© Teknik Elektro Itenas | No.2 | Vol. 2
Juli - Desember 2014

Transformasi *Pitch* Suara Manusia Menggunakan Metode PSOLA

SUSETYO BAGAS BHASKORO[1], IRNA ARIANI[2], ANANDHA A. ALAMSYAH[3]

1. Prodi Teknik Informatika Universitas Widyatama
2. Teknik Media Digital dan Game Institut Teknologi Bandung
3. Institut Teknologi Sepuluh Nopember Surabaya

Email:susetyo.bagas@widyatama.ac.id

ABSTRAK

Kemampuan pengubahan suara yang dilakukan Dubber untuk beragam bentuk suara menjadi perhatian khusus dengan melakukan rekayasa suara, di dalam perkembangan teknologi di kenal sebuah teknikpitch shifting yang digunakan untuk mengubah suara manusia di bagian timbre dan pitch. Penelitian ini menggunakan metodepitch shifting PSOLA (Pitch Synchronous Overlap Add) untuk merubah pitch sekaligus timbre suara. Proses yang dilakukan meliputi perekaman suara sehingga didapatkan sinyal suara. Sinyal hasil perekaman kemudian diolah untuk menemukan posisi pitch dari sinyal pada domain waktu. Setelah posisi pitch diketahui, jarak antar pitch akan dikalikan dengan bilangan skala pergeseran yang sudah ditentukan. Hasil dari perkalian tersebut adalah perubahan pada pitch suara, sehingga menghasilkan suara yang lebih tinggi atau lebih rendah. Perubahan juga terjadi pada timbre sehingga menghasilkan karakter suara yang berbeda dengan suara aselinya.Hasil pengujian pitch dan timbre dengan menggunakan metode PSOLA menunjukkan keberhasilan mencapai 98% berdasarkan sinyal sinus.

***Kata kunci**: Pitch, Timbre,Pitch Shifting, PSOLA.*

ABSTRACT

The ability of converts sound done in various forms of a dubber sound, becomes a special attention in doing an engineering design sound. In the development of technology the pitch of shifting know a technique that is used to turn the human voice in the timbre and pitch. This study using methods pitch shifting psola (pitch synchronous overlap add) to change the pitch as well as the timbre sound. The process was about recording a sound so obtained up a noise. Recording signals then processed the results to find the position of the pitch signals on the domain of time. After the position of the pitch known, the distance between the pitch will be multiplied by the number of the scale of a shift that had been determined. The result of the multiplication of the sound is a change in pitch , so producing a higher or lower, Also happens to change the timbre that produces characters a different voice with the original sound. The examination result of pitch and timbre using PSOLA method shows the success as big as 98% for signal sinus examination.

***Keywords**: Pitch, Timbre, Pitch Shifting, PSOLA.*

1. PENDAHULUAN

Perkembangan teknologi multimedia menjadikan film animasi berkembang dengan pesat.Saat ini terdapat beragam film animasi yang dijadikan media hiburan sampai dengan media pembelajaranyang terbagi menjadi berbagai kategori umur.Karakteristik yang dimiliki oleh film animasi biasanya gambar, karakter figur, alur cerita dan suara.Suara di dalam film animasi dilakukan oleh seorang *Dubber* untuk menggambarkan emosional karakter figur dalam berkomunikasidengan lawan bicaranya di sebuah alur cerita**(Bhaskoro, 2013)**.Seorang *Dubber* biasanya memiliki peran yang beragam sehingga di dalam menghasilkan karakter suara juga harus beragam.

Sebagai contohnya seorang *Dubber* memiliki peran untuk mengisi suara karakter figur dengan suara yang normal artinya suara yang digunakan oleh seorang *Dubber* adalah suara aseli yang tidak ada perubahan. Namun situasi dapat berbeda jika seorang *Dubber*memiliki peran yang berbeda dengan karakteristik suara aselinya maka fungsi seorang *Dubber* adalah menyamarkan suara aselinya dengan menaikkan maupun menurunkan *pitch* dan *timbre*. Aktifitas untuk mempertahankan konsistensi *pitch* dan *timbre* dalam ukuran tertentu merupakan hal yang sulit dilakukan oleh manusia.Oleh karena itu teknologi dibutuhkan untuk membantu konsistensi keluaran suara dalam ukuran *pitch* dan *timbre* tertentu.

Proses untuk mengubah karakteristik suara manusia bergantung dari kemampuan untuk mengontrol sistem organ penghasil suara manusia diantaranya mulut dan rongga hidung serta kemampuan pernafasan manusia. Manusia yang mampu mengontrol sistem organ suara akan dapat menghasilkan suara yang berbeda. Ganguan pada sistem pernafasan seperti hidung tersumbat juga dapat mengubah suara yang dihasilkan manusia walaupun hal tersebut mungkin tidak diinginkan atau disengaja **(Patton, 2014)**.

Melalui perkembangan teknologi informasi saat ini, proses pengubahan suara manusia dapat dilakukan melalui proses komputasi. Namun untuk melakukan perubahan suara dibutuhkan sebuah aplikasi yang dikembangkan secara khusus.Aplikasi yang dikembangkan merupakan aplikasi yang memanfaatkan media suara manusia untuk diolah dengan komputasi komputer. *Input* suara manusia diolah dan dikomputasi dengan metode *pitch shifting*. Metode *pitch shifting*melakuan perubahan*input*suara manusia (frekuensi suara) dengan memanfaatkan pergeseran *pitch*suara, sehingga *output* yang dihasilkan adalah suara manusia (frekuensi suara) yang berbeda tanpa mengubah kata yang diucapkan.

Meskipun tujuan kami membantu teknik *Dubbing* untuk film animasi, namun beberapa batasan di dalam penelitian ini masih tersedia dan kemungkinan masih dapat dikembangkan, diantaranya proses pengujian menggunakan sampel kata yang diucapkan tidak lebih dari 1 detik, skala pergeseran *pitch* dibatasi pada angka 0.5 sampai 2, dan penelitian ini belum berjalan secara *real-time.*

Beberapa peneliti telah berhasil melakukan penelitian yang terkait dengan pengolahan suara manusia melalui komputasi teknologi informasi. Penelitian yang telah dilakukan oleh Dimple Garg, Sukhvinder Kaur, Dinesh Arora menjelaskan bahwa untuk proses pengenalan suara manusia ada beberapa parameter yang dimanfaatkan, yaitu *cepstrum*, *pitch* dan *formant*. Beberapa metode yang digunakan adalah *Mel-Frequency Cepstrum Coefficient* (MFCC) dan *Linear Predictive Coding* (LPC) **(Kaur, 2014)**. Penelitian ini memfokuskan untuk mengenali suara manusia berdasarkan identitas suara masing-masing pembicara.

Penelitian selanjutnya yang dilakukan oleh Zulkarnain, Barmawi Andriana melakukan penelitian tentang komputasi suara yang memiliki tujuan akhir untuk mengenali suara

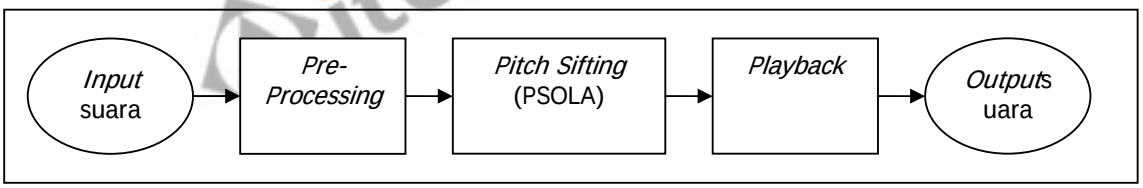
manusia.Metode yang digunakan adalah LPC dengan parameter yang digunakan untuk mengenali suara manusia adalah *formant* **(Zulkarnain, 2013)**.

Penelitian awal yang telah kami lakukan tentang sinyal suara hampir sama dengan penelitian lainnya. Penelitian kami juga fokus untuk mengenali suara manusia dengan parameter yang digunakan adalah *cepstrum*. Usulan kami melakukan modifikasi algoritma *mel-frequency cepstrum coefficients* (MFCC)di blok diagramnya untuk mempersingkat langkah kerja menghasilkan keluaran parameter *cepstrum*.Modifikasi yang dilakukan berkaitan dengan *homomorphic*, hal ini di dalam*digital signal processing* (DSP) dikenal dengan perubahan struktur yang sama **(Bhaskoro, 2013)**. Menurut Alan V. Oppenheim"Dalam sains dan teknik, *homomorphic* itu biasa digunakan untuk menemukan sinyal yang sulit untuk dipahami atau dianalisa dengan melakukan beberapa penyesuaian teknis penyelesaiannya. Strategi *homomorphic*melakukan konversi sinyal menjadi linear pada sistem konvensional" **(Oppenheim, 1992)**.

