# 携帯端末による実測値を用いた音量調整システムの提案と試作

赤池　英夫 *1　角田　博保 *1

## A Proposal and Trial Implementation of Audio Volume Control System Using Actual Values Mesured by Smartphone

Hideo Akaike*1 and Hiroyasu Kakuda*1

**Abstract** – **Nowadays, various multimedia contents are watched by televisions or PCs. But, quality of picture and audio and volume of sound vary per the contents. Especially, due to the diverse range of sound volume, listeners often should adjust it. There are automatic audio volume control features by system level or application software level, although, these might not coordinate with the hearing sense of listeners. So, we propose a system for an audio volume control system based on actual values measured by smartphone putted on listener.**

**Keywords : input device, mobile, wearable, AV system, AGC.**

### 1. はじめに

今日、さまざまな動画コンテンツをテレビや PC で視聴することが一般的である。しかし、コンテンツごとに画質、音質、音量などの異なることがある。そのため、とりわけ音量に応じたボリューム調整がしばしば必要となる。音量調節がシステム (OS) によりサポートされている場合もあるが、実際のスピーカの性能や設置位置などにより音量が変わるうえ、視聴者が一時的に移動することもあり、固定された音量調節では不十分なこともある。つまり、必ずしも実際に視聴者が聴いている音量に連動するとは限らない。そこで、視聴者が普段身に付けていると期待されるスマートフォンで視聴者が聞いている (感じている) であろう音量を計測し、その値にもとづき音を発している機器 (ここでは PC) の音量調整を行なうこととした。

### 2. 関連研究・製品

PC 以前から動画コンテンツはテレビジョンによって提供されている。そこでもコンテンツ毎の音量の違いによるボリューム変更が必要となる。たとえば、Han ら [1] はチャネル間の音量の違いを補償する自動音量制御システムを提案した。

Windows 7 以降には自動音量調整機能が OS レベルあるいはサウンドカードのドライバの設定として提供されており、環境音を考慮したり、通信アクティビティ時にあらかじめ設定された音量に調節される。同様の機能を提供するサードパーティ製ソフトウェアもいくつか存在する (たとえば GoVolume [2])。ただし、実際に視聴者の聞いている音量に必ずしも連動するものではなく、視聴者の移動にも対応しない。また、MacOS の iTunes にはライブラリ内の曲の音量を正規化する機能がある [3]。これは変換はオフラインで行われ、リアルタイムにストリームとして与えられるコンテンツに対するものではない。

ヘッドフォンやイヤフォンを装着し、携帯端末上のコンテンツを視聴する際に、外界音をマイクで拾い、雑音の大きさに応じて音量を調節するアプリケーションがある (たとえば The Volume [4])。本稿で仮定する視聴環境、視聴対象の再生音量の調節に環境音を使うのではなく視聴対象の音自身を用いる点が異なる。

### 3. 提案システム

提案システムの概略を図 1 に示す。

#### 3.1 前提

ひとりで PC 上のマルチメディアコンテンツを視聴しており、これが主たる音源であることを仮定する。また、他の音源からは高々生活騒音程度の音が発生している程度とする (ただし突発的な発生は許容する)。コンテンツとしてはプロフェッショナルによる適切な音量設定のされたものを対象とする。つまり、コンテンツの再生開始時から終了時まで、音量は刻々と変化するものの、突発的な大音量あるいはコソコソ声で聞き取りにくい箇所は製作者の意図・演出によるものであり、それらを無理に調整する必要はない、と考える。なお、以降、上記「コンテンツ」の同義語として「視聴対象」を用いる。

視聴対象の素性 (たとえばドラマか、楽曲か、楽曲であればそのジャンルは何か、など) が分かれば、より

*1: 電気通信大学大学院 情報理工学研究科 情報・ネットワーク工学専攻
*1: Department of Communication Engineering and Informatics, Graduate School of Informatics and Engineering, The University of Electro-Communications.

