1€ Filter: A Simple Speed-based Low-pass Filter for Noisy Input in Interactive Systems

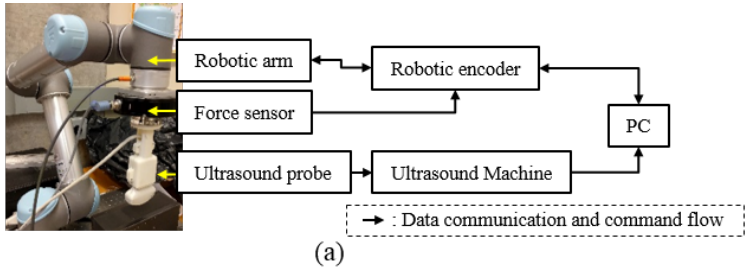
Géry Casiez, Nicolas Roussel, Daniel Vogel

Rodolfo Finocchi
Seminar Presentation
Group 2: Synthetic Tracked Aperture Ultrasound Imaging

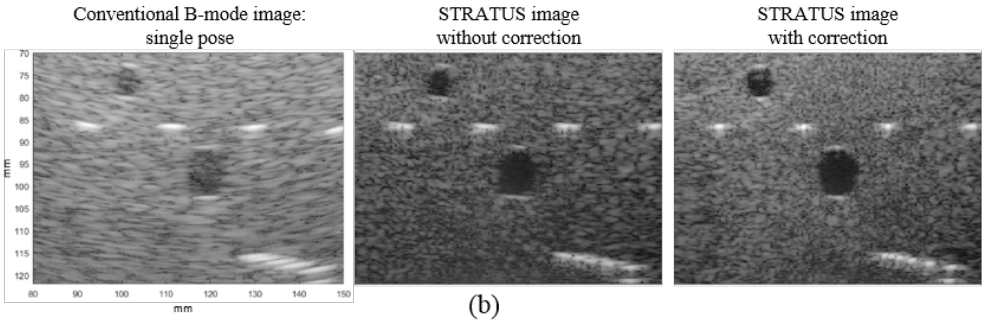
MUSiiC
Research Laboratory

Project Overview

- Aperture size of the ultrasound transducer limits image quality
- Synthetic tracked aperture imaging shows improvement
- Goal: bring system from autopilot to co-robotic freehand using virtual fixtures and force control



(a)



(b)

Admittance Control

$$\dot{x}_{des} = Kf$$

$$\dot{q}_{des} = \min_{\dot{q}} \|\dot{x}_{des} - J\dot{q}_{des}\|$$

- Probe must be tracked accurately
- Robot movements must be predictable

Paper Selection

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.

- Relevance to problem of interactive co-robotic control
- Useful comparison of different types of filters
- Ease of integration with existing admittance control framework

Noise and Jitter

Noise reduces the accuracy and precision of a signal:

introduces jitter, leading to many different observed values for a singe true one.

