Proceedings of the Institute of Acoustics

THE AUTOMATIC RECOGNITION OF MUSICAL INSTRUMENTS

Z A TSERETELI

TBILISI, REPUBLIC OF GEORGIA

1. INTRODUCTION

It is well known, that every musical instrument has its own timbre by means of which man distinguishes this instrument from others. Can we make the computer do the same? In other words, can we make it automatic the process of recognition? Let's specify, what it means "automatic recognition of musical instrument"? It means to create such a computer program, which will be able to detect whether or not the given instrument sound in the given interval of music.

One willn't be able to recognize musical instrument, if he doesn't already know how does it sound. Likewise, the spectrum of the note must previously be established (in order to make computer "know" what the timbre of this note looks like), then the spectrum of the musical fragment must be obtained and finally, these two spectra must be compared each to other in order to detect, whether or not the given note sound in the given fragment. It's obvious, that if the problem is solved for a single note, one can easily fulfil the recognition of whole instrument by considering each possible note the instrument can play.

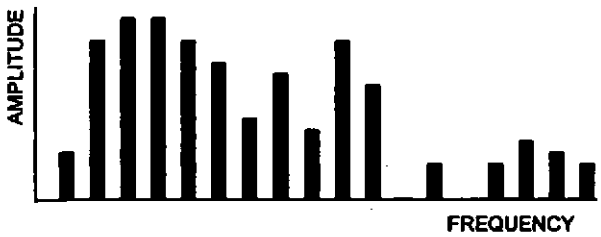
In this paper we describe the algorithms, which provide the establishment of spectra of the note and musical fragment, comparison and final decision. Experiments, which caused the creature of these algorithms and computer programmes, where fulfilled by us on the sounds of piano. It should be noted, that if we consider spectra of musical sounds as they are usually represented in the literature in musical acoustics (see fig. 1; for processing purposes it is more convenient to represent this information as the sequence of pairs, where first components are frequencies and second - amplitudes), we'll see that two different notes of the same instrument don't look alike at all [2]. That is why, as soon as we've got convinced, that program had successfully decomposed the cord played on piano into single notes, we made conclusion that this method will work in the case of any other instrument and the group of different instruments, when playing simultaneously. We took in account too the fact, that piano notes have much difficult spectra then the notes of other instruments, though this method must be examined in any situation of course. This paper is only the first attempt (as far as I am informed) of solving this problem, hence there are a lot of questions in it to be improved and investigated in future.

2. OBTAINING THE SPECTRA

According the Fouriet theorem, every periodic function can be represented as the sum: $A_0+A_1\sin(\omega+\varphi_1)+A_2\sin(2\omega+\varphi_2)+\ldots+A_k\sin(k\omega+\varphi_k)+\ldots$, where ω is the frequency, which is related with period as following: $\omega=1/T$ [1]. Components of this sum are called harmonics. We call A_i the amplitude of i-th harmonic, φ_i - the phase. The frequency of i-th harmonic is $i\omega$. Let's call ω the main frequency. Sequence of pairs $(\omega,A_1), (2\omega,A_2)\ldots (k\omega,A_k)$. is called the spectrum (note that we need to operate only on finite sums).

The experiments were fulfilled on the digital records made on 44.1 kHz frequency. It means 44100

Fig. 1



entries per second. Consider some fragment of the record which possesses N entries. Assume, that these are values of some discrete function f and let's extend this function using the formula: $f(x+N)=f(x)$. Thus we'll obtain some periodic oscillation with some frequency. In the case when the digital record is made on 44.1 kHz, the frequency of this oscillation, measured in Hz, will be 44100/ N Hz.

As we've already noted, first we must obtain the spectra of the given note of given instrument and the musical fragment. Of course, we must take this latter not very big. On the other hand, it mustn't be very small, because as we saw, the frequency of the fragment depends on its length: small fragments have big frequencies. Frequencies if high harmonics are multiples if the main frequency. Thus if the fragment is small, its spectrum is poor with harmonics and recognition of the instrument from such spectrum is much difficult, than from the rich one. It reminds the natural case: if the duration of the musical interval is very small, man is unable to recognize instruments, until he listens the fragment if enough duration. Here we have the first problem: how to select N - duration of the musical interval to be considered? It is the subject of further investigations. For the time being, it's obvious that we must take same N both for note and musical fragment, because in this case harmonics from these two spectra to be compared, will be of the same frequencies.

