FPGzAm

Song identification system

Yafim Landa

Pranav Sood

FPGZAM LISTENS TO AUDIO AND TELLS YOU WHETHER IT'S A SONG IT KNOWS ABOUT.

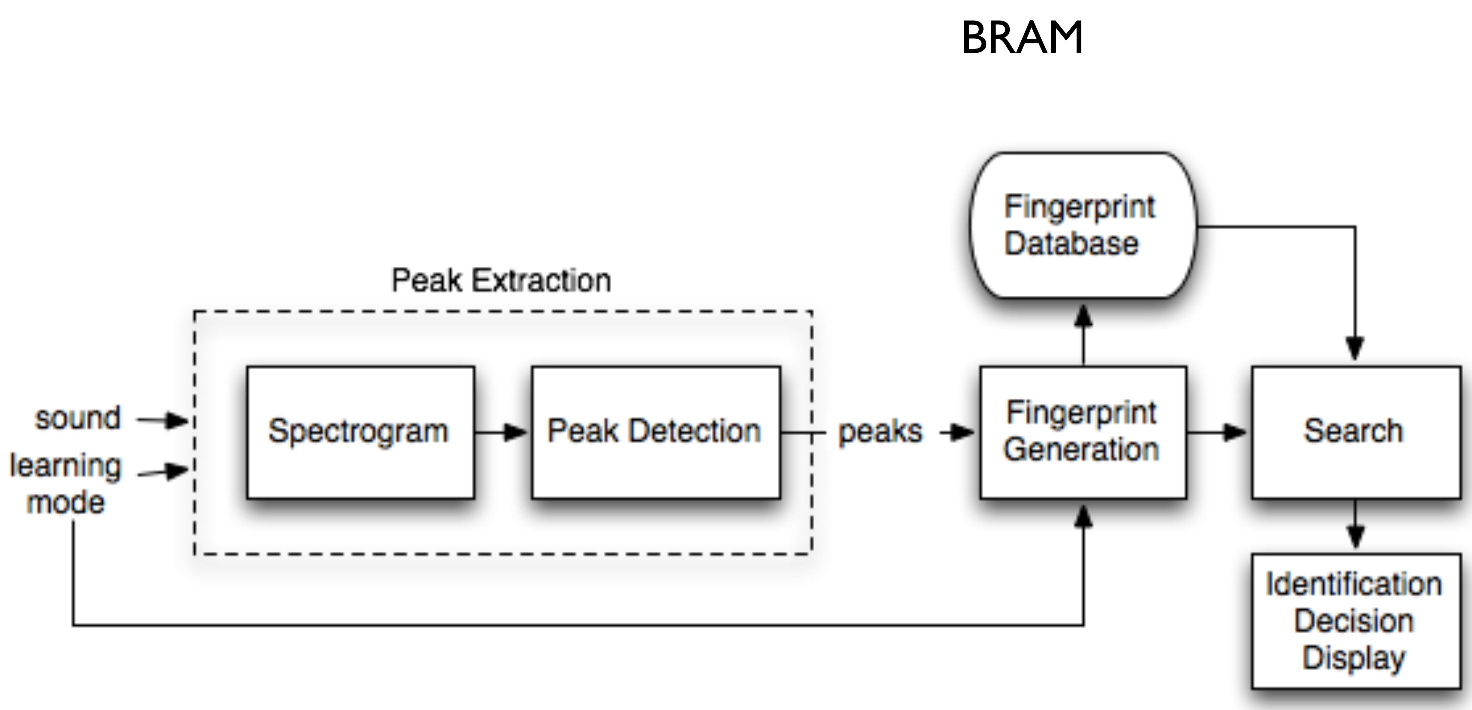
Overview

1. Teach the system a set of known songs
   - For example, a Pink Floyd album
2. Let the system listen to an unknown piece of music
3. If the unknown piece of music is from the known set of songs, tell us more about it

Behavior: Two Modes

- Learning mode
  - Listen to known audio from an MP3 player through the AC97 codec
  - Pass through peak detection and store the peaks in BRAM
- Recognizing mode
  - Listen to unknown audio from a microphone through the AC97 codec
  - Pass through peak detection and search