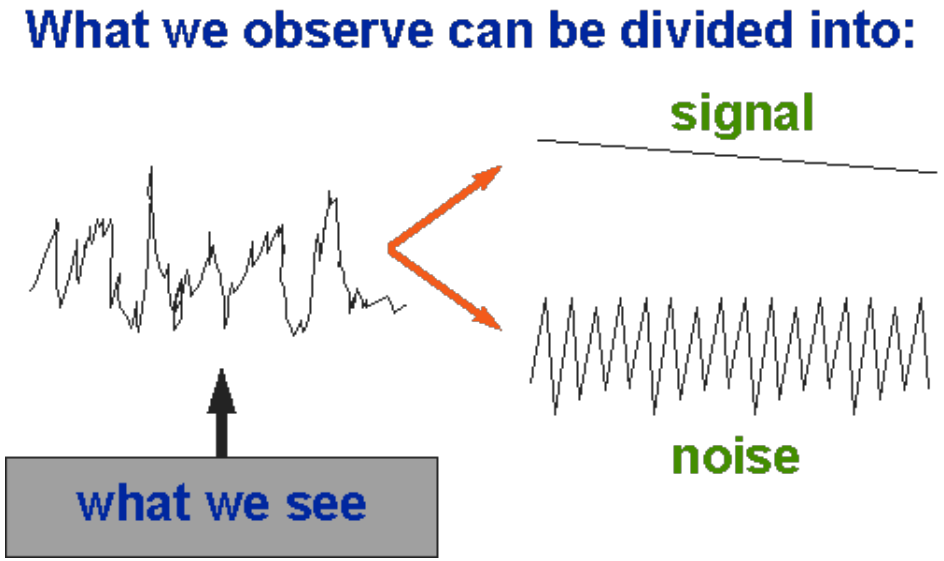
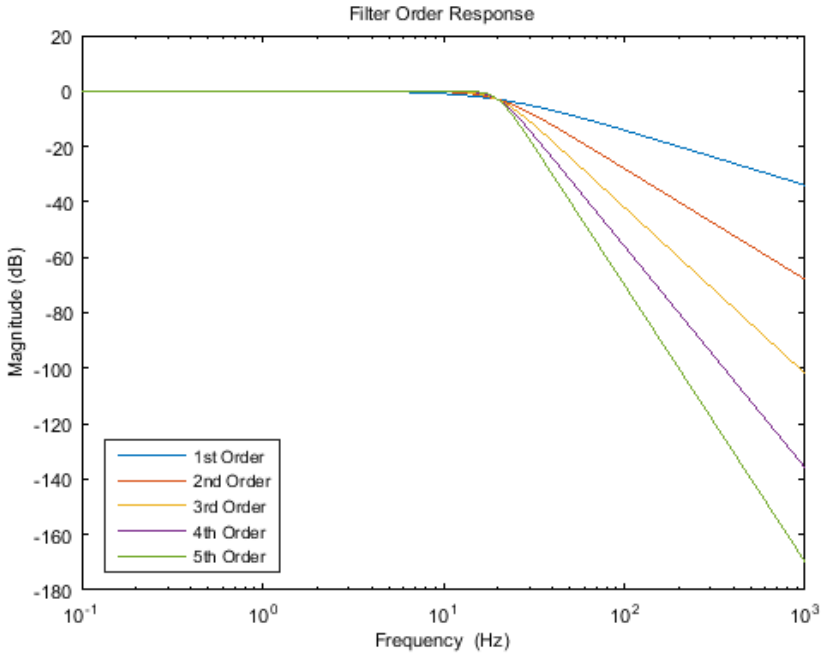


Image source: http://www.socialresearchmethods.net/kb/expclass.php

Filtering and Lag

Filtering noisy signals reduces, and possibly removes, the unwanted parts of the signal.

However, it introduces time latency, or lag, which reduces system responsiveness



Filtering and Lag

Filtering noisy signals reduces, and possibly removes, the unwanted parts of the signal.

However, it introduces time latency, or lag, which reduces system responsiveness

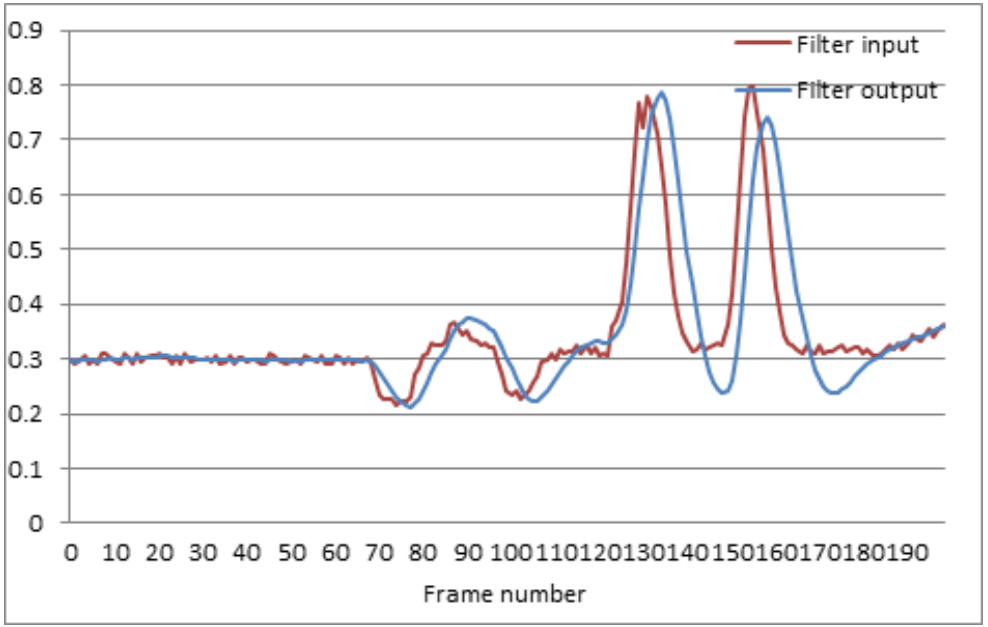


Image source: https://i-msdn.sec.s-msft.com/dynimg/IC584344.png

Filter Comparisons: Moving Average Filter

By the Central Limit Theorem, averaging enough successive values of a noisy signal should produce a better estimate of the true one. Moving average low-pass filter: average of last *n* data values is used to estimate current values.

