フェンシング競技フルーレ種目における、突きに伴って発光する剣を有した有効突き判定装置の研究開発

フェンシング競技フルーレ種目における、突きに伴って発光する剣を有した有効突き判定装置の研究開発

Research and Development of a Light-Emitting Sword Tip Accompanying Thrusts and a Device for Judging Valid Thrusts in the Foil Event of Fencing Competitions

藤田海渡[1]
Kaito Fujita
粟國晴楽[2]
Seira Aguni
中西蓮[3]
Ren Nakanishi
森竜樹[4]
Tatsuki Mori
西川哲夫[5]
Tetsuo Nishikawa

概要

フェンシング競技フルーレ種目における、電気審判機を利用した得点判定に伴う2つの課題である1）設置の手間と2）判定の分かり難さを解決するために、電気審判機を使わず、剣先を光らせて突きの可視化を行えるように設計し、また、突きの有効性を自動判定する装置のプロトタイプを構築し、その有効性を確認した。

「突き」の瞬間に剣先のブレード上部に張られたLEDを発光させることで、突きの可視化を可能にした。また、「突き」の瞬間の剣先位置を、剣先のブレード下部に張られた白色LEDによって照らされた黄色テープの画像から取得し、剣先位置の周辺画像を鍔の下の外側に設置したカメラで撮影し、周辺画像の画像処理によって、剣先で突いた部位が有効面であるか否かを判定する。これによって、LEDや環境光の影響を受けず、剣先を正確にカメラに捉えることができ、かつ剣先スイッチが押された瞬間の剣先位置を取得することが可能となった。

[1] 工学研究科数理工学専攻
[2] 東京理科大学大学院経営学研究科経営学専攻
[3] 株式会社メイテック
[4] 工学部数理工学科講師
[5] 工学部数理工学科教授

メタルジャケットやユニフォームを使わずに判定が可能な方法、すなわち１）画像中の画素の HSV 色空間を用いた判定方法（前回の論文で提案）に加えて、メタルジャケットとユニフォームを使用した場合の、２）縞模様検出を用いた判定方法を提案し、その有効性を確認した。さらに、HSV 色空間を用いた場合に、有効突き判定の高速化の検討を行い、Raspberry Pi Zero を用いた場合に、剣先検出と有効突き判定の合計時間が 0.087 秒という十分実用的な判定時間を達成できた。

１．はじめに

フェンシング競技のフルーレ種目とは、2 人の選手が向かい合い、片手に持った剣で互いの有効面を攻防する競技である。この判定には従来、電気審判機が使用されている。選手の胴体部分に装着したメタルジャケットを剣で突くと、選手と有線接続されてコート中央脇に設置された電気審判機で検出され、色ランプが光る。

電気審判機には、2 つの問題がある。

１．設置に手間がかかる。

２．判定が見難く分かりにくい。

電気審判機は、赤、青のランプの点灯により、どちらの選手の得点かを表示する。剣が突いた位置と得点ランプの位置が全く異なるので、剣の動きの中のどこでランプが点灯したかの判断が困難である。つまり、いかに技が決まったかが分かりにくい。このことは、フェンシング観戦への興味を減退させ、フェンシングの普及を阻害している可能性がある。この問題の解決には、電気審判機を使わず、①剣が突いた瞬間の剣先位置が可視化できることが望ましい。また、②突いた位置が有効面かどうかを、電気審判機を用いずに判断できる必要がある。

前回、この二つの課題を解決する有効突き判定装置及び有効突き判定アルゴリズムの報告(文献[３])を行った。今回は、文献[３]で報告した有効突き判定装置の剣先検出の方式の改善、及び、メタルジャケットの縞模様検出を用いた判定方法の提案を行った。また、HSV 色空間を用いた場合に、有効突き判定の高速化の検討を行った。尚、本論文のうち有効突き判定装置の概要や使用例、突きの可視化方法と HSV 色空間を用いた有効突き判定アルゴリズムに関しては、文献[３]の英語の内容を日本語で解釈し直したものである。

