Fast & Efficient Delay Estimation Using Local All-Pass & Kalman Filters

Beth Jelfs and Christopher Gilliam

School of Engineering, RMIT University, Australia

RMIT UNIVERSITY

20th November 2019

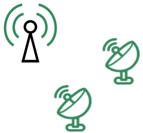
Outline

Delay Estimation

Delay Between 2 or More Spatially Separated Sensors

Communications
Delay between mobile and base stations gives location

Sonar
Delay between sensors represents direction of arrival



Radar
Delay receiving reflection of transmitted pulse gives range

Biology
Delay between sensors represents conduction velocity

Wide Range of Different Applications

Time-Varying Delay Estimation

The problem:

$$x_1(t) = f(t) + e_1(t)$$
$$x_2(t) = f\big(t - \tau(t)\big) + e_2(t)$$

- $x_1(t)$ and $x_2(t)$ are the signals at each sensor at time t
- $f(t)$ is the signal of interest
- $\tau(t)$ is the time-varying delay
- $e(t)$ are additive Gaussian noises

Time-Varying Delay Estimation

The problem:

$$x_1(t) = f(t) + e_1(t)$$
$$x_2(t) = f\big(t - \tau(t)\big) + e_2(t)$$

- $x_1(t)$ and $x_2(t)$ are the signals at each sensor at time t
- $f(t)$ is the signal of interest
- $\tau(t)$ is the time-varying delay
- $e(t)$ are additive Gaussian noises

Requirements:

- ↬ Robust
- ↬ Accurate
- ↬ Real time operation

Time-Varying Delay Estimation

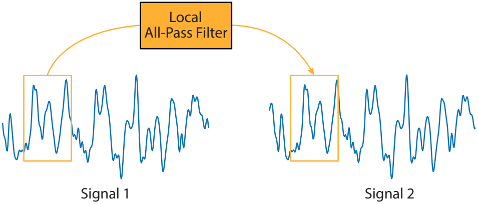
The problem:

$$x_1(t) = f(t) + e_1(t)$$
$$x_2(t) = f\big(t - \tau(t)\big) + e_2(t)$$

- $x_1(t)$ and $x_2(t)$ are the signals at each sensor at time t
- $f(t)$ is the signal of interest
- $\tau(t)$ is the time-varying delay
- $e(t)$ are additive Gaussian noises

Our Previous Solution:

- Robust ✓
- Accurate ✓
- Real-time operation



Time-Varying Delay Estimation

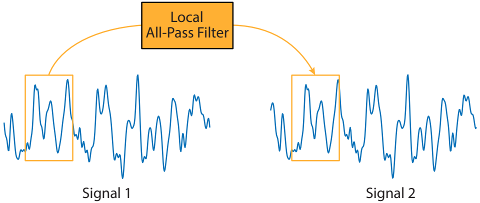
The problem:

$$x_1(t) = f(t) + e_1(t)$$
$$x_2(t) = f\big(t - \tau(t)\big) + e_2(t)$$

- $x_1(t)$ and $x_2(t)$ are the signals at each sensor at time t
- $f(t)$ is the signal of interest
- $\tau(t)$ is the time-varying delay
- $e(t)$ are additive Gaussian noises

Our Previous Solution:

- Robust ✓
- Accurate ✓
- Real-time operation



Delay can be estimated from local all-pass (LAP) filter
Need a real-time solution

All-Pass Filters

- Frequency response

$$H(\omega) = \frac{P\left(e^{j\omega}\right)}{P\left(e^{-j\omega}\right)}$$

  - p real digital filter
  - $P\left(e^{j\omega}\right)$ is forward filter
  - $P\left(e^{-j\omega}\right)$ is backward version

