Final Project part III

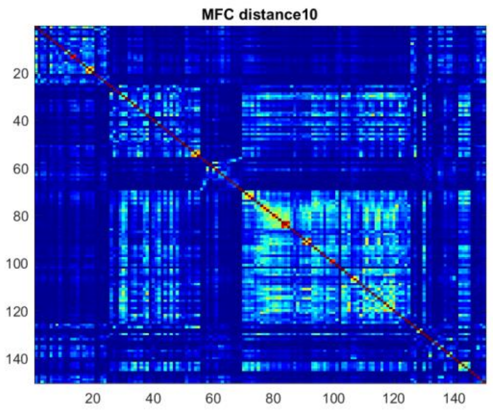
Zhi Jie Huang

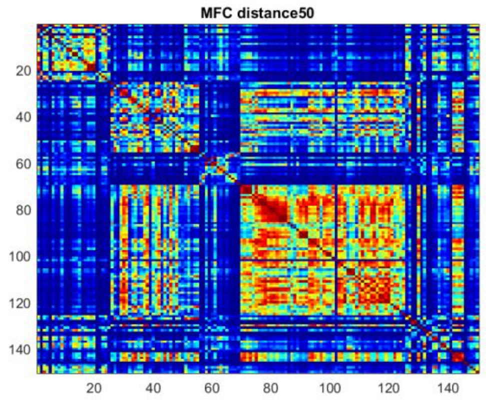
ECEN 4632

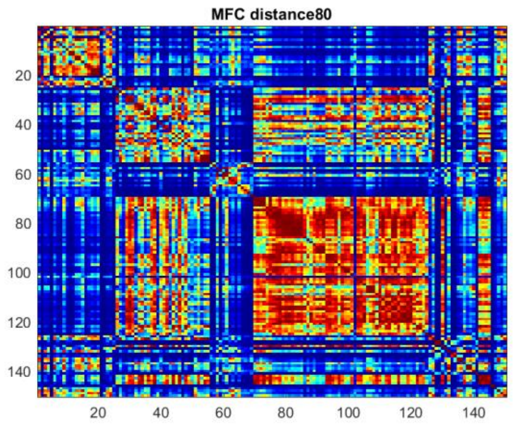
Project Partner: Richard Smith

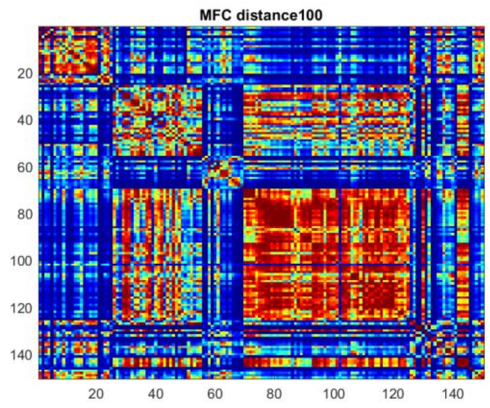
This part of the project is to utilize the MFCC and chroma coefficient that I calculated earlier in the final project part I and final project part II. Since there are only have 12 bands in the chroma representation, therefore I must merge some of the Mel banks. Since 40 is not divisible by 12, I pick 36 as the nearest divisible integer. In each of the two track, I am interested in comparing the distribution of the note. I am using the multivariate Gaussian distribution, this approach collapse all the frames together. To calculate the distance between the two Gaussian distribution, I need to first calculate the mean and the covariance for each matrix. I calculate a symmetric version of the Kullback-Leibler divergence. With parameter gamma varies between 10~100. After I calculate the distance matrix for 150 songs, I must average the distance within the same genres and do comparison to different genre. With this method, I can see the distinct different between some genres, but not others.

