G-30

カメラを用いたバーチャル楽器におけるフレームレートによる演奏制限の解決法

Solution against Restricting Performance by Video Frame Rate in A Virtual Musical Instrument

田中 純之介 [1]　　勝間 亮 [1]
Junnosuke Tanaka　Ryo Katsuma

概 要

現在までに，単眼カメラとＰＣのみで演奏できるバーチャル楽器が提案されている．その一例として，カメラの画面内に仮想のピアノ鍵盤を設定し，AR マーカを利用して指の位置を認識することで鍵盤を鳴らすというものがある．このバーチャル楽器プログラムでは，カメラのフレームレートによる演奏時の遅延や早い動作の認識漏れ，打鍵の誤認識が発生するという問題がある．この問題により，特に同じ音を連続して鳴らす動作において，速い指の動作を必要とする曲が演奏できない．本稿では鍵盤の連打動作に着目し，検出しやすい演奏者の特殊な動作に連打の意味を持たせることで，今まで演奏不可能であった曲を演奏できるように改良する．

1. はじめに

スマートフォンさえあれば特別な楽器を持参せずともすぐ演奏できる環境を作ることを目的とし，そのプロトタイプとして開発されたのが本稿で扱われるカメラを用いたピアノ型バーチャル楽器プログラム [1] である．このプログラムはカメラとＰＣのみで演奏できるよう実装されており，既存楽器としてカメラの画面内に仮想のピアノ鍵盤を設定し，既存のマーカ位置特定ツールである QPToolkit を用いて指の位置を認識することで鍵盤を鳴らすというアルゴリズムで動作している．

このプログラムは仮想鍵盤の範囲にマーカがあれば音が鳴るので，鍵盤から鍵盤への指の移動を打鍵と誤認識してしまう問題がある．また，カメラのフレームレートの制限による演奏時の遅延や早い動作の認識漏れが発生するという問題もある．そのため，速い指の動きを求められる曲を演奏することができない．そこで本稿ではこれらの問題を解決し，このプログラムをより実用的な楽器として使用できるようにするために次の２つの機能を追加する．

１つ目は仮想鍵盤の音を指定する役割と弾いて音を鳴らす役割を分離させ，音を指定するマーカと音を鳴らすマーカに対応させる機能である．これにより鍵盤の位置指定の正確性と打鍵の精度が上がると考えられる．２つ目の機能は連打で音を鳴らす役割を特定のマーカに与えることでフレームレートを超える速度の演奏に対応するものである．

本稿では音を指定するマーカを鍵盤，音を鳴らすマーカを鍵盤スイッチ，連打で音を鳴らすマーカを連打スイッチと呼ぶことにする．

これらの機能を検証するために従来のプログラムと鍵盤スイッチを用いたプログラム，連打スイッチを用いたプログラムそれぞれに対して次の実験を行った．

1. 演奏可能な BPM の測定
2. 演奏パターンを用いた比較

結果として鍵盤スイッチによって打鍵の誤認識は解消されたが，連打スイッチは演奏に使える段階には達しておらず，プログラムの構造的な問題が残った．

2. 問題設定

カメラによって指の位置を常に取得可能であるとし，その座標を (x, y) で表す．の座標系において，カメラに映る範囲の xy 平面上の線分 $x = a_1$ を $y = b_1$，$y = b_2$，$y = b_3$，$\cdots$，$y = b_n$ で区切り，それぞれの領域に対してピアノ鍵盤の音階の順になるように音を割り当てる．ピアノ鍵盤の音階とは，C，C#，D，D#，E，F，F#，G，G#，A，A#，B の順列であり，B の次は C に，C の前は B となる．また xy 平面上の線分 $x = c_1$，$x = c_2$，$y = d_1$，$y = d_2$ で区切られる領域を鍵盤スイッチの領域とし，線分 $x = e_1$，$x = e_2$，$y = f_1$，$y = f_2$ で区切られる領域を連打スイッチの領域とする (図 1)．鍵盤，鍵盤スイッチ，連打スイッチに対応するマーカはデザインでそれぞれ区別する (図 2)．

3. 提案システム

実際に楽器演奏に用いるプログラムについて述べる．実装は Windows 10 (TOSHIBA 社 dynabook CPU:Intel Core i7-4510U 2.60GHz RAM:8.00GB) 上で Visual Studio 2017 を用いて行った．使用したプログラミング言語は C++である．

