# 基于Kalman滤波的二维目标跟踪

## 一、选题内容

使用Kalman滤波跟踪一个二维匀速运动目标,匀加速目标，转弯目标，量测为 X，Y，X速度，Y速度。

## 二、卡尔曼滤波器

卡尔曼滤波器由一系列递归数学公式描述。它们提供了一种高效可计算的方法来估计过程的状态， 并使估计均方误差最小。 卡尔曼滤波器应用广泛且功能强大： 它可以估计信号的过去和当前状态， 甚至能估计将来的状态， 即使并不知道模型的确切性质。

### 1、被估计的过程信号

卡尔曼滤波器用于估计离散时间过程的状态变量。 这个离散时间过程由以下离散随机差分方程描述：

定义观测变量，得到量测方程：

随机信号和分别表示过程激励噪声和观测噪声。 假设它们为相互独立，正态分布的白色噪声：

实际系统中，过程激励噪声协方差矩阵Q和观测噪声协方差矩阵R 可能会随每次迭代计算而变化。 但在这儿我们假设它们是常数。

当控制函数或过程激励噪声为零时，差分方程(1.1)中的n\*n阶增益矩阵A将过去k-1时刻状态和现在的k时刻状态联系起来。实际中A可能随时间变化， 但在这儿假设为常数。n\*l阶矩阵 B代表可选的控制输入的增益。量测方程(1.2)中的m\*n阶矩阵H表示状态变量对测量变量的增益。 实际中H可能随时间变化，但在这儿假设为常数。

### 2、离散卡尔曼滤波器算法

卡尔曼滤波器用反馈控制的方法估计过程状态：滤波器估计过程某一时刻的状态，然后以（含噪声的）测量变量的方式获得反馈。因此卡尔曼滤波器可分为两个部分：时间更新方程和测量更新方程。 时间更新方程负责及时向前推算当前状态变量和误差协方差估计的值，以便为下一个时间状态构造先验估计。测量更新方程负责反馈――也就是说，它将先验估计和新的测量变量结合以构造改进的后验估计。

时间更新方程也可视为预估方程， 测量更新方程可视为校正方程。 最后的估计算法成为一种具有数值解的预估－校正算法。

（1）时间更新（预测）

①向前推算状态变量

②向前推算误差协方差

其中和为初始估计

（2）测量更新（校正）

①计算卡尔曼增益

② 由观测变量更新估计

③更新误差协方差

## 三、二维目标跟踪

### （1）、匀速运动

1、问题描述

在二维XY平面内跟踪一个匀速运动的目标；并在不考虑实际可行性的情况下，使目标先做一段匀速直线运动，转90度角后继续做一段匀速直线运动，并实现转弯跟踪。量测为目标的 X和Y坐标，估计目标的位置和速度。

2、过程模型

观测变量

3、滤波器方程和参数

取t=T

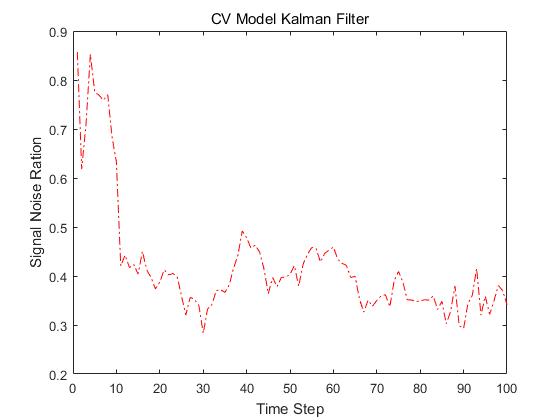
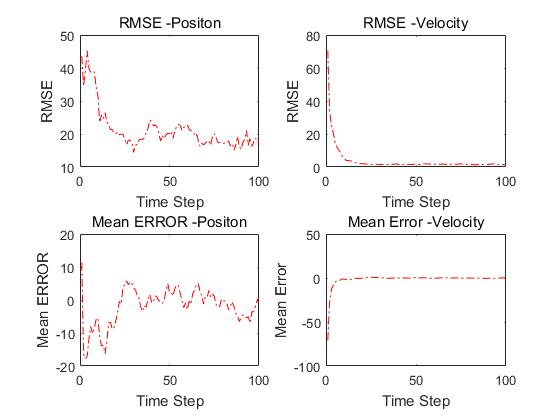
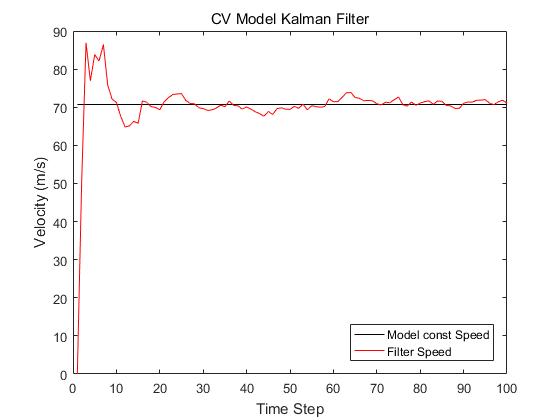
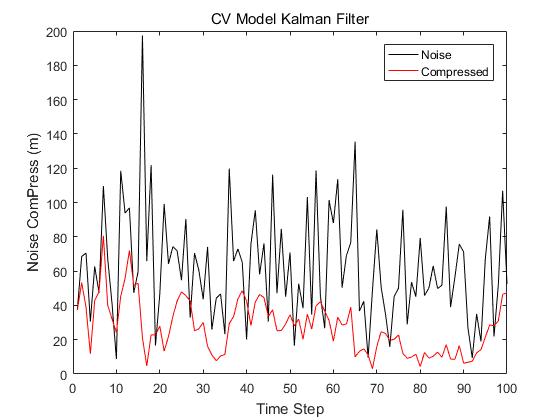
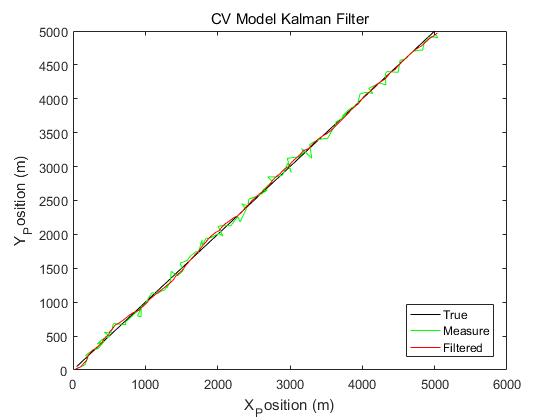
时间更新方程

测量更新方程

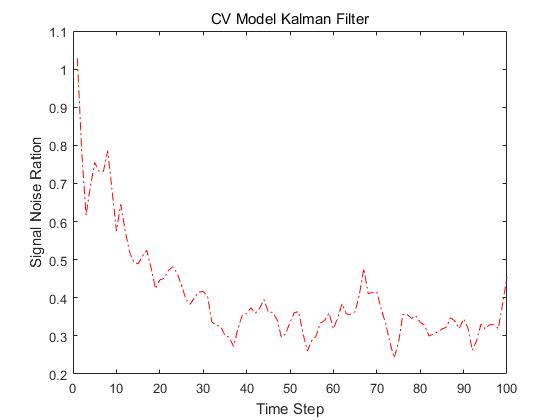
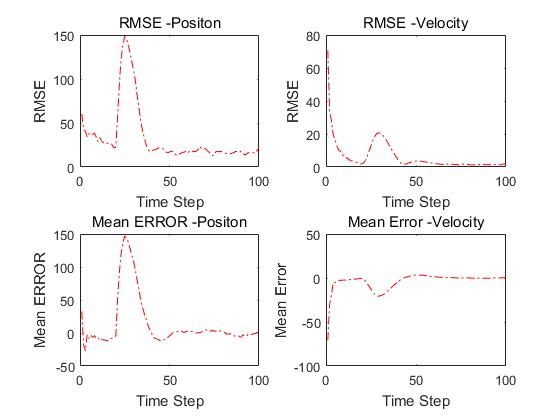
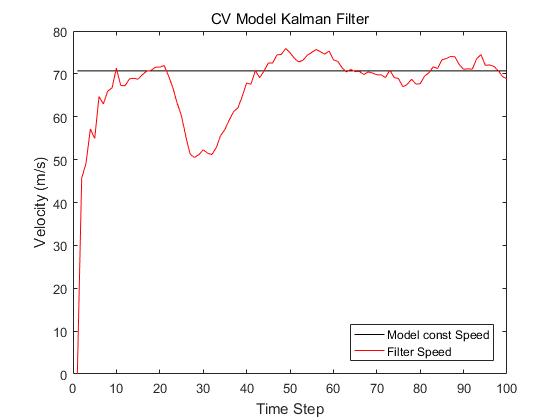
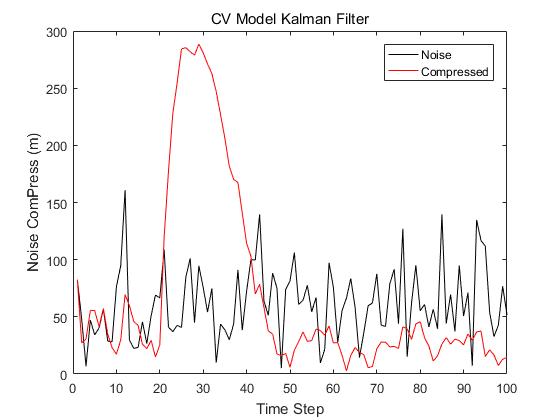
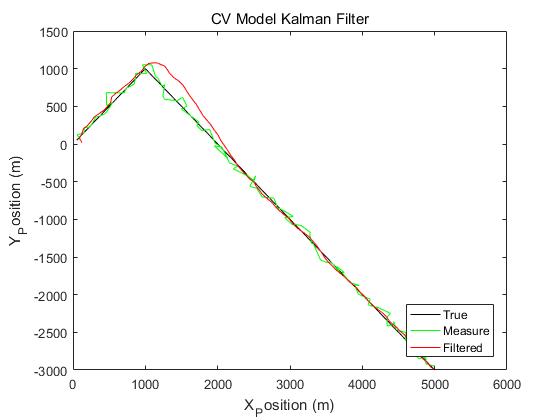
假设过程激励噪声方差 Q=1，令滤波器的初始条件取第一次的量测，令。

