F-033 野球ゲームにおける盛り上がり箇所の自動検出 －光学文字認識の一考察－

A Study of Optical Character Recognition Analysis for Hot Spots Auto Detection In Baseball TV Game

赤木　信也*
Shinya Akagi
*NTTデータ先端技術株式会社, Email: shinyaresearchresource@gmail.com

はじめに

動画の盛り上がり箇所の検出は, ハイライト動画の作成に応用することができ, 動画編集を手助けできる技術である. 盛り上がりと一言で述べても定義は様々であり,「草」や「w」といったコメントに注目したもの[1], 笑い声や相槌に注目したもの[2]が存在する. また, ハイライト動画の作成に関する研究として, 音声や画像に着目したものがいくつか研究されている[3].
著者の過去研究[4]において, 切り抜き動画を対象とした盛り上がり箇所の検出実験を実施したが, 音声振幅やコメントを用いた方法では,盛り上がり箇所をピンポイントで推定できた結果を上位10箇所に含む場合が10件中1件程度しかなく, 検出精度の低さが課題として挙げられていた. 特にプロ野球の練習試合動画については, 音声振幅を用いた方法では得点シーンやホームラン箇所の推定に失敗しており, 個別に最適化された方法を試した方が良いという結果が得られていた.
その後の過去研究[5]において, 画像解析によって盛り上がり箇所（ピンチ, チャンス, 得点シーン, ホームラン）を検出できないか分析したところ, テンプレートマッチングを利用することで高い精度で盛り上がり箇所を検出できることが示唆された.
本研究では, 場面の切り分け・結合段階を見据えて, 過去研究で不足していた項目（ボール, ストライクなど）の追加調査を実施するとともに, 場面の重要性評価に利用する試合の得点情報について, 現在の光学文字認識技術でどの程度の精度で抽出できるか検証した結果を報告する.

研究の技術領域

本研究の目的は,ハイライト動画の作成を支援または自動化することにある.本研究の技術領域は下記の通りいくつか分かれており, 盛り上がり箇所の検出は１番目の技術領域である.
（1）盛り上がり箇所の検出
（2）場面の切り分け・結合
（3）タイトル・説明文の作成

過去研究

野球などのスポーツに対するハイライト動画の作成方法としては, (1)カラーヒストグラム, カメラショットの種類, 動作情報などの視覚的特徴, およびゼロクロスレート, 周波数スペクトル, 信号エネルギーレベルなどの音声特徴を利用する方法, (2)字幕を利用したり, スコアボックスの内容を光学文字認識を適用して利用したりする方法, (3)スローモーションリプレイを検出して利用する方法, (4)マルチモーダルのアプローチを利用する方法が研究されてきた[6]. マルチモーダルのアプローチの研究としては, GMMを用いた得点シーンに関する6種類のラベル付与について, 音声だけの場合はF値が最大21.0%, 動画像だけの場合はF値が最大35.8%, 音声と動画像を組み合わせた場合はF値が最大45.5%となるという研究がある[7].
本研究では, プロ野球ゲームに対して, (1)視覚的特徴を用いる方法, (2)光学文字認識を用いる方法を適用した結果について報告する.

プロ野球ゲームの特徴

研究対象とした実況パワフルプロ野球2022の特徴としては,投球・打撃時にストライク, ボール, アウトのカウント, ランナーの有無が固定の位置に表示されたり, 得点時に固定の位置に総得点が表示されたり, ホームラン時やゲームセット時に総得点が表示されたりする. これらの視覚的特徴を利用し, 画像解析を用いてピンチやチャンス, 得点シーン, ホームラン, ゲームセットを自動検出する方法を検討した.

ピンチ, チャンスの自動検出方法

野球におけるピンチ, チャンスは, ランナーが2塁以上にいる状態を指す. ランナーの状態は, なし, 1塁, 2塁, 3塁, 1・2塁, 1・3塁, 2・3塁, 満塁の8パターンであり, 固定の位置に表示される. ただ, 常に表示されている訳ではなく, 打者が打ったり走者が盗塁したりした場合などには画面が切り替わり, 一時的に非表示となる. パターンが少ないことおよび一時的な非表示にも対応できることを踏まえ, 予めランナーの表示パターンを取得しておき, 各表示パターンでテンプレートマッチングを実施してランナーの状態を分類する方法を採用した.

