Basic Python and Particle Filter

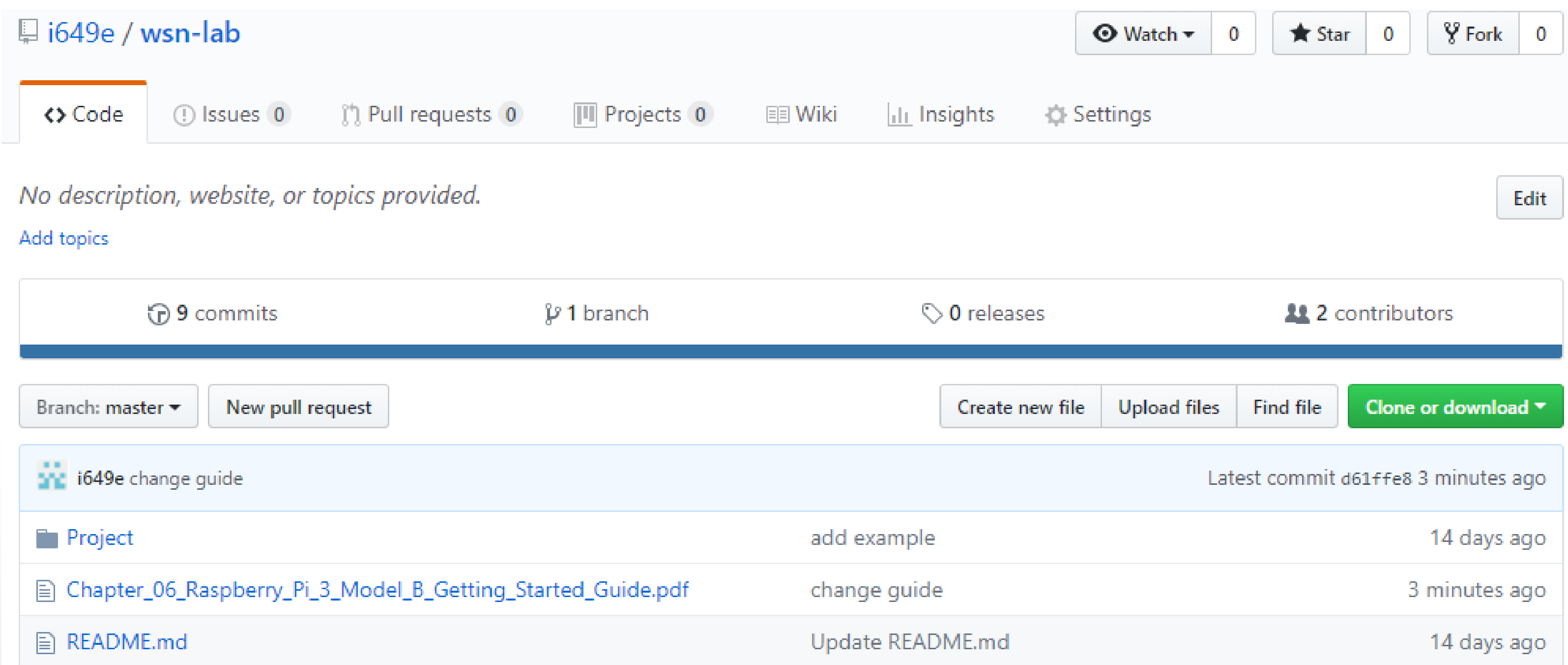
Nantawat Pinkam

I649E Wireless Sensor Networks

July 4, 2018

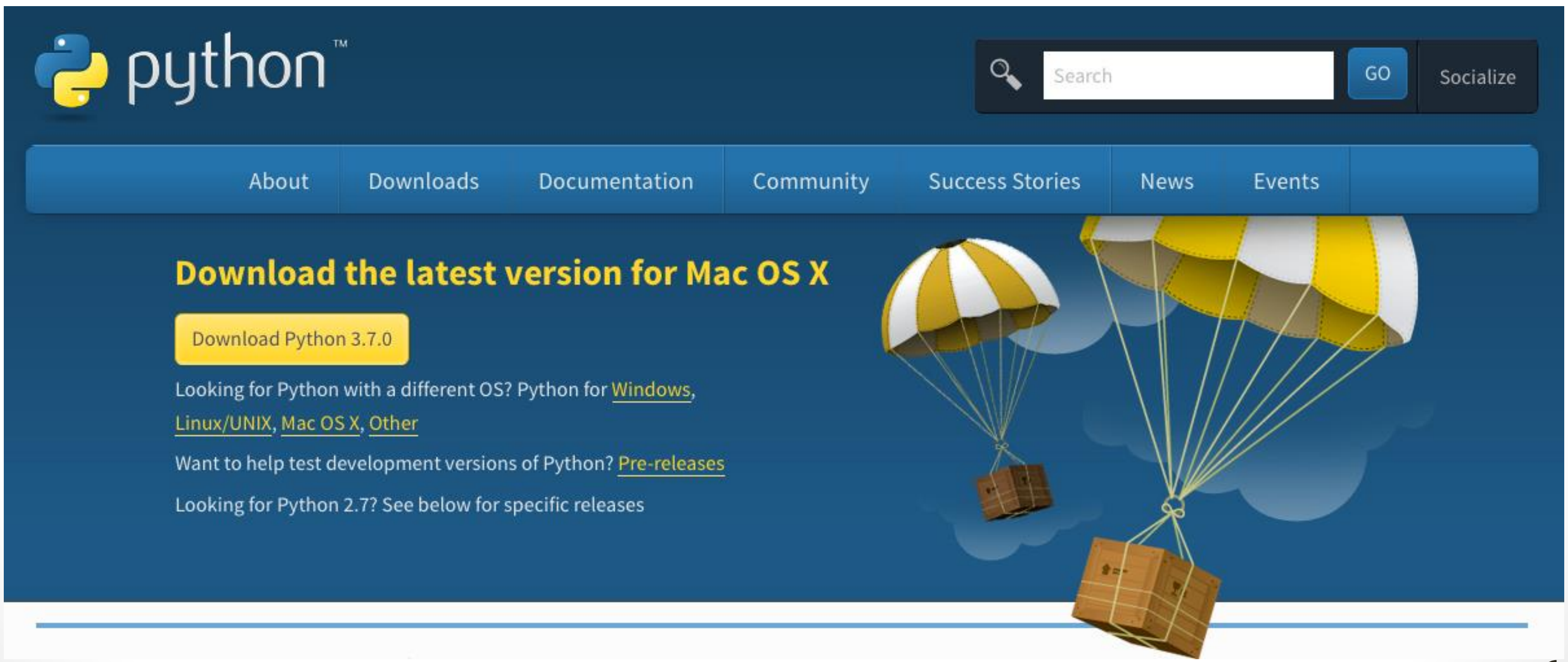
I649E's Github

https://github.com/i649e/wsn-lab



Python Installation

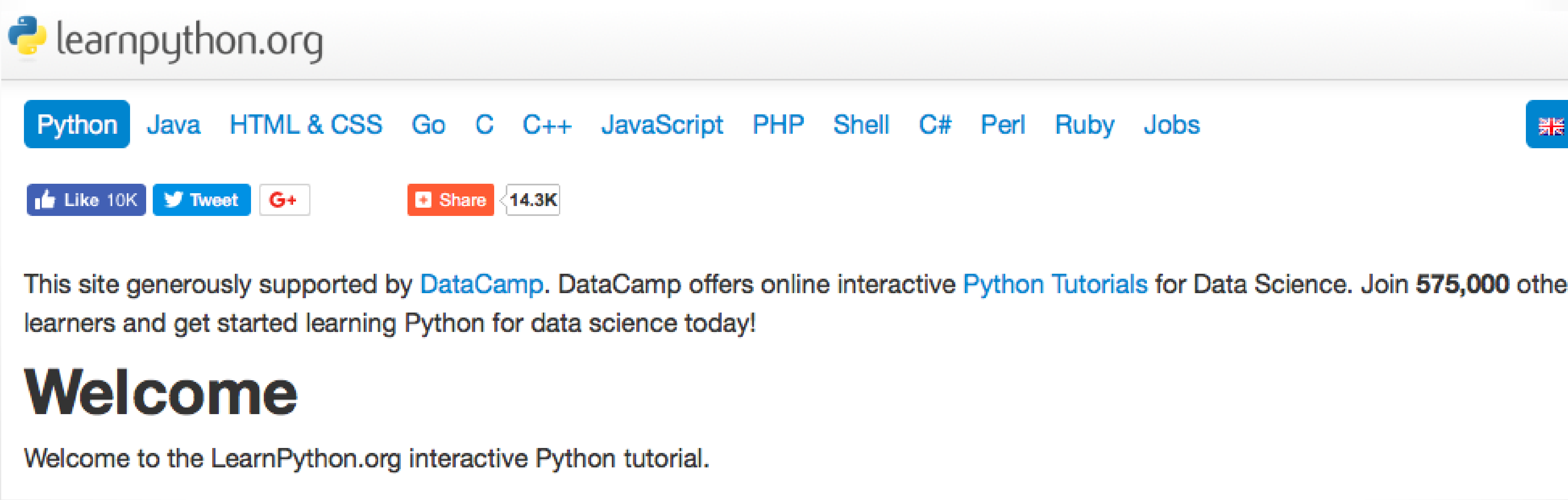
https://www.python.org/downloads/



Python References

https://github.com/i649e/wsn-lab/tree/master/Reference

https://www.learnpython.org



Python Module Installation

In command prompt (Windows) or Terminal (macOS/Linux)

Python 2

```
python -m pip install --upgrade pip
python -m pip install <module name>
```

Python 3

```
python3 -m pip install --upgrade pip
python3 -m pip install <module name>
```

Basic Python

- Mostly same as C, C#, C++
- Operators
  - =, +, -, *, /
  - // (integer divide), % (modulation), ** (a^b)
- Boolean
  - >, <, >=, <=, ==, !=
  - and, or
- Variable Assignment
  - A=B
  - A=B=1
  - A,B,C = 1,2,3 → A=1; B=2; C=3
  - A+=1. → A=A+1
- Print value
  - print(A)
  - print(“A+B = ” + str(2)) (The contents in print must be the same type)

