Mouse Tracking Using Kalman Filter And OpenCV

Xiaohe He

hexh@shanghaitech.edu.cn
SIST, Shanghaitech Univsity, Shanghai, China

***Abstract*—In this project, we discussed an approach to track mouse path with the help of Kalman Filter and OpenCV. In the beginning, we discussed the Kalman Filter in detail. After that, the KalmanFilter module in OpenCV and the implementation in Python are also be covered. Anaconda was used to design and test the proposed method. This project found that if you make a sudden turn at high speed, the prediction line will have a wider trajectory, which is consistent with the momentum of the mouse movement at the time.**

***Index Terms*—Mouse tracking, Kalman Filter, OpenCV, Python, Prediction, Correction.**

I. Introduction

In the class EE251, we learned many approaches to estimate signals. Such as MVUE, BLUE, MLE, MMSE, LMMSE, Kalman Filter, etc. Kalman filter is one of the most common approaches used in varies fields like guidance, navigation, and control of vehicles, particularly aircraft and spacecraft, robotic motion planning and control[1]. Besides, recent decades, the use of computer vision in many applications has been significantly increased from manufacturing application into military applications [2-4]. A major aspect of computer vision applications is object tracking. Tracking objects in the real time environment is not a trivial task and has been a popular research topic in the computer vision field. [3,5,6]

This project focuses on tracing mouse path in using Kalman Filter and OpenCV. The goal of this project is to reviewing Kalman Filter and learning OpenCV. The specific contributions of this work are the combination of these two tools and the identification of the effect of this approach.

II. Kalman Filter

Kalman Filter is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe. This technique is published first by R.E. Kalman, in 1960 as a recursive solution for linear filtering in discrete data [7]. Kalman Filter is one of the most common approaches for estimating linear state that is assumed to have a Gaussian distribution [8]. Here are some operations of Kalman Filter:

1) *Some notations*[9]
    - $\hat{s}[n|n-1]$: state estimation at time n predicted based on the parameter at time n-1.
    - $\mathbf{M}[n|n-1]$: covariance of the estimated state at time n.
    - $\mathbf{K}[n]$: Kalman gain.
    - $\hat{s}[n|n]$: state estimation at time n updated based on the observation process or measurement at time n.
    - $\mathbf{M}[n|n]$: covariance of the estimated state at time n.
    - $\mathbf{A}$: the transition matrix.
    - $\mathbf{B}$: control matrix(not used if there is no control).
    - $\mathbf{Q}$: process noise matrix.
    - $\mathbf{H}$: measurement matrix.
    - $\mathbf{R}$: measurement noise covariance matrix.
2) *Basic Kalman Filter Operation*
   In general, the operation of discrete time Kalman filtering has the form that can be seen in Fig. 1. There are two main operations, the prediction step and correction step. The prediction step in this process is responsible for predicting the states to obtain a priori estimation of the next step based on the projection of the current state. Meanwhile, the correction step in this process will improve the a priori estimation resulting in the prediction step, based on obtained measurement results and obtains a posteriori estimation. For discrete time systems, these two operations are expressed using the notations as shows in Fig. 1.[10]

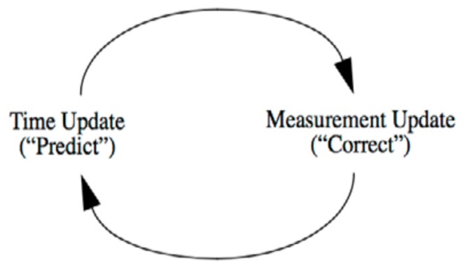

Fig. 1. Basic operation of the Kalman Filter

3) *Prediction Step*
   The "Predict" step can be mathematically expressed in the following equations. In this equations, the future state $\hat{s}[n|n-1]$ is predicted based on the process module $\mathbf{A}$ and the current state $\hat{s}[n-1|n-1]$. The prediction covariance $\mathbf{M}[n|n-1]$ is also calculated based on the process $\mathbf{A}$ and the uncertainty of the process model $\mathbf{Q}$. Equations are as following:

$$\hat{s}[n|n-1] = \mathbf{A}\hat{s}[n-1|n-1] \tag{1}$$

$$\mathbf{M}[n|n-1] = \mathbf{A}\mathbf{M}[n-1|n-1]\mathbf{A}^\top + \mathbf{B}\mathbf{Q}\mathbf{B}^\top \tag{2}$$