Problem: creates lag of *n* times the sampling period

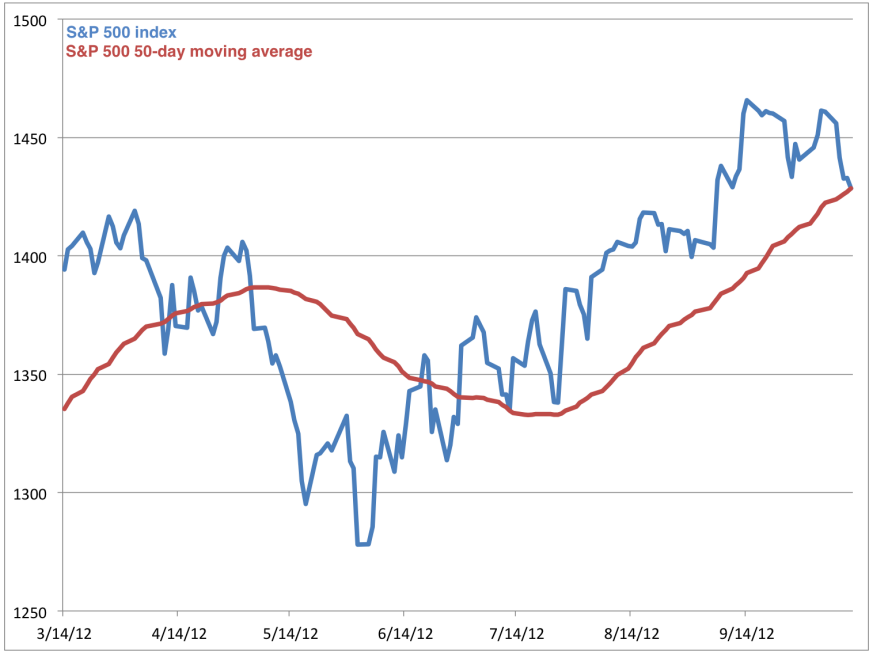


Image source: http://static4.businessinsider.com/image/507862e2ecad04db7d000016-1502-1127/sp-500-50-day-ma.png?maxX=600

Filter Comparisons: Exponential Smoothing

- $\hat{X}_i = \alpha X_i + (1 - \alpha)\hat{X}_{i-1}$
  - First term: contribution of new input data
  - Second term: adds inertia from previous values
- As α decreases, jitter is reduced, but lag increases

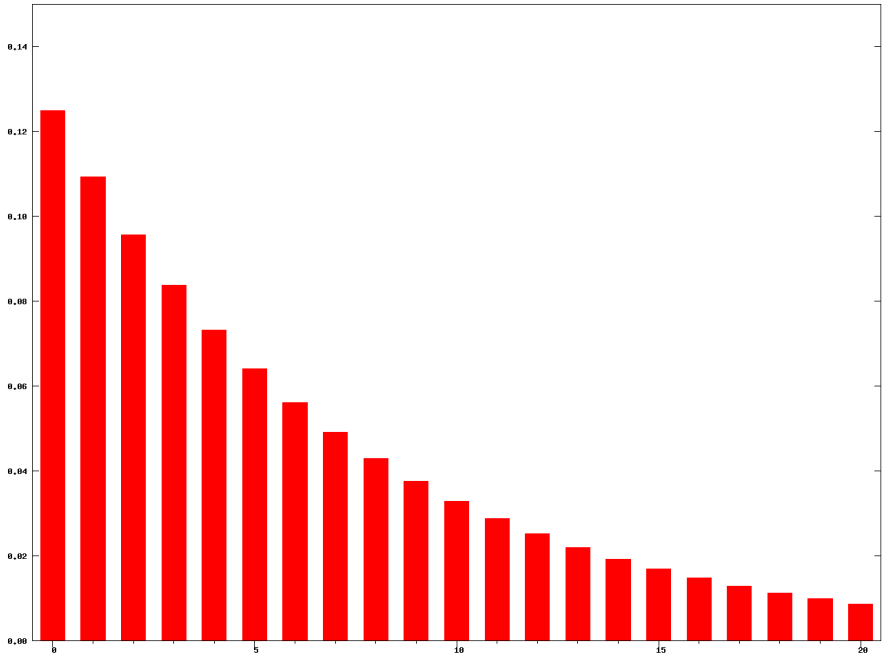


Image source:
https://upload.wikimedia.org/wikipedia/commons/b/ba/Exponential_moving_average_weights_N%3D15.png

