Extended Kalman Filer Localization

<u>คำสั่ง</u> จงดูตัวอย่าง Code Matlab ในไฟล์ main_ekf_find_mistake.m ที่ให้สำหรับงาน EKF localization ในโปรแกรมดังกล่าวมีการสร้าง landmark หลายจุด โดยการสุ่ม โดยเราสมมุติว่าเราทราบตำแหน่งของจุดเหล่านี้ทั้งหมด ตำแหน่งจริงคือตัวแปร X (สีน้ำเงิน) และตำแหน่งที่ทำการ Predict คือตัวแปร mu (สีแดง) ในโปรแกรมหุ่นยนต์เริ่มต้นที่ตำแหน่งที่สุ่มขึ้นจุดหนึ่ง (x, y, th) และค่าเริ่มต้นของการทำนาย mu ก็สุ่มขึ้นมาเช่นเดียวกัน เมื่อหุ่นยนต์เคลื่อนที่ไปเรื่อยๆ ก็จะทำการทำนายตำแหน่งใหม่ตาม landmark ที่ตรวจพบ โดยมีวัตถุประสงค์ เพื่อให้ตำแหน่งที่ทำนายมีค่าใกล้เคียงกับตำแหน่งจริงมากที่สุด จงดำเนินการดังนี้

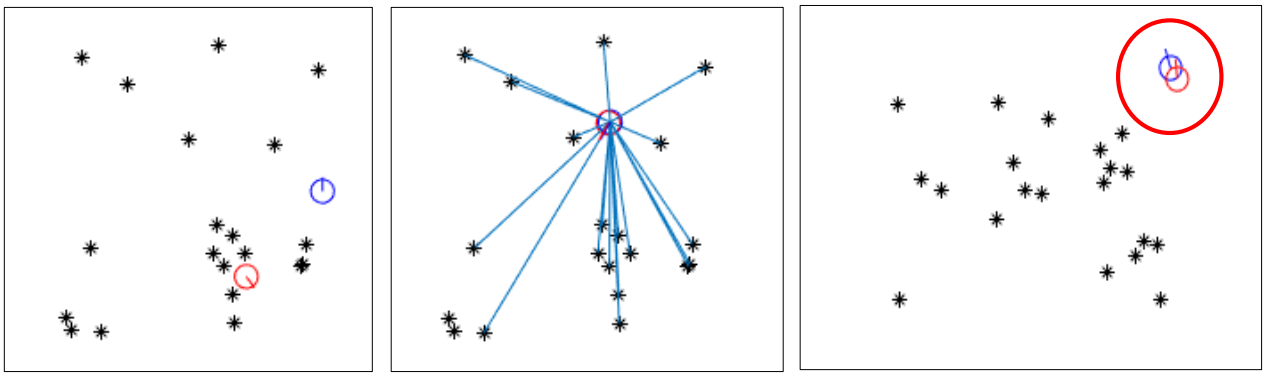
<u>คำตอบ</u>

1. ทดลอง รันโปรแกรม หลายๆ รอบ รายงานผลที่สังเกตุเห็น

จากการทดลองรันโปรแกรมหลายๆ รอบ พบว่าเมื่อเริ่มต้นโปรแกรมหุ่นยนต์จริงสีน้ำเงิน และหุ่นยนต์ที่ได้จากการประมาณค่าความน่าจะเป็นสีแดงอยู่ห่างกันตามที่กำหนดไว้ในโปรแกรม (ดังแสดงในรูปที่ 1 ก.1) เมื่อระยะเวลาผ่านไปตัวสีแดงจะขยับเข้าใกล้ตัวสีน้ำเงิน (ดังแสดงในรูปที่ 1 ก.2) และในกรณีที่หุ่นยนต์ทั้งสองเคลื่อนที่ห่างออกจากจุด Landmark พบว่าหุ่นยนต์ทั้ง 2 เคลื่อนที่ห่างออกจากกันเอง (ดังแสดงในรูปที่ 1 ก.3) จนกระทั่งเคลื่อนที่เข้าใกล้ Landmark หุ่นยนต์ก็เข้าใกล้กันอีกครั้ง ซึ่งโดยหลักการแล้ว การทำงานของโปรแกรมถูกต้อง แต่ในบางครั้งสังเกตุเห็นได้ว่า บางจุดขณะเกิดการเลี้ยวตำแหน่งของหุ่นยนต์ตัวที่ทำนายค่าหลุดออกมาจากหุนยนต์ตัวจริง (ดังแสดงในรูปที่ 2 ก.1) ทั้งที่มีการลู่เข้าตำแหน่งจริงแล้ว (ดังแสดงในรูปที่ 2 ก.2) ตามหลักการไม่น่าเป็นไปได้

