TÌM TẦN SỐ CƠ BẢN CỦA TÍN HIỆU TIẾNG NÓI DÙNG HÀM TỰ TƯƠNG QUAN

Nguyễn Nhật Tùng, Ninh Hải Hoàng

Khoa Công nghệ thông tin, Trường Đại học Bách Khoa, Đại học Đà Nẵng

nhattungnguyen.2kgl@gmail.com, ninhhaihoang@gmail.com

Nhóm 6, lớp HP: 18N15

Điểm	Bảng phân công nhiệm vụ		Chữ ký của SV
	Nguyễn Nhật Tùng (nhóm trưởng)	Viết báo cáo (tr. 1, 3), viết mã cài đặt thuật toán (tr. 9-12), viết báo cáo kết quả thực nghiệm và kết luận (tr. 12-17).	
	Ninh Hải Hoàng	Viết báo cáo (tr. 1, 3), đọc tài liệu, viết báo cáo về cơ sở lý thuyết và tài liệu tham khảo (tr. 3-9)	

Lời cam đoan: Chúng tôi, gồm các sinh viên có chữ ký ở trên, cam đoan rằng báo cáo này là do chúng tôi tự viết dựa trên các tài liệu tham khảo liệt kê ở cuối báo cáo. Các số liệu thực nghiệm và mã nguồn chương trình nếu không chỉ dẫn nguồn tham khảo đều do chúng tôi tự làm. Nếu vi phạm thì chúng tôi xin chịu trách nhiệm và tuân theo xử lý của giáo viên hướng dẫn.

TÓM TẮT— Việc tìm tần số cơ bản là công việc quan trọng trong việc nhận diện giọng nói, xác định ngữ điệu của người nói, là cơ sở để tái tạo lại âm thanh giống với tiếng nói tự nhiên của con người. Sử dùng hàm tự tương quan để tìm tần số cơ bản trên miền thời gian là cách đơn giản và phổ biến, thông dụng để tìm chu kỳ cơ bản của tín hiệu giọng nói, và cũng là cách được sử dụng trong thuật toán được trình bày sau đây. Bài viết này trình bày 2 phiên bản của thuật toán: phiên bản không phân đoạn tiếng nói-khoảng lặng và phiên bản có phân đoạn tiếng nói-khoảng lặng. Kết quả thử nghiệm với 4 mẫu tín hiệu cho thấy các tần số cơ bản được tìm tự động có giá trị gần đúng với tần số cơ bản tìm bởi phần mềm Wave Surfer[1], với sai số trung bình là 22.2770 Hz và phương sai là 14.2337 Hz (đối với phiên bản có phân đoạn tiếng nói-khoảng lặng trên tín hiệu mẫu) hoặc với sai số trung bình là 43.8399 Hz, phương sai 46.5804 Hz (đối với phiên bản không phân đoạn tiếng nói-khoảng lặng). Kết quả đó cũng cho thấy sử dụng phiên bản có phân đoạn tiếng nói-khoảng lặng của thuật toán giúp tìm được tần số cơ bản chính xác hơn so với phiên bản còn lại.

Từ khóa— xử lý tín hiệu số, tự động đo tần số cơ bản, tự động tìm cao độ giọng nói, phân biệt tiếng nói-khoảng lặng, hàm tự tương quan, tín hiệu tiếng nói.

Mục lục

I. ĐẶT VẤN ĐỀ

Tần số cơ bản của tín hiệu tiếng nói đặc trưng cho cao độ của tiếng nói. Trong lĩnh vực xử lý tín hiệu số, nó là nghịch đảo của chu kỳ cơ bản. Trong lĩnh vực âm học, nó là tốc độ rung của dây thanh trong bộ máy phát âm của con người, là các âm tiết trên thang âm nhạc, và còn là một đặc trưng cơ bản cho ngữ điệu của tiếng nói. Việc tìm tần số cơ bản giúp xác định ngữ điệu của người nói, giúp nhận diện giọng nói; là cơ sở để tái tạo lại âm thanh giống với tiếng nói tự nhiên của con người, hỗ trợ việc giao tiếp tự nhiên giữa máy tính với con người và giữa con người với con người.

Việc dùng hàm tự tương quan để tìm tần số cơ bản trên miền thời gian là cách đơn giản và phổ biến, thông dụng để tìm chu kỳ cơ bản của tín hiệu giọng nói, và cũng là cách được sử dụng trong thuật toán được trình bày sau đây. Cụ thể: Thuật toán này chia tín hiệu đầu vào thành khung tín hiệu (có độ dài từ 10ms đến 30ms[2]) và xử lý từng khung tín hiệu. Sau đó thuật toán kiểm tra tín hiệu trong khung có tính tuần hoàn hay không. Nếu có thì thuật toán tính ra tần số cơ bản cho khung tín hiệu đó, ngược lại nếu không tuần hoàn (âm vô thanh) thì tín hiệu trong khung có tần số cơ bản không xác định.

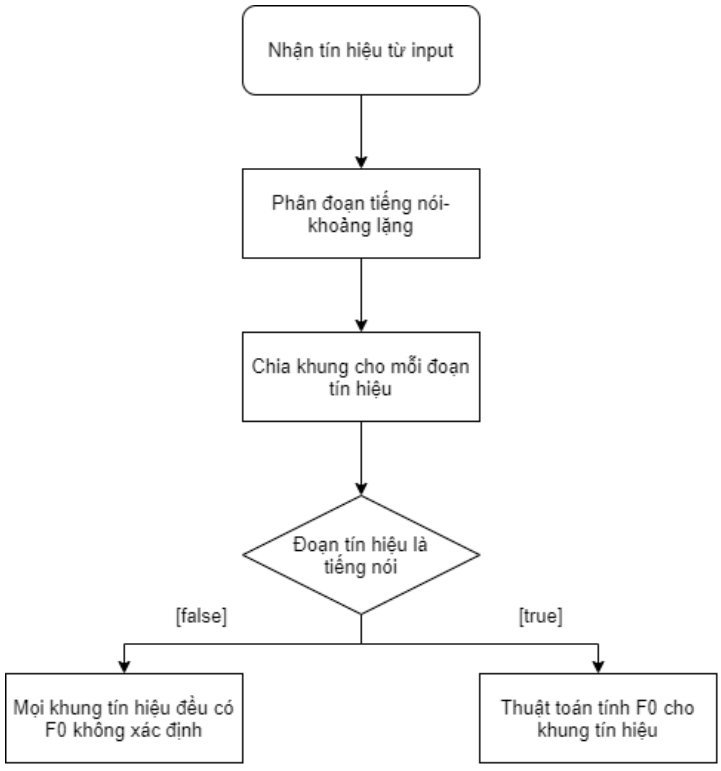
Tuy nhiên tần số cơ bản tìm được bằng thuật toán trên có thể trả về giá trị không chính xác, vì vậy chúng tôi đưa ra thêm thuật toán tương tự nhưng có thực hiện thêm việc phân đoạn tín hiệu nguồn thành tiếng nói-khoảng lặng trước khi tiến hành chia khung và xử lý nhằm giảm sai sót khi tính toán trên các khung tín hiệu không tuần hoàn.

Bài viết này sử dụng phiên bản có phân đoạn tiếng nói-khoảng lặng của thuật toán làm thuật toán chính và có bố cục như sau: Phần II trình bày tổng quan về cơ sở lý thuyết liên quan tới hàm tự tương quan, nguyên lý của các thuật toán, những vấn đề phát sinh trong thuật toán và cách khắc phục. Phần III ghi mã nguồn cách cài đặt thuật toán bằng Matlab. Phần IV trình bày kết quả thu được và các đánh giá kết quả đó khi áp dụng thuật toán lên các dữ liệu mẫu; so sánh thuật toán hiện tại với thuật toán khi không thực hiện chia tiếng nói-khoảng lặng. Cuối cùng là kết luận rút ra được trình bày ở phần V.

II. CƠ SỞ LÝ THUYẾT VÀ CÁC THUẬT TOÁN

A. Sơ đồ khối thuật toán

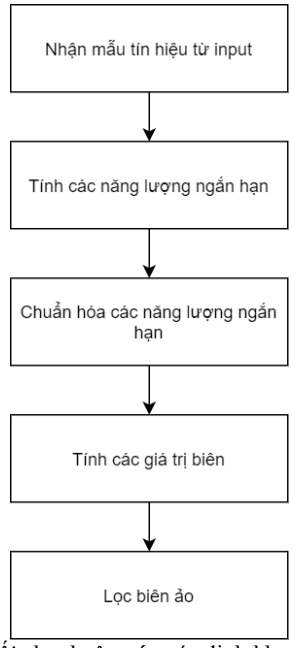
Nội dung thuật toán tìm tần số cơ bản của tín hiệu tiếng nói dùng hàm tự tương quan phiên bản có phân đoạn tiếng nói-khoảng lặng được tóm tắt bằng sơ đồ khối trong hình sau.



