北京大學

本科实验报告

课程名称：　自动控制理论

姓　　名：　金镇雄

学　　院：　元培

　　系：　智能科学系

专　　业：　智能科学与技术

年　　级：　19

学　　号：　1900094619

2022 年　　5 月　　5 日

实验二、卡尔曼滤波实验

一、 实验目的

- 了解卡尔曼滤波器的原理
- 了解基于卡尔曼滤波器的物体跟踪
- 了解系统噪声和观测噪声的协方差矩阵对滤波结果的影响

二、 实验器材

PC 机、MATLAB

三、 实验原理

1. 概述

卡尔曼滤波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。卡尔曼滤波是一种递归的估计，即只要获知上一时刻状态的估计值以及当前状态的观测值就可以计算出当前状态的估计值。

我们无法建立一个完美的模型来消除系统的不确定因素，这些不确定性包括系统模型的自身缺陷、系统过程扰动以及得到观测数据的传感器的观测误差。这些误差往往会影响估计值，所以卡尔曼滤波算法通过最小化误差方差求解卡尔曼增益，以此来进行后验估计，已达到滤波效果。卡尔曼滤波有两个基本假设：状态转移方程与观测方程为线性方程；观测噪声和系统噪声均为服从正态分布的高斯噪声。

2. 离散卡尔曼滤波的基本模型

假设一离散线性动态系统的模型如下所示：

$$\begin{cases} X_k = AX_{k-1} + Bu_k + w_{k-1} \\ Z_k = HX_k + v_k \end{cases}$$

其中X_k为系统状态矩阵，Z_k为状态矩阵的观测量，A为状态转移矩阵，B为控制输入矩阵，H为状态观测矩阵，w_{k-1}为系统噪声向量，v_k为观测噪声向量。

3. 卡尔曼滤波的递归过程

卡尔曼滤波通过“预测”与“更新”两个过程来对系统的状态进行最优估计。预测过程主要利用时间更新方程建立对当前状态的先验估计，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值；更新过程负责反馈，利用测量更新方程在预估过程的先验估计值及当前测量变量的基础上建立起对当前状态的改进的后验估计。

4. 卡尔曼滤波的五个基本迭代公式

A. 符号说明

符号	定义
$\hat{X}_k$	当前时刻的状态的最优估计值
$\hat{X}_k^-$	由上一时刻的最优估计值得到的当前时刻状态的先验估计值
A	状态转移矩阵
P_k	当前时刻真实值与最优估计值之间的协方差矩阵
P_k^-	当前时刻真实值与预测值之间的协方差矩阵
K_k	卡尔曼滤波增益矩阵，其取值标准为使得误差方差阵为极小
H	观测矩阵
Q	系统噪声w_k的协方差
R	观测噪声v_k的协方差
I	单位矩阵