4、模拟实验

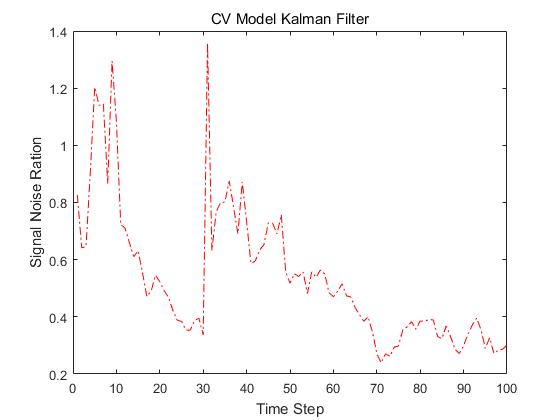
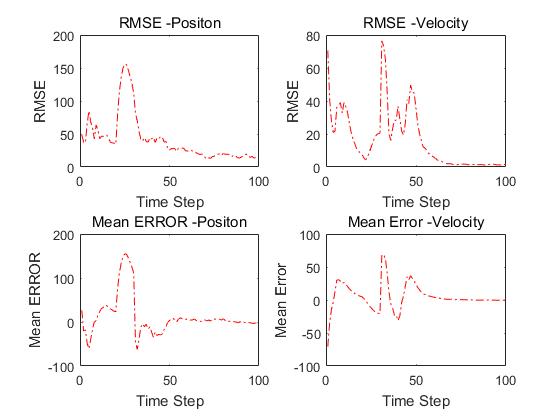
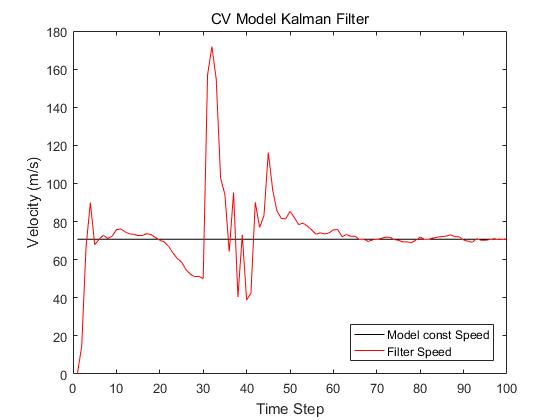
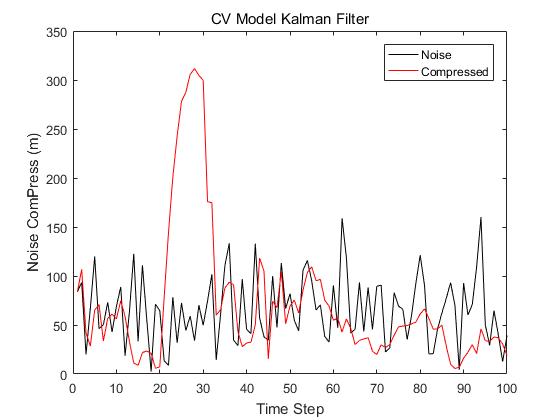
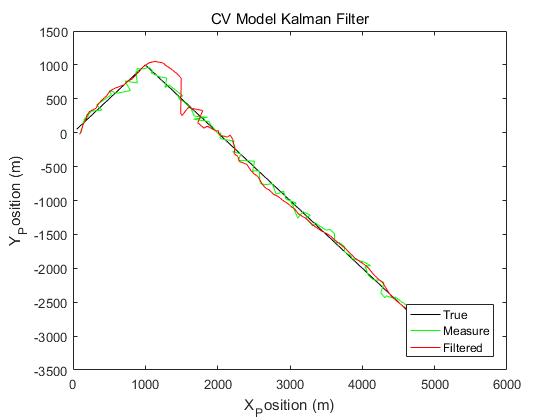
跟踪匀速直线运动，取初始位置（0，0），速度（50，50），加速度（0，0），T=1。



跟踪匀速直线的转弯运动，取初始位置（0，0），速度（50，50），加速度（0，0），直角转弯后速度为（50，-50）T=1。



转弯目标跟踪改进：观察以上转弯运动的仿真结果，可以发现，采用卡尔曼滤波可以实现较好的带转弯目标跟踪，但在转角处需要较长的时间进行调整，通过在迭代前增加一个前后两次速度夹角的比较来判别目标是否发生转弯，可以使系统较快地恢复平稳，以下为带转角的仿真效果，



5、结论

基于卡尔曼滤波跟踪二维匀速直线运动目标，可以达到较好的结果，但对于跟踪带转角的二维匀速直线运动目标，在转角处存在较大的误差，转角过后，可以较好地跟踪目标。

### （2）、匀加速运动

1、问题描述

在二维XY平面内跟踪一个匀加速运动的目标；并使目标先做一段匀减速直线运动，转90度角后继续做一段匀加速直线运动，实现转弯跟踪。量测为目标的 X和Y坐标，估计目标的位置、速度加速度。对于匀加速的曲线运动也可视为转弯运动。

2、过程模型

观测变量

3、滤波器方程和参数

取t=T

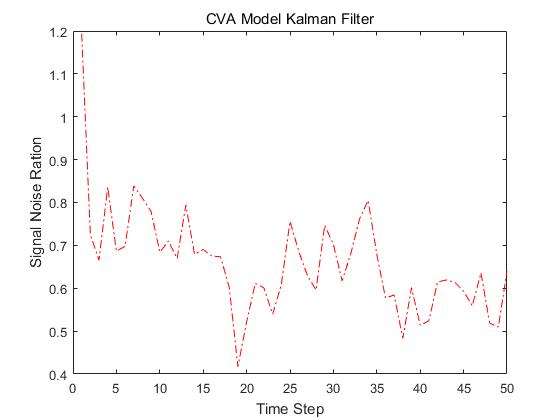
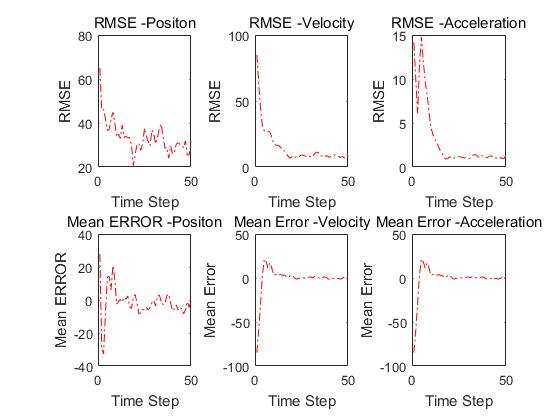
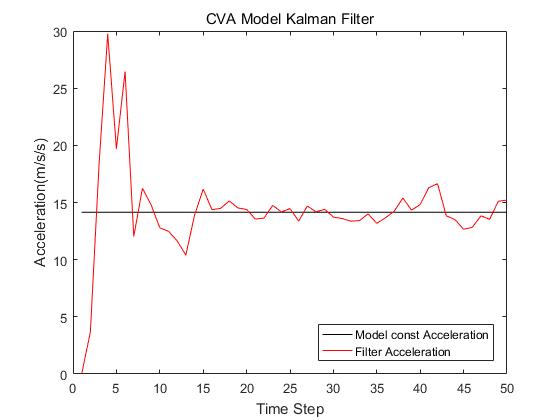
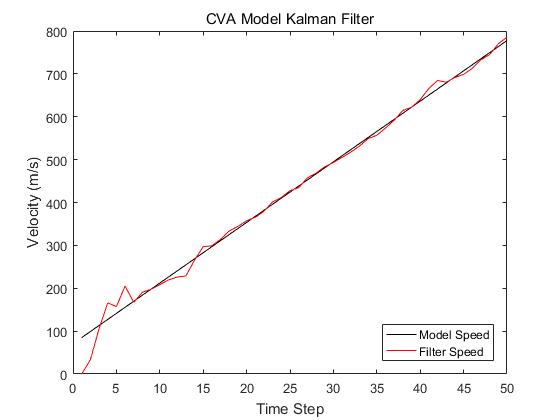
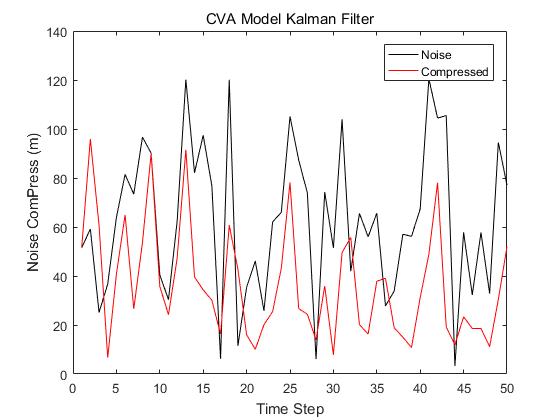
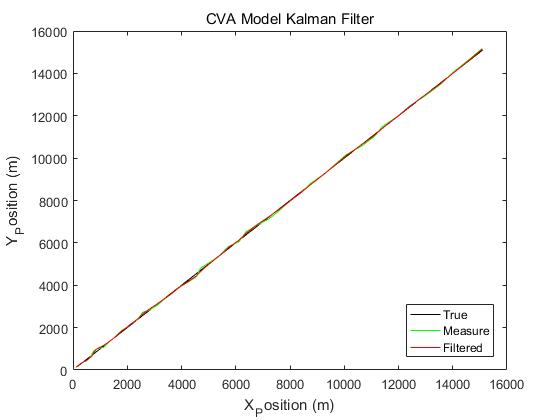
时间更新方程