Let's take 5512 as the value of N. Then the frequency of the interval will be 44100/5512. It is about 8 Hz. Our choice is motivated by the fact, that interval with duration of 1/8 of second is quite small and at the same time has enough length for recognition. In other words, the spectrum with main frequency $\omega=8$ is rich enough for automatic recognition. It's quite sufficient to consider our problem in the range 8-8000 Hz, so the value of k, the number of spectral components, must be taken equal to 1000.

Now we have complete background to realise the program, based on the well known technique of practical harmonic analysis [1]. On the input we have N values of some function and on the output we obtain k spectral components: pairs consisting of amplitude and frequency of every harmonic.

3. REDUCTION OF THE SPECTRUM

Obtained spectrum of musical fragment is ready for comparison, while the spectrum of the note must be reduced previously. There is no need to operate on the entire set of spectral components of the note, but only on the significant ones.

Amplitude A_i is called maximal, if $A_i > A_{i-1}$ and $A_i > A_{i+1}$. We aren't interested in harmonics with non-

maximal amplitudes. Further, if the amplitude is quite small it mustn't be considered too. We call harmonics significant if appropriate amplitude is maximal and is bigger than some limit of significance.

On the figure 2 you see the entire spectrum of piano note a^1 obtained by the program mentioned in the previous section. Observe, that all "little" amplitudes to be discarded are below the horizontal line signed with "lim" and the frequencies of significant harmonics to be retained are specified below. Value of the limit of significance depends on many factors (f.e. the volume the record is made, force the note is played and ets.) and may vary for particular cases. Although we noticed some relationship between the biggest amplitude and "lim". Further experimentation may clear up the precise nature of this relationship.

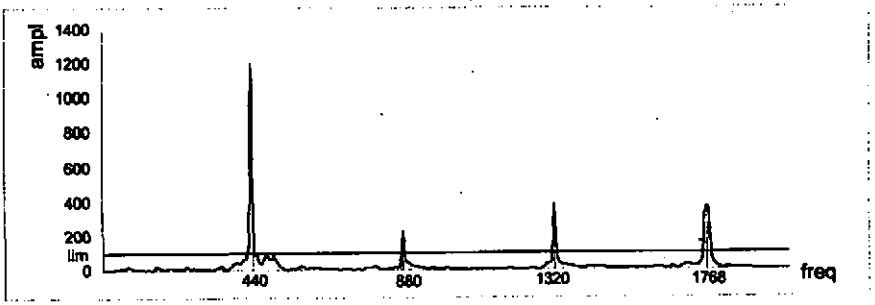


Fig.2

Frequency of the first significant harmonic determines the height of the note (in our example 440 Hz) and other frequencies of reduced spectrum may be deviated from multiples of this frequency as it is evident from the chart. Taking in account all above, we can now realise the automatic reduction of the note spectrum. Such reduced spectrum is ready to be compared with the entire spectrum of musical interval.

4. COMPARISON

Given the reduced spectrum of some note of some instrument and whole spectrum of some musical interval we have to work out the algorithm of comparison and decision: whether or not this note sound in this interval.

The idea is following: 1) consider first harmonic of reduced spectrum and select appropriate (with same frequency) harmonic from musical fragment spectrum. If this latter isn't significant in the sense of previous section, then the note doesn't sound at all; 2) if the harmonic from cord spectrum is significant we have to do following: consider all harmonics from reduced spectrum and select appropriate harmonics from non-reduced one (again there may be some deviations from appropriate frequencies). Now consider the ratios of all possible pairs of amplitudes from reduced spectrum and compare them with appropriate ratios of selected amplitudes. If many of them are close enough with one another, it means that the shape of the note spectrum (see fig. 1) is retained in the cord (with some accuracy of course), so the note sounds. If there are a lot of differences, then we are dealing with some different shape. It may mean that the note really doesn't sound, or its spectrum is distorted by other sounds taking part in the cord.

The meaning of the term "close enough" is that the difference between appropriate ratios is less than some precpecified number. The meaning of the term "many of them" is that the number of ratios which are close enough is more than some given number. These two constants depend on many factors and their universal values must be established under the experiments.

