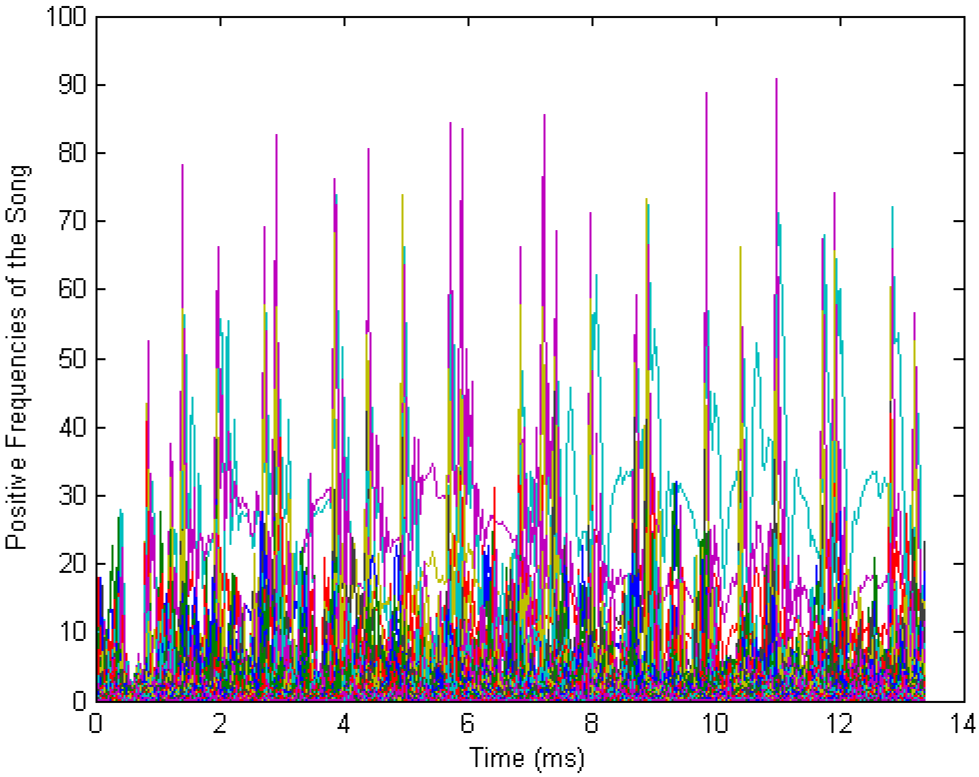
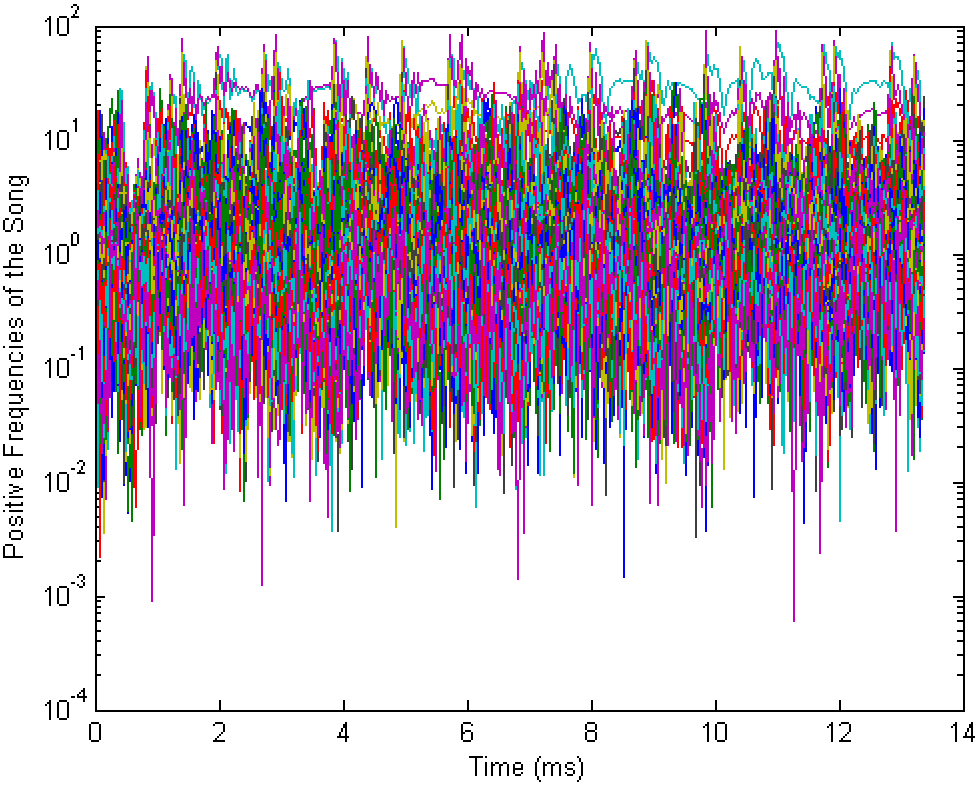
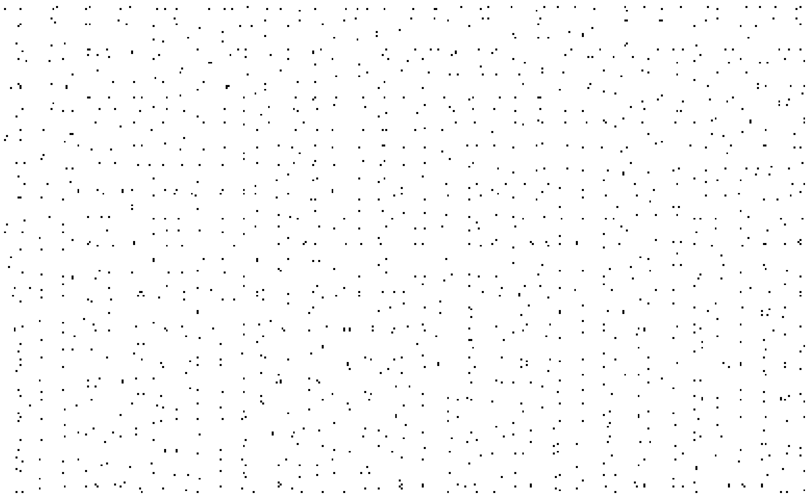
```
%NAME: Bob Wagner & Tyler Roberts

function songMatch = myshazam(clip)
% Database songs located in same folder.
% If desired, add more songs.S
songs=cellstr(['Shake It Off.mp3              ';...
    'New Romantics.mp3             ';'Wonderland.mp3                ';...
    'Welcome To New York.mp3       ';'All You Had To Do Was Stay.mp3']);
%Construct database
Library = [];
[Library,song_id,hTSizes] = make_database(songs,Library);
%__________________________________________________________________________
%Produce clip values
clipValues = make_table(clip);

% Generate match values
matchValues = [];
for i=1:size(clipValues,1)
    %for each song sector, compute hash function on match value row
    % NOTE: we appended our databases (one for each song) together, instead
    % of having them combined/distributed across the one db. It is
    % equivalent to use the relative hashing functions for each sector and
    % accumulate the collisions into one list. If we combined the dbs into
    % one db, we would get the exact same number of collisions associated
    % with their appropriate songs as we did in the list generated by our
    % method.
    cumulativeSize = 0; %offset for relative db access
    for j=1:size(hTSizes,2)
        %compute relative index
        relativeIndex = mod(floor(((clipValues(i,4)-clipValues(i,3))*(2^16))+...
        (abs(clipValues(i,1))*(2^8))+abs(clipValues(i,2))),...
        hTSizes(j))+1;
        %compute global db index
        globalIndex = relativeIndex + cumulativeSize;
        %if we have data there for the relative (song) db, add to list
        curr_db = Library(1:50, cumulativeSize+1:cumulativeSize+hTSizes(1,j));
        colIndex = findCollision(relativeIndex, curr_db);
        if(~(colIndex == 1))
            for k=1:floor(colIndex/2)
                add = [Library((k*2)-1,globalIndex)-clipValues(i,3),...
