**Класс NoiseDetection**

***Описание констант***

|  |  |
| --- | --- |
| **Константа** | **Описание** |
| MAX\_DATA\_SIZE 20000 | Максимально допустимый размер сигнала (в отсчетах) |
| const int SAMPLE\_RATE = 16000 | Частота дискретизации звуковых данных (используется в функции filtr) |
| const int VAR\_BLOCK\_SIZE = 186 | Длина блок анализа сигнала в отсчетах (~12мс, используется в функции checkWithVariationZCR) |
| const int MIN\_PERIOD = 43 | Минимально возможная длина квазипериода в отсчетах (~3мс, используется в функции checkWithPeriod) |
| const int MAX\_PERIOD = 145 | Максимально возможная длина квазипериода в отсчетах (~9мс, используется в функции checkWithPeriod) |
| const int TH\_PERIOD\_LEN = 75 | Порог минимально допустимой длины для голосового квазипериода (используется в функции checkWithPeriod) |
| const int TH\_PERIOD\_NUM = 7 | Минимальное количество подряд идущих голосовых квазипериодов для определения присутствия голоса в сигнале (используется в функции checkWithPeriod). |
| const float TH\_ZCR = 0.06 | Порог минимального значения относительной частоты перехода через 0 для блоков анализа (используется в функции checkWithVariationZCR) |
| const int TH\_VARIATION\_BAND\_1 = 95 | Порог значения вариации для сигнала обработанного полосовым фильтром с полосой пропускания 100-500Гц (используется в функции checkWithVariationZCR) |
| const int TH\_VARIATION\_BAND\_2 = 20 | Порог значения вариации для сигнала обработанного полосовым фильтром с полосой пропускания 1-10Гц (используется в функции checkWithVariationZCR) |
| const float TH\_VAR\_PART\_1 = 0.81 | Первое минимальное значение отношения количества блоков анализа, на которых превышен порог TH\_VARIATION\_BAND\_1, к общему количеству блоков анализа для сигнала обработанного полосовым фильтром с полосой пропускания 100-500Гц. (используется в функции checkWithVariationZCR) |
| const float TH\_VAR\_PART\_2 = 0.4 | Второе минимальное значение отношения количества блоков анализа, на которых превышен порог TH\_VARIATION\_BAND\_1, к общему количеству блоков анализа для сигнала обработанного полосовым фильтром с полосой пропускания 100-500Гц (используется в функции checkWithVariationZCR) |
| const float TH\_VAR\_PART\_3 = 0.1 | Третье минимальное значение отношения количества блоков анализа, на которых превышен порог TH\_VARIATION\_BAND\_2, к общему количеству блоков анализа для сигнала обработанного полосовым фильтром с полосой пропускания 1-10Гц. (используется в функции checkWithVariationZCR) |
| const float TH\_ZCR\_PART = 0.4 | Минимальное значение отношения количества блоков анализа, на которых превышен порог TH\_ZCR, к общему количеству блоков анализа (используется в функции checkWithVariationZCR) |

***Описание функций***

***Функция void normVol(unsigned char data[],unsigned int dataSize)*** - увеличивает уровень звукового сигнала data до максимально возможного значения без появления искажений. Находится максимальное значение отсчета по всему входному сигналу.

for(unsigned int i = 1; i < dataSize;i++)

if (data[i] > max) max = data[i];

Если максимум больше уровня 0 (128 для 8 бит без знака), то каждый отсчет сигнала умножается на 127/max с учетом знака и преобразуется в 8-битное целое без знака.

if (max < 255 && max > 128)

{

for(unsigned int i = 0; i < dataSize;i++)

{

norm = (float)(data[i] - 128) \* 127.0 / (float)(max - 128);

if (norm > 127) norm = 127;

if (norm < -128) norm = -128;

norm += 128.0;

data[i] = (unsigned char)norm;

}

}

***Функция int checkWithPeriod(unsigned char data[], unsigned int dataSize)*** - проверка сигнала data на присутствие в нем голоса с помощью вычисления длины квазипериода.

Для вычисления длины квазипериода можно использовать массив величин

,

где *n0* – номер отсчета цифровых данных, начиная с которого выполняются рассчеты, *x* – отсчет сигнала, MIN\_PERIOD ≤ k≤ MAX\_PERIOD (при используемой частоте дискретизации 16000Гц значения для MIN\_PERIOD *= 43,*MAX\_PERIOD *= 145*).

k = MIN\_PERIOD;

while(k < MAX\_PERIOD)

{

for(int i = 0;i < k;i++)

{

if ((pos+i+k) >= (dataSize - 1)) break;

L[k-MIN\_PERIOD] += abs(data[pos+i] - data[pos+i+k]);

}

k++;

}

Затем определяется значение mini, при котором величина *L* принимает минимальное значение.

unsigned int min = L[0];

mini = 0;

for(int i = 1;i < MAX\_PERIOD-MIN\_PERIOD;i++)

{

if(min > L[i])

{

min = L[i];

mini = i;

}

}

periodLen = mini+MIN\_PERIOD представляет собой длину квазипериода. Принимая конец найденного квазипериода за начало следующего, находится длина второго квазипериода и так далее Параллельно производится подсчет максимального количества идущих подряд квазипериодов (maxPeriodNum) с длиной большей либо равной константе TH\_PERIOD\_LEN.