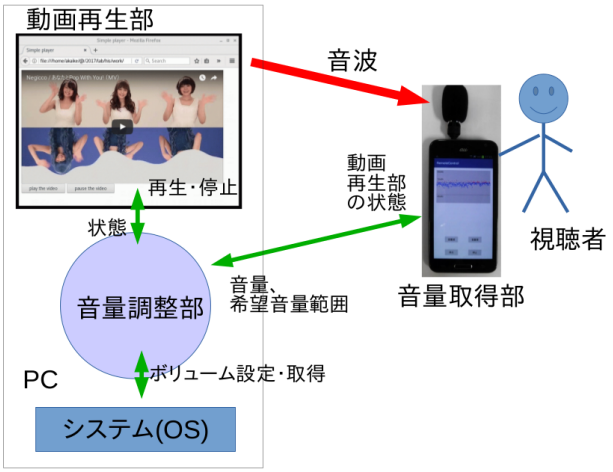


図 1　提案システム概略図
Fig. 1　system overview

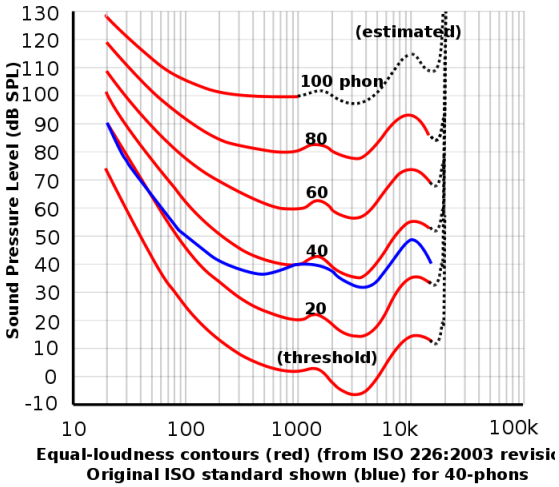


図 2　等ラウドネス曲線 (wikipedia (https://ja.wikipedia.org/wiki/等ラウドネス曲線/) より引用)
Fig. 2　equal-loudless contuours

きめ細やかな扱いが可能と思われる。たとえば MP3 ファイルにおける ID3 タグ [5] のようなメタ情報などは有益である。しかし、ここではそのような詳細については不明であり、もっぱら音量に依るシステムの作成を目指す。

各種の測距センサーを用いたり画像処理などを介して、PC と視聴者の距離を求め、それに応じて音の減衰を考慮して音量調整を行なうことも考えられるが、導入の容易さを鑑み、可能な限り現有のソフト/ハード以外は追加せず、一般的で入手の容易な機材を用いることとした。

### 3.2　音量の取得

スマートフォンのマイクロフォンが大気中を伝わる音圧を対応する電気信号に変換し、適度な増幅を得た後、最終的にソフトウェア的に一定時間間隔でサンプリングされた電気信号の値の列が得られる。これらの値は音圧の程度を表しているが、人間の可聴域 (20Hz ～ 20kHz) における音圧の範囲は広く (20$\mu$Pa ～ 20Pa)、また聴覚を含む人間の感覚量にはヴェーバー・フェヒナーの法則が当てはまるため、音の強さを対数尺度で表わすことが多い。具体的に、音の強弱の尺度として騒音計などで表示される音圧レベルは、基準となる音圧 (20$\mu$Pa) の 2 乗の値と測定された音圧の 2 乗の値の比のデシベル値となっている。

特性にもよるがマイクが捉える音圧は可聴域の周波数に対して一律に扱われる (物理量) のに対し、人間の感じる音圧 (音の大きさ) は周波数によって影響を受ける (心理量)。よく知られている等ラウドネス曲線は、周波数の変化に対し、等しい大きさに聞こえる音圧レベルを等高線として描いたものである (図 2)。とりわけ 40dB、1kHz の純音の音圧レベルを基準とした A 特性による補正を施した音圧レベルが騒音レベルとして扱われ、騒音の基準として使用されている。騒音の目安 (都心・近郊向け、出典「全国環境研協議会　騒音少委員会」) を図 3 に示す [6]。