Hình 1. Sơ đồ khối cho thuật toán xác định tần số cơ bản.

B. Thuật toán phân đoạn tiếng nói-khoảng lặng

Nội dung thuật toán phân đoạn tiếng nói-khoảng lặng được tóm tắt bằng sơ đồ khối trong hình sau.

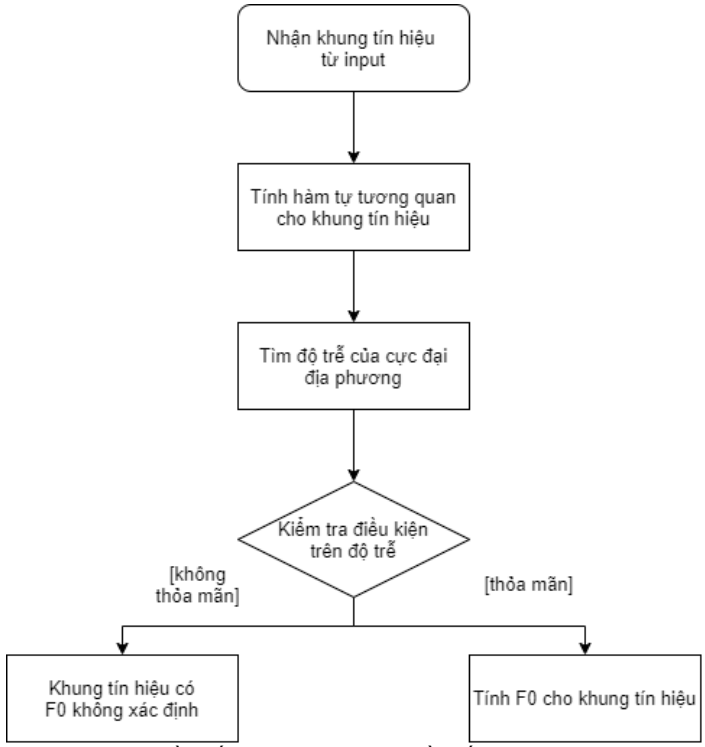


Hình 2. Sơ đồ khối cho thuật toán xác định khoảng lặng-tiếng nói.

Chi tiết thuật toán có thể xem thêm trong tài liệu tham khảo [3] do nhóm chúng tôi thực hiện.

C. Thuật toán tìm tần số cơ bản của tín hiệu

Sau khi chia khung tín hiệu, việc tìm tần số cơ bản của tín hiệu được thực hiện dựa theo sơ đồ khối dưới đây.



Hình 3. Sơ đồ khối cho thuật toán tìm tần số cơ bản của tín hiệu

D. Thuật toán tính hàm tự tương quan

Hàm tự tương quan $xx[\tau]$ của tín hiệu rời rạc $x[n]$ được tính theo công thức:

$$xx[\tau] = \sum_{m=0}^{\infty} x[m]x[m+\tau]$$

(trong đó $\tau \in Z$ là độ trễ (lag) của hàm tự tương quan, có đơn vị là số mẫu)

Vì hàm tự tương quan có tính đối xứng nên thuật toán chỉ lấy các giá trị $\tau \epsilon N$.

Khi xử lý tín hiệu ở khung thứ t ($t \geq 0$) có chiều dài N (mẫu), công thức hàm tự tương quan trở thành:

$$xx_t[\tau] = \sum_{m=t+1}^{t+N} x[m]x[m+\tau]$$

Từ công thức trên có thể suy ra giá trị hàm tự tương quan của một tín hiệu tại độ trễ τ chính là tích vô hướng (dot product) của tín hiệu đó x[n] với chính nó bị dịch phải τ mẫu x[$n+\tau$]:

$$xx_t[\tau] = x[n] \cdot x[n+\tau]$$

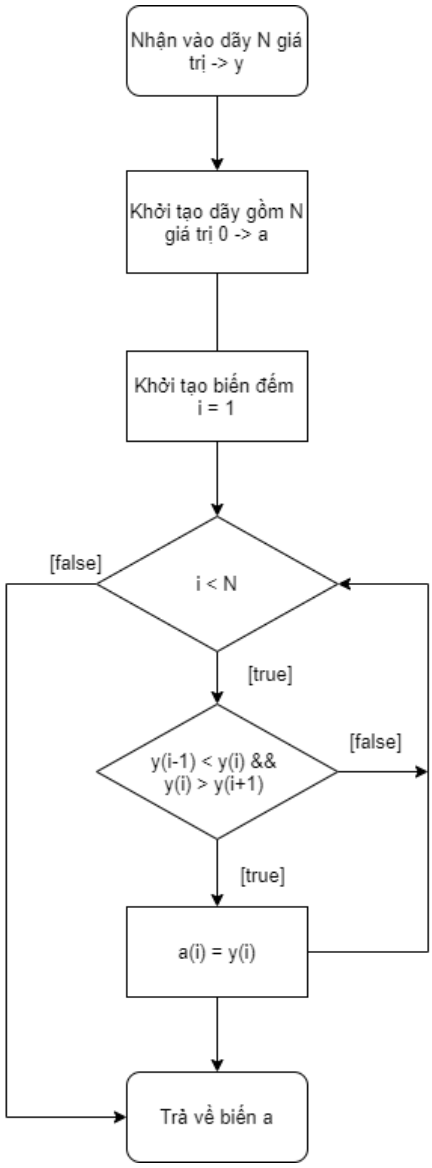
Thuật toán ở đây sử dụng hàm tự tương quan được chuẩn hoá bằng công thức:

$$r_t(\tau) = \frac{xx_t(\tau)}{xx_t(0)}$$

E. Thuật toán tìm độ trễ của cực đại địa phương

Để tìm độ trễ của cực đại địa phương, chúng tôi thực hiện tìm giá trị cực đại địa phương r_{max}, sau đó tìm chỉ số của nó trong khung tín hiệu t_{max} (mẫu). Khi đó độ trễ của cực đại địa phương chính là t_{max}.

Thuật toán tìm cực đại địa phương được thực hiện dựa theo sơ đồ khối sau.



Như vậy cực đại địa phương là các giá trị khác giá trị 0 trong biến trả về.

F. Kiểm tra điều kiện trên độ trễ giá trị cực đại địa phương

Độ trễ giá trị cực đại địa phương t_{max} (mẫu) được tìm ra sau bước tính hàm tự tương quan không thể được dùng để tính tần số cơ bản ngay mà phải được thuật toán kiểm tra bằng 3 điều kiện sau đây:

1. Khung tín hiệu có tuần hoàn hay không.

Để thuật toán phân biệt được khung tín hiệu là âm vô thanh (có thể tính được tần số cơ bản) hay âm vô thanh (tần số cơ bản không xác định), ta tìm ra một giá trị của hàm tự tương quan lấy làm mốc để phân biệt âm hữu thanh và âm vô thanh, ký hiệu là R_0. Cách xác định R_0 được trình bày như sau:

1. Quan sát hàm tự tương quan của khung tín hiệu chứa âm vô thanh để tìm giá trị của cực đại địa phương (R_{0a}). Làm tương tự với khung tín hiệu chứa âm hữu thanh để tìm giá trị của cực đại địa phương (R_{0b}).
2. Tính trung bình cộng của R_{0a} và R_{0b} ta được R_0.
3. Lặp lại bước 1 ở các khung khác nhau và các tín hiệu khác nhau. Qua vài vòng lặp, ta nhận được giá trị R_0 có giá trị gần như không đổi qua mỗi vòng lặp.

Thuật toán được trình bày trong bài sử dụng giá trị R_0=0.27.

Như vậy, ta kiểm tra điều kiện sau có thỏa mãn hay không:

$$r_{max} \geq R_0$$

2. Độ trễ có thuộc vào chu kỳ âm tiếng nói của con người hay không.