测量更新方程

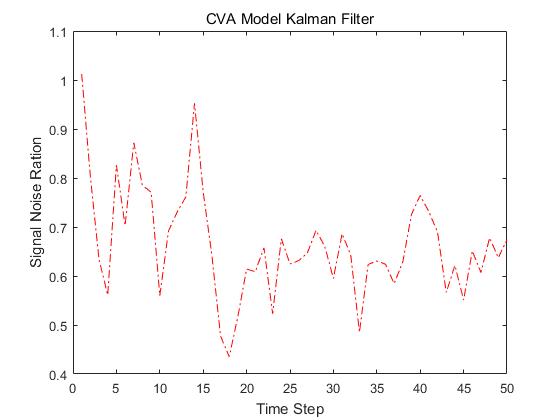
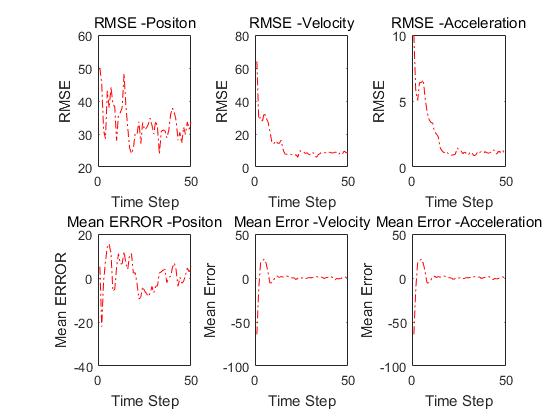
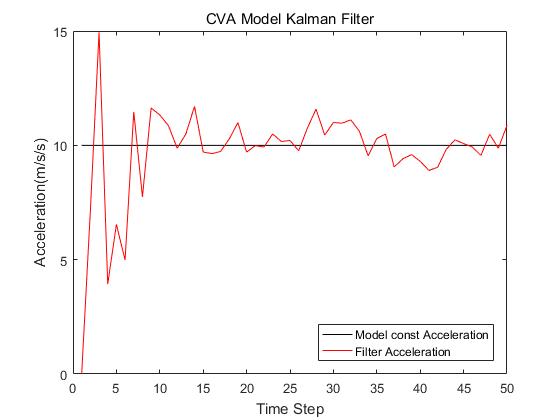
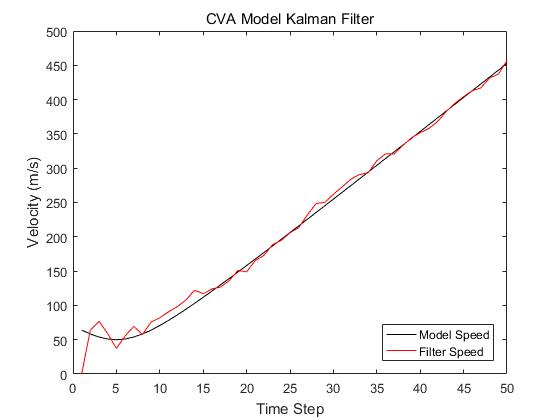
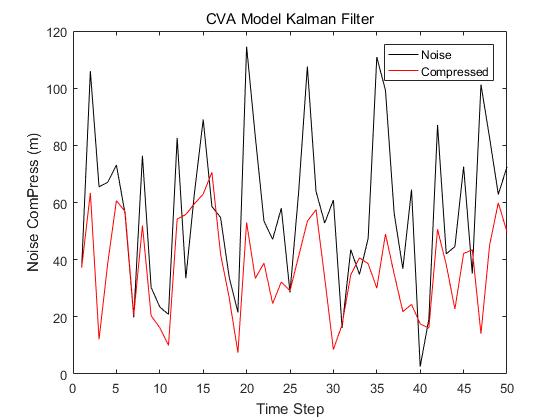
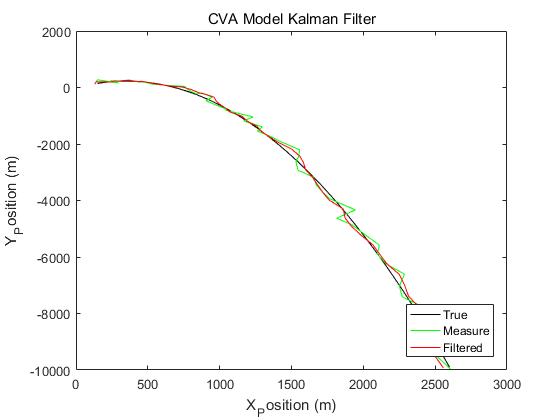
假设过程激励噪声方差 Q=1，令滤波器的初始条件取第一次的量测，令。

4、模拟实验

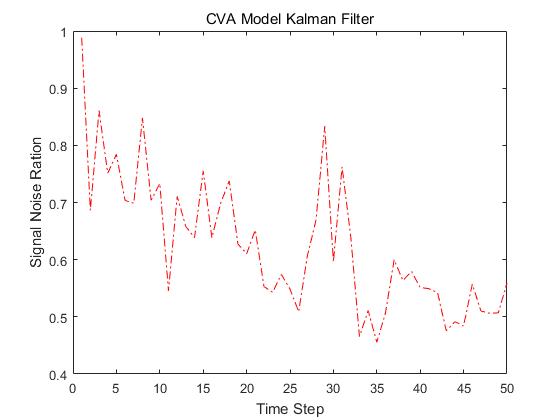
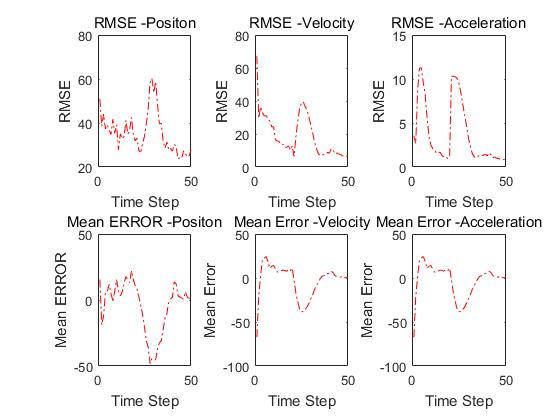
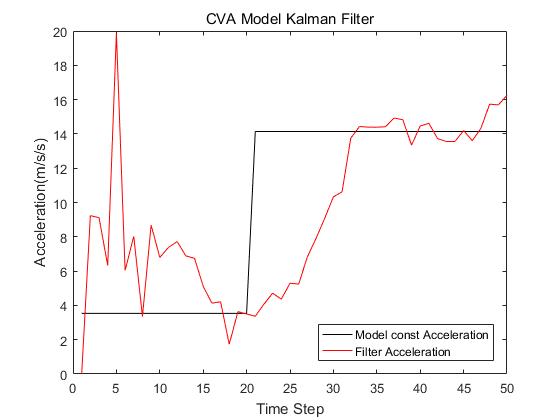
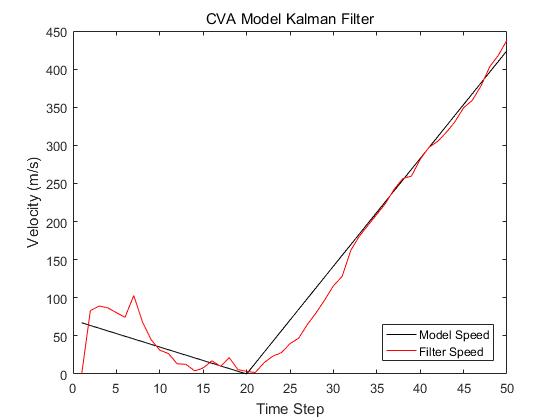
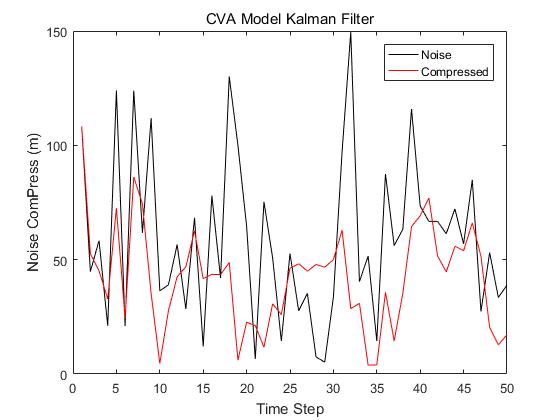
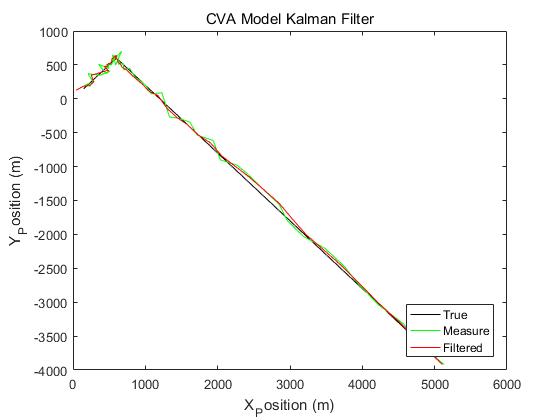
跟踪匀加速直线运动，取初始位置（100，100），速度（50，50），加速度（10，10），T=1。