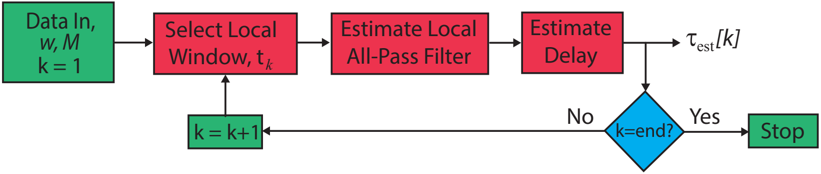
- Filtering operation

$$x_2[k] = h[k] * x_1[k] \iff p[-k] * x_2[k] = p[k] * x_1[k]$$

  - k denotes discrete time

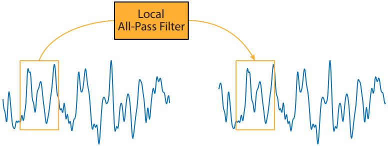
- All-pass filter can be obtained by estimating $p[k]$

LAP Framework

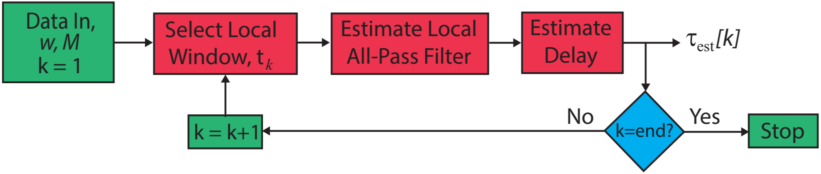


Inputs:

- Data - signals from different sensors
- w - window size of the local region $\mathcal{W}$
- M - size of the filter basis



LAP Framework



Estimating the filter:

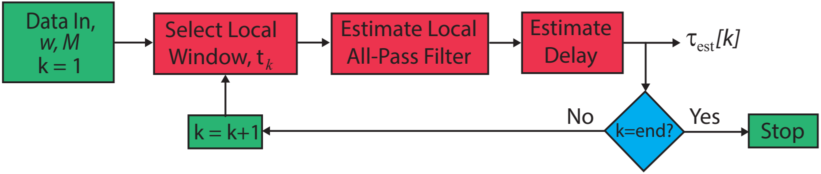
- For current time t_k select the local region $\mathcal{W}$
- Solve the following minimisation:

$$\min_{c} \sum_{k \in \mathcal{W}} \left| p_{\mathrm{app}}[k] * x_1[k] - p_{\mathrm{app}}[-k] * x_2[k] \right|^2$$

- c - coefficient of the filter basis
- p_{app} - filter basis approximation of p
  - In this case Gaussian & first derivative

$$p_{\mathrm{app}}[k] = \mathrm{e}^{-k^2/2\sigma^2} + c\,k\,\mathrm{e}^{-k^2/2\sigma^2}$$

LAP Framework



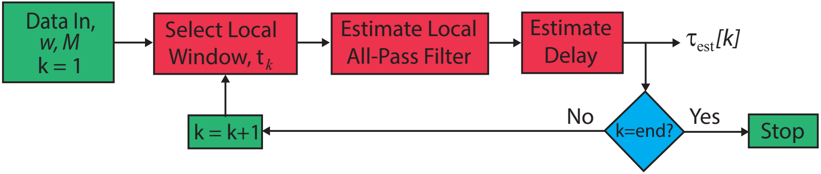
Estimating the delay:

- Extracted from the impulse response p_{app}

$$\tau_{\text{est}} = 2\frac{\sum_k k p_{\text{app}}[k]}{\sum_k p_{\text{app}}[k]}$$

- Repeated for each time sample k
- w defines the time over which the delay is assumed constant
- M defines the maximum size of delay

LAP Framework

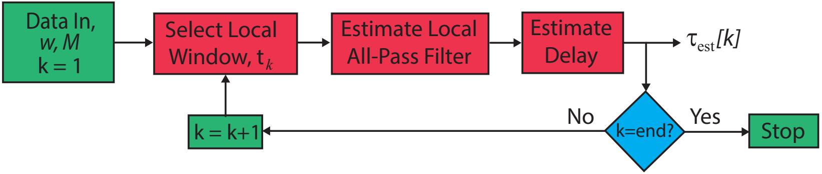


- Fast & efficient to estimate delays
- Large delays require large filters
- Large filters require large windows