２．装置の開発

２．１　試作装置の構成と動作

１）有効突き判定装置の動作と構成

①有効突き判定装置の概要

図１は、有効突判定装置の、フェンシングフルーレ種目の対戦時の動作を示す概要図である。フルーレ種目においては、図１（A）に示すように、メタルジャケット A、及びメタルジャケット B をそれぞれ身につけたフェンサーA とフェンサーB が剣 A、B を持って対峙する。図１（A）に示すように、フェンサーA の剣 A がフェンサーB のメ

タルジャケットB上を突いた場合、剣Aに設置された有効突判定装置Aが作動する。

有効突判定装置の動作について、図２を用いて説明する。有効突判定装置は、発光・剣先画像撮影部、中央制御部、記憶部から構成される。発光・剣先画像撮影部は、剣先スイッチ、カメラ、発光部、発音部から構成され、中央制御部では、CPUが有効突判定装置内の各種動作を制御する。記憶部では、有効突判定装置を制御する各種プログラムが記憶される。図２に示す剣先スイッチが押されると、中央制御部を介して、発光部、発音部が作動するとともに、剣先スイッチを視野に含むカメラがシャッターを切り、剣先の剣先画像を撮影し、その剣先画像を基に、剣Aが突いた場所がメタルジャケットB上であるかどうかを判定する。

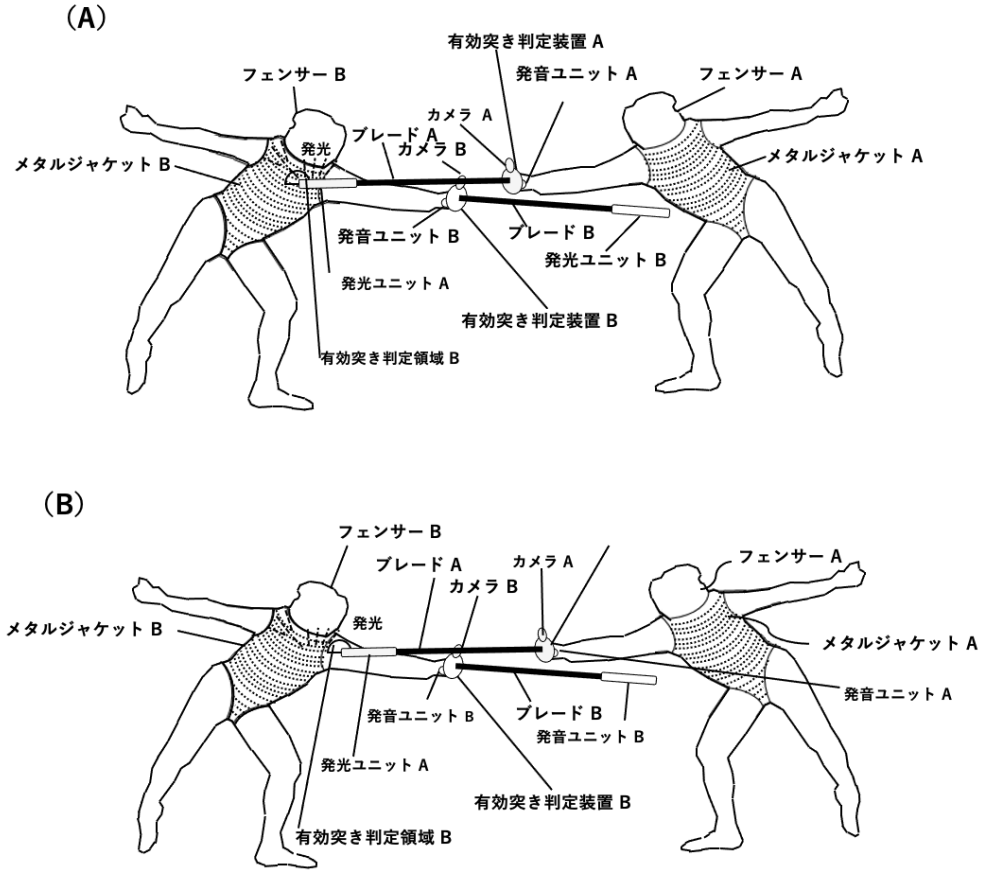

図１　有効突判定装置の、フェンシングフルーレ種目の対戦時の動作を示す概要図

(A) 突きが有効な場合、(B) 突きが有効でない場合

②有効突き判定装置の動作の概要

以上の動作を、図１と図２を用いて説明する。

図１（A）に示すように、フェンサーAの剣Aがフェンサー B のメタルジャケットB上を突いた場合、発光部A（図２における発光部）が発光すると同時に、鍔の外側に設置されたカメラA（図２におけるカメラ）がシャッターを切り、剣Aが突いた瞬間の剣先画像を撮影する。撮影した剣先画像を基に、剣Aが突いた周辺の有効突判定領域Bの剣先周辺画像を取得し、その剣先周辺画像を基に、突いた部分がメタルジャケ