B. 预测过程

根据上一时刻的最优估计的结果，预测当前时刻的状态估计值：

$$\hat{X}_k^- = A\hat{X}_{k-1}$$

先验估计均方误差：

$$P_k^- = AP_{k-1}A^T + Q$$

C. 更新过程

由估计误差方差 0 最小原则得到卡尔曼滤波增益：

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

对当前时刻状态进行最优估计：

$$\hat{X}_k = \hat{X}_k^- + K_k\left(Z_k - H\hat{X}_k^-\right)$$

更新当前时刻的估计均方误差：

$$P_k = (I - K_k H)P_k^-$$

四、 实验内容

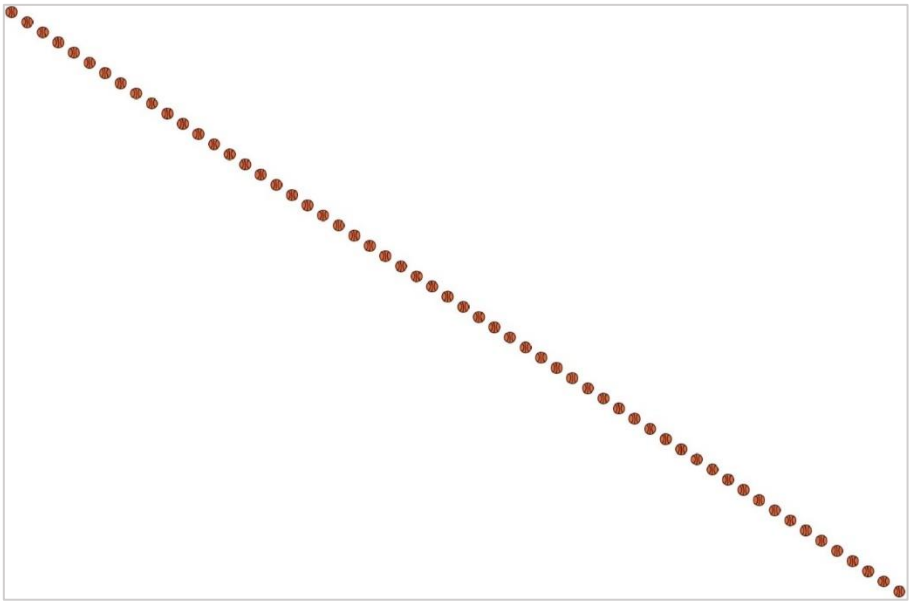
1. 问题描述

在本实验中简单模拟基于卡尔曼滤波器的物体跟踪。

假定一个小球从左上方到右下方移动，且其移动速度始终一致，追踪器（tracker）观测小球的位置，且其观测周期为 1s。由于某时刻追踪器观测得到的位置与小球实际位置间有一定的误差，需要用卡尔曼滤波器去除观测误差的作用。在整个物体跟踪的过程中，卡尔曼滤波器用之前观测到的值预测下一步的状态，然后根据观测值对自己作出修正，如

此不断迭代，以实现跟踪。

下图为小球的实际移动轨迹：



图一 实际移动轨迹

2. 系统分析与建模

定义观测向量：

$$Z = [x, y]^T$$

定义被观测的状态向量：

$$X = [x, \dot{x}, y, \dot{y}]^T$$

其中x, y 分别表示目标物体在坐标系中的 X 轴、Y 轴的坐标位置；$\dot{x}, \dot{y}$表示其微分，即物体的相对速度。假设物体不受外力的作用，即物体做匀速直线运动。

假设目标物体不受外力作用，但物体的位置可能受到随机扰动的影响，且此随机扰动不随时间变化，且符合高斯噪声规律，则由经典物理可知：

$$\begin{cases} x_k = x_{k-1} + \dot{x}_{k-1} \cdot \Delta t + w_1 \\ \dot{x}_k = \dot{x}_{k-1} + w_2 \end{cases}$$

$$\begin{cases} y_k = y_{k-1} + \dot{y}_{k-1} \cdot \Delta t + w_3 \\ \dot{y}_k = \dot{y}_{k-1} + w_4 \end{cases}$$

上式中Δt为采样周期，$w_i (i = 1,2,3,4)$为噪声。

将上式转换成矩阵形式：

$$\begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \\ y_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

另一方面，假设追踪器观测到的位置信息有符合高斯白噪声的误差，则观测方程为：

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

从而可得到t_k时刻被估计状态的状态方程和线性观测方程：

$$\begin{cases} X_k = AX_{k-1} + w_{k-1} \\ Z_k = HX_k + v_k \end{cases}$$

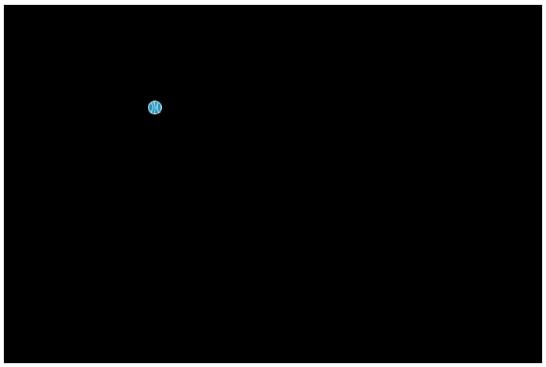
其中，$\mathrm{A} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$，$\mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$，$w_k = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$，$v_k = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$.