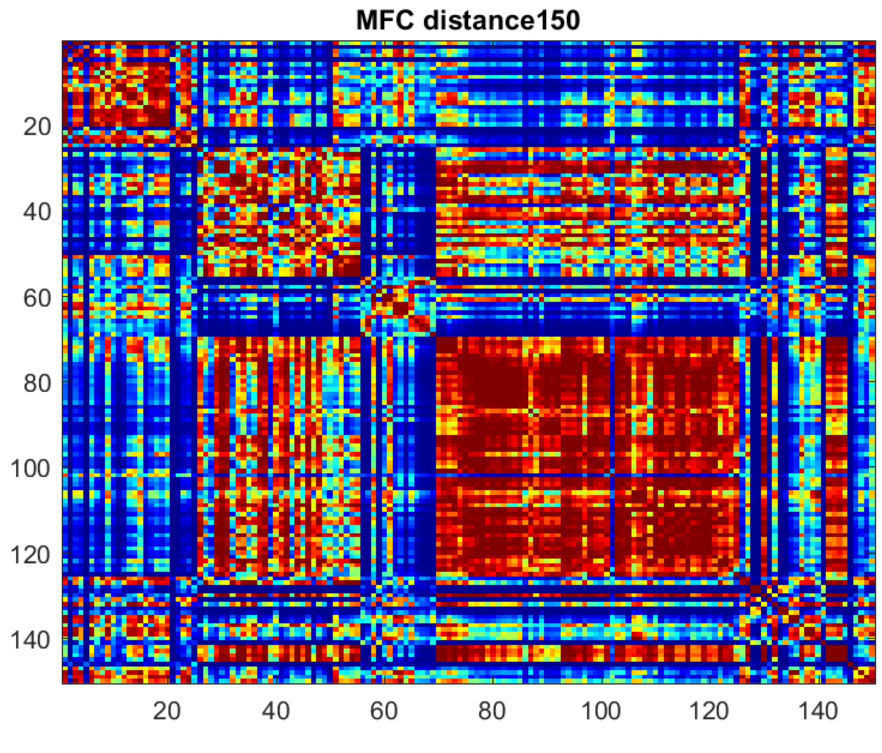
There are 6 genres in this project, classical, electronics, jazz/blues, metal/punk, pop/rock and world. Each genre has 25 songs, it makes the total of 150 songs. The following 5 picture vary gamma from 10 to 150 of the MFCC to maximizes the separation between the different genres. I think gamma equal to 100 looks the best for the distance matrix. The idea of the distance matrix is to compute the distance between every two track of the 150 songs. As you can see below in the MFCC distance 100 graph, the first 25 songs are classical, as you can see it has a small distance within the first 25 songs, that's why you can see a lot of red in the top left area. And it's really blue to another other songs, therefore, you can clearly see the first region is classical. The second genres is electronics, the index 26~50 are electronics, as you can see in the MFCC distance100 graph, it does have a small distance between electronics; however, it also have a very small distance between metal/punk, pop/rock and electronics. Due to the similar instruments played in that music. But you can still see electronics has a relatively large between world, jazz/blues and classical. Metal/punk and pop/rock music are similar type, therefore I have a hard time distinguish between the two. World doesn't really have a small distance to anything, which makes sense, because world music is very rich, it included various instrument.


