Rodolfo Finocchi
Seminar Presentation: Critical Review
4/26/2016

The paper entitled "1€ Filter: A Simple Speed-based Low-pass Filter for Noisy Input in Interactive Systems" presents a simple algorithm to filter noisy signals for high precision and user responsiveness[i]. Noisy signals are problematic in interactive systems because they affect the accuracy and precision of the measured information, leading to undesirable and unpredictable behaviors. Noise can be caused by heat and magnetic fields, limits of sensor resolution, and sometimes unstable numerical computation.

When noisy signals are used to control interactive systems, they introduce jitter, where many different values are observed for a single true one. Low-pass filters are tools that help to reduce or remove this unwanted noisy part of a signal. Filters can be implemented as hardware or software, depending on the application. An inherent problem with filtering, however, is the introduction of time latency, or lag, which reduces the system's responsiveness. In cases where interactive feedback is critical to system behavior, the reduction of system responsiveness brought on by lag can make performing the desired tasks difficult and unintuitive.

In our project, we are implementing admittance control of the UR5 for use in Synthetic Tracked Aperture Ultrasound (STRATUS) imaging. Our system uses forces measured by the Robotiq 150 force-torque sensor attached to the robot end effector as input to define desired end effector velocities:

$$\dot{x}_{des} = Kf \quad (1)$$

where $f \in \mathbb{R}^{6x1}$is the measured forces and torques and K is a diagonal matrix of nonlinear gains. These velocities are then converted into robot joint velocities, subjected to certain geometric constraints referred to as virtual fixtures, and sent to the robot. Because the forces measured play such a critical part of the robot control, noise in the signals can greatly affect the behavior of the robot. In fact, the blue curve in Figure 1 shows the velocity commands in the x direction of the robot base coordinates calculated directly by (1). The orange curve is the result of filtering these raw commands with the 1€ filter, as described in detail below. It is clear from the jitter shown on the blue curve the extent to which noise affects the output. Since the STRATUS system must be able to accurately track the robot end effector for accurate image reconstruction and since the user needs to be able to expect robot behavior, as much noise as possible should be removed from the signal. Simultaneously, for safety and intuitiveness, the presence of lag should be minimal.

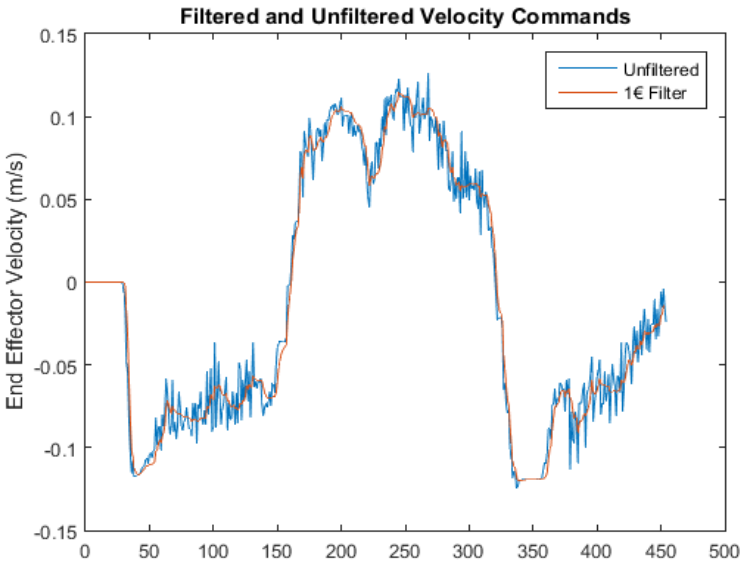

Figure 1 Robot velocity commands without (blue) and with (orange) 1€ filtering

This paper presents a description of various types of filters before describing how the 1€ filter works. The results of an experiment that compare the effectiveness of the 1€ filter to that of others are also presented. According to the authors, jitter should be less than 1 mm mean-to-peak, and lag should be below 60 ms. The first type of filter the paper describes is a moving average, in which *n* previous data values are used to produce a better estimate of the current one. While this is one of the most commonly used filters, it introduces a significant amount of lag. Namely, since all *n* values are weighted equally, the magnitude of lag amounts to *n* times the sampling period. In the STRATUS imaging case, collecting ultrasound (US) images in real time greatly reduces the sampling rate. Therefore, a moving average filter would require that the value of *n* be changed whenever the system begins and stops collecting images, an extremely tedious task.

