我竟然用OpenCV實現了卡爾曼濾波

3D視覺初學者 今天

以下文章來源於OpenCV學堂

OpenCV學堂

專注計算機視覺開發技術分享，技術框架使用，包括OpenCV，Tensorflow，Pytorch教程與案例，相關算法詳解，最新CV方向論文，硬核...

點擊上方

重磅乾貨，第一時間送達

本文轉自|OpenCV學堂

卡爾曼濾波原理

卡爾曼濾波最早可以追溯到Wiener濾波，不同的是卡爾曼採用狀態空間來描述它的濾波器，卡爾曼濾波器同時具有模糊/平滑與預測功能，特別是後者在視頻分析與對象跟踪應用場景中被發揚光大，在離散空間(圖像或者視頻幀)使用卡爾曼濾波器相對簡單。假設我們根據一個處理想知道一個變量值如下：

$$x_{k+1} = \Phi x_k + w_k$$

其中x_k是在k时刻的状态

Φ是从k到k+1时刻的变换矩阵

w_k是k时刻相关白噪声的协方差矩阵

对观测值建模如下：

$$z_k = Hx_k + v_k$$

其中z_k是在k时刻对x的实际测量值

H是状态矩阵与测量矩阵无噪声链接

v_k是测量错误

这样就得到两个协方差矩阵

$$Q = E\left[w_k w_k^T\right]$$

$$R = E\left[v_k v_k^T\right]$$

$$P_k = E\left[e_k e_k^T\right] = E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right]$$

假设$\hat{x}_k$前一个评估为$\hat{x}_k^{'}$，可以得到

$$\hat{x}_k = \hat{x}_k^{'} + K_k(z_k - H\hat{x}_k^{'})$$

其中K_k被称为卡尔曼增益，有了它就可以更新测量模型

从而更新状态空间的下个预测

中间进行一系列的等级变换，微分求解得到

$$K_k = P'_k H^T \left(H P'_k H^T + R\right)^{-1}$$

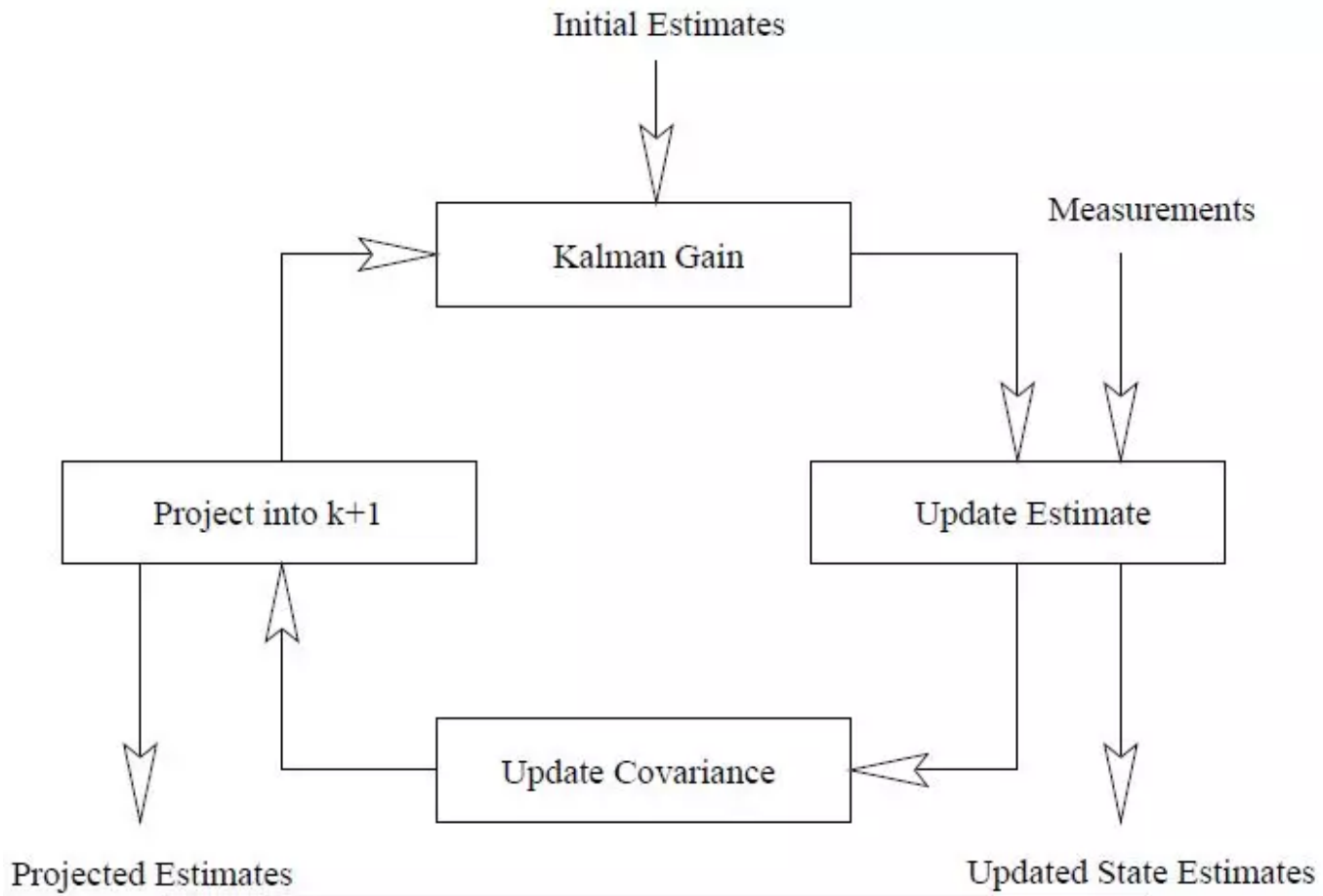
$$\begin{aligned} P_k &= P'_k - P'_k H^T \left(H P'_k H^T + R\right)^{-1} H P'_k \\ &= P'_k - K_k H P'_k \\ &= (I - K_k H) P'_k \end{aligned}$$