4) *Correction Step*
The next operation in the Kalman Filter is the "correct" step. In this step, the predicted state is corrected based on the difference between the real measured result and the expected measurement result from the measurement model **H**. Once the real measured result is obtained, we can calculate the difference between the real measurement and the measurement model. The Kalman gain, **K**[n], is then calculated to update the estimated state $\hat{s}[n|n-1]$ to become $\hat{s}[n|n]$. Equations are as following:

$$\mathbf{K}[n] = \mathbf{M}[n|n-1]\mathbf{H}^\top[n](\mathbf{C}+\mathbf{H}[n]\mathbf{M}[n|n-1]\mathbf{H}^\top[n])^{-1} \quad (3)$$

$$\hat{s}[n|n] = \hat{s}[n|n-1] + \mathbf{K}[n](\mathbf{x}[n] - \mathbf{h}(\hat{s}[n|n-1])) \quad (4)$$

$$\mathbf{M}[n|n] = (\mathbf{I} - \mathbf{K}[n]\mathbf{H}[n])\mathbf{M}[n|n-1] \quad (5)$$

5) *Some block diagrams of Kalman Filter*[11]
Scalar state-scalar observation Kalman Filter block diagram as shown in Fig.2. It is interesting to note that the dynamical model for the signal is an integral part of the estimator.

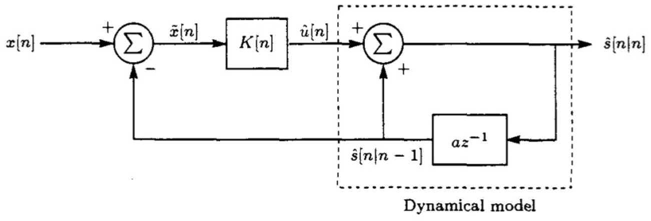

Fig. 2. Scalar state-scalar observation Kalman Filter

A first-order recursive filter with time varying coefficients as shown in Fig. 3.

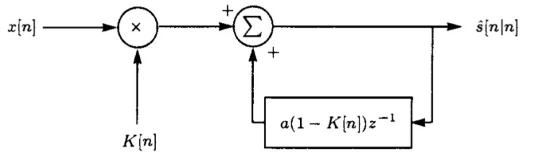

Fig. 3. Kalman Filter as time varying filter

The Kallman Filter is driven by the unrelated innovation sequence and in steady-state can also be viewed as a whitening filter. So that the input to the Kalman Filter is the innovation sequence $\tilde{s} = x[n] - \hat{s}[n|n-1]$ as shown in Fig. 4.

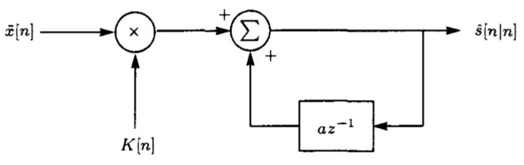

Fig. 4. Innovation-driven kalman Filter

We know from the discussion of the vector space approach that $\tilde{x}[n]$ is unrelated with $\{x[0], x[1], \ldots, x[n-1]\}$. Alternatively, if we view $\tilde{x}$ as the Kalman Filter *output* and if the filter attains steady-state, then it becomes a linear invariant whitening filter as shown in Fig. 5.

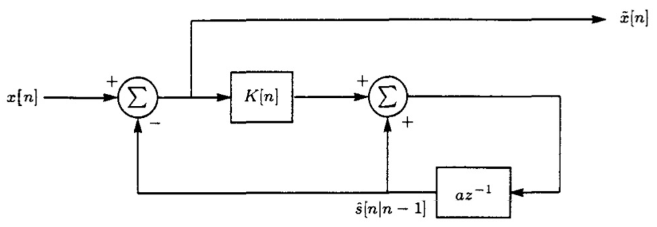

Fig. 5. Whitening filter interpretion of Kalman Filter

6) *Kalman Filter Algorithm:*[10]
Predict:

$$\hat{s}[n|n-1] = \mathbf{A}\hat{s}[n-1|n-1] \quad (1)$$

$$\mathbf{M}[n|n-1] = \mathbf{A}\mathbf{M}[n-1|n-1]\mathbf{A}^\top + \mathbf{B}\mathbf{Q}\mathbf{B}^\top \quad (2)$$