Basic Python

- Container
  - List: A=[1, " apple" ,3] (can be added more variables later)
    - A.append(x), A.extend(x), A.insert(i, x), A.remove(x), A.pop([i]), A.clear(), A.count(x), A.reverse(), A.copy()
  - Tuple: A=(1, "apple" ,3) (cannot be changed)
  - Dictionary: A={key:value, 1: "apple"}
    - A.get(key), A.has_key(key), A.items(), A.keys(), A.update(B), A.value()
  - Set: A={key1, key2, "apple"}
- Useful commands
  - len(A) Number of elements in A
  - A[-1], A[0], A[1], A[-1:1] Support positive and negative indexes

Statement Blocks

: colon

```
parent statement:
indent    statement block1...
          .
          .
          .
          parent statement:
indent              statement block2...
                    .
                    .
                    .
```

Loop and Function

range(x) 0, 1, 2, …, x-1

Python	C

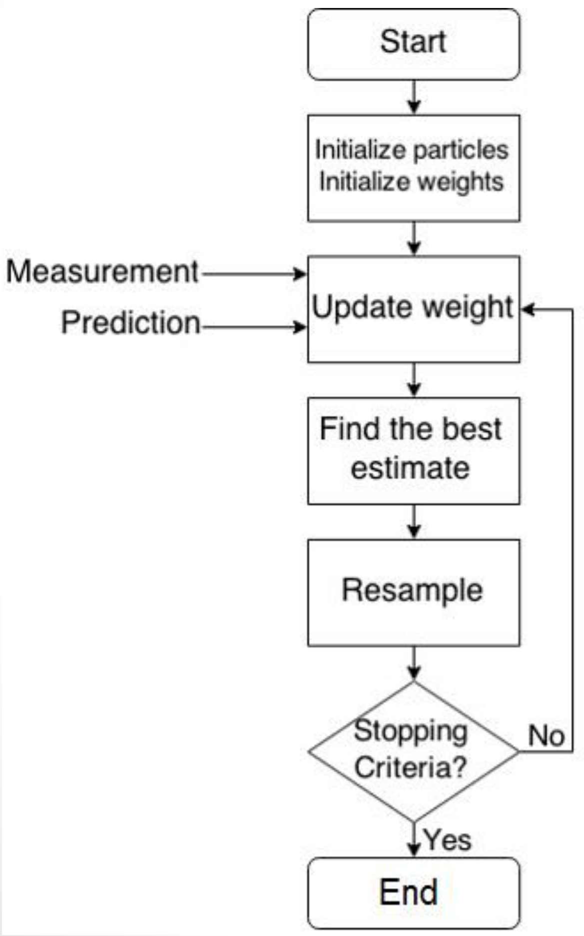
`for i in range(5):`
`        print(i)`
	`for ( i = 0; i < 5; i++ ){`
	`        printf("%d\n", i);`
	`}`

`i=0`
`while( i < 5 ):`
`        print(i)`
`        i+=1`
	`int i=0;`
	`while( i < 5 ){`
	`        printf("%d\n",i);`
	`        i++;`
	`}`

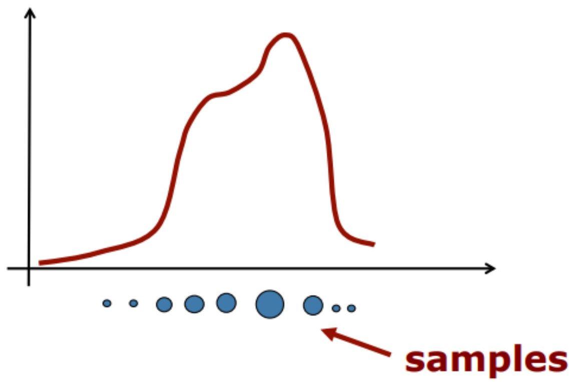
`def A_plus_B (A, B):`
`        return A+B`
`A, B = 1, 2`
`AB = A_plus_B(A, B)`
`print(AB)`
	`int A_plus_B(int A, int B){`
	`        return A+B;`
	`}`
	`int main(){`
	`        int A=1;int B=2;int AB;`
	`        AB = A_plus_B(A,B);`
	`        printf("%d\n",AB);`
	
