EKF
HW guide

Date：2021/05/04

Ultimate goal & Environment

- **Locate your cart correctly** **while move toward target points**
  - Use EKF localization
- **Landmarks correspondences & position are known**
  - Unknown correspondence for bonus question
- **Actuator & sensor were both perturbed**
  - Stochastically with known variances

- **Utilize MATLAB as environment**
  - Hand in script(s) that can be validate by actual running

Provided interface to physical world

- **X_t = VehicleModel(v, w, X_t-1)**
  - v : Forward speed
  - w : Rotation speed
  - X_t-1 : Previous state [X, Y, Theta]
  - X_t : Return state after actuating

Command given by navigation control

This is what you should estimate: Localizing + Navigation algorithm DO NOT access these variables except the initial value

- **SenseData = SensorModel(X_t, LM)**
  - X_t : Actual state [X, Y, Theta]
  - LM : Landmark coordinates
  - Data : Sensor readings [IsSensed, Relative distance, Relative angle]
    - If landmark is in cart's left hand side, Relative angle > 0

Example of “SenseData^T”

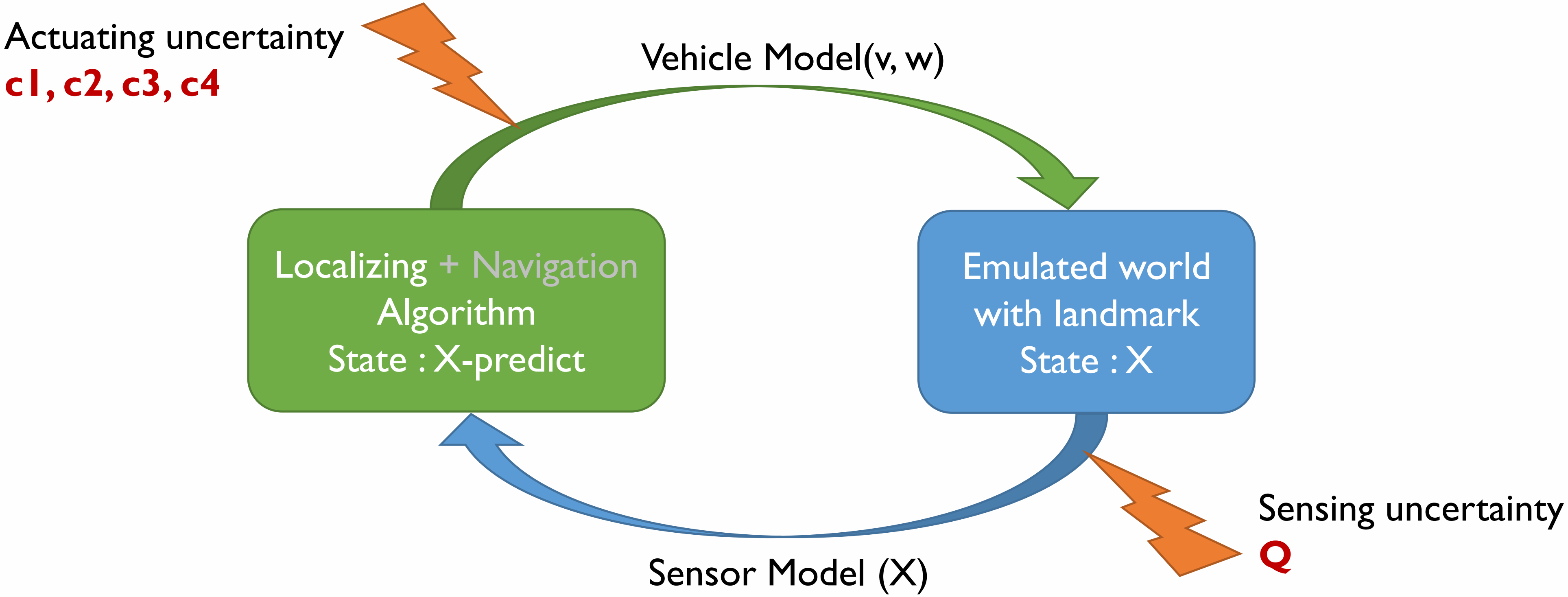
Size: $(3*N)^T$
N: count of LMs

Is correspond LM sensed?	1	0	0	…	1
Sensed relative distance	13.3	x	x	…	5.3
Sensed relative direction	0.75	x	x	…	-0.12

Algorithm you should build up

- **Xe_t = Estimate(v, w, Xe_t-1, SenseData)**
  - [v, w] : Command data
  - Xe_t-1 : Previous estimated state
  - Data : Sensor readings
  - Xe_t : Return estimated state after actuating + sensing
- **[v, w] = Navigation(goal, Xe_t)**
  - Goal : Target state
  - Xe_t : Estimated state
  - [v, w] : Generated command

Each state/variable is related as shown



Do NOT modify any red colored parameters

Self check list & tips

- **Can algorithm handle multiple landmarks simultaneously?**
- **How to relate sensor variance to state space?**
- **Does it works stably at all condition?**