Update:

$$\mathbf{K}[n] = \mathbf{M}[n|n-1]\mathbf{H}^\top[n](\mathbf{C}+\mathbf{H}[n]\mathbf{M}[n|n-1]\mathbf{H}^\top[n])^{-1} \quad (3)$$

$$\hat{s}[n|n] = \hat{s}[n|n-1] + \mathbf{K}[n](\mathbf{x}[n] - \mathbf{h}(\hat{s}[n|n-1])) \quad (4)$$

$$\mathbf{M}[n|n] = (\mathbf{I} - \mathbf{K}[n]\mathbf{H}[n])\mathbf{M}[n|n-1] \quad (5)$$

III. OpenCV's KalmanFilter module

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision.[12] Originally developed by Intel, it was later supported by Willow Garage and is now maintained by Itseez.[13] The library is cross-platform and free for use under the open-source BSD license. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics[14].

"KalmanFilter" is the module we used in this project. In this module, there are 3 interfaces in the module KalmanFilter[15]:

1) init: initialize the dimension of Kalman Filter;
2) predict: predict the result of estimation;
3) correct: input the measured result, get the optimal estimation.

Besides, there are 10 complex parameters, which go hand in hand with the principle of Kalman Filter:

Basic parameters matrices:

1) Mat transitionMatrix: state transition matrix **A**
2) Mat controlMatrix: control matrix **B** (not used if there is no control)

3) Mat measurementMatrix: measurement matrix **H**
4) Mat processNoiseCov: process noise covariance matrix **Q**
5) Mat measurementNoiseCov: measurement noise covariance matrix **R**

Kalman Filter formulas matrices:

1) Mat statePre: predicted state $\hat{s}[n|n-1]$
2) Mat statePost: corrected state $\hat{s}$
3) Mat errorCovPre: priori error estimate covariance matrix $\mathbf{M}[n|n-1]$:
4) Mat gain: Kalman gain matrix **K**[n]:
5) Mat errorCovPost: posteriori error estimate covariance matrix $\mathbf{M}[n|n]$:

Before using this module, we need to initial basic parameters matrices, especially $transitionMatrix$, $processNoiseCov$, $measurementNoiseCov$.

$measurementMatrix$ is 1 in common, $controlMatrix$ should be 0 if it does not exists. $errorCovPost$ is better to be initialed as 1 than 0. $statePost$ should be initialed nearing to the next state.

IV. Kalman Filter for Mouse Tracking

In object tracking implementation, the Kalman filter predicts the object's next position from the previous state information about the object. It then verifies the result of the prediction using the result of the object detection process in the next following steps[9].

We model perturbations as noise inputs, so that the velocity components in the x and y directions at time n are

$$v_x[n] = v_x[n-1] + u_x[n] \tag{6}$$

$$v_y[n] = v_y[n-1] + u_y[n] \tag{7}$$

Without the noise perturbations $u_x[n]$, $u_y[n]$ the mouse would be constant, and hence the mouse would be modeled as moving in a straight line. From the equations of motion the position at time n can be expressed as:

$$r_x[n] = r_x[n-1] + v_x[n]\Delta \tag{9}$$

$$r_y[n] = r_y[n-1] + v_y[n]\Delta \tag{10}$$

where Δ is the time interval between samples. In this discretized model of the equations of motion the mouse is modeled as moving at the velocity of the previous time instant and then changing abruptly at the next time instant, an approximation to the true continuous behavior. We choose the signal vector as consisting of the position and velocity components or

$$s[n] = \begin{bmatrix} r_x[n] \\ r_y[n] \\ v_x[n] \\ v_y[n] \end{bmatrix} \tag{11}$$

Then we have that

$$\begin{bmatrix} r_x[n] \\ r_y[n] \\ v_x[n] \\ v_y[n] \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_x[n-1] \\ r_y[n-1] \\ v_x[n-1] \\ v_y[n-1] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ u_x[n] \\ u_y[n] \end{bmatrix} \tag{12}$$