MFC distance10

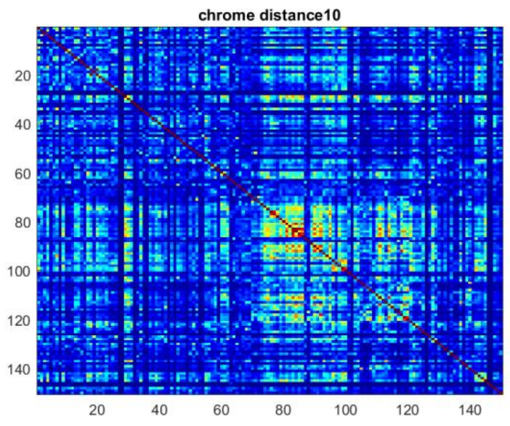

MFC distance50

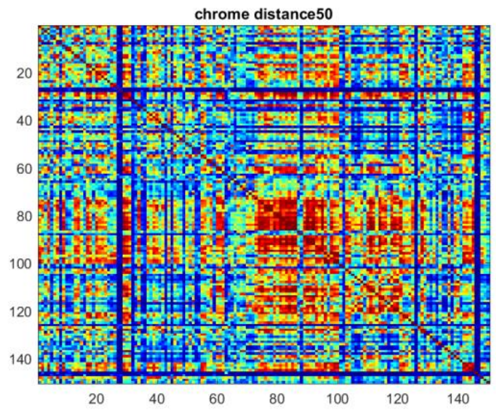

MFC distance80

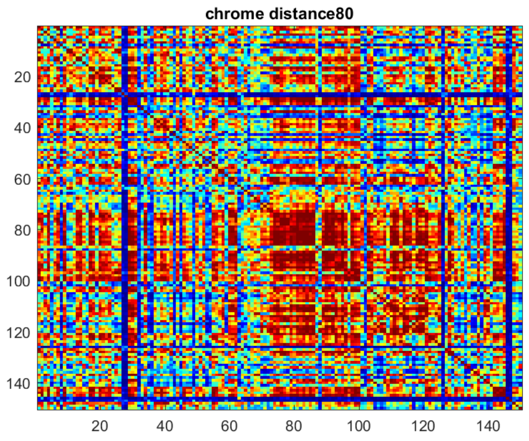

MFC distance100

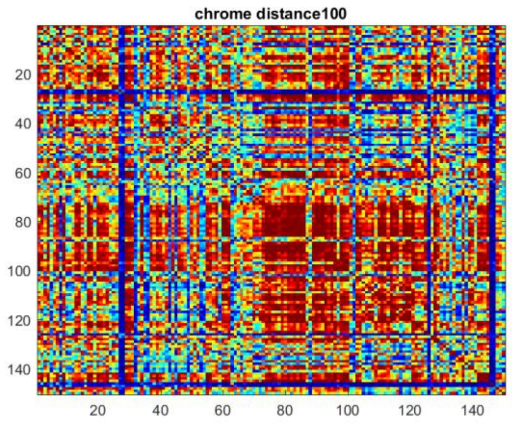

MFC distance150

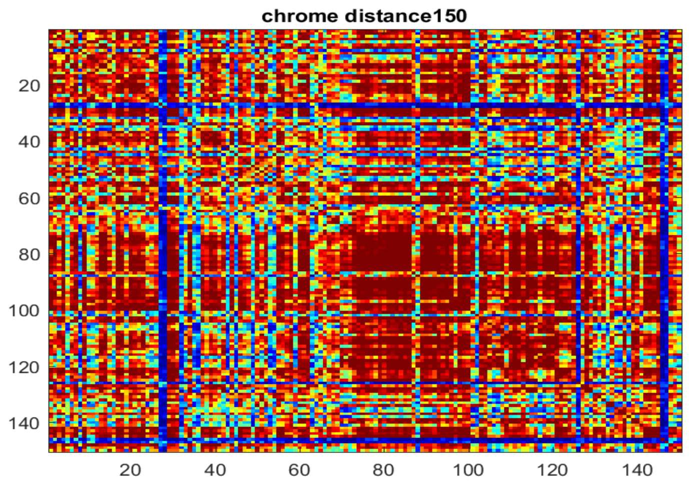
For the Chroma coefficient, I must do the same procedure for the distance matrix. The result are shown below. The distance matrix for Chroma clearly have more noise than the MFCC distance matrix, it is due to we destroy all the melody and bring everything into the first octave. As you can see in the result, it is very hard to use Chroma to distinguish one genres from another.



