

# 目次

第 1 章	序論	2
1.1	スポーツ教育の現状	2
1.2	Data Volley	2
1.3	データを用いたバレーボール指導	3
1.4	本論文の構成	3
第 2 章	本研究が目指すシステム	4
2.1	先行研究との比較	4
第 3 章	映像内のコートの検出	5
3.1	アンテナを用いた検出	5
3.2	GUI を用いた手動指定	5
第 4 章	映像内の選手の検出	6
4.1	OpenPose を用いた検出	6
4.2	AlphaPose を用いた検出	6
第 5 章	射影変換を用いた選手の位置推定	7
5.1	射影変換	7
5.2	推定結果	7
第 6 章	推定した選手の位置の精度評価	8
6.1	精度評価に用いる基準点	8
6.2	精度の評価結果	8
第 7 章	結論	9
7.1	まとめ	9
7.2	今後の課題	9
7.3	本研究の展望	9
付録 A	VideoPose3D を用いた三次元姿勢推定	10
付録 B	作成したソースコード	11
参考文献		17

# 第 1 章

## 序論

### 1.1 スポーツ教育の現状

現在、スポーツ教育の現場では、従来から行われてきた、熟練した指導者の「目」に基づく「感覚的な指導」に比べ、カメラのような機械の「目」を用いてデータ化した選手の動きなどに基づく「定量的な指導」に注目が集まっている。[1] 選手の動きをデータ化するにあたって、さまざまなアプローチが行われてきた。[2] この「データ」には、球技におけるボールの位置、選手の位置、選手の姿勢などが含まれるが、本研究ではバレーボールの試合中における、特に選手の位置に注目を絞る。

### 1.2 Data Volley

バレーボールの解析に用いるツールとして、Data Volley[3] を紹介する。これはイタリアの Genius Sports Italy 社が開発するソフトウェアで、現在バレーボール業界で広く用いられている。

図 1.1 に Data Volley の GUI を示す。これはアナリストと呼ばれる専門家やコーチが用いるソフトウェアで、各選手のポジションをあらかじめ登録し、コマンドで選手がボールに触れたときの行動を試合を観ながらリアルタイムで入力する。それに基づいて、プレイの決定率や、選手たちの位置をあとから見直せるものである。

図 1.1 Data Volley の GUI

しかし、Data Volley には次に示すような欠点が挙げられる。

- 複雑な入力コマンド
- 瞬間的なプレイの判断の要求
- 記録される位置の精度の低さ
- 人的入力ミス

一つ目は「複雑な入力コマンド」である。以下に、入力コマンド例を二つ示す。

```
15S.14#K1a26P4.17+14Q15#.K3a25E.  
3S16.5#a19PZ6.12/3D6E12P17C#.7a5D.
```

これらの意味は割愛するが、それぞれ1ラリー中での両チームのボールのやりとりを示している。このような複雑なコマンドを覚えるだけでなく、選手たちのプレイを見ながら入力を行う必要があるため、ある程度の練習が必要である。

二つ目は「瞬間的なプレイの判断の要求」である。Data Volley はバレーボールの試合中のプレイを解析するため、入力を行うユーザにはもちろんバレーボールの知識が要求される。さらに、入力にはプレイの種類だけでなく、ユーザからみたその評価も含むため、バレーボールのルールを知っているだけでなく、プレイの評価もできることがユーザに求められる。これは、先述の「複雑な入力コマンド」に重ね、Data Volley の仕様の敷居を高めており、1.1 節にて述べたように、このソフトウェアのユーザをアナリストやコーチに限定しまっている要因である。

三つ目は「記録される位置の精度の低さ」である。Data Volley では、図 1.2 に示すように、コート of 片面を  $6 \times 6$  に 36 等分して扱う。バレーボールコートの片面は 9m 四方であるため、Data Volley で記録される位置の最小単位は 1.5m 四方ということになる。

図 1.2 Data Volley におけるコートの分割

また、ボールに触れていない選手の位置は入力しないため、ボールに触れた瞬間でしか選手の位置を見直せないことも同様に欠点である。

最後に「人的入力ミス」を挙げた。これはいうまでもなく、ユーザがコマンドを入力をする際に起こすミスのことである。例えば背番号の 4 と 5 や、レシーブの一種を表す D と返球の一種を表す F は最も一般的な QWERTY 配列のキーボード上では隣り合っているが、これらが間違って入力された場合は大きく意味が異なり致命的である。