另外，定义w_k和v_k的协方差矩阵 Q 和 R，是非对角线元素都是零的对角矩阵（表示各分量上的噪声线性无关）。

3. 实验步骤

A. 追踪器设计

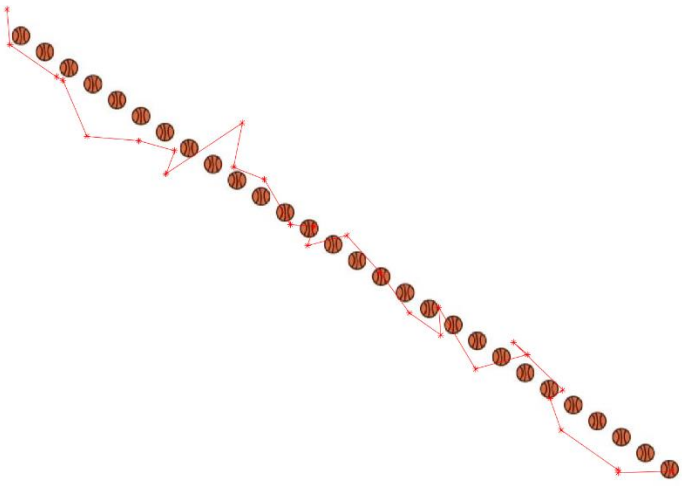
在本实验中假设其背景不变，将图像与其背景相减提取的前景即是目标物体，且将目标物体的中心设为其位置。



B. 加入高斯噪声

上面设计的追踪器得到的物体位置信息几乎没有误差，与观测位置与实际位置一致。因此，需要另外加入-30~30（单位：像素）的高斯噪声。

加入高斯噪声后的观测轨迹如下：



C. 卡尔曼滤波器

根据上一时刻的均方误差和最优估计结果以及从追踪器获取的观测量Z_k，基于卡尔曼滤波器的五个公式直接估计当前状态。

4. 卡尔曼滤波器的 MATLAB 编程实现

A. 定义矩阵

```
% 采样周期
dt = 1;
% 状态转移矩阵
A = [ 1  dt  0   0
      0  1   0   0
      0  0   1  dt
      0  0   0   1 ];
% 观测阵
H = [ 1  0  0  0
      0  0  1  0 ];
% 系统噪声的协方差矩阵
Q = 0.01*eye(4);
% 观测噪声的协方差矩阵
R = [r1 0
     0  r2];
% 初始状态量
X = [0, 0, 0, 0]';
% 初始均方误差
P = 100*eye(4);
```