跟踪匀加速曲线运动，取初始位置（100，100），速度（50，50），加速度（0，-10），T=1。



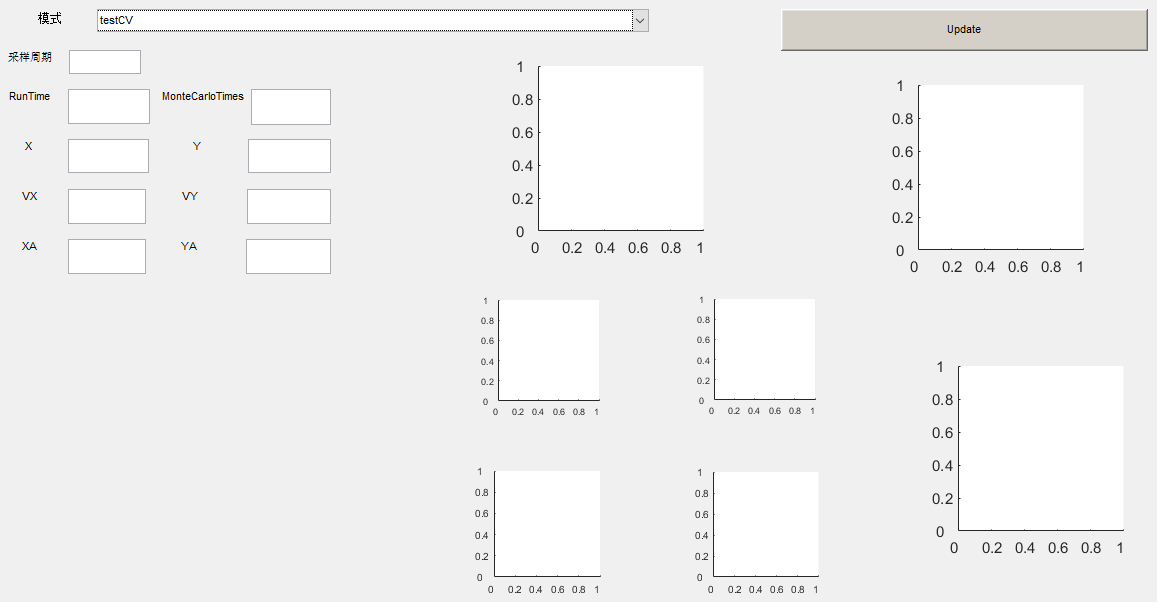
目标先做一段匀减速直线运动，转90度角后继续做一段匀加速直线运动，实现转弯跟踪，取初始位置（100，100），速度（50，50），加速度（-2.5，-2.5），减速至0后，直角转弯，加速度为（10，-10），T=1。



5、结论

基于卡尔曼滤波跟踪二维匀加速运动目标，可以达到较好的结果，但对于跟踪带转角的二维匀加速直线运动目标，在转角处存在较大的误差，转角过后，可以较好地跟踪目标。

## 四、GUI实现



模式下拉菜单:

testCV:匀速直线运动

testCV\_UA：匀加速直线运动

testCV\_Turn：转弯匀速直线运动

testCVA\_turn：转弯匀加速直线运动

testCV\_Turn2：带比较的转弯匀速直线运动

采样周期：采样周期

RunTime:运行次数

MontecarloTime:蒙特卡洛次数

X、Y:目标初始x,y坐标

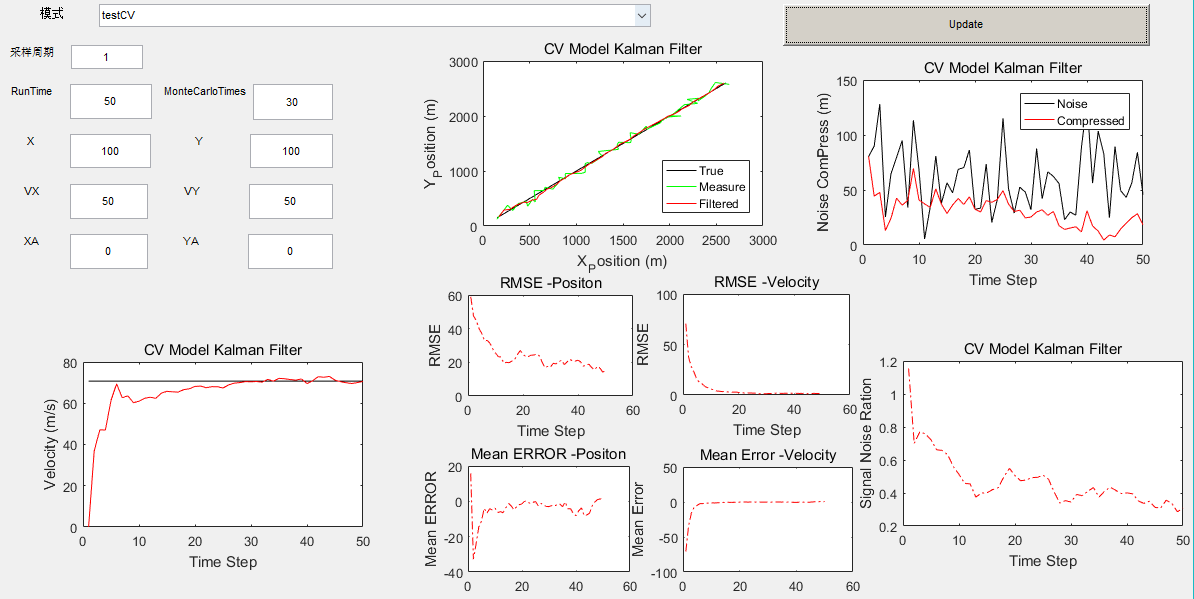
VX、VY：目标初始x,y速度

XA、XY: 目标初始x,y加速度

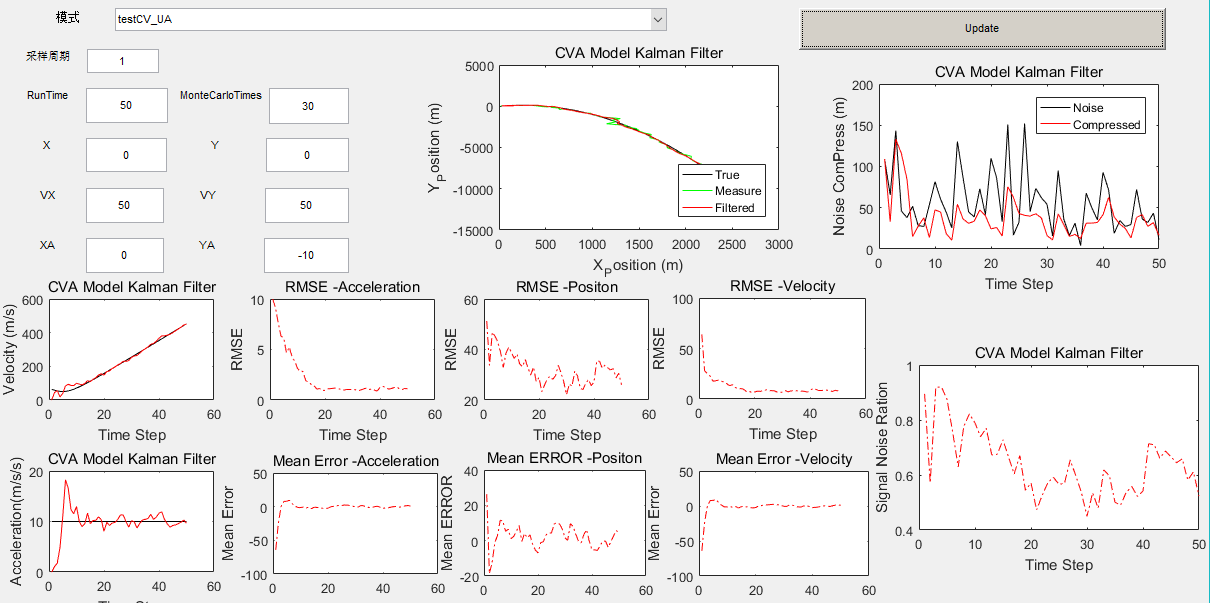
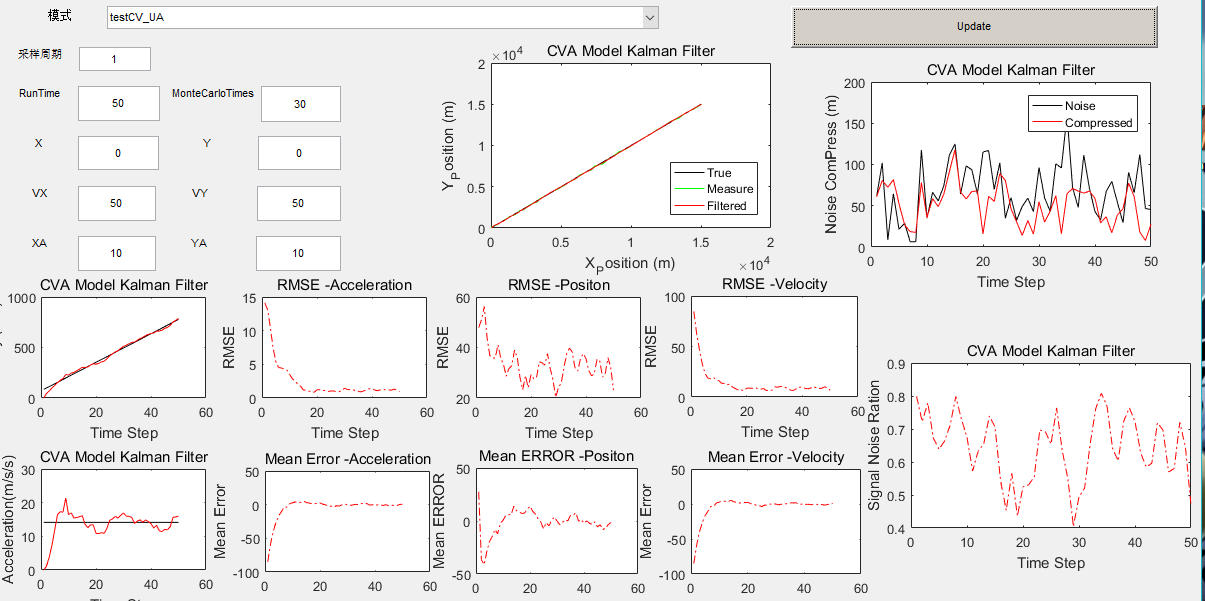
Update:更新界面

