MUMT-605 Final Project

Time/pitch shifting via PSOLA Algorithm

Graham Smith

PSOLA – Overview

- PSOLA – Pitch Synchronous Overlap and Add
- Method for time-stretching and pitch-shifting in the time-domain
- Computationally efficient (especially compared to frequency domain approaches) depending on F0 estimation algorithm
- Works best for harmonic monophonic sounds
- Time-stretching – adjusting the rate of acoustic events without modifying the frequency/pitch
- Pitch-shifting – modifying the perceived pitch of a sound without adjusting the rate of acoustic events

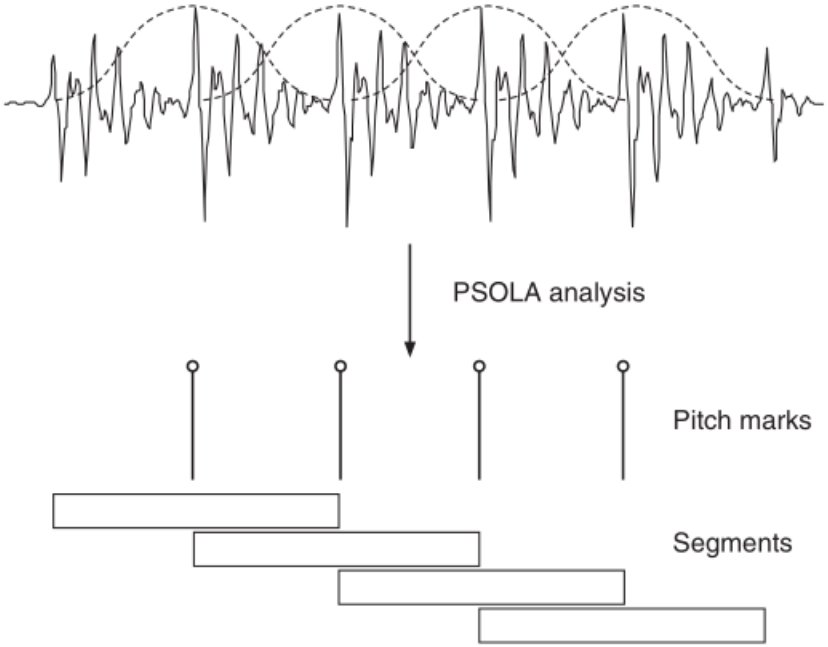
F0 Estimate – YIN Algorithm

- Quite a wide variety of approaches to F0 estimation exist
- Computationally efficient time-domain method
- Published in 2002 by Alain de Cheveingé and Hideki Kawahara
- Similar to autocorrelation method
  - Relies on a harmonic/periodic model for signal analysis
  - Takes an analysis frame of the signal and measures similarity to delayed versions of the signal
  - The amount of delay corresponding to the maximum similarity measure is considered the fundamental frequency

PSOLA – Analysis Stage

- Step 1 – Determine the pitch period and pitch marks
  - Pitch period requires F0 estimation
  - Pitch marks defined to occur where maximum value is
  - Unvoiced sections have pitch period of previous frame
- Step 2 – Extract a segment centered at each pitch mark using a Hanning window with a length of 2 pitch periods
  - Two pitch periods ensures capturing enough data as the underlying signal is harmonic
  - Hanning allows for nice cross-fading and has other properties