Block Diagram

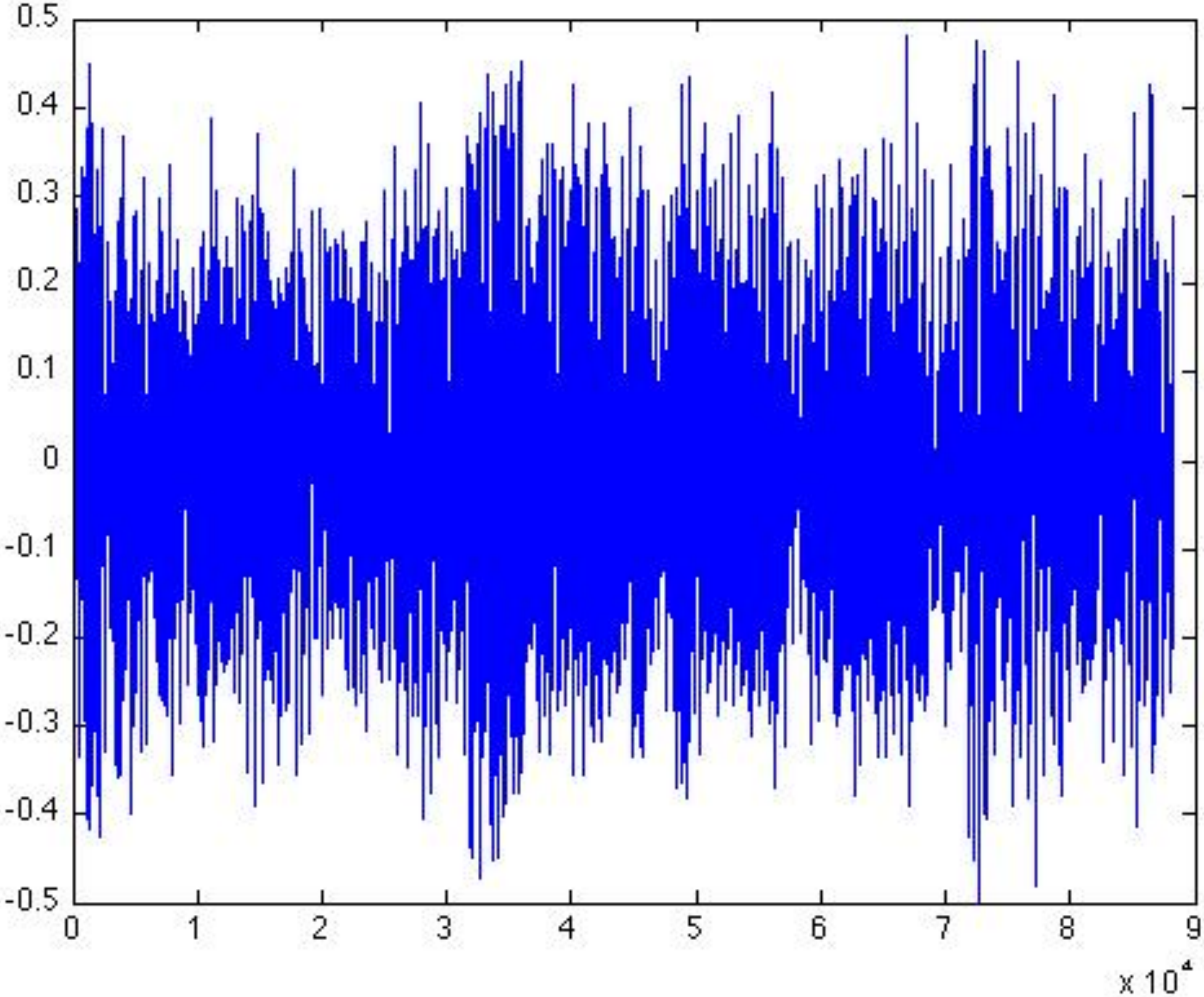


Peak Extraction

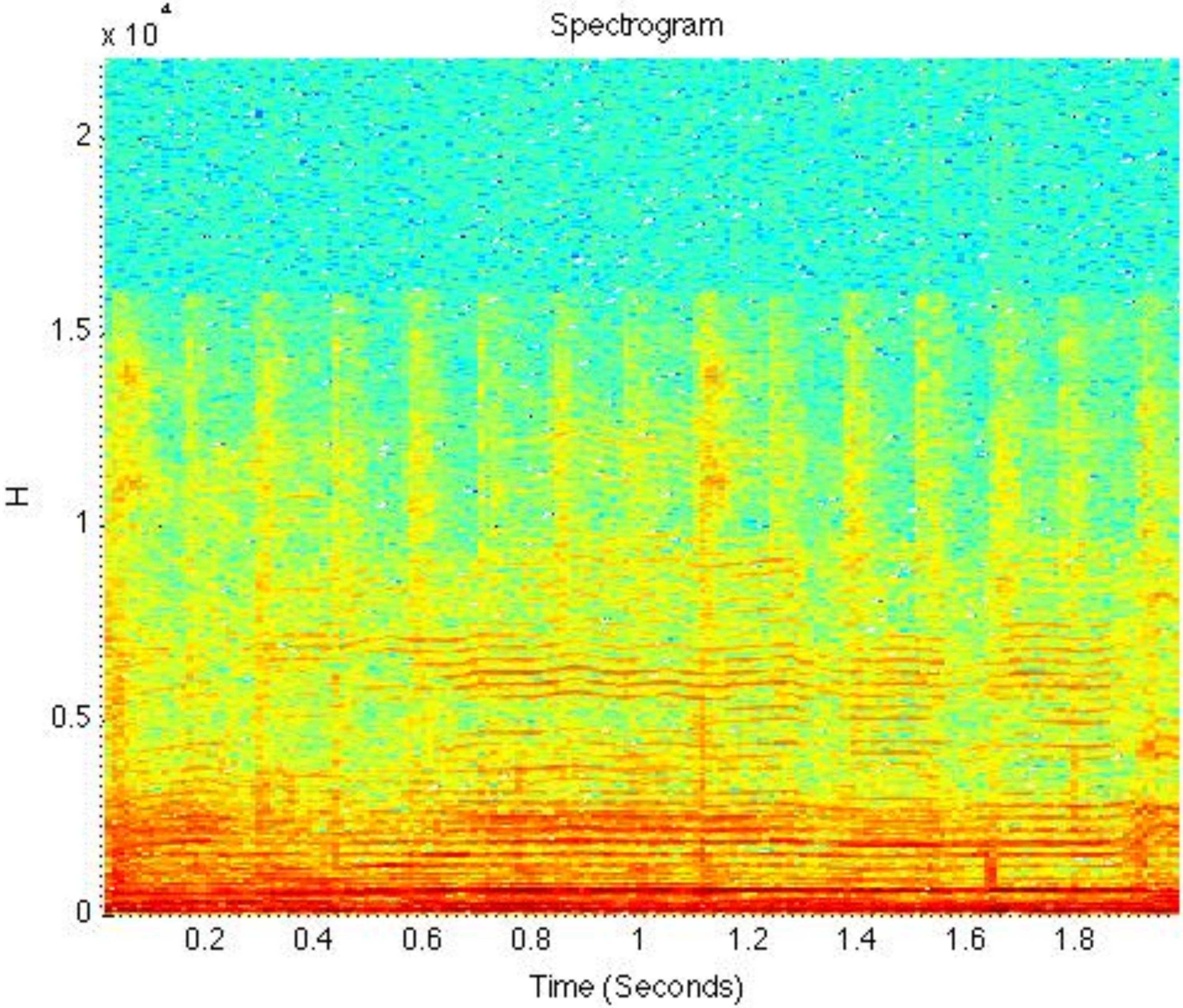
- Sound is sampled at 48kHz with 8bits/ sample
- Create a spectrogram using FFTs
  - Unscaled, pipelined FFT
  - F (frequency , time) = intensity
  - 1024 window size, 50% overlap
  - 48,000/512 ≈ 90 windows/second