<u>จุดสังเกตุที่สำคัญ</u>

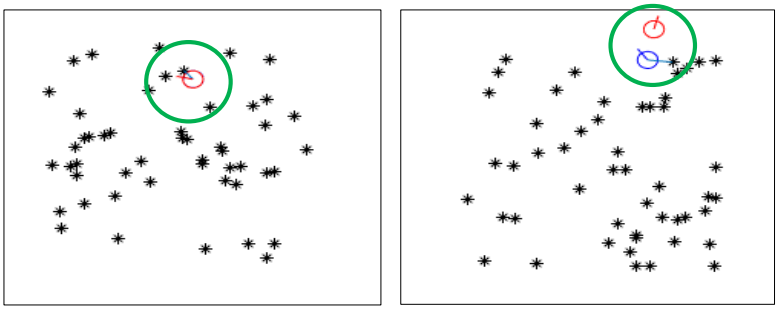
- โดยมากแล้วหุ่นยนต์กระโดดออกขณะเลี้ยว ดังนั้นต้องพิจารณาส่วนที่เกี่ยวกับการนำค่ามุมมาใช้ปันพิเศษ



ก.1 ก.2 ก.3

รูปที่ 1 แสดงการรันโปรแกรมและสังเกตุผลลัพย์ที่เกิดขึ้น

Extended Kalman Filer Localization

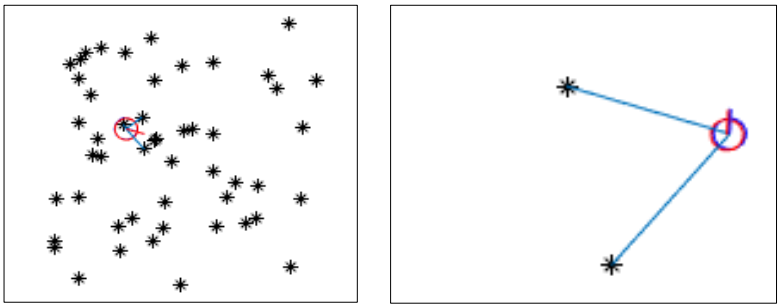


ก.1 ก.2

รูปที่ 2 แสดงการลู่เข้าค่าจริงแล้วหลังจากนั้นลู่ออก

2. ทดลองปรับค่า Sensor range ต่างๆ จาก 1, 2, 5, 10 แต่ละครั้งรันหลายๆ รอบ รายงานผลที่สังเกตเห็น

จากการทดลองพบว่า ที่ Sensor range น้อยๆ จะสามารถทำงานได้ดีในกรณีที่ landmark อยู่ติดกันและอยู่ใกล้หุ่นยนต์เท่านั้น (ดังแสดงในรูปที่ 3 ก.1) กรณีนอกเหนือจากนั้นมีการทำนายผิดพลาดมากขึ้น และเมื่อเพิ่มค่า Sensor range สามารถทำงานได้ดีถึงแม้ว่า landmark จะมีน้อยก็ตาม (ดังแสดงในรูปที่ 3 ก.2) แต่ในสถาณการจริงนั้นถ้า Sensor range มีค่ามากราคาของเซนเซอร์จะค่อนข้างสูง ดังนั้นการเพิ่มจำนวน landmark เป็นเรื่องที่น่าสนใจ



