization

Our case, **corrective gradient refinement (CGR)**:

3) Resampling



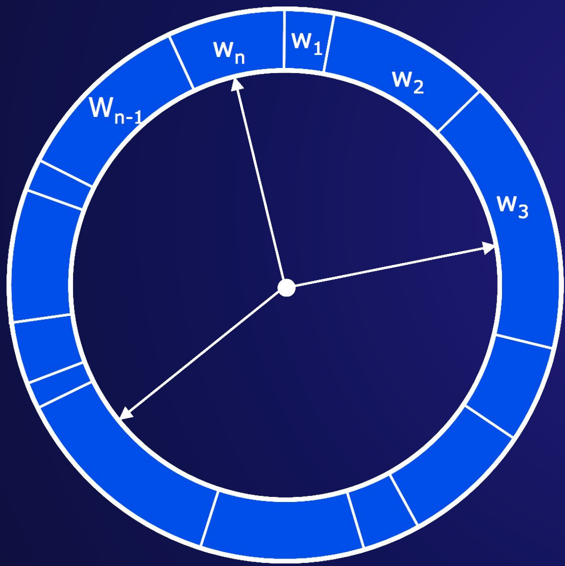
- Roulette wheel
- O(n log n)

Source: https://www.youtube.com/watch?v=eAqAFSrTGGY

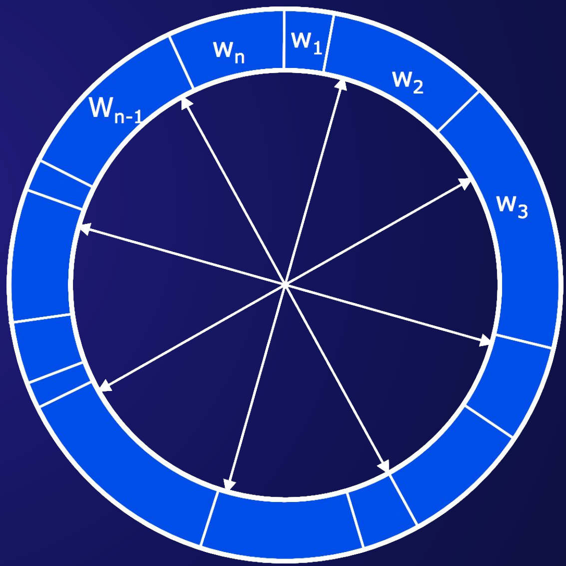
I) Localization

Our case, **corrective gradient refinement (CGR)**:

3) Resampling



VS



- Roulette wheel
- O(n log n)

- **Low variance**
- O(n)

Source: https://www.youtube.com/watch?v=eAqAFSrTGGY

I) Localization

Our case, **corrective gradient refinement**: **(CGR)**

1) Particles initialization

2) Weights update based on measurements

3) Resampling → *low variance resampling*

4) Particles propagation Through motion model

5) Refinement

I) Localization

Our case, **corrective gradient refinement (CGR)**:

1) Particles initialization

2) Weights update based on measurements

3) Resampling → *low variance resampling*

4) Particles propagation Through motion model → *Odometry*

5) Refinement

I) Localization

Our case, **corrective gradient refinement (CGR)**:

5) Refinement → **Correcting** sample estimates that contradicts the observation

I) Localization

Our case, **corrective gradient refinement (CGR)**:

5) Refinement → **Correcting** sample estimates that contradicts the observation

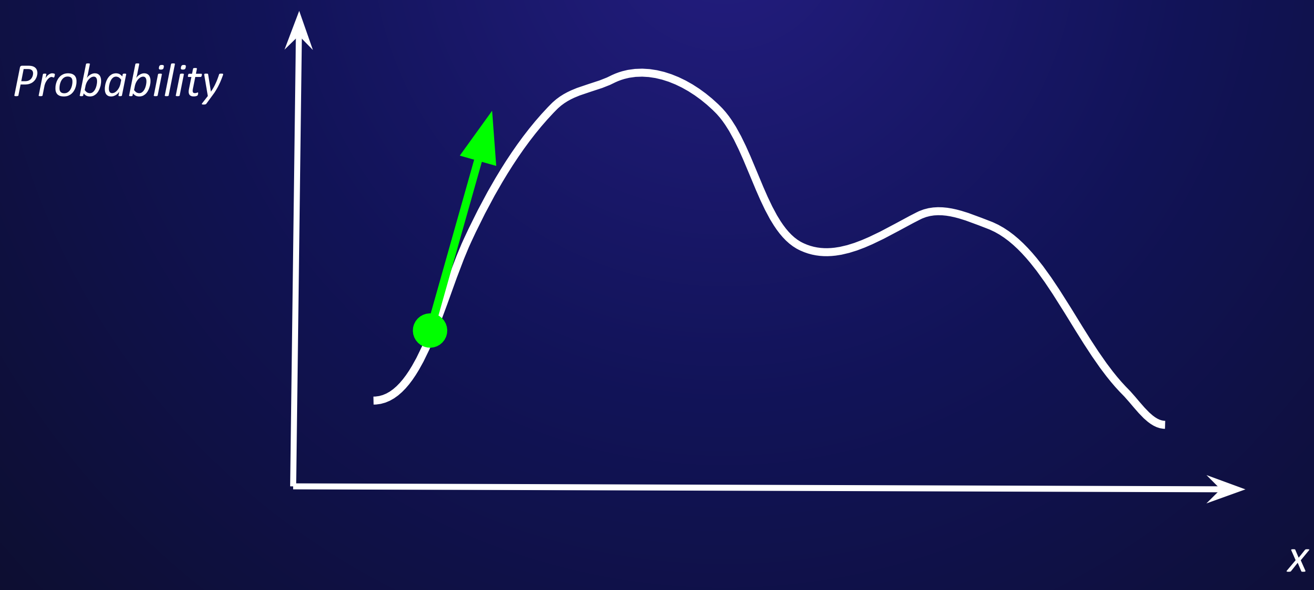
→ Using the **gradient** of the probability of the observation

