УДК 004.4

Дахин Антон Викторович, магистрант,
Белгородский государственный национальный исследовательский университет, Белгород

Михелев Владимир Михайлович, к.т.н.,
Белгородский государственный национальный исследовательский университет, Белгород

РАЗРАБОТКА И ТЕСТИРОВАНИЕ АЛГОРИТМА ШУМОПОДАВЛЕНИЯ НА ОСНОВЕ СОВРЕМЕННЫХ WEB-ТЕХНОЛОГИЙ

Аннотация: В работе рассматривается процесс разработки и тестирования алгоритма шумоподавления на основе динамической компрессии и энергетической детекции голосовой активности с применением современных стандартов web-разработки для обработки звука.

Ключевые слова: Обработка звука, Шумоподавление, Web-разработка, JavaScript, Компрессия, Детекция голосовой активности.

Современные информационные системы для голосового и текстового общения играют ключевую роль в обеспечении эффективной удалённой коммуникации. Такие системы должны сочетать высокую функциональность, удобство пользовательского интерфейса и обработку аудиосигналов в реальном времени. В данной статье рассматривается вопрос реализации алгоритма шумоподавления для такой информационной системы, использующего современные web-технологии.

Современные подходы к подавлению шума можно разделить на следующие группы:

1. *Спектральное вычитание.* Этот метод основывается на оценке спектра шума и его вычитании из спектра входного сигнала [1] и крайне эффективен для стационарного шума. При работе с речью могут возникать искажения.

2. Фильтрация на основе адаптивных алгоритмов. Этот метод основан на алгоритмах, которые подстраиваются под характеристики шума [2]. Такой метод хорошо подходит для изменяющихся условий, но требует больших вычислительных ресурсов.

3. Методы на основе машинного обучения. Нейронные сети могут эффективно разделять речь и шум. Но итоговые модели требуют больших вычислительных затрат для использования;

4. Динамическая компрессия и энергетическая детекция голосовой активности. Данный подход сочетает анализ энергии сигнала для детекции голосовой активности с динамическим усилением для подавления шума. Особенностью метода является низкая вычислительная сложность, что позволяет использовать его в реальном времени для обработки звука.

В рамках статьи для реализации будет выбран алгоритм, основанный на принципе динамической компрессии и детекции голосовой активности. Ключевым фактором для выбора является низкая вычислительная сложность, которая позволит обрабатывать звук в реальном времени на любом типе устройств и отсутствие искажений в речи.

Первой важной задачей в разрабатываемой алгоритме является определение речи. Способ её детекции основан на анализе энергии сигнала и описывается следующими формулами:

$$summ = \sum_{i=0}^{window} x_i^2 \quad (1.1)$$

$$rms = \sqrt{\frac{summ}{window}} \quad (1.2)$$

$$energy = 20 * log_{10}(rms + 10^{-10}), \quad (1.3)$$

где x_i – это амплитуда сигнала в момент времени, $window$ – размер окна обработки. Энергия $energy$ в децибелах отражает громкость сигнала и используется для будущего сравнения с заданным порогом:

$$hasVoice = energy > NoiseFl + (EnergyTh - NoiseFl), \quad (1.4)$$

Данный метод определение речи можно улучшить и обновлять порог с помощью экспоненциального сглаживания, если энергия сигнала ниже указанного минимального порога:

$$coef = e^{-\frac{1}{vadRate*sampleRate}} \quad (1.5)$$

$$NoiseFl = NoiseFl * coef + energy * (1 - coef) \quad (1.6)$$

где $time$ – время сглаживания, $sampleRate$ – частота дискретизации.

Второй важной задачей в разрабатываемой алгоритме является подавление шума. Для начала выполняется вычисляется уровень сигнала в децибелах:

$$level = 20 * log_{10}(|sample| + 10^{-10}) \quad (1.8)$$

Расчёт усиления определяется в зависимости от вычисленного уровня сигнала и статуса голосовой активности. Если речь не обнаружена, исходный сигнал уменьшается на 90%. В случае, если речь обнаружена, выполняется плавный переход по следующей формуле:

$$targetGain = ratio + \frac{(1-ratio)*(level-(noiseLevel+offset))}{knee} \quad (1.9)$$

где $offset$ – это фиксированное смещение, обеспечивающее расстояние между уровнем шума и полезным сигналом, $noiseLevel$ – уровень шума. Такой подход называется мягким коленом (soft knee). Основная его задача заключается в реализации плавного перехода между сигналом и подвергающимся компрессии сигналами.

Заключительным этапом подавления шума выполняется сглаживание усиления. Это позволяет избежать артефактов при резких изменениях параметров:

$$gain = gain * coef + targetGain * (1 - coef), \quad (1.10)$$

где $coef$ – коэффициент атаки или затухания, зависящий от направления изменения усиления.

