計測自動制御学会東北支部 第 309 回研究集会 (2017.6.23)
資料番号 309-1

バーチャル ボール バランシング：バーチャルリアリティ教育教材の開発

Virtual Ball Balancing: Development of a Teaching Material for Virtual Reality Education

○岩谷 靖

○ Yasushi Iwatani

弘前大学 大学院理工学研究科

Hirosaki University, Department of Science and Technology

キーワード： 仮想現実 (virtual reality)，教育教材 (teaching material)，
3 軸加速度センサ (tri-axis accelerometer)，画像処理 (image processing)

連絡先： 〒 036-8561　青森県弘前市文京町 3，弘前大学 大学院理工学研究科
岩谷 靖，Tel.: (0172) 39 - 3697，Fax.: (0172) 39 - 3697，E-mail: iwatani.at.hirosaki-u.ac.jp

1. はじめに

バーチャルリアリティ（Virtual Reality；以下 VR と略記）の定義は人や文脈によって様々であり，統一的な見解は見られない．広義には「計算機などを使って合成された，実際には存在しない人工的な世界」と定義される [1]．広義の定義は広すぎて，単なるコンピュータシミュレーションも含まれてしまう．狭義の定義の一例としては，「人間が実際の環境を利用しているのと本質的に同等な状態でコンピュータが生成した人工環境」が挙げられる [2]．また狭義の定義では，「三次元の空間性 [2]」や「没入感 [3]」のほか，デバイスとしての「ヘッドマウントディスプレイ [4]」が必須要素とされることも多い．狭義の定義は VR の最終理念や最新形態を表しており，後述する教育目的には敷居が高すぎる．本稿では広義の意味，かつ感覚フィードバック（Sensory Feedback）[3] を有するシステムを VR と呼ぶことにする．

感覚フィードバックは計測・シミュレーション・提示の三つから構成される．より具体的には，(1) 現実世界の何らかの物理量をセンサで計測し，(2) その計測量に基づき，コンピュータ内に構築した人工環境を更新し，(3) 更新された人工環境を何らかのデバイスに提示して人の知覚に刺激を与える，ことを繰り返す．

本稿では，これら三つの技術的な構成要素を学ぶことのできる教育教材：バーチャル ボール バランシング実験装置を紹介する．この装置は，機械系学科の学部三年生を対象とした 180 分の実験課題用に開発している．その教育目的は，学生が一から VR 装置を組み立てることで，計測・シミュレーション・提示を理解することと，それらを組み合わせた 感覚フィードバック・VR の実装を習得することにある．出来合いの実験装置では，(a) 中身がブラックボックス化

されていて数少ない変数を操作することしかできず，基礎要素の習得に不向きである，(b) 予算の都合上，学生数に対して十分な数の実験装置を準備することが困難である，などの問題がある．提案装置は単純な構造をしており，安価に自作可能である．そのため，学生一人につき一台の実験装置を準備することへの金銭的障壁も低い．提案装置は単純である一方，様々な物理現象を VR 化することの学習も可能である．

2. 実験装置

開発した実験装置の模式図を Fig. 1 に示す．この実験装置は，人間が現実世界のシーソーの角度を調節し，仮想世界（ディスプレイ内）のシーソー上を動く球を目視して手動制御するものである．これは，Ball & Beam 実験装置[5] の VR・手動制御版と言える．

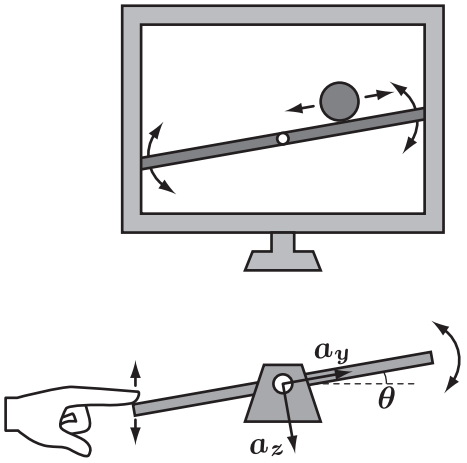

Fig. 1 Virtual ball balancing system.