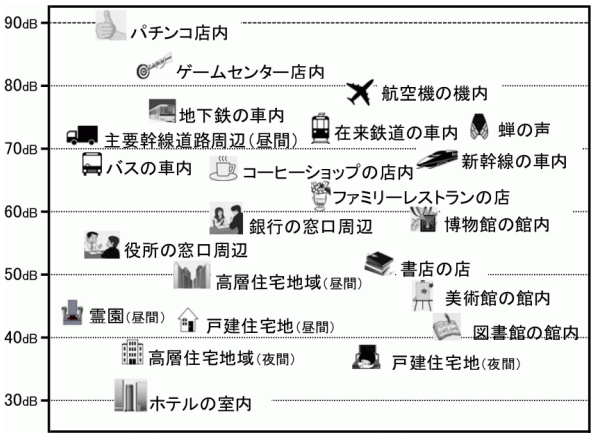


図 3　騒音の目安 (都心・近郊向け)
Fig. 3　noise guideline(for urban)

本提案システムは定量的に騒音の観点から音量を捉えることにした。ただし、視聴者が自ら選択して聞いている音は「不快または望ましくない」ものではないため騒音のイメージはともなわず、あくまで量的な側面からの便宜的な流用である。

### 3.3　音量調節

音量の調節には主にシステムレベルで行なうものとアプリケーションレベルで行なうものがある。前者はシステム全体の音量を決めるマスターボリュームの変更 [1] となり、そのシステムで可能な最大音量まで上

1: 一般的にはマスターボリュームだけでなく、より多岐にわたる入出力レベルの設定が可能と思われる。

げることができる。しかし、システム全体の音量に影響するため、各種イベントの通知音 (たとえば時報やメール着信音など) も意図せず大きく (あるいは小さく) なってしまう。一方、後者は特定のアプリケーションのみの音量調節となり他のアクティビティに対して影響を与えない。しかし、マスターボリュームに上限が抑えられるため調節の範囲が狭められ、希望の音量が出せない場合がある。今回は試験的に前者を採用するものとしたが、まずはアプリケーションレベルで調節し、必要であればさらにシステムレベルで調節することを検討している。

視聴者が希望する音量の範囲を仮定し、「希望音量範囲 (VR)」と呼ぶことにし、範囲の上限音量 ($VR_u$) と下限音量 ($VR_l$) の組で示す。さらに、希望上限音量を越え、速やかに音量調節の必要な限界的な音量を「最大定格音量 ($VR_{max}$)」と呼ぶことにする。これは、不適切なボリューム設定だけでなく、元来コンテンツに内在するもの (たとえばホラー映画での叫び声) もあれば、視聴者によるもの (たとえばくしゃみやせき)、それ以外の外界に由来するもの (たとえば時計のアラーム音や近隣を通過する緊急車両によるサイレン音) などにより発生しうる。

本システムでは、音量を直接設定するのではなく、ボリュームの設定により間接的に音量を調節するため何らかの制御を必要とする。具体的には聞いている音量が希望する範囲を逸脱していれば、ボリュームを上げ下げすることで音量を範囲内に収めるような制御を行なう。具体的な手法として、長く使われ実績のある PID 制御を用いる。

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

ここで $u(t)$ はボリュームの操作量、$e(t)$ は音量の目標値と現在の音量の差、$K_p$、$K_i$、$K_d$ は各項に対する比例定数である。積分および微分は直近の一定期間内に対して行なうものとする。

随時音量および音量の推移により、以下のような分類を行ないそれぞれの制御を行なう。

- ケース 1(音量が $VR_{max}$ を越えた): 強制的にボリュームを絞り、その結果音量が低下したなら、不適切なボリューム設定によるものとみなし、$VR_u$ に速やかに近づけるようにする。これは次の場合とは別系統の制御で実現する (図 4(b))。
- ケース 2(音量が VR 外): VR に収まるよう制御する。$VR_u$ を上回った場合も、$VR_l$ を下回った場合も、ともに希望音量範囲の中央値 ($(VR_u+VR_l)/2$) を目標値とする (図 4(a))。
- ケース 3(音量が VR 内): 大抵の場合、特に制御の必要はないと考えられるが、一定期間 $VR_l$ に近い音量が続くときには、そもそも音量不足であったり、視聴者が音源から遠ざかったことが原因と考えられる。ここではヒューリスティックではあるが、低音量での推移を検出した際、その音量変化の範囲が、それまでの音量変化の範囲と比較し、同程度であれば $VR_u$ を目標値として音量制御を行なうこととした (図 4(c)、(d))。 [2]

