NON-LINEAR PARTIAL DIFFERENTIAL EQUATIONS FOR THE REMOVAL OF NOISE IN AUDIO SIGNAL

Saten Harutunyan
Liana Minasyan
Nelson Mkrtchyan
Supervisor: Michael Poghosyan

May 27, 2019

Contents

1 Abstract

This paper expands on the topic of audio signal noise removal by the usage of partial differential equations(PDE's), particularly nonlinear diffusion equations. Several methods for noise removal are mentioned, and two of methods are implemented by our group. However, only one of them gives the desired results and this method,is discussed in a more detailed way in section 5. Both of the implemented methods are based on usage of nonlinear diffusion equations. The first one uses it on the original signal and the other one on the short time Fourier transform of the signal. The solutions for the mentioned methods use the finite difference method for the numerical solution of the PDE.

2 Introduction

Virtually all audio signals contain distortions resembling some type of noise, either additive or convolution. There are many types of convolution noises, such as electronic, acoustic, electromagnetic, electrostatic, noise due to communication channel distortion and fading. Noise and distortion limit the capacity of data transmission in telecommunication and the accuracy of results in signal measurement systems. Noise reduction is essential in applications such as cellular mobile communication, speech recognition, medical signal processing, music and speech restoration, in music recording studios, and in many other commercial industries.

Since in this work we will be dealing with additive noise it is worth mentioning some types of it which differ in frequency, in the spectrum or time characteristics. Some representatives of such noise are White noise, Black noise, Gaussian noise, Pink noise or Flicker noise, Brown noise or Brownian noise, Contaminated Gaussian noise, Power-law noise and others.

3 Some popular techniques for audio denoising

3.1 Statistical methods for noise removal from audio signal

There are various statistical methods that use the variances of the noise and contaminated signal and make use of spectral subtraction method; when the short-time power spectrum of statistically estimated noise profile is subtracted from the short-time power spectrum of the noisy audio. Short time power spectrum resembles the square of the magnitude of short time Fourier transform of the given signal. Even though the statistical methods have rather low computational complexity, they use static estimation parameters while noise is noise profile is non-static in time.

3.2 Wavelet transforms for audio noise removal

here are methods that use complex wavelets for reducing noise and have proved to be excellent techniques. At first, the given signal is transformed into wavelet representation then some operations are performed on the wavelet coefficients and then the modified coefficients are used to reconstruct the processed version of the signal.

3.3 Nonnegative Matrix Factorization(NMF)

There are approaches that use non-negative matrix factorization and these methods are able to deal with non-stationary noise profile(non-stationary signals are not constant in their statistical parameters over time) in contrast to many statistical methods

3.4 Ideal Masking

One of the recent methods is ideal masking for effective audio separation. IM follows a feature extraction process, which determines if the specific segment of the data is noise dominant or speech-dominant and a mask estimation process for the re-synthesis of the signal. The feature extraction process, in turn, requires a Deep Neural Network.

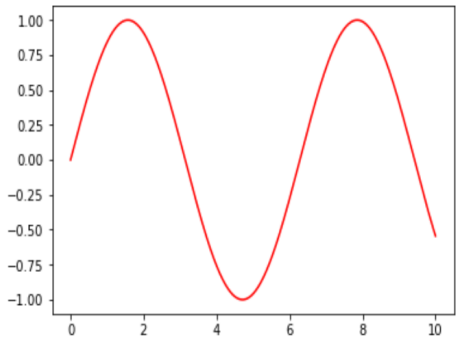
4 Methodology

4.1 Common methods used

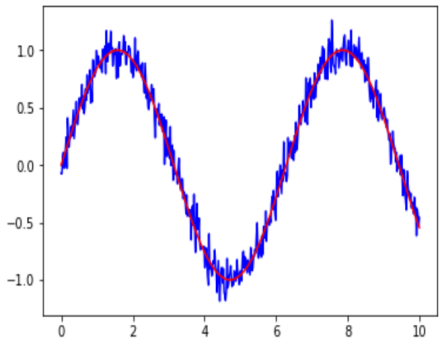
Both our implemented versions make use of the following methods. We start with the original sound that contains no noise and make use of the additive noise to have a contaminated signal. Different nonlinear diffusion equations are then applied to the different transformations of the signal for both cases which has a smoothing effect on a signal applied and produces different results depending on the nature of signal which will be demonstrated in this paper.

