Algorithm: denoise based on property of overtone

Let's call it "overtone-based denoise"

# Assumption:

# 1. noise do not overflow signal everywhere

# 2. (two times fundamental frequency) < N/2

# 3. Highest peak is at either fundamental frequency or the 1st overtone

**procedure** OBD(A) # input is an array which is magnitude of DFT

N = length(A);

L = ceiling(N/2);

Mark = array of zeroes with length L;

ave = sum(A) / N;

count = 5;

Find = 0;

# if after 5 search, no suitable peak is found: exit and report exception

while ((there is no 1 in Mark) && (count > 0)):

{

count--;

p = searchPeak(A, 1, L); // circular-even

# now we find the highest peak

range = interval[p-10, p+10];

pf = searchPeak(A, 0.5\*range);

if A(pf) > sqrt(ave \* A(p)):

# the peak is at the 1st overtone

range = interval[pf-10, pf+10];

if 3\*p < L: # see if we can search on

p2 = searchPeak(A, 3\*range);

if A(p2) > sqrt(A(p) \* ave): # still, use amplitude at p to check

for (i = 1; pf \* i < L; i++) :

Mark(searchPeak(A, i\*range));

Find = 1;

ff = pf; # fundamental frequency

else:

Mark(pf) = -1;

Mark(p) = -1;

else: # if we can't, then guess we find it.

Mark(pf);

Mark(p);

Find = 1;

ff = pf;

else:

# the peak is at the fundamental frequency

p1 = searchPeak(A, 2\*range); # check if its 1st overtone exists

if A(p1) >= sqrt(ave \* A(p)): # if so

if 3\*p < N/2:

p2 = searchPeak(A, 3\*range);

if A(p2) > sqrt(A(p1) \* ave):

for (i = 1; p \* i < L; i++) :

Mark(searchPeak(A, i\*range));

Find = 1;

ff = p;

else:

Mark(p) = -1;

else:

Mark(p) = 1;

Mark(p1) = 1;

Find = 1;

ff = p;

else: # this is not funamental fequency

Mark(p) = -1;

}

if (Find == 0):

output("No legal fundamental frequency found.");

return;

Find all corresponding peaks on the other side;

keep = array of 0 with size N;

For all peaks P:

eps = findSpread(P, A, ff);

for (i = P - eps; i <= P + eps; i++):

keep(i) = 1;

For i = 1 to N;

if (keep(i) ! = 1)

A(i) = 0;

output("Procedure complete.")

return;

end

**procedure** searchPeak(A, Mark, init, end)

# only for array with all elements nonnegative

# suppose no 2 elements have same value

max = 0; # value

maxI = -1; # index

for (i = init; i < end; i++):

if (Mark(i) != -1):

if (A(i) > max):

max = A(i);

maxI = i;

return maxI;

end;

**procedure** findSpread(p, A, ff)

# each peak is at least ff's indices away from each other

ave = sum(A) / length(A);

totalA = sum(A(ceiling(p-0.5\*ff) : ceiling(p+0.5\*ff)) - ave);

# area under curve with average deducted

eps = 1; # epsilon

AP = 0;

for (eps = 1; AP < 0.9 \* totalA; eps++):

AP = sum(A(p-eps : p+eps) - ave);

return eps;

end;