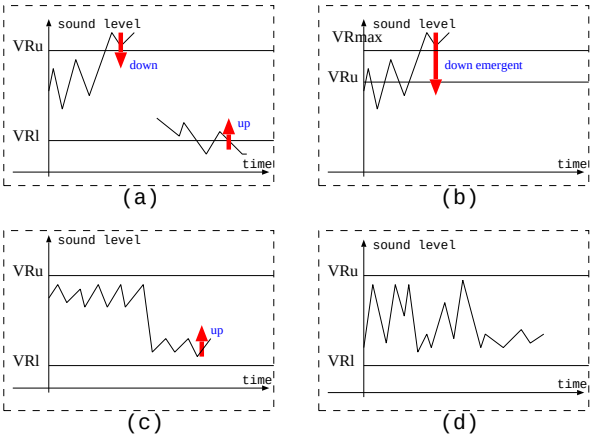


図 4　随時音量および音量推移による分類
Fig. 4　categories by occasional sound volume and sound transition

### 3.4　試作

提案システムの試作を行なった。現時点でデファクトスタンダード的な立場にある YouTube 上のコンテンツを選択した。YouTube はそのコンテンツを Web ブラウザ上の独自のアプリケーションで視聴できるように API を公開している [7]。この API は、視聴対象の選択、再生、停止、早送り、巻き戻し、音量の取得・設定、視聴対象の長さの獲得などの機能を提供している。これにより YouTube を対象とした動画プレイヤーを模擬することが可能である。

音量の取得のためのスマートフォンとして Android バージョン 4.0.4 の搭載された、SAMSUNG 社製 GALAXY SII を用いた [3]。内蔵マイクではゲイン不足が懸念されたため、外付けのエレクトリックコンデンサマイク (全指向性、$-25 \pm 3$dB) を接続した。また、視聴対象のカバー範囲から、サンプリングレート 16,000Hz、モノラル、符号付き 16 ビット PCM としてデータを取得した。

音量 (騒音レベルに連動) を得るために、3.2 節の手順に当たる処理を行なったが、A 特性の補正を行なうために、マイクから得られた音圧に応じた値に短時間 (ハミング窓を使用)FFT を掛け、得られた周波数毎のスペクトラムレベルに A 特性の補正値を加えた (今

2: 不正確な表現ではあるが、それまでの音量変化の範囲 (分散) が大きければ、低音量推移が特別とは扱えないと考えた。
3: 単純に手近にあった機材を利用した。

回用いた標本数から最小周波数分解能は 31.25Hz である)。標準的な方法ではないが補正値の計算に、1/3 オクターブでの A 特性値を FFT の各ビンの周波数に応じて線形補間を行なった。最終的に補正済みの各周波数 $f$ 毎のレベル $L_f$ から、

$$\alpha \times 10\log_{10}(\sum_f 10^{\frac{L_f}{10}}) + \beta$$

として音量を得た。ここで $\alpha$ と $\beta$ は未知定数であるが、別途用意した騒音計 [4] の値にフィットするようトライアンドエラーで決定した。最終的に帳尻合わせであるが、音量 $l$ が 40dBA 以上の場合、40dBA で 1 倍、100dBA で 0.84 倍となるように指数係数 $e^{\frac{(l-40)\times\ln(0.84)}{60}}$ を乗じた。