Để tránh việc thuật toán tính ra những tần số cơ bản quá cao hoặc quá thấp so với tần số cơ bản mà tiếng nói con người có thể tạo ra, ta quy ước tần số cơ bản của tiếng nói con người có thể tạo ra là từ 70 Hz đến 400 Hz[2] sau đó ta chuyển đổi chúng từ miền tần số sang miền thời gian rời rạc (số mẫu) bằng công thức đã được chứng minh sau:

$$T_{concrete} = \frac{F_{sampling}}{F_{analog}}$$

Trong đó:

- $T_{concrete}$ là chu kỳ trên miền thời gian rời rạc, đơn vị là số mẫu
- $F_{sampling}$ là tần số lấy mẫu, đơn vị là mẫu/giây
- F_{analog} là tần số của tín hiệu liên tục, đơn vị là Hz

Như vậy, ta kiểm tra điều kiện sau có thỏa mãn hay không

$$T_{concrete_{min}} \leq t_{max} \leq T_{concrete_{max}}$$

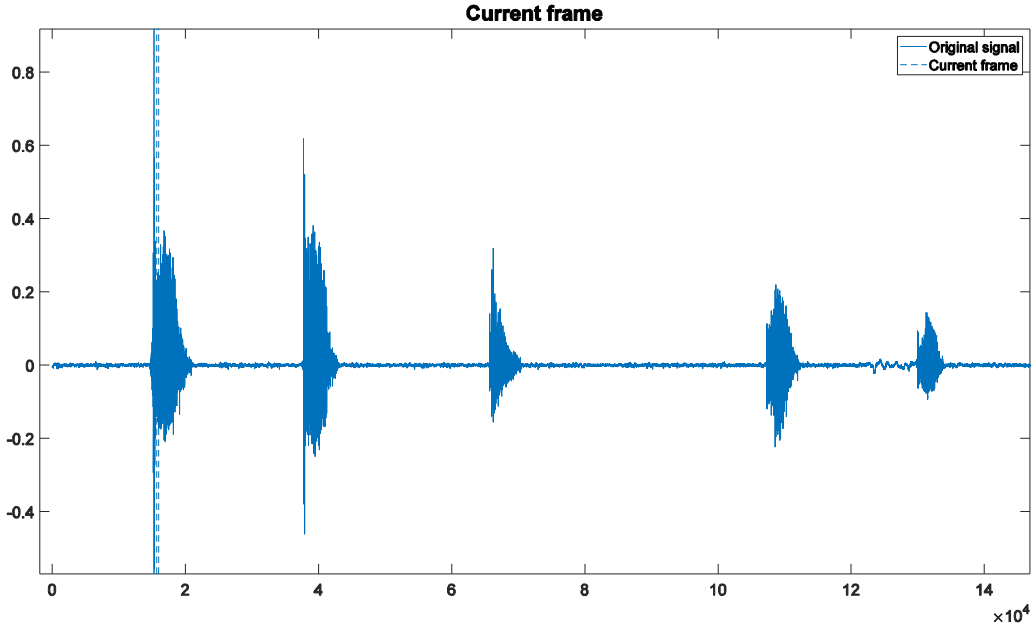
3. Độ trễ có đồng thời là cực đại địa phương trên miền D không.

Sau khi tìm ra điểm cực đại địa phương ở chỉ số t_{max} (mẫu) cho khung tín hiệu có chiều dài N (mẫu), ta mở rộng điều kiện của cực đại địa phương ra miền D: $[local_{left}; local_{right}]$, với $local_{left} = t_{max} - \frac{N}{100}$ và $local_{right} = t_{max} + \frac{N}{100}$. Với điều kiện điểm cực đại địa phương tìm được là đúng trên miền $E \in D$, thuật toán chỉ kiểm tra tính đúng đắn của điểm cực đại địa phương với 2 điểm ở 2 đầu mút của miền D, là $local_{left}$ và $local_{right}$.

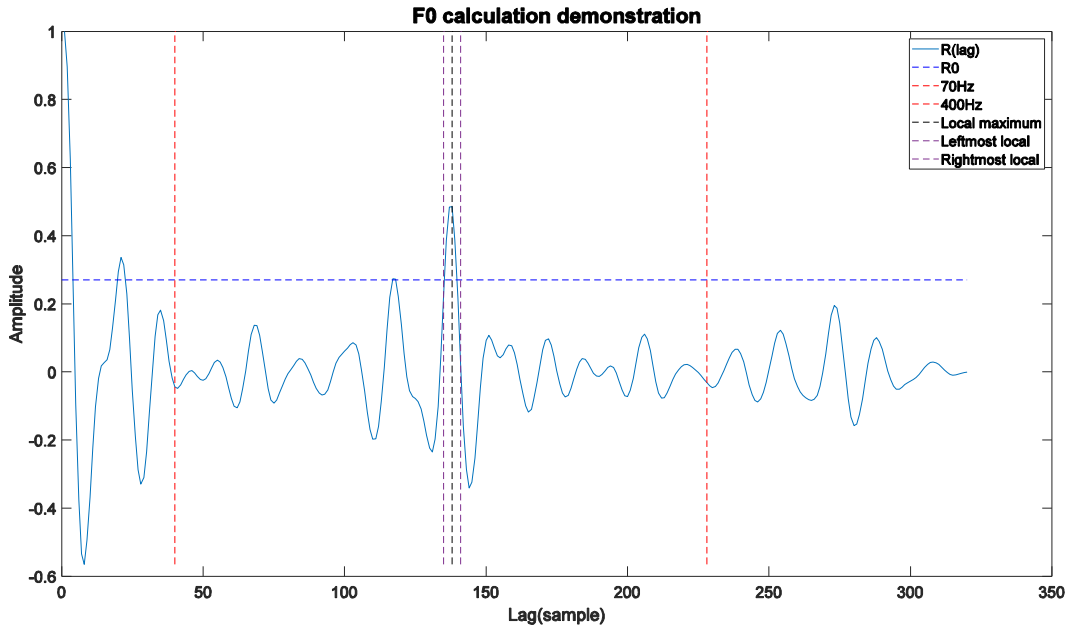
Như vậy, ta kiểm tra điều kiện sau có thỏa mãn hay không:

$$r_{max} \geq \max(r(local_{left}), r(local_{right}))$$

Việc kiểm tra 3 điều kiện trên được minh họa như trong hình bên dưới.



Hình 1. Vị trí của khung đang được xử lý trên tín hiệu nguồn.



Hình 2. Các điều kiện để kiểm tra điểm cực đại địa phương.

Ở trường hợp trong hình trên, thuật toán xác định điểm cực đại tìm được là thỏa mãn điều kiện và chuyển sang bước tính tần số cơ bản cho điểm cực đại này.

G. Tính F0 cho khung tín hiệu

Sau khi đã biết độ trễ của giá trị cực đại địa phương, ta tìm được chu kỳ của tín hiệu trong khung t_{max} (mẫu). Tiếp theo ta chuyển đổi chu kỳ đó từ miền thời gian rời rạc sang miền thời gian liên tục bằng công thức:

$$T_{analog} = \frac{T_{concrete}}{F_{sampling}}$$

Trong đó:

- $T_{concrete}$ là chu kỳ của tín hiệu trong miền thời gian rời rạc, đơn vị là số mẫu;
- $F_{sampling}$ là tần số lấy mẫu, đơn vị là mẫu/giây
- T_{analog} là chu kỳ của tín hiệu trong miền thời gian liên tục, đơn vị là giây.

Với $T_{concrete} = t_{max}$, ta có công thức tìm chu kỳ cơ bản T_0 của tín hiệu trong khung là:

$$T_0 = \frac{t_{max}}{F_{sampling}}$$

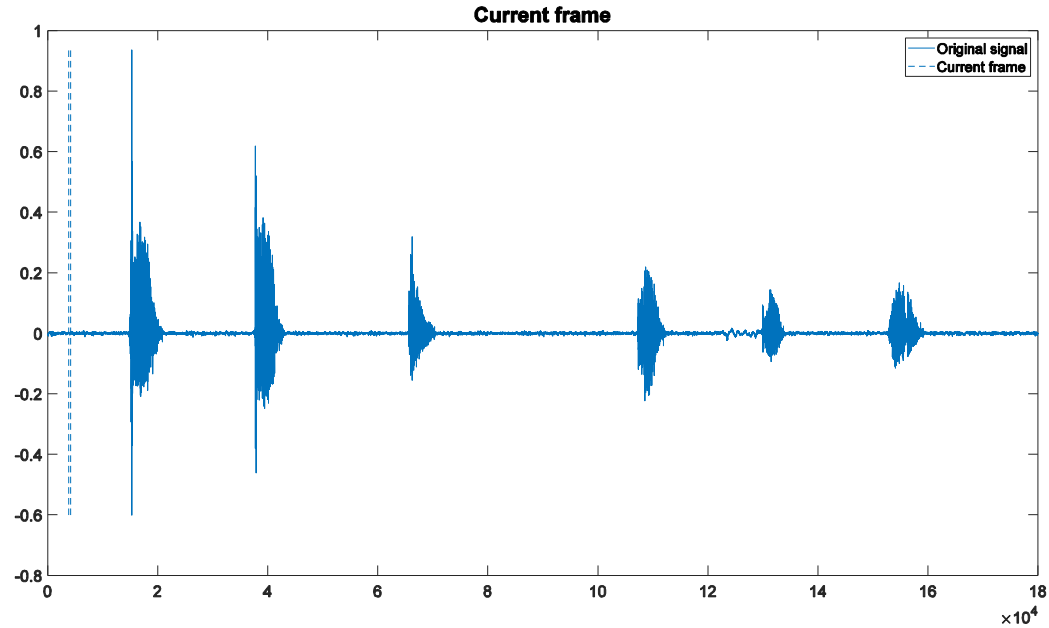
Cuối cùng ta tính tần số cơ bản F_0 của tín hiệu trong khung bằng công thức:

$$F_0 = \frac{1}{T_0}$$

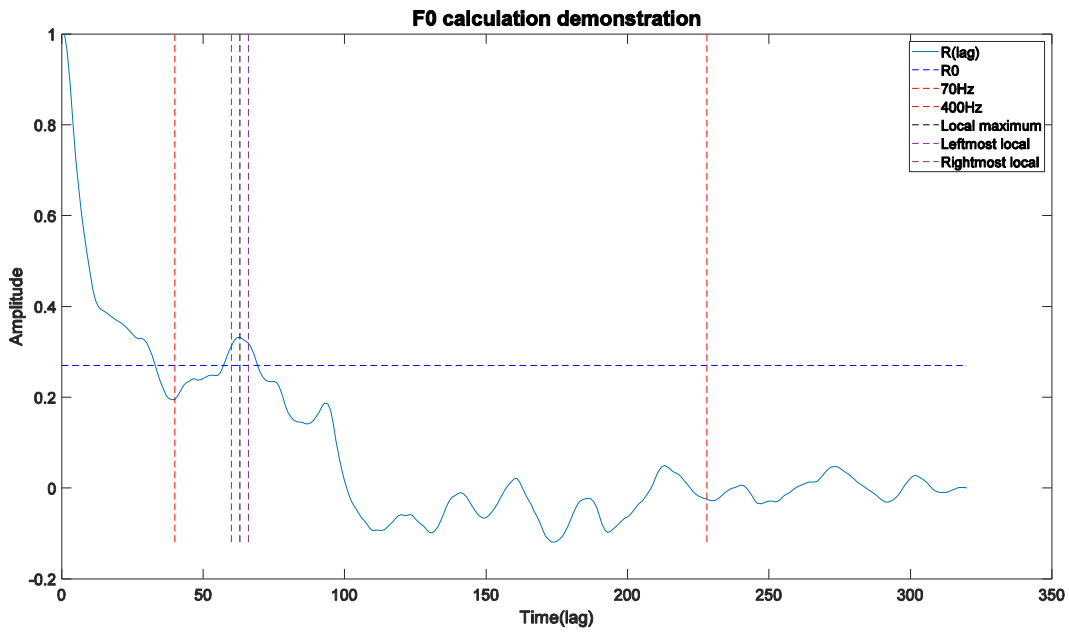
H. Vấn đề với hàm tự tương quan và khung tín hiệu không chứa tiếng nói

1. Vấn đề

Trong một số trường hợp, hàm tự tương quan tính ra các giá trị đặc biệt khiến việc xác định điểm cực đại địa phương thỏa mãn các điều kiện của thuật toán ở phần **I.F**, làm thuật toán xác định cả tần số cơ bản cho khung tín hiệu không chứa tiếng nói như trường hợp trong ảnh sau:



Hình 3. Vị trí của khung đang được xử lý trên tín hiệu nguồn (khung không chứa tiếng nói).



Hình 4. Kết quả hàm tự tương quan làm xuất hiện điểm cực đại địa phương phù hợp mọi điều kiện của thuật toán.

2. Giải pháp đề xuất và kết quả

Để giải quyết vấn đề tìm được cả F0 với âm vô thanh, chúng tôi sử dụng bộ lọc tiếng nói-khoảng lặng để loại bỏ những những khoảng lặng và chỉ xử lý trong các khoảng tiếng nói.

Chi tiết về thuật toán, cơ sở lý thuyết và mã nguồn của bộ lọc được trình bày trong tài liệu tham khảo [3] do chúng tôi thực hiện.

Kết quả của thuật toán sau khi sử dụng bộ lọc phân biệt tiếng nói-khoảng lặng được trình bày trong phần **IV.C** của tài liệu này.

III. MÃ CHƯƠNG TRÌNH CÀI ĐẶT CÁC THUẬT TOÁN

Mã chương trình ở đây viết trên ngôn ngữ Matlab.

A. Thuật toán phân đoạn tiếng nói-khoảng lặng

Ở đây chỉ trình bày hàm tổng quát của thuật toán. Chi tiết về các thuật toán phụ thuộc được trình bày trong tài liệu tham khảo [3] do nhóm thực hiện.

```
function b = svfilter(y, F)
% Silence - Voiced filter: Returns vector of boundaries using [0;1] normalization
%Inputs:    y: Audio signal to find boundaries
%           F: Sampling frequency

flen = 10;  % frame length in ms
cond = 20;  % minimum length (frames) for a span to be silence span
E0=0.55;     % threshold for [0;1] normalization

senergy = seframes(y,F,flen);       % find short-time energy of input signal
nsenergy = datanormalize(senergy);  % normalize short-time energy using standard
distribution
vb = svboundaries(nsenergy, E0);    % find silence - voiced boundaries
b = vbfilter(vb, cond);             % filter out virtual boundaries
b=b*flen*F/1000;    % convert frames to samples
b(1)=1;             % reset first boundary to first sample
b(length(b))=length(y);  % reset last boundary to last sample
end
```

B. Thuật toán tính hàm tự tương quan

```
function R = autocorrel(x)
% Return the normalized autocorrelation of signal x
N=length(x);    % length (samples) of input signal
R=zeros(1,N);   % initialize output vector with N samples
for k=0:N-1
   fself = timeshift(x,-k,0,0); % future self is the input signal right-shifted by k
samples
   R(k+1)= sum(x.*fself);       % autocorrelation value at (k+1) sample
end
R=R/R(1);   % normalize to [-1;1]
end
```

C. Thuật toán dịch thời gian tín hiệu

```
function m = timeshift(x,k,defaultL,defaultR)
%Time left-shifting vector x by k: x[n-k]
%Usage: timeshift(x,k,defaultL,defaultR)
%       where:  x is the target vector
%               k is the value to shift
%               defaultL: default value for elements added from the left
%               defaultR: default value for elements added from the right
m=x;
len=length(x);
for i=1:1:len
    rval=defaultL;
    if(i-k>len)
```

```
        rval=defaultR;
    else
        if(i-k>0)
            rval=x(i-k);
        end
    end
    m(i)=rval;
end
```

D. Thuật toán chia khung tín hiệu

```
function frames = splitx(x, flen)
% Split input signal into frames of specified length
%   x: input signal
%   flen: length of each frame (samples)
%   OUTPUT: mattrix of Nx2 elements, where
%           N is number of frames;
%           column 1: starting sample (inclusive);
%           column 2: ending sample (inclusive);

% constants
N=length(x);
C = ceil(N/flen);   % number of frames

frames = zeros(C,2);   % initialize matrix
for k=1:C
    % calculating left and right bounds
    rightB = k*flen;
    leftB = rightB-flen+1;
    if(rightB>N)    % if right boundary overflow frame boundary
        rightB=N;   % set right boundary to frame boundary
    end
    % storing left and right boundary index to (k)th frame
    frames(k,1)=leftB;
    frames(k,2)=rightB;
end
```

E. Thuật toán tìm độ trễ của cực đại địa phương

```
function peakis = peaks(y)
%PEAKS Exclude all non-peak elements in the input signal
%   y: Input signal

N=length(y);    % length of input signal
peakis = zeros(1,N);    % let all samples be zero
% for samples in:
for i=2:N-1
    if y(i)>y(i-1) && y(i)>y(i+1)   % if sample(i) is a peak
        peakis(i) = y(i);   % save peak to index i
    end
end
```