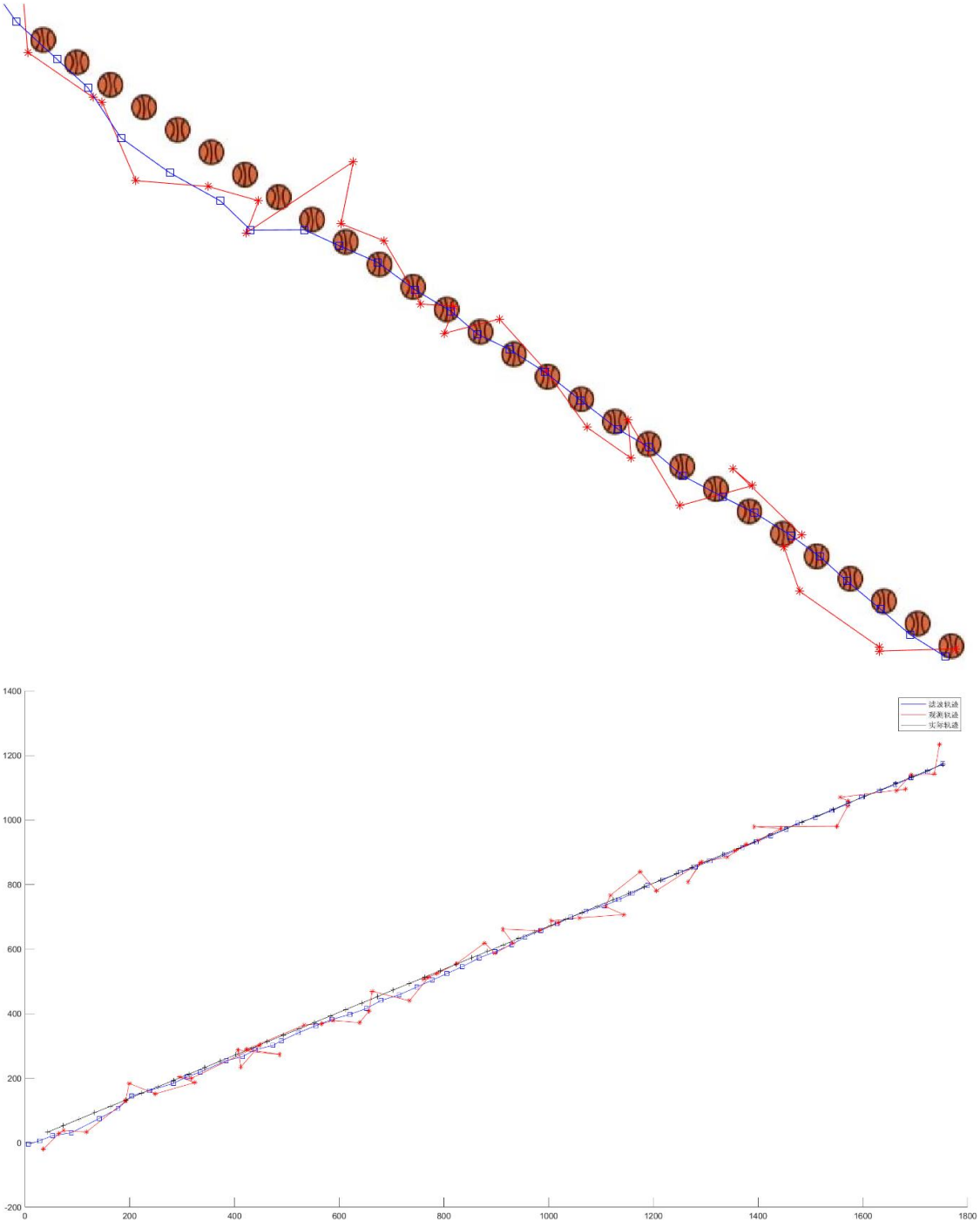
B. 求X_k的估计$\hat{X}_k$

```
% 状态一步预测
Xk_ = A*X;

% 一步预测均方误差
Pk_ = A*P*A' + Q;

% 滤波增益
K = Pk_*H'*inv(H*Pk_*H' + R);

% 观测值
z = [xm ym]';

% 状态估计
X = Xk_ + K*(z - H*Xk_);

% 估计均方误差
P = Pk_ - K*H*Pk_;
```