                        Library((k*2),globalIndex)];
                matchValues = [matchValues; add];
            end
        end
        cumulativeSize = cumulativeSize+hTSizes(1,j);
    end
end

% Generate histograms
for i=1:size(song_id,2)    %foreach song
    figure(i+4);
    subplot(1, size(song_id,2), i)
```

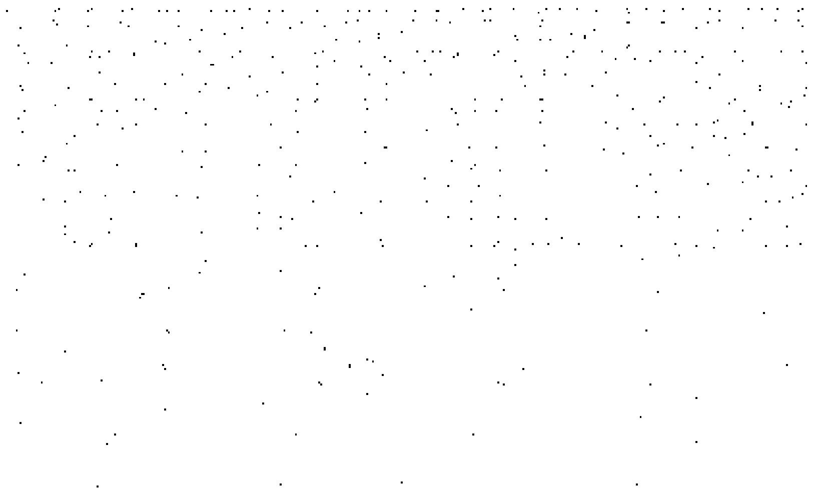
```
        xlabel('# of Occurances');
        ylabel('Time Difference');
        title(strcat('Song',num2str(i)));
        currList = [];
        for j=1:size(matchValues,1) %for each value, add to song histogram
           if(matchValues(j,2) == i)
              currList = [currList, matchValues(j,1)];
           end
        end
        hist(currList)
    end
```

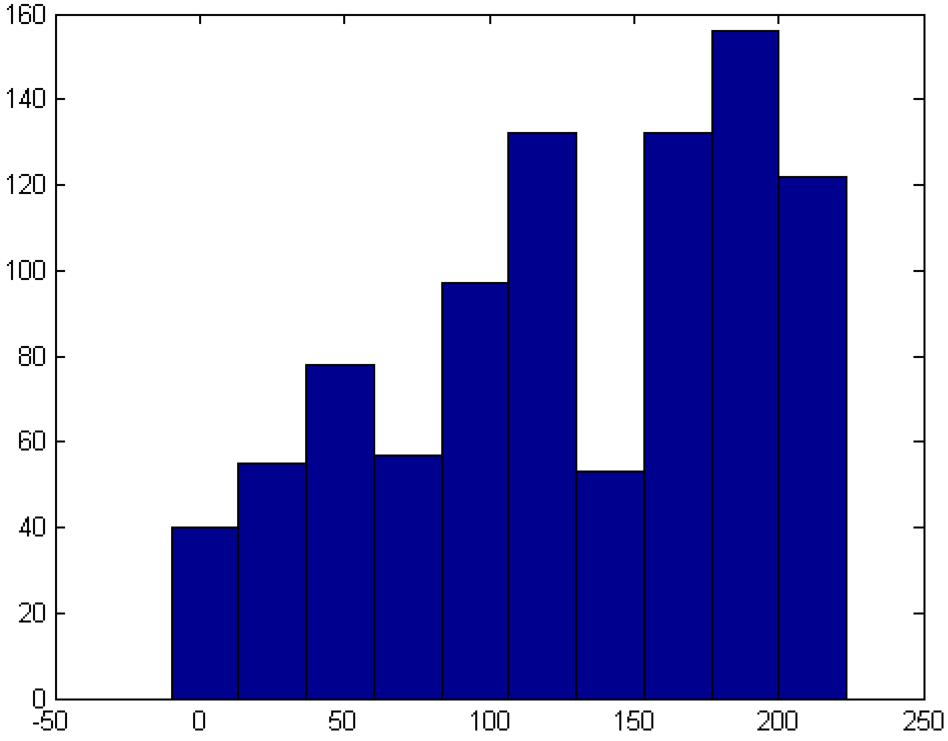
```
Warning: Image is too big to fit on screen; displaying at 17%

The average number of frequencies that have met the threshold per second i
Warning: Image is too big to fit on screen; displaying at 17%
The average number of frequencies that have met the threshold per second i
Warning: Image is too big to fit on screen; displaying at 17%
The average number of frequencies that have met the threshold per second i
Warning: Image is too big to fit on screen; displaying at 17%
The average number of frequencies that have met the threshold per second i
Warning: Image is too big to fit on screen; displaying at 17%
The average number of frequencies that have met the threshold per second i
The average number of frequencies that have met the threshold per second i
```




10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Positive Frequencies of the Song
0
2
4
6
8
10
12
14
Time (ms)







Published with MATLAB® R2014a