I) Localization

Our case, **corrective gradient refinement (CGR)**:

5) Refinement → **Correcting** sample estimates that contradicts the observation

→ Using the **gradient** of the probability of the observation

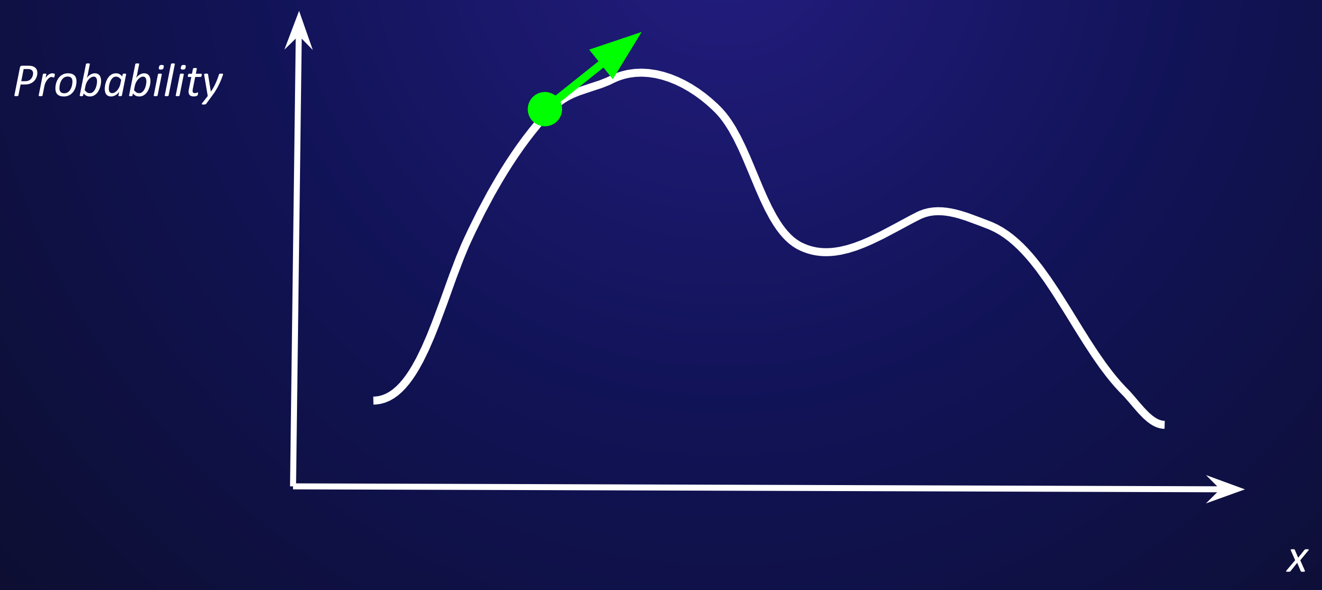


I) Localization

Our case, **corrective gradient refinement (CGR)**:

5) Refinement → **Correcting** sample estimates that contradicts the observation

→ Using the **gradient** of the probability of the observation

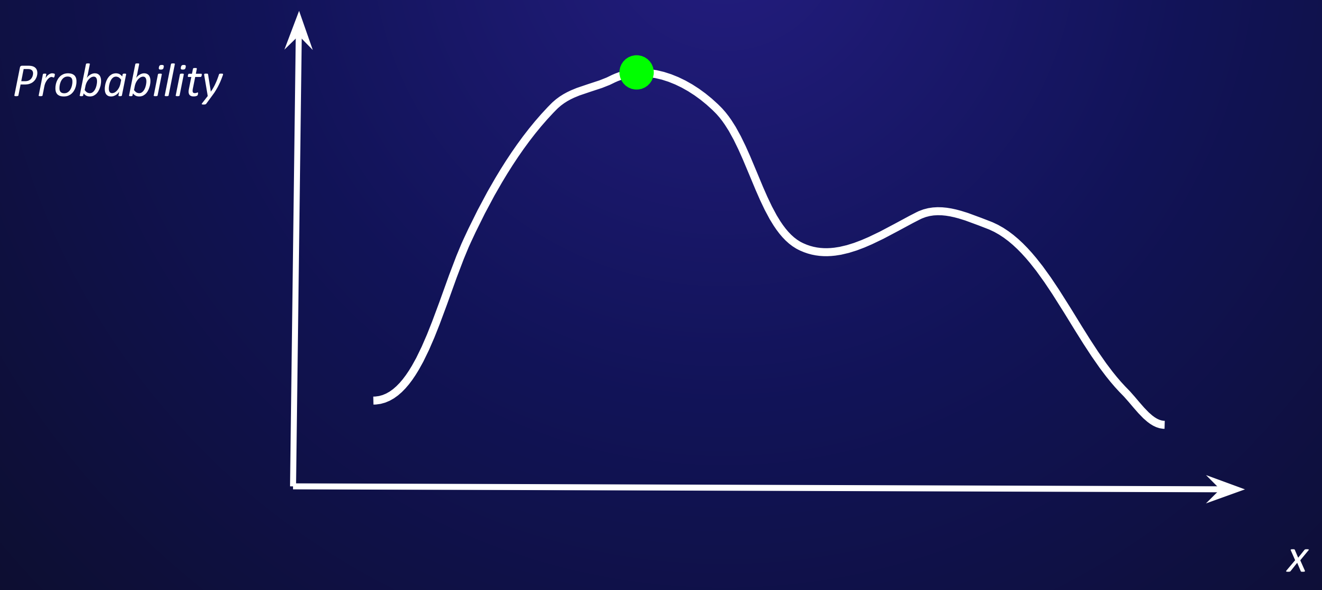


I) Localization

Our case, **corrective gradient refinement (CGR)**:

5) Refinement → **Correcting** sample estimates that contradicts the observation

→ Using the **gradient** of the probability of the observation

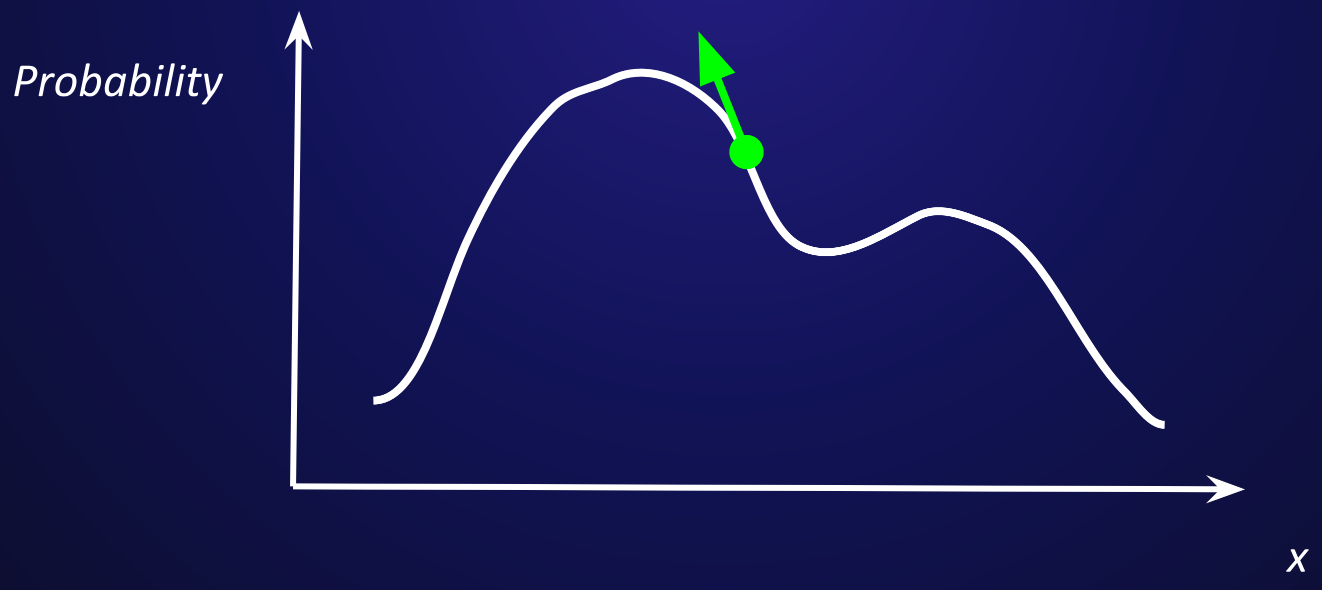


I) Localization

Our case, **corrective gradient refinement (CGR)**:

5) Refinement → **Correcting** sample estimates that contradicts the observation

→ Using the **gradient** of the probability of the observation



I) Localization

Our case, **corrective gradient refinement (CGR)**:

5) Refinement → **Correcting** sample estimates that contradicts the observation

→ Using the **gradient** of the probability of the observation

Algorithm 1 The Refine step of CGR

Let $q^0 = \left\{ x^j_{q^0} \right\}_{j=1:m}$
for $i = 1$ to r **do**
    $q^i \leftarrow \{\}$
    for $j = 1$ to m **do**
        $x^j_{q^i} \leftarrow x^j_{q^{i-1}} + \eta \left[\frac{\hat{\delta}}{\delta x} p(y_t|x) \right]_{x = x^j_{q^{i-1}}}$
        $q^i \leftarrow q^i \cup x^j_{q^i}$
    end for
end for

I) Localization

Our case, **corrective gradient refinement (CGR)**:

5) Refinement → **Correcting** sample estimates that contradicts the observation

→ Using the **gradient** of the probability of the observation

→ **Acceptance test** to be sure that the correction did not make it worse

Why choose CGR particle filter?

- Particle filters good for **non-linear systems**
- Particle filters work for any **arbitrary noise distribution**
  VS kalman filters work for gaussian noise
- Fit our needs & sensors (**Depth camera** based localization)
- Computation **speed**
- **Source code** available

In practice...

- **Embedding** ROS c++ code into Loomo
  - **J**ava **N**ative **I**nterface
  - Limited debugging tools
  - Not enough **computing power**

In practice...