The measurements are noisy observation of the range and bearing

$$R[n] = \sqrt{r_x^2[n] + r_y^2[n]} \tag{13}$$

$$\beta[n] = arctan\frac{r_y[n]}{r_x[n]} \tag{14}$$

And from Kay's book[16] we could know that:

$$\mathbf{C} = \begin{bmatrix} \delta_R^2 & 0 \\ 0 & \delta_\beta^2 \end{bmatrix} \tag{15}$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

$$\mathbf{h(s[n])} = \begin{bmatrix} \sqrt{r_x^2[n] + r_y^2[n]} \\ arctan\frac{r_y[n]}{r_x[n]} \end{bmatrix} \tag{17}$$

$$\mathbf{BQB}^\top = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \delta_u^2 & 0 \\ 0 & 0 & 0 & \delta_u^2 \end{bmatrix} \tag{18}$$

$$\mathbf{H}[n] = \left.\begin{bmatrix} \frac{r_x[n]}{\sqrt{r_x^2[n]+r_y^2[n]}} & \frac{r_y[n]}{\sqrt{r_x^2[n]+r_y^2[n]}} & 0 & 0 \\ \frac{-r_y[n]}{r_x^2[n]+r_y^2[n]} & \frac{r_x[n]}{r_x^2[n]+r_y^2[n]} & 0 & 0 \end{bmatrix}\right|_{s[n]=\hat{s}[n|n-1]} \tag{19}$$

Then by the Kaman Filter Algrithm(Eq.(1)-(5)), we could tracking mouse path.

V. implementation in Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code,[17][18] notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.[19]

Besides, Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.[20]

Besides, it is easy to use. That's also why I chose python as the language to implement this project.

To implement this project, here are some steps in the code of this project[21]:

1) Import packages cv2, numpy, pyplot, etc..
2) After the packages import, we create an empty frame, of size 800 x 800, and then initialize the arrays that will take the coordinates of the measurements and predictions of the mouse movements.
3) Then, we declare the mouse move Callback function, which is going to handle the drawing of the tracking. The

mechanism is quite simple; we store the last measurements and last prediction, correct the Kalman with the current measurement, calculate the Kalman prediction, and finally draw two lines, from the last measurement to the current and from the last prediction to the current

4) The next step is to initialize the window and set the Callback function. OpenCV handles mouse events with the setMouseCallback function; specific events must be handled using the first parameter of the Callback (event) function that determines what kind of event has been triggered (click, move, and so on)
5) Now we're call the "KalmanFilter" module in OpenCV and initial the parameters.
   - $dynamParams$: This parameter states the dimensionality of the state
   - $MeasureParams$: This parameter states the dimensionality of the measurement
   - $ControlParams$: This parameter states the dimensionality of the control
   - $vector.type$: This parameter states the type of the created matrices that should be CV_32F or CV_64F
6) From this point on, the program is straightforward; every mouse movement triggers a Kalman prediction, both the actual position of the mouse and the Kalman prediction are drawn in the frame, which is continuously displayed. A sample is shown as in Fig. 6. (green line is the measured path the mouse run, while, the red one is predicted by the Kalman Filter)

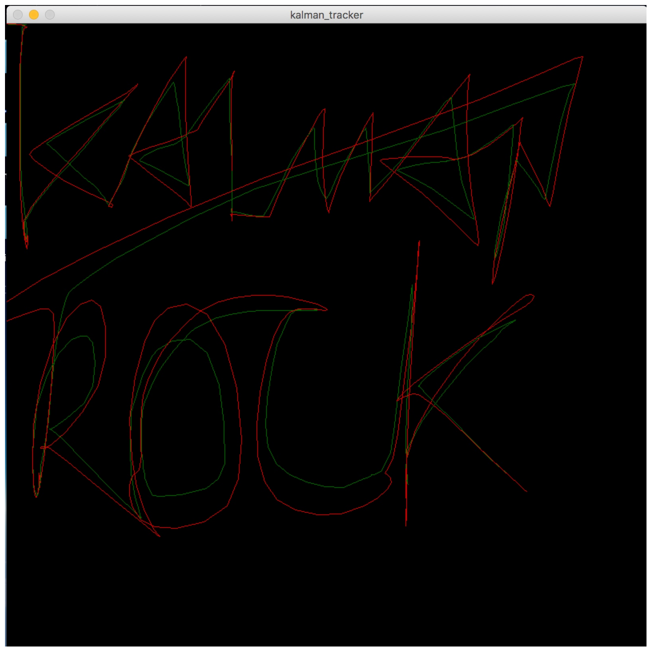

Fig. 6. Kalman tracker

VI. Conclusion

In this project I found that when moving the mouse around, if you make a sudden turn at high speed, the prediction line will have a wider trajectory, which is consistent with the momentum of the mouse movement at the time. while, if you move the mouse slowly and less sudden turning at high speed, the prediction line will have a good matching with the mouse path.