测试：

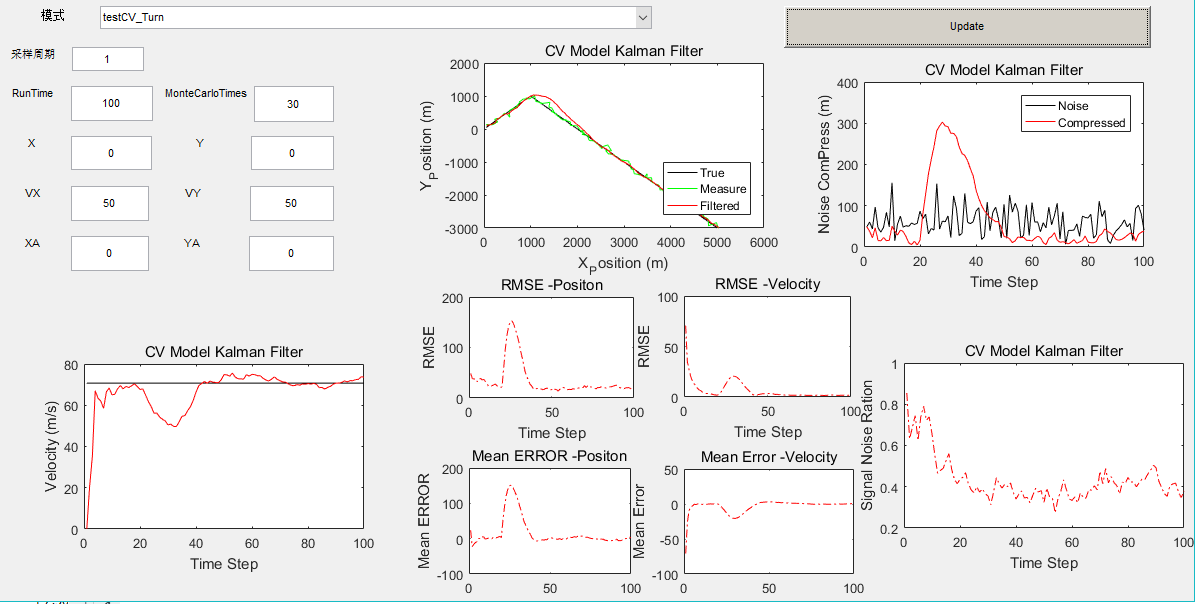
1、testCV:匀速直线运动



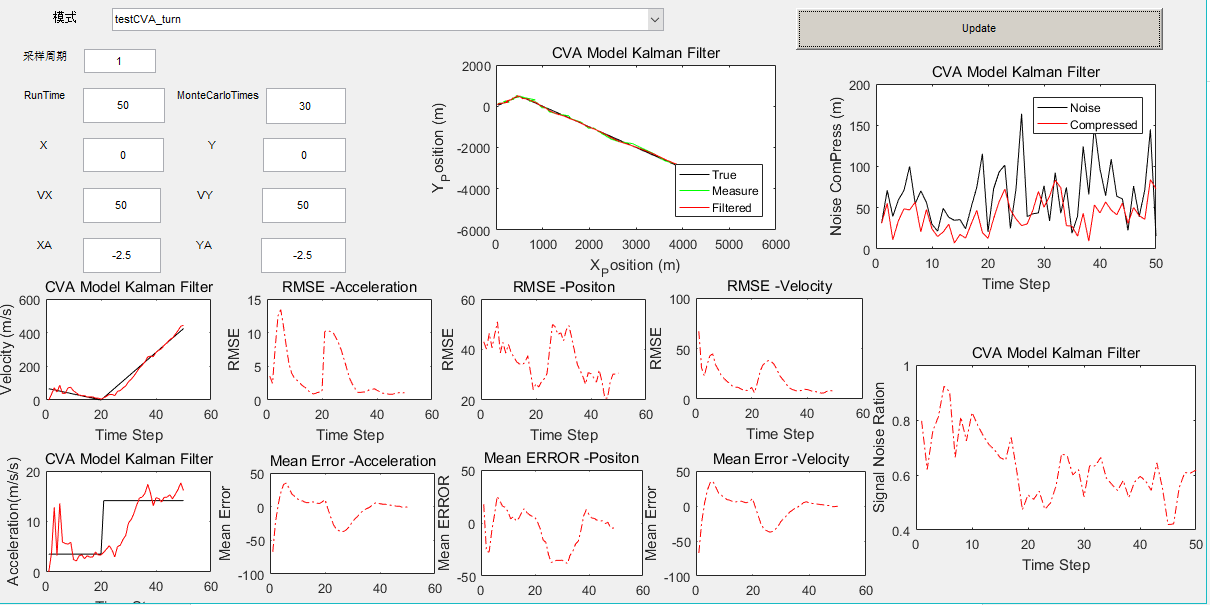
2、testCV\_UA：匀加速运动



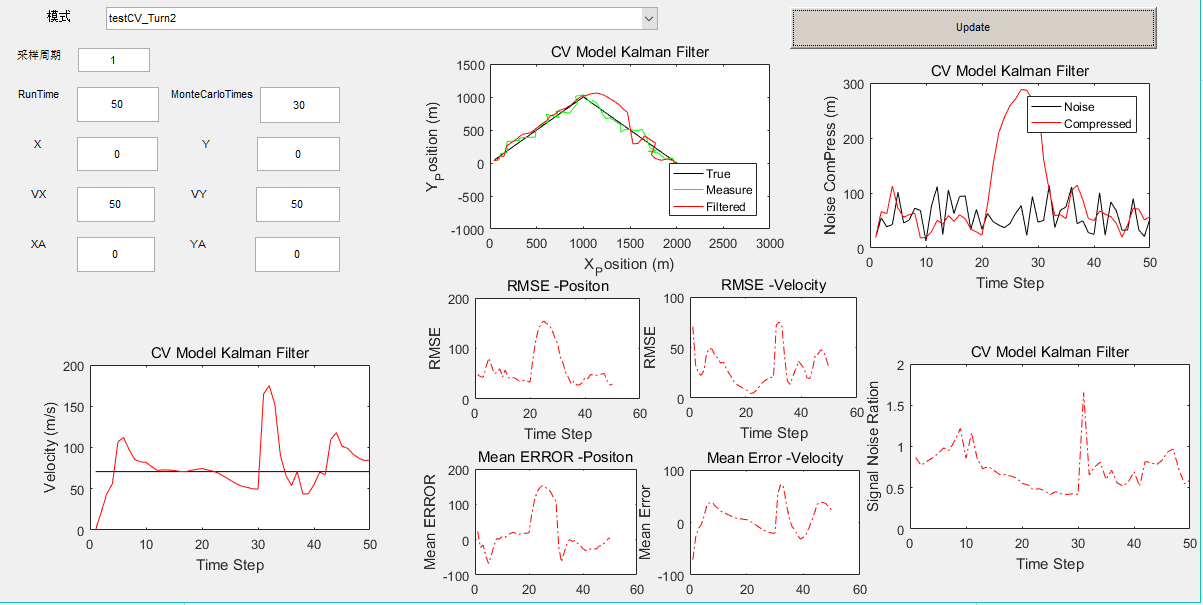
3、testCV\_Turn：转弯匀速直线运动



4、testCVA\_turn：转弯匀加速直线运动



5、testCV\_Turn2：带比较的转弯匀速直线运动



## 五、附录（主要matlab代码，matlab版本：R2016b,win10）

### 1、基于卡尔曼滤波的二维匀速直线运动的目标跟踪

%Test the standard kalman Filter use CV Model,

clear all

close all

T = 1; %sample period

FCV = [1 T 0 0; 0 1 0 0;0 0 1 T;0 0 0 1];

BCV = [T^2/2 0; T 0; 0 T^2/2; 0 T];

QCV = 1; %Model noise covariance

%Run Time

RunTime = 100;