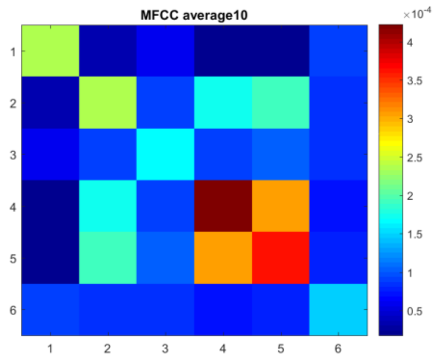


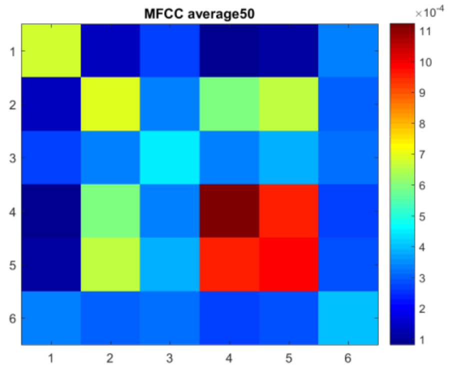


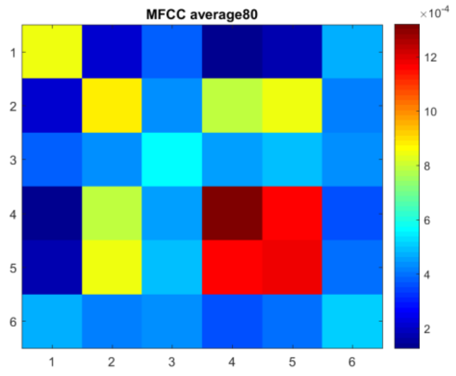


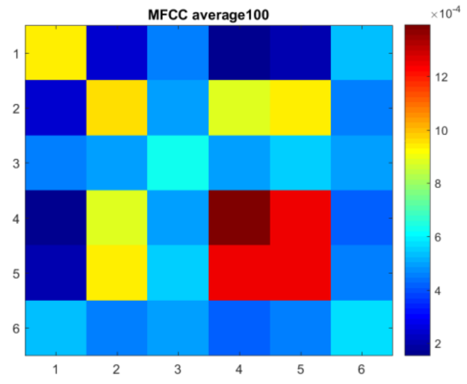


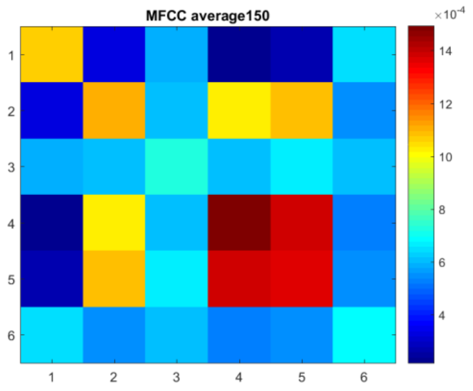
After I calculated the distance matrix between each 2 tracks of the 150 songs, I am going to average the value of each 25x25 matrix to get an average between genres. The below figures illustrate the separation between genres, as you can see at gamma equal to 100, it has the best separation between genres.