[1] 大阪府立大学, Osaka Prefecture University, Sakai, Osaka 599–8531, Japan

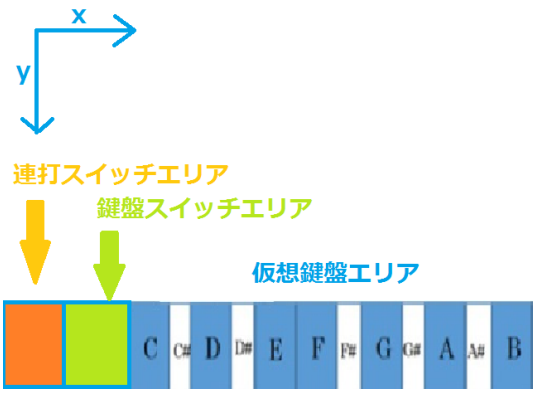

図 1: カメラ画面内における xy 平面上の各領域のイメージ

図 2: マーカデザインごとのマーカの役割

3.1 QPToolkit

QPToolkit とは Web カメラを用いて専用のマーカを認識し，画面内における二次元位置座標や三次元位置座標の値を返すプログラムである．

専用のマーカは QPToolkit の Web サイト [2] で提供されているマーカのデータを紙に印刷することで使用できる．マーカデータの中央部には何も描かれておらず，いくつかの条件はあるが，自由にマーカのデザインを決めることできる．また，マーカのデザインの違いによってマーカを区別できる．

QPToolkit は接続されたカメラから得られるビデオ画像の中からマーカを認識し，その座標情報を QPServer と呼ばれるサーバを用いて TCP/IP 経由で送信する．他のプログラムはそのデータを TCP/IP 経由で受信することであらかじめ設定したマーカ名やマーカの個数，座標情報の取得が可能となる (図 3)．ビデオのフレーム毎に座標情報を送信し続けるので 1 秒間にフレームレート個の座標情報が送られる．

3.2 ピアノ型バーチャル楽器プログラム

基本となるピアノ型バーチャル楽器プログラムについて説明する．このプログラムはカメラ内の座標のある特定の範囲を仮想の鍵盤とし，カメラ内に映る指の二次元的な座標を調べ，範囲内に指があれば座標ごとに割り振られた仮想鍵盤上の音が鳴るというものである (図 4)．

図 3: QPToolkit が表示するマーカの座標情報

図 4: カメラ画面内の仮想鍵盤のイメージ

位置座標の取得には QPToolkit を用い，音を鳴らすシステムとして OpenAL[3] を使用する．OpenAL は wav などの音声ファイルを読み込み，C++などのプログラム上で再生，停止などが出来る API である．これらを用いた楽器演奏プログラムのアルゴリズムを以下に示す．

1. QPServer が動いているサーバの IP アドレスとポート番号を指定し，QPServer に TCP ソケットを接続．用意した 2 オクターブ分 (24 個の異なるピアノ音) の音声ファイルを読み込み．
2. プログラムを続けるなら (3)，そうでなければ終了．
3. サーバからデータを受信．
4. 認識したマーカの個数が 1 個以上ならば (5)，0 個なら (2).
5. マーカが仮想鍵盤上にあれば (6)，仮想鍵盤上になければ (7).
6. ひとつ前に受信したマーカの位置が今と同じでなければ鍵盤の位置に対応した音を再生し (2)．位置が同じであれば停止し (2).
7. ひとつ前に受信したマーカが仮想鍵盤上にあれば今，再生している音を停止し (2)．仮想鍵盤上になければそのまま (2).

3.3 鍵盤スイッチの実装

鍵盤スイッチの機能を追加した楽器プログラムについて説明する．このプログラムはカメラ内の座標においてある２つの範囲を指定する．１つは鳴らす音を指定する仮想鍵盤の範囲であり，もう１つは鍵盤スイッチの範囲である．カメラ内に映る指の二次元的な座標を調べ，仮想鍵盤の範囲内に指があり，なおかつ鍵盤スイッチの範囲に指があるとき座標ごとに割り振られた仮想鍵盤上の音が鳴るというものである．演奏の際は左手に鍵盤スイッチ，右手に鍵盤マーカを取り付ける．

プログラムのフローチャートを図 5 に示す．マーカを一つずつ調べていく場合，最初に鍵盤マーカを認識したとき鍵盤スイッチの ON/OFF はまだ確認できていないので音を鳴らすことができない．そこで，複数のマーカを認識したとき，それらのマーカ名や座標といったデータをひとまず全て配列に格納するようにした．そして認識したマーカの中の鍵盤スイッチの ON/OFF を確認してからまとめて鍵盤マーカに対応した音を鳴らすという手順でプログラムを実装した．このとき，鍵盤と鍵盤スイッチに対応するマーカはマーカ名を用いて区別する．