Penelitian yang dilakukan saat ini mencoba untuk menghasilkan keluaran parameter yang berbeda dari penelitian sebelumnya. Pada penelitian ini fokus terhadap*pitch* dan *timbre*. Tujuan penelitian ini melakukan pencarian parameter *pitch*dan *timbre*untuk merubah intonasi suara dari setiap manusia menjadi suara yang memiliki intonasi tinggi atau intonasi rendah diluar kemampuan dari manusia itu sendiri. Jenis suara yang dihasilkan ini mampu digunakan sebagai suara penyamaran dari sinyal suaraaseli yang dimiliki oleh setiap personal.

2. DESAIN SISTEM

Perancangan sistem yang di jadikan penelitian memiliki beberapa langkah di dalam penyelesaiannya. Beberapa langkah tersebut seperti pada Gambar 1:



Gambar 1. Blok Diagram *Pitch Shifting*

Gambar 1 menjelaskan blok diagram menggunakan metode *pitch shifting*. Namun, sebelum menggunakan metode *pitch shifting* di Gambar 1 terlihat blok diagram yang menjelaskan tentang *pre-processing*. Kegiatan*pre-processing* tersebut penting untuk dilakukan karena sebagai normalisasi data suara yang akan diolah.

Pitch Shifting merupakan teknik yang digunakan untuk merubah *pitch* dari sinyal suara tanpa mempengaruhi durasi ataupun kecepatan dari sinyal suara tersebut. Kebalikan dari proses *pitch shifting* adalah *time stretching* yang digunakan untuk merubah durasi dan kecepatan sinyal suara tanpa merubah *pitch*. Selain menggunakan *pitch shifting* terdapat cara sederhana untuk merubah *pitch*, yaitu dengan melakukan *resample* pada sinyal suara namun cara ini juga mempengaruhi durasi dan kecepatan dari sinyal suara.

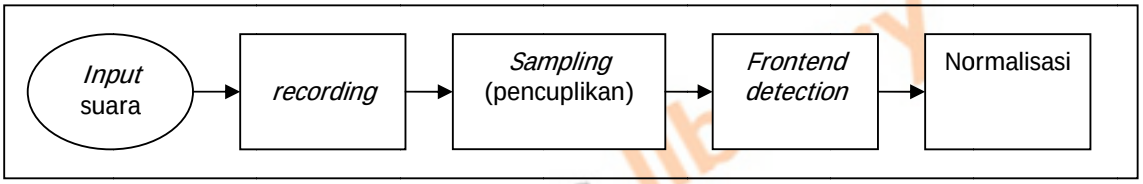
Pitch Synchronous Overlapp Add (PSOLA) merupakan variasi dari algoritma SOLA **(Shrawankar, 2011)**, digunakan untuk pemrosesan sinyal suara yang didasarkan pada *pitch* inputan suara. Algorithma PSOLA disusun dengan 2 tahap, tahap pertama disebut *analysis phase* dan tahap kedua disebut *synthesis phase* **(Oppenheim, 1992)**, **(Patton, 2014)**.

a. *Analysis algorithm*
   1. Mencari dan menandai letak *pitch period* dari masukan sinyal suara. Fase ini disebut juga dengan *pitch mark*;
   2. Membagi sinyal menjadi beberapa *frame* dimana tiap *frame* tadi terdapat *pitch*.

b. *Synthesis algorithm*
   1. Memilih segmen atau *frame* yang memiliki data *pitch mark;*
   2. *Overlap and add segment* yang dipilih.

3.1 *Pre-processing*

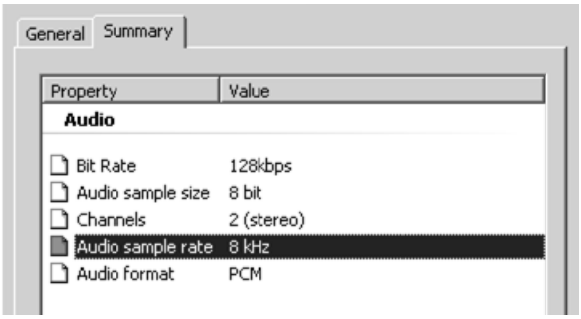
Pre-processing digunakan untuk menghasilkan sebuah sinyal keluaran suara yang memiliki nilai merata dalam jumlah sinyal yang sama. Proses ini dibagi menjadi beberapa proses didalamnya, yaitu: (i) *sampling*, (ii) *frontend detection*, (iii) *normalisasi*.



Gambar 2. Blok Diagram *Pre-Processing*

3.1.1 *Recording*

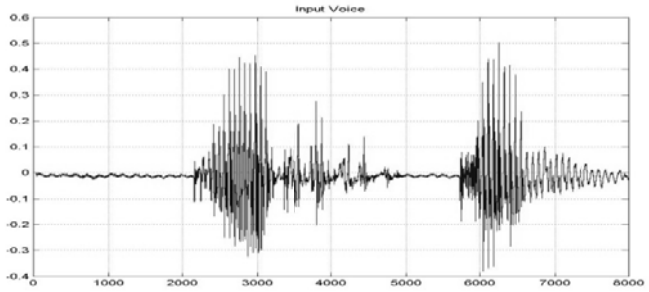
Perekaman suara dilakukan pada kecepatan 8000Hz dengan resolusi (tingkat kuantisasi) 8 bit (1 *byte*) artinya dalam waktu satu detik didapatkan data sebanyak 8000 *byte*, *channel stereo*, lama merekam suara adalah 1 *second* dan disimpan dengan ekstensi .wav. Gambar 3 adalah*properties* hasil pengaturan suara.



Gambar 3. *Properties* Perekaman Suara

Beberapa hal yang diperhatikan lainnya dalam perekaman suara diantaranya:(1) kondisi lingkungan yang minim*noise*, (2) penggunaan perangkat yang sama,seperti penentuan peralatan *microphone, soundcard, volume* perekaman, frekuensi besar *audio sampling*, dan pelafalan atau pengucapan kata-kata, (3) pelafalan dengan intonasi normal (tidak terlalu cepat dan tidak terlalu lambat),(4) amplitudo (keras lemah) dengan intonasi normal (tidak

terlalu tinggi dan tidak terlalu lemah).Gambar 4 adalah hasil gelombang suara dari proses perekaman suara.



Gambar 4. Gelombang Sinyal Perekaman Suara

3.1.2 *Sampling*

Melakukan proses *sampling* karena membutuhkan proses pengambilandatasinyal kontinyu untuk setiap periode tertentu. Proses *sampling* sinyal menurut aturan *nyquist* adalah frekuensi *sampling* harus lebih besar dua kali dari frekuensi maksimum. Jika sinyal *sampling* tidak memenuhi syarat maka akan terjadi *aliasing*. *Aliasing* adalah suatu efek dimana sinyal yang dihasilkan memiliki frekuensi yang berbeda dari sinyal aslinya**(Oppenheim, 1992)**. Persamaan kriteria *nyquist* adalah:

$$f_s \geq 2 \times f_{\max} \quad (1)$$

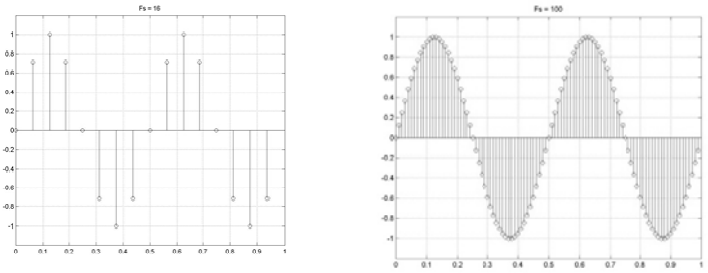
Dimana:
f_s = Frekuensi sinyal sampling
f_{max} = Frekuensi nilai maksimum sinyal informasi disampel

Kecepatan pencuplikan mengikuti pengaturan pada saat proses perekamansuara didasarkan dengan asumsi bahwa sinyal percakapan (*speech*) berada pada daerah frekuensi 300-3400Hz. Semakin tinggi frekuensi *sampling*, maka semakin baik sinyal *digital* yang dihasilkan. Sehingga, jika kecepatan *sampling* dan daerah frekuensi dimasukkan di persamaan *nyquist* maka akan memenuhi kriteria dari rumus tersebut.

$$f_s \geq 2 \times f_{\max} \approx 8000\,Hz \geq (2 \times 3400\,Hz) \quad (2)$$

Dimana:
f_s = 8000Hz
f_{max} = 3400Hz (2 x 3400Hz = 6800Hz)

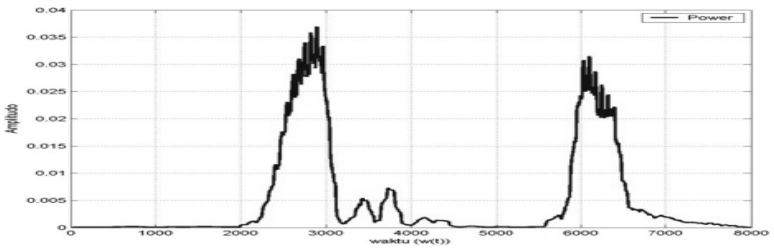
Gambar 5 adalah hasil *sampling* dengan jumlah data 16 dan 100:



Gambar 5. Frekuensi *Sampling*

3.1.3 *Power*

Power atau energi menjadi salah satu bagian blok diagram yang dibutuhkan untuk menghasilkan proses *frontend detection*. *Power* dapat digunakan untuk melihat tinggi dan rendahnya sinyal suara yang dimasukkan atau dengan pengertian lainnya nilai yang dapat membedakan satu *frame* dengan lainnya. Hal ini dapat memudahkan untuk memisahkan sinyal *voiced* dan sinyal yang kosong (*silent*) atau *noise*. Gambar 6 adalah proses dari *power*untuk memperlihatkan letak *voiced* dan *noise*.