五、 实验结果与分析

1. 观测轨迹与滤波轨迹

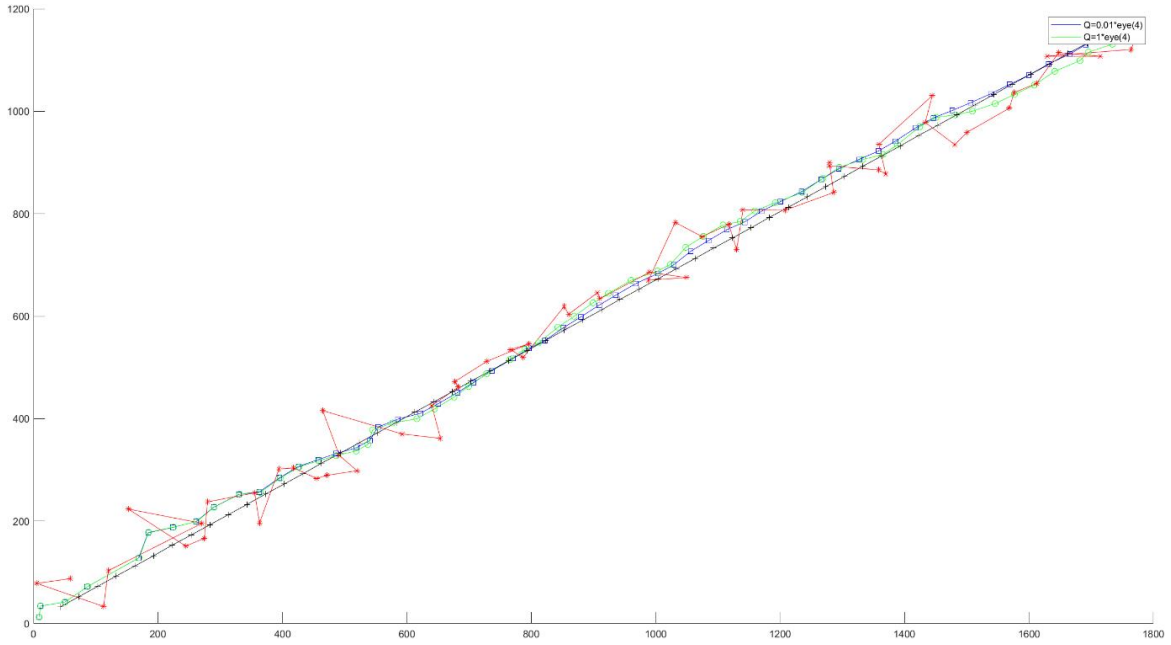
下图中红色点线图表示观测轨迹；蓝色点线图表示基于卡尔曼滤波器求得的滤波轨迹。小球是从图的左上方到右上方移动的，可发现一开始卡尔曼滤波器不能准确地估计物体的位置，但过一段时间后进入稳定状态，估计出来的位置与实际位置几乎一致。可见，装有卡尔曼滤波器的追踪器明显滤除了观测误差，准确估计出物体位置。