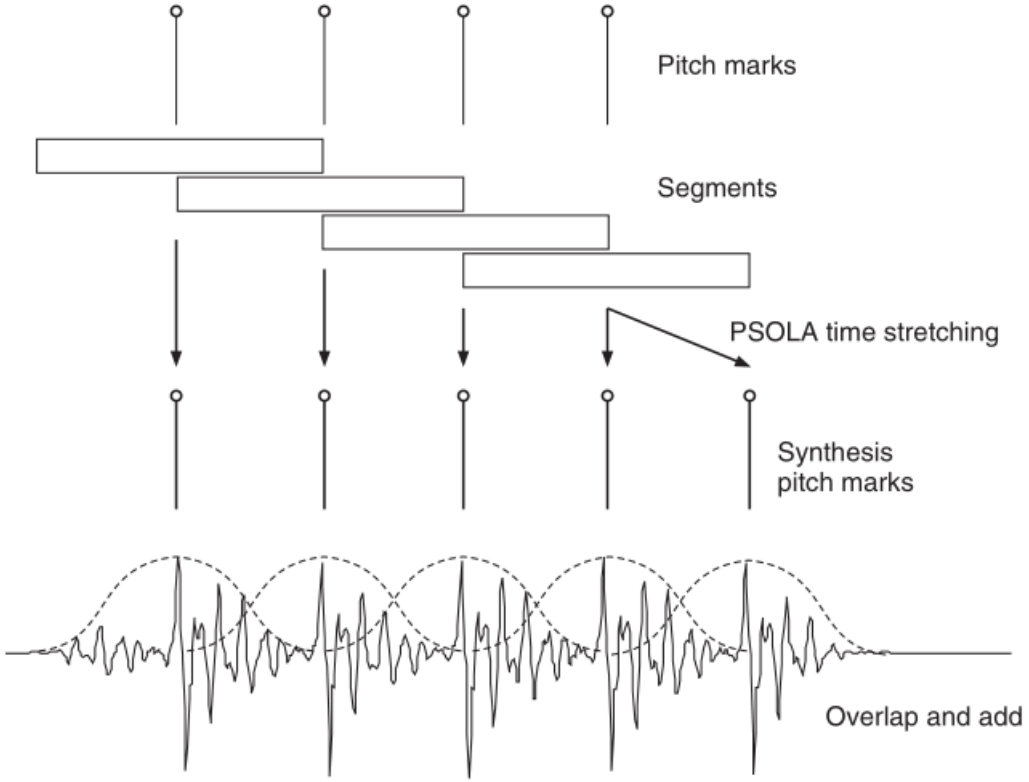
PSOLA Analysis Illustration



PSOLA – Time-Scaling Synthesis

- α = time-scaling parameter
- Steps:
  1) For every synthesis pitch-mark t_k, choose the analysis segment which minimizes $|\alpha * t_i - t_k|$
  2) Extract the windowed segment centered around the pitch mark t_i
  3) Overlap and add the extracted segment at t_k
  4) Determine $t_{k+1} = t_k + P(t_i)$ for next iteration