アウト, ボール, ストライクの自動検出方法

アウトカウントはヒッティングの結果, ボール, ストライクカウントは一球ごとの結果を分析することに役立ち, 場面の切り分け・結合に利用できると考えられる. アウトの状態は, 0アウト, 1アウト, 2アウトの3パターン, ボールの状態は, 0ボール, 1ボール, 2ボール, 3ボールの4パターン, ストラクの状態は, 0ストライク, 1ストライク, 2ストライクの3パターンであり, 固定の位置に表示される. こちらも常に表示されている訳ではなく, 打者が打ったり走者が盗塁したりした場合などには画面が切り替わり, 一時的に非表示となる. こちらもパターンが少ないことおよび一時的な非表示にも対応できることを踏まえ, 予め各種表示パターンを取得しておき, 各種表示パターンでテンプレートマッチングを実施してアウト, ボール, ストライクの状態を分類する方法を採用した.

得点シーンの自動検出方法

ホームランを除く得点シーンでは, 得点時に固定の位置に総得点が表示される. 具体的には「C 0 3回表 1 M」のような総得点表示がなされる. 得点シーンについても, 画像の共通部分として「回」の部分があり, テンプレートマッチングを適用できそうだったため, 予め「回」の部分の画像を取得しておき, テンプレートマッチングを実施して得点シーンの検出を行う方法を採用した.

ホームラン, ゲームセットの自動検出方法

ホームラン時にはホームランという文字が表示される. 過去研究では, 光学文字認識の日本語OCR精度および速度が高くなさそうだったため, 予めホームランの文字画像を取得しておき, テンプレートマッチングを実施してホームランの検出を行う方法を採用していた. しかし, 試合中の2本のホームラン箇所を両方検出できてはいたが, 再現率0.5, 適合率1.0, F値0.667であり, 他のテンプレートマッチングのF値0.99より精度が低い結果となっていた. これは文字画像の完全一致検索が難しいことに起因しており, 他の動画に適用した際に検出に失敗したり, 誤検知したりする懸念が残っていた.
そこで, 本研究では, ホームランの本塁帰還時、および、ゲームセット時に「0-5」のような総得点が表示されることに着目し, 予め「-」の部分の画像を取得しておき, テンプレートマッチングを実施してホームラン, ゲームセットの検出を行う方法を採用した. なお, ホームランの本塁帰還時とゲームセット時では, 総得点の表示秒数が異なり, ゲームセット時の方が長く表示されるため, 数値基準を設けて判定することにした.

実験方法

実況パワフルプロ野球2022の対戦動画（1時間36秒）において, テンプレートマッチングの閾値調整によって, どの程度まで分類精度を高めることができるか分析した. 本来は別動画を用意して, 検証結果まで分析することが望ましいが, 今回は学習段階までの分析としている. テンプレートマッチングの値が画面ブラックアウト時に1になる問題については, 未然に判定式を用意して正例として検出されないように対処した.
また, 得点情報を光学文字認識でどの程度の精度で検出できるか分析した. 得点情報は, アウト, ボール, ストライクのカウントなどが表示される際に表示されたり, 得点シーンで表示されたりしており, スコアと得点シーンで先攻, 後攻それぞれ取得し, 計4箇所計測した. 光学文字認識としては, Tesseract, Cloud Vision API, Azure AI Visionを比較した.

実験結果

各処理はマルチプロセスで実行しており, 25プロセスを同時に実行した. 1プロセスは単体だと約15分かかる処理だが, マルチプロセスで実行することで, 全て

の処理が完了するのに2時間2分24秒となった.
各テンプレートマッチングについて, 各項目の正解数, 正解率, 再現率, 適合率の結果を表に示す.

項目: 閾値	正解数	正解率	再現率	適合率	F値
ランナー0: 0.80	841	0.996	0.986	0.996	0.991
ランナー1: 0.80	357	0.999	1.000	0.994	0.997
ランナー2: 0.80	140	1.000	1.000	1.000	1.000
ランナー3: 0.80	9	1.000	1.000	1.000	1.000
ランナー12: 0.80	239	0.999	1.000	0.988	0.994
ランナー13: 0.80	101	0.999	1.000	0.990	0.995
ランナー23: 0.80	164	0.999	1.000	0.994	0.997
ランナー123: 0.80	0	1.000	0.000	0.000	
アウト0: 0.90	760	0.805	0.986	0.518	0.679
アウト1: 0.90	652	0.998	0.991	0.998	0.994
アウト2: 0.90	439	1.000	1.000	1.000	1.000
ボール0: 0.90	919	0.996	0.988	0.996	0.992
ボール1: 0.90	565	0.992	0.950	0.998	0.973
ボール2: 0.90	252	0.997	0.956	1.000	0.978
ボール3: 0.90	115	0.999	0.957	1.000	0.978
ストライク0: 0.90	741	0.998	0.991	0.997	0.994
ストライク1: 0.90	626	0.993	0.962	1.000	0.981
ストライク2: 0.90	484	0.995	0.963	1.000	0.981
得点シーン: 0.50	60	1.000	1.000	1.000	1.000
ホームラン&ゲームセット: 0.80	19	1.000	1.000	1.000	1.000