The second type of filter described is this paper is an exponential smoothing filter. The discrete-time realization of such a filter is described by:

$$\hat{X}_i = \alpha X_i + (1-\alpha)\hat{X}_{i\text{-}1} \quad (2)$$

where $\hat{X}_i$ and X_i are the raw and filtered data, respectively, at time i and α is a smoothing factor. The first term of (2), weighted by α, denotes the contribution of new input data values, while the second term, weighted by $(1-\alpha)$, adds inertia from previous values. Jitter decreases with the value of α, while lag behaves inversely. Since the contribution of older values decreases exponentially, the lag of these filters is lower than that of high *n* moving average filters.

Double exponential smoothing filters extend (2):

$$\hat{X}_i^{[2]} = \alpha\hat{X}_i + (1-\alpha)\hat{X}^{[2]}{}_{i\text{-}1} \quad (3)$$

$$P_{t+\tau} = \left(2 + \frac{\alpha\tau}{1-\alpha}\right)\hat{X}_i - \left(1 + \frac{\alpha\tau}{1-\alpha}\right)\hat{X}_i^{[2]} \quad (4)$$

This filter type was presented by LaViola[ii] for predictive tracking with the purpose of predicting

positions τ time steps in the future. This method seems to be better at maintaining both lag and jitter low, but introduce overshoot, particularly when the measurements abruptly change direction or magnitude.

Kalman filters make assumptions about the system generating the signal. They are typically used for navigation and tracking and work well when combining data from different sensors. They rely on a process model and a measurement model. A Kalman filter requires initial guesses, $\hat{x}_0$, Σ_0 where $\hat{x}_0$is the predicted state guess and Σ_0 is the predicted system covariance. An iterative prediction-correction algorithm then follows with:

$$\hat{x}_t^{'} = A\hat{x}_{t\text{-}1} + Bu_{t\text{-}1} \quad (5)$$

$$\Sigma'_t = A\Sigma_{t\text{-}1}A^T + Q \quad (6)$$

for the prediction steps and:

$$K_t = \Sigma_t^{'}H^T\left(H\Sigma_t^{'}H^T + R\right)^{-1} \quad (7)$$

$$\hat{x}_t = \hat{x}_t^{'} + K_t\left(z_t\text{-}H\hat{x}_t^{'}\right) \quad (8)$$

$$\Sigma_t = (1 - K_tH)\Sigma_t^{'} \quad (9)$$

used for correction. In (5) $\hat{x}_t^{'}$ is the predicted value at time t, A is the state transition matrix that defines the planned transition from one state to another, $\hat{x}_{t\text{-}1}$is the corrected value of the previous time step, B is the input command transition matrix, and $u_{t\text{-}1}$ is the external command input of the previous time step. In (6), Σ'_t is the predicted covariance matrix, $\Sigma_{t\text{-}1}$is the corrected covariance matrix of the previous time step and Q is the covariance in the Gaussian noise. In (7), K_t is the matrix that minimizes the posterior correction, H is the linear measurement model that maps between system state and observations and R is the covariance of the sensors, obtained from the manufacturer. In (8), $\hat{x}_t$ is the correction of the system state, and z_t is the sensor measurement. Lastly, in (9), Σ_t is the corrected covariance of the measured error. Kalman filters can be difficult to understand and require knowledge of advanced mathematics. In some cases, the process and measurement noise covariances must be determined empirically. If this is done improperly, the filter can actually degrade the signal, by creating artificial "overshooting" movements, for example. They also require a language or library with matrix operation and are considerably slower (about 135 times slower) than double exponential smoothing predictors, with similar jitter and lag performance[ii].

The 1€ filter aims to address the weaknesses of all other presented filters. It is an adaptive first-order low-pas filter, meaning its cutoff frequency varies for each new sample according to an estimate of the signal's speed, or its derivative value. The reason behind this is that humans are more sensitive to jitter at low speeds, and more sensitive to lag at high speeds. Therefore, as the rate of change in the signal increases, meaning that the system is being used at a higher speed, the cutoff frequency also increases to decrease lag. Alternately, lower rates of change in the signal causes the cutoff frequency to approach its minimum value, decreasing jitter. The 1€ filter is formulated as:

$$\alpha = \frac{1}{1+\frac{\tau}{T_e}} \quad (10)$$

where α has the same functionality as in (2) but a different formulation, T_e is the sampling period and

$$\tau = \frac{1}{2\pi f_C} \quad (11)$$

$$\hat{X}_i = \left(X_i + \frac{\tau}{T_e}\hat{X}_{i\text{-}1}\right)\frac{1}{1+\frac{\tau}{T_e}} = \alpha X_i + (1-\alpha)\hat{X}_{i\text{-}1} \quad (12)$$

$$f_c = f_{c_{min}} + \beta \left|\dot{\hat{X}}_i\right| \quad (13)$$

where f_c is the cutoff frequency, $\dot{\hat{X}}_i$ is the derivative of the signal, and $f_{c_{min}}$ and β are the intercept and slope, respectively, of f_c. These two last values are the only two configurable parameters making the 1€ filter very easy to tune. To do so, the paper recommends setting β to 0 and $f_{c_{min}}$ to a value like 1 Hz. Then the system is moved slowly while $f_{c_{min}}$ is adjusted to remove jitter. After that, the system can be moved at faster speeds while β is increased in order to minimize lag.