ットB上であることを判定する。判定の方法としては、判定領域の剣先周辺画像の画素の色空間情報、特に色相（H）の分布情報を用いる。

図１（B）では、剣Aがフェンサーの腕の部分を突いており、突きが有効でない場合を示している。この場合は、有効突判定領域Bの剣先周辺画像を基に、突いた部分がメタルジャケットB上でないことを、上記と同様の方法で判定する。

このように、本装置によって、剣A、Bによる突きの瞬間を可視化でき、かつメタルジャケット部分を突けたかどうかの判定を電気審判機なしでできるようになる。これにより、非常に手軽にかつ正確に剣の攻防を認知できるようになり、効率的な練習が可能になる。

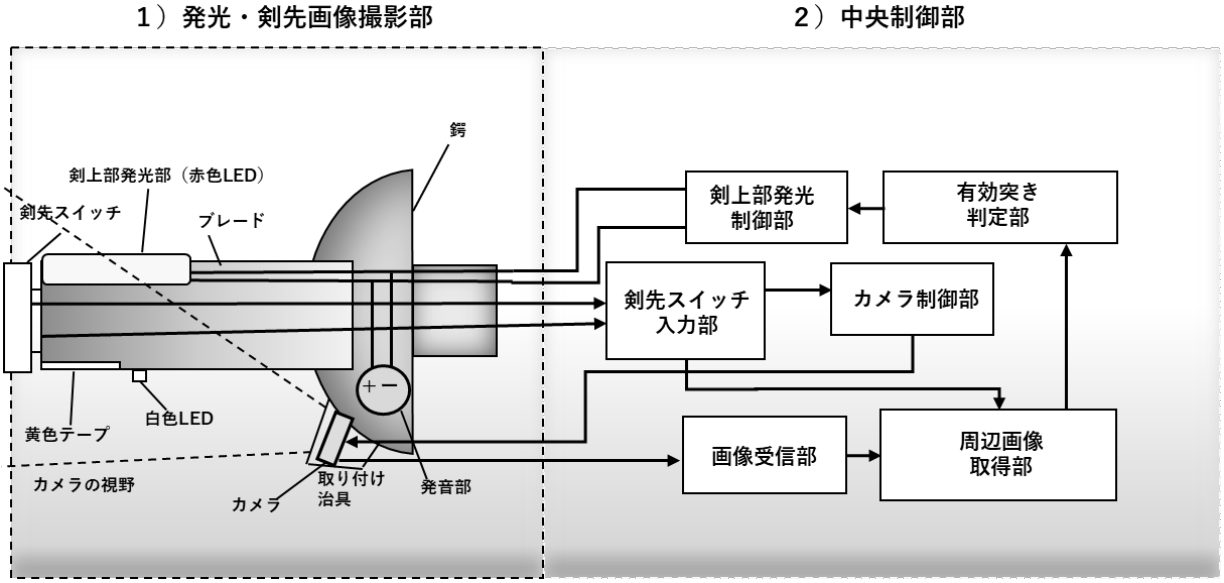

図２　有効突き判定装置の、発光・剣先画像撮影部と中央制御部の内部構成図

③有効突き判定装置の内部構成

図２は、有効突き判定装置の、発光・剣先画像撮影部、中央制御部の内部の構成図である。発光・剣先画像撮影部は、剣先スイッチ、カメラ、剣上部発光部（赤色LED)、発音部、ブレード、黄色テープ、白色LED、取り付け治具、鍔から構成される。また、中央制御部は、剣先スイッチ入力部、カメラ制御部、画像受信部、剣先座標・剣先周辺画像取得部、有効突き判定部、剣上部発光制御部から構成される。