Filter Comparisons: Double Exponential Smoothing

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

- Used by LaViola* to predict positions τ steps in the future
- Good Jitter-Lag balance but introduces overshoot

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

Filter Comparisons: Kalman Filter

Commonly used in robotics for navigation and tracking: rely on a process model and measurement model

1) Start with an initial guess:

$$\hat{x}_0, \Sigma_0$$

2) Compute prior (prediction):

$$\hat{x}_t' = A\hat{x}_{t-1} + Bu_{t-1}$$
$$\Sigma'_t = A\Sigma_{t-1}A^T + Q$$

3) Compute posterior (correction):

$$K_t = \Sigma_t' H^T (H\Sigma_t' H^T + R)^{-1}$$
$$\hat{x}_t = \hat{x}_t' + K_t(z_t - H\hat{x}_t')$$
$$\Sigma_t = (1 - K_t H)\Sigma_t'$$

4) Repeat steps 2 and 3

- An improperly tuned filter can increase and even degrade the signal.
- Considerably slower than other methods (~135 times)
- Complicated

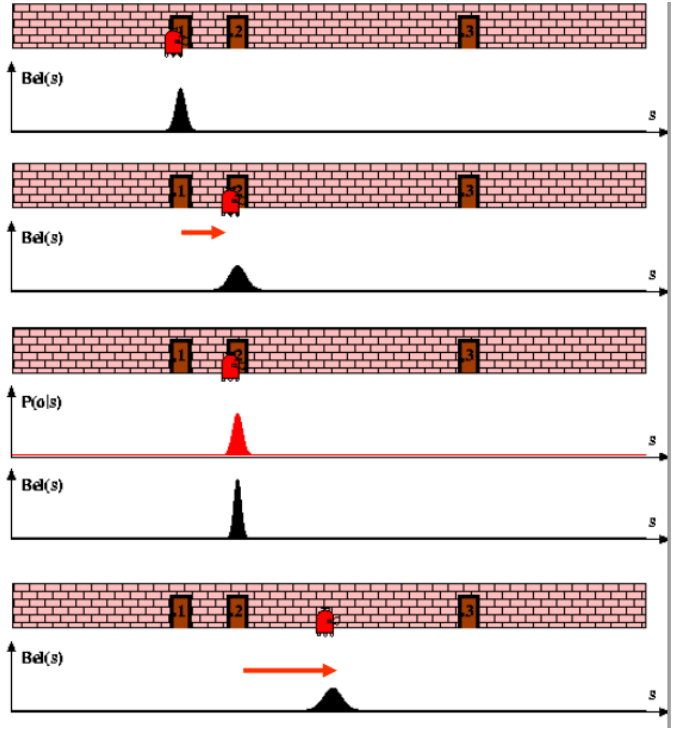


Image from: GD Hager slides

The 1€ Filter

Adaptive 1st-order low-pass filter: cutoff frequency adapts according to an estimate of the signal's speed, or derivative

$$\alpha = \frac{1}{1 + \frac{\tau}{T_e}}$$

$$\tau = \frac{1}{2\pi f_C}$$

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

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

People are more sensitive to jitter at low speeds, and more sensitive to lag at high speeds. Therefore, at low rates of change in $\hat{X}_i$, f_c is minimized to reduce jitter. As the rate of change of $\hat{X}_i$ increases, f_c is increased to reduce lag.

Interactive Demo and Results

http://www.lifl.fr/~casiez/1euro/InteractiveDemo/

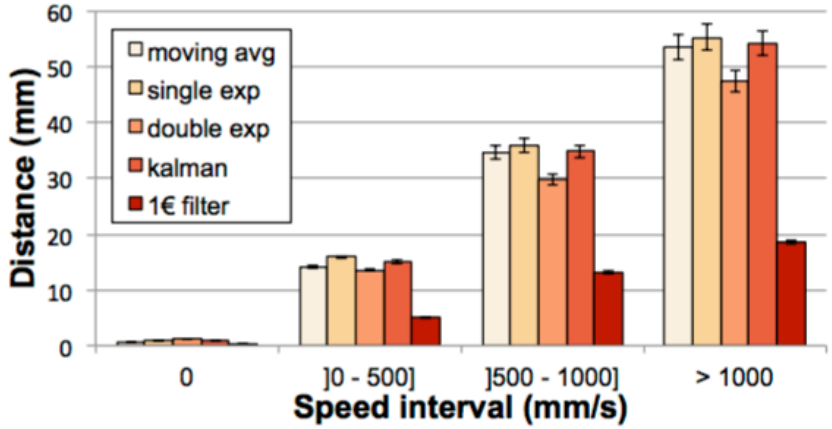


Figure 1. Mean distance between filtered and true cursor position for each speed interval and filter. Error bars represent 95% CI.