実験装置の仕組みは次の通りである．現実世界のシーソーには加速度センサが取り付けられており，シーソーの長手方向の加速度 a_y と，シーソーと垂直下向き方向の加速度 a_z をマイクロコンピュータが読み込む．マイクロコンピュータでは，加速度からシーソーの傾き角

$$\theta = \mathrm{atan2}(a_{y0} - a_y, a_{z0} - a_z) \tag{1}$$

を計算し，PC に送信する．ここで a_{y0} と a_{z0} はそれぞれ，センサ軸が水平である時の a_y，a_z の値である．PC では，仮想世界のシーソーの角度を，受信した角度と同一になるように表示する．また，仮想世界のシーソー上の球を，物理モデル

$$m\ddot{x} = mg\sin\theta \tag{2}$$

に従い動かす．ここで，x は画像平面における球の水平方向の位置，g は重力加速度である．m は球の質量を意味するが，消去し得るので実際には使用されない．

実験装置に用いた材料と，その選定理由を以下にまとめる．

加速度センサは，3 軸方向の加速度が計測できる KXR94-2050 (KIONIX, Inc.) である．シーソーの角度は，加速度センサのほかにポテンショメータやエンコーダでも計測可能である．ポテンショメータやエンコーダは他の実験装置[6] で使用しており，学生が多様なセンサに触れる機会を設けるために加速度センサを用いることとした．

マイクロコンピュータは Arduino Uno である．この選定理由は，他の実験・実習課題でも Arduino を利用しており[6, 7]，それらと統一することで連携的な学習効果を期待するためである．同様に PC 側の画像処理は，他のプログラミングの講義でも使用している Processing で行う．Arduino と Processing のプログラムは，それぞれ付録 A，B にまとめた．

PC を除いた実験装置の作成費用は，Arduino Uno が約三千円，加速度センサが約千円，その他部材が約千円の計五千円程度である．Arduino Uno の代わりに安価な互換品を利用することで，予算削減も可能である．

3. 課題内容

以下では，180 分（90 分× 2）の講義時間における実験課題例を述べる．前半 90 分は，主に現実世界での計測問題を取り扱う．前半の課題は，ガイダンス・加速度センサ値の取得・シーソーの傾き角の取得・Arduino と Processing の連携，の項目からなる．後半 90 分は，シミュ

レーション・提示の問題と，VR としての統合化設計に取り組む．後半の課題は，シーソーの描画・球の動作生成・調整と考察・まとめ，の項目からなる．各項目の詳細と，想定する時間設定を以下にまとめる．

3.1 前半の課題

3.1.1 ガイダンス（10 分）

この課題で取り組む内容と，その工学的位置付けや社会的意味を説明する．

3.1.2 加速度センサ値の取得（30 分）

Arduino の `analogRead` 関数を用いて，センサが取得した値を Arduino に取り込む．Arduino が取得したセンサ値は，Arduino のシリアルモニタに表示する．

シーソーの傾きに応じて，y 軸と z 軸のセンサ値が変化すること，および，x 軸のセンサ値が変化しないことを確認する．つぎに加速度センサをシーソーから取り外すなどして，加速度センサの軸が鉛直上向きの場合にセンサの値が最大に，鉛直下向きの場合に最小になり，それらの間では傾き角に応じて滑らかにセンサ値が推移することを確認する．また加速度軸が水平の場合のセンサ値も取得し，記録する．

時間が許せば，平均化による雑音処理の仕組みと効果を確認する．

3.1.3 シーソーの傾き角の取得（30 分）

加速度とシーソー角度の関係式 (1) を説明し，シーソー角度を Arduino のシリアルモニタに表示するプログラムを作成する．分度器などを用いて測定した角度と，加速度センサを通じて Arduino が計算した角度を比較し，正しく角度データが取得できていることを確認する．

3.1.4 Arduino と Processing の連携（20 分）

Arduino・Processing 間の通信プログラムを作成する．具体的には，Arduino が取得した角度データを，Processing の `println` 関数を用いて Processing のコンソール画面に表示する．

3.2 後半の課題

3.2.1 シーソーの描画（20 分）

Processing が受信した角度の値に応じて傾き角が変化するシーソーを，Processing のキャンバス上に描画するプログラムを作成する．

3.2.2 球の動作生成（40 分）

球の運動方程式 (2) の説明，および微分方程式の近似解を得るためのオイラー法の説明を行い，球の動作生成と描画を行う．

3.2.3 調整と考察（20 分）

プログラム内のパラメータの調整や，プログラム全体の整理を行う．これらを通じて，VR における技術的な構成要素である計測・シミュレーション・提示の基礎要素と，それらを組み合わせて感覚フィードバック・VR を構成する手続きについての理解を深める．

