Week 2

SUBFORUMS

All

Assignment: Estimating a Vehicle Trajectory

Week 2

GD

Does anyone get the estimated trajectory that is close to the true trajectory?

Guangfei Duan Assignment: Estimating a Vehicle Trajectory · 21 days ago

I have been tuning the measurement noise variances and couldn't get the estimation to be close to the true trajectory. Has anyone got a close one? I have tried v_var = 0.01 ~10 and om_var = 0.01~10.

Also, what are the Jacobian matrices in the prediction step? The notebook gives both F_km and L_km as zero matrices. I think this shouldn't be the case.

1 Upvote Reply Follow this discussion

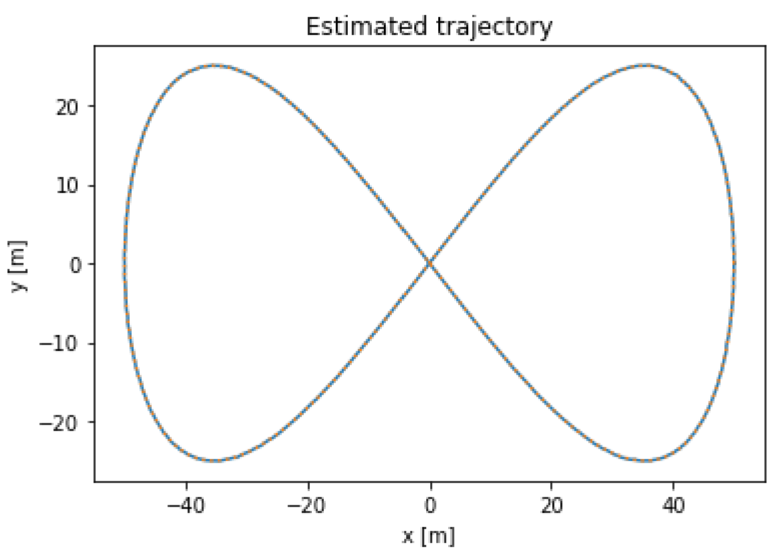
Earliest **Top** **Most Recent**

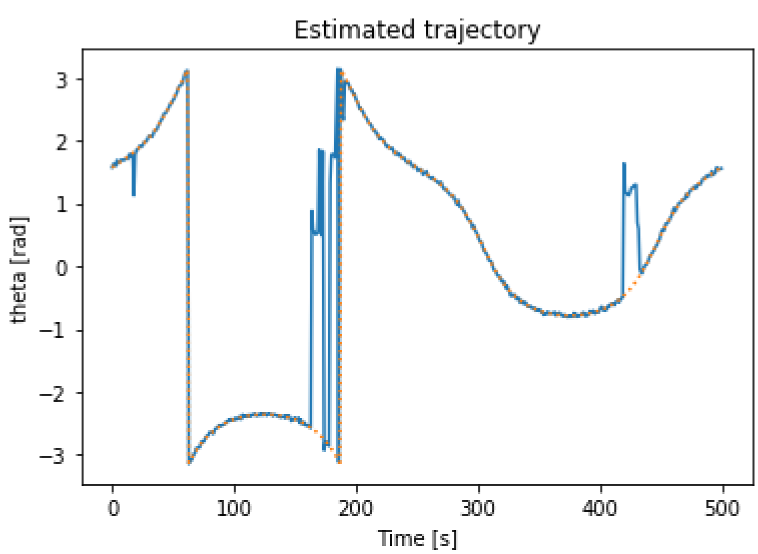
LK

Lea Kantor · 21 days ago

1. Yes, I get a trajectory very close to the ground truth (see below, ground truth in orange dots over estimated in solid blue).

2. The input and measurement noise covariance matrices are given in the variables Q_km and cov_y respectively.
3. F_km and L_km (and all other matrices, like Hk, are only initialized to zero matrices with the correct dimensions, to allocate memory, but you are supposed to insert the expressions for the derivatives in each of their elements separately. For example: **F_km[0, 0] = 1**





3 Upvotes   Hide 8 Replies

GD Guangfei Duan · 20 days ago

Thanks for the advice.

May I ask what are the F_km and L_km matrices that you used? I thought F_km should be zero matrix and L_km should be the matrix coefficient multiplied by (input+w), as shown in the following picture:

$$\begin{bmatrix} \cos\theta_{k-1} & 0 \\ \sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix}$$

But I still couldn't get correct trajectory. Any help would be appreciated.