The 1€ filter had the smallest Standard Error of the Mean (SEM) at 0.004 mm.

It was 0.013 mm for double exponential smoothing, 0.015 for the Kalman filter and moving average, and 0.016 for single exponential smoothing.

Image source: 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.

Assessment

Pros

- Very easy to understand
- Clear comparison and explanation of various filter types
- Sample code in a number of languages
- Great website with demo and clear instructions

Cons

- Few images or illustrations
- Failed to explain certain concepts e.g. Kalman Filtering
- Few examples of applications outside cursor tracking
- Little quantitative analysis

APPENDIX A - 1€ FILTER

Algorithm 1: 1€ filter

EXT: First time flag: *firstTime* set to *true*
Data update rate: *rate*
Minimum cutoff frequency: *mincutoff*
Cutoff slope: *beta*
Low-pass filter: *xfilt*
Cutoff frequency for derivate: *dcutoff*
Low-pass filter for derivate: *dxfilt*
IN : Noisy sample value: *x*
OUT: Filtered sample value

```
1 if firstTime then
2 |   firstTime ← false
3 |   dx ← 0
4 else
5 |   dx ← (x - xfilt.hatxprev()) * rate
6 end
7 edx ← dxfilt.filter(dx, alpha(rate, dcutoff))
8 cutoff ← mincutoff + beta * |edx|
9 return xfilt.filter(x, alpha(rate, cutoff))
```

Algorithm 2: *Filter* method of Low-pass filter

EXT: First time flag: *firstTime* set to *true*
IN : Noisy sample value : *x*
Alpha value : *alpha*
OUT: Filtered value

```
1 if firstTime then
2 |   firstTime ← false
3 |   hatxprev ← x
4 end
5 hatx ← alpha * x + (1 - alpha) * hatxprev
6 hatxprev ← hatx
7 return hatx
```

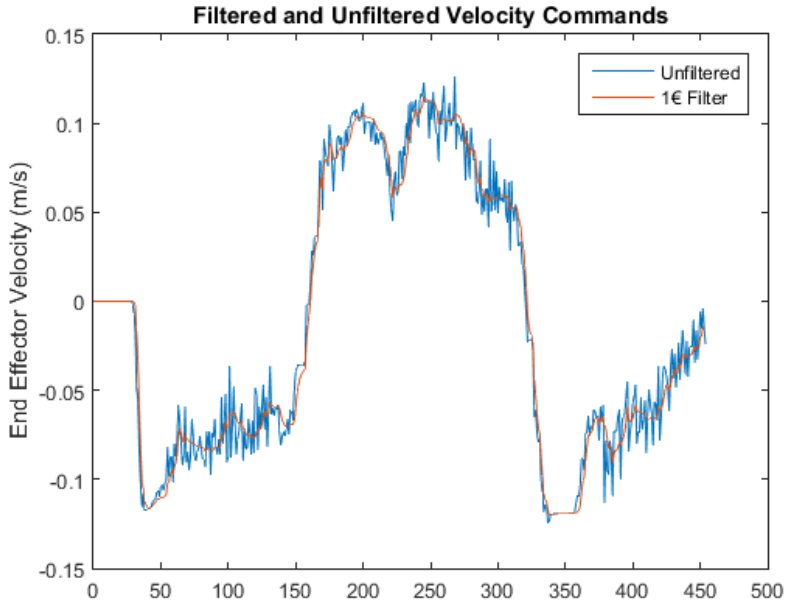
Algorithm 3: Alpha computation

IN : Data update rate in Hz: *rate*
Cutoff frequency in Hz: *cutoff*
OUT: Alpha value for low-pass filter

```
1 tau ← 1.0 / (2*π*cutoff)
2 te ← 1.0 / rate
3 return 1.0 / (1.0 + tau/te)
```

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.

Relevance to Our Project



Questions?