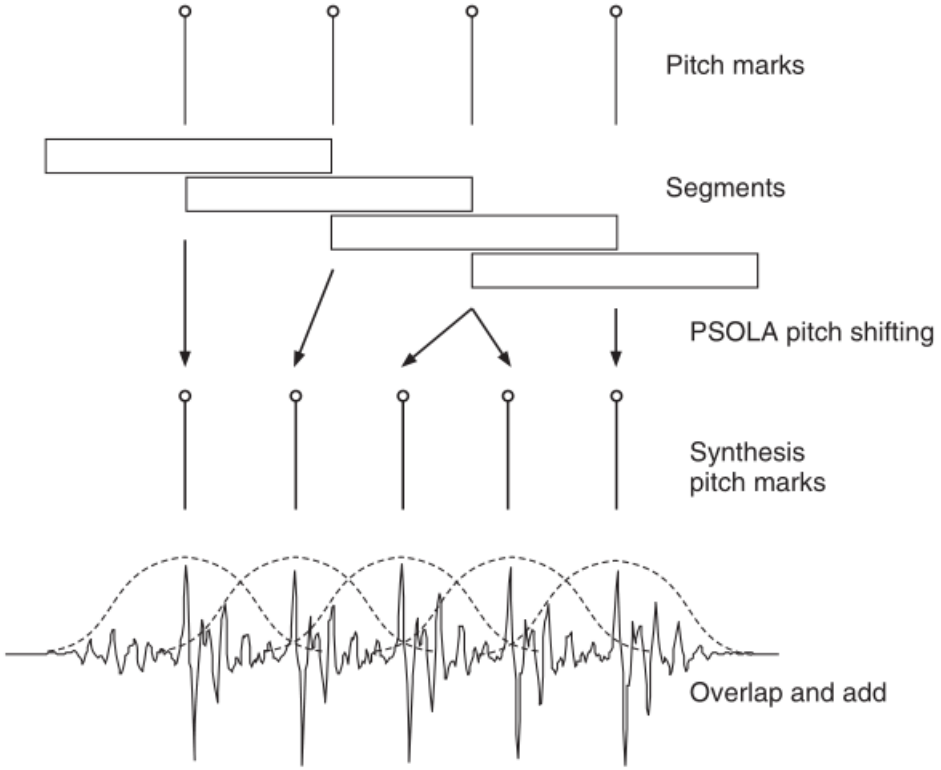
Time-scaling Illustration



PSOLA – Pitch Scaling Synthesis

- β = pitch scaling parameter
- Steps:
  1) Determine analysis segment which minimizes $|t_i - t_k|$
  2) Overlap and add the windowed segment at t_k
  3) Determine next $t_{k+1} = t_k + P(t_i)/\beta$
- Similar to time-scaling but pitch period changes
- Intuitively this makes sense as we're adjusting F0
- Destroys nice windowing property we had before