%MontoCarlo Times

MonteCarloTimes = 30;

%True Target Measurement

Z\_Init = [0 0]';

V\_CV = [50 50]';

A\_CV = [0 0]';

for i = 1 : RunTime

Z\_Real(1, i) = Z\_Init(1,1) + V\_CV(1,1) \* (T \* i);

Z\_Real(2, i) = V\_CV(1,1);

Z\_Real(3, i) = A\_CV(1,1);

Z\_Real(4, i) = Z\_Init(2,1) + V\_CV(2,1) \* (T \* i);

Z\_Real(5, i) = V\_CV(2,1);

Z\_Real(6, i) = A\_CV(2,1);

end

%Noise Target Measurement

r = 50;

R = (diag([50 50]))^2;

Noise\_Add(1,:) = r \* randn(1,RunTime \* MonteCarloTimes);

Noise\_Add(2,:) = r \* randn(1,RunTime \* MonteCarloTimes);

for i = 1 : MonteCarloTimes

Run\_Start = (1 + (i-1)\*RunTime);

Run\_End = (i \* RunTime);

Z\_Noise(1,Run\_Start: Run\_End) = Z\_Real(1,:) + Noise\_Add(1,Run\_Start:Run\_End);

Z\_Noise(2,Run\_Start: Run\_End) = Z\_Real(4,:) + Noise\_Add(2,Run\_Start:Run\_End);

end

XCVE = zeros(6,MonteCarloTimes);

%Filter

for i = 1 : MonteCarloTimes

X\_Init(1,1) = Z\_Noise(1,1 + (i-1) \* RunTime);

X\_Init(2,1) = Z\_Noise(1,1 + (i-1) \* RunTime) - Z\_Noise(1,1 + (i-1) \* RunTime);

X\_Init(2,1) = X\_Init(2,1)/T;

X\_Init(3,1) = 0;

X\_Init(4,1) = Z\_Noise(2,1 + (i-1) \* RunTime);

X\_Init(5,1) = Z\_Noise(2,1 + (i-1) \* RunTime) - Z\_Noise(2,1 + (i-1) \* RunTime);

X\_Init(5,1) = X\_Init(5,1)/T;

X\_Init(6,1) = 0;

P\_Init = [1000 0 0 0 0 0; 0 2000 0 0 0 0; 0 0 0 0 0 0;

0 0 0 1000 0 0; 0 0 0 0 2000 0; 0 0 0 0 0 0];

XCVE(:, 1 + (i-1) \* RunTime) = X\_Init(:,1);

PCVE = P\_Init;

XCVI = X\_Init(:,1);

PCVI = P\_Init;

Z = zeros(2,1);

for j = 2 : RunTime

Z = Z\_Noise(:, j + (i-1) \* RunTime);

[X\_E, P\_E, S\_s, V\_v] = KalmanCV(FCV, BCV, QCV, XCVI, PCVI, R, Z);

XCVE(:,j + (i - 1) \* RunTime) = X\_E;

XCVI = X\_E;

PCVI = P\_E;

end

end

%Drawing Figure

%time step-Position figure

figure

plot(Z\_Real(1,1:RunTime),Z\_Real(4,1:RunTime),'K');

hold on

plot(Z\_Noise(1,1:RunTime),Z\_Noise(2,1:RunTime), 'G');

hold on

plot(XCVE(1,1:RunTime),XCVE(4,1:RunTime),'R');

xlabel('X\_Position (m)');

ylabel('Y\_Position (m)');

title('CV Model Kalman Filter');

legend('True', 'Measure', 'Filtered','Location','southeast');

%time step-noise compress figure

figure

plot(sqrt((Z\_Noise(1,1:RunTime) - Z\_Real(1,1:RunTime)).^2+(Z\_Noise(2,1:RunTime) - Z\_Real(4,1:RunTime)).^2),'K');

hold on

plot(sqrt((XCVE(1,1:RunTime) - Z\_Real(1,1:RunTime)).^2+(XCVE(4,1:RunTime) - Z\_Real(4,1:RunTime)).^2),'r');

xlabel('Time Step');

ylabel('Noise ComPress (m)');

title('CV Model Kalman Filter');

legend('Noise', 'Compressed', 'Location','northeast');

%time step-velocity figure

figure

Velocity = V\_CV \* ones(1,RunTime);

plot(1:RunTime,sqrt((Velocity(1,:)).^2+(Velocity(2,:)).^2),'K');

hold on

plot(1:RunTime,sqrt((XCVE(2,1:RunTime)).^2+(XCVE(5,1:RunTime)).^2),'r');

xlabel('Time Step');

ylabel('Velocity (m/s)');

title('CV Model Kalman Filter');

legend('Model const Speed', 'Filter Speed','Location','southeast');

%statistics

for i = 1 : RunTime

Last = ((MonteCarloTimes-1)\* RunTime+i);

X\_Mean\_ERROR(i)= mean(sqrt(XCVE(1,i:RunTime:Last).^2+XCVE(4,i:RunTime:Last).^2)) - sqrt(Z\_Real(1,i)^2+Z\_Real(4,i)^2);

V\_Mean\_ERROR(i)= mean(sqrt(XCVE(2,i:RunTime:Last).^2+XCVE(5,i:RunTime:Last).^2)) - sqrt(Z\_Real(2,i)^2+Z\_Real(5,i)^2);

Z\_X\_Real = sqrt(Z\_Real(1,i)^2+Z\_Real(4,i)^2)\*ones(1,MonteCarloTimes);

Z\_V\_Real = sqrt(Z\_Real(2,i)^2+Z\_Real(5,i)^2)\*ones(1,MonteCarloTimes);

X\_Cov\_ERROR(i) = norm(sqrt(XCVE(1,i:RunTime:Last).^2+XCVE(4,i:RunTime:Last).^2)-Z\_X\_Real)/sqrt(MonteCarloTimes);

V\_Cov\_ERROR(i) = norm(sqrt(XCVE(2,i:RunTime:Last).^2+XCVE(5,i:RunTime:Last).^2)-Z\_V\_Real)/sqrt(MonteCarloTimes);

SNA(i) = std(sqrt(XCVE(1,i:RunTime:Last).^2+XCVE(4,i:RunTime:Last).^2)-Z\_X\_Real)/r;

end

figure

subplot(2,2,1);

plot(X\_Cov\_ERROR,'r-.');

xlabel('Time Step');

ylabel('RMSE');

title('RMSE -Positon');

subplot(2,2,2);

plot(V\_Cov\_ERROR,'r-.');

xlabel('Time Step');

ylabel('RMSE');

title('RMSE -Velocity');

subplot(2,2,3);

plot(X\_Mean\_ERROR,'r-.');

xlabel('Time Step');

ylabel('Mean ERROR');

title('Mean ERROR -Positon');

subplot(2,2,4);

plot(V\_Mean\_ERROR,'r-.');

xlabel('Time Step');

ylabel('Mean Error');

title('Mean Error -Velocity');

%SNR

figure

plot(SNA,'r-.');

xlabel('Time Step');

ylabel('Signal Noise Ration');

title('CV Model Kalman Filter');

### 2、CV模型的卡尔曼滤波器

function [XCVE, PCVE, Ss, Vv] = KalmanCV(F, B, Q, XCVI, PCVI, R, Z)

%KalmanCV the Standard Kalman Filter Use CV Model

%Input:

%F one step transfer Matrix

%B system noise driver Matrix

%Q system noise covariance Matrix

%XCVI the last state estimate Matrix

%PCVI the last covariance of the state Matrix

%R the measurement noise covariance Matrix

%Z the measurement Matrix

%Output:

%XCVE the updated state estimate

%PCVE the updated covariance of the state

%Ss the measurement prediction coviance

%Vv the innovation or measurement residual

XI(1:2,1) = XCVI(1:2,1);

XI(3:4,1) = XCVI(4:5,1);

PI(1:2,1:2) = PCVI(1:2,1:2);

PI(3:4,3:4) = PCVI(4:5,4:5);

XP = F \* XI;

PP = F \* PI \* F' + B \* Q \* B';

H = [1 0 0 0;0 0 1 0];

Ss = H \* PP \* H' + R;

K = (PP \* H')/ Ss;

PE=(eye(4)-K\*H)\*PP;

Vv = Z - H \* XP;

XE = XP + K \* Vv;

XCVE = [(XE(1:2,1))' 0 (XE(3:4,1))' 0]';

PCVE = [PE(1:2,1:2) zeros(2,4)

0 0 0 0 0 0

zeros(2,3) PE(3:4,3:4) zeros(2,1)

0 0 0 0 0 0];

End

### 3、基于卡尔曼滤波跟踪二维匀加速直线运动目标，在二维平面内，当加速度方向和速度方向相同时，即为匀加速直线运动，当加速度方向和速度方向不同时，即为曲线运动，也可视为转弯目标

clear all

close all

T = 1; %sample period

FCV = [1 T T^2/2 0 0 0; 0 1 T 0 0 0;0 0 1 0 0 0;

0 0 0 1 T T^2/2; 0 0 0 0 1 T;0 0 0 0 0 1];

QCV = 1; %Model noise covariance

%Run Time

RunTime = 50;

%MontoCarlo Times

MonteCarloTimes = 30;

%True Target Measurement

Z\_Init = [100 100]';

V\_Init = [50 50]';

A\_CV = [0 -10]';

for i = 1 : RunTime

Z\_Real(1, i) = Z\_Init(1,1) + V\_Init(1,1) \* (T \* i)+ 0.5\*A\_CV(1,1)\*(T \* i)^2;

