E-004

モバイル機器用チューナアプリケーションの開発を通した AIR と C 言語ライブラリの連携

Linkage with AIR and C Language Library through the Development of Tuner Application for Mobile Devices.

直江 憲一† 金子 邦彦‡
Kenichi Naoe Kunihiko Kaneko

1．まえがき

軽量で容易に持ち運びできるモバイル機器用アプリケーションの開発では，バイトコードの高級言語が好まれる．モバイル機器用アプリケーション開発分野において，Adobe Systems 社が展開する Flash プラットフォームの一つである Adobe AIR がマルチプラットフォーム展開容易性や情報デザインを意識した開発のしやすさで，モバイル機器の普及とともに注目されている．音楽情報分野のモバイル機器用アプリケーション開発現場においても Roland 社の Air Recorder や V-Drums が Adobe AIR で製作されるなど業界のスタンダートとなっている．高級言語の普及とは裏腹に，高速性を必要とする場合には，依然として C 言語等の低級言語で書かれたライブラリソフトなどとの連携が必要であり，その連携におけるバッファ管理，バイナリデータの変換は依然として重要な研究課題である．音楽情報分野においても，モバイル機器の Adobe AIR 処理系にて周波数解析等のリアルタイム音響信号処理を行う場合，その演算能力に合わせてパフォーマンスチューニングを行う必要がある．本稿では，Adobe AIR 処理系にて動作する単音音源のコア周波数を解析する単純なチューナアプリケーションを制作し，AIR とその内部で実行される C 言語フーリエ変換ライブラリとの連携におけるバッファ管理について，解析結果周波数の正確さと精度の観点で報告する．

2．実験概要

2.1 実験環境

チューナアプリケーションの制作にあたり，AIR 処理系で動作する C 言語フーリエ変換ライブラリとして Scott らの ALF を使用した [1]．しかし ALF は元々録音済みの音声信号をすべてバッファに読み込んだ上で周波数解析を実施するためのライブラリであり，チューナのようなマイクロフォンからリアルタイムに送られてくる音声信号を逐次取得できるような仕様ではなかった．これは，マイクロフォンからのリアルタイムな音声取得が ALF の開発当時 AIR 処理系ではサポートされていなかったためである． AIR 処理系で上記機能が追加された時期は ALF の発表と同時期である．Scott らも将来の展望として述べていることではあるが，この機能を用いれば ALF をリアルタイムに使用することができるようになる．そのため，我々は ALF のプログラムを一部修正しマイクロフォンからの音声信号

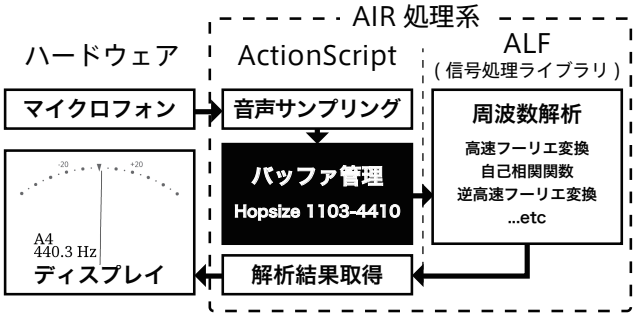

図 1　チューナアプリケーションの構成

をリアルタイムに取得・処理できるようにして使用した．制作したチューナアプリケーションの構成を図 1 に示す．図中の ActionScript とは Adobe AIR 処理系で動作するアプリケーションを制作するためのバイトコード言語である．まず，モバイル機器に搭載されているマイクロフォンセンサより AIR 処理系にて音声信号をサンプリングし，サンプリングした音声信号を ALF で取り扱えるデータ型に AIR 処理系で変換を行う．マイクロフォンとの連携は AIR 処理系の Microphone API を使用し，音声サンプリングは AIR 処理系の SampleDataEvent が発するタイミングで逐次行われる．このタイミングは非公開であり制御もできない．その後バッファ管理を行いながら ALF へサンプリングデータ逐次送り，周波数解析を行う．ALF では周波数[Hz]単位で音高を計測し，計測結果のみを ActionScript で取得した．このサイクルを逐次繰り返してリアルタイムで動作するチューナアプリケーションを実現した．なお，ALF は C 言語で書かれたライブラリであるが，AIR 処理系で動作するために Adobe Alchemy フレームワークによって ActionScript Byte Code(.abc)に変換された上で使用されている．これは C 言語で書かれた数多く存在する財産的アルゴリズムについて実行速度が十分に早い状態で AIR 処理系でも利用できるようにするものである [2] [3]．ただし，ネイティブの C 言語で実行されるよりは実行速度が 10 倍程遅い．しかし，本稿で焦点をあてたのはライブラリとの連携におけるバッファ管理と解析結果の相関であり，フーリエ変換の実行速度は取り扱わない．その他の実験環境を表 1 に示す．

2.2 実験手順

バッファ管理では，ActionScript 部分でサンプリングしたバイト配列データをどれだけ ALF へ渡すかどうかを管理している．このサイズを Hopsize と呼ぶ．Hopsize の単位はバイトであり，式 1 で定義される．