	`        return 0;`
	`}`

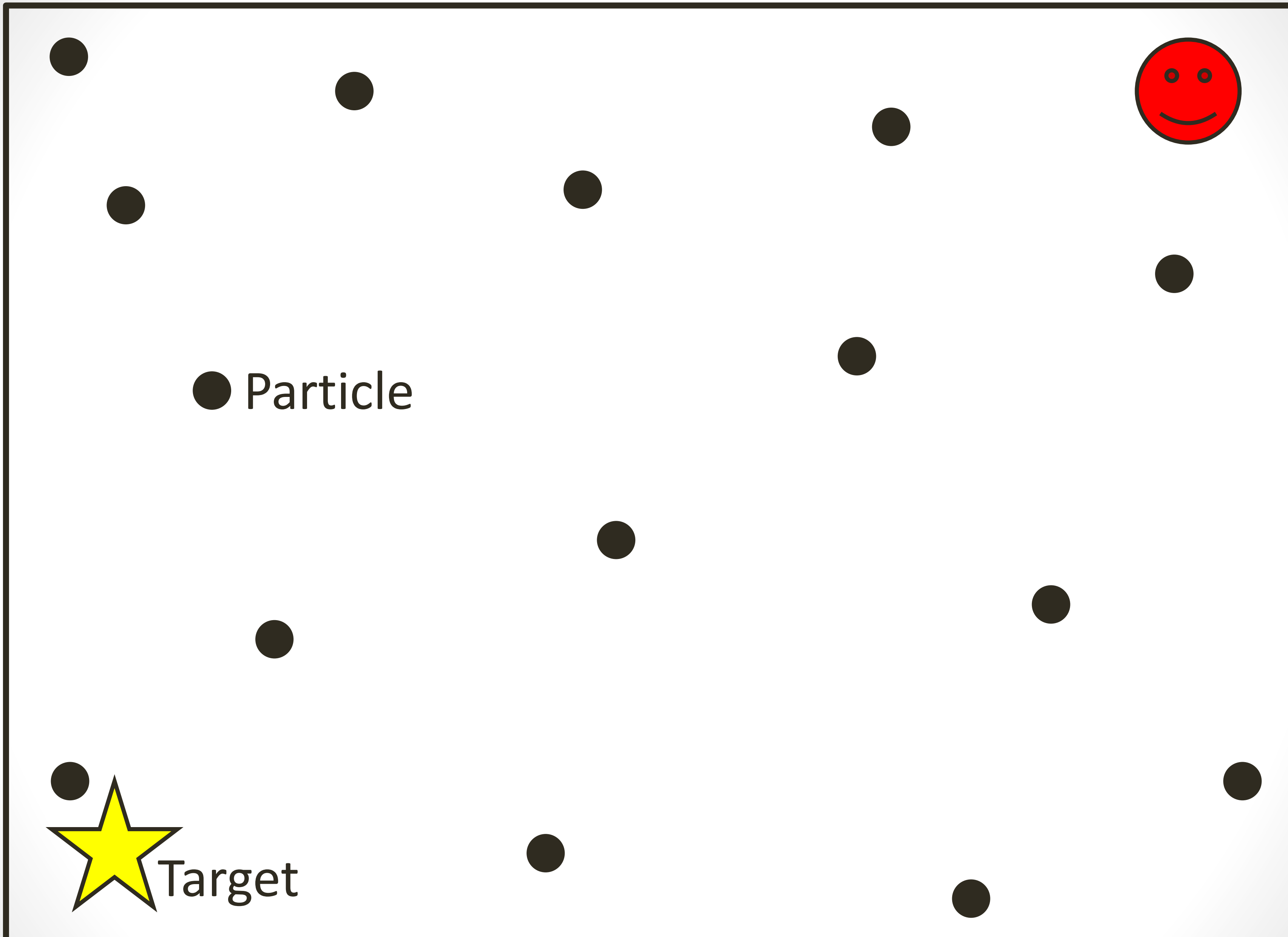
Sample-based Localization

Particle Filter

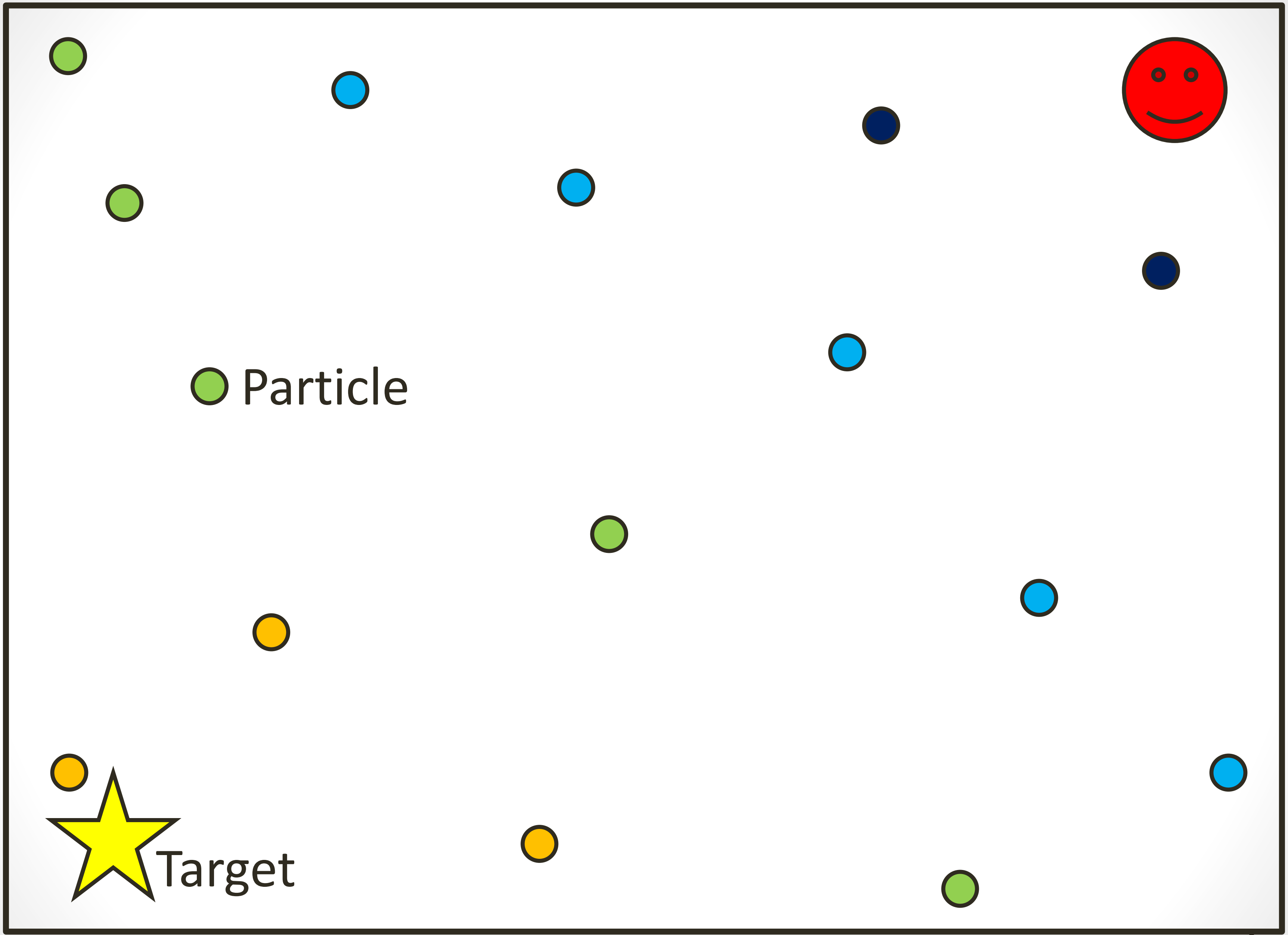


- Bayesian based filter that sample the whole workspace by a weight function derived from the belief distribution of previous stage.
- Represent belief by random samples. (e.g. each particle is the hypothesis of the target state)