Pitch shifting illustration



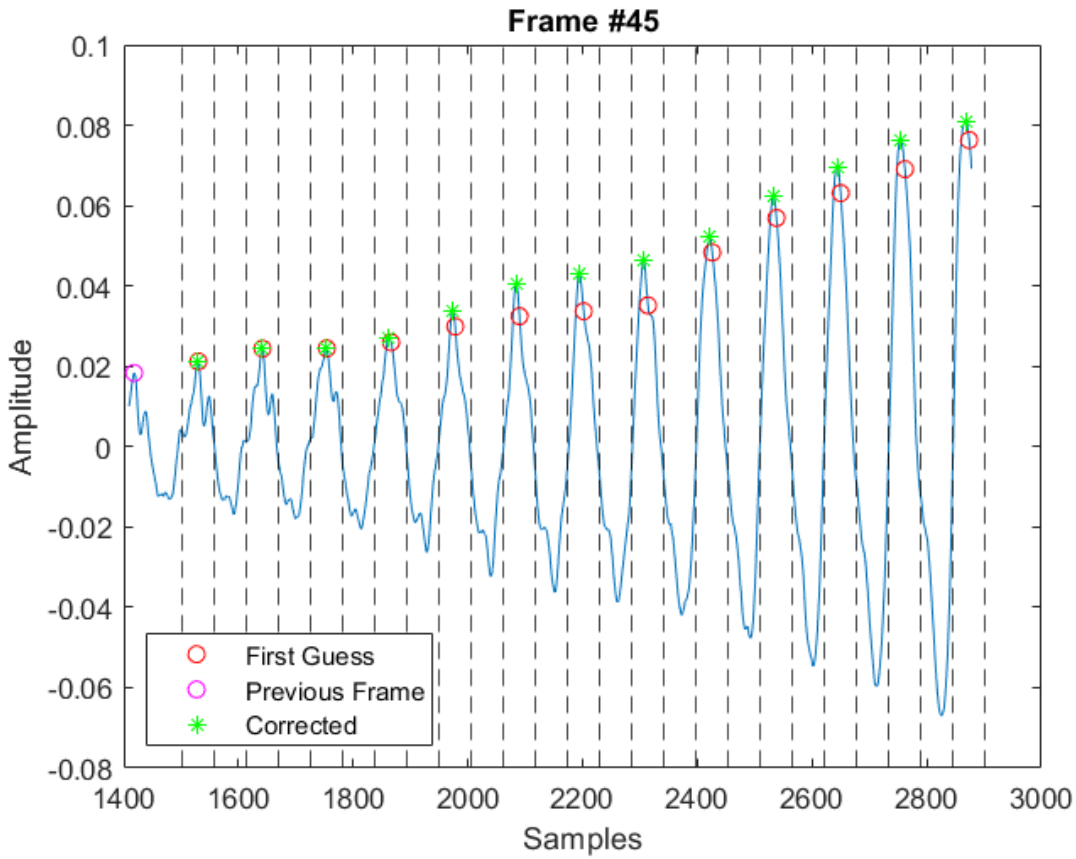
Issues with Pitch Marks

- F0 estimates are useful but there are limitations
- Smallest unit of discretization in time being a sample has ramifications on pitch periods you can’t represent
  - @Fs = 44100 Hz, A440 has a P0 = 44100/440 = 100.22 samples
  - We can only operate in integer samples…
- Use F0 to make an educated guess as to pitch mark location based on P0
- Refine by searching over local range for the max
- Which analysis frame to choose?

Pitch Mark Finding Algorithm

- First frame
  - Find the max value and its corresponding sample index
  - Use F0 estimate for frame to guess where all the other pitch marks for the frame are based on a constant P0
  - Examine an area of P0/2 centered around PM guesses to find the max and set this as the corrected pitch mark
- Subsequent frames
  - Use location of last pitch mark from previous frame to estimate PM locations based on current frame's F0 estimate