- CGR running on **distant machine**
  - Odometry & depth maps sent over **WIFI**
  - **Latency**
  - Bandwidth overload
  - Issues when **switching** between **hotspots**
  - Not reliable

In practice...

Solution:

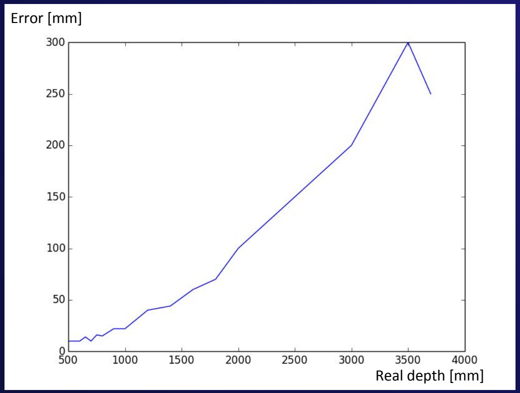
adding computing power

Mini pc

Battery

In practice...

- Embedded Depth camera
  - Intel realsense
  - Very **noisy output**
  - Bad **accuracy**



→ Need to be close to the walls

In practice...

- Embedded Depth camera Usable under **some conditions**:
  - Be **close to** the **walls**
  - **Adapt** head **orientation** in some areas
  - **Adapt speed** in some areas
  - No **large hall** crossing

In practice...

Solution:

Better depth camera

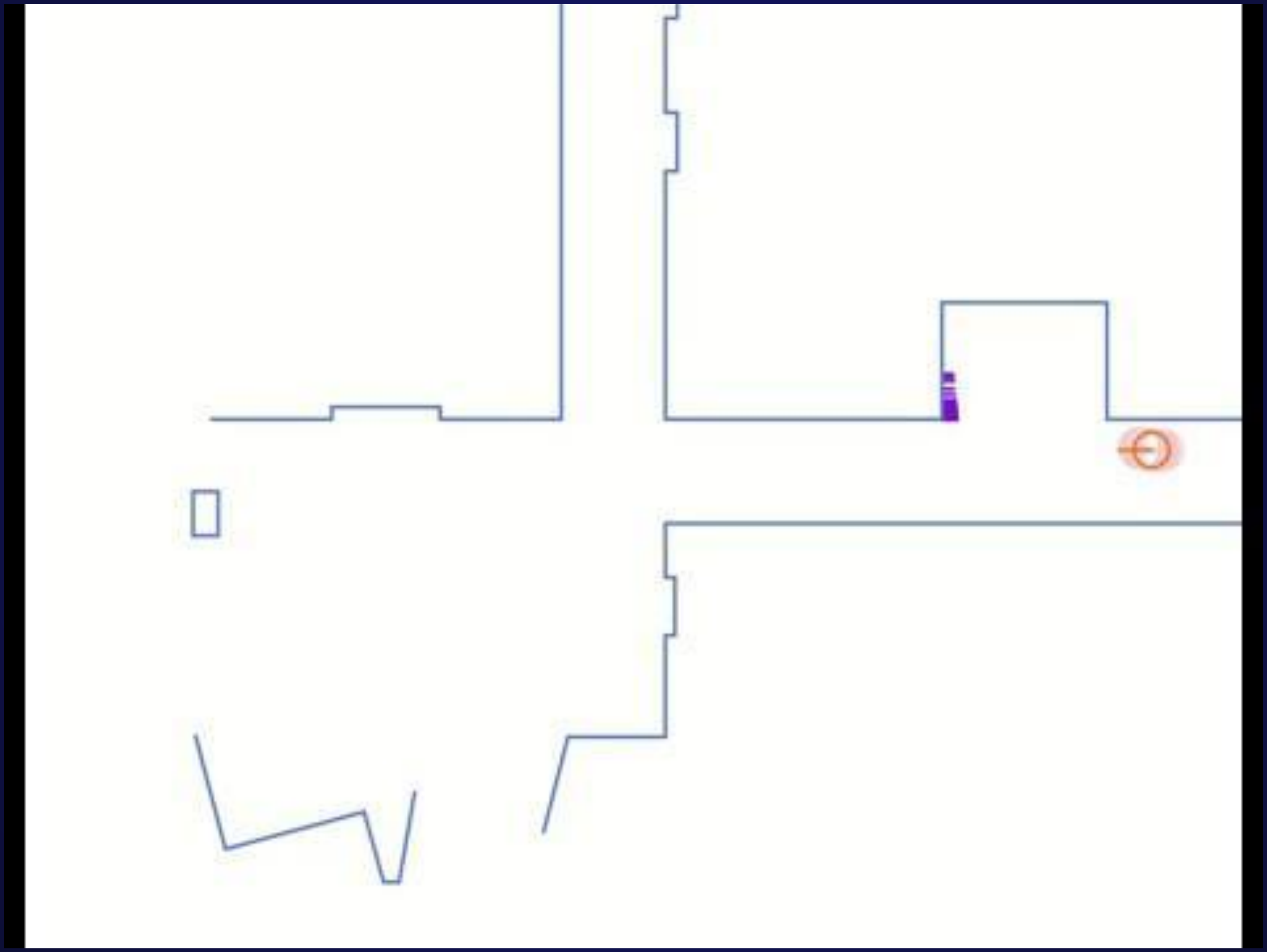
PMD picoflexx

Time-Of-Flight camera

+-1cm error at 4m depth

Cheap

CGR in action



Navigation system

Our solutions:

I) Localization

→ Particle filter : corrective gradient refinement

II) Path computation

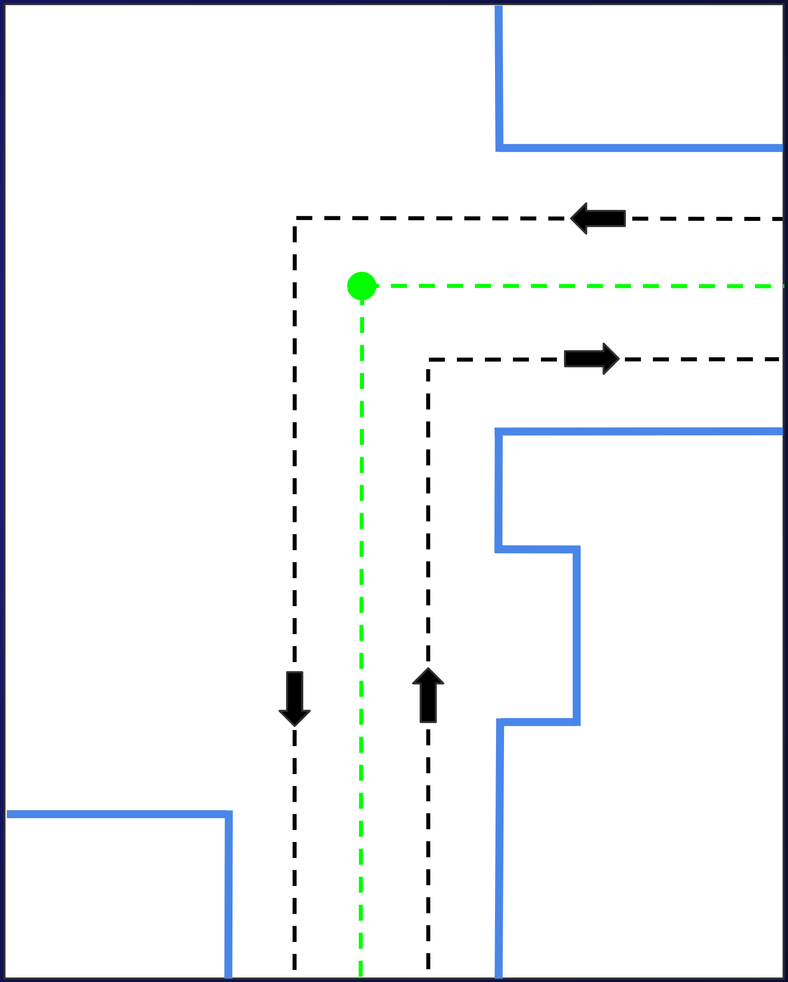
→ Hardcoded trajectories

III) Path following

IV) Obstacle avoidance

II) Path computation

- **Hardcoded** base trajectories (centered)
- Corridors divided into **2 aisles**
- **Automatic computation** of aisles paths from base trajectory
- Pro’s & con’s:
  + fast, simple, **control over trajectory**
  - not automatically adaptable
    to a new, bigger building



Navigation system

Our solutions:

I) Localization

→ Particle filter : corrective gradient refinement

II) Path computation

→ Hardcoded trajectories

III) Path following

→ PD controller

IV) Obstacle avoidance

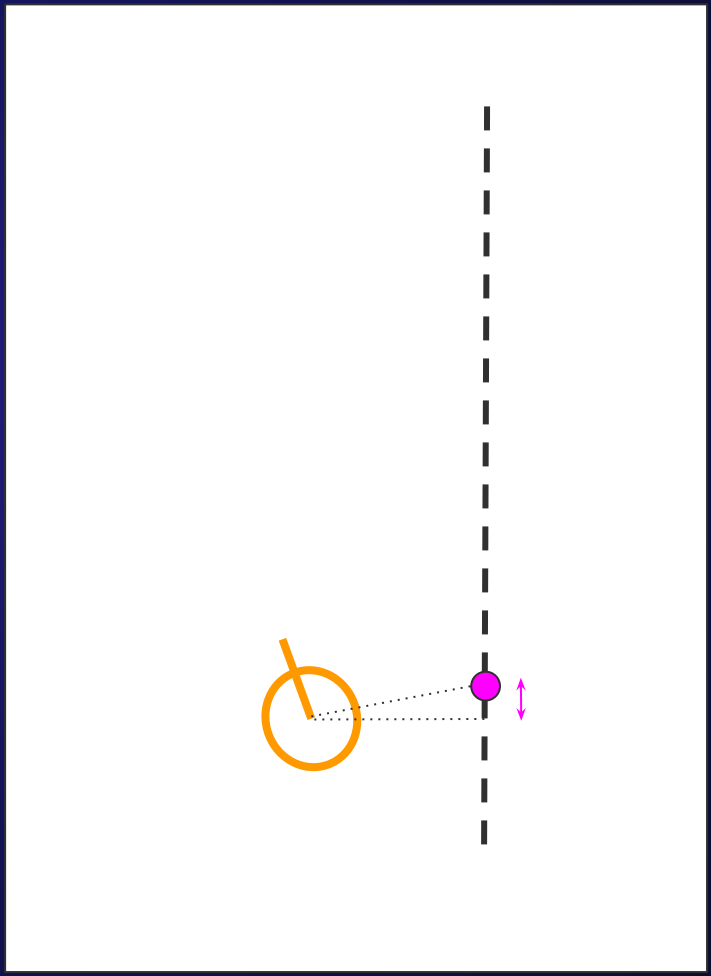
III) Path following

- 2 controls:
  - **linear** velocity
  - **angular** velocity

- **Pure pursuit**

- Proportionnal & Derivative (PD) **controller** for **angle** towards destination