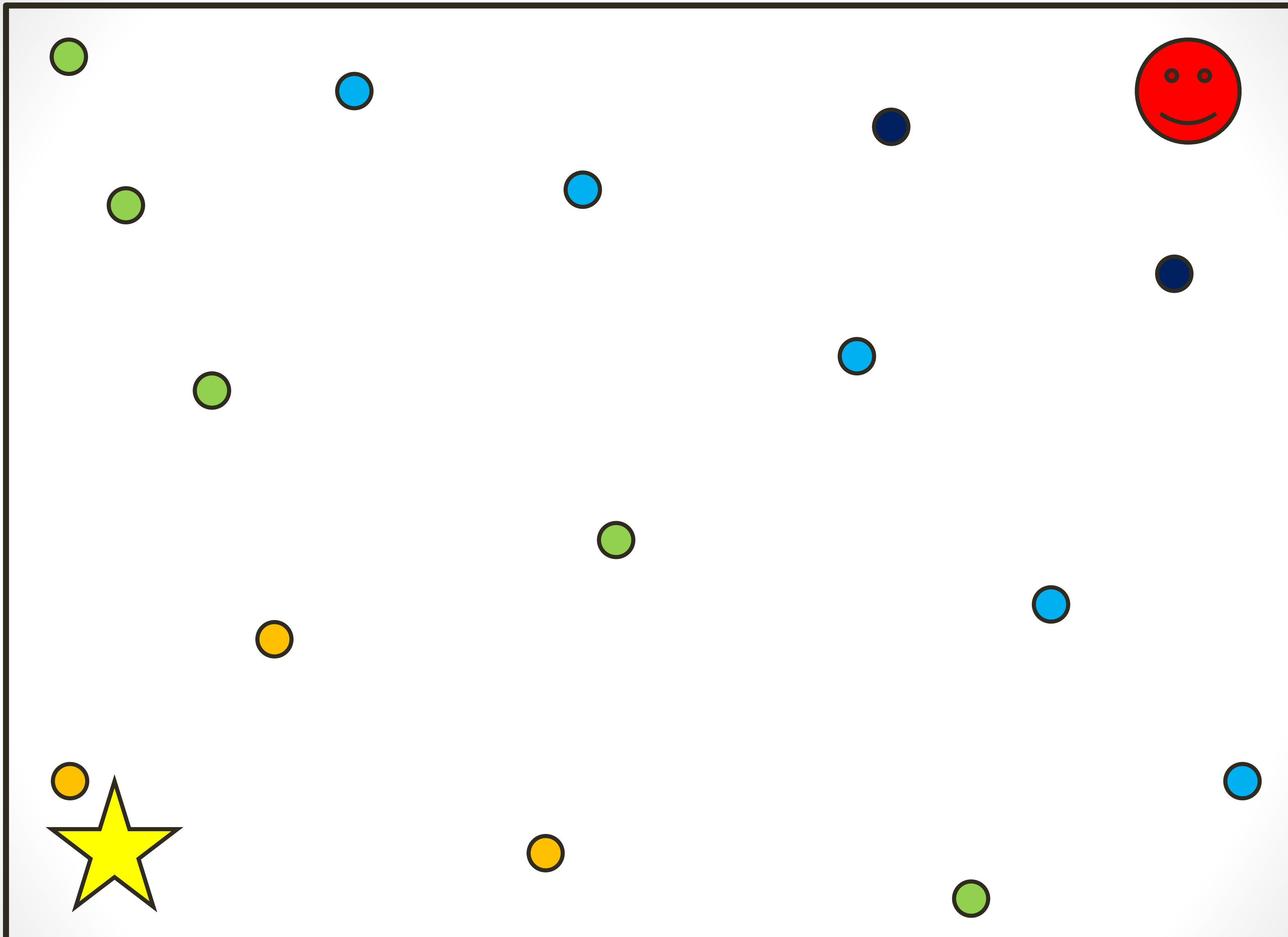




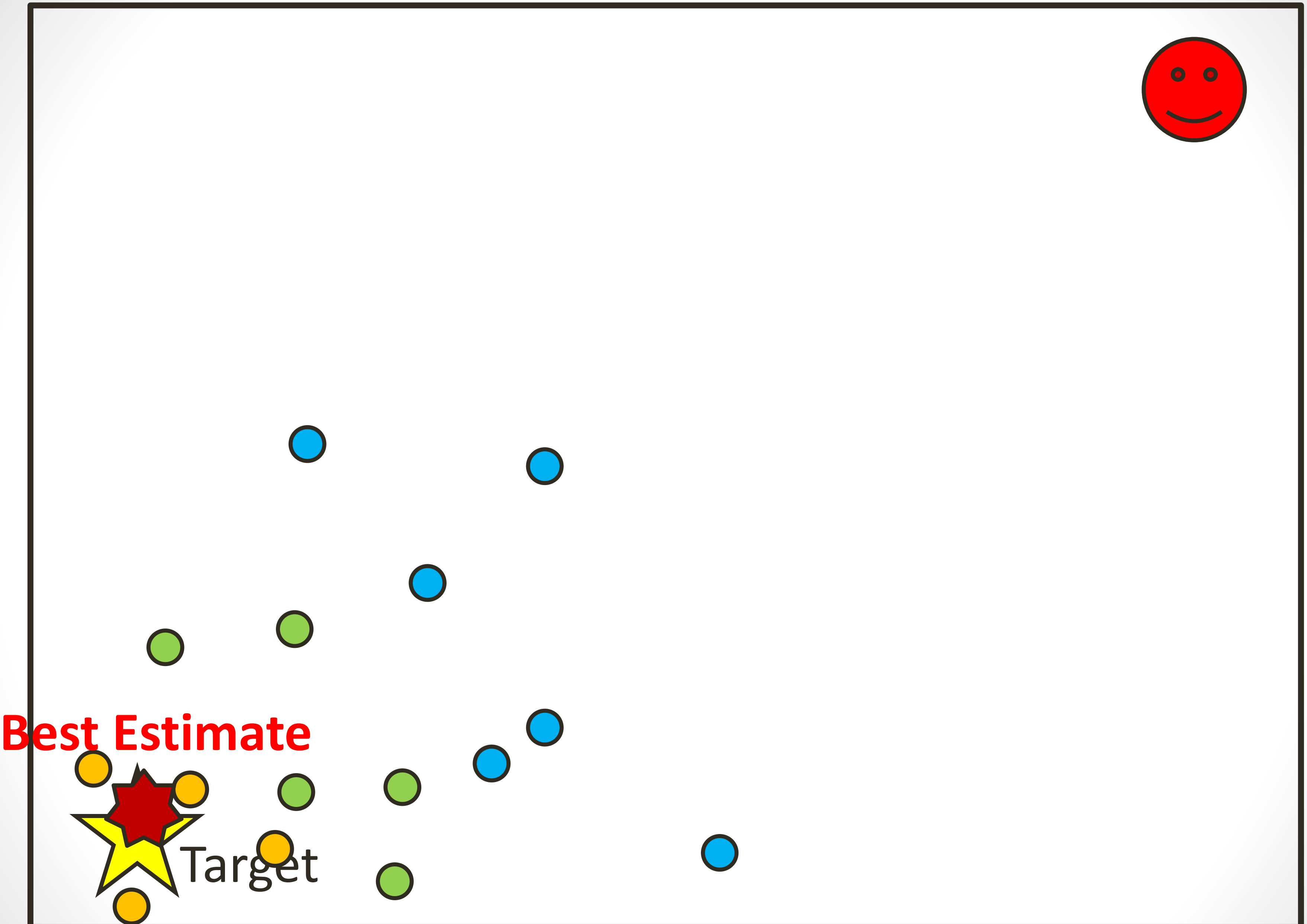
1 Randomly place particle all over the workspace.