3.4 連打スイッチの実装

連打スイッチの機能を追加した楽器プログラムについて説明する．このプログラムも鍵盤スイッチと同様にカメラ内の座標においてある２つの範囲を指定する．１つは鳴らす音を指定する仮想鍵盤の範囲であり，もう１つは連打スイッチの範囲である．カメラ内に映る指の二次元的な座標を調べ，仮想鍵盤の範囲内に指があり，なおかつ連打スイッチの範囲に指があるとき座標ごとに割り振られた仮想鍵盤上の音が一定のテンポで連続して鳴るというものである．テンポ設定はプログラム起動時に設定できるものとする．演奏の際は左手に連打スイッチ，右手に鍵盤マーカを取り付ける．

プログラムのフローチャートを図 6 に示す．このプログラムでは連続して送られてくるマーカのデータを毎回読み取るためにループ処理を用いていることに着目し，以下のような処理を行う．まずループ処理を行うごとにループカウンタの値を 1 ずつ増やしていく．ループカウンタはループ回数を数えるための変数である．また音を鳴らすきっかけとなるトリガーという変数を用意し，ループカウンタの値とトリガーの値が同じとき音を鳴らす．トリガーは読み込んだ鍵盤ごとに初めて連打音を鳴らすときのループカウンタの値が代入されている．

ループカウンタはある値を超えると 0 にリセットされる．その値を操作することで音を鳴らすテンポを設定する．今回，テンポを表す尺度として BPM を用いた．BPM とは Beat Per Minutes の略であり，1 分間に刻まれるビートの数のことである．4 分の 4 拍子の四分音符を基準として式 (1) を元にテンポ設定を行った

$$L = \frac{60000\text{ms}}{T \times BPM} \tag{1}$$

式 (1) の L は一回音を鳴らしてから次に音が鳴るまでのループ回数であり，ループカウンタはこの値を超えるとリセットされる．T はプログラムの 1 ループにかかる平均処理時間であり，この値は事前に計測して求めた結果，32ms となった．

事前に行った計測では 1 分間プログラムを実行し，1 ループにかかる処理時間を毎回出力する操作を行った．出力結果を見るとループにかかる処理時間には各操作ごとにばらつきがあった．そこであまり処理時間のかからない操作には一定時間，処理を停止する動作を加えることで処理時間の均一化を計った．これを元にもう一度計測を行って得られた結果が上記の平均処理時間 T である．

図 7: 使用したマーカとその取り付け方

4. 実験

実験の内容とその結果，考察について述べる．専用のマーカを印刷したものを図 7 のように指に貼り付け使用した．Web カメラには解像度 960×720，フレームレート 30fps の SanwaSUPPLY 製 CMS-V37BK を用い，机のような平らな面の場所に設置し，設置面からの高さを 3cm に固定して使用した (図 8)．またカメラと仮想鍵盤の距離は 25cm に設定した (図 9)．

4.1 演奏可能な BPM の測定

この実験では従来のプログラムと鍵盤スイッチを用いたプログラム，連打スイッチを用いたプログラムそれぞれの演奏可能な BPM を求める．メトロノームを基に設定した BPM に合わせて仮想鍵盤中央に位置する C の音を 32 回鳴らす．音を鳴らすタイミングが合わせられなくなったり正確に鍵盤を鳴らせなくなったら演奏不可能であるとする．BPM を 5 刻みで上げていき，演奏不可能になったら，そこから BPM を 1 ずつ下げて演奏可能な BPM を探す．