- Constant linear velocity



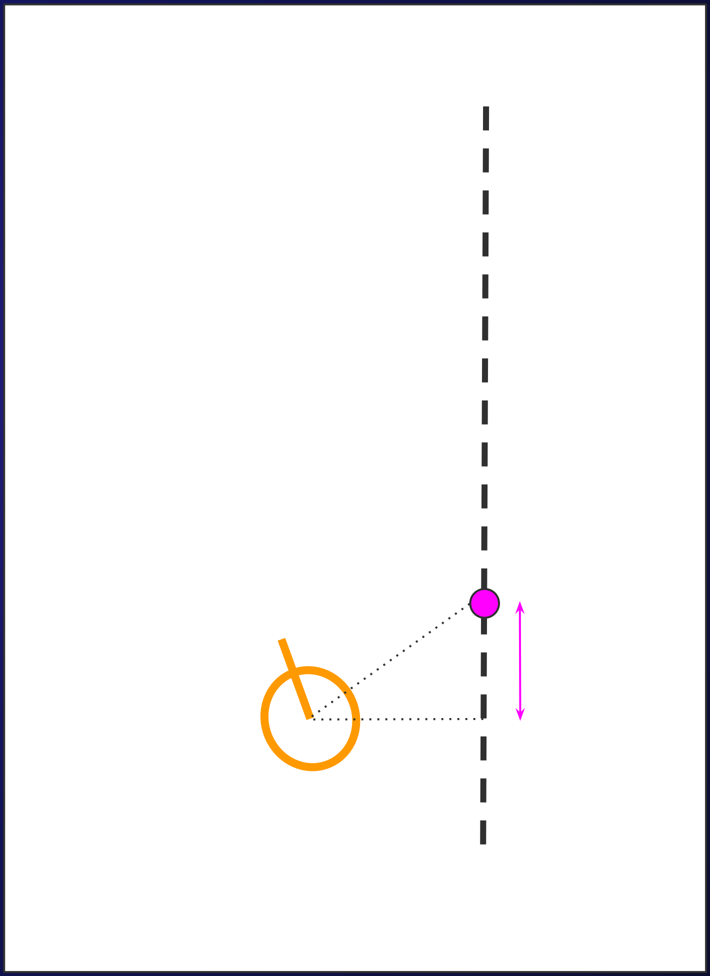
III) Path following

- 2 controls:
  - **linear** velocity
  - **angular** velocity

- **Pure pursuit**

- Proportionnal & Derivative (PD) **controller** for **angle** towards destination

- Constant linear velocity



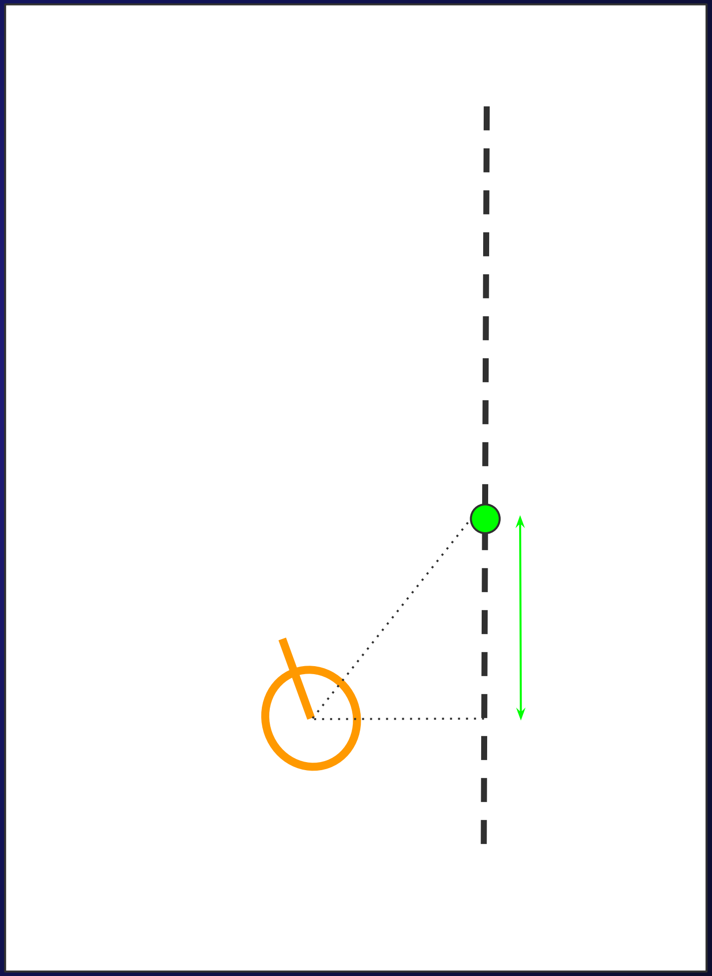
III) Path following

- 2 controls:
  - **linear** velocity
  - **angular** velocity

- **Pure pursuit**

- Proportionnal & Derivative (PD) **controller** for **angle** towards destination