2 Calculate the weight of each particle



3 Find the best estimate and resample the low weight particles



Repeat 2-3 (calculate weight > best estimate > resample)

Random Samples as Particles

$$\mathcal{X} = \{\langle x^{[i]}, w^{[i]} \rangle\}_{i=1,\ldots,N}$$

state hypothesis

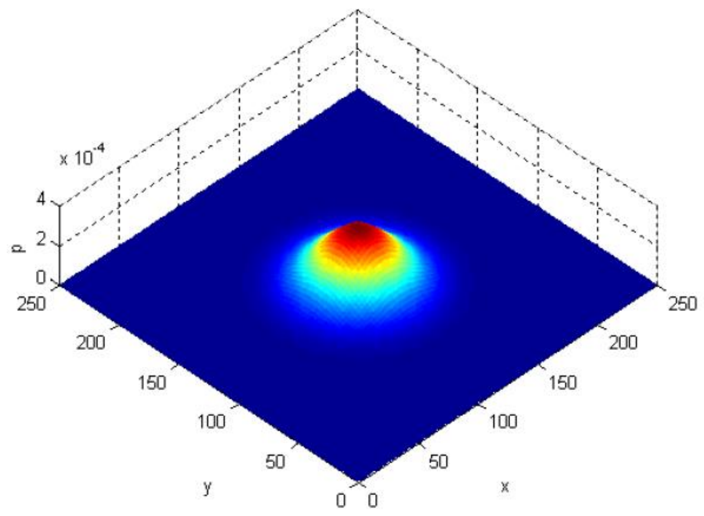
Example:

- Location (x, y, z)
- Measurement

importance weight

How much should I trust this particle?

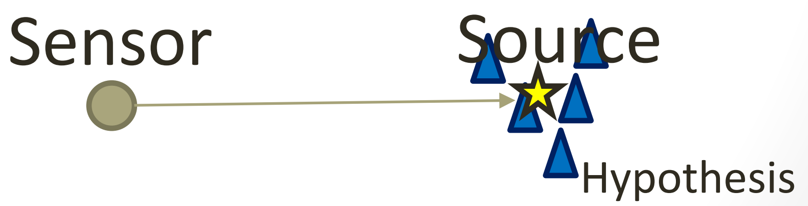
Belief Space



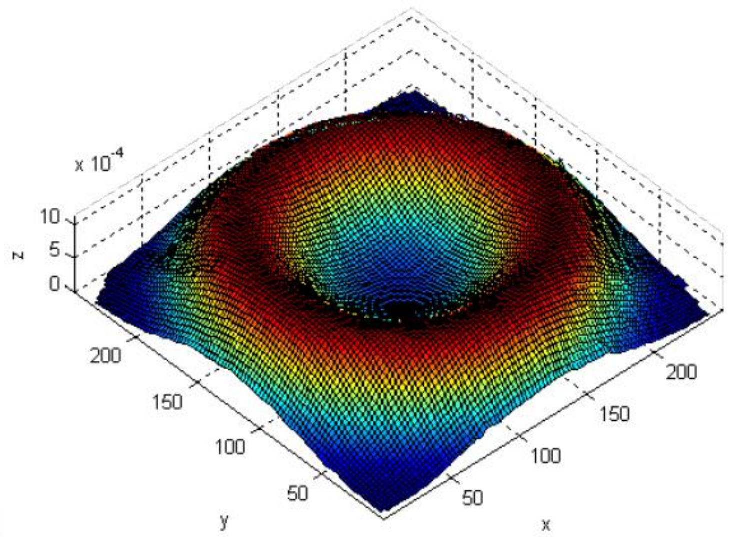
Gaussian belief space

Example

- Measurements from a sensor with **known** direction to the source.



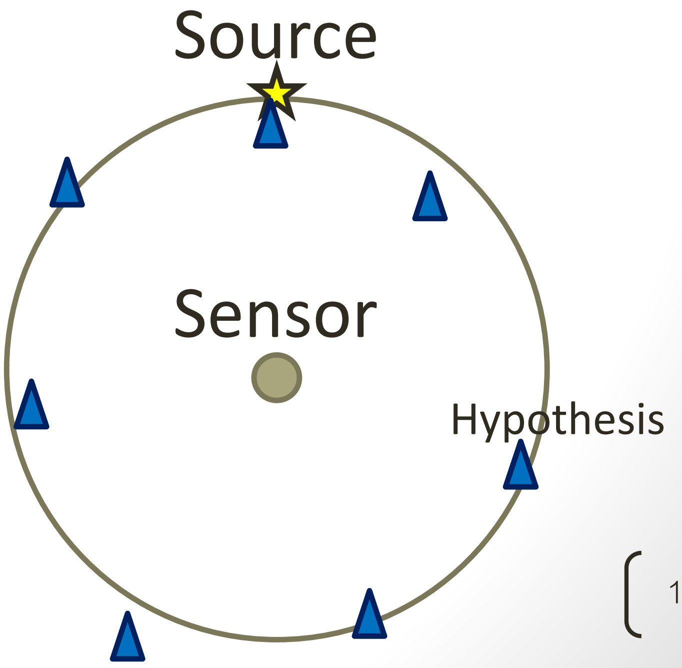
Belief Space



Non-Gaussian belief space

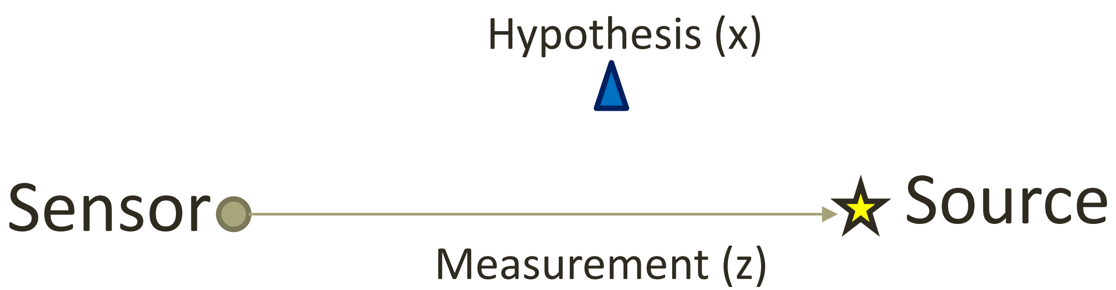
Example

- Measurements from a static sensor with **unknown** direction to the source.