各光学文字認識について, 各項目の正解数, 正解率, 認識精度, 誤認識率, 未検出率, 誤検出率の結果を表に示す. Tesseractの括弧内の数字は, layoutの値である. 正解数が得点シーンの先攻と後攻で異なるのは, 得点が入った瞬間の変更演出によって目視でも数値を認識できず, 得点シーンの正解が一方は存在するが一方はFalseのままとなっているためである. 正解データは真陽性, 誤認識（数値間違い）, 未検出に分類でき, 不正解データは真陰性, 誤検出に分類できる. 認識精度は真陽性／正解数で求められる.

項目: ツール	正解数	正解率	認識精度	誤認識率	未検出率	誤検出率
スコア(先攻): tesseract(6)	1851	0.854	0.768	0.043	0.189	0.057
スコア(後攻): tesseract(6)	1851	0.730	0.521	0.195	0.284	0.054
得点シーン(先攻): tesseract(6)	60	0.866	0.467	0.000	0.533	0.128
得点シーン(後攻): tesseract(6)	56	0.901	0.661	0.071	0.268	0.096
スコア(先攻): tesseract(7)	1851	0.859	0.795	0.043	0.162	0.075
スコア(後攻): tesseract(7)	1851	0.724	0.525	0.193	0.282	0.069
得点シーン(先攻): tesseract(7)	60	0.886	0.467	0.000	0.533	0.107
得点シーン(後攻): tesseract(7)	56	0.899	0.661	0.071	0.268	0.097
スコア(先攻): tesseract(8)	1851	0.829	0.727	0.081	0.192	0.066
スコア(後攻): tesseract(8)	1851	0.713	0.501	0.300	0.199	0.066
得点シーン(先攻): tesseract(8)	60	0.926	0.717	0.117	0.167	0.070
得点シーン(後攻): tesseract(8)	56	0.935	0.661	0.036	0.304	0.060
スコア(先攻): cloud vision api	1851	0.939	0.890	0.006	0.104	0.010
スコア(後攻): cloud vision api	1851	0.776	0.566	0.111	0.323	0.006
得点シーン(先攻): cloud vision api	60	0.957	0.950	0.000	0.050	0.043
得点シーン(後攻): cloud vision api	56	0.977	0.643	0.036	0.321	0.018
スコア(先攻): azure ai vision	1851	0.940	0.890	0.004	0.106	0.008
スコア(後攻): azure ai vision	1851	0.914	0.837	0.003	0.160	0.006
得点シーン(先攻): azure ai vision	60	0.963	0.950	0.000	0.050	0.037
得点シーン(後攻): azure ai vision	56	0.971	0.857	0.018	0.125	0.027

光学文字認識4プロセスの処理時間は, Tesseract(6)が43分59秒, Tesseract(7)が42分20秒, Tesseract(8)が58分34秒, Cloud Vision APIが26分11秒, Azure AI Vision（Standard）が20分59秒となった. 別作業もしながら計測しており, 平均値も計測していないので, 参考情報程度の結果である.
Cloud Vision AIはデフォルト設定を使用しており, デフォルトでは言語が自動選択になっているので, 英語を明示的に指定すると精度が変わる可能性はある.
Cloud Vision APIだけの誤認識の特徴として, 「5」が正解の時に「05」と出力されてしまうことがあった.
Tesseract, Cloud Vision APIについて, 初めは解析対象画像サイズは動画標準のものを使用していたが, Azure AI Visionの解析対象画像サイズの最小値が50だったため, 50×50にリサイズして同じ画像サイズで再計測した. その際, 画像サイズを拡大することで認識精度が5〜10%程度改善した項目があり, 画像サイズの調整はOCRの精度を高める上である程度効果があることが判明した. なお, 画像サイズと認識精度の相関性, 最適な画像サイズに関しては未調査である.