Подавления шума можно улучшить добавлением адаптивного вычисления уровня шума в тех местах, где детектор речевой активности не обнаружил речь:

$$coef = e^{-\frac{1}{noiseRate*sampleRate}} \quad (1.11)$$

$$noiseLevel = noiseLevel + coef + level * (1 - coef) \quad (1.12)$$

Для обработки аудиосигналов в современных веб-приложениях используется AudioWorklet, являющаяся частью Web Audio API. Особенность этого способа заключается в том, что вычисления выполняются в отдельном потоке, минимизируя задержки и снижая нагрузку на основной поток браузера. Полученный в рамках статьи AudioWorklet, реализующий алгоритм шумоподавления на основе метода динамической компрессии и детекции голосовой активности показан на листинге 1.

Листинг 1

AudioWorklet для шумоподавления

```
    this.releaseCoef = this.calcCoef(this.release);
    this.updateCoef = this.calcCoef(this.updateRate);
    this.gain = 1; this.windowSize = 128;
    this.energyThresh = -35; this.noiseFloor = -30;
    this.vadCoef = this.calcCoef(0.1);
    this.isVoice = false;
    this.buffer = new Float32Array(this.windowSize);
```

```
    this.index = 0;
    this.threshold = this.noiseLevel + this.offset;
  }
  calcCoef(time) {return Math.exp(-1 / (time * sampleRate));}
  updateVAD(sample) {
    this.buffer[this.index] = sample;
    this.index = (this.index + 1) % this.windowSize;
    if (this.index === 0) {
      let sum = 0;
      for (let val of this.buffer) sum += val * val;
      const rms = Math.sqrt(sum / this.windowSize);
      const energy = 20 * Math.log10(rms + 1e-10);
      if (energy < this.noiseFloor + 10) {
        this.noiseFloor = this.noiseFloor * this.vadCoef +
          energy * (1 - this.vadCoef);
      }
      this.isVoice = energy > this.noiseFloor +
        (this.energyThresh - this.noiseFloor);
    }
  }
  calcGain(level) {
    if (!this.isVoice || level < this.threshold) return this.ratio;
    return level < this.threshold + this.knee ?
      this.ratio + (1 - this.ratio) * (level - this.threshold) /
this.knee : 1;
  }
  updateGain(target) {
    const coef = this.gain < target ? this.attackCoef : this.releaseCoef;
    this.gain = this.gain * coef + target * (1 - coef);
  }
  process(inputs, outputs) {
    const input = inputs[0],output = outputs[0];
    for (let ch = 0; ch < input.length; ch++) {
      for (let i = 0; i < input[ch].length; i++) {
        const sample = input[ch][i];
        this.updateVAD(sample);
        const level = 20 * Math.log10(Math.abs(sample) + 1e-10);
        if (!this.isVoice) {
          this.noiseLevel = this.noiseLevel * this.updateCoef +
            level * (1 - this.updateCoef);
          this.threshold = this.noiseLevel + this.offset;
        }
        this.updateGain(this.calcGain(level));
        output[ch][i] = sample * this.gain;
      }
    }
    return true;
  }
}
registerProcessor('noise-processor', NoiseProcessor);
```

Для оценки эффективности алгоритма шумоподавления применяются такие метрики оценки, как Среднеквадратичная ошибка (Mean Squared Error, MSE) и Отношение сигнала к шуму (Signal-to-Noise Ratio, SNR).

Среднеквадратичная ошибка представляет собой среднеквадратичное отклонение между значениями эталонного сигнала и сигнала, подвергнутого обработке алгоритмом шумоподавления [1]. Эта метрика отражает усреднённую величину квадрата ошибок по всем временным отсчётам, позволяя оценить точность восстановления сигнала. Меньшее значение этой метрики свидетельствует о увеличении эффективности алгоритма. Среднеквадратичную ошибку можно вычислить с помощью формулы:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i)^2, \quad (4.1)$$

где N – это общее число отчетов сигнала, y_i – амплитуда эталонного сигнала в определённом отсчёте, x_i – амплитуда обработанного сигнала в определённом отчёте.

Отношение сигнала к шуму характеризует отношение мощности полезного сигнала к мощности остаточного шума в обработанном сигнале, выраженное в децибелах (дБ) [2]. Эта метрика отражает способность алгоритма сохранять целостность полезного сигнала. Высокие её значения указывают на эффективное подавление шумовой составляющей, что особенно важно для задач, связанных с восприятием аудиосигналов. Отношение сигнала к шуму можно вычислить с помощью формулы:

$$P_{signal} = \frac{1}{N}\sum_{i=1}^{N} {y_i}^2 \quad (4.3)$$

$$P_{noise} = MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i)^2 \quad (4.4)$$

$$SNR = 10 * log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) \quad (4.5)$$

Для проведения вычислительного эксперимента была подготовлена эталонная запись голоса с минимальным уровнем фонового шума. Эталонная запись дополнялась различными видами шумов в различном процентном отношении относительно максимальной амплитуды сигнала: белым, розовым, синим и сложным шумом езды в автомобиле.