F. Thuật toán tìm tần số cơ bản

```
function f0 = fundfreq(y,Fs)
%FUNDFREQ Return fundamental frequency of the input signal
%   y: input signal
%   Fs: sampling rate

    % CONSTANTS
    N=length(y);    % length of y (samples)
    f0=NaN;         % let initial f0 be undetermined
    R0 = 0.27;      % pitch detection threshold
    f0min = 70;     % minimum f0 (Hz)
```

```
    f0max = 400;    % maximum f0
    dfmin=max(1,floor(Fs/f0max));   % minimum delay (sample): using proven formula
    dfmax=min(N,floor(Fs/f0min));   % minimum delay (sample)
    dpeak=max(1,floor(N/100));% number of samples being locals to peak
    % PROCESS
    R=autocorrel(y);                % autocorrelation function of y
    [peak,peaki] = max(peaks(R));   % find largest local maximum and its index
    tc = peaki-1;                   % concrete period
    localleft=max(2,peaki-dpeak);   % left most local sample
    localright=min(peaki+dpeak,N);  % right most local sample

    if( peak >= R0...%)  % if greater than threshold (possibly not a noise)
        && tc>=dfmin && tc <=dfmax...   % inside acceptable f0 range
        && peak >= max(R(localleft),R(localright)))  % peak is truly local maximum
        T0=tc/Fs; % fundamental period
        f0=1/T0;    % fundamental frequency
    end
end
```

G. Thuật toán tổng hợp (phiên bản có phân đoạn tiếng nói-khoảng lặng)

```
function ctour = pitchcontour(y,Fs)
%PITCHCONTOUR Return the fundamental frequencies, if exists, frame-by-frame
%   y: voice signal
%   Fs: sampling rate

    flen=20;                % frame length (milliseconds)
    felms = flen*Fs/1000;    % frame length (samples)

    vsbounds=svfilter(y,Fs);    % discriminate silence-voiced areas
    ctour=[];   % initialize contour map
    silenced = 1;   % let first area be silenced area
    for i=1:length(vsbounds)-1  % for each areas
       sig = vsbounds(i):vsbounds(i+1);     % silenced signal boundaries
       frames = splitx(sig,felms);          % split silenced signal into frames
       N=size(frames,1);                    % get number of frames
       F0s = NaN(1,N);                      % all frames have undefined f0
       if silenced ~= 1             % if not a silenced area
           for k=1:N                % loop through each frame
                framek = sig(frames(k,1):frames(k,2));    % specify working frame
                yframek=y(framek);                        % y(framek)
                f0 = fundfreq(yframek,Fs);  % find f0 of this frame
                F0s(k) = f0;                % assign f0 to output vector
           end
       end
       ctour=[ctour F0s];               % append reusult to output vector
       silenced = -silenced;            % switch mode: silenced - voiced
    end
end
```

H. Thuật toán tổng hợp (phiên bản không phân đoạn tiếng nói-khoảng lặng)

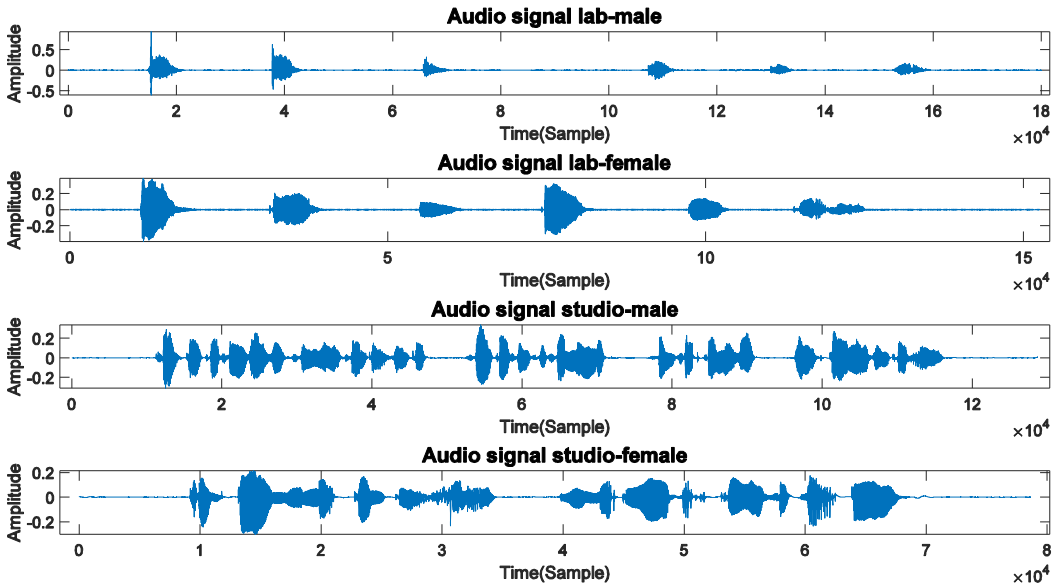
```
function F0s = pitchcontournosv(y,Fs)
%PITCHCONTOUR Return the fundamental frequencies, if exists, frame-by-frame
%   y: voice signal
%   Fs: sampling rate

flen=20;                % frame length (milliseconds)
felms = flen*Fs/1000;    % frame length (samples)
frames = splitx(y,felms);% frames boundaries in y
N = size(frames,1);     % number of rows == number of frames
F0s=zeros(1,N);         % initialize output vector
```

```
for k=1:N                % loop through each frame
    framek = frames(k,1):frames(k,2);    % specify working frame
    f0 = fundfreq(y(framek),Fs);   % find f0 of this frame
    F0s(k) = f0;                % assign f0 to output vector
end
end
```

IV. KẾT QUẢ THỰC NGHIỆM

A. Dữ liệu mẫu

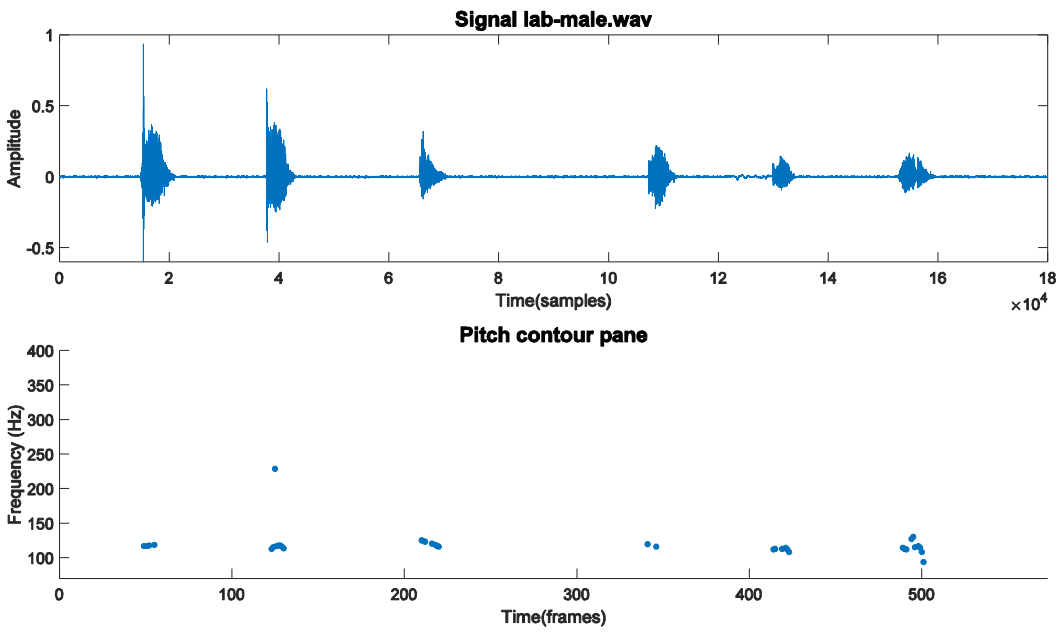
Dữ liệu mẫu được sử dụng để đánh giá thuật toán là 4 tín hiệu giọng nói được thu âm bởi 2 người khác nhau trong 2 môi trường khác nhau, được lấy mẫu với tần số lấy mẫu là 16 kHz, độ dài trung bình là 8 giây.