$$w_{min} = 2M + 1$$

- Equivalent to assuming large delays slowly varying

LAP Framework



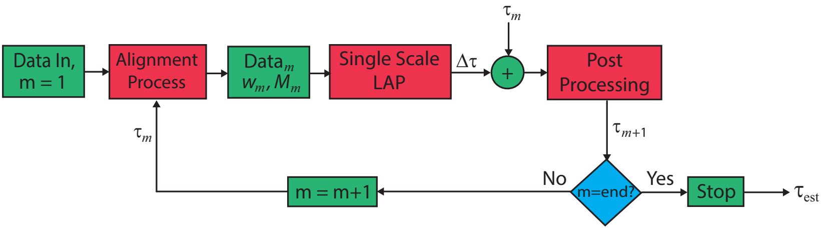
- Fast & efficient to estimate delays
- Large delays require large filters
- Large filters require large windows

$$w_{min} = 2M + 1$$

- Equivalent to assuming large delays slowly varying

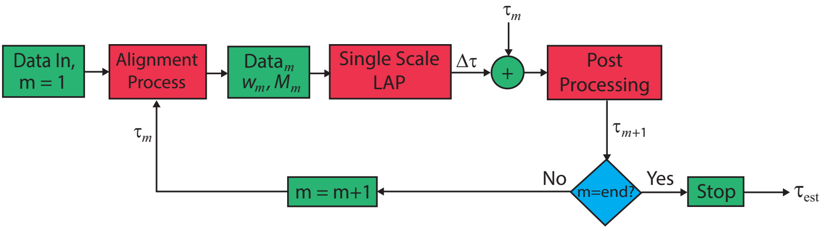
Larger windows $\looparrowright$ more accurate delay estimation
Larger windows $\looparrowright$ restrict the amount of time-variation in the delay

Multiscale LAP



- Implements several different values of M sequentially
- First uses the largest value of M to estimate the delay
- Uses estimate to warp delayed signal closer to original signal
- Repeats with the next value of M

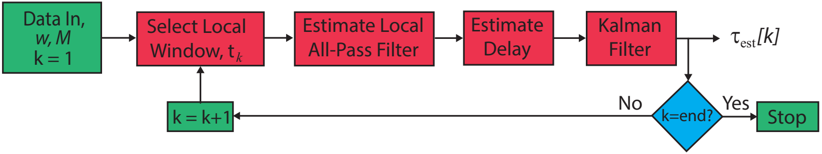
Multiscale LAP



- Implements several different values of M sequentially
- First uses the largest value of M to estimate the delay
- Uses estimate to warp delayed signal closer to original signal
- Repeats with the next value of M

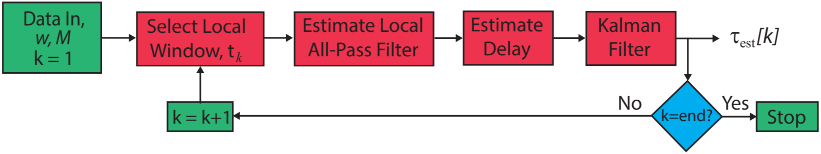
Enables estimation of both quickly and slowly varying delays
Requires the entire signal before processing

LAP + Kalman Filter



- Single scale LAP estimates per sample delay
- Requires only the samples in the local region $\mathcal{W}$
- M can be chosen based on prior knowledge

LAP + Kalman Filter

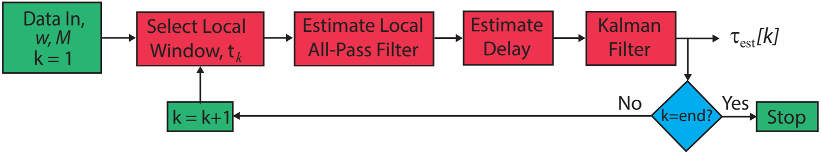


- Single scale LAP estimates per sample delay
- Requires only the samples in the local region $\mathcal{W}$
- M can be chosen based on prior knowledge

Choice of w

- Want maximum possible variation
- Maintain accuracy

LAP + Kalman Filter



- Single scale LAP estimates per sample delay
- Requires only the samples in the local region $\mathcal{W}$
- M can be chosen based on prior knowledge

Choice of w

- Want maximum possible variation
- Maintain accuracy

Kalman Filter

- Assume output of the LAP is a noisy version of the true delay
- Use Kalman filter to model the structure of the delay & the noise