- Constant linear velocity



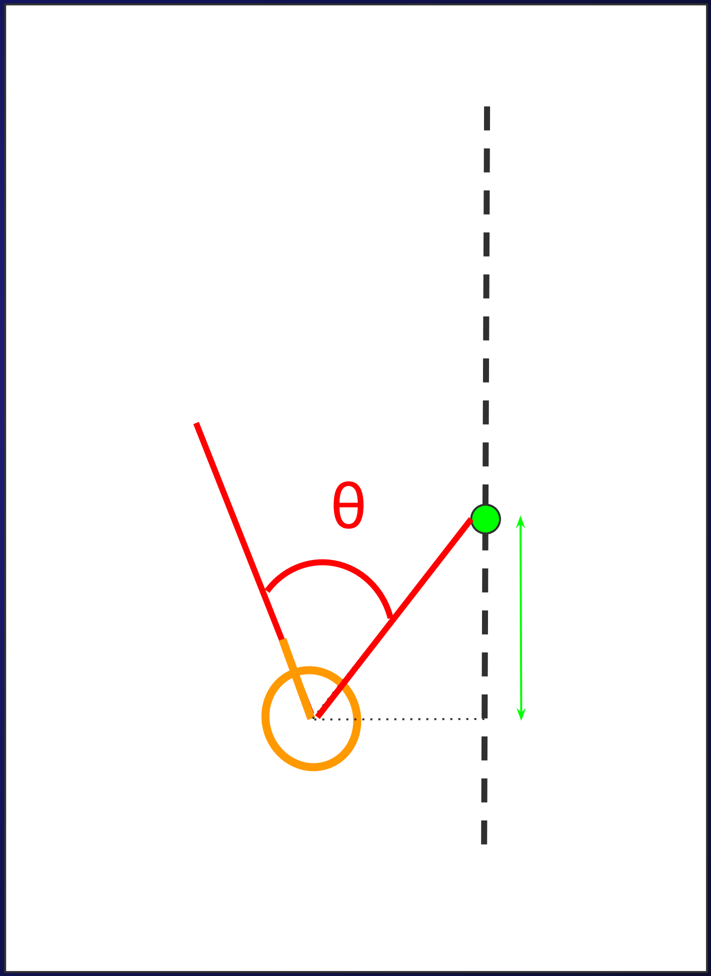
III) Path following

- 2 controls:
  - **linear** velocity
  - **angular** velocity

- **Pure pursuit**

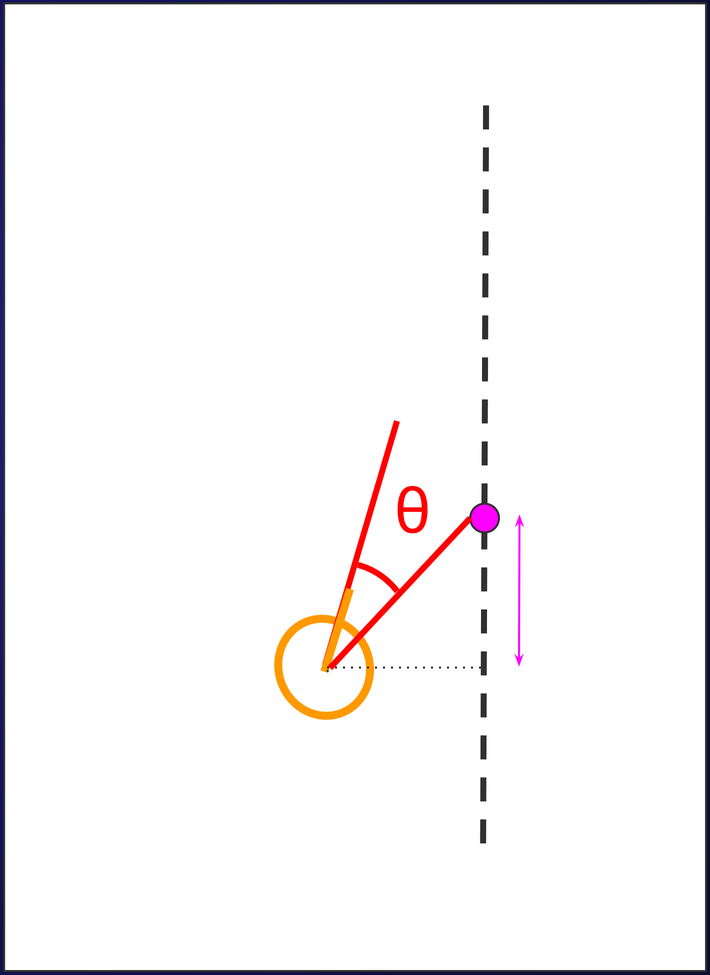
- Proportionnal & Derivative (PD) **controller** for **angle θ** towards destination