0 Upvotes

LK

Lea Kantor · 20 days ago

Your L_km should be multiplied by delta_t (T), but since it is equal 1 here, omitting it makes no difference in that specific case.

F_km is **not** a zero matrix. It is a Jacobian matrix. You should take the expression for x(k) in the motion model and create a matrix of derivatives by the elements of x(k-1).

for example : x0(k) = x0(k-1) + delta_t * cos(x2(k-1))*v(k)

so, f_km[0,0] is dx0(k)/dx0(k-1) which is 1

another example: x1(k) = x1(k-1) + delta_t *sin(x2(k-1))*v(k)

so f_km[1,2] is dx1(k)/dx2(k-1) which is equal delta_t*cos(x2(k-1))*v(k)

where: x0 is x, x1 is y, x2 is theta.

2 Upvotes

GD

Guangfei Duan · 20 days ago

Thank you very much for the clarifications. I forgot that theta is part of the state variables. However, after I changed my F_km matrix, I still cannot get a right trajectory. I attached my code for the main prediction-update scheme here. Can you help me to check where I made mistakes?

```python
def measurement_update(lk, rk, bk, P_check, x_check):

    # 1. Compute measurement Jacobian
    x=x_check[0,0]
    y=x_check[0,1]
    th=x_check[0,2]
    d=0

    star1=lk[0]-x-d*np.cos(th)
    star2=lk[1]-y-d*np.sin(th)
    den=np.sqrt(star1**2+star2**2)
    frac=star1**2+star2**2

    Hx=np.mat([[-star1/den,     -star2/den,     (star1*d*np.sin(th)-star2*d*np.cos(th))/den],
               [star2/frac,     -star1/frac,    -1-d*(np.sin(th)*star2+np.cos(th)*star1)/frac]])
    Mx=np.identity(2)

    # 2. Compute Kalman Gain
    Kk=P_check@Hx.T@inv(Hx@P_check@Hx.T+Mx@cov_y@Mx.T)

    # 3. Correct predicted state (remember to wrap the angles to [-pi,pi])
    yk=np.mat([den, np.arctan2(star2,star1)-th])
    ym=[rk,bk]
    x_check=x_check+(Kk@(ym-yk).T).T
    x_check[0,2]=wraptopi(x_check[0,2])

    # 4. Correct covariance
    P_check=(np.identity(3)-Kk@Hx)@P_check

    return x_check, P_check
```

```
#### 5. Main Filter Loop #######################################################################
x_check = x_est[0,:]
x_check=x_check.reshape((1,3))
P_check=P_est[0];
for k in range(1, len(t)):
    delta_t = t[k] - t[k - 1]  # time step (difference between timestamps)

    # Propagate uncertainty
    theta=x_check[0,2]
    Fu=np.mat([[np.cos(theta), 0],
               [np.sin(theta), 0],
               [0,1]])
    inputC=np.mat([v[k-1],om[k-1]])
    x_check=x_check+delta_t*(Fu@inputC.T).T
    x_check[0,2]=wraptopi(x_check[0,2])

    Fx=np.mat([[1,0, -delta_t*np.sin(theta)*v[k-1]],[0,1, delta_t*np.cos(theta)*v[k-1]], [0,0,1]])
    Lx=delta_t*Fu
    P_check=Fx@P_check@Fx.T+Lx@Q_km@Lx.T

    # Update state estimate using available landmark measurements
    for i in range(len(r[k])):
        x_check, P_check = measurement_update(l[i], r[k, i], b[k, i], P_check, x_check)

    # Set final state predictions for timestep
    x_est[k, 0] = x_check[0,0]
    x_est[k, 1] = x_check[0,1]
    x_est[k, 2] = x_check[0,2]
    P_est[k, :, :] = P_chec
```

3 Upvotes

Raphaell Maciel de Sousa · 15 days ago

I found the problem, you should use wraptopi for all angles. Including here:

yk=np.mat([den, wraptopi(np.arctan2(star2,star1)-th)])

1 Upvote

NB

N Srujan Babu · 12 days ago

I had similar problems with theta.. wraptopi didn't help. Did it help you?

0 Upvotes

Raphaell Maciel de Sousa · 12 days ago

you should use wraptopi for everywhere when u calculate some angle

0 Upvotes

coursera

NB

N Srujan Babu · 12 days ago

Thanks a lot. I forgot to use wraptopi for innovation. Now I'm getting good trajectory

0 Upvotes

Raphaell Maciel de Sousa · 11 days ago

great!

0 Upvotes

JS

Reply

Reply

1

JS

Reply

Reply