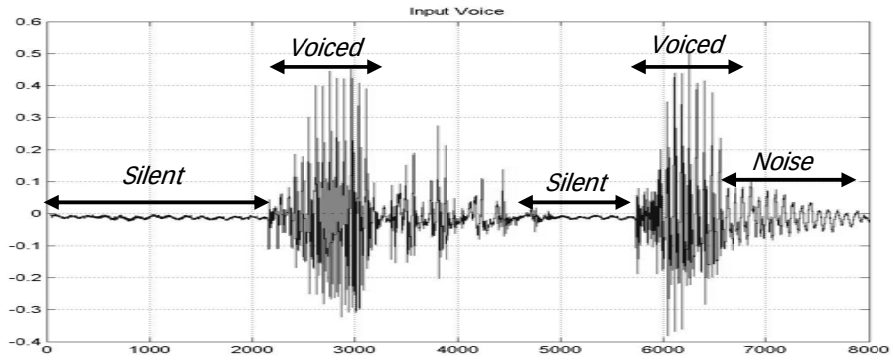
Gambar 6. Sinyal *Power*

Gambar 6 terlihat bahwa *voiced* terletak diantara data ke-2000 sampai data ke-7503. setelah diketahui dimana letak *voiced* dan dimana letak *noise*, maka dapat dilanjutkan ke langkah selanjutnya untuk mengambl nilai *voiced*-nya saja.

3.1.4 *Frontend Detection*

Frontend detection digunakan untuk mengambil data sinyal suara yang berisi *voiced*. Sehingga penggunaan *frontenddetection* ini untuk memisahkan sinyal *noise* dengan sinyal yang berisi data penting berupa ucapan tersebut. Biasanya dalam pengucapan terdapat sinyal *silent (noise)*, pada awal dan akhir dari ucapan, untuk menghilangkan sinyal yang *silent* (*noise)* tersebut, maka penggunaan proses *frontend detection* sangat membantu.

Gambar 7adalahkategori suara *voiced* atau *silent* (*noise)*.*Unvoice* adalah derah dimana *vocal cord* tidak berfungsi. *Silence* adalah daerah dimana sinyal bicara tidak diucapkan. *Voice*adalah daerah dimana sinyal bicara diucapkan.



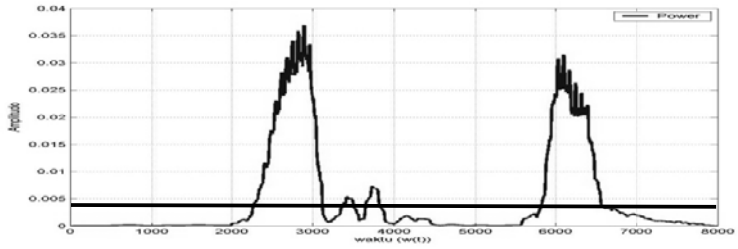
Gambar 7. Pembagian Sinyal Suara

Selanjutnya sinyal suara yang diambil adalah sinyal suara yang berkategori selain *silent* dan *noise*, karena sinyal selain *silent*dan*noise*tersebut terdapat nilai yang penting untuk dijadikan sebuah fitur suara, namun demikian sebelum mendapatkan sinyal *voiced*, terlebih dahulu harus menentukan batasan *(threshold)* dengan menggunakan hasil dari *standard deviasi* dan rata–rata pada proses sebelumnya.Hasil tersebut akan digunakan sebagai parameter untuk menentukan awal dan akhir dari suara, *voiced*akan memiliki nilai *power* yang melebihi nilai dari standar deviasi dan rata-rata dari *voiced.*

$$voiced \geq R_s + S_d \quad (3)$$

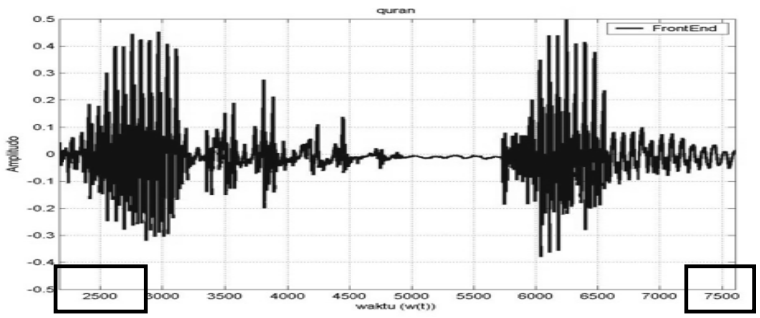
Dimana:
R_s = Rata-rata
S_d = Standar Deviasi

Berdasarkan rumus (3)didapatkan nilai awal dan akhir dari *voiced*. Gambar 8adalah*threshold* dari *voiced*.



Gambar 8. *Threshold* Ucapan

Gambar 8terlihat bahwa sinyal *voiced* terletak pada data ke–2164 sampai data ke–7601. Berarti datake-0 sampai data ke-2163 dan data ke-7602 sampai data ke-8000 dihilangkan. Gambar 9 menampilkan bentuk sinyal *frontend detection* dan menampilkannya sesuai dengan nilai data yang hasilnya melebihi dari *threshold* yaitu dari data 2164 hingga 7601.

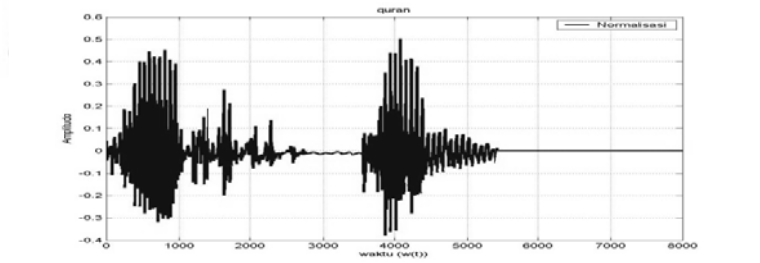


Gambar 9. *Frontend Detection*

3.1.5Normalisasi

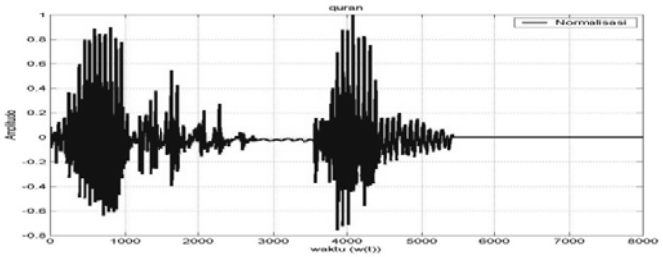
Normalisasi dilakukan untuk mengembalikan jumlah data yang hilang karena melewati proses *frontend detection*. Jumlah data harus dikembalikan ke-8000 karena untuk menyamakan dan memudahkan proses penghitungan dari pola kata yang diucapkan.Sebelumnya dapat dipahami bahwa proses *frontend detection* akan menghasilkan sinyal keluaran dengan nilai baru dari sebuah sinyal pola kata yang dimasukkan. Hasil jumlah datanya-pun berbeda-beda antara pola kata yang satu dengan lainnya, maka apabila hasil seperti ini tetap dibiarkan dan dilanjutkan kedalam proses perhitungan, maka akan menghasilkan perhitungan dengan jumlah *frame* yang berbeda-beda dan rumus yang berbeda pula untuk setiap sinyal masukan.

Normalisasi yang dilakukan disini terbagi menjadi dua bagian, diantaranya (i) normalisasi panjang data, normalisasi ini bertujuan untuk menambahkan jumlah data hingga mencapai jumlah yang sudah ditentukan. Cara kerjanya adalah, sinyal yang melalui proses *frontend detection* pada akhir nilainya ditambahkan beberapa data hingga mencapai panjang 8000 data.



Gambar 10. Normalisasi Panjang Data

Selanjutnya, (ii) normalisasi amplitudo, normalisasi ini bertujuan untuk menyamakan jarak dekat atau jauhnya mulut dengan *microphone* pada saat pengucapan. Cara kerjanya adalah sinyal masukan diperiksa secara keseluruhan untuk mendapatkan nilai maksimumnya. Setelah mendapatkan nilai maksimumnya, maka setiap nilai dari sinyal tersebut dibagi dengan nilai maksimum dari sinyal tersebut. Sehingga disetiap sinyal ucapan yang dimasukkan memiliki tinggi amplitudo sebesar 1 untuk nilai maksimumnya.