Peak Detection

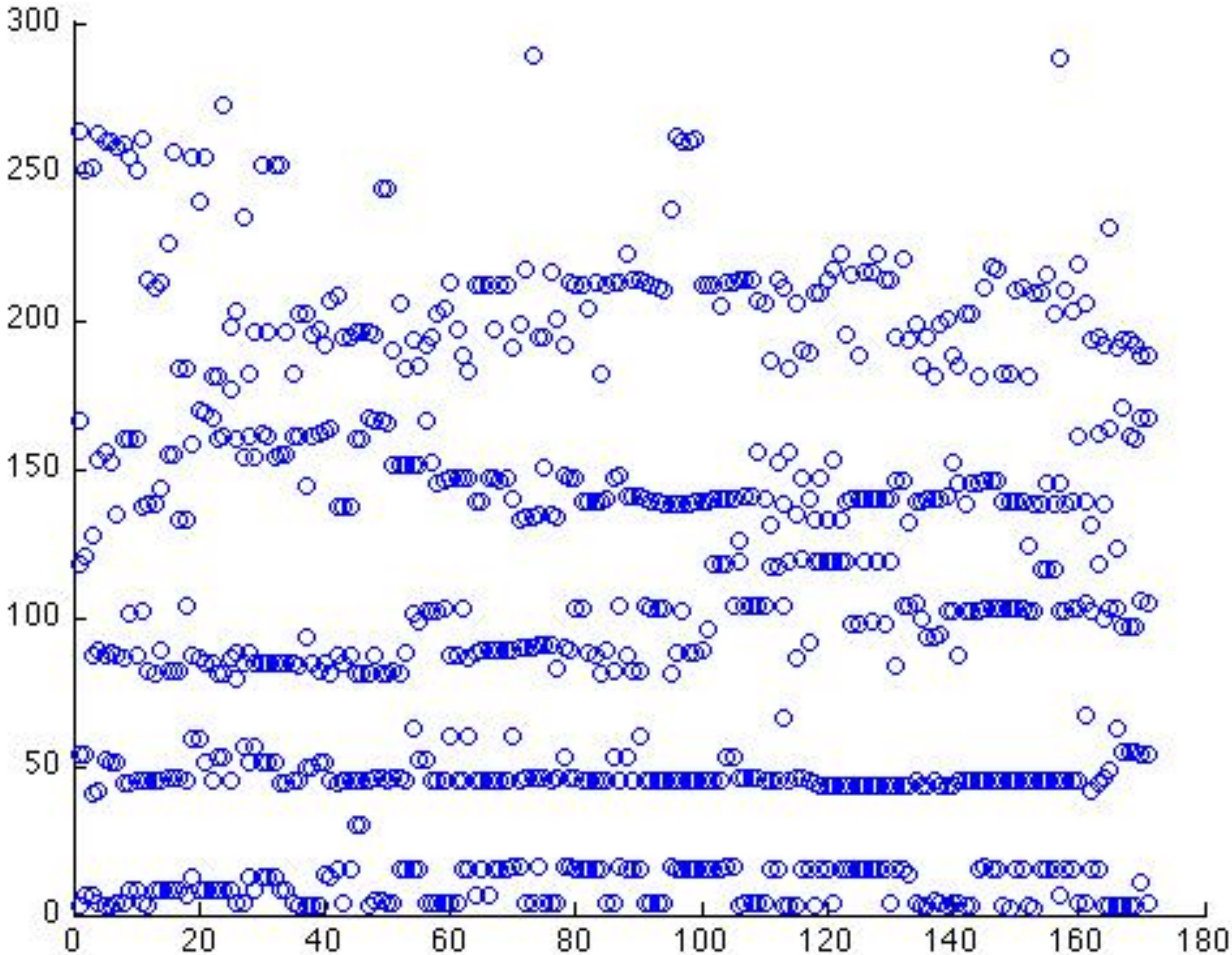
- We chose five frequency ranges to focus on
  - 0 to 40Hz, 40Hz to 80Hz, 80Hz to 120Hz, 120Hz to 180Hz and 180Hz to 300Hz
- Look at the spectrogram for each time window
  - Extract the maximum frequency from each range
  - Record these five numbers in the BRAM
- Memory for 2-second song, ¼-second clip
  - 10kbits for the peaks
  - 2kbit for the clip