Белый шум – это случайный сигнал с равномерной спектральной плотностью мощности на всех частотах в слышимом диапазоне [3]. Энергия такого шума равномерно распределена по всем частотам, а амплитуда не зависит от частоты. Этот тип шума тестирует равномерную обработку всех частот, выявляя общую производительность алгоритма.

Розовый шум характеризуется спектральной плотностью мощности, обратно пропорциональной частоте [3]. В таком шуме низкие частоты содержат больше энергии, чем высокие, что делает звук более мягким и глубоким по сравнению с белым шумом.

Синий шум имеет спектральную плотность мощности, пропорциональную частоте, что означает увеличение энергии с ростом частоты [3]. Такой шум Данный вид шума оценивает точность обработки высокочастотных компонентов, критичных для чёткости звука.

Сложные шумы – это шумы, которые не имеют точной формулы получения. Большинство шумов, окружающих нас, можно назвать сложными. В рамках вычислительного эксперимента в качестве сложного шума будет использовать автомобильный шум, взятый с набора NoiseX-92.

Полученные в результате выполнения вычислительного эксперимента данные о Среднеквадратичной ошибке (MSE) и об Отношении сигнала к шуму (SNR) показаны на рисунке 1.

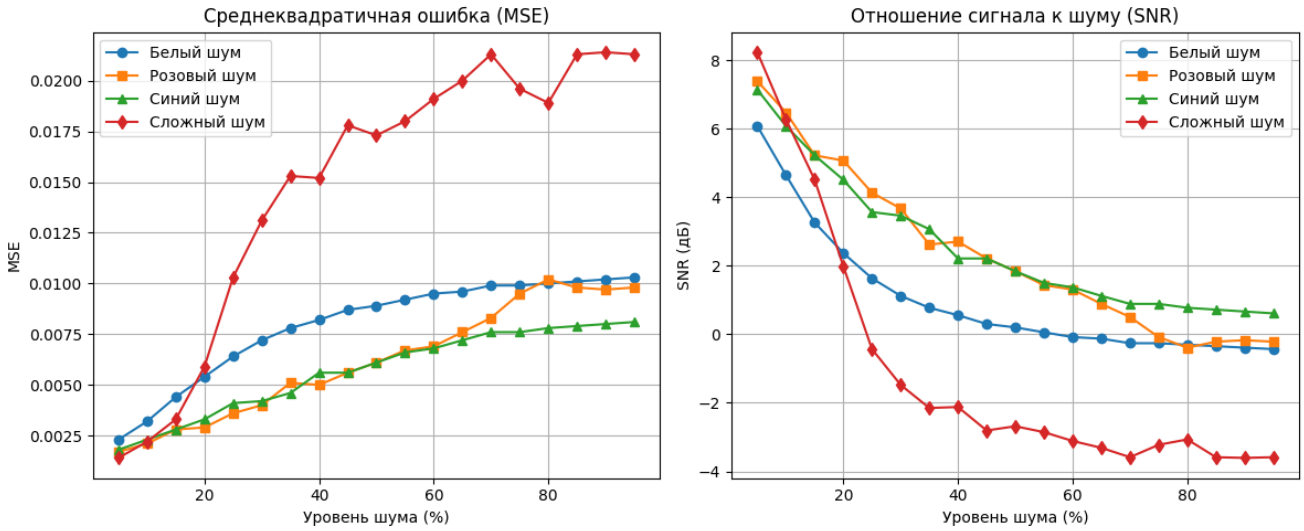

Рис. 1 Диаграмма вариантов использования

Разработанный алгоритм шумоподавления демонстрирует высокую эффективность при низком и среднем уровнях зашумления для различных типов шума, обеспечивая низкий MSE и положительный SNR, что сохраняет разборчивость речи. Для белого шума эффективность сохраняется до 50% зашумления, для розового — до 60%, для голубого — до 70%, а для сложного шума — до 20%. Однако при высоких уровнях зашумления производительность алгоритма резко падает: SNR становится отрицательным, а MSE растёт. В результате появляются артефакты, искажающие речь, и сбои в определении речи, что значительно ухудшает качество обработки звука и разборчивость речи.

Список литературы:

1. Айфичер Э. С., Джервис Б. У. Цифровая обработка сигналов: практический подход / Э. С. Айфичер, Б. У. Джервис ; пер. с англ. — 2-е изд. — Москва : Вильямс, 2004. — 992 с.

2. Ахмад Х. М. Введение в цифровую обработку речевых сигналов: учеб. пособие / Х. М. Ахмад, В. Ф. Жирков; Владимирский гос. ун-т. — Владимир : Изд-во Владим. гос. ун-та, 2007. — 192 с.

3. Оппенгейм, А. Цифровая обработка сигналов : учебник / А. Оппенгейм, Р. Шафер. — 3-е изд., испр. — Москва : Техносфера, 2012. — 1048 с.

4. Документация Web Audio API [Электронный ресурс] — URL: https://www.w3.org/TR/2021/REC-webaudio-20210617/ (дата обращения: 08.06.2025). — Режим доступа: свободный.