考察

処理時間は, 1時間36秒の動画に対して2時間2分24秒かかっており, リアルタイム処理が不可能な時間となっている. しかし, これは1台のPCで25プロセスを実行しているからであり, 複数台構成にすればリアルタイム処理は実現可能だと考える. 光学文字認識の処理は, TesseractよりもCloud Vision APIやAzure AI Visionの方が高速であることが示された.
ランナー, アウト, ボール, ストライク, 得点シーンのテンプレートマッチングは, アウト0の適合率が少々低いが, それ以外は再現度, 適合率, ともに非常に高い値を示しており, 本手法によって高精度な分類が可能であることが示された.
ホームラン&ゲームセットは, 再現率1.000, 適合率1.000であり, 文字画像を用いたホームラン箇所の検出が再現率0.500, 適合率1.000であったことを踏まえると, 文字画像よりも「-」などの共通部分を用いた手法の方が高精度な判定が可能であることが示された. この知見は実際のプロ野球動画にも応用でき, 実際のプロ野球動画においても, ホームランやゲームセット時に総得点を表示すれば, コンピュータを用いたホームラン, ゲームセットの高精度な判定が実現できることを示している. 本研究を踏まえ, コンピュータによる自動解析にも対応した, プロ野球の試合における表示デザイン, 表示変更タイミングなどの業界標準化が進むと良いと考える.
光学文字認識は, Tesseractだと, (6), (7)がましではあるが, 認識精度が45%〜80%程度なので, このまま使用することは難しいという結果が得られた. 未検出率が20%〜50%程度と高く, 数字を適切に検出できていない. Tesseractには学習機能があるので, 数字画像に対する特化学習を実施し, 精度改善できるかは今後の課題である. 認識精度は, Azure AI Visionの80%〜95%が最も高いが, これでもギリギリ実用可能な範囲だと考える. 未検出率が10%程度あるので, こちらが改善されれば90%〜程度の認識精度を実現できそうである. OCR全般の課題として未検出率の向上が求められる.

まとめ

テンプレートマッチングを利用することで, ランナー, アウト, ボール, ストライク, 得点シーン, ホームラン, ゲームセットを高精度で分類および判定できることが示された. 光学文字認識については, Tesseractをそのまま使用することは難しく, 特化学習で精度改善できるか試す価値はあると考える. 現状だと, Azure AI Visionの認識精度が80〜95%程度で一番高く, ギリギリ実用可能な範囲であると考える.

おわりに

本研究で利用したプログラムなどはGithub[8]およびGoogleドライブ[9]に公開している.

参考文献

[1] tf0101: YouTube Liveの生放送から盛り上がった箇所を自動抽出するCLIを作った話, https://qiita.com/tf0101/items/efb2484a0b5b1cdc8291, 2019-07-08
[2] 河原達也, 須見康平, 緒方淳,後藤真孝: 音声会話コンテンツにおける聴衆の反応に基づく音響イベントとホットスポットの検出, 情報処理学会論文誌 52 (12), 3363-3373, 2011-12-15
[3] 望月貴裕: 映像自動要約技術の最新動向, NHK技研R＆D2020年夏号, 4-15, 2020-08-15
[4] 赤木信也: 盛り上がり検出のための音声解析の一考察, 情報科学技術フォーラム講演論文集(FIT), 22巻, 2号, pp.423-424, 2023-08-23
[5] 赤木信也: 野球ゲームにおける盛り上がり箇所の自動検出一画像解析の一考察ー, 情報処理学会全国大会講演論文集, 86巻, 2号, pp.59-60, 2024-03-01
[6] Po-Chyi Su, Chi-Heng Lan, Chin-Song Wu, Zi-Xin Zeng, Wei-Yu Chen: Transition effect detection for extracting highlights in baseball videos, EURASIP Journal on Image and Video Processing volume 2013, 27, 2013-05-04
[7] Koichi Shinoda, Kazuki Ishihara, Sadaoki Furui, Takahiro Mochizuki: Automatic Score Scene Detection for Baseball Video, Large-scale knowledge resources: construction and application, LKR 2008, pp. 226–240, 2008-03.
[8] Github: create_baseball_highlightリポジトリ, https://github.com/ShinyaAkagil/create_baseball_highlight
[9] Googleドライブ: output.mp4, https://drive.google.com/file/d/1sJ0ygXSMOhBk9mCsJXWAqBgZ7Zq-A-jJ/view?usp=sharing