So, Kalman Filter has better performance in less sudden turning scene. If there are many sudden turnings at high speed , it will need more time to rightly predict the path.

Acknowledgment

First and foremost, I would like to show my deepest gratitude to Prof. Luo, a respectable, responsible and resourceful scholar, who has provided us with brilliant lectures in the class EE251. In these lectures, I learned much about signals estimation and detection.

I shall extend my thanks to our T.A. for all his kindness and answering our questions, which helped us a lot in learning.

Last but not least, I' d like to thank all my friends for their encouragement and support.

All of you ROCK! And have fun in vacation!

References

[1] Paul Zarchan; Howard Musoff (2000). "Fundamentals of Kalman Filtering: A Practical Approach". *American Institute of Aeronautics and Astronautics, Incorporated.* ISBN 978-1-56347-455-2.
[2] Y. S. Murijanto et al., "Machine Vision Implementation in Rapid PCB Prototyping," *Journal of Mechatronics, Electrical Power, and Vehicular Technology*, vol. 02, no. 2, pp. 79-84, 2014.
[3] M. Mirdanies et al., "Object Recognition System in Remote Controlled Weapon Station using Sift and Surf Methods," *Journal of Mechatronics, Electrical Power, and Vehicular Technology*, pp. 99-108, 2013.
[4] E. S. Maarif et al., "A Trajectory Generation Method Based on Edge Detection for Auto-Sealent Cartesian Robot," *Journal of Mechatronics, Electrical Power, and Vehicular Technology* , vol. 05, no. 1, pp. 27-36, 2014.
[5] Y. Hongpeng et al., "A Robust Object Tracking Algorithm Based on Surf and Kalman Filter," *Intelligent Automation & Soft Computing*, vol. 19, no. 4, pp. 567-579, 2013.
[6] M. M. Khan et al., "Tracking Occluded Objects Using Kalman Filter and Color Information," *International Journal of Computer Theory and Engineering*, Vol. 6, No. 5, , vol. 6, no. 5, pp. 438-442, 2014.
[7] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, p. 35 45, 1960.
[8] H. A. Patel and D. G. Thakore, "Moving Object Tracking Using Kalman Filter",*International Journal of Computer Science and Mobile Computing*, vol. 2, no. 4, p. 326 332,2013.
[9] Steven M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, p. 448-449, 1993
[10] Liana Ellen Taylor, Midriem Mirdanies and Roni Permana Saputra, "Optimized object tracking technique using Kalman filter", *Journal of Mechatronics, Electrical Power, and Vehicular Technology*, vol. 07, p. 57-66, 2016.
[11] Steven M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, p. 437-442, 1993
[12] Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV". Queue. pp. 40:4040:56.
[13] "Itseez leads the development of the renowned computer vision library OpenCV". http://itseez.com
[14] OpenCV User Site: http://opencv.org/
[15] "opencvKalmanFilter" http://www.xuebuyuan.com/1655511.html
[16] Steven M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, p. 456-463, 1993
[17] Summerfield, Mark. "Rapid GUI Programming with Python and Qt. Python is a very expressive language, which means that we can usually write far fewer lines of Python code than would be required for an equivalent application written in, say, C++ or Java"

[18] McConnell, Steve (30 November 2009). *Code Complete*, p. 100. ISBN 9780735636972.

[19] Kuhlman, Dave. ”A Python Book: Beginning Python, Advanced Python, and Python Exercises”.

[20] ”About Python”. Python Software Foundation. Retrieved 24 April 2012., second section ”Fans of Python use the phrase ”batteries included” to describe the standard library, which covers everything from asynchronous processing to zip files.”

[21] Joe Minichino and Joseph Howse, “Learning OpenCV 3 Computer Vision with Python“, p. 151-152.