Kalman Filter Model

State vector:

- Based on the LAP estimate of the delay, τ_{LAP}

$$\boldsymbol{\tau}_k = \begin{pmatrix} \tau_k \\ \dot{\tau}_k \\ \ddot{\tau}_k \end{pmatrix}$$

Process:

- Governed by the following equations:

$$\begin{aligned} \boldsymbol{\tau}_k &= A_k \boldsymbol{\tau}_{k-1} + u_k \\ \tau_{\text{LAP}_k} &= C_k \boldsymbol{\tau}_k + v_k \end{aligned}$$

- u and v independent Gaussian noise processes

Transition matrix:

- For a given sampling period Δt

$$A_k = \begin{pmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}$$

Measurement matrix:

$$C_k = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

Kalman Filter Updates

Prediction Updates:

- Prior state estimate

$$\boldsymbol{\tau}_{k|k-1} = A_k \boldsymbol{\tau}_{k-1}$$

- Prior state error covariance

$$P_{k|k-1} = A_k P_k A_k^T + Q_k$$

- Q - process noise covariance

Correction Updates:

- State update

$$\boldsymbol{\tau}_k = \boldsymbol{\tau}_{k|k-1} + K\left(\tau_{\mathsf{LAP}_k} - C_k \boldsymbol{\tau}_{k|k-1}\right)$$

- State error covariance update

$$P_k = \left(I - KC_k\right) P_{k|k-1}$$

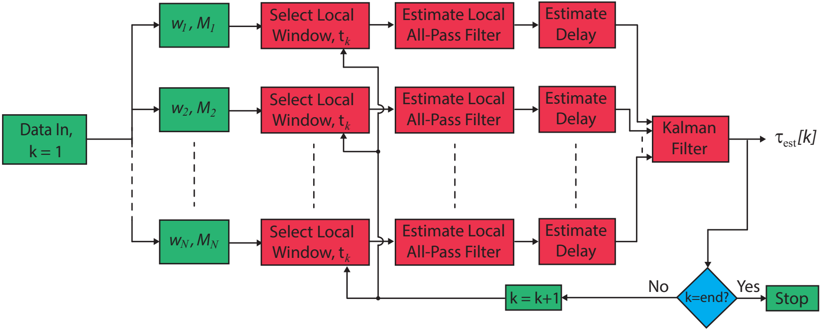
Kalman Gain:

- Update

$$K = P_{k|k-1} C_k^T \left(C_k P_{k|k-1} C_k^T + R_k\right)^{-1}$$

- R - measurement noise covariances

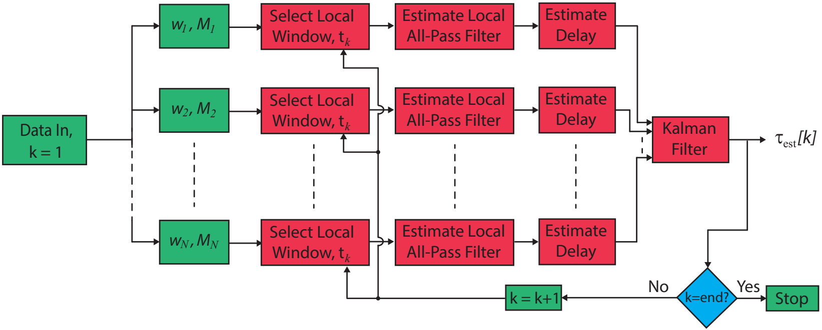
LAP + Kalman Fusion



LAP + Kalman Filter:

- Allows short window lengths without loss of accuracy
- Still limited by the size of the half support of the LAP filter

LAP + Kalman Fusion



LAP + Kalman Filter:

- Allows short window lengths without loss of accuracy
- Still limited by the size of the half support of the LAP filter

LAP + Kalman Filter Fusion:

- Different values of M implemented separately
- Can be implemented in parallel $\hookrightarrow$ fast & efficient computation

Kalman Filter Fusion

State vector fusion:

- Produces filtered state vectors
- Combines to give an updated estimate

Measurement fusion:

- Combines the measurements
- Then updates the state vector

Measurement fusion preferable & can be obtained by:

- Augmenting the observation vector
- Weighting the observations

Equivalent for identical measurement matrices $\looparrowright$ We have implemented an augmented observation:

$$\tau_{\text{LAP}_k} = \left[\tau^1_{\text{LAP}_k} \dots \tau^N_{\text{LAP}_k}\right]^T$$
$$C_k = \left[C^1_k \dots C^N_k\right]^T$$
$$R_k = \text{diag}\left[R^1_k \dots R^N_k\right],$$

where N is the number of LAP filters to be fused.