0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
1
2
3
4
5
6
7
8
9
x 10^4



Spectrogram
x 10^4
2
1.5
1
0.5
0
H
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
Time (Seconds)

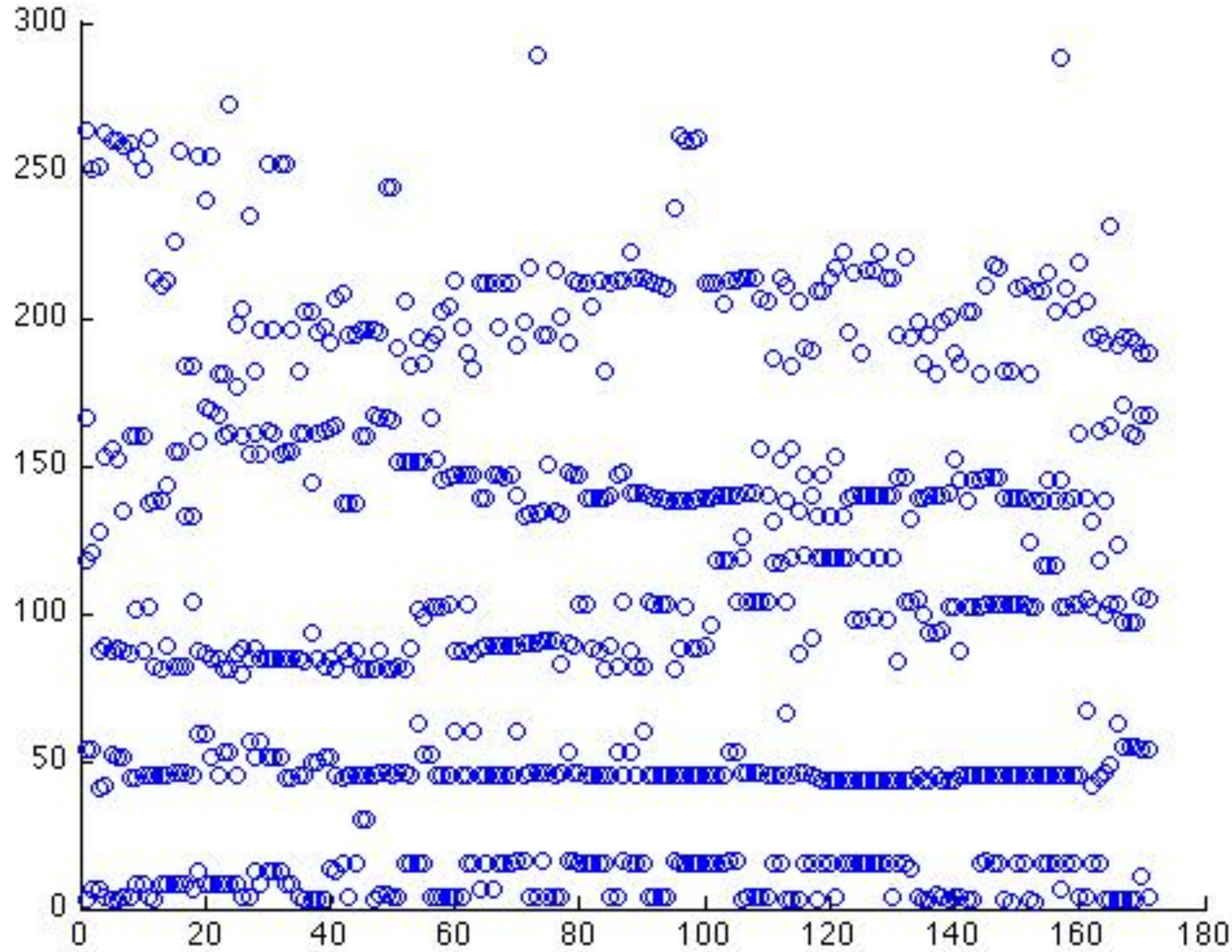


300
250
200
150
100
50
0
0
20
40
60
80
100
120
140
160
180

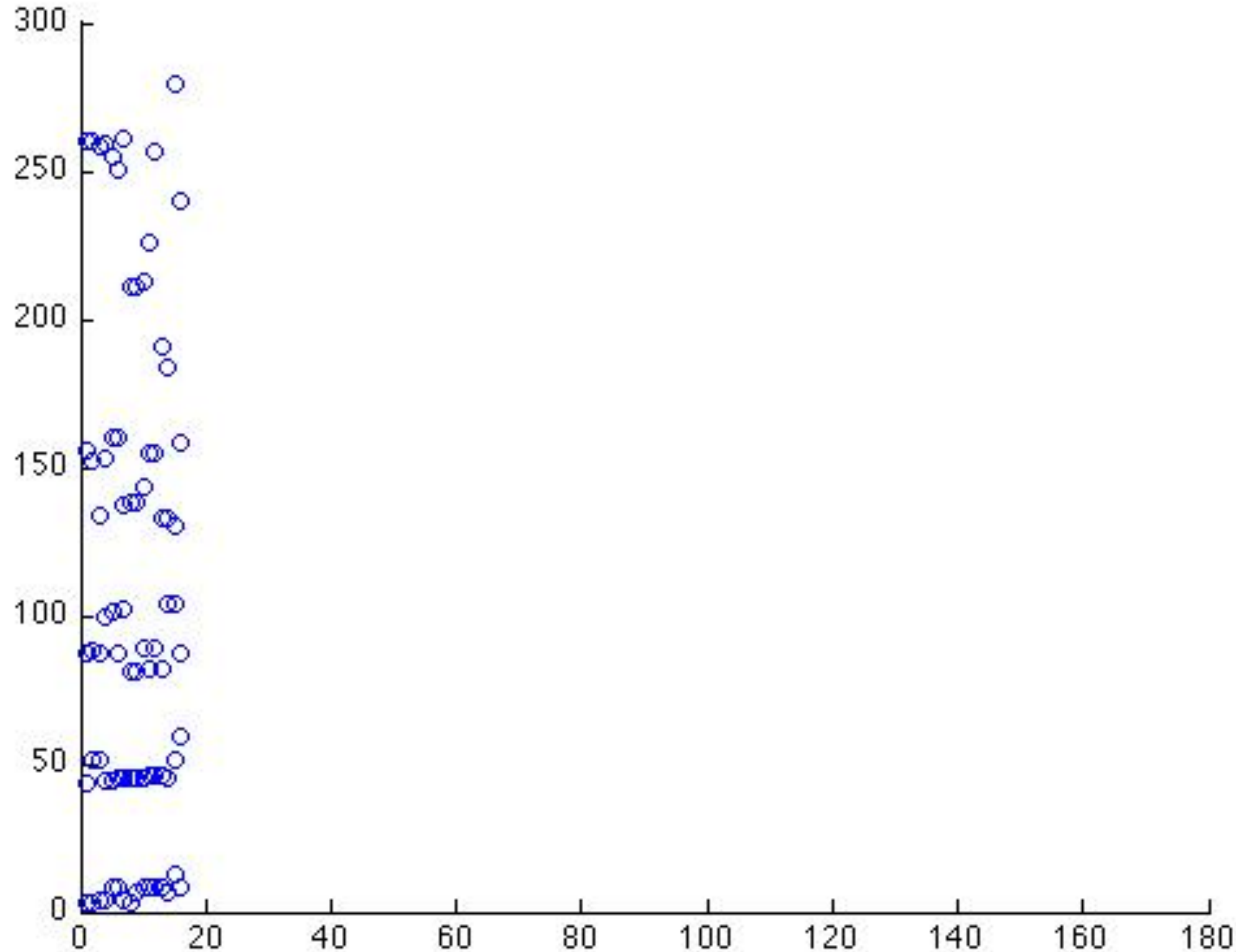
Search

- Received peaks from peak extraction
  - Song: 180 windows, 5 peaks/window = 900 values
  - Clip: 22 windows, 5 peaks/window = 110 values
- Want to check whether the clip belongs to the song
- Strategy: find the best match for the clip offset within the song