- Constant linear velocity



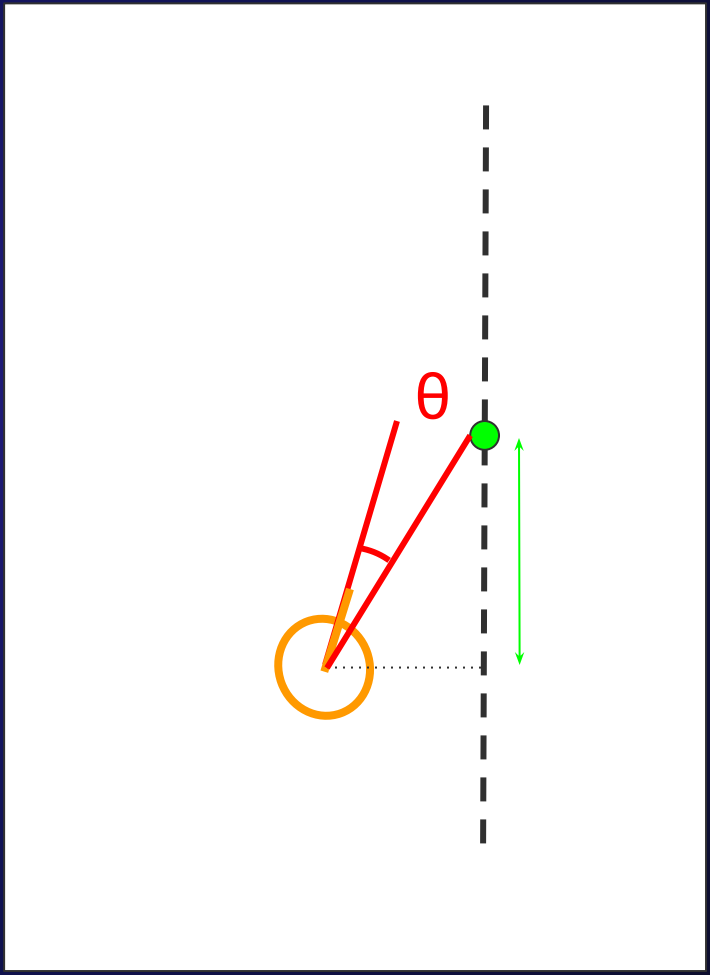
III) Path following

- 2 controls:
  - **linear** velocity
  - **angular** velocity
- **Pure pursuit**
- Proportionnal & Derivative (PD) **controller** for **angle θ** towards destination
- Constant linear velocity



III) Path following

- 2 controls:
  - **linear** velocity
  - **angular** velocity
- **Pure pursuit**
- Proportionnal & Derivative (PD) **controller** for **angle θ** towards destination
- Constant linear velocity

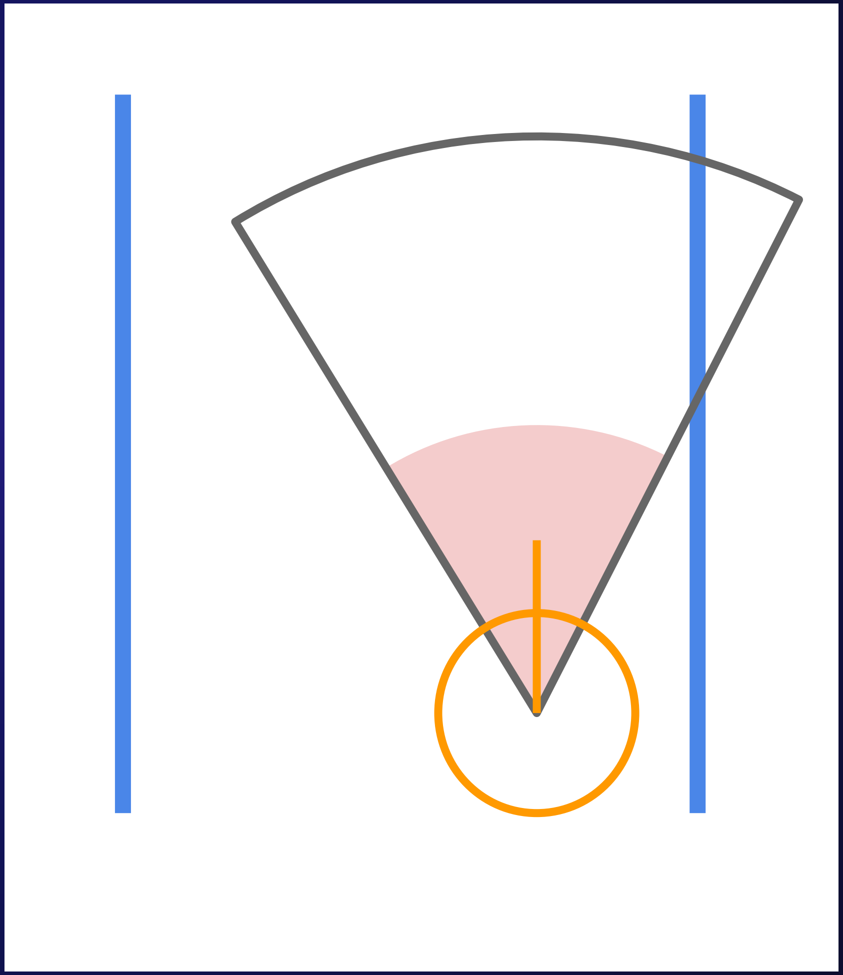


III) Obstacle avoidance

- **Ultrasonic** sensor
- **Stop** & wait
- Short **threshold**
- **Limit** max speed of robot!

➡ **Need a more intelligent way**

(Depth cam based, obstacles extraction, dynamic avoidance)



Sources

[1] Biswas, Joydeep, Brian Coltin, and Manuela Veloso. "Corrective gradient refinement for mobile robot localization." 2011 IEEE/RSJ international conference on Intelligent Robots and Systems. IEEE, 2011.

[2] Biswas, Joydeep, and Manuela Veloso. "Fast sampling plane filtering, polygon construction and merging from depth images." RSS, RGB-D Workshop. 2011.

[3] M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.