Synthetic Data

1st Channel:
White Gaussian noise filtered using FIR filter with known spectral properties

Synthetic Data

1st Channel:
White Gaussian noise filtered using FIR filter with known spectral properties

$\Rightarrow$

2nd Channel:
Generated via interpolation using $\tau(t)$

Synthetic Data

1st Channel:
White Gaussian noise filtered using FIR filter with known spectral properties

$\Rightarrow$

2nd Channel:
Generated via interpolation using $\tau(t)$

$\Rightarrow$

Each channel corrupted by Gaussian noise
shaped by FIR filter to simulate acquisition device

Synthetic Data

1st Channel:
White Gaussian noise filtered using FIR filter with known spectral properties

⇒

2nd Channel:
Generated via interpolation using $\tau(t)$

⇒

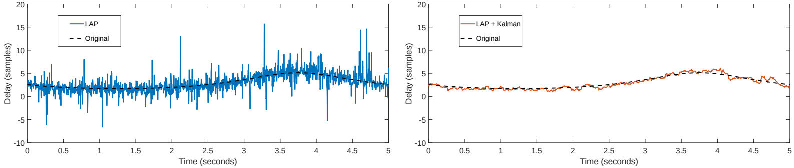
Each channel corrupted by Gaussian noise
shaped by FIR filter to simulate acquisition device

- Generate 5 seconds of synthetic data
- Sampling rate $F_s = 2048$Hz
- $w = 2M + 1$ in all simulations

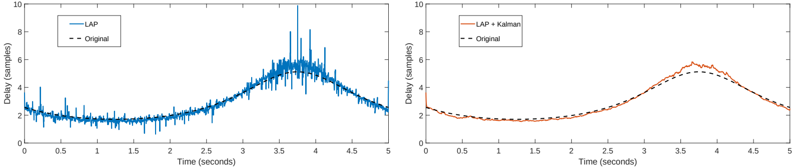
Comparison of LAP & LAP + Kalman

Single scale LAP algorithm with $M = 8$ and the LAP + Kalman filter

10 dB SNR

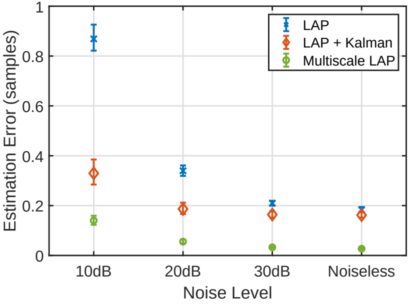


30 dB SNR



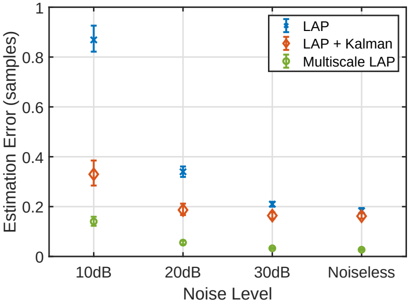
LAP + Kalman gives a smoother estimate of the delay

Multiscale LAP Comparison



- Multiscale LAP:
  - $M = \{2, 4, 8\}$
  - $w = 512$
- LAP & LAP + Kalman
  - $M = 8$
  - $w = 17$
- Average mean absolute error obtained from 100 realisations

Multiscale LAP Comparison



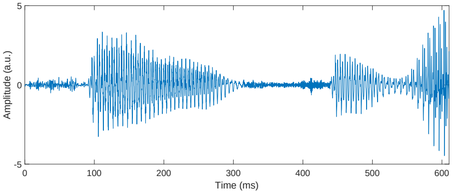
- Multiscale LAP:
  - $M = \{2, 4, 8\}$
  - $w = 512$
- LAP & LAP + Kalman
  - $M = 8$
  - $w = 17$
- Average mean absolute error obtained from 100 realisations