3.2.4 まとめ（10 分）

実験課題を総括し，講義を終了する．

4. 発展課題の例

3 章で述べた課題内容は，シーソー上の球の動作について最も単純なモデルを採用している．これらで物足りない場合は，以下の課題の導入も可能である．

4.1 雑音処理

加速度センサは雑音の影響を受けやすい．付録の Arduino のプログラムでは，100 データの平均値を当該時刻の加速度としている．平均化しない場合との比較や，IIR 型のローパスフィルタの導入などにより，雑音処理の手法や効果を学習できる．なお 100 データの平均化には，20 ミリ秒強の計算時間が掛かっている．

4.2 微分方程式の解法

付録の Processing のプログラムでは，微分方程式 (2) の近似解をオイラー法で求めている．ルンゲ・クッタ法などを導入することで，精度の良い微分方程式の数値解法の仕組みや効果を学ぶことができる．

4.3 物理現象の精緻なモデル化

微分方程式 (2) では，球が質点としてモデル化されており球の回転に関する慣性項が導入されていない．また摩擦の項も存在しない．これらをモデル化することで，より現実に即したシミュレータとなる．

また，現実世界ではシーソーの端が存在する．球がシーソー外に飛び出た後は，球は水平方向には等速直線運動，鉛直方向には自由落下する（摩擦の無い場合）．これのモデル化は，球がシーソー上にあるか否かを判別し，微分方程式を切り替えることで達成される．さらに厳密には，シーソー端点での球の回転や，シーソー側面での球の転がりのモデル化も必要となる．

さらに現実世界では，シーソーが素早く動く場合，球がシーソーから離れて浮き上がる可能性もある．これは相補性システム [8] としてモデル化され得る．その後，球がシーソー上に落ちた場合の運動を再現するには，球とシーソーの衝突・跳ね返り現象をモデル化しなければならない．衝突を繰り返した後に跳ね返りが終了し，再び球がシーソーに接してシーソー上を動くことを考えよう．この場合，理論上は有限時間内に無限回の衝突現象が発生しているが，無限回の衝突現象を正確に模擬するには無限の計算時間が必要となる．実際には理論を適切に近似した計算が必要となる [9]．不適切な近似計算では，反発係数が 1 未満であったとしても跳ね返り高さがゼロに収束せず，定常振動のような定常的な跳ね返りが残ることもある．

提案する実験装置は単純なものであるが，突き詰めれば様々な物理現象のシミュレーションを考えることができる．

4.4 情報通信量

付録の Arduino-Processing 間のシリアル通信では，シーソーの角度情報について 2 バイトの情報を送受信している（合図信号も含めて 3 バイトのデータの送受信）．シリアル通信は 1 バイトの通信が基本であり，角度を 1 バイトで表現すると分解能は約 1 度である．角度分解能 1 度ではシーソーの動きが滑らかで無いことが目視でも確認できる．これらの相違を通じて，データの分解能や情報通信量の概念を学習できる．

5. おわりに

本稿では，安価に自作可能なバーチャルリアリティ教育教材を紹介した．実験課題の教育目的は，学生が一から VR 装置を組み立てることで，計測・シミュレーション・提示の基礎要素を理解することと，それらを組み合わせた感覚フィードバック・VR の実装を習得することにある．装置自体は単純な構造であるが，発展的な課題として様々な物理現象の VR 化の学習も可能である．

参考文献

1) 廣瀬：バーチャル・リアリティ，コンピュータソフトウェア，**11**-5，65/75 (1994)

2) 舘・佐藤・廣瀬（監修），日本バーチャルリアリティ学会（編集）：バーチャルリアリティ学，コロナ社 (2010)

3) W. R. Sherman and A. B. Craig: Understanding virtual reality: interface, application, and design, Morgan Kaufmann (2002)

4) P. Brey: Virtual reality and computer simulation, In K. Himma and H. Tavani editors, Handbook of Information and Computer Ethics, John Wiley & Sons (2008)

5) 平田：Arduino による Ball & Beam 実験装置，計測と制御，**54**-3，188/191 (2015)

6) 岩谷・葛西：失敗から学ぶ，計測と制御，**54**-3，178/183 (2015)

7) 岩谷・佐川・城田・本井：学生が授業のために購入可能なロボット教材の開発，第 57 回自動制御連合講演会，594/597 (2014)

8) A. J. van der Schaft and J. M. Schumacher: Complementarity modeling of hybrid systems, IEEE Transactions on Automatic Control, **43**-4, 483/490 (1998)