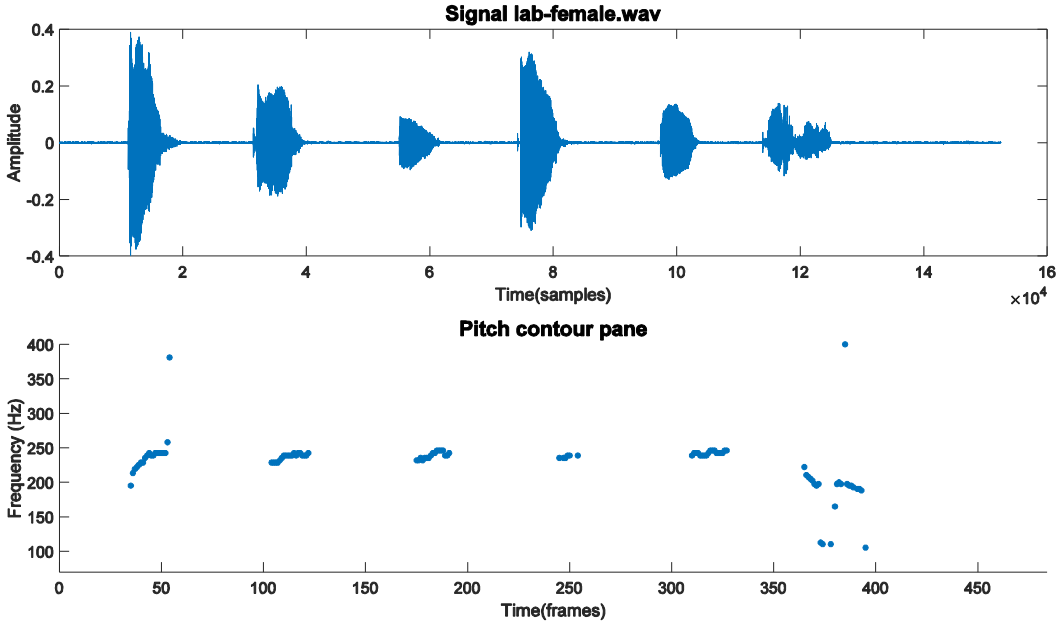
Hình 5. Các tín hiệu mẫu được sử dụng để đánh giá thuật toán.

B. Kết quả thu được

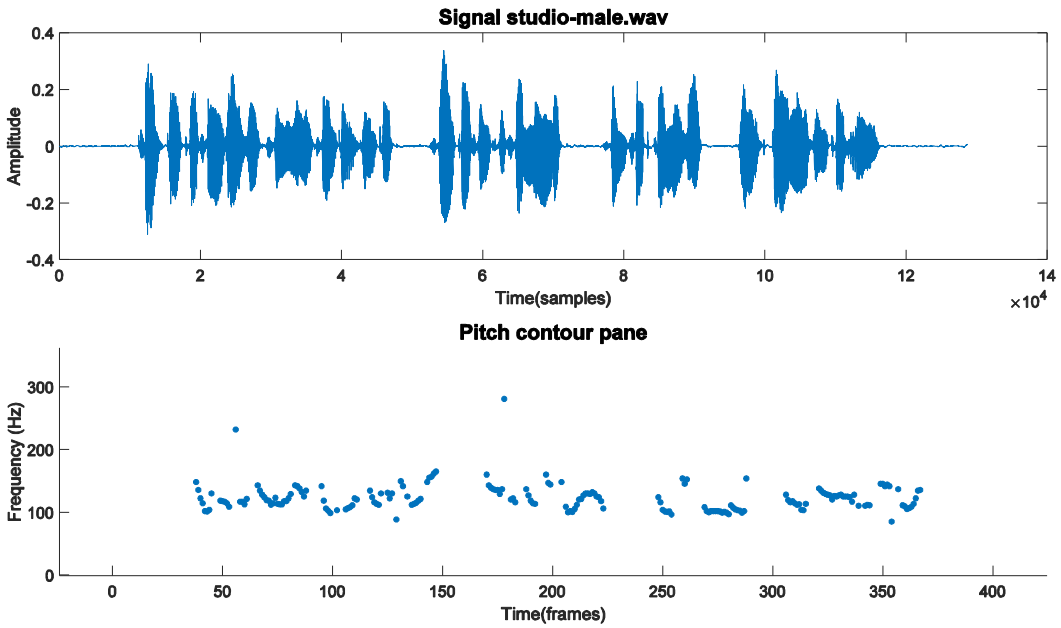
Áp dụng thuật toán cho từng dữ liệu mẫu, ta thu được kết quả như trong các hình bên dưới.



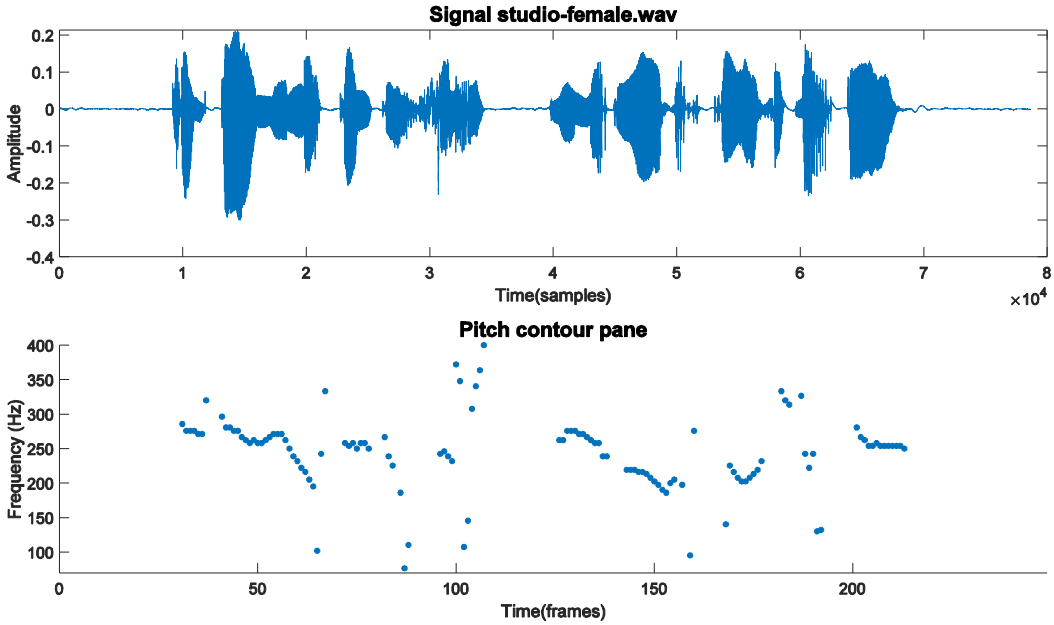
Hình 6. Kết quả tính tần số cơ bản (hình dưới) cho tín hiệu mẫu (hình trên).



Hình 7. Kết quả tính tần số cơ bản (hình dưới) cho tín hiệu mẫu (hình trên).

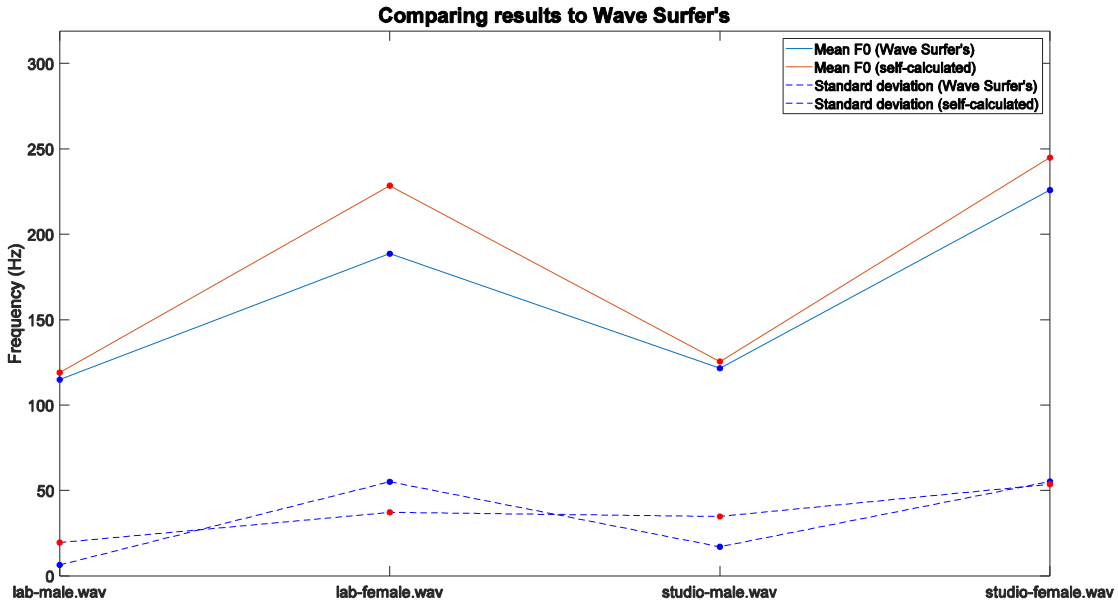


Hình 8. Kết quả tính tần số cơ bản (hình dưới) cho tín hiệu mẫu (hình trên).



Hình 9. Kết quả tính tần số cơ bản (hình dưới) cho tín hiệu mẫu (hình trên).