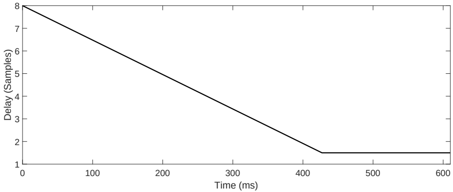
- Computation time - to process 5 seconds of data
- Latency - time taken to provide an estimate of the current delay

	Computation Time (ms)	Latency (ms)
LAP	2.9	8.3
LAP + Kalman	34.5	8.3
Multiscale LAP	35.9	5000.0

Speech

- Real world speech signal and introduce a known delay
- Short 610.4ms speech signal with 5000 samples (sampling rate of 8192Hz)
- Linearly decreasing delay from 8 samples to 1.5 samples until 3500 samples
- Constant for the remaining samples

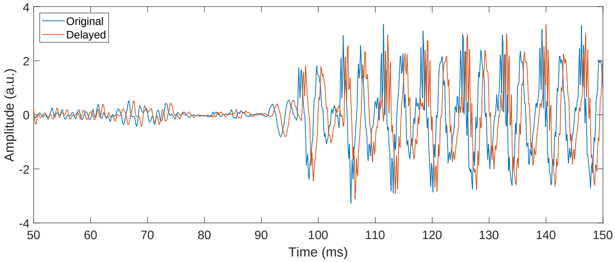




Speech

- Real world speech signal and introduce a known delay
- Short 610.4ms speech signal with 5000 samples (sampling rate of 8192Hz)
- Linearly decreasing delay from 8 samples to 1.5 samples until 3500 samples
- Constant for the remaining samples

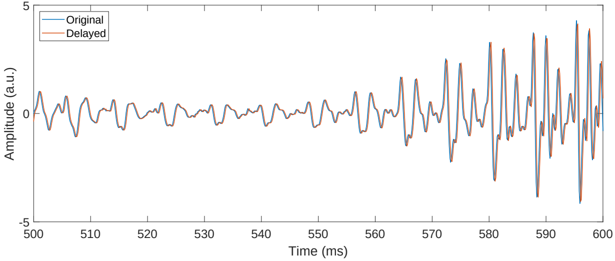
50–150ms ↬ delays of 7.2–5.7 samples



Speech

- Real world speech signal and introduce a known delay
- Short 610.4ms speech signal with 5000 samples (sampling rate of 8192Hz)
- Linearly decreasing delay from 8 samples to 1.5 samples until 3500 samples
- Constant for the remaining samples

500–600ms ↬ constant delay of 1.5 samples



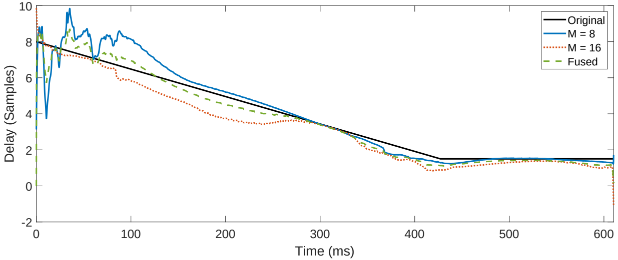
Filter Fusion Results

Estimation problem is not straightforward

- Speech signal is non-stationary
- Several different frequency components
- Non-constant delay

Mean Absolute Errors

	LAP	LAP + Kalman
$M{=}8$	1.001	0.408
$M{=}16$	0.620	0.509
fused	–	0.310



Larger errors ↬ Small amplitude of signal

Conclusions

LAP + Kalman Filter

- Solution provides small errors and accurate tracking of delay
- Results not as accurate as multiscale framework
- Lower latency $\looparrowright$ capable of working in real-time

LAP+ Kalman Filter Fusion

- Combine multiple single scale LAP filters
- Better estimates than individual filters
- Different scales can be implemented in parallel

The End

Thank you for listening