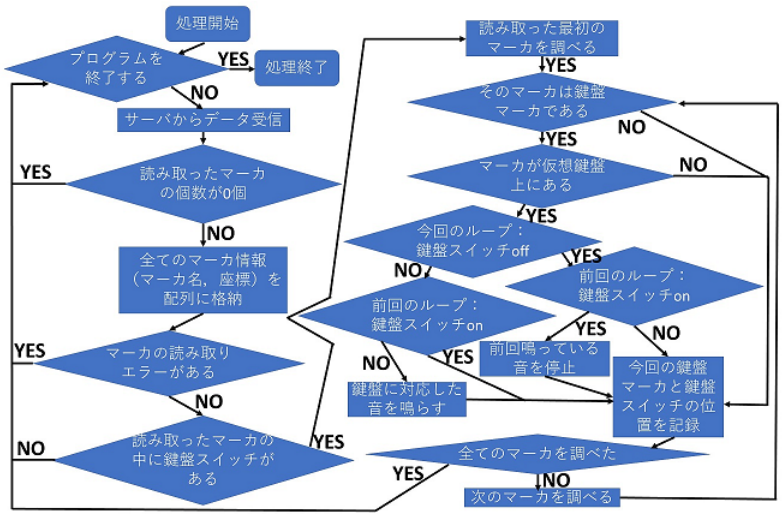

図 5: 鍵盤スイッチを実装したプログラムのフローチャート図

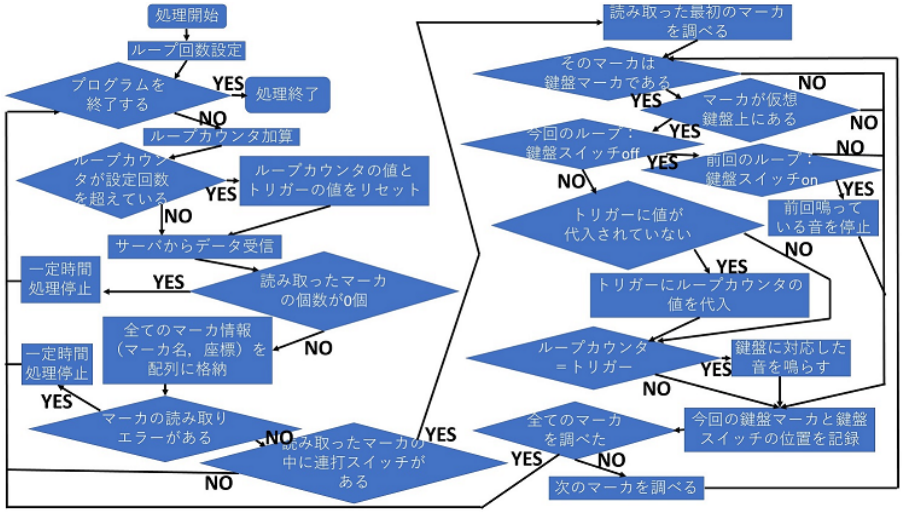

図 6: 連打スイッチを実装したプログラムのフローチャート図

図 8: 使用した Web カメラ

4.1.1　従来のプログラムの BPM 測定

まず $BPM = 80$ からスタートして BPM を上げていった．結果として演奏可能な $BPM = 105$ であった．

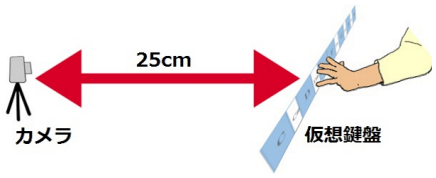

図 9: カメラと仮想鍵盤間の距離のイメージ

4.1.2　鍵盤スイッチを用いたプログラムの BPM 測定

まず $BPM = 80$ からスタートして BPM を上げていった．結果として演奏可能な BPM は従来のプログラムと同様に 105 であった．

従来のものと鍵盤スイッチを用いたもので演奏可能な BPM は同じ値となった．しかし，従来の鍵盤はマーカを上下移動させる際に他の鍵盤の領域に入り，演奏ミス

となることがが多かった．一方で鍵盤スイッチは鍵盤が固定されており，動かすのは鍵盤スイッチのみなので容易に安定して音を鳴らすことができた．

4.1.3 連打スイッチを用いたプログラムの BPM 測定

この実験は上記２つの実験とは異なり，あらかじめ設定した BPM と実際に演奏したときの BPM が一致するかどうかを検証する．あらかじめ設定したものとおなじ BPM をメトロノームにも設定する．メトロノームとタイミングを合わせて連打音を鳴らすことで，BPM が一致しているか検証する．BPM が異なる場合はメトロノームの BPM を適宜変更し，一致する BPM を探す．

各 BPM を式 (1) に代入して得られるループ回数 L とそのときの実際の BPM をまとめたのが表 1 である．表 1 ではプログラムにおけるループ回数は整数値なので，ループ回数 L は整数値に最も近い値のみをピックアップしている．また使用しているメトロノームの BPM の上限が 300 なので同様に BPM の上限は 300 としている．表 1 中の 70-80 といった値は実際の BPM の測定が困難であったためおおよそ値の範囲を示したものである．測定が困難であった理由は連打音発生の間隔が安定しなかったためである．