9) K. H. Johansson, J. Lygeros, S. Sastry, and M. Egerstedt: Simulation of Zeno hybrid automata, Proc. of the 38th IEEE Conference on Decision and Control, 3538/3543 (1999)

A　付録：Arduino プログラム

```
 1: void setup()
 2: {
 3:   Serial.begin( 9600 ); // 通信設定
 4: }
 5:
 6: void loop()
 7: {
 8:   long  y = 0, z = 0;            // 加速度を格納する変数．平均化の加算用に long 型で宣言
 9:   const long y0 = 501, z0 = 504; // 水平時の加速度の値（定数）
10:
11:   // 加速度の取得と平均化
12:   for ( int i = 0 ; i < 100 ; i++ )
13:   {
14:     y += analogRead( 1 );
15:     z += analogRead( 2 );
16:   }
17:
18:   y /= 100;
19:   z /= 100;
20:
21:   // 角度計算．-π～π rad. を 0～2 π rad. に変換し，小数第三位までを 2 バイト整数値として送信
22:   int s = (int) ( ( atan2( y0 - y, z0 - z ) + 3.1416 ) * 1000.0 );
23:
24:   // 通信
25:   Serial.write( 0 );               // 通信開始の合図
26:   Serial.write( highByte( s ) );  // 上位バイト
27:   Serial.write( lowByte(  s ) );  // 下位バイト
28: }
```

B　付録：Processing プログラム

```
 1: // 通信設定
 2: import processing.serial.*;
 3: Serial myPort;
 4:
 5: // グローバル変数と，一回だけ初期化が必要な変数の宣言
 6: float s = 0.0; // シーソーの角度
 7: float x = 0.0; // 球の水平方向の位置
 8: float v = 0.0; // 球の水平方向の速度
 9:
10: void setup()
11: {
12:   myPort = new Serial( this, Serial.list()[0], 9600 ); // 通信設定
13:   size( 800, 400 ); // キャンバスサイズの設定
14:   x = width / 2;    // 球の水平方向初期位置をキャンバスの中央に設定
15: }
16:
17: void draw()
18: {
19:   // 定数の設定
20:   final float scale = 4.0; // ピクセル・メートル比：1m あたりのピクセル数
21:   final float fps = 60.0;  // 画像の更新頻度： frames per second
22:   final float g = 9.80665; // 重力加速度
23:   final float h = 10.0;    // シーソーの厚さ
24:   final int   r = 20;      // 球の直径
25:
```

```
26:   // シーソーの四端点の座標．シーソーはキャンバス横幅よりも長く描画する．
27:   float x1 = width  / 2 + width * cos( s );
28:   float y1 = height / 2 + width * sin( s );
29:   float x2 = width  / 2 - width * cos( s );
30:   float y2 = height / 2 - width * sin( s );
31:   float x3 = width  / 2 - width * cos( s ) - h * sin( s );
32:   float y3 = height / 2 - width * sin( s ) + h * cos( s );
33:   float x4 = width  / 2 + width * cos( s ) - h * sin( s );
34:   float y4 = height / 2 + width * sin( s ) + h * cos( s );
35:
36:   // 球の水平方向の速度・位置のオイラー法による更新．
37:   v += g * sin( s ) * scale / fps;
38:   x += v * scale / fps;
39:
40:   // 球の鉛直方向の位置．球はシーソー上面と常に接することを仮定．
41:   float y = height / 2 + ( x - width / 2 ) * tan( s ) - r / cos( s );
42:
43:   // キャンバスの更新
44:   background( 255 );
45:
46:  // シーソーと球の描画
47:   fill( 200 );
48:   quad( x1, y1, x2, y2, x3, y3, x4, y4 );
49:   ellipse( x, y, 2 * r, 2 * r );
50:
51:   // シーソーの回転軸の描画
52:   fill( 255 );
53:   ellipse( width / 2, ( height + h ) / 2, h / 1.5, h / 1.5 );
54: }
55:
56: // シリアル通信による角度データの取得
57: void serialEvent( Serial port )
58: {
59:   if ( port.available() > 2 ) // 通信開始合図も含めて 3 バイトのデータが 1 セット
60:   {
61:     if ( port.read() == 0 )   //通信開始合図の確認
62:     {
63:       // 角度データの受信と復元
64:       s = ( port.read() * 256 + port.read() ) / 1000.0 - 3.1416;
65:     }
66:   }
67: }
```