Thus the main idea of comparison is that if the shape of note spectrum is retained in the cord spectrum, the note sounds. The other notes which take part in the cord have own harmonics and some of them may be of the same frequency as the some harmonic from reduced spectrum being considered. So they change values of amplitudes and consequently, the ratios where these amplitudes participate are changed too. That is why we must consider all possible combinations of amplitudes and if we are lucky and many of them are unchanged (or changed a little bit) then the recognition is possible.

In appendix we enclose sample program of recognition based on the algorithm already described. The code is given on the Turbo Pascal language. We divided the range 8-8000 Hz on 28 intervals and specified the value of maximal deviation for each one. This data are placed in the array "devarr". First components are deviations (we took them equal to 1/4 of the tone) and the second components are frequencies. The constant "redharmnum" possesses the maximal number of harmonics in reduced spectrum. Components of the array "mincoin" are minimal numbers of ratios should be in coincidence with the accuracy given in "accur" (f.e. if the reduced spectrum contains 8 spectral components, minimum number of pairs should be in coincidence is 5). "dim" is the number of spectral components in non-reduced spectrum, "lim" is the limit of significance. Spectral components of the fragment are loaded in the file with name "cord1" (assigned with variable "sourcef") and reduced spectrum of the note is loaded in the file named "d" (assigned with variable "sourcen").

	d		CORD1		CORD2	
NN	freq	ampl	freq	ampl	freq	ampl
1	152	929.4992	152	704.1992	152	733.8037
2	296	468.0112	296	345.8308	288	389.5172
3	440	138.2488	432	495.6247	440	1071.619
4	592	391.9101	592	258.1224	592	645.6127
5	736	123.529	736	237.6025	736	99.90437
6	888	66.49587	888	109.6469	880	238.4142
7	1032	148.1926	1032	130.4624	1032	72.72081
8	1328	64.56365	1328	255.9274	1320	276.8962
9	1480	103.2274	1440	159.5261	1480	159.6716
10	1632	123.9852	1632	130.353	1632	62.12057
11	1784	52.81161	1768	94.58027	1784	271.5673
12	1936	57.1374	1928	57.2394	0	57.2394
13	2088	116.6307	2088	72.77373	2088	271.4308
14	2240	93.07915	2240	115.001	2248	59.80568

Table 1

In the table 1 we arrange the spectral components extracted by the recognition program from the piano note *d* and two different cords. In the first cord note takes part, in the other - it doesn't. Number of pairs (1,i), i>1 which ratios coincide with appropriate ones is 3 for cord1. Thus the program continues work and considers pairs (2,i), i>2, then pairs (3,i), i>3 and so on. For cord1 the maximum, which is equal to 8, is reached on pairs (6,i). It is equal to the necessary limit of coincidence (see appendix), so the note sounds in the cord1. As for cord2, one can check that on every stage the result is less than necessary limit, so the note isn't recognized in cord2.

5. RESUME

The process of automatic recognition of musical instrument consists of following stages: establishment of spectra for given note and given musical fragment, reduction of the note spectrum and comparison. There are four constants the result of this process depends on: unit of measure (i.e. duration of interval being considered), limit of significance of harmonics, accuracy of coincidence of ratios and limit of coincidence. Optimal selection of the values for these constants is the subject of future investigations.

Interesting question arises concerning the timbre of instrument: listening two different sounds of same instrument we guess that we're dealing with one instrument, but when looking at their timbres (see fig. 1) they are different. Why? In other words, what is the common characteristic of sounds of musical instrument expressed in numbers? Our ear claims, that such a characteristic exists.

Finally, we list up some possible applications of the method outlined here. First, the program of automatic establishment of scores may be obtained. Further, some visual effects (on the computer screen) may be related with the sounds of different musical instruments making pleasant listening and deeper perception of the piece. Finally, as the vowels in human's speech can be considered as the sounds of some height made by some musical instrument, in our opinion, technique described here may be useful for automatic identification of a person by the timbre of the voice.