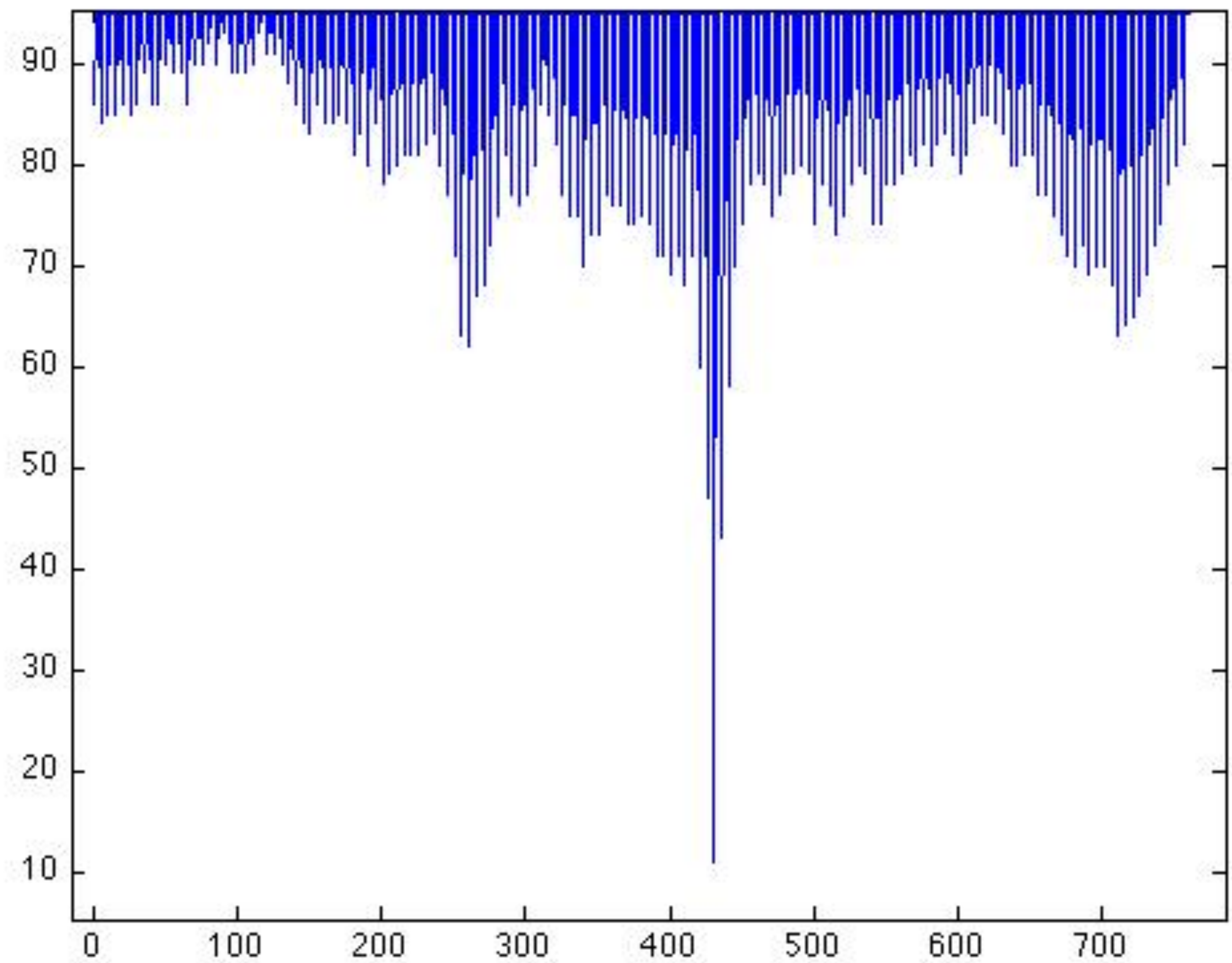
- Call the frequency peaks vector YSong and YClip
  - $[f1_{t=1},f2_{t=1},f3_{t=1},f4_{t=1},f5_{t=1}, f1_{t=2},f2_{t=2},f3_{t=2},\ldots]$
- Go through the whole song and compare the Y vectors