また表 1 より設定した BPM と実際の BPM が異なることが見て取れる．この原因として 1 ループの平均処理時間 T の値の設定が間違っていることが考えられる．そこで表 1 の各ループ回数とそれに対応する明確な値を持つ実際の BPM を式 (2) に代入し，平均を取った．

$$T = \frac{60000\text{ms}}{L \times BPM} \tag{2}$$

これにより実際の BPM を元にした平均処理時間 $T = 37$ms 得られた．この値を式 (1) に代入し，設定 BPM を元に，ループ回数 L をもう一度算出した．その結果を表 2 にまとめた．そして，もう一度設定した BPM と実際に演奏したときの BPM が一致するかどうかを検証した．その結果，設定した BPM と実際の BPM はおおむね一致したが，連打音の間隔にはまだ偏りがあり，安定はしなかった．

4.2 演奏パターンを用いた比較

この実験では従来のプログラムと鍵盤スイッチを用いたプログラム，連打スイッチを用いたプログラムそれぞれに対してある特定の演奏パターンをメトロノームに合わせて 2 回演奏することで比較を行う．またそのときの BPM は上記の実験を元に適宜変更していく．演奏パターンは G，G，G，G，E，E，E，E，B，B，B，B，F，F，F，F とする．連打と指移動の多いパターンとしてこのパターンを用いた．

表 1: 設定 BPM から割り出したループ回数と実際の BPM($T = 32$ms)

設定 BPM	ループ回数 L	実際の BPM
81	23.1	70-80
85	22.1	70-80
89	21.1	70-80
93	20.2	80
98	19.1	85
104	18.0	90
110	17.0	95
117	16.0	100
125	15.0	105
133	14.1	110
144	13.0	120-125
156	12.0	130-140
170	11.0	140-150
187	10.0	150-160
208	9.0	160-170
234	8.0	180-190
267	7.0	220-230
300	6.3	240-250

表 2: 設定 BPM から割り出したループ回数 L($T = 37$ms)

設定 BPM	ループ回数 L
83	19.0
87	18.1
92	17.2
98	16.1
105	15.0
112	14.1
121	13.0
131	12.1
143	11.0
157	10.1
175	9.0
197	8.0
225	7.0
263	6.0
315	5.0

4.2.1 従来のプログラム

まず $BPM = 80$ からスタートして BPM を上げていった．結果として演奏可能な $BPM = 105$ であった．しかし，指を移動させるときに他の鍵盤の音を鳴らして

しまうこともあった．

4.2.2 鍵盤スイッチを用いたプログラム

まず $BPM = 80$ からスタートして BPM を上げていった．結果として演奏可能な BPM は従来のプログラムと同様に 105 であった．従来のプログラムに比べて指を移動させるときの誤った打鍵はなくなった．しかしマーカを２つ使う特殊な手法なので練習が必要である．

4.2.3 連打スイッチを用いたプログラム

連打スイッチを用いなくても $BPM = 105$ までは演奏可能であることが分かったので $BPM = 105$ からスタートして表 2 を基準に BPM を上げていった．結果として，BPM が 105-143 の間は連打スイッチが ON になってから音がなり始めるまでの遅延が若干あった．また $BPM = 157$ 以降は相対的に遅延がかなり大きなものとなり，演奏に支障をきたした．

4.3 考察

以上の実験から連打スイッチはまだ演奏に使える段階ではないことが分かる．考えられる原因としては，1 ループにかかる処理時間が長いことが挙げられる．これはプログラム構造的な見直しが必要な問題である．また時間のかからない処理に一定時間の処理停止を行う際の一定時間を厳密に定義すればループ処理時間のばらつきが減少すると考えられる．これらのプログラムの構造的な問題を解決することを今後の課題とする．

5. まとめ

単眼カメラとＰＣのみで演奏できるピアノ型バーチャル楽器における，カメラのフレームレートによる演奏時の遅延，早い動作の認識漏れ，打鍵の誤認識が発生するという問題に対して，鍵盤スイッチと連打スイッチという新たな機能を追加することで解決を試みた．結果として鍵盤スイッチによって打鍵の誤認識は解消されたが，連打スイッチは演奏に使える段階には達しなかった．演奏に耐える楽器を作るためにプログラムの構造的な問題を解決することを今後の課題とした．

参考文献

[1] Junnosuke Tanaka and Ryo Katsuma: “Proposal of Virtual Musical Instrument Using Single Camera and Verification of Playability,” *Proc. of IEEE Consumer Communications and Networking Conference (CCNC)*, 2018.

[2] Sunao Hashimoto: Kougakunabi QPToolkit Webcamera wo tsukatta kanntannichikeisoku (Engineering Navi QPToolkit Simple position measurement using Web camera)(online), `http://kougaku-navi.net/QPToolkit/`(2011.01.05).

[3] Free Software Foundation: LGPL,OpenAL Soft(online), `http://kcat.strangesoft.net/openal.html`.