To compare the 1€ filter with other techniques, the authors created a Python application that periodically samples the XY position of their system cursor and adds Gaussian noise with 50 dB signal-to-noise ratio (SNR) and displayed the filtered cursor positions. The moving average filter was tuned first and its performance was used as a baseline. They found that averaging more than 14 data values only increased increased lag, so they used that value for *n*. Every other filter was

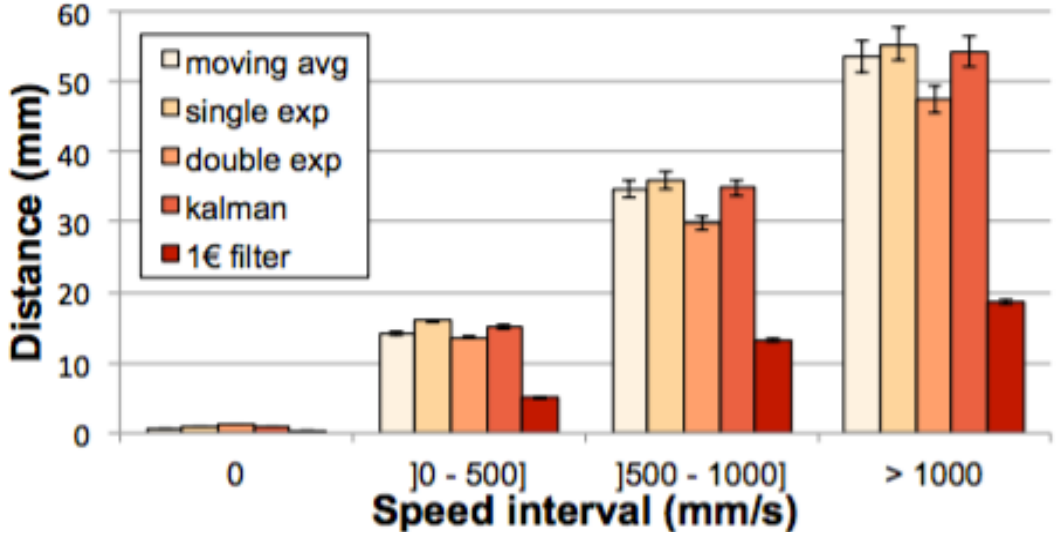

Figure 2 Mean distance between filtered and true cursor position for each speed interval and filter. Error bars represent 95% CI.

tuned as to achieve the same jitter and optimize lag if possible. For one hour, they logged the system cursor at 60 Hz during regular desktop use, added white noise and applied the filters. The distance between each filtered cursor position to the true one, binned into four speed intervals, were recorded. The results are shown in Figure 2. As seen, the 1€ filter had the lowest error across all speed intervals. Additionally, the authors computed the Standard Error of the Mean (SEM) in

mm for each filter. The 1€ filter had the smallest value (0.004), followed by LaViola's double exponential smoothing (0.013), the moving average and Kalman filters (0.015), and finally the single exponential smoothing filter (0.016). Therefore, of the filters discussed, the 1€ leads to the best performance in interactive systems with noisy input. In our use of the 1€ filter, the behavior of the robot greatly improves. As previously mentioned, the orange curve Figure 1 above shows the result of passing the raw velocity commands through the 1€ filter. The jitter is virtually unnoticeable.

Overall, this paper was very easy to understand and the algorithm was easy to implement. The comparisons between the different filter types made the ways in which the 1€ is better than existing methods clear. Something that was also very useful was the authors' website, which gives clear instructions on how to tune the filter, has an interactive demo for filter comparison and sample code in a number of languages. The paper could have benefited from more images, more quantitative analyses of the benefits of the 1€ and a mention of additional possible implementations of the filter. In the end, this paper proved to be very useful in helping us reduce both jitter and lag in control of the UR5 for the STRATUS system.

[i] Casiez, Géry, Nicolas Roussel, and Daniel Vogel. "1€ Filter: A Simple Speed-based Low-pass Filter for Noisy Input in Interactive Systems." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012.

[ii] LaViola, J. J. Double exponential smoothing: an alternative to kalman filter-based predictive tracking. In Proc. of EGVE ’03, ACM (2003), 199–206.