Gambar 11. Normalisasi Amplitudo

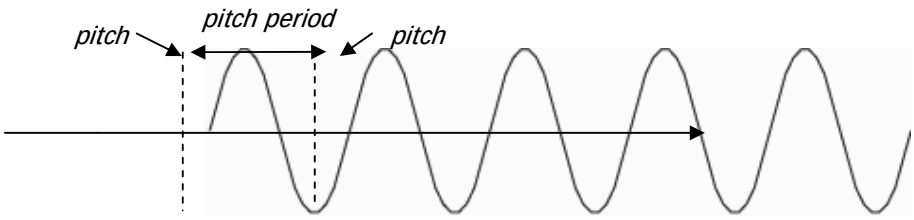
Setelah sinyal masukan melewati keseluruhan proses diatas, maka sinyal tersebut sudah siap untuk didapatkan fitur cirinya, fungsi keseluruhan *preprocessing* ini adalah untuk menghasilkan sebuah sinyal keluaran suara yang memiliki nilai merata dalam jumlah sinyal yang sama meskipun pola kata yang digunakan berbeda-beda, sehingga penghitungan dapat dilakukan dengan mudah.

3.2 *Pitch* dan *Timbre*

Secara umum terdapat 2 faktor yang berpengaruh membentuk ciri suara manusia yaitu *pitch* dan *timbre* (warna suara). *Pitch* berpengaruh terhadap frekuensi dasar (*fundamental frequency*) yang dimiliki oleh setiap benda yang bergetar dan mengeluarkan bunyi. Timbre merupakan muatan harmonik dari suara yang mempengaruhi karakteristik suara yang membuat kita bisa membedakan antara suara yang satu dengan yang lainnya **(Naotoshi, 2008)**.

3.2.1 *Pitch*

Berbagai macam suara yang dapat didengar manusia merambat melalui udara dan dipantulkan ke segala arah. Salah satu parameter yang dapat digunakan untuk membedakan berbagai jenis suara adalah *pitch* atau frekuensi dasar dari suara tersebut. Perbedaan tinggi – rendah suara berhubungan dengan jarak antar *pitch* pada gelombang *(pitch period)*. Panjang jarak tersebut berpengaruh pada frekuensi. Semakin pendek jarak (rapat) semakin tinggi frekuensi sebaliknya semakin lebar jarak semakin rendah frekuensi. Pada lingkup musik tinggi rendah suara diwakili dengan notasi. Setiap notasi memiliki standar frekuensi dan disimbolkan dengan angka atau huruf. Frekuensi sendiri merupakan banyaknya getaran per detik (misal: dari rapatan gelombang ke rapatan berikutnya) yang biasa dinyatakan dalam satuan Hz.

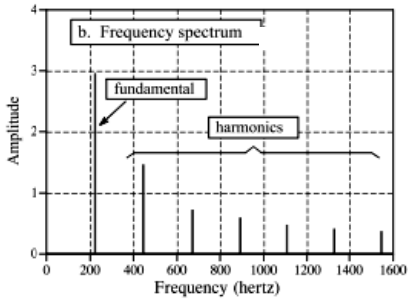


Gambar 12. *Pitch* dan *Pitch Period*

3.2.2 Timbre

Faktor lain yang menjadi ciri suara adalah *timbre*. *Timbre* dapat disebut sebagai kunci inti dari karakter suara manusia. *Timbre* merupakan faktor dari suara yang membuat kita bisa membedakan antara suara yang satu dengan yang lainnya, walaupun *pitch* dan level

kekerasan (*loudness*, dipengaruhi oleh amplitudo) suaranya sama. Sebagai ilustrasi suara yang dihasilkan oleh gitar yang memainkan nada ”A” berbeda dengan suara yang dihasilkan piano walaupun memainkan nada yang sama. Perbedaan karakter suara antara gitar dengan piano disebabkan perbedaan *timbre*. Getaran gelombang suara cukup kompleks, dan biasanya bergetar dalam beberapa frekuensi secara simultan. Inilah sebenarnya yang menyebabkan karakter suara masing-masing benda berbeda dikarenakan “muatan harmonik” *timbre*yang berbeda pula. Gambar 13 merupakan ilustrasi dari sebuah sinyal suara yang memiliki fundamental frekuensi sama dengan muatan harmonik berbeda.



Gambar 13. Muatan Harmonik Pada Domain Frekuensi (Patton, 2014)

3.3 *Frame* Blocking

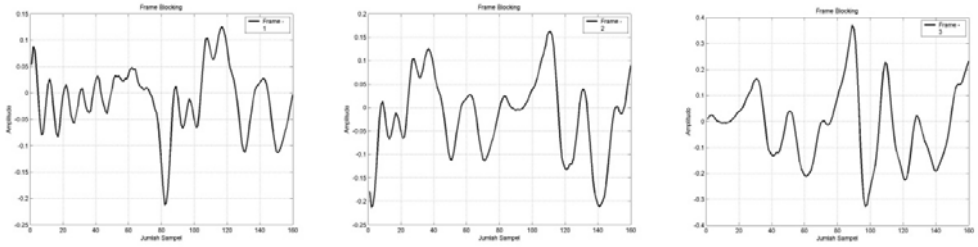
Frame Blocking(framming) adalah proses pembagian suara menjadi beberapa *frame* yang nantinya dapat memudahkan dalam perhitungan dan analisa suara, karena proses analisa akan berhasil dengan baik apabila sinyal yang dianalisa memiliki parameter yang tetap (berubah dengan lambat) terhadap waktu (*time invariant*). *Framming* dibuat sedemikian rupa sehingga sinyal suara dapat dianggap sebagai sinyal yang tidak berubah terhadap waktu.

Pada tahap ini, sinyal ucapan di*blocking* kedalam beberapa *frame*dari keseluruhan sampel N, dan dipisahkan dengan M sampel. *Frame*yang pertama terdiri dari N sampel sinyal pertama dan *frame* yang kedua dimulai M sampel setelah M sampel pertama dimulai (selalu *overlap*). *Frame*yang ketiga dimulai 2M sampel setelah M pertama (M sampel dan *frame* yang kedua). Proses ini berlanjut sampai dengan keseluruhan sampel sinyal terhitung.

Satu *frame* terdiri dari beberapa sampel tergantung tiap berapa detik suara akan disampel dan berapa besar frekuensi *sampling*nya. Pengambilan sampel di penelitian ini dilakukan setiap 20*ms*, dan frekuensi *sampling* yang digunakan sebesar 8000Hz, sedangkan lama rekam selama 1 detik. Parameter yang sering digunakan adalah N, untuk jumlah sampel pada analisis *frame blocking* dan M, untuk jarak antara *frame*satudengan *frame*lainnya.

F_s=8000Hz (berarti 8000 sampel tiap 1 detik)
Disampling tiap 20 ms = 0.02 detik
N =20ms ~ (8000*0.02s) = 160 sampel → (banyak data/*frame*)
M =10ms ~ (8000*0.01s) = 80 sampel
Overlapping = (N-M)=10ms ~ 8000*(N-M)=80 sampel
Panjang data (L) = length(X)
Banyak *Frame* = (L-banyakdata)/overlapping
= (8000 - 160) / 80
= 7840 / 80 = 98 *frame*

Mengikuti penghitungan diatas, maka setiap sinyal ucapan yang dimasukkan memiliki jumlah *frame* sebanyak 98 buah dan 160 data sampel per-*frame*-nya. Oleh karena itu kenapa proses *preprocessing* dibutuhkan, hal itu untuk menjawab dalam pemakaian rumus yang sudah ditentukan pada proses *framming*, sehingga sinyal yang dimasukkan akan menghasilkan jumlah *frame* yang sama.



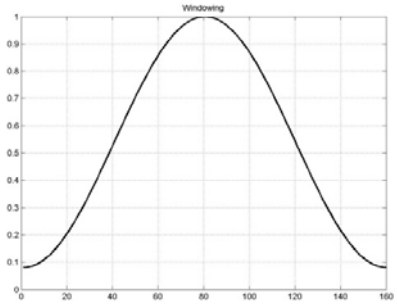
Gambar 14. Sinyal *Framming*

3.4 Windowing

Proses *windowing*adalah suatu proses *weighting* yang berfungsi untuk mengurangi efek diskontinuitas pada ujung-ujung *frame* yang dihasilkan oleh proses *framming*. Berikut adalah blok diagram dari sebuah proses *windowing* terhadap keluaran proses *framming*.Fungsi *windowing* yang digunakan pada penelitian ini adalah *window hamming*, yang mempunyai persamaan seperti berikut:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2 \times phi \times n}{(n-1)}\right), 0 \le n \le (n-1) \quad (4)$$

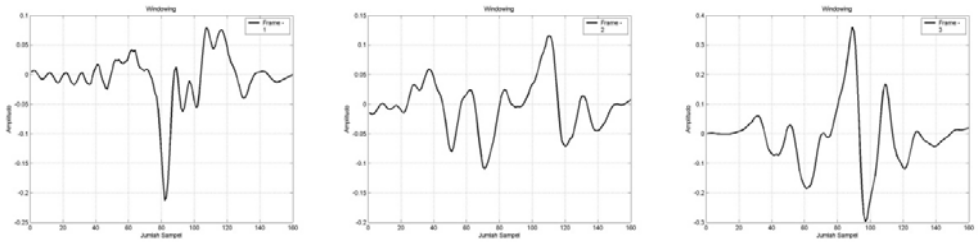
Nilai (n) dalam *windowing* yang digunakan adalah sebanyak 160, karena sampel data pada setiap *frame*adalah 160 sampel. Berikut ini adalah nilai dan tampilan dari fungsi *windowing* dengan jumlah (n) = 160.