4.2 Additive Noise

In our work, a white Gaussian, the most common type of noise, is generated and added to a signal to have a noisy signal. This noise is entirely random, with equal amplitudes at each frequency band which means its spectral density has the same power in any given bandwidth. It is unpredictable and uncorrelated and is similar to many background noises like wind, noise in electronic components. Therefore it can be a good candidate for simulating a non-additive noise. Here is a visual representation of adding noise to a signal; additive white Gaussian noise (AWGN) on sine function,



(a) Simple Sine Wave



(b) Sine wave with a Gaussian noise

Figure 1: Adding White noise to a signal

4.3 Nonlinear diffusion equation

For image processing, diffusion processes are considered to be one of the most effective denoising techniques. Even though linear diffusion is highly effective in removing noise, it removes or at least severely degrades, important image features along with the noise. However nonlinear diffusion processes allow treating details better. It turns out diffusion filters, particularly the nonlinear ones, are effective for denoising other signals, such as audio signals.

4.4 Smoothing with diffusion equation

Noise removal from audio signal aims at removing the noise without lowering the quality of the original one. Many experiments show that the amplitudes of the actual signal(y-axis values) change smoothly as a function of the x-axis values, while many kinds of noise are represented as rapid, random changes in amplitude from point to point in the signal. Therefore for reducing the noise, a process called smoothing is being used. It modifies the data points of a signal so that individual points that are higher than the immediately adjacent points (those of noise) are reduced, and points that are lower than the adjacent points are increased. This eventually leads to a smoother signal, and since the original signal is actually smooth, it will not be much distorted by smoothing, but the noise will be reduced. The nonlinear diffusion equation used in our work does the work of the mentioned smoothing. In our first model, we feed it with the noisy signal, in the second model, the input is the short time Fourier transform of the noisy signal in this case smoothing will rather remove the original signal instead of the noise.

5 Some popular techniques used for audio signal denoising

5.1 Statistical methods for noise removal from audio signal

There are various statistical methods that use the variances of the noise and contaminated signal and make use of spectral subtraction method; when the short-time power spectrum of statistically estimated noise profile is subtracted from the short-time power spectrum of the noisy audio. Short time power spectrum resembles the square of the magnitude of short time Fourier transform of the given signal. Even though the statistical methods have rather low computational complexity, they use static estimation parameters while noise is noise profile is non-static in time.

5.2 Wavelet transforms for audio noise removal

here are methods that use complex wavelets for reducing noise and have proved to be excellent techniques. At first, the given signal is transformed into wavelet representation then some operations are performed on the wavelet coefficients and then the modified coefficients are used to reconstruct the processed version of the signal.

5.3 Nonnegative Matrix Factorization(NMF)

There are approaches that use non-negative matrix factorization and these methods are able to deal with non-stationary noise profile(non-stationary signals are not constant in their statistical parameters over time) in contrast to many statistical methods

5.4 Ideal Masking

One of the recent methods is ideal masking for effective audio separation. IM follows a feature extraction process, which determines if the specific segment of the data is noise dominant or speech-dominant and a mask estimation process for the re-synthesis of the signal. The feature extraction process, in turn, requires a Deep Neural Network.

6 Model 1

6.1 Perona Malik Diffusivity Model

In the first version of our implementation we have used the following one-dimensional diffusion equation:

$$v_t = \partial_x(g(v_x^2)v_x), \qquad (x,t) \in \mathbb{R} \times (0,+\infty)$$

with the initial condition,

$$v(z,0) = s(z)$$

This PDE is solved using the finite difference method,

$$v_{i,k+1} = v_{i,k} + \frac{\iota}{h}(g((u_{i,k})^2)u_{i,k} - g((u_{i-1,k}^2)u_{i-1,k})$$

$$u_{i,k} = \frac{1}{h}(v_{i+1,k} - v_{i,k}), \quad h = \Delta x \quad \text{and} \quad \iota = \Delta t$$