Để đánh giá sai số của thuật toán, thay vì đo thủ công nhóm sử dụng phần mềm Wave Surfer để tìm giá trị trung bình và phương sai của tần số cơ bản cho mỗi dữ liệu mẫu. Kết quả so sánh được thể hiện trong hình dưới đây.



Hình 10. So sánh kết quả tìm được với kết quả từ Wave Surfer.

Tính toán cho thấy sai số trung bình (Root mean square error – RMSE) giữa kết quả của thuật toán và kết quả chuẩn cho 4 mẫu dữ liệu là 22.2770 Hz, phương sai là 14.2337 Hz.

Sai số của thuật toán phụ thuộc vào một số tham số sau:

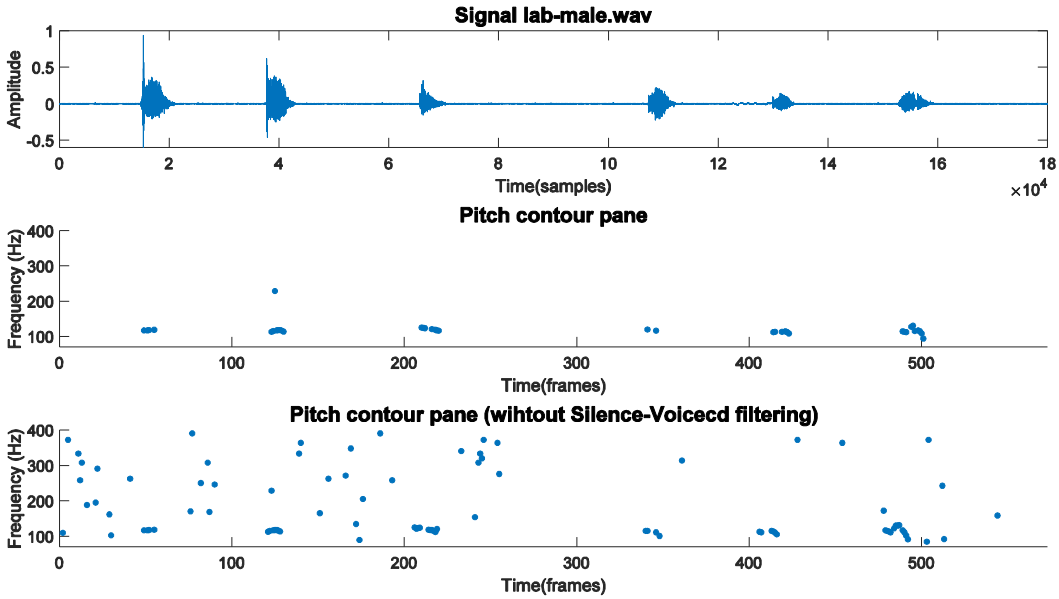
- Sai số của thuật toán phân đoạn tiếng nói-khoảng lặng: Thuật toán phân đoạn tiếng nói-khoảng lặng được sử dụng có sai số 0.0420 giây. Sai số của thuật toán này càng nhỏ thì các tín hiệu được xử lý sẽ càng đầy đủ. Ngược lại, khi sai số của thuật toán này lớn thì hoặc thuật toán sẽ phải xử lý càng nhiều tín hiệu không có tiếng nói (unvoiced) hoặc thuật toán sẽ bỏ sót càng nhiều tín hiệu có tiếng nói (voiced).
- Giá trị ngưỡng của hàm tự tương quan (R0): Xác định giá trị R0 không chính xác làm thuật toán không phát hiện đúng loại âm (vô thanh hay hữu thanh) của khung tín hiệu, khiến thuật toán thực hiện tính

toàn thiếu, hoặc thừa. Cụ thể: nếu R0 quá nhỏ thì thuật toán sẽ coi một số khung tín hiệu chứa âm hữu thanh là khung âm hữu thanh và tính tần số cơ bản sai; ngược lại khi R0 quá lớn thì thuật toán coi một số khung hữu thanh là khung vô thanh và không thực hiện tính toán.

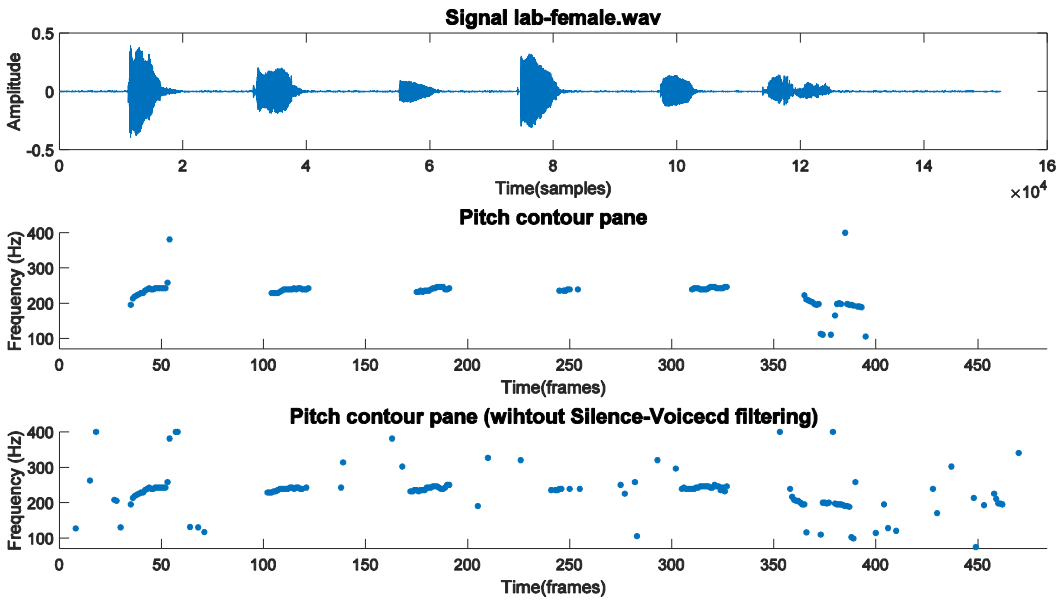
- Kích thước (số tín hiệu) của một khung: Cần có giá trị tối thiểu bằng 2 lần chu kỳ của tín hiệu trong khung, thường lấy giá trị từ 10ms đến 30ms. Nếu kích thước khung nhỏ hơn giá trị tối thiểu đồng nghĩa khi tính hàm tự tương quan của khung tín hiệu thì độ trễ nhỏ hơn chu kỳ tín hiệu dẫn đến không thể xác định được tần số cơ bản.

C. So sánh với phiên bản không chia tiếng nói-khoảng lặng của thuật toán

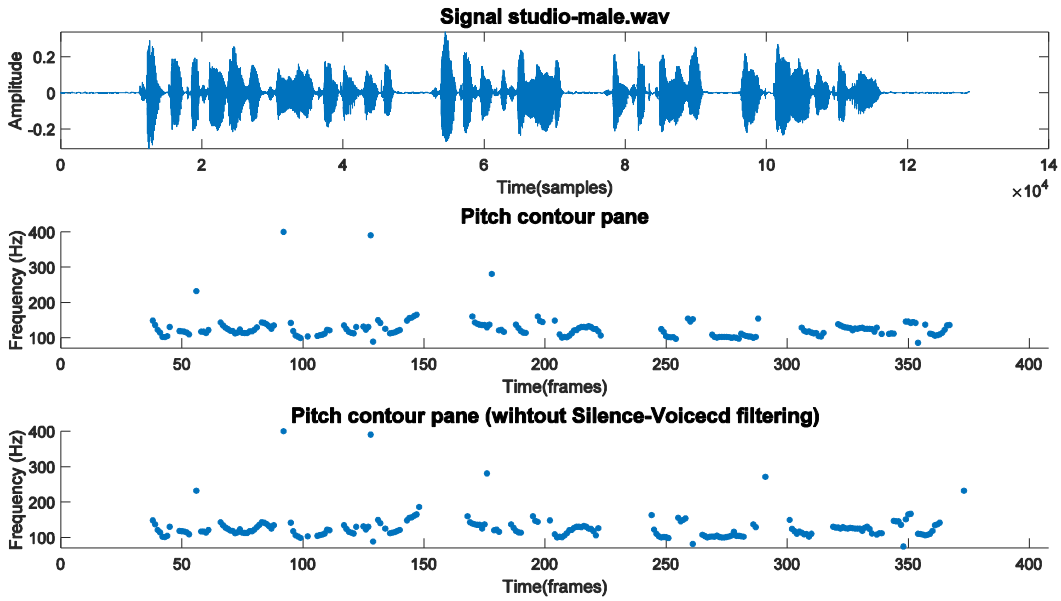
Khi sử dụng phiên bản không thực hiện chia tiếng nói-khoảng lặng của thuật toán để tìm tần số cơ bản cho 4 mẫu tín hiệu trên và so sánh kết quả với kết quả thu được bằng phiên bản còn lại, ta thu được các kết quả như dưới đây.