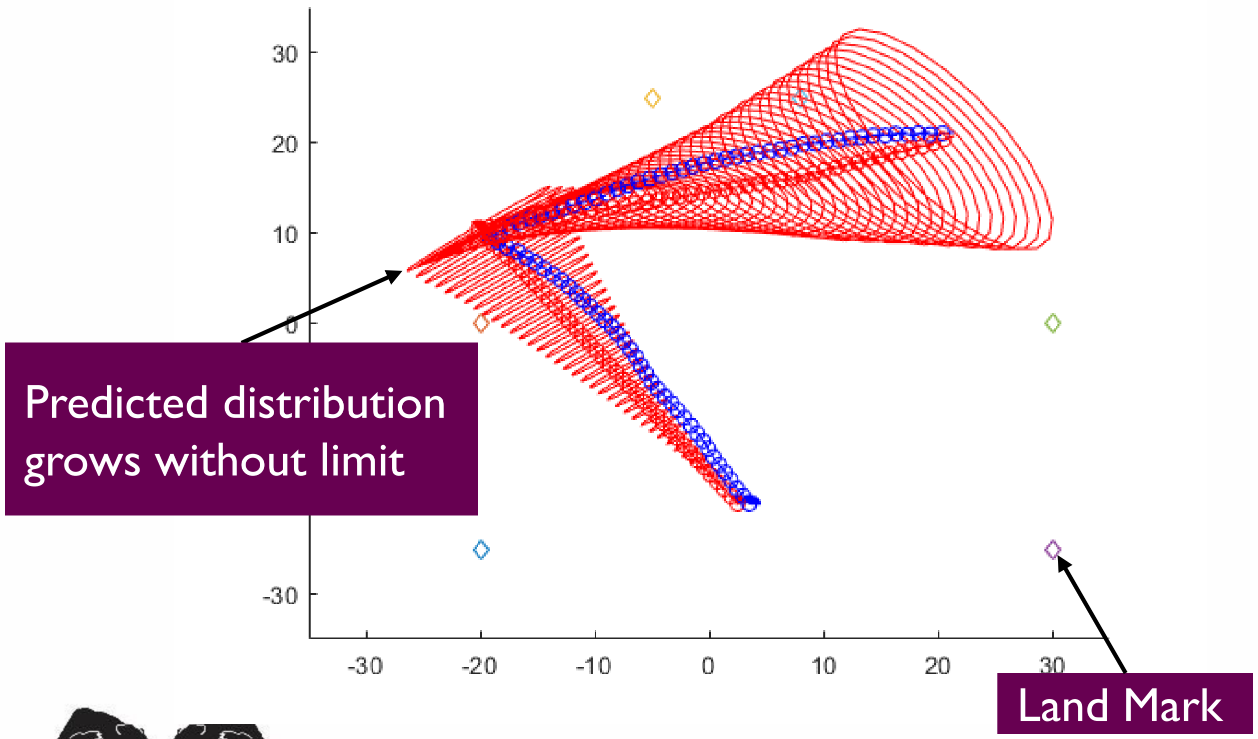
- **Visualize your result**
  - Direction of car's heading
  - Variance of states
  - Use different color for before & after filtering