  - Revise using the same method of searching for a local maximum

Pitch Mark Finding Illustration



Pitch-Shifting Verification - I

- Fs = 44.1 kHz, β = 1.5
- Use sine wave stimuli at 441 Hz
  - Ensures signal is periodic given sampling frequency
  - P0 = 100 samples
  - Use sine wave over cosine so signal doesn’t start on a discontinuity
- Theoretically we should see a signal at 1.5*441 = 661.5 Hz at output

Pitch Shifting Verification - II

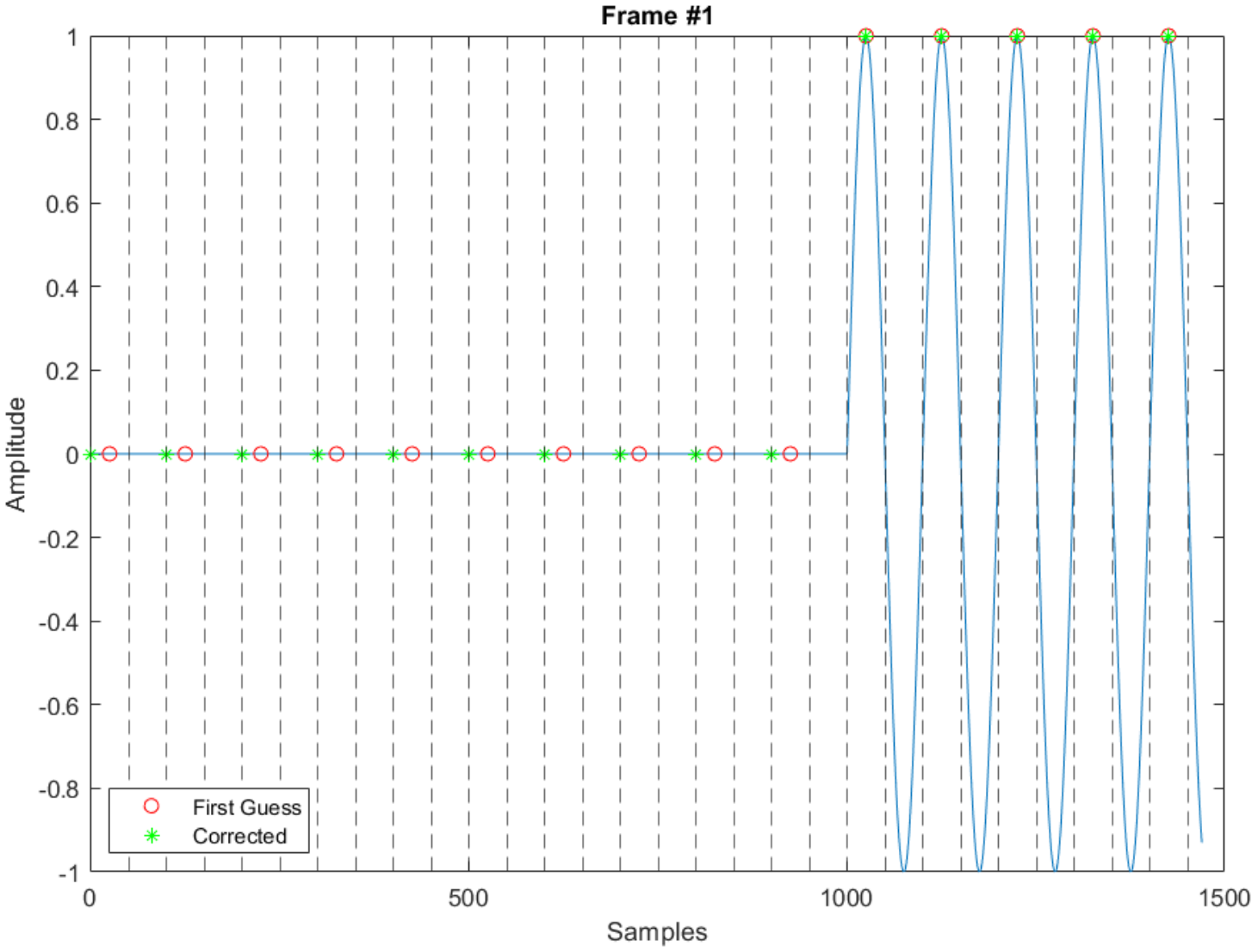
- YIN measured F0 on output signal = 658.2090 Hz
- Measured ratio = 658.2090 / 441 = 1.4925
- Error is introduced from rounding when calculating the synthesis pitch mark
  1) Recall for pitch shifting: $t_{k+1} = t_k + P(t_i)/\beta$
  2) $P(t_i)/\beta = 100/1.5 = 66.67$ samples
  3) Rounds to 67 in code
  4) β = (P0 input / P0 output) = 100 / 67 = 1.4925
- Integer values for samples impose limitations on the algorithm