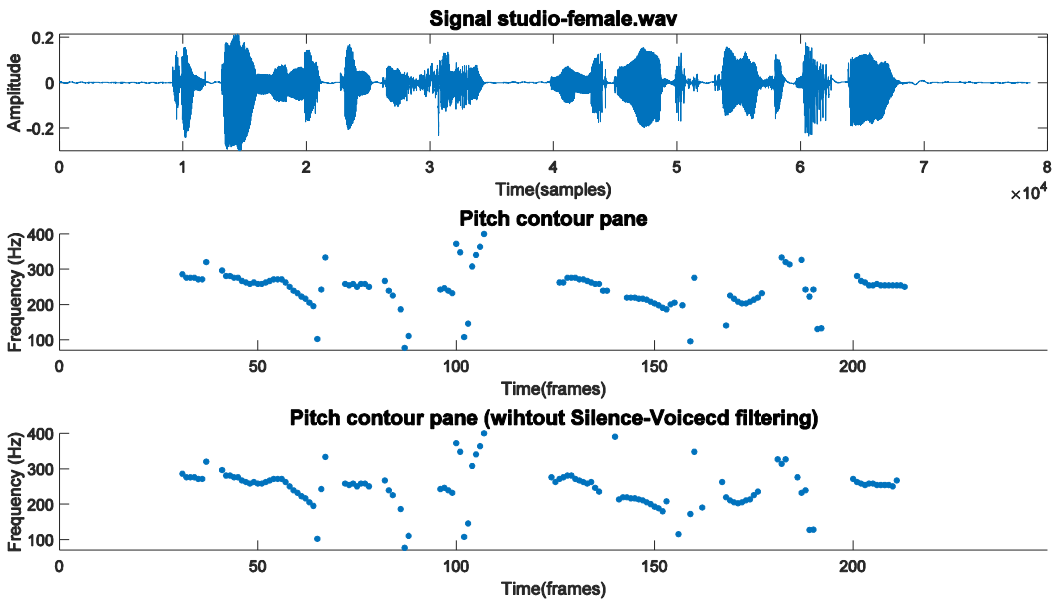
Hình 11. Tần số cơ bản khi không (hình thứ 3) và có (hình thứ 2) chia tiếng nói-khoảng lặng cho tín hiệu mẫu (hình thứ nhất).



Hình 12. Tần số cơ bản khi không (hình thứ 3) và có (hình thứ 2) chia tiếng nói-khoảng lặng cho tín hiệu mẫu (hình thứ nhất).

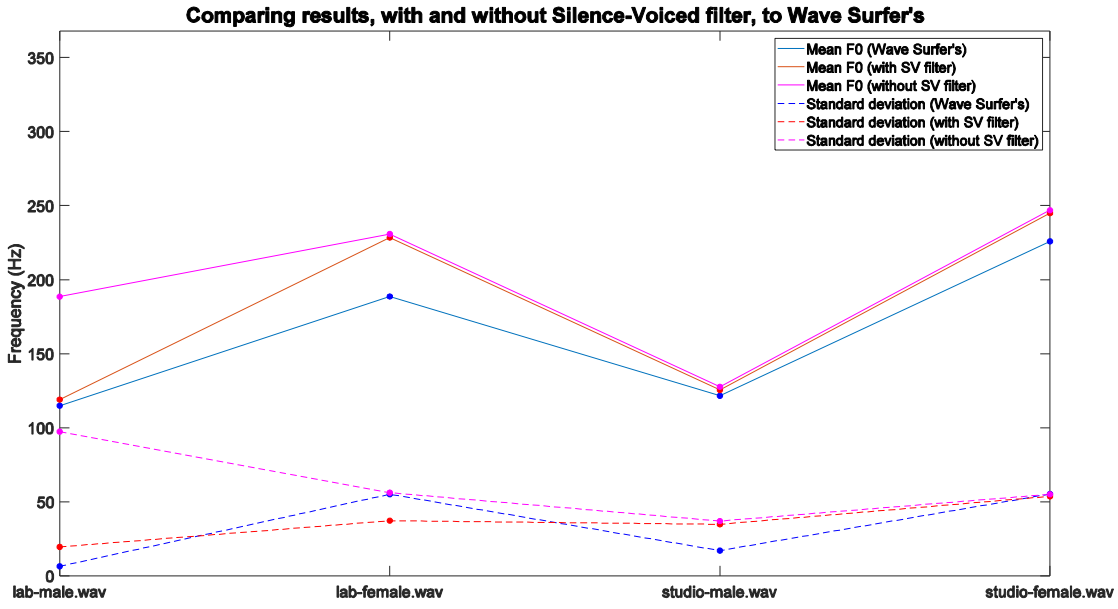


Hình 13. Tần số cơ bản khi không (hình thứ 3) và có (hình thứ 2) chia tiếng nói-khoảng lặng cho tín hiệu mẫu (hình thứ nhất).



Hình 14. Tần số cơ bản khi không (hình thứ 3) và có (hình thứ 2) chia tiếng nói-khoảng lặng cho tín hiệu mẫu (hình thứ nhất).

Tính trung bình tần số cơ bản của từng mẫu dữ liệu và so sánh kết quả với nhau và với kết quả chuẩn, ta được đồ thị như dưới đây.



Hình 15. So sánh kết quả tìm được khi có và không thực hiện chia tiếng nói-khoảng lặng với kết quả từ Wave Surfer.

Kết quả tính toán cho thấy sai số trung bình giữa tần số cơ bản mà thuật toán này tìm được và kết quả chuẩn cho 4 mẫu dữ liệu là 43.8399 Hz (lớn hơn hơn 2 lần so với kết quả của thuật toán có thực hiện chia tiếng nói-khoảng lặng), và phương sai là: 46.5804 Hz (lớn hơn hơn 3 lần so với kết quả của thuật toán có thực hiện chia tiếng nói-khoảng lặng).

V. KẾT LUẬN

Nhóm đã cài đặt thành công thuật toán tự động tìm tần số cơ bản của tín hiệu tiếng nói trên miền thời gian bằng hàm tự tương quan với 2 phiên bản là phiên bản có thực hiện phân đoạn tiếng nói-khoảng lặng và phiên bản không thực hiện phân đoạn tiếng nói-khoảng lặng.

Kết quả thử nghiệm với 4 mẫu tín hiệu cho thấy các tần số cơ bản được tìm tự động có giá trị gần đúng với tần số cơ bản tìm bởi phần mềm Wave Surfer, với sai số trung bình là 22.2770 Hz và phương sai là 14.2337 Hz (khi dùng phiên bản phân đoạn tiếng nói-khoảng lặng trên tín hiệu mẫu) hoặc với sai số trung bình là 43.8399 Hz, phương sai 46.5804 Hz (khi dùng phiên bản không phân đoạn tiếng nói-khoảng lặng). Kết quả đó cũng cho thấy sử dụng phiên bản có phân đoạn tiếng nói-khoảng lặng của thuật toán tìm được tần số cơ bản chính xác hơn so với phiên bản còn lại.

Trong tương lai nhóm sẽ tìm cách giảm sai số của thuật toán bằng cách thử nghiệm thuật toán với nhiều mẫu dữ liệu khác để tìm ra ngưỡng năng lượng chuẩn chính xác hơn, sử dụng thêm bộ lọc trung vị giúp để giảm độ nhiễu của tín hiệu, hoặc nghiên cứu sử dụng các thuật toán khác hiệu quả hơn.

VI. TÀI LIỆU THAM KHẢO

[1] Link: http://www.speech.kth.se/wavesurfer/index.html
[2] Tran Van Tam, “Xác định tần số cơ bản của tín hiệu tiếng nói dùng hàm tự tương quan”, Đại học Đà Nẵng, 2019.
[3] Nguyen Nhat Tung, Ninh Hai Hoang, “Phân đoạn tín hiệu thành tiếng nói và khoảng lặng dựa vào năng lượng ngắn hạn của tín hiệu”, Khoa Công nghệ thông tin, Trường Đại học Bách Khoa, Đại học Đà Nẵng, 2020.
[4] Matthieu Hodgkinson, “CS425 Audio and Speech Processing”, National University of Ireland, Maynooth, pp.49-63, 2012.