Gambar 15. *Windowing*(n) = 160

Tiap *frame* sinyal hasil *frame blocking* dikalikan dengan fungsi *window*:

$$x(n) = x_i(n) \times w(n), 0 \le n \le (n-1) \quad (5)$$



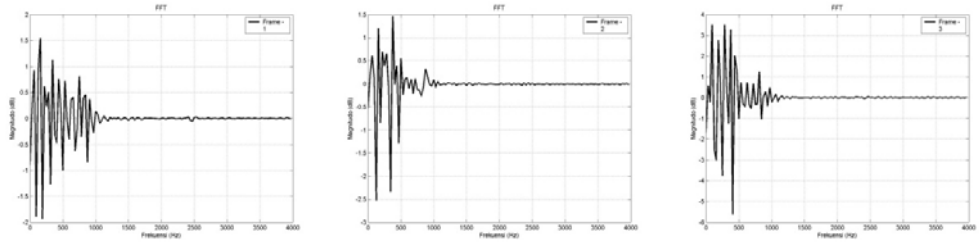
Gambar 16. *Frame Windowing*

3.5 FFT

Fast Fourier Transform (FFT)adalah suatu metode yang efisien untuk pengolahan sinyal, FFT ini digunakan untuk menyederhanakan komputasi data, dan hasil keluaran dari proses ini adalah sinyal *spectrum*. Penelitian ini digunakan FFT 256 titik karena data yang akan diproses sebanyak 160 buah data. Hasil sinyal yang di FFT merupakan suatu sinyal yang *simetris* atau menghasilkan bentuk pencerminan dari sisi sebelah kiri terhadap sisi sebelah kanannya, sehingga dari data sebanyak 256 data hanya diambil sebanyak 128. Gambar 17 adalahproses FFT.

Gambar 17. Proses FFT

Frame sinyal hasil *windowing*akan melewati proses fungsi *FFT*. Sinyal yang dihasilkan proses ini akan sulit untuk dilihat dengan penglihatan, karena nilai hasil *FFT* terdapat nilai dalam bentuk *imajiner*, sehingga sulit untuk ditampilkan dalam bentuk gambar. Gambar 18 adalah bentuk imajiner dari FFT.

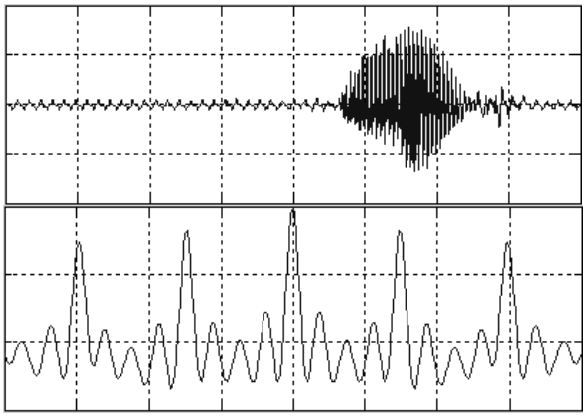


Gambar 18. Sinyal Imajiner FFT

3.6 *Autocorrelation*

Autocorrelation merupakan *cross-correlation* dari suatu sinyal kepada sinyal itu sendiri. Pada pemrosesan sinyal *cross-correlation* merupakan metode pengukuran dari 2 gelombang sinyal sebagai fungsi jarak waktu antara 2 sinyal tersebut. *Autocorrelation*dapat digunakan untuk mendeteksi *pitch* (fundamental frekuensi) pada suatu sinyal *periodic*.

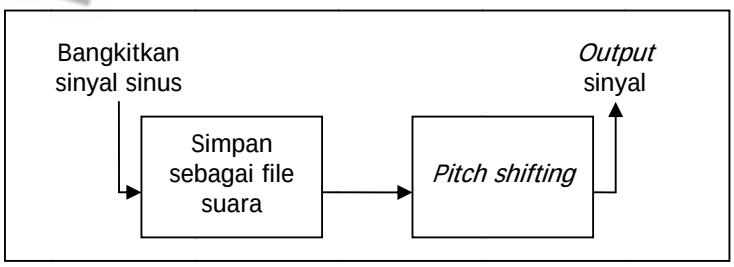
Sebuah cara untuk mendeteksi *pitch* pada sinyal suara diperlukan suatu nilai autokorelasi. Nilai autokorelasi suatu sinyal suara akan menunjukkan bagaimana bentuk gelombang itu membentuk korelasi dengan diri sendiri sebagai fungsi perubahan waktu. Bentuk yang mirip (memiliki korelasi) pada setiap *lag* waktu tertentu menunjukkan perulangan bentuk (periodik) pola sinyal suara. Berdasarkan pola tersebut nantinya akan didapatkan nilai estimasi dari *pitch* (fundamental frekuensi).



Gambar 19. Autokorelasi

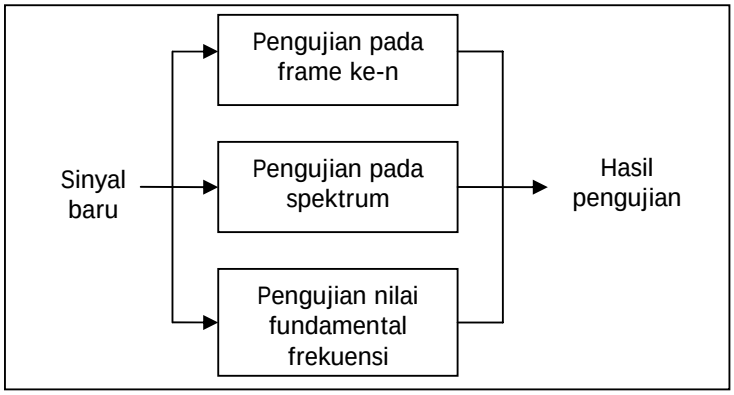
3. PENGUJIAN

Penggunaan sinyal sinus bertujuan memudahkan pengamatan terhadap hasil dari *pitchshifting*. Sinyal sinus lebih mudah diamati dari pada sinyal suara karena sinyal sinus selalu bersifat periodik terhadap domain waktu.Percobaan pertama akan dibangkitkan sinyal sinus dengan frekuensi sampling 8000 Hz durasi 1 detik dan frekuensi 100 Hz. Setelah dibangkitkan data sinyal sinus akan disimpan dengan ekstensi *.wav*. Penggunaan ekstensi *.wav*dilakukan agar sinyal tersebut dapat digunakan untuk pengujian dengan suara. Selain itu, jika disimpan sebagai file suara, sinyal sinus tersebut akan lebih mudah digunakan kembali jika memang diperlukan.



Gambar 20. Pengujian Aplikasi Menggunakan Sinyal Sinus

Gambar 20 menunjukan alur melakukan *pitch shifting*. Seperti yang sudah dijelaskan, hal yang pertama dilakukan adalah membangkitkan sinyal sinus dan menyimpannya dalam bentuk file suara. Sinyal tersebut kemudian dikomputasi dengan fungsi PSOLAdan kemudian menghasilkan *output* sinyal sinus baru yang *pitch*-nya sudah bergeser, atau dengan kata lain frekuensinya sudah berubah sesuai dengan skala yang ditentukan.



Gambar 21. Pengujian Sinyal Sinus

Pengujian hasil *pitch shifting* dari sinyal sinus dilakukan dengan 3 cara: (1) membandingkan bentuk sinyal *input* dengan sinyal *output* pada domain waktu. Pengamatan dilakukan pada *frame* tertentu yang sudah dilakukan. Dari pengamatan terhadap sinyal dapat dilihat pergeseran *pitch*; (2) membandingkan bentuk sinyal *input* dengan sinyal *output* pada domain frekuensi (spektrum). Dari pengamatan terhadap sinyal dapat dilihat pergeseran nilai frekuensi dasar pada domain frekuensi; (3) membandingkan nilai dari frekuensi dasar.