† 九州大学大学院システム情報科学府
‡ 九州大学大学院システム情報科学研究院

$$Hopsize = \frac{サンプリング周波数}{x} \quad \cdots\cdots (1)$$

ただし，x は 11～40 の任意の整数であり，Hopsize の剰余は丸め込まれる．表 1 にもある通り本稿ではチューナアプリケーションのサンプリング周波数は 44100 [Hz]のみを使用した．つまり Hopsize は 1103～4009 の値を取りうる．チューナアプリケーションについて Hopsize のみを変えた状態で Android 端末にデプロイし，簡易なシンセサイザーを用いて 440[Hz]の音を端末のマイクロフォンに入力する．Hopsize が周波数解析結果にどう影響するかを観測する．解析結果は式 2, 3 で示す通りサンプル数 100 の平均値と分散値で評価した．

$$平均値 = \frac{1}{100}\sum_{i=1}^{100} 解析結果周波数_i \quad \cdots\cdots (2)$$

$$分散値 = \frac{1}{100}\sum_{i=1}^{100}\left(Average - 解析結果周波数_i\right)^2 \cdots (3)$$

表 1　実験環境

モバイル機器	Samsung Galaxy SII SC-02C
オペレーティングシステム	Google Android 2.3.3
Adobe AIR Version	3.3.0.3650
Adobe Alchemy Version	0.5a
ALF Version	1.0.2
サンプリング周波数	44100 Hz
音源周波数	440 Hz

3. 実験

表 2 に実験結果を示す．図 2 は実験結果のち Hopsize と解析結果周波数の分散値の関係をグラフにしたものである．表 2 から読み取れる通り，周波数の平均値は 440.3 [Hz]で一貫しており，音源周波数が 440 [Hz]であることから正しい解析結果が得られていることが分かる．0.3 [Hz]の違いは音源の簡易なシンセサイザーの誤差によるものと推察する．一方，周波数の分散値では Hopsize によって違いが見受けられる．分散値は Hopsize が 1000 付近から増える事に小さくなっていき，3150 のとき最も小さくなる．しかし 4000 付近では逆に大きくなってしまっている．いずれの Hopsize でも平均値は同じ値をとっているが，分散値が大きければチューナの針は大きく振れていることを意味し，精度は低い状態である．逆に分散値が小さければチューナの針はブレずに安定しており，精度が高い状態だといえる．この実験結果より，モバイル機器上の AIR 処理系で動作する ALF を使用したフーリエ変換等の信号処理演算を含む周波数解析では Hopsize=3150 のとき解析結果周波数の精度が最良となる．

表 2　Hopsize と周波数解析結果

Hopsize	1103	1521	2100	3150	4009
周波数の平均	440.3	440.3	440.3	440.3	440.3
周波数の分散	2.8	0.8	0.6	0.1	9.7

4. おわりに

解析結果周波数の精度が最も良かった Hopsize=3150 時のアプリケーション内の各処理時間内訳を図 3 に示す．内訳はマイクロフォンがサンプリングデータを取得するたびに起こる処理の内訳であり，各項目は図 1 の各処理と対応している．全体の 60%近くを周波数解析時間が占めており，1 サイクルで 92 [msec]程要している事がわかる．2011 年 10 月に発表された Adobe AIR ネイティブ拡張を利用することで，C 言語のロジックが AIR 処理系外のネイティブ処理系で動作できるようになり，更なる高速化が期待できる．

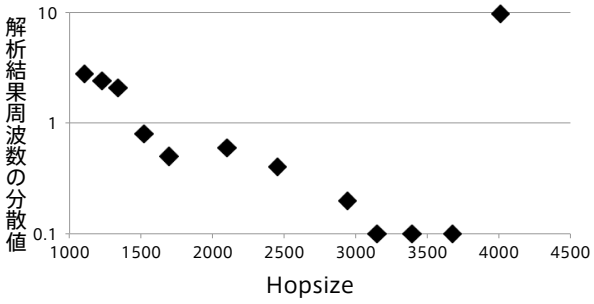

図 2　Hopsize と解析結果周波数分散値の関係

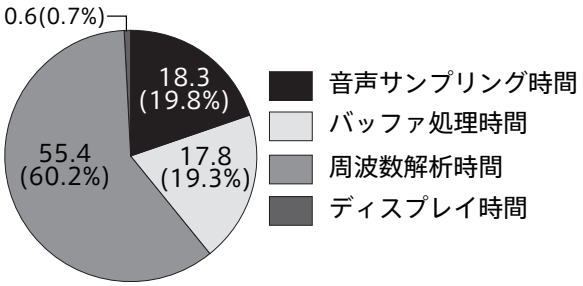

図 3　アプリケーションの各処理時間内訳
(Hopsize=3150, 単位はミリ秒)

5. 謝辞

本研究を進めるにあたって，ALF に関する質問に快く対応して下さった Drexel University Electrical & Computer Engineering METlab(博士課程) Jeffrey Scott 氏に感謝いたします．

6. 参考文献

[1] Jeffrey J. Scott et al., "An Audio Processing Library for MIR Application Development in Flash," *ISMIR*, pp. 643-648, 2010.

[2] T. Doll, J. Scott, C. Hahn, P. Diefenbach, and Y. Kim R. Migneco, "An Audio Processing Library for Game Development in Flash," *in Proc. of the IEEE Games Innovations Conference (ICE-GIC 2009)* , pp. 201-209, Aug. 2009.

[3] R. Migneco, J. J. Scott, and Y. Kim T. M. Doll, "An Audio DSP Toolkit for Rapid Application Development in Flash," *in IEEE International Workshop on Multimedia Signal Processing* , 2009.