图二 观测轨迹与滤波轨迹

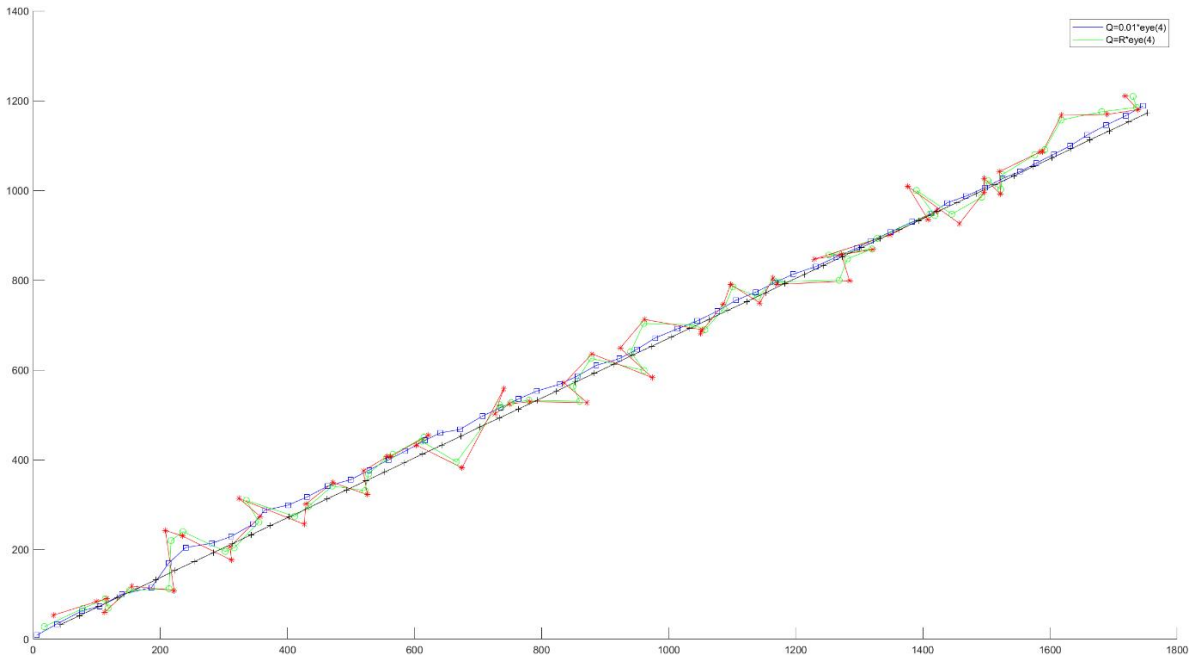
2. R 固定，Q 不同时滤波结果对比

图三表示 R 固定，系统噪声的协方差矩阵 Q 取 0.01 和 1 时的滤波结果，其中红线为观测轨迹，蓝线和绿线分别为 $Q = 0.01\,I$ 和 $Q = 1\,I$ 时的滤波轨迹。可发现 $Q = 0.01\,I$ 时的滤波结果更优，更接近于实际轨迹。



图三 Q = 0.01 与 Q = 1

图四表示 R 固定，Q 取 0.01（蓝线）和 Q 大小接近于 R（绿线）时的滤波结果。可以发现，Q 很大接近于 R 时几乎无法滤除观测误差，滤波轨迹接近于观测轨迹。



图四 Q = 0.01 与 Q ≈ R

可以总结出如下结果：

Q 越小，滤波结果更加接近由系统状态估计值

Q 越大，滤波结果更加接近于观测值。

六、讨论

1. Q 和 R 大小与滤波结果

根据求解卡尔曼滤波增益的公式：

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \quad (1)$$

可发现 R 值大小会直接影响 K 值：**R 越大，K 值越小；R 越小，K 值越大。**

$$P_k^- = AP_{k-1}A^T + Q \quad (2)$$

由式(1)和(2)可知：**Q 越大，P_k^-越大，K 值越小；Q 越小，P_k^-越小，K 值越大。**

根据求X_k的估计$\hat{X}_k$的公式：

$$\hat{X}_k = \hat{X}_k^- + K_k\left(Z_k - H\hat{X}_k^-\right) = (1 - K_k H)\hat{X}_k^- + K_k Z_k \quad (3)$$

可知 K 值大小会直接影响滤波结果：

K 值越大，$\hat{X}_k^-$的系数越小，Z_k的系数越大，滤波结果更加接近于观测值所反算出来的状态变量。

K 值越小，$\hat{X}_k^-$的系数越大，Z_k的系数越小，滤波结果更加接近于由系统状态估计值给出的递归结果。

这与实验结果一致：Q 越小，K 越大，滤波结果更加接近由系统状态估计值；Q 越大，K 越小，滤波结果更加接近于观测值。

七、代码

1. main.m

```
clear;
clear TrackKalmanQR;
clear TrackKalman;
clear GetBallPos;
close all;
global random;
global r1;
global r2;

NoOfImg = 58;
Xmsaved  = zeros(2, NoOfImg);
Xhsaved  = zeros(2, NoOfImg);
Xqrsaved = zeros(2, NoOfImg);
Xsaved = zeros(2,NoOfImg);
random = 25*randn(size(Xsaved));
r1 = cov(random(1,:));
r2 = cov(random(2,:));

for k = 1:NoOfImg
  [xm, ym,x,y] = Tracker(k);
  [xh, yh] = Kalman(xm, ym);
```

```
  figure(1);
  hold on
  plot(xm, ym, 'r*')
  plot(xh, yh, 'bs')

  pause(1)

  Xmsaved(:, k) = [xm ym]';
  Xhsaved(:, k) = [xh yh]';
  Xsaved(:,k) = [x y]';
end

figure
hold on
plot(Xhsaved(1,:), Xhsaved(2,:), 'b')
plot(Xmsaved(1,:), Xmsaved(2,:), 'r')
plot(Xsaved(1,:),Xsaved(2,:),'k')
plot(Xhsaved(1,:), Xhsaved(2,:), 'bs')
plot(Xmsaved(1,:), Xmsaved(2,:), 'r*')
plot(Xsaved(1,:),Xsaved(2,:),'k+')

legend('滤波轨迹','观测轨迹','实际轨迹')
```