## 1.3 データを用いたバレーボール指導

本節では、本研究で注目する、「選手の位置」に基づき行われる指導の例を示す。

- 相手の攻撃に対する防衛陣形の指導
- 相手の陣形に対する攻撃の指導
- コート内での選手のエリア管理

これらはどれもチーム全員の位置情報に基づくため、Data Volley では実現ができない指導である。現状、多くのチームは動画を後から見直し、ホワイトボードなどを用いてこれを行っている。

## 1.4 本論文の構成

最後に、本論文の構成を示して序論を締めくくる。第 2 では、本章で述べた背景に基づいた、本研究の目的及び、その先行研究との比較を述べる。第 3, 4, 5 章では、この目的を実現するためのアルゴリズムを述べる。第 5 章ではさらに、これらのアルゴリズムを用いて得られた結果を述べている。第 6 章では、得られた結果の精度評価を行っている。最後に、第 7 章では本研究のまとめと課題、及び展望を述べている。

また、付録 A, B ではそれぞれ、7.3 節で述べる選手の三次元姿勢の推定と、本研究で作成したプログラムのソースコードを紹介している。

## 第 2 章

# 本研究が目指すシステム

本研究の目的を、1.1 節でも述べたように、選手の位置に注目した解析システムの構築とする。解析対象には、現在 Data Volley のユーザが最も用いている情報端末を用いて撮影したバレーボールの試合映像を用いる。さらに、Data Volley のようなシステムの構築の基礎とするため、1.2 節に示した Data Volley の欠点を踏まえて、「(できる限り) 人の手を介さない」、さらに、「扱いやすい」システムを目指す。

ここでいう「扱いやすさ」とは、活用ができるユーザ数が限られてしまっている Data Volley に比べて、より一般的に多くの人が使えらることを指す。

### 2.1 先行研究との比較

## 第 3 章

# 映像内のコートの検出

3.1 アンテナを用いた検出

3.2 GUI を用いた手動指定

## 第 4 章

# 映像内の選手の検出

4.1 OpenPose を用いた検出

4.2 AlphaPose を用いた検出

## 第 5 章

# 射影変換を用いた選手の位置推定

### 5.1 射影変換

### 5.2 推定結果

## 第 6 章

# 推定した選手の位置の精度評価

### 6.1 精度評価に用いる基準点

### 6.2 精度の評価結果



## 第 7 章

# 結論

### 7.1 まとめ

本研究では、情報端末の内蔵カメラを用いて撮影したバレーボールの試合映像と、その映像内でコートが写っている位置の情報から、自動で選手の位置を推定・追跡した。さらに、推定した選手の位置の精度の評価を行った。その結果、カメラと選手の距離が増大するほど、推定位置の誤差も増大することが予想できた。

### 7.2 今後の課題

本研究では、選手が床上にいる前提で解析を行った。しかし、バレーボール選手は試合中にジャンプを行うため、空中にいる選手の位置推定も考慮する必要がある。また、ボールの位置も同様に空中であるため、本研究では推定を行っていないが、Data Volley のようなプレイの解析を行うためには選手の位置だけでなくボールの位置の情報も必要となるため、これも同様に検討の必要がある。

### 7.3 本研究の展望

最後に、本研究の展望を述べる。

付録 A に示す VideoPose3D[4] は、AlphaPose のようなアルゴリズムを用いて検出した選手の二次元姿勢情報から、その三次元姿勢を推定するアルゴリズムである。これを用いることで、映像内の選手の実際の姿勢が推定できる。これと本研究で得た選手の位置情報を組み合わせることで、バレーボールの試合の三次元再現が可能であると予測する。

## 付録 A

# VideoPose3D を用いた三次元姿勢推定

## 付録 B

# 作成したソースコード

以下に、本研究で作成したソースコードを示す。

リスト B.1, B.2, B.3 に示す、index.html, index.css, index.js の三つを合わせて、ブラウザ上で動くように作成した。これには、3 で述べた、コート of 四隅を手動で指定する GUI も含まれる。

また、本研究では AlphaPose を用いて映像内の選手の位置を推定した結果を一度ファイルに保存し、B.1 のプログラムで読み込む形をとった。AlphaPose を用いて解析を行う過程は、AlphaPose のソース [5] を参考にした。