- At each offset, calculate the difference between the song and the clip
  - sum(abs(sign(YSongClip - YClip)))
- Possibly shift YClip for higher accuracy



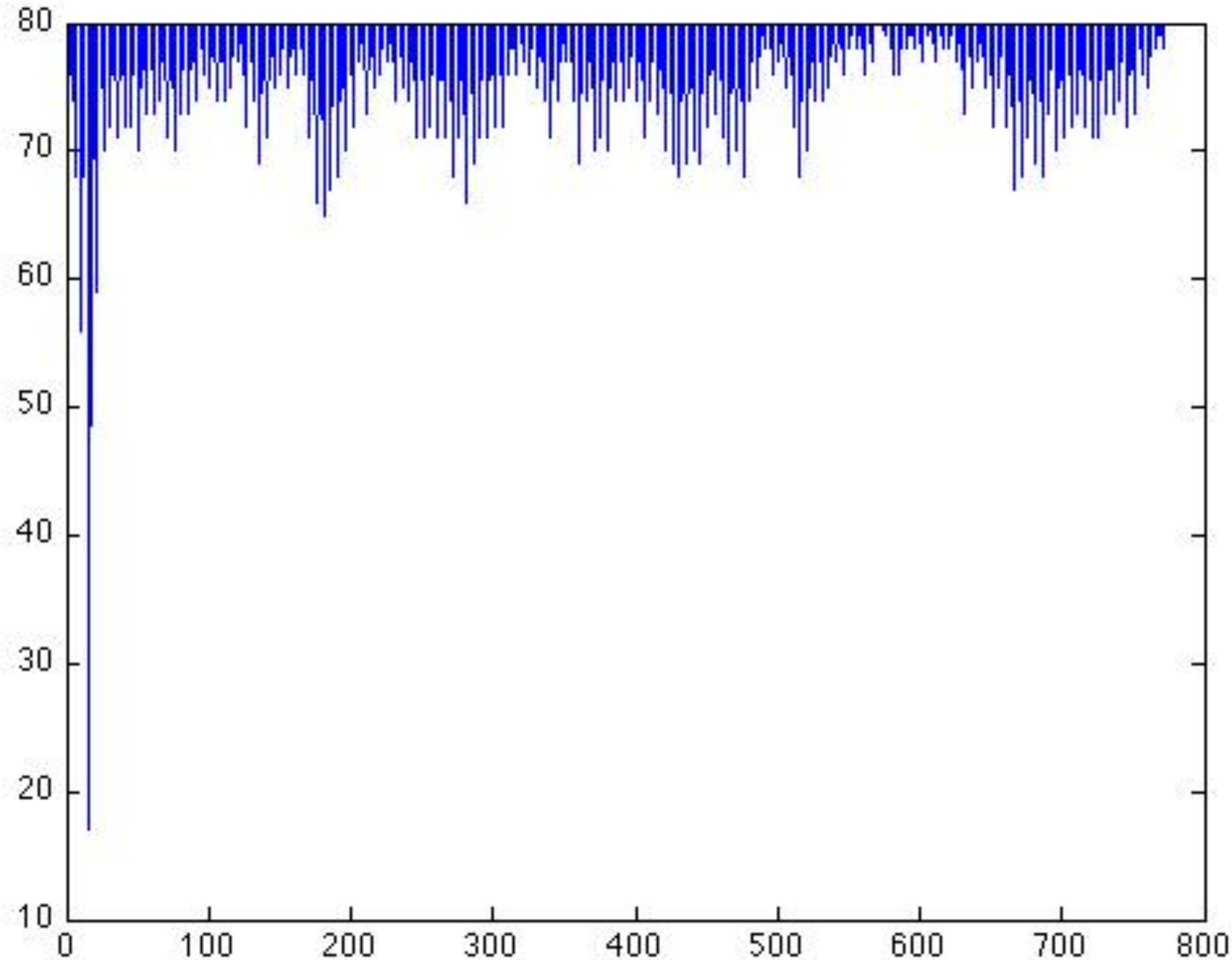
300
250
200
150
100
50
0
0
20
40
60
80
100
120
140
160
180