- **About 1~4 hours work**
  - Seek for help if longer than expected

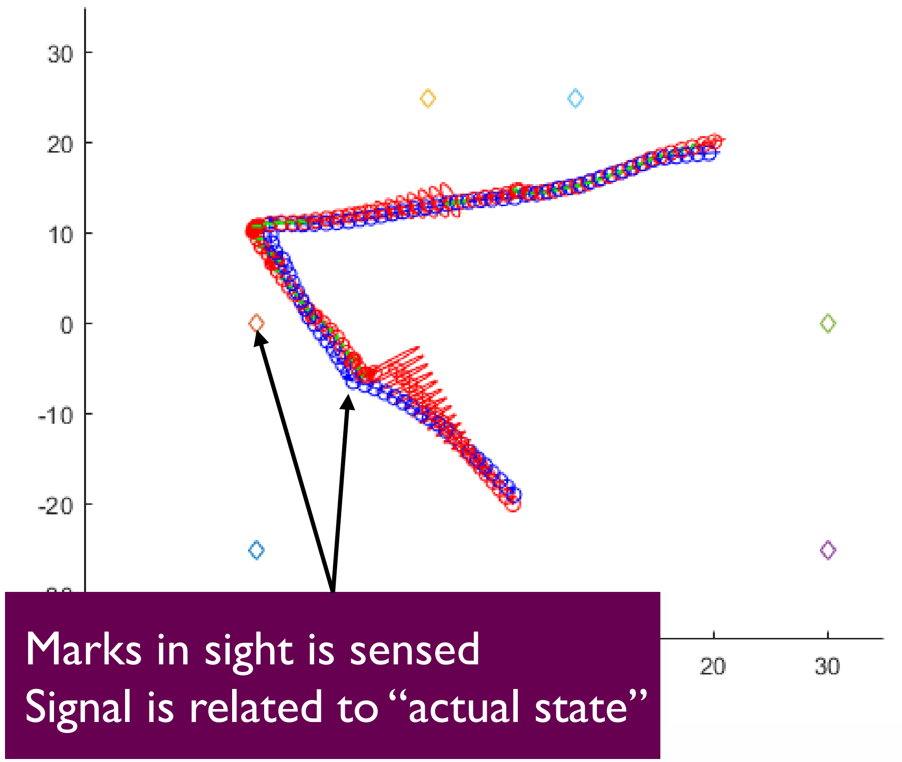
For your reference

Red: predicted distribution
Green: updated distribution
Blue: actual position

Without sensor



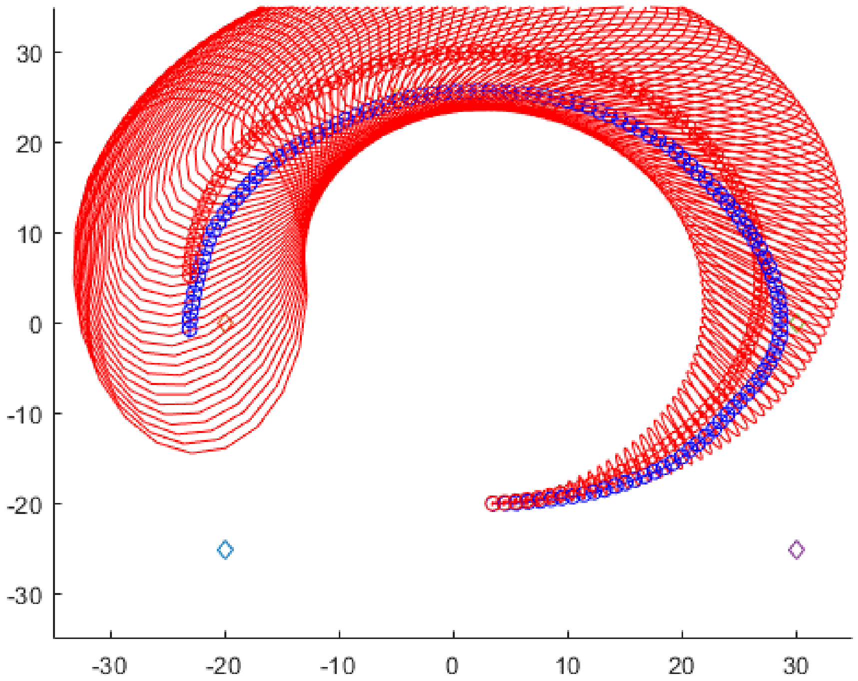
With sensor



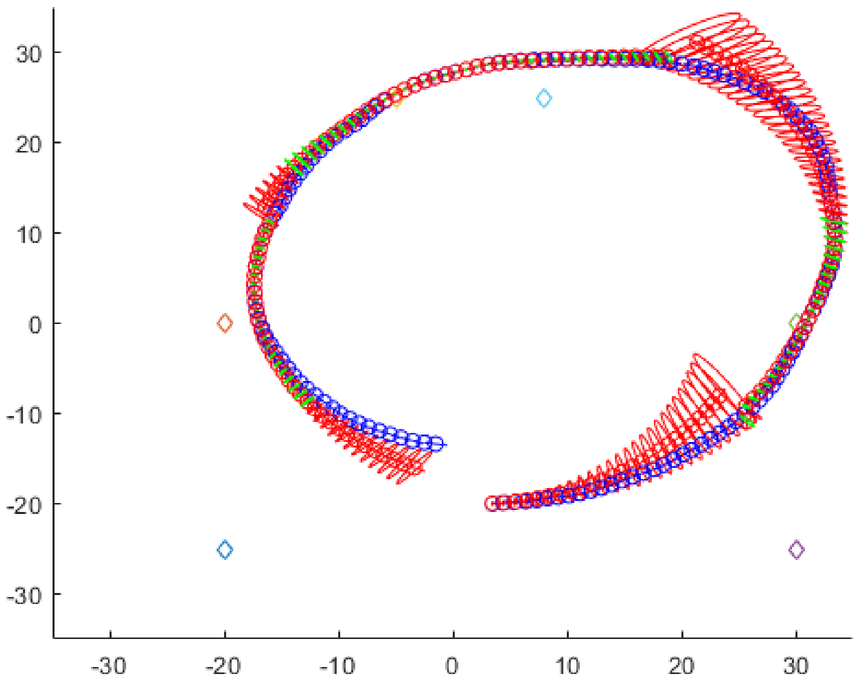
continued

Red: predicted distribution
Green: updated distribution
Blue: actual position

Without sensor



With sensor



Bonus

- **Try same problem with unknown correspondence**
- **Use function SensorModelUC(X_t, LM) instead of original**
- **Example of "SenseDataUC^T"**
  - It is possible to return null data if nothing was sensed

Size: $(2*N)^T$
N: count of sensed LMs

Sensed relative distance	13.3	6.8	3.9	…	5.3
Sensed relative direction	0.75	-0.13	0.44	…	-0.12