リスト B.1 index.html

```
1 <html>
2   <head></head>
3   <body>
4     動画ファイル: <input id="video_input" type="file" accept="video/*"><br>
5     解析データ: <input id="pose_input" type="file" accept=".json"><br>
6     <div style="position: relative">
7       <video id="video" controls></video>
8       <canvas id="video_canvas"></canvas>
9     </div>
10    <canvas id="board_canvas"></canvas>
11  </body>
12  <link rel="stylesheet" href="index.css">
13  <script src="https://docs.opencv.org/3.4.1/opencv.js"></script>
14  <script src="index.js"></script>
15 </html>
```

リスト B.2 index.css

```
1 #video_canvas {
2   position: absolute;
3   top: 0;
4   left: 0;
5 }
6
7 #board_canvas {
```

```

8     width: 500px;
9     height: 800px;
10    background: #f3bf88;
11    margin-top: 32px;
12 }

```

---

リスト B.3 index.js

---

```

1  const COURT_CANVAS_WIDTH  = 500
2  const COURT_CANVAS_HEIGHT = 800
3  const COURT_CORNERS_ON_BOARD = [
4      [100, 100],
5      [100, 700],
6      [400, 700],
7      [400, 100]
8  ]
9  const COURT_GATES_ON_BOARD = [
10     [100, 300],
11     [400, 300],
12     [400, 500],
13     [100, 500]
14 ]
15
16 const video      = document.getElementById("video")
17 const videoCanvas = document.getElementById("video_canvas")
18 const videoContext = videoCanvas.getContext("2d")
19 const boardCanvas = document.getElementById("board_canvas")
20 const boardContext = boardCanvas.getContext("2d")
21
22 const courtCorners = []
23 const playerPos    = []
24
25 var M = null
26
27 function getModifiedPos(event) {
28     const clientRect = videoCanvas.getBoundingClientRect()
29
30     x = event.pageX - clientRect.left - window.pageXOffset
31     y = event.pageY - clientRect.top  - window.pageYOffset
32
33     courtCorners.forEach(function (pos) {
34         const [preX, preY] = [...pos]
35
36         if (
37             Math.abs(preX - x) / video.videoWidth < 0.05 &&
38             Math.abs(preY - y) / video.videoHeight < 0.05
39         ) [x, y] = [preX, preY]

```

```

40     })
41
42     return [x, y]
43 }
44
45 function draw(pos) {
46     videoContext.lineWidth    = 4
47     videoContext.strokeStyle = "blue"
48     videoContext.fillStyle   = "blue"
49
50     videoContext.clearRect(0, 0, videoCanvas.width, videoCanvas.height)
51     videoContext.beginPath()
52     videoContext.moveTo(...courtCorners[0])
53     for (let i = 1; i < courtCorners.length; i++)
54         videoContext.lineTo(...courtCorners[i])
55     videoContext.stroke()
56
57     courtCorners.forEach(function (corner) {
58         videoContext.beginPath()
59         videoContext.arc(...corner, 5, 0, 2 * Math.PI, false)
60         videoContext.fill()
61     })
62 }
63
64 document.getElementById("video_input").addEventListener("change", function () {
65     video.src = window.URL.createObjectURL(this.files[0])
66     console.log("Loading video...")
67 })
68
69 video.addEventListener("progress", function (event) {
70     console.log("Loaded video.")
71
72     videoCanvas.width  = video.videoWidth
73     videoCanvas.height = video.videoHeight - 64
74
75     console.log("Canvas size changed: " + [videoCanvas.width, videoCanvas.height])
76 })
77
78 video.addEventListener('loadedmetadata', function (e) {
79     let time = video.currentTime;
80     requestAnimationFrame(function me() {
81         if (time !== video.currentTime) {
82             time = video.currentTime;
83             video.dispatchEvent(new CustomEvent("timeupdate"));
84         }
85         requestAnimationFrame(me);