2. Tracker.m

```
function [xh, yh] = Kalman(xm, ym)
persistent A H Q R
persistent X P
persistent firstRun
global r1;
global r2;

if isempty(firstRun)
      % 采样周期
      dt = 1;
      % 状态转移矩阵
      A = [ 1  dt  0   0
            0  1   0   0
            0  0   1  dt
            0  0   0   1 ];
      % 观测阵
      H = [ 1  0  0  0
            0  0  1  0 ];
      % 系统噪声的协方差矩阵
      Q = 0.01*eye(4);
      % 观测噪声的协方差矩阵
      R = [r1 0
           0  r2];
      % 初始状态量
      X = [0, 0, 0, 0]';
      % 初始均方误差
      P = 100*eye(4);
end
% 状态一步预测
Xk_ = A*X;

% 一步预测均方误差
```

```
Pk_ = A*P*A' + Q;

% 滤波增益
K = Pk_*H'*inv(H*Pk_*H' + R);

% 观测值
z = [xm ym]';

% 状态估计
X = Xk_ + K*(z - H*Xk_);

% 估计均方误差
P = Pk_ - K*H*Pk_;

xh = X(1);
yh = X(3);
```

3. Kalman.m

```
function [xc, yc, x, y] = Tracker(index)
persistent bg
persistent firstRun
global random

if isempty(firstRun)
  bg = imread('Img/bg.jpg');
  firstRun = 1;
end

img = imread(['Img/', int2str(index), '.jpg']);
imshow(img)

fg = imabsdiff(img, bg);
fg = (fg(:,:,1) > 10) | (fg(:,:,2) > 10) | (fg(:,:,3) > 10);

stats = regionprops(logical(fg), 'area', 'centroid');
area_vector = [stats.Area];
[~, idx] = max(area_vector);
centroid = stats(idx(1)).Centroid;

% 实际位置
x = centroid(1);
y = centroid(2);

% 观测位置（含高斯噪声）
xc = centroid(1) + random(1,index);
yc = centroid(2) + random(2,index);
```

4. CreateImg.m

```
clear;
close all;
global R
global C
global res;

ball = imread('ball.jpg');
```

```
[R,C,~] = size(ball);

init_res();
tmp = res;

imwrite(uint8(res),'Img28/bg.jpg','jpg');
% figure;
for i = 1:28
    move_img(ball,i*20,i*30);
%     imshow(uint8(res));
    imwrite(uint8(res),['Img28/',num2str(i),'.jpg'],'jpg');
    res = tmp;
end

function init_res()
global res;
global n;
n = 600;
res = zeros(n, 1.5*n, 3);
for i = 1:n
    for j = 1:1.5*n
        a = uint8(256);
        res(i,j,1:3)=[a,a,a];
    end
end
end

function move_img(A,x,y)
global res;
global R;
global C;
global n;
tras = [1 0 x; 0 1 y; 0 0 1];
for i = 1 : R
    for j = 1 : C
        temp = [i; j; 1];
        temp = tras * temp;
        p = temp(1, 1);
        q = temp(2, 1);
        if (p <= n) && (q <= 1.5*n) && (p >= 1) && (q >= 1)
            res(p,q,:) = A(i,j,:);
        end
    end
end

end
```