最终得到下一个空间状态：

$$P'_{k+1} = E\left[e'_{k+1} e'^T_{k+1}\right] = E\left[(\Phi e_k + w_k)(\Phi e_k + w_k)^T\right]$$

$$\begin{aligned} P'_{k+1} &= E\left[e'_{k+1} e'^T_{k+1}\right] \\ &= E\left[\Phi e_k (\Phi e_k)^T\right] + E\left[w_k w_k^T\right] \\ &= \Phi P_k \Phi^T + Q \end{aligned}$$

最終卡爾曼濾波完整的評估與空間預測模型工作流程如下：



Description	Equation
Kalman Gain	$K_k = P'_k H^T \left(HP'_k H^T + R\right)^{-1}$
Update Estimate	$\hat{x}_k = \hat{x}'_k + K_k\left(z_k - H\hat{x}'_k\right)$
Update Covariance	$P_k = \left(I - K_k H\right) P'_k$
Project into $k+1$	$\hat{x}'_{k+1} = \Phi\hat{x}_k$ $P_{k+1} = \Phi P_k \Phi^T + Q$

OpenCV API

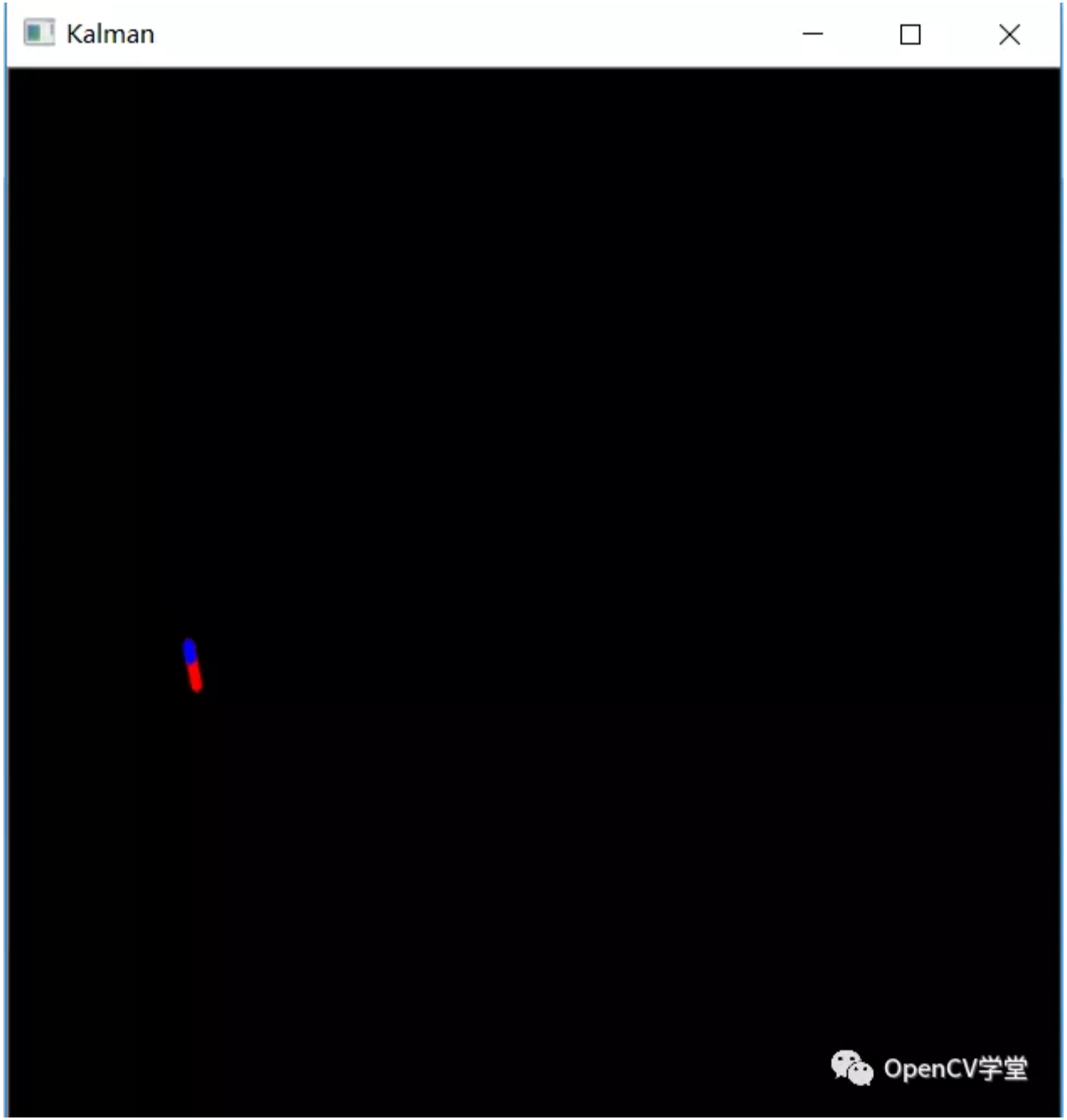
```
cv::KalmanFilter::KalmanFilter(
    int dynamParams,
    int measureParams,
    int controlParams = 0,
    int type = CV_32F
)
# dynamParams表示state的维度
# measureParams表示测量维度
# controlParams表示控制向量
# type表示创建的matrices
```