ก.1 ก.2

รูปที่ 3 แสดงการปรับ Sensor range และ Landmark

3. ทดลองปรับค่า Parameter อื่นๆ รายงานผลที่สังเกตเห็น

จากโปรแกรมข้างต้นนั้น Parameter ที่สำคัญแลมีผลกระทบอย่างมากกับการทำนายค่าคือ Sensor range และ landmark จากการทดลองปรับจำนวนของ landmark พบว่ามีความสอดคล้องกับ Sensor range เนื่องจากถ้า landmark เยอะสามารถลด Sensor range ได้ หรือถ้าต้องการลด landmark ก็สามารถทำได้โดยการเพิ่ม Sensor range ได้ในทำนองเดียวกัน

4. ในโปรแกรมนั้นยังมีข้อผิดพลาดอยู่บางจุด ถ้าสังเกตจากการปรับค่าพารามิเตอร์ตามรายการด้านบนและทำการ ทดลองซ้ำๆ จะเห็นว่าบางครั้งตำแหน่งที่ทำนายได้ของหุ่นยนต์มีการกระโดดไปมาหลังจากที่ค่าทำนายลู่เข้าไปแล้ว ซึ่งไม่ควรจะเกิดขึ้น จงศึกษาโปรแกรมให้ละเอียดและปรับแก้โปรแกรมเพื่อแก้ปัญหาดังกล่าว หากสะดวกกว่า นักศึกษาสามารถเขียนโปรแกรมของตัวเองขึ้นมาใหม่ได้ทั้งหมด

จากการศึกษาพบว่า ปัญหาที่เกิดขึ้นในขั้นตอนการหา error_z เพื่อนำไปอัพเดทในการทำนาย (ดังแสดงในรูปที่ 4 ก.1) ซึ่งต้องทราบหาค่า z_bar ที่หาได้จาก function (ดังแสดงในรูปที่ 4 ก.2) แต่เนื่องจากใน atan2 ใน Matlab สามารถใช้งานได้ในช่วง -pi ถึง pi เท่านั้น เมื่อค่าที่ส่งเข้ามาไม่อยู่ในช่วงที่กำหนดทำให้การคำนวณค่าผิดพลาด จึงเกิดปัญหาดังที่กล่าวมาข้างต้น

```
% find error
err_z = (z(:,i)-z_bar);
```

ก.1

```
function z_bar = find_z_bar(mu,landmark,i)
x = mu(1); y= mu(2); th= mu(3);

dis2 = (x-landmark(1,i))^2+(y-landmark(2,i))^2;
z_bar(1,1) = sqrt(dis2);
z_bar(2,1) = atan2(y-landmark(2,i), x-landmark(1,i)) - th;
end
```

ก.2

รูปที่ 4 function หา error_z

ดังนั้นการแก้ปัญหาควรเขียน function เพื่อมาป้างกันในกรณีที่ err_z เกิน pi และน้อยกว่า -pi (ดังแสดงในรูปที่5 ก.2) และแทนการคำนวณหาค่า err_z (ดังแสดงในรูปที่ 4 ก.1) ด้วย function ที่สร้างขึ้นมาแก้ปัญหานี้ (ดังแสดงในรูปที่ 4 ก.2) โปรแกรมที่ได้รับการแก้ปัญหาแสดงดัง ไฟล์แนบ (main_ekf_find_mistake.m)

```
function err = correct_angle_err(z,zbar)
dum = z-zbar;
if (dum > pi)
    err = dum - 2*pi;
elseif (dum < -pi)
    err = dum + 2*pi;
else
    err = dum;
end
end
```

ก.1

```
err_z = correct_angle_err(z(:,i), z_bar);
```

ก.2

รูปที่ 5 function แก้ err_z ที่เกิน pi และน้อยกว่า –pi