4.1 Pembangkitan Sinyal Sinus

Membangkitkan sinyal sinus yang dikembangkandengan *tools* matlab sebagai berikut:

```
Fs=8000;
t=(1:8000)/Fs;
f=100;
x=sin(2*pi*f*t);
wavwrite(x, Fs, 'sinfs8000f100.wav');
```

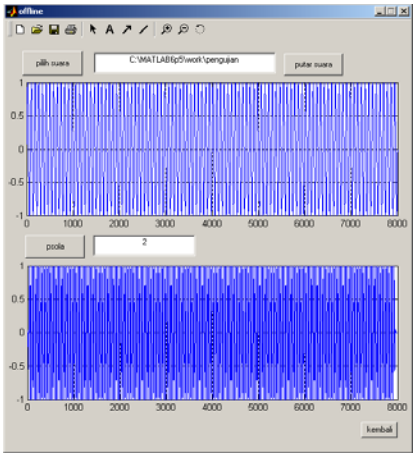
Bila dijalankan fungsi di atas akan membangkitkan sinyal sinus dengan frekuensi sampling 8000 Hz frekuensi 100 Hz dan durasi 1 detik. Setelah itu fungsi akan mengkonversi sinyal tersebut menjadi file suara dengan nama file `'sinfs8000f100.wav'.`

4.2 Pengujian Sinyal

Menguji kinerja aplikasi, sinyal sinus yang akan diujikan akan di *Shift Up* dan *Shift Down*. *Shift Up* bertujuan untuk meningkatkan nilai frekuensi. Hasil dari *Shift Up* akan membuat jarak antar *pitch* semakin rapat dengan kata lain semakin tinggi frekuensi. *Shift Down* bertujuan untuk menurunkan nilai frekuensi. Hasil dari *Shift Down* akan membuat jarak antar *pitch* semakin lebar dengan kata lain semakin rendah frekuensi. Pengujian kali ini skala untuk *Shift Up* adalah 2 dan skala untuk *Shift Down*adalah 0.5.

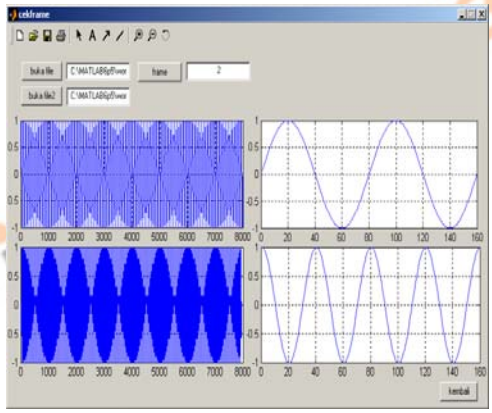
4.2.1 *Shift Up*

Percobaan ini menggunakan menu *offline*. Langkah pertama yang dilakukan adalah memanggil file `'sinfs8000f100.wav'` untuk kemudian dilakukan *Shift Up*. Isikan nilai skala dengan 2 dan tekan tombol PSOLA pada layar menu untuk memulai proses.



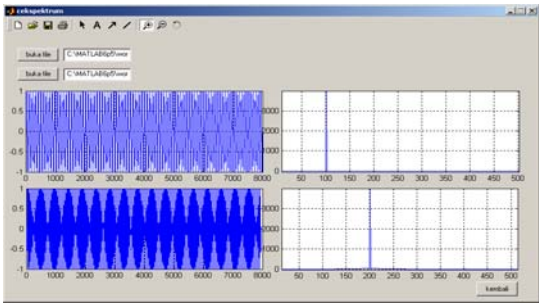
Gambar 22. Perubahan Sinyal Setelah *Shift Up*

Dari Percobaan di atas didapat sinyal hasil dari *Shift Up* dengan skala perubahan 2. jika dilihat dengan kasat mata sinyal *output* tampak lebih rapat dibanding sinyal awal. Untuk Melakukan pengujian lebih dalam sinyal akan diamati pada *frame* ke-n, domain frekuensi, dan dicari nilai fundamental frekuensinya.



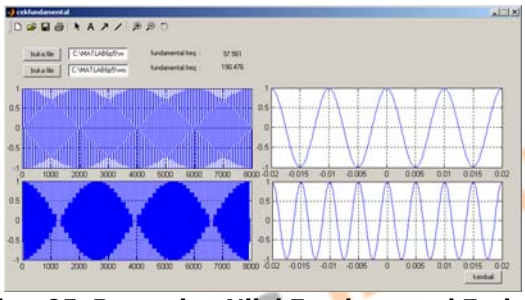
Gambar 23. Perubahan Sinyal *Frame* ke-2 Hasil *Shift Up*

Gambar 23adalah hasil pengamatan sinyal pada *frame* ke-2. Pada sinyal asal dapat dilihat bahwa pada *frame* tersebut terjadi 2 kali getaran, sedangkan pada sinyal hasil terjadi 4 kali getaran. Dari percobaan ini dapat dilihat bahwa *pitch* pada sinyal *output* telah berubah sehingga terjadi pergeseran *pitch* yang menyebabkan *wavelength*berubah menjadi lebih pendek. Hal ini akan berdampak pada nilai frekuensi. Untuk melihat perubahan frekuensi, sinyal harus dilihat pada spektrumnya (domain frekuensi).



Gambar 24. Pergeseran Nilai Frekuensi Hasil *Shift Up*

Gambar 24adalah perbedaan frekuensi sinyal asal dan sinyal hasil. Pada sinyal awal menunjukkan bahwa frekuensi yang memiliki *magnitude* terbesar berada pada indek ke-100, sedangkan pada sinyal hasil menunjukkan bahwa frekuensi yang memiliki *magnitude* terbesar berada pada indek ke-200.

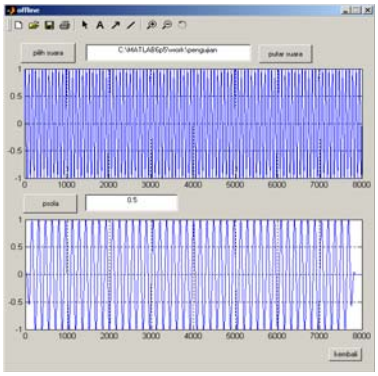


Gambar 25. Pencarian Nilai Fundamental Frekuensi

Percobaan terakhir menunjukkan nilai fundamental frekuensi dari sinyal awal adalah 97.561 dan untuk sinyal hasil 190.476.

4.2.2 *Shift Down*

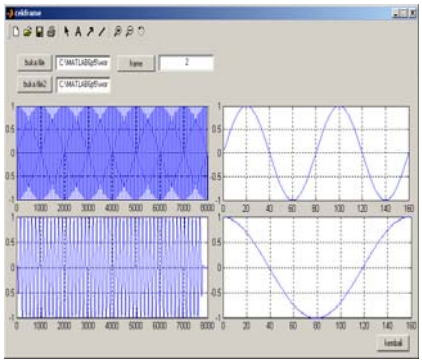
Langkah yang dilakukan untuk percobaan ini sama dengan percobaan *Shift Up*, menggunakan menu *offline*. Langkah pertama yang dilakukan adalah memanggil file `'sinfs8000f100.wav'` untuk kemudian dilakukan *Shift Down*. Isikan nilai skala dengan 0.5 dan tekan tombol PSOLA pada layar menu untuk memulai proses.



Gambar 26. Perubahan Sinyal Setelah *Shift Down*

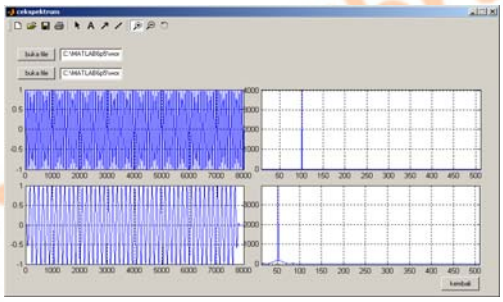
Dari Percobaan di atas didapat sinyal hasil dari *Shift Down* dengan skala perubahan 0.5. jika dilihat dengan kasat mata sinyal *output* tampak lebih renggang dibanding sinyal awal. Untuk

Melakukan pengujian lebih dalam sinyal akan diamati pada *frame* ke-n, domain frekuensi, dan dicari nilai fundamental frekuensinya.



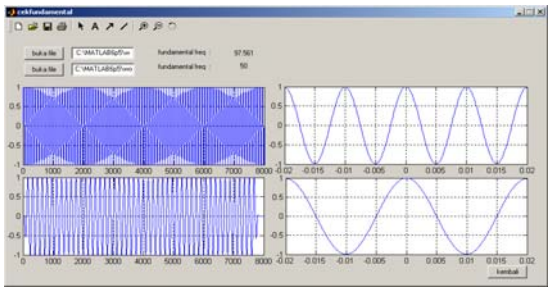
Gambar 27. Perubahan Sinyal Hasil *Shift Down*

Gambar 27adalah hasil pengamatan sinyal pada *frame* ke-2. Pada sinyal asal dapat dilihat bahwa pada *frame* tersebut terjadi 2 kali getaran, sedangkan pada sinyal hasil terjadi hanya 1 kali getaran. Dari percobaan ini dapat dilihat bahwa *pitch* pada sinyal *output* telah berubah sehingga terjadi pergeseran *pitch* yang menyebabkan *wavelength*berubah menjadi lebih panjang. Hal ini akan berdampak pada nilai frekuensi. Untuk melihat perubahan frekuensi, sinyal harus dilihat pada spektrumnya (domain frekuensi).



Gambar 28. Pergeseran Nilai Frekuensi Hasil *Shift Down*

Gambar 28 adalah perbedaan frekuensi sinyal asal dan sinyal hasil. Pada sinyal awal menunjukkan bahwa frekuensi yang memiliki magnitude terbesar berada pada indek 100, sedangkan pada sinyal hasil menunjukkan bahwa frekuensi yang memiliki *magnitude* terbesar berada pada indek 50.