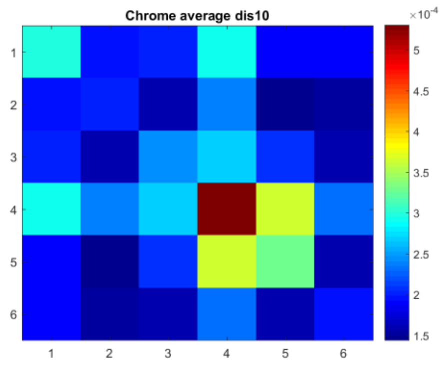


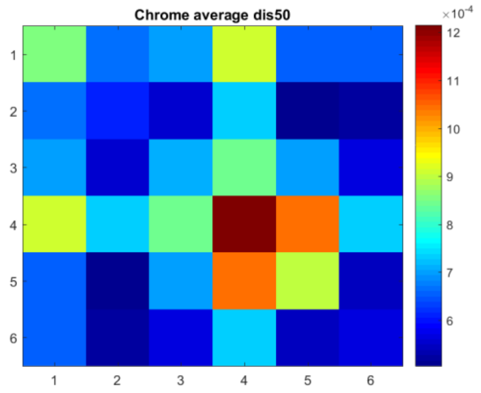


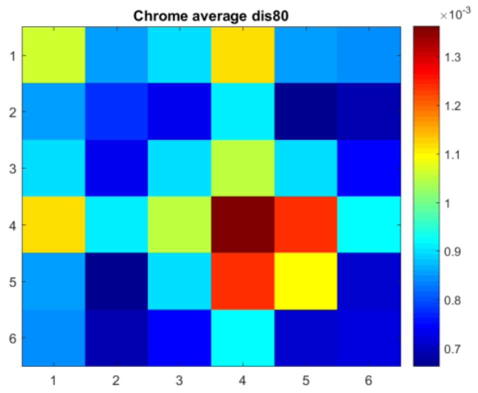


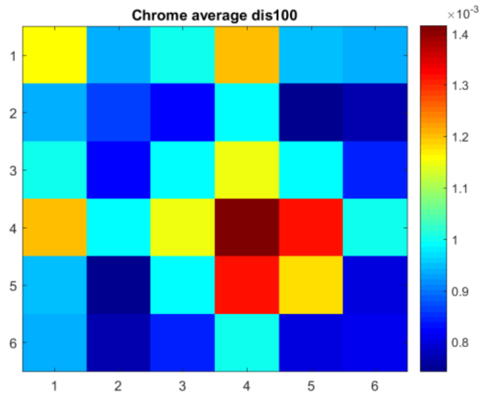


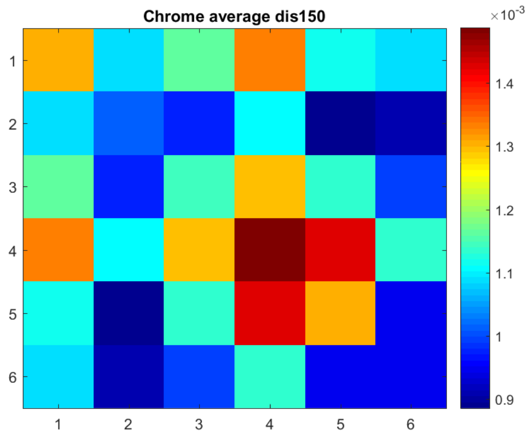
The chroma average distance matrix doesn't have a really good distinction between genres.











Conclusion:

Throughout this project, I have learned a lot about how to apply Digital Signal Processing techniques to the real world. The next step of the algorithm is to try out using part of a data as a training set and run the rest of the data through the algorithm.

Appendix:

Main.m

```
clear all;close all;clc;
tic
pathName = 'C:\Users\JayHuang\Documents\GitHub\ECEN4632-DSP\Final_Project\';
fileList = dir(fullfile(pathName,'*.wav'));
fileName = dir('*.wav');
NumSong = size(fileName,1);
mfcc1 = cell(150,1);
Chrome1 = cell(150,1);
fftSize = 512;
w = hann(fftSize);
fftSizeC = 2048;
parfor i = 1:NumSong
    %--------Extract the length---------%
    [song,fs] = audioread(fileName(i).name);
    ExtractedSong = SongExtract(song,120,fs);
     mfccCoe = mfcc(ExtractedSong,fs,fftSize,w);
    % demochrom('track201-classical');
    mfcc1{i} = Cov_Mu(mfccCoe);
    Chrome1{i} = mychroma(ExtractedSong',fs,fftSizeC);
end
MfccD = zeros(150,150);
chromeD = zeros(150,150);

for i = 1:NumSong
    for j = 1:i
        MfccD(i,j) = distance(mfcc1{i},mfcc1{j});
        MfccD(j,i) = MfccD(i,j);
        chromeD(i,j) = distance(Chrome1{i},Chrome1{j});
        chromeD(j,i) = chromeD(i,j);
    end
end

DMFCCavg = zeros(6,6);
DChromeavg = zeros(6,6);
%---------Calculate the average matrix---------

for i = 1:25:150
    for j = 1:25:150
        i_ = fix(i/25);
        j_ = fix(j/25);
        DMFCCavg(i_+1,j_+1) = mean(mean(MfccD(i:i+24,j:j+24)))/625;
        DChromeavg(i_+1,j_+1) = mean(mean(chromeD(i:i+24,j:j+24)))/625;
    end
end

%gcf get current figure
figure
colormap jet
imagesc(DChromeavg);
title('Chrome average dis150');
colorbar
saveas(gcf,'Chrome_average_dis150','png');
```

```
figure
colormap jet
imagesc(DMFCCavg);
title('MFCC average150');
colorbar
saveas(gcf,'MFCC_average150','png');

figure
colormap jet
imagesc(MfccD)
title('MFC distance150');
saveas(gcf,'MFC_distance150','png');


figure
colormap jet
imagesc(chromeD);
title('chrome distance150');
saveas(gcf,'chromedistance150','png');
colorbar
toc
```

distance.m

```
function [d] = distance(mfcc1,mfcc2)
    mu1 = mean(mfcc1,2);
    co1 = cov(mfcc1');
    mu2 = mean(mfcc2,2);
    co2 = cov(mfcc2');
    K = 12;
    gam = 150;
    iCo1 = pinv(co1);
    iCo2 = pinv(co2);
    KL = 0.5*trace(co1*iCo2) + trace(co2*iCo1) + trace(((mu1-
mu2)'*(iCo1+iCo2)*(mu1-mu2)))-K;
    d = 1 - exp(-gam/(KL + eps));
end
```