Z\_Real(2, i) = V\_Init(1,1) + A\_CV(1,1)\*(T \* i);

Z\_Real(3, i) = A\_CV(1,1);

Z\_Real(4, i) = Z\_Init(2,1) + V\_Init(2,1) \* (T \* i)+ 0.5\*A\_CV(2,1)\*(T \* i)^2;

Z\_Real(5, i) = V\_Init(2,1) + A\_CV(2,1)\*(T \* i);

Z\_Real(6, i) = A\_CV(2,1);

end

%Noise Target Measurement

r = 50;

R = (diag([r r]))^2;

Noise\_Add(1,:) = r \* randn(1,RunTime \* MonteCarloTimes);

Noise\_Add(2,:) = r \* randn(1,RunTime \* MonteCarloTimes);

for i = 1 : MonteCarloTimes

Run\_Start = (1 + (i-1)\*RunTime);

Run\_End = (i \* RunTime);

Z\_Noise(1,Run\_Start: Run\_End) = Z\_Real(1,:) + Noise\_Add(1,Run\_Start:Run\_End);

Z\_Noise(2,Run\_Start: Run\_End) = Z\_Real(4,:) + Noise\_Add(2,Run\_Start:Run\_End);

end

XCVE = zeros(6,MonteCarloTimes);

%Filter

for i = 1 : MonteCarloTimes

X\_Init(1,1) = Z\_Noise(1,1 + (i-1) \* RunTime);

X\_Init(2,1) = Z\_Noise(1,1 + (i-1) \* RunTime) - Z\_Noise(1,1 + (i-1) \* RunTime);

X\_Init(2,1) = X\_Init(2,1)/T;

X\_Init(3,1) = 0;

X\_Init(4,1) = Z\_Noise(2,1 + (i-1) \* RunTime);

X\_Init(5,1) = Z\_Noise(2,1 + (i-1) \* RunTime) - Z\_Noise(2,1 + (i-1) \* RunTime);

X\_Init(5,1) = X\_Init(5,1)/T;

X\_Init(6,1) = 0;

P\_Init = [1000 0 0 0 0 0; 0 2000 0 0 0 0; 0 0 500 0 0 0;

0 0 0 1000 0 0; 0 0 0 0 2000 0; 0 0 0 0 0 500];

XCVE(:, 1 + (i-1) \* RunTime) = X\_Init(:,1);

PCVE = P\_Init;

XCVI = X\_Init(:,1);

PCVI = P\_Init;

Z = zeros(2,1);

for j = 2 : RunTime

Z = Z\_Noise(:, j + (i-1) \* RunTime);

[X\_E, P\_E, S\_s, V\_v] = KalmanCVA(FCV,QCV,XCVI, PCVI, R, Z);

XCVE(:,j + (i - 1) \* RunTime) = X\_E;

XCVI = X\_E;

PCVI = P\_E;

end

end

%Drawing Figure

%time step-Position figure

figure

plot(Z\_Real(1,1:RunTime),Z\_Real(4,1:RunTime),'K');

hold on

plot(Z\_Noise(1,1:RunTime),Z\_Noise(2,1:RunTime), 'G');

hold on

plot(XCVE(1,1:RunTime),XCVE(4,1:RunTime),'R');

xlabel('X\_Position (m)');

ylabel('Y\_Position (m)');

title('CVA Model Kalman Filter');

legend('True', 'Measure', 'Filtered','Location','southeast');

%time step-noise compress figure

figure

plot(sqrt((Z\_Noise(1,1:RunTime) - Z\_Real(1,1:RunTime)).^2+(Z\_Noise(2,1:RunTime) - Z\_Real(4,1:RunTime)).^2),'K');

hold on

plot(sqrt((XCVE(1,1:RunTime) - Z\_Real(1,1:RunTime)).^2+(XCVE(4,1:RunTime) - Z\_Real(4,1:RunTime)).^2),'r');

xlabel('Time Step');

ylabel('Noise ComPress (m)');

title('CVA Model Kalman Filter');

legend('Noise', 'Compressed', 'Location','northeast');

%time step-velocity figure

figure

Velocity(1,:) = Z\_Real(2,:);

Velocity(2,:) = Z\_Real(5,:);

plot([1:RunTime],sqrt((Velocity(1,:)).^2+(Velocity(2,:)).^2),'K');

hold on

plot([1:RunTime],sqrt((XCVE(2,1:RunTime)).^2+(XCVE(5,1:RunTime)).^2),'r');

xlabel('Time Step');

ylabel('Velocity (m/s)');

title('CVA Model Kalman Filter');

legend('Model Speed', 'Filter Speed','Location','southeast');

%time step-Acceleration figure

figure

Acceleration(1,:) = Z\_Real(3,:);

Acceleration(2,:) = Z\_Real(6,:);

plot([1:RunTime],sqrt((Acceleration(1,:)).^2+(Acceleration(2,:)).^2),'K');

hold on

plot([1:RunTime],sqrt((XCVE(3,1:RunTime)).^2+(XCVE(6,1:RunTime)).^2),'r');

xlabel('Time Step');

ylabel('Acceleration(m/s/s)');

title('CVA Model Kalman Filter');

legend('Model const Acceleration', 'Filter Acceleration','Location','southeast');

%statistics

for i = 1 : RunTime

Last = ((MonteCarloTimes-1)\* RunTime+i);

X\_Mean\_ERROR(i)= mean(sqrt(XCVE(1,i:RunTime:Last).^2+XCVE(4,i:RunTime:Last).^2)) - sqrt(Z\_Real(1,i)^2+Z\_Real(4,i)^2);

V\_Mean\_ERROR(i)= mean(sqrt(XCVE(2,i:RunTime:Last).^2+XCVE(5,i:RunTime:Last).^2)) - sqrt(Z\_Real(2,i)^2+Z\_Real(5,i)^2);

A\_Mean\_ERROR(i)= mean(sqrt(XCVE(3,i:RunTime:Last).^2+XCVE(6,i:RunTime:Last).^2)) - sqrt(Z\_Real(3,i)^2+Z\_Real(6,i)^2);

Z\_X\_Real = sqrt(Z\_Real(1,i)^2+Z\_Real(4,i)^2)\*ones(1,MonteCarloTimes);

Z\_V\_Real = sqrt(Z\_Real(2,i)^2+Z\_Real(5,i)^2)\*ones(1,MonteCarloTimes);

Z\_A\_Real = sqrt(Z\_Real(3,i)^2+Z\_Real(6,i)^2)\*ones(1,MonteCarloTimes);

X\_Cov\_ERROR(i) = norm(sqrt(XCVE(1,i:RunTime:Last).^2+XCVE(4,i:RunTime:Last).^2)-Z\_X\_Real)/sqrt(MonteCarloTimes);

V\_Cov\_ERROR(i) = norm(sqrt(XCVE(2,i:RunTime:Last).^2+XCVE(5,i:RunTime:Last).^2)-Z\_V\_Real)/sqrt(MonteCarloTimes);

A\_Cov\_ERROR(i) = norm(sqrt(XCVE(3,i:RunTime:Last).^2+XCVE(6,i:RunTime:Last).^2)-Z\_A\_Real)/sqrt(MonteCarloTimes);

SNA(i) = std(sqrt(XCVE(1,i:RunTime:Last).^2+XCVE(4,i:RunTime:Last).^2)-Z\_X\_Real)/r;

end

figure

subplot(2,3,1);

plot(X\_Cov\_ERROR,'r-.');

xlabel('Time Step');

ylabel('RMSE');

title('RMSE -Positon');

subplot(2,3,2);

plot(V\_Cov\_ERROR,'r-.');

xlabel('Time Step');

ylabel('RMSE');

title('RMSE -Velocity');

subplot(2,3,3);

plot(A\_Cov\_ERROR,'r-.');

xlabel('Time Step');

ylabel('RMSE');

title('RMSE -Acceleration');

subplot(2,3,4);

plot(X\_Mean\_ERROR,'r-.');

xlabel('Time Step');

ylabel('Mean ERROR');

title('Mean ERROR -Positon');

subplot(2,3,5);

plot(V\_Mean\_ERROR,'r-.');

xlabel('Time Step');

ylabel('Mean Error');

title('Mean Error -Velocity');

subplot(2,3,6);

plot(V\_Mean\_ERROR,'r-.');

xlabel('Time Step');

ylabel('Mean Error');

title('Mean Error -Acceleration');

%SNR

figure

plot(SNA,'r-.');

xlabel('Time Step');

ylabel('Signal Noise Ration');

title('CVA Model Kalman Filter');

### 4、CVA模型的卡尔曼滤波器

function [XCVE, PCVE, Ss, Vv] = KalmanCVA(F,Q,XCVI, PCVI, R, Z)

%Input:

%F one step transfer Matrix

%Q system noise covariance Matrix

%XCVI the last state estimate Matrix

%PCVI the last covariance of the state Matrix

%R the measurement noise covariance Matrix

%Z the measurement Matrix

%Output:

%XCVE the updated state estimate

%PCVE the updated covariance of the state

%Ss the measurement prediction coviance

%Vv the innovation or measurement residual

XI = XCVI;

PI = PCVI;

XP = F \* XI;

Q= diag([Q Q Q Q Q Q]);

PP = F \* PI \* F'+Q;

H = [1 0 0 0 0 0;0 0 0 1 0 0];

Ss = H \* PP \* H' + R;

K = (PP \* H')/ Ss;

PE=(eye(6)-K\*H)\*PP;

Vv = Z - H \* XP;