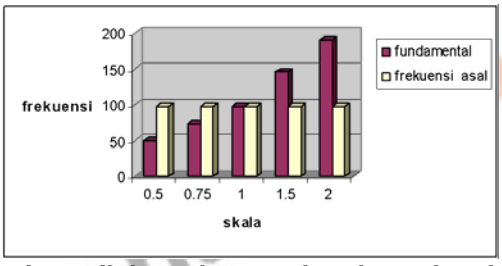
Gambar 29. Pencarian Nilai Fundamental Frekuensi

Percobaan terakhir menunjukkan nilai fundamental frekuensi dari sinyal awal adalah 97.561 dan sinyal hasil adalah 50. Percobaan pengujian aplikasi terhadap sinyal sinus didapatkan tabel sebagai berikut:

Tabel 1. Pengujian Aplikasi Terhadap Sinyal Sinus

skala	fundamental	frekuensi asal	realisasi perubahan skala	akurasi(%)
2	190.476	97.561	1.952378512	97.61893
1.5	145.455	97.561	1.490913377	99.39423
1	97.561	97.561	1	100
0.75	73.4	97.561	0.752349812	99.68767
0.5	50	97.561	0.512499872	97.561
			rata - rata akurasi	98.85236

Tabel 1 berisi data hasil percobaan perubahan skala untuk *pitch shifting* dan efeknya terhadap perubahan frekuensi. Jika dilihat dari data tabel 1, perubahan frekuensi relatif berbanding lurus dengan nilai dari skala perubahan yang ditentukan. Walaupun tidak benar 100% akurat (rata–rata nilai akurasi adalah 98%) namun realisasi perubahan skala sudah mendekati nilai dari skala perubahan yang diinginkan. Dengan kata lain aplikasi berjalan sesuai dengan yang diinginkan pada percobaah dengan menggunakan sinyal sinus.



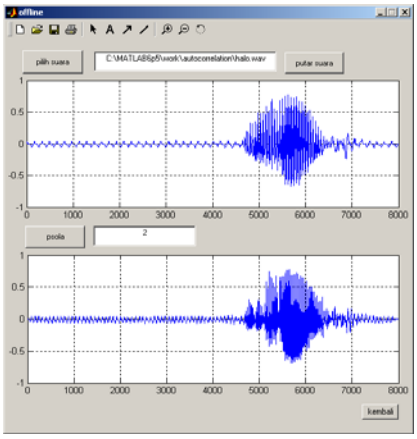
Gambar 30. Grafik Perubahan Nilai Fundamental Frekuensi Pada Pengujian Sinyal Sinus

4.3 Pengujian Menggunakan Sinyal Suara

Pengujian dengan sinyal suara akan memanfaatkan file suara dengan frekuensi sampling 8000 Hz dan durasi 1 detik. File suara yang diujikan berisi file pengucapan kata /halo/. Sama halnya dengan pengujian sinyal sinus, Pengujian hasil *pitch* shifting dari sinyal suara dilakukan dengan 3 cara: (1) membandingkan bentuk sinyal *input* dengan sinyal *output* pada domain waktu. Pengamatan dilakukan pada *frame* tertentu yang sudah dilakukan. Dari pengamatan terhadap sinyal dapat dilihat pergeseran *pitch*; (2) membandingkan bentuk sinyal *input* dengan sinyal *output* pada domain frekuensi (spektrum). Pengamatan terhadap sinyal dapat dilihat pergeseran nilai frekuensi dasar pada domain frekuensi; (3) membandingkan nilai dari frekuensi dasar.

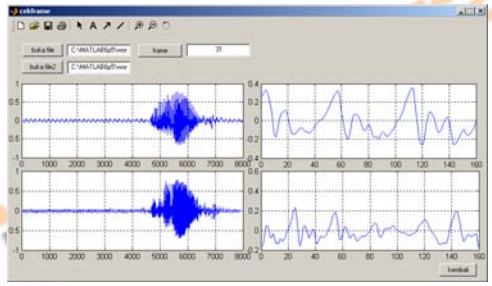
4.3.1 Pengujian *Shift Up*

Langkah yang dilakukan untuk percobaan ini sama dengan percobaan *Shift Up dan Shift Down*pada sinyal sinus, menggunakan menu *offline*. Langkah pertama yang dilakukan adalah memanggil file `'halo.wav'` untuk kemudian dilakukan *Shift Down*. Isikan nilai skala dengan 2 dan tekan tombol PSOLA pada layar menu untuk memulai proses. Hasil dari percobaan ini akan menampilkan perbedaan bentuk sinyal awal dengan sinyal hasil. Dengan skala perubahan 2 seharusnya bentuk sinyal hasil terlihat lebih rapat daripada bentuk sinyal asalnya.



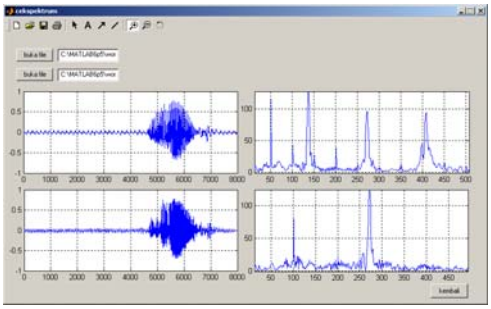
Gambar 31. Percobaan *Shift Up* Pada Sinyal Suara

Dari percobaan di atas didapat sinyal hasil dari *Shift Up* dengan skala perubahan 2. jika dilihat dengan kasat mata sinyal *output* tampak lebih rapat dibanding sinyal awal. Hal ini menunjukkan bahwa aplikasi telah berhasil mengubah *pitch* dari sinyal awal. Namun masih diperlukan lagi pengujian lebih lanjut untuk mengetahui tingkat akurasi aplikasi terhadap sinyal suara. Untuk Melakukan pengujian lebih dalam sinyal akan diamati pada *frame* ke-n, domain frekuensi, dan dicari nilai fundamental frekuensinya.



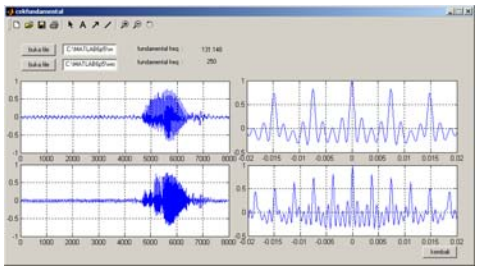
Gambar 32. Perubahan Sinyal *Frame* Hasil *Shift Up* dengan Skala 2

Gambar 31adalah hasil pengamatan sinyal pada *frame* ke-31. Jika dibandingkan dengan hasil pengujian pada sinyal sinus, pengamatan perubahan sinyal pada sinyal suara lebih sulit untuk dilihat. Pengujian pada sinyal sinus lebih mudah diamati karena sinyal sinus selalu periodik terhadap domain waktu.



Gambar 33. Pergeseran Nilai Frekuensi *Shift Up* dengan Skala 2

Pada domain frekuensi perbedaan antara sinya asal dan sinyal hasil relatif lebih mudah diamati. Gambar di atas menunjukkan perbedaan frekuensi sinyal asal dan sinyal hasil. Pada sinyal awal menunjukkan bahwa frekuensi yang memiliki *magnitude* terbesar berada pada kisaran indek 130-140, sedangkan pada sinyal hasil menunjukkan bahwa frekuensi yang memiliki magnitude terbesar berada pada kisaran indek 260-280.

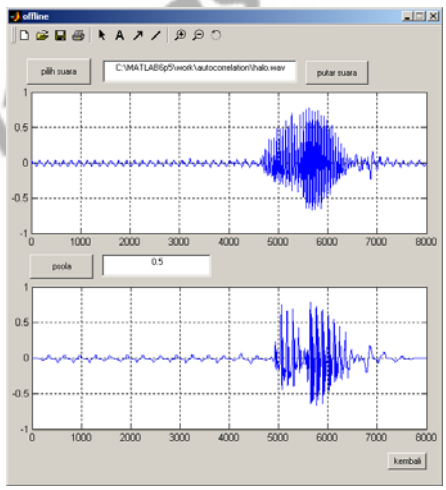


Gambar 34. Pencarian Nilai Fundamental Frekuensi

Percobaan terakhir menunjukkan nilai fundamental frekuensi dari sinyal awal diperkirakan berada pada nilai 131.148 dan untuk nilai fundamental frekuensi dari sinyal hasil diperkirakan berada pada nilai 250.