6. APPENDIX

```
program recognition;
  { Some strange way of arrangement of operators is caused by the limitations on space }
   const
     devarr:array[0..26,1..2] of integer=((0,0),(1,278),(2,588),(3,880),(4,988),(5,1397),(6,1662),
       (7,1976),(8,2218),(9,2490),(10,2794),(11,3136),(12,3323),(13,3521),(14,3952),(15,4187),
       (16,4435),(17,4699),(18,4979),(19,5275),(20,5588),(21,5920),(22,6272), (23,6645),
       (25,7041),(27,7459),(29,8000));  redharmnum=16;
      mincoin:array[1..redharmnum] of byte=(1,2,2,3,3,4,4,5,5,6,7,7,8,8,9,9);
     dim=1000; note='d';  fragm='cord1'; accur=1; lim=50;
   type maintype=record  ampl:real; freq:integer     end;
   var i,j,k,l:integer;  spfrag:array[1..dim] of maintype; spnote,spcomp:array[1..redharmnum] of
maintype;  coinarr:array[1..redharmnum] of byte; sourcef,sourcen:text; max,v,deviation:byte;
procedure compare;
begin
       l:=1; j:=0;  repeat  j:=j+1;  coinarr[j]:=0;  for l:=i+1 to k do
            if spcomp[l].freq<>0 then              { Non-significant harmonics aren't taken in account }
            if abs(spnote[i].ampl/spnote[l].ampl-spcomp[i].ampl/spcomp[l].ampl)<accur
              then coinarr[j]:=coinarr[j]+1;  i:=i+1
      until (i>k-1) or (coinarr[j]>=k-i)                end;
 begin
assign(sourcef,fragm);                          { Reading the spectrum of the fragment }
      reset(sourcef);  l:=1;  while not eof(sourcef) do
      begin  readln(sourcef,spfrag[i].ampl,spfrag[i].freq);  i:=i+1       end;  close(sourcef);
      assign(sourcen,note);                             { Reading the reduced spectrum of the note }
      reset(sourcen);  readln(sourcen,k);  i:=0;
      while not eof(sourcen) do
      begin  i:=i+1;  readln(sourcen,spnote[i].ampl,spnote[i].freq)    end;    close(sourcen);
```

```
for i:=1 to k do                                    { Selecting appropriate amplitudes }
begin
    j:=spnote[i].freq div 8;
   if (abs(spfrag[j].ampl)>abs(spfrag[j-1].ampl)) and
      (abs(spfrag[j].ampl)>abs(spfrag[j+1].ampl)) and (abs(spfrag[j].ampl)>lim) then
       spcomp[i]:=spfrag[j]              else
     begin
   l:=1;                                            { Taking in account deviations }
  while (l<=26) and (spnote[i].freq>=devarr[l,2]) do l:=l+1; deviation:=devarr[l,1];
       v:=0;  l:=1;
       repeat
          if (abs(spfrag[j-l].ampl)>abs(spfrag[j-l-1].ampl)) and
          (abs(spfrag[j-l].ampl)>abs(spfrag[j-l+1].ampl)) and (abs(spfrag[j-l].ampl)>lim) then
    begin  spcomp[i]:=spfrag[j-l];  v:=1   end
          else
          if (abs(spfrag[j+l].ampl)>abs(spfrag[j+l-1].ampl)) and
 (abs(spfrag[j+l].ampl)>abs(spfrag[j+l+1].ampl)) and  (abs(spfrag[j+l].ampl)>lim) then
          begin  spcomp[i]:=spfrag[j+l];  v:=1              end;
          l:=l+1  until (l>deviation) or (v=1);
       if v=0 then spcomp[i].freq:=0                { Fixing non-significant harmonics }
     end      end;
   if spcomp[1].freq=0 then.                        { First harmonic isn't significant }
   writeln('Note '+note+' does not sound in '+fragm+' at all!')
   else
   begin  compare;
          max:=coinarr[1];                          { Counting the pairs being in coincidence }
          for i:=2 to j do if coinarr[i]>max then max:=coinarr[i];
    if max>=mincoin[k] then writeln('Note '+note+' sound in '+fragm+' : ',
    'm.c.=',mincoin[k],', r.c.=',max)
          else writeln('Note '+note+' not recognized in '+fragm+' : ',
                'm.c.=',mincoin[k],', r.c.=',max)          end
end.
```

7. REFERENCES

[1] G M FIKHTENGOLTS, 'Course of Differential and Integral Calculus', vol. 3, "NAUKA", Moscow 1966
[2] L A KUZNETSOV, 'Acoustics of Musical Instruments', "LEGPROMBITIZDAT", Moscow 1989