Cov_Mu.m

```
function [mfcc1] = Cov_Mu(mfcc)
%Merge the Mel bank to 12 bands
t = zeros(1,36);
t(1) = 1; t(7:8) = 5; t(15:18) = 9;
t(2) = 2; t(9:10) =6; t(19:23) = 10;
t(3:4)=3; t(11:12)=7; t(24:29) = 11;
t(5:6)=4; t(13:14)=8; t(30:36) = 12;
mel2 = zeros(12,size(mfcc,2));
    for i = 1:12
        mel2(i,:) = sum(mfcc(t==i,:),1);
    end
    len = size(mel2,2);
    mfcc1 = mel2(:,1:len-1);
end
```

SongExtract.m

```
function [output] = SongExtract(song,NumSec,fs)
SongLength = length(song);
GrabLength = NumSec*fs;
Mid = floor(SongLength/2);
    if SongLength <= GrabLength
        output = song;
    else
        output = song(Mid-GrabLength/2:Mid+GrabLength/2-1);
    end
end
```

mychroma.m

```
function [C] = mychroma (signal,samplingRate,lengthFFT)
%
%
%  Computes the 12 x nframes chroma representation
%
%  The computation procedes in two steps:
%    1) an estimate of the instantaneous frequency is computed using the
derivative of the
%    phase. This idea relies on the concept of the analytical signal.
%    2) the energy associated with the instantaneous frequency is folded back
into the octave
%    defined by A0.


A0    = 27.5;

frameSize   = lengthFFT/2;
frameSize2 = lengthFFT/4;

nchr = 12;

s = length(signal);
nFrames = 1 + floor((s - (frameSize))/(frameSize2));

T = (frameSize)/samplingRate;

win = 0.5*(1-cos([0:((frameSize)-1)]/(frameSize)*2*pi));
dwin = -pi / T * sin([0:((frameSize)-1)]/(frameSize)*2*pi);

norm = 2/sum(win);

iF = zeros(1 + frameSize, nFrames);
S = zeros(1 + frameSize, nFrames);

nmw1 = floor(frameSize2);
nmw2 = frameSize - nmw1;

ww = 2*pi*[0:(lengthFFT-1)]*samplingRate/lengthFFT;

%
% computation of the instantaneous frequency, using the analytical expression
of the derivative of the phase

for ifrm = 1:nFrames

    u = signal((ifrm-1)*(frameSize2) + [1:(frameSize)]);

    wu = win.*u;
    du = dwin.*u;

    wu = [zeros(1,nmw1),wu,zeros(1,nmw2)];
    du = [zeros(1,nmw1),du,zeros(1,nmw2)];

    t1 = fft(fftshift(du));
    t2 = fft(fftshift(wu));

    S(:,ifrm) = t2(1:(1 + frameSize))'*norm;

    t = t1 + j*(ww.*t2);
    a = real(t2);
    b = imag(t2);
    da = real(t);
    db = imag(t);

    instf = (1/(2*pi))*(a.*db - b.*da)./((a.*a + b.*b)+(abs(t2)==0));

    iF(:,ifrm) = instf(1:(1 + frameSize))';
end;


%
% now we find the local maxima of the instantaneous frequency

f_ctr = 1000;
f_sd = 1;

f_ctr_log = log2(f_ctr/A0);
fminl   = oct2hz(hz2oct(f_ctr)-2*f_sd);
fminu  = oct2hz(hz2oct(f_ctr)-f_sd);
fmaxl  = oct2hz(hz2oct(f_ctr)+f_sd);
fmaxu = oct2hz(hz2oct(f_ctr)+2*f_sd);

minbin  = round(fminl * (lengthFFT/samplingRate) );
maxbin = round(fmaxu * (lengthFFT/samplingRate) );

%
% compute the second order derivative of the instantaneous frequency to
detect the peaks