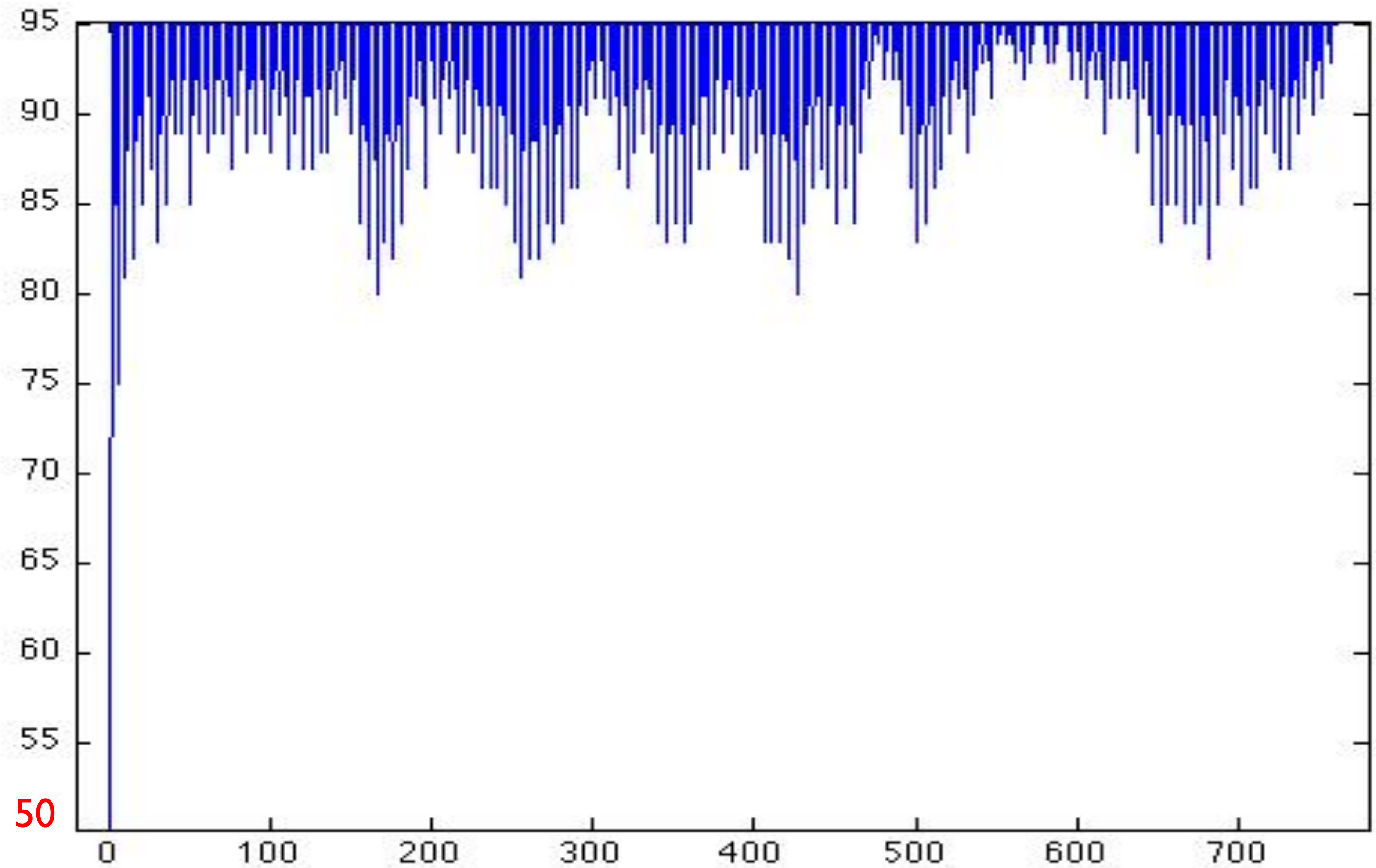
300
250
200
150
100
50
0
0
20
40
60
80
100
120
140
160
180