```

```

86     });
87 });
88
89 function warp(point) {
90     product = [
91         M.data64F[0] * point[0] + M.data64F[1] * point[1] + M.data64F[2],
92         M.data64F[3] * point[0] + M.data64F[4] * point[1] + M.data64F[5],
93         M.data64F[6] * point[0] + M.data64F[7] * point[1] + M.data64F[8]
94     ]
95
96     return [product[0] / product[2], product[1] / product[2]]
97 }
98
99 video.addEventListener("timeupdate", function () {
100     const frameRate = 30
101     const frame = Math.round(frameRate * video.currentTime)
102
103     draw()
104     drawLine()
105     videoContext.fillStyle = "green"
106
107     for (idx in playerPos[frame]) {
108         videoContext.beginPath()
109         videoContext.arc(...playerPos[frame][idx], 5, 0, 2 * Math.PI, false)
110         videoContext.fill()
111
112         const warpedPlayerPos = warp(playerPos[frame][idx])
113         if (warpedPlayerPos[1] > COURT_CANVAS_HEIGHT / 2) {
114             boardContext.fillStyle = "blue"
115         } else {
116             boardContext.fillStyle = "red"
117         }
118
119         boardContext.beginPath()
120         boardContext.arc(...warpedPlayerPos, 5, 0, 2 * Math.PI, false)
121         boardContext.fill()
122     }
123 })
124
125 videoCanvas.addEventListener("click", function (event) {
126     pos = getModifiedPos(event)
127
128     if (JSON.stringify(courtCorners.slice(-1)[0]) == JSON.stringify(pos)) {
129         console.log("Popped_court_corner:_" + courtCorners.pop())
130         draw()
131     } else {

```

```

132     courtCorners.push(pos)
133     draw()
134     console.log("Added_court_corner:" + pos)
135
136     if (courtCorners.length == 5) {
137         M = cv.getPerspectiveTransform(
138             cv.matFromArray(4, 1, cv.CV_32FC2, courtCorners.slice(0, 4).flat()),
139             cv.matFromArray(4, 1, cv.CV_32FC2, COURT_CORNERS_ON_BOARD.flat())
140         )
141
142         console.log("Calculated_perspective_transform:" + M.data64F)
143     }
144 }
145 })
146
147 document.getElementById("pose_input").addEventListener("change", function () {
148     const reader = new FileReader()
149     reader.onload = function () {
150         console.log("Loading_pose_json...")
151
152         let lastImg = ""
153         JSON.parse(reader.result).forEach(function (personData) {
154             if (lastImg != personData.image_id) {
155                 playerPos.push({})
156                 lastImg = personData.image_id
157             }
158
159             playerPos.slice(-1)[0][parseInt(personData.idx, 10)] = [
160                 personData.box[0] + personData.box[2] / 2,
161                 personData.box[1] + personData.box[3]
162             ]
163         })
164
165         console.log("Loaded_pose_json.")
166     }
167
168     reader.readAsText(this.files[0])
169 })
170
171 function drawLine() {
172     boardContext.lineWidth = 4
173     boardContext.strokeStyle = "white"
174
175     boardContext.clearRect(0, 0, boardCanvas.width, boardCanvas.height)
176
177     boardContext.beginPath()

```

```
178     boardContext.moveTo(...COURT_CORNERS_ON_BOARD[0])
179     boardContext.lineTo(...COURT_CORNERS_ON_BOARD[1])
180     boardContext.lineTo(...COURT_CORNERS_ON_BOARD[2])
181     boardContext.lineTo(...COURT_CORNERS_ON_BOARD[3])
182     boardContext.closePath()
183     boardContext.stroke()
184
185     boardContext.beginPath()
186     boardContext.moveTo(...COURT_GATES_ON_BOARD[0])
187     boardContext.lineTo(...COURT_GATES_ON_BOARD[1])
188     boardContext.lineTo(...COURT_GATES_ON_BOARD[2])
189     boardContext.lineTo(...COURT_GATES_ON_BOARD[3])
190     boardContext.closePath()
191     boardContext.stroke()
192
193     boardContext.beginPath()
194     boardContext.moveTo(0, COURT_CANVAS_HEIGHT / 2)
195     boardContext.lineTo(COURT_CANVAS_WIDTH, COURT_CANVAS_HEIGHT / 2)
196     boardContext.stroke()
197 }
198
199 drawLine()
```

---



## 参考文献

- [1] 日本スポーツ協会、“公認スポーツ指導者養成テキスト 共通科目 III”、2016
- [2] 渡辺裕、“スポーツ情報処理の研究開発動向”、映像情報メディア年報 2018 シリーズ、2018
- [3] Genius Sports Italy Srl, “Data Volley 4”,  
‘<https://www.dataproject.com/Products/EN/en/Volleyball/DataVolley4>’, 2018
- [4] Dario P., et al., “3D human pose estimation in video with temporal convolutions and semi-supervised training”, In *Computer Vision and Pattern Recognition (CVPR) 2019*, 2019
- [5] MVIG-SJTU, “AlphaPose”, ‘<https://github.com/MVIG-SJTU/AlphaPose>’, 2021