Particle Weight

- By using likelihood function to determine the weight.
- Basically, we compare the hypothesis with the measurement and calculate the weight.



Weight = $N(x|z,\sigma^2)$

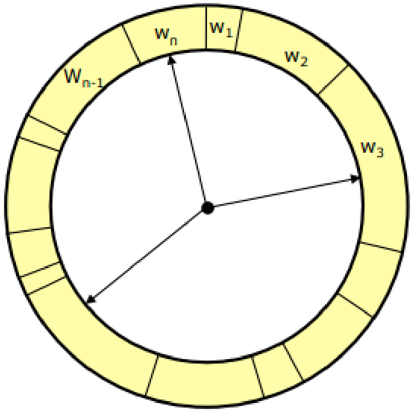
N = normal distribution

Estimation

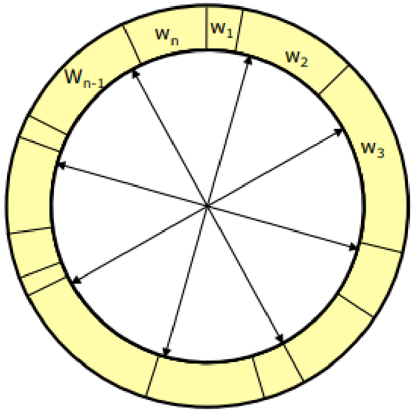
- Estimate the most likely state from all/some particles.
- Method of estimation: weight average, mean, top particles, etc.
- Particle filter can run with finite iterations or until the criteria are met. (e.g. no more particles improvement)

Resample

- Survival of the fittest: Replace unlikely samples by more likely ones
- Avoid many samples cover unlikely states
- Needed as we have a limited number of samples



- Roulette wheel



- Stochastic universal sampling

Demo: 1D Particle Filter

Requirements: need to install python modules and jupyter notebook

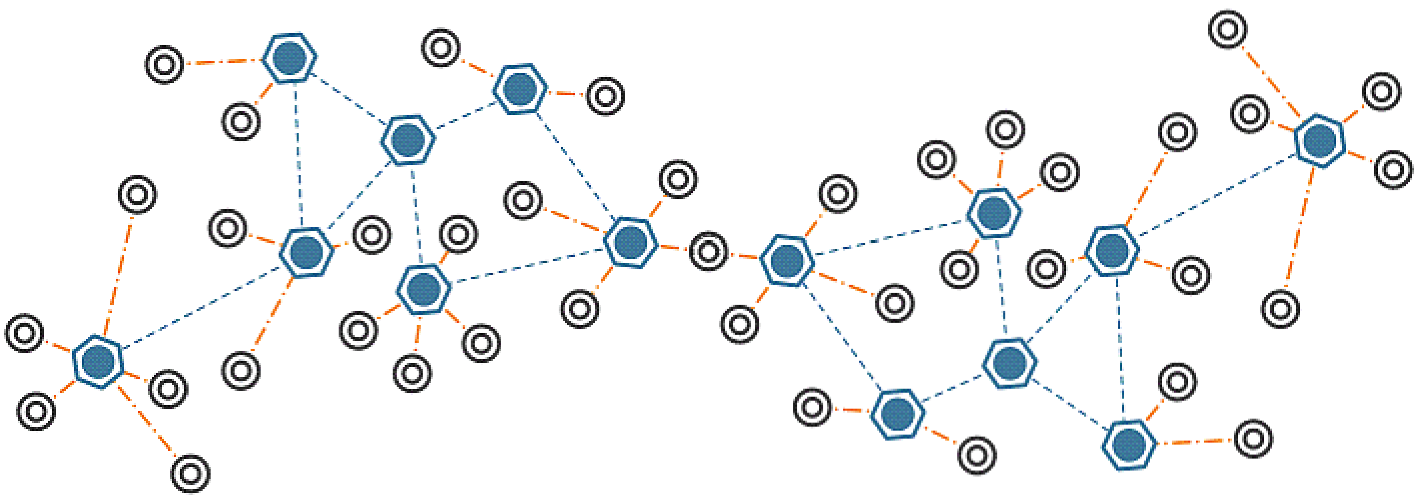
```
python3 -m pip install numpy scipy matplotlib jupyter
```

Download `Tutorial 2 1D Particle Filter.ipynb` from Github
In Command prompt/Terminal, `cd` to your downloaded file directory then type:

```
jupyter notebook
```

Open `Tutorial 2 1D Particle Filter.ipynb` from jupyter notebook
You also can run `1D_pf.py` from the class Github.

Question and Answer



THANK YOU.