In the PDE, function $g(x)$ is the diffusivity factor, which is a bounded, non-increasing funvtion. The Perona Malik diffusivity is used for $g(x)$ which is the following equation,

$$g(y^2) = \frac{1}{1 + \frac{y^2}{\lambda^2}}$$

Our aim was to find the best values for λ, the threshold parameter. It turns out that values between 0.1 - 0.25 behave well in case of image processing. Trying different ranges of λ, we were unable to detect noise reduction. We could observe slight changes graphically and numerically, but the resulted audio was more noisy than the input audio.Turns out, the application of PDE to denoise the audio signal,applied on the data generated from .wav or .wave file extensions is quite inefficient. The amplitude of noise in different parts of the signal is relatively low, whereas our signal may have parts where the amplitude is very high. So after applying the diffusion equation for smoothing the signal, we will change the amplitudes of the signal in a way that in all parts we will not have relatively high or low values. Which means we will remove some parts of our desired signal. This hypothesis was proved after some tests. The resulted signal had more noise in it than the initial one. By researching the problem, we concluded that applying Diffusion equation on the signal directly does not produce very good results, however, it turned out applying it on a Fourier-transformed version of the signal does in fact produce some good outcomes. During our research, we came across a paper that did the audio signal denoising in the

mentioned way and tried to implement it. The details are discussed in the description of the Model 2.

6.2 Step-by-step guide for implementing the algorithms

6.2.1 Step 1: Adding noise to the sound clip

The sound clips used in our implementations are of format .wav. The file with the .wav or .wave file extension is a Waveform Audio File Format. This is a way of storing audio files, which was created by Microsoft and IBM. This format contains the audio data, sample rate, and bit rate. There are Python libraries that support working with this format. So we read the audio using the python library, wavfile. This allows us to convert our audio signal to an integer array and manipulate the array. Firstly we add to it a randomly generated white noise which is also an array of numbers and add it to the data. This way we create a sound clip with artificial noise.

6.2.2 Step 2: The Windowing process

The idea of the windowing is that we divide our sound clip into regions of a certain size and allow some overlap between the windows. This allows the Discrete Fourier Transform(DFT) only to affect small sections of the signal to damage the sound clip less than in case of bigger windows. Thus, it is desirable to have a window size small enough to have fewer losses but also big enough to obtain enough information about the frequencies. We use a Hamming window function as a windowing function which can be described as follows,

$$\xi(n) = \alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right),$$

where $\alpha = 25/46$ and $\beta = 1 - \alpha$, N is the desired window size and n is a data point position starting at 0 and ending in N. Each window represents a window of data of the sound clip multiplied with the windowing function. Afterward, a DFT is applied on each window which results in a complex array corresponding to each window. The real and imaginary parts of a transformed window are separated for all windows, resulting in 2 different matrices,RePart and ImPart respectively, as the algorithm states, with dimensions number of window and size of the window, where each row corresponds to the real/imaginary part of each transformed window.

6.2.3 Step 3: Creating the Power Spectrum of the sound clip

Our next step is creating the power spectrum of the sound clip which as mentioned is the square of the magnitude of short time Fourier transform of the given signal. We get the magnitude by adding the squares of the real and imaginary matrices and then taking the square root of it. The magnitude of the STFT (short-time Fourier transform) is described by the following equation,

$$STFT = (Re(STFT)^2 + Im(STFT)^2)^{\frac{1}{2}},$$

where $Re(STFT)$ and $Im(STFT)$ are respectively the RePart and ImPart in the algorithm. By this step, we prepare to be used in the PDE which will remove the sound and keep the noise profile. And the spectrogram is the $|STFT|^2$

6.2.4 Step 4: Using the Partial Differential Equation

The nonlinear diffusion equation chosen for the process of removing the signal is the following,

$$u_t = q_x\ C(u)\ u_{xx}$$

$$C(u) = \frac{1}{1 + e^{-k(u-\eta)}}$$

Here u is chosen to be the data in the power spectrum of our signal q_x is the scaling factor for of the diffusive term and η is an estimate of the maximum value of the noise.Boundary conditions are all set to be 0 so that each Fourier window starts and the ends in silence. The finite element method for solving the pde is the following,

$$u_{t+1,j,i} = \frac{q_x \delta t}{\delta x^2} C(u_{t,j,i})(u_{t,j,i-1} - 2 * (u_{t,j,i}) + u_{t,j,i+1})$$

with the initial condition

$$u_{0,x,k} = |STFT|$$

where i is used as the time index, j and k as the spatial indices. where the STFT is the short-time Fourier transform of the audio signal. Applying the PDE to the power spectrum of the sound clip removes the sound and returns the power spectrum of the noise. We used $\eta = 0.05$ and k = 0.5 and T=10, i.e. run FDM 10 times, and take as output u(T,:,:)

6.2.5 Step 5: Spectral Subtraction

Having the power spectrum of the noise from the previous step we subtract it from the power spectrum of the sound clip and get the power spectrum of the original audio.