代碼演示

```
import cv2
from math import cos, sin, sqrt
import numpy as np

if __name__ == "__main__":

    img_height = 500
    img_width = 500
    kalman = cv2.KalmanFilter(2, 1, 0)

    cv2.namedWindow("Kalman", cv2.WINDOW_AUTOSIZE)

    while True:
        state = 0.1 * np.random.randn(2, 1)

        # 初始化
        kalman.transitionMatrix = np.array([[1., 1.], [0., 1.]])
        kalman.measurementMatrix = 1. * np.ones((1, 2))
        kalman.processNoiseCov = 1e-5 * np.eye(2)
        kalman.measurementNoiseCov = 1e-1 * np.ones((1, 1))
        kalman.errorCovPost = 1. * np.ones((2, 2))
        kalman.statePost = 0.1 * np.random.randn(2, 1)

        while True:
```

```
        def calc_point(angle):
            return (np.around(img_width/2 + img_width/3*cos(angle), 0).astype(int),
                    np.around(img_height/2 - img_width/3*sin(angle), 1).astype(int))

        state_angle = state[0, 0]
        state_pt = calc_point(state_angle)
        # 预测
        prediction = kalman.predict()
        predict_angle = prediction[0, 0]
        predict_pt = calc_point(predict_angle)

        measurement = kalman.measurementNoiseCov * np.random.randn(1, 1)

        # 生成测量
        measurement = np.dot(kalman.measurementMatrix, state) + measurement
        measurement_angle = measurement[0, 0]
        measurement_pt = calc_point(measurement_angle)

        # plot points
        def draw_cross(center, color, d):
            cv2.line(img,
                     (center[0] - d, center[1] - d), (center[0] + d, center[1] + d),
                     color, 1, cv2.LINE_AA, 0)
            cv2.line(img,
                     (center[0] + d, center[1] - d), (center[0] - d, center[1] + d),
                     color, 1, cv2.LINE_AA, 0)

        img = np.zeros((img_height, img_width, 3), np.uint8)

        cv2.line(img, state_pt, measurement_pt, (0, 0, 255), 3, cv2.LINE_AA, 0)
        cv2.line(img, state_pt, predict_pt, (255, 0, 0), 3, cv2.LINE_AA, 0)

        # 校正预测与测量值差异
        kalman.correct(measurement)

        # 更新noise矩阵与状态
        process_noise = sqrt(kalman.processNoiseCov[0,0]) * np.random.randn(2, 1)
        state = np.dot(kalman.transitionMatrix, state) + process_noise
```

```
            cv2.imshow("Kalman", img)

            code = cv2.waitKey(100)
            if code != -1:
                break

        if code in [27, ord('q'), ord('Q')]:
            break

    cv2.destroyWindow("Kalman")
```


Kalman
OpenCV学堂