Если в записанных данных обнаружится не менее TH\_PERIOD\_NUM подряд идущих квазипериодов с длиной большей либо равной константе TH\_PERIOD\_LEN (предположительно данные содержат речь) функция возвращает 1, иначе возвращается 0.

***Функция int sgn\_8b\_unsigned(unsigned char &x)*** – возвращает -1 если аргумент x < 128, иначе 1. Используется для подсчета количества пересечений сигналом нулевого уровня.

***Функция void filtr(int N,int F1,int F2, unsigned char data[], unsigned int dataSize) -*** обработка сигнала data цифровым полосовым фильтром с полосой пропускания от F2 до F1 Гц, N – порядок фильтра.

***Функция*** ***int checkWithVariationZCR(unsigned char data[], unsigned int dataSize)*** – определяет, является ли неголосовой сигнал посторонним шумом или полезным сигналом с помощью вычисления уровня вариации и частоты пересечения уровня 0.

Вначале по исходному сигналу вычисляется относительная частота пересечения сигналом уровня 0 на последовательных блоках исходного сигнала длиной N =VAR\_BLOCK\_SIZE отсчетов:



где x – отсчет сигнала, k – номер блока сигнала, N-количество отсчетов в блоке,



Одновременно выполняется подсчет количества блоков для которых эта величина превышает порог TH\_ZCR:

for (int i = 0;i < blockNum;i++)

{

sumZCR = 0;

for (int j = (i\*VAR\_BLOCK\_SIZE); j < ((i + 1)\*VAR\_BLOCK\_SIZE - 1); j++)

sumZCR += abs(sgn\_8b\_unsigned(data[j + 1]) - sgn\_8b\_unsigned(data[j]));

if ((float)sumZCR / (2 \* VAR\_BLOCK\_SIZE) > TH\_ZCR) zcrOverNum++;

}

Затем выполняется обработка исходного сигнала цифровым полосовым фильтром с полосой пропускания 100-500Гц.

filtr(100,500,100,data,dataSize);

На каждом блоке обработанного фильтром сигнала вычисляется значение вариации:



где x – отсчет сигнала, k – номер блока сигнала, N-количество отсчетов в блоке. Одновременно выполняется подсчет количества блоков, на которых эта величина превышает порог TH\_VARIATION\_BAND\_1:

for (int i = 0;i < blockNum;i++)

{

sumVar = 0;

for (int j = (i\*VAR\_BLOCK\_SIZE);j < ((i+1)\*VAR\_BLOCK\_SIZE - 1);j++)

sumVar += abs(data[j+1] - data[j]);

if (sumVar >= TH\_VARIATION\_BAND\_1) varOverNum++;

}

Вычисляются отношения количества блоков анализа, на которых превышен порог TH\_VARIATION\_BAND\_1 и порог TH\_ZCR, к общему количеству блоков анализа:

float varRate = (float)varOverNum/blockNum;

float zcrRate = (float)zcrOverNum/blockNum;

Если величина varRate > TH\_VAR\_PART\_1 или если varRate > TH\_VAR\_PART\_2 и zcrRate > TH\_ZCR\_PART, то функция checkWithVariationZCR возвращает значение 1.

if (varRate > TH\_VAR\_PART\_1)

{

return 1;

}

else

{

if (varRate > TH\_VAR\_PART\_2 && zcrRate > TH\_ZCR\_PART)

return 1;

}

Если оба условия не выполняются, то дополнительно производится обработка исходного сигнала цифровым полосовым фильтром с полосой пропускания 1-10Гц и вычисляется вариация c подсчетом количества блоков для которых значение вариации превышает порог TH\_VARIATION\_BAND\_2:

varOverNum = 0;

for (int i = 0; i < blockNum; i++)

{

sumVar = 0;

for (int j = (i\*VAR\_BLOCK\_SIZE); j < ((i + 1)\*VAR\_BLOCK\_SIZE - 1); j++)

sumVar += abs(dataOrig[j + 1] - dataOrig[j]);

if (sumVar >= TH\_VARIATION\_BAND\_2) varOverNum++;

}

Вычисляется отношение varRate количества блоков анализа, на которых превышен порог TH\_VARIATION\_BAND\_2, к общему количеству блоков анализа:

varRate = (float)varOverNum/blockNum;

Если величина varRate превышает значение TH\_VAR\_PART\_3 функция checkWithVariationZCR возвращает значение 1, иначе 0.

***Функция int checkNoise(unsigned char data[], unsigned int dataSize)*** – определяет, является ли заданный сигнал полезным сигналом (функция возвращает 1) или посторонним голосовым или неголосовым сигналом (возвращает 0). Вначале выполняется нормализация громкости:

normVol(data,dataSize);

Нормализованный сигнал проверяется на присутствие в нем голоса. Если в сигнале присутствует голос, то считается, что сигнал является посторонним и функция возвращает 0. В противном случае сигнал проверяется функцией checkWithVariationZCR и функцияcheckNoise возвращает результат этой проверки.

if (checkWithPeriod(data,dataSize) == 1)

return 0;

else

return checkWithVariationZCR(data,dataSize);