XE = XP + K \* Vv;

XCVE = XE;

PCVE = PE;

End

### 5、GUI

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

popup\_sel\_index = get(handles.popupmenu1, 'Value');

RT=str2double(get(handles.RT,'String'));

MT=str2double(get(handles.MT,'String'));

X=str2double(get(handles.X,'String'));

Y=str2double(get(handles.Y,'String'));

VX=str2double(get(handles.VX,'String'));

VY=str2double(get(handles.VY,'String'));

XA=str2double(get(handles.XA,'String'));

YA=str2double(get(handles.YA,'String'));

T=str2double(get(handles.T,'String'));

switch popup\_sel\_index

case 1

[Z\_Real,Z\_Noise,XCVE,RunTime,MonteCarloTimes,r]=FtestCV\_A(T,RT,MT, [X Y]', [VX VY]', [XA YA]');

case 2

[Z\_Real,Z\_Noise,XCVE,RunTime,MonteCarloTimes,r]=FtestCV(T,RT,MT, [X Y]', [VX VY]', [XA YA]');

case 3

[Z\_Real,Z\_Noise,XCVE,RunTime,MonteCarloTimes,r]=FtestCV\_Turn(T,RT,MT, [X Y]', [VX VY]', [XA YA]');

case 4

XA=-(VX/20);

YA=-(VY/20);

[Z\_Real,Z\_Noise,XCVE,RunTime,MonteCarloTimes,r]=FtestCVA\_Turn(T,RT,MT, [X Y]', [VX VY]', [XA YA]');

case 5

[Z\_Real,Z\_Noise,XCVE,RunTime,MonteCarloTimes,r]=FtestCV\_Turn2(T,RT,MT, [X Y]', [VX VY]', [XA YA]');

end

axes(handles.axes1);

cla;

%time step-Position figure

plot(Z\_Real(1,1:RunTime),Z\_Real(4,1:RunTime),'K');

hold on

plot(Z\_Noise(1,1:RunTime),Z\_Noise(2,1:RunTime), 'G');

hold on

plot(XCVE(1,1:RunTime),XCVE(4,1:RunTime),'R');

xlabel('X\_Position (m)');

ylabel('Y\_Position (m)');

if popup\_sel\_index==1 || popup\_sel\_index==3 || popup\_sel\_index==5

title('CV Model Kalman Filter');

else

title('CVA Model Kalman Filter');

end

legend('True', 'Measure', 'Filtered','Location','southeast');

axes(handles.axes2);

cla;

%time step-noise compress figure

plot(sqrt((Z\_Noise(1,1:RunTime) - Z\_Real(1,1:RunTime)).^2+(Z\_Noise(2,1:RunTime) - Z\_Real(4,1:RunTime)).^2),'K');

hold on

plot(sqrt((XCVE(1,1:RunTime) - Z\_Real(1,1:RunTime)).^2+(XCVE(4,1:RunTime) - Z\_Real(4,1:RunTime)).^2),'r');

xlabel('Time Step');

ylabel('Noise ComPress (m)');

if popup\_sel\_index==1 || popup\_sel\_index==3|| popup\_sel\_index==5

title('CV Model Kalman Filter');

else

title('CVA Model Kalman Filter');

end

legend('Noise', 'Compressed', 'Location','northeast');

if popup\_sel\_index==1 || popup\_sel\_index==3|| popup\_sel\_index==5

axes(handles.axes10);

cla;

set(handles.axes10,'Visible','off');

axes(handles.axes11);

cla;

set(handles.axes11,'Visible','off');

axes(handles.axes12);

cla;

set(handles.axes12,'Visible','off');

axes(handles.axes13);

cla;

set(handles.axes13,'Visible','off');

axes(handles.axes3);

cla;

set(handles.axes3,'Visible','on');

%time step-velocity figure

Velocity(1,:) = Z\_Real(2,:);

Velocity(2,:) = Z\_Real(5,:);

plot(1:RunTime,sqrt((Velocity(1,:)).^2+(Velocity(2,:)).^2),'K');

hold on

plot(1:RunTime,sqrt((XCVE(2,1:RunTime)).^2+(XCVE(5,1:RunTime)).^2),'r');

xlabel('Time Step');

ylabel('Velocity (m/s)');

title('CV Model Kalman Filter');

%legend('Model Speed', 'Filter Speed','Location','southeast');

end

if popup\_sel\_index==2 ||popup\_sel\_index==4

axes(handles.axes3);

cla;

set(handles.axes3,'Visible','off');

axes(handles.axes10);

cla;

set(handles.axes10,'Visible','on');

%time step-velocity figure

Velocity(1,:) = Z\_Real(2,:);

Velocity(2,:) = Z\_Real(5,:);

plot(1:RunTime,sqrt((Velocity(1,:)).^2+(Velocity(2,:)).^2),'K');

hold on

plot(1:RunTime,sqrt((XCVE(2,1:RunTime)).^2+(XCVE(5,1:RunTime)).^2),'r');

xlabel('Time Step');

ylabel('Velocity (m/s)');

title('CVA Model Kalman Filter');

%legend('Model Speed', 'Filter Speed','Location','southeast');

axes(handles.axes12);

cla;

set(handles.axes12,'Visible','on');

Acceleration(1,:) = Z\_Real(3,:);

Acceleration(2,:) = Z\_Real(6,:);

plot([1:RunTime],sqrt((Acceleration(1,:)).^2+(Acceleration(2,:)).^2),'K');

hold on

plot([1:RunTime],sqrt((XCVE(3,1:RunTime)).^2+(XCVE(6,1:RunTime)).^2),'r');

xlabel('Time Step');

ylabel('Acceleration(m/s/s)');

title('CVA Model Kalman Filter');

% legend('Model const Acceleration', 'Filter Acceleration','Location','southeast');

end

%statistics

for i = 1 : RunTime

Last = ((MonteCarloTimes-1)\* RunTime+i);

X\_Mean\_ERROR(i)= mean(sqrt(XCVE(1,i:RunTime:Last).^2+XCVE(4,i:RunTime:Last).^2)) - sqrt(Z\_Real(1,i)^2+Z\_Real(4,i)^2);

V\_Mean\_ERROR(i)= mean(sqrt(XCVE(2,i:RunTime:Last).^2+XCVE(5,i:RunTime:Last).^2)) - sqrt(Z\_Real(2,i)^2+Z\_Real(5,i)^2);

A\_Mean\_ERROR(i)= mean(sqrt(XCVE(3,i:RunTime:Last).^2+XCVE(6,i:RunTime:Last).^2)) - sqrt(Z\_Real(3,i)^2+Z\_Real(6,i)^2);

Z\_X\_Real = sqrt(Z\_Real(1,i)^2+Z\_Real(4,i)^2)\*ones(1,MonteCarloTimes);

Z\_V\_Real = sqrt(Z\_Real(2,i)^2+Z\_Real(5,i)^2)\*ones(1,MonteCarloTimes);

Z\_A\_Real = sqrt(Z\_Real(3,i)^2+Z\_Real(6,i)^2)\*ones(1,MonteCarloTimes);

X\_Cov\_ERROR(i) = norm(sqrt(XCVE(1,i:RunTime:Last).^2+XCVE(4,i:RunTime:Last).^2)-Z\_X\_Real)/sqrt(MonteCarloTimes);

V\_Cov\_ERROR(i) = norm(sqrt(XCVE(2,i:RunTime:Last).^2+XCVE(5,i:RunTime:Last).^2)-Z\_V\_Real)/sqrt(MonteCarloTimes);

A\_Cov\_ERROR(i) = norm(sqrt(XCVE(3,i:RunTime:Last).^2+XCVE(6,i:RunTime:Last).^2)-Z\_A\_Real)/sqrt(MonteCarloTimes);

SNA(i) = std(sqrt(XCVE(1,i:RunTime:Last).^2+XCVE(4,i:RunTime:Last).^2)-Z\_X\_Real)/r;

end

axes(handles.axes4);

cla;

plot(X\_Cov\_ERROR,'r-.');

xlabel('Time Step');

ylabel('RMSE');

title('RMSE -Positon');

axes(handles.axes6);

cla;

plot(V\_Cov\_ERROR,'r-.');

xlabel('Time Step');

ylabel('RMSE');

title('RMSE -Velocity');

axes(handles.axes7);

cla;

plot(X\_Mean\_ERROR,'r-.');

xlabel('Time Step');

ylabel('Mean ERROR');

title('Mean ERROR -Positon');

axes(handles.axes8);

cla;

plot(V\_Mean\_ERROR,'r-.');

xlabel('Time Step');

ylabel('Mean Error');

title('Mean Error -Velocity');

if popup\_sel\_index==2 ||popup\_sel\_index==4

%set(handles.axe3,'Visible','off');

axes(handles.axes11);

cla;

set(handles.axes11,'Visible','on');

plot(A\_Cov\_ERROR,'r-.');

xlabel('Time Step');

ylabel('RMSE');

title('RMSE -Acceleration');

axes(handles.axes13);

cla;

set(handles.axes13,'Visible','on');

plot(V\_Mean\_ERROR,'r-.');

xlabel('Time Step');

ylabel('Mean Error');

title('Mean Error -Acceleration');

end

axes(handles.axes5);

cla;

%SNR

plot(SNA,'r-.');

xlabel('Time Step');

ylabel('Signal Noise Ration');

if popup\_sel\_index==1 || popup\_sel\_index==3 || popup\_sel\_index==5

title('CV Model Kalman Filter');

else

title('CVA Model Kalman Filter');

end