動画プレイヤーとスマートフォンの仲介を行なう、動画プレイヤーと同一 PC 上で動作する音量調整部は Java で記述した。動画プレイヤーは Firefox 54.0.1 で動作する JavaScript で記述し、音量調整部とは WebSocket で接続した。その音量調整部での受け口には WebSocket の Java 参照実装である tyrus [8] を用いた。測定された音量や動画プレイヤーへの操作 (再生、中断)、音量調整部への各種パラメタ (希望音量範囲、最大定格音量) などは Wi-Fi 経由で音量調整部に送信した。なお、希望音量範囲の下限 ($VR_l$ はその時点での周囲の騒音レベル (いわゆる暗騒音) を基準点としている。これは視聴対象が再生されていない状態で取得された音量に基づいている。

動画プレイヤーおよび音量調整部は FreeBSD 10.3R 上で動作させたため、マスターボリュームの設定には `mixer` コマンドを用いた。PID 制御の 3 種の係数は、恣意的に選んだ適当なコンテンツを用いトライアンドエラーで決定した。

### 3.5 動作検証

試作システムの動作確認を行なった。視聴対象は YouTube のトップページの異なるジャンルの数種を恣意的に選択した。音量制御における係数や、3.3 節のケース 3 のためのパラメタの値に応じて収束時間や振動など挙動は変化するものの、適切な場合には 1 秒程度で希望音量範囲に収まることを確認した。ただし、より詳細な検証は今後の課題である。

## 4. おわりに

本稿では、身に付けた携帯端末 (スマートフォン) で計測された音量の値を用い、PC 上で再生されている視聴対象の音量調節を行なうシステムを提案し、試作について示した。

現時点では、試作システムの簡単な動作確認を行なっただけであり、提案システムを評価するための被験者実験を計画中である。

本提案の実現のためにさまざまなパラメタが存在し、試作システムにおいては手動で試行錯誤して決定した。今後は自動チューニングについても検討する。今回、視聴対象の素性は不明であるとしたが、それが利用できればより適切な音量調節が可能になると期待できる。YouTube を対象とすれば、YouTube data API 経由で視聴対象に関する有益な情報が得られる可能性がある。

現在、視聴者の移動については、音量変化からの類推に過ぎず正確ではない。可能な限り追加の (特殊な) 機材を追加しない方針から、携帯端末のみで実現できる手法を検討している。携帯端末のセンサー情報から所持者の体勢や行動を推定する研究が多くあり、それらからの知見を利用できないかと考えている。実用化の点からは、システム利用者毎のプロファイルを用意したり、システム利用時の履歴の活用 (たとえば何度も再生される同一コンテンツの扱い) が考えられる。

4: 専門家の用いるものではなく廉価版であるが、出荷時に十分な校正がなされていると説明のあったものを使用した。

## 参考文献


[1] Kyu-Phil Han et al., Automatic Volume Control System for compensation of Volume Difference Between TV Channels, IEEE trans. on Comsumer Electronics, Vol. 43, No.4, 1997.

[2] リアルタイム自動音量調整フリーソフト「GoVolume」 - さくら電算 -、`http://www.sakura-densan.com/govolume/`(2017 年 7 月 20 日確認)

[3] iTunes で「音量を自動調整」を使って曲の音量を一定にそろえる、`https://support.apple.com/ja-jp/HT201724`(2017 年 7 月 20 日確認)

[4] ザ・ボリューム:自動音量調整 (The Volume))、`https://play.google.com/store/apps/details?id=com.mightyworksinc.androidapp.mightyvolume&hl=ja`(2017 年 7 月 21 日確認)

[5] The MP3 Tag Standard、`http://id3.org/`(2017 年 7 月 20 日確認)

[6] 環境省ホームページ「一般環境雑音について」 、`http://www.env.go.jp/air/ippan/`(2017 年 7 月 20 日確認)

[7] iframe 組み込みの YouTube Player API リファレンス、`https://developers.google.com/youtube/iframe_api_reference?hl=ja`(2017 年 7 月 20 日確認)

[8] Tyrus - Reference implementation of Java API for WebSocket - JSR 356、`https://github.com/tyrus-project/tyrus`(2017 年 7 月 20 日確認)

[9] YouTube Data API の概要、`https://developers.google.com/youtube/v3/getting-started?hl=ja`(2017 年 7 月 21 日確認)