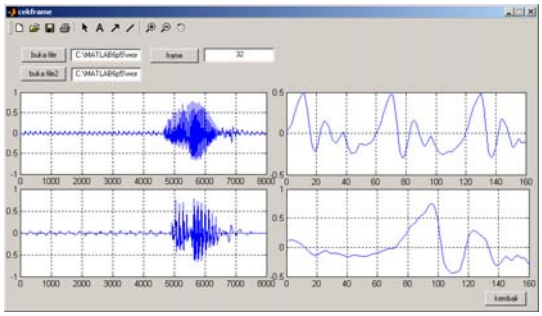
4.3.2 Pengujian *Shift Down*

Langkah yang dilakukan untuk percobaan ini sama dengan percobaan *Shift Up dan Shift Down*pada sinyal sinus, menggunakan menu *offline*. Langkah pertama yang dilakukan adalah memanggil file `'halo.wav'` untuk kemudian dilakukan *Shift Down*. Nilai di isikan dengan skala 0.5.



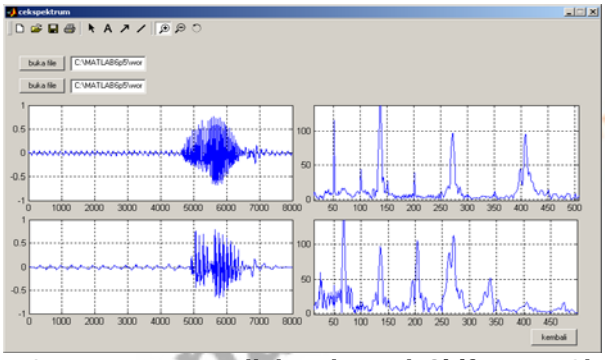
Gambar 35. Percobaan *Shift Down* Sinyal Suara

Percobaan di atas didapat sinyal hasil dari *Shift Up* dengan skala perubahan 0.5. jika dilihat dengan kasat mata sinyal *output* tampak lebih renggang dibanding sinyal awal. Hal ini menunjukkan bahwa aplikasi telah berhasil mengubah *pitch* dari sinyal awal. Namun masih diperlukan lagi pengujian lebih lanjut untuk mengetahui tingkat akurasi aplikasi terhadap sinyal suara. Melakukan pengujian lebih dalam sinyal akan diamati pada *frame* ke-n, domain frekuensi, dan dicari nilai fundamental frekuensinya.



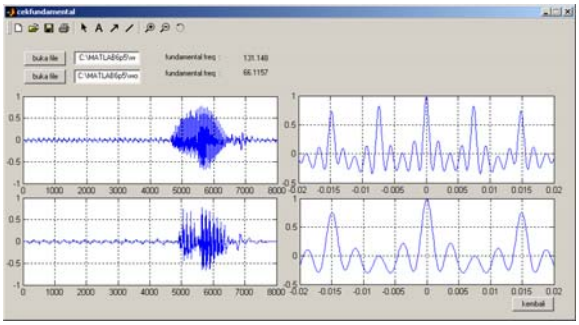
Gambar 36. Perubahan Sinyal Hasil *Shift Down* Skala 0.5

Gambar 35 adalah hasil pengamatan sinyal pada *frame* ke-31. Jika dibandingkan dengan hasil pengujian pada sinyal sinus, pengamatan perubahan sinyal pada sinyal suara lebih sulit untuk dilihat. Pengujian pada sinyal sinus lebih mudah diamati karena sinyal sinus selalu periodik terhadap domain waktu.



Gambar 37. Pergeseran Nilai Frekuensi *Shift Down* Skala 0.5

Pada domain frekuensi perbedaan antara sinya asal dan sinyal hasil relatif lebih mudah diamati. Gambar di atas menunjukkan perbedaan frekuensi sinyal asal dan sinyal hasil. Pada sinyal awal menunjukkan bahwa frekuensi yang memiliki magnitude terbesar berada pada kisaran indek 130-140, sedangkan pada sinyal hasil menunjukkan bahwa frekuensi yang memiliki magnitude terbesar berada pada kisaran indek 60-70.



Gambar 38. Pencarian Nilai Fundamental Frekuensi

Percobaan terakhir menunjukkan nilai fundamental frekuensi dari sinyal awal diperkirakan berada pada nilai 131.148 dan untuk nilai fundamental frekuensi dari sinyal hasil diperkirakan berada pada nilai 66.1157.

Percobaan di atas didapatkan data – data yang akan menunjang proses analisa kinerja dari aplikasi. Data tersebut kemudian akan ditabulasi dan dikelompokkan berdasarkan pengujian pada sinyal sinus dan pada sinyal suara. Analisa ditekankan pada akurasi pengaruh skala pergeseran *pitch* terhadap fundamental frekuensi. Berikut rumus penghitungan akurasi.

$$\text{realisasi skala} = \frac{f_0 output}{f_0 input}, \quad \text{akurasi} = \frac{realisasi\_skala}{input\_skala} x\ 100\% \quad (6)$$

dimana

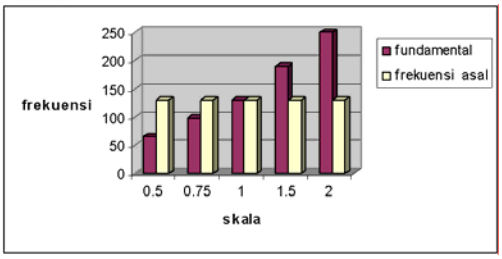
f_0 = fundamental frekuensi

Percobaan pengujian aplikasi terhadap sinyal suara didapatkan 5 tabel data pengujian sinyal suara ”halo”, ”selamat pagi”, ”apa kabar”, ”baik sekali”, dan ”sampai jumpa”. Tabel 2 hasil pengujian terhadap sinyal suara ucapan halo.

Tabel 2. Pengujian Aplikasi Terhadap Sinyal Suara Ucapan “Halo”

skala	fundamental	frekuensi asal	realisasi Perubahan skala	akurasi(%)
2	250	131.148	1.906243328	95.31217
1.5	190.476	131.148	1.452374417	96.82496
1	131.148	131.148	1	100
0.75	98.77	131.148	0.753118614	99.58591
0.5	66.1157	131.148	0.504130448	99.18068
			rata – rata akurasi	98.18074

Percobaan tabel 2, hasil yang ditampilkan tidak jauh berbeda dengan hasil dari percobaan pitch shifting terhadap sinyal sinus. Perubahan frekuensi relatif berbanding lurus dengan nilai dari skala perubahan yang ditentukan. Hal ini menunjukkan bahwa aplikasi yang dibangun berhasil melakukan perubahan pitch sesuai dengan skala yang ditentukan (rata–rata akurasi 98%). Semakin tinggi nilai skala pergeseran semakin besar pula nilai frekuensi dasar, sebaliknya semakin rendah nilai pergeseran semakin rendah pula nilai frekuensi dasar.



Gambar 39. Grafik Perubahan Nilai Fundamental Frekuensi Pada Pengujian Sinyal Suara Ucapan “Halo”

5. KESIMPULAN

Perubahan sinyal sinus frekuensi relatif berbanding lurus dengan nilai dari skala perubahan yang ditentukan. Walaupun tidak benar-benar akurat (rata –rata akurasi 98%) namun realisasi perubahan skala sudah mendekati nilai dari skala perubahan yang diinginkan,dengan kata lain aplikasi berjalan sesuai dengan yang diinginkan pada percobaan dengan menggunakan sinyal sinus. Sedangkan perubahan sinyal suara frekuensi relatif berbanding lurus dengan nilai dari skala perubahan yang ditentukan. Hal ini menunjukkan bahwa aplikasi yang dibangun berhasil melakukan perubahan *pitch* sesuai dengan skala yang

ditentukan (rata–rata akurasi 98%). Semakin tinggi nilai skala pergeseran semakin besar pula nilai frekuensi dasar, sebaliknya semakin rendah nilai pergeseran semakin rendah pula nilai frekuensi dasar.

DAFTAR RUJUKAN

Bhaskoro, Susetyo Bagas. (2013).*Cepstrum Parameter for Human Voice Recognition,* Engineering International Conference, Semarang, pp. II115-II119.

Patton, Joshua.(2014).Pitch_Synchronous_Overlap-Add. Dipetik 1 November 2014, dari http://www.researchgate.net/publication/242507840_ELEC_484_Project_-_*Pitch*_Synchronous_Overlap-Add

Kaur, Sukhvinder, Dinesh Arora Dimple Garg. (2012).*Comparative Analysis of Speech Processing Techniques for Gender Recognition,* International Journal of Advances in Electrical and Electronics Engineering, pp. 278-283.

Zulkarnain, Barmawi Andriana. (2013).*Speech Recognition System Based on Linear Predictive Coding (LPC) and Hidden Markov Model (HMM) using Matlab for Speaker Identification*, Engineering International Conference, Semarang, pp. II82-II86.

Shrawankar, Urmila, Rashmi Makhijani. (2011).*Speech Enhancement Using Pitch Detection Approach For Noisy*, International Journal Of Engineering Science And Technology (IJEST), vol. 3, no. 2.

Naotoshi, Seo sonots. (2008).Pitch Detection. Dipetik 20 November 2014,dari *ENEE632 Project 4 Part I: Pitch Detection*. http://note.sonots.com/?plugin=attach&refer=SciSoftware%2FPitch&openfile=pitch.pdf

John G, Dimitris G. Manolakis. Proakis. (1992).*Digital Signal Processing Principles, Algorithms, and Applications*. New York, US: McMillan.