Time-Scaling Verification - I

- Fs = 44100, α = 1.5
- Stimuli
  - Sine wave at 441 Hz for 3 seconds
  - Pad beginning and end with 10*P0 samples of silence
- Expect duration of each “event” to be scaled by 1.5
  - Silence sections are 1.5*10*100 = 1,500 samples
  - Sine wave is 1.5*3*44100 = 198,450 samples

Time-Scaling Verification - II

- Not the case…
  - First silence at 1475 (or 25 samples under)
  - Tone duration measured at 198,475 (or 25 samples over)
  - 25 samples = P0/4
- Root cause unknown
  - Error likely introduced by pitch marks in unvoiced sections during the first frame
  - Corrected marks moved 25 samples from initial guess


Frame #1
Amplitude
Samples
First Guess
Corrected
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500

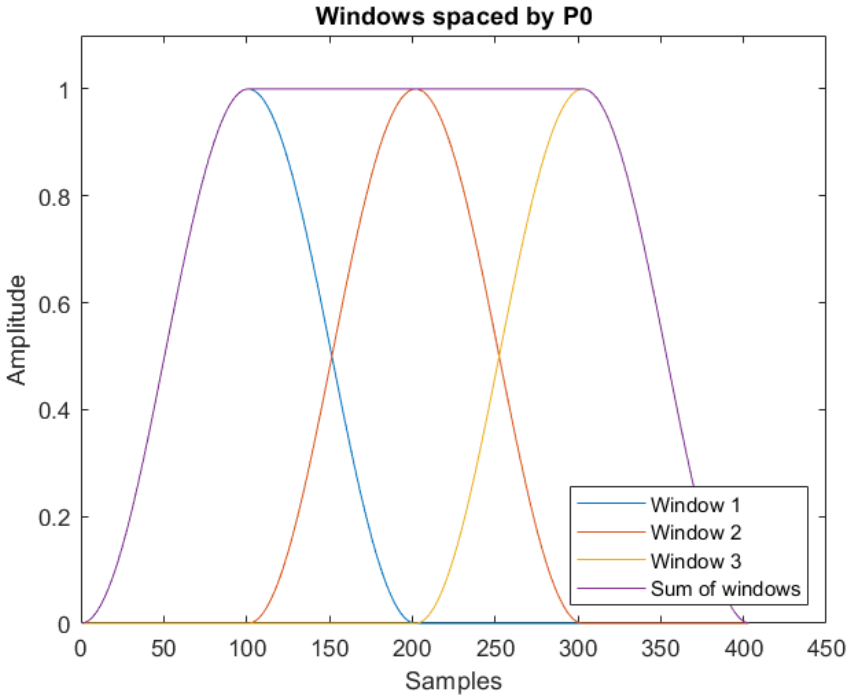
Perceptual Evaluations

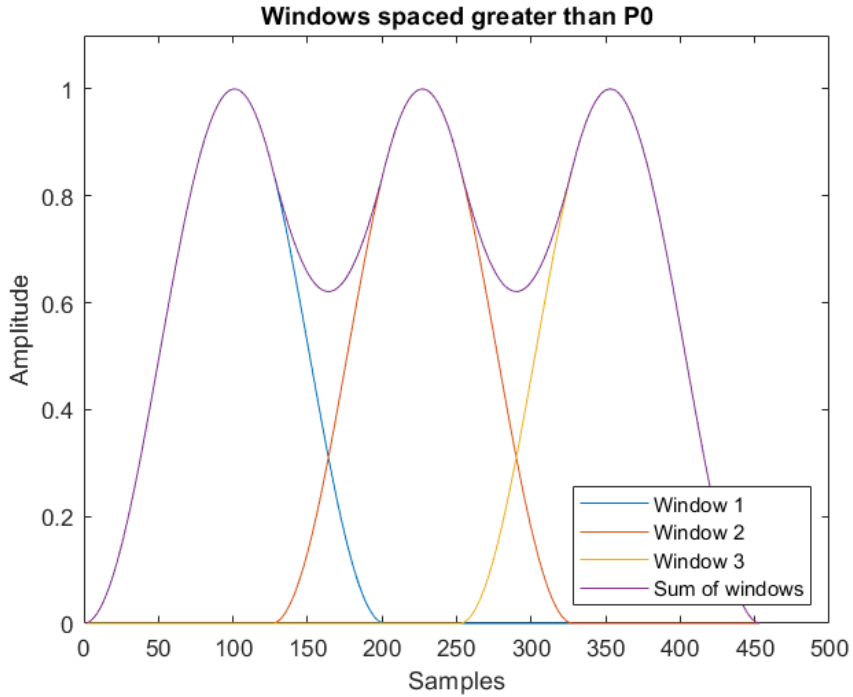
- Algorithm performs quite well for slight adjustments
- Limited to monophonic harmonic signals based on design
- Masking from original sources can help hide artifacts when harmonized
- Combining both pitch/time scaling made the artifacts more present
- Extreme pitch shifting down is interesting timbrally
  - Flute becomes bassoon
  - Vocals sound like smokers

Limitations of Algorithm

- F0 estimations can affect things
- Integer limitations on sample values
- Pyschoacoustic aspects of speech intelligibility and pitch perception affect parameter limits
  - Pitch-shifting limits
    - Lower limits on pitch shifting generated by lack of overlap between segments
    - Upper limits on pitch shifting generated by too much overlap
  - Time-shifting limits
    - Upper limit based on discarded segments
    - Lower limit based on artifacts introduce by repeititon

Window Length/Spacing and P0





Audio Examples

- Flute
  - Original
  - Major third
  - Bassoon (down 4 octaves)
- Voice
  - Original
  - Played at 1.5x speed
  - Smoker (down 4 octaves)