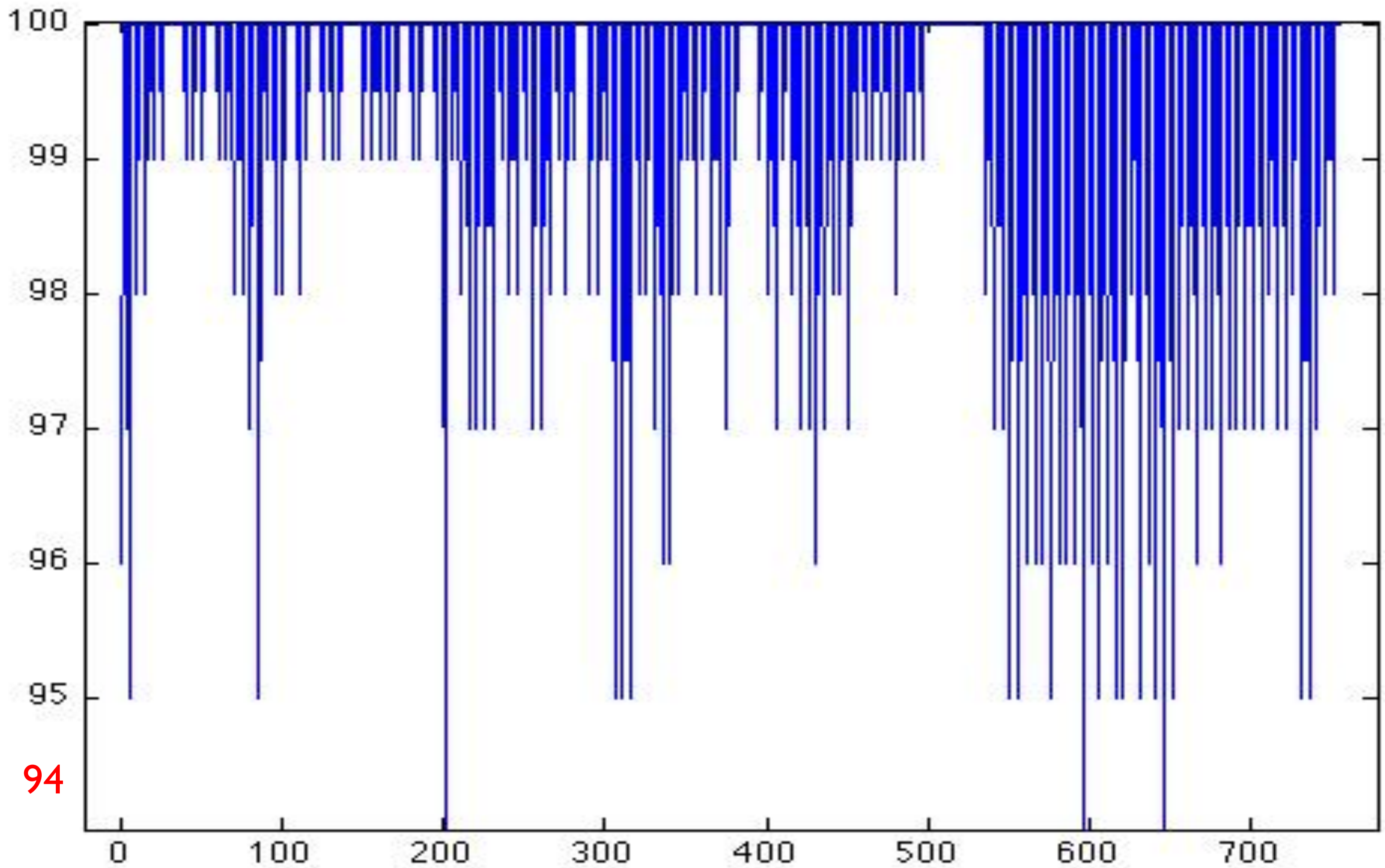
90
80
70
60
50
40
30
20
10
0
100
200
300
400
500
600
700


80
70
60
50
40
30
20
10
0
100
200
300
400
500
600
700
800

Noisy Sample



Mismatch



Shifting Clip

- We may want better precision
  - We can get the error margin further down by using a shifting clip
- Lower delta for a match, same delta for a mismatch
  - Stays at ~90 for a mismatch
  - Goes from ~10 to 0 for a perfect match
  - Goes from 50 to 30 for a noisy match

Timeline

- Week 1
  - Read previous work
  - Plan out an approach to the problem
- Week 2
  - Begin making the FFT module
  - Simulate the project in MATLAB
- Week 3 (now)
  - Begin making the search algorithm in ModelSim
  - Finish making the FFT module
- Week 4
  - Finish the search algorithm in ModelSim and begin porting to the labkit
  - Finish peak extraction
- Week 5
  - Debug both peak/search independently on the labkit
- Week 6
  - Debug interconnection between the two modules
  - Write the user interface on the labkit