ddif    = [iF(2:maxbin, :);iF(maxbin,:)] - [iF(1,:);iF(1:(maxbin-1),:)];

% clean a bit

dgood = abs(ddif) < .75*samplingRate/lengthFFT;

dgood = dgood .* ([dgood(2:maxbin,:);dgood(maxbin,:)] >  0 |
[dgood(1,:);dgood(1:(maxbin-1),:)] > 0);

p  = zeros(size(dgood));
m = zeros(size(dgood));

%
%  try to fit a second order polynomial locally
```

```
for t = 1:size(iF,2)
    ds = dgood(:,t)';
    lds = length(ds);

    st = find(([0,ds(1:(lds-1))]==0) & (ds > 0));
    en = find((ds > 0) & ([ds(2:lds),0] == 0));
    npks = length(st);
    frqs = zeros(1,npks);
    mags = zeros(1,npks);

    for i = 1:length(st)
        bump = abs(S(st(i):en(i),t));
        frqs(i) = (bump'*iF(st(i):en(i),t))/(sum(bump)+(sum(bump)==0));
        mags(i) = sum(bump);

        if frqs(i) > fmaxu
            mags(i) = 0;
            frqs(i) = 0;
        elseif frqs(i) > fmaxl
            mags(i) = mags(i) * max(0, (fmaxu - frqs(i))/(fmaxu-fmaxl));
        end

        if frqs(i) < fminl
            mags(i) = 0;
            frqs(i) = 0;
        elseif frqs(i) < fminu
            mags(i) = mags(i) * (frqs(i) - fminl)/(fminu-fminl);
        end
        if frqs(i) < 0
            mags(i) = 0;
            frqs(i) = 0;
        end
    end

    bin = round((st+en)/2);
    p(bin,t) = frqs;
    m(bin,t) = mags;
end

ncols = size (p,2);

%
% clean up

Pocts = hz2oct(p+(p==0));
Pocts(p(:)==0) = 0;

nzp = find(p(:)>0);

[hn,hx]  = hist(nchr*Pocts(nzp)-round(nchr*Pocts(nzp)),100);
centsoff = hx(find(hn == max(hn)));

Pocts(nzp) = Pocts(nzp) - centsoff(1)/nchr;
```

```
PoctsQ = Pocts;
PoctsQ(nzp) = round(nchr*Pocts(nzp))/nchr;

Pmapc = round(nchr*(PoctsQ - floor(PoctsQ)));
Pmapc(p(:) == 0) = -1;
Pmapc(Pmapc(:) == nchr) = 0;

C = zeros(nchr,ncols);
for t = 1:ncols;
    C(:,t)=(repmat([0:(nchr-
1)]',1,size(Pmapc,1))==repmat(Pmapc(:,t)',nchr,1))*m(:,t);
end

return;
```

oct2hz.m

```
function [hz] = oct2hz(octs,A440)

if nargin < 2;
    A440 = 440;
end

hz = (A440/16).*(2.^octs);

return;
```

hz2oct.m

```
function [oct] = hz2oct(freq, A440)

if nargin < 2;
    A440 = 440;
end
oct = log(freq./(A440/16))./log(2);

return;
```

demochrom.m

```
function demochrom (filename)

BasePath = ['Z:/ECEN4632/FinalProject/'];

% BasePath = ['/Users/francois/class/4532/tracks/'];

[x,fs] = audioread([BasePath '/' filename '.wav']);


x = x';

fftlen= 2048;

[C] = mychroma (x, fs, fftlen);

figure;

tt = [1:size(C,2)]*fftlen/4/fs;

imagesc(tt,[1:12],20*log10(C+eps));

axis xy
set (gca, 'YTick', [1:12]);
set (gca, 'YTickLabel', {'A',
'A#','B','C','C#','D','D#','E','F','F#','G','G#'});
set (gca, 'XTick', [0:60:ceil(tt(end))]);
caxis(max(caxis)+[-60 0])

title(['Chroma of ' filename]);
colormap('jet');hold on;colorbar;

return;
```