6.2.6 Step 7: Unwindowing

Unwindowing is the final step when we use the Inverse Fourier Transform to the data we have after step 6. Since in this step we have to recover the original data we also take care of the overlapped sections. This step and all the previous steps mentioned are clearly explained in the implemented code.

7 Conclusion

Comparing the results of the sound clips with added noise that are being denoised with spectral subtraction model we can observe that the noise is actually removed but there are some slight changes to the original audio. This is also noticable in the spectrograms of the audio with added noise and denoised one,

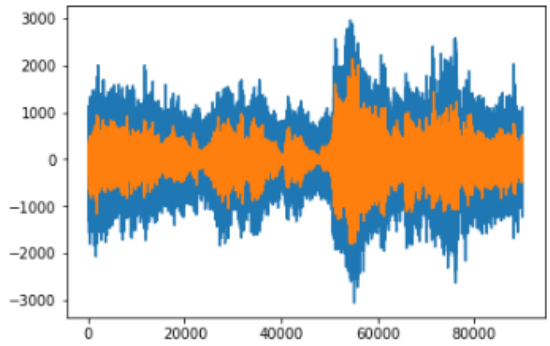


Figure 2: Data of the contaminated and denoised audio

the blue signal data is of the sound clip with the additive noise and the red one is of the denoised version of the noisy sound clip. Here we see the smoothing effect of the diffusion equation which is pretty consistent throughout the signal, but also it damages our audio file slightly. Maybe there are better parameter choices that would allow to eliminate this damage. So with the results of both of our implementations we can conclude that applying diffusion equation to the Power Spectrum of the signal produces better results rather then applying it to the direct signal.

8 Algorithms used for Method 2

8.0.1 Algorithm 1: Windowing

```
1: procedure Windowing
2:   i = 0
3:   Position = 0
4:   for i in NumberofWindows   do
5:       Window   WF * SoundClip(Position : Position + WindowSize)
6:       FourierWindow =   Fourier[Window]
```

```
7:        RePart[i] = Real[FourierWindow]
8:        ImPart[i] = Imaginary[FourierWindow]
9:        Max = Max[Abs[FourierWindow]]
10:       if Max > MaxValue then
11:           MaxValue = Max
12:        i = i + 1
13:       Position = Position + WindowSize * Overlap
14: return [RePart, ImPart, MaxValue]
```

8.0.2 Algorithm 2: UnWindow

```
1: procedure Windowing
2:  i = 0
3:  Position = 0
4:  for i in NumberofWindows do
5:       Window = WF * SoundClip(Position : Position + WindowSize)
6:       FourierWindow = Fourier[Window]
7:       RePart[i] = Real[FourierWindow]
8:       ImPart[i] = Imaginary[FourierWindow]
9:        Max = Max[Abs[FourierWindow]]
10:      if Max > MaxValue then
11:          MaxValue = Max
12:      i = i + 1
13:      Position    Position + WindowSize * Overlap
14:  return [RePart, ImPart, MaxValue]
```

8.0.3 Algorithm 3: Main

```
1: procedure Main:Spectral Subtraction
2:       SoundClip = import the sound clip
3:       ClipLen = length of SoundClip
4:       WindowSize = Desired window size
5:       Overlap = 1 - desired overlap percentage
```

```
6:      [RePart, ImPart, MaxValue] =
            Windowing(SoundClip ,WindowSize, ClipLen, Overlap)
8:      STFT = (RePart2 + ImPart2)
9:      Phase = Arg[RePart + i ImPart]
10:     STFT2 = PDESignalRemoval[STFT/MaxValue]
11:     STFT  = STFT - STFT2
12:     RePart =  STFT * cos(Phase)
13:     ImPart  = STFT * sin(Phase)
14:     SoundClip = Unwindow[RePart,ImPart,MaxValue]
```

References

[1] Jarrod Jay Shipton. The application of non-linear partial differential equations for the removal of noise in audio signal processing. October,2017

[2] Martin Welk, Achim Bergmeister, and Joachim Weickert Denoising of Audio Data by Nonlinear Diffusion