システムの動作の概略を、図２を用いて説明する。発光・剣先画像撮影部と中央制御部について、剣先スイッチが押されてから、有効突き判定を行うまでの流れについて説明する。剣先の下部に黄色テープを付着させており、黄色テープは白色LEDで常に照らされている。剣先スイッチからのコードは、中央制御部内の剣先スイッチ入力部に接続されている。ブレードの先端には、剣上部発光部が設置されており、剣上部発光部には、剣上部発光制御部からのコードが接続されている。

２）有効突き判定装置の動作の詳細

①剣先スイッチの作動から剣先周辺画像の取得までの流れ

剣先スイッチが押されると、剣先スイッチ入力部を介して、剣上部発光制御部へ情報が送られる。剣上部発光制御部は、剣先スイッチ入力部から情報を受け取ると、剣上部発光部と鍔内部に設置された発音部に、それぞれ第 1 発光指示、及び第 1 発音指示を送り、剣上部発光部を一定時間連続的に発光させると共に、発音部を一定時間連続的に発音させる。

剣先に設置された黄色テープは、白色 LED によって常に照らされており、鍔の外側には、取り付け治具によって、カメラがカバーされた状態で設置される。カメラの視野は、剣先スイッチの周辺をカバーしている。剣先スイッチが押されると剣先スイッチ入力部を介し、カメラ制御部に情報が送られ、その情報により、カメラのシャッターが切られる。シャッターが切られるタイミングは、剣上部発光部の発光、及び発音部の発音のタイミングと同期しており、突いた瞬間を可視化することができる。撮影された剣先画像は、画像受信部で受信され、剣先周辺画像取得部に送られ、剣先座標とその周辺画像に基づく剣先周辺画像の取得処理が行われる。

剣先座標の取得は、白色 LED によって照らされた黄色テープの画像を取得し、黄色テープ領域の y 座標の最大位置の座標として取得する。

②突きの可視化

LED 制御部は、有効突き判定部から情報を受け取ると、発光部と発音部に、それぞれ第 2 発光指示、及び第 2 発音指示を送る。突きが有効と判定された場合、第 2 発光指示、及び第 2 発音指示では、それぞれ発光部を一定時間点滅させ、発音部に一定時間断続音を発音させる。突きが無効と判定された場合、第 2 発光指示、及び第 2 発音指示では、発光部、及び発音部は、それぞれ発光、及び発音を行わない。

このようにして、剣 A、B による突きの瞬間を可視化でき、かつメタルジャケット部分を突いた場合は、最初に一定時間、剣先の発光部が連続的に発光し、発音部が連続的に発音した後、一定時間発光部が点滅し、発音部が断続音を発音するが、メタルジャケット以外の場所を突いた場合は、一定時間の発光、発音の後、何も起らないこととなり、メタルジャケット以外の場所を突けたかどうかの判定を電気審判機なしで可能になる。

3）先行技術の改良点

文献[３]では、剣先の上部に設置された青色 LED の発光によって、突きの可視化を実現し、同時に青色 LED の画像を青色でマスク化して検出する方法で、剣先検出も行っていた。ところが、青色 LED の発光はメタルジャケットに反射するため、青色でマスク化する方法では、正確な剣先座標を求めることができない場合があった。

また、剣先スイッチが押されてから青色 LED を発光させ、剣先座標を求める方法では、青色 LED の発光に 0.1 秒程度の時間がかかり、その間にブレードが曲がり剣先位置がカメラの視野から外れてしまうことがあった。

そこで、今回は、LED の役割毎に、2 つの LED を用いることとした。剣先の上部に設置された青色 LED は、突きの可視化のために用い、剣先の下部に設置された白色 LED は、直接剣先位置を求めるのではなく、剣先の下部に貼られた黄色テープを常時照射し、その反射光(剣先スイッチが押された瞬間の)から、剣先座標を求めることとした。このようにすることによって、LED や環境光の影響を受けず、剣先を正確にカメラに捉えることができ、かつ剣先スイッチが押された瞬間の剣先位置を取得することが可能となった。

取得された剣先周辺画像は、有効突き判定部に送られて、有効突きの判定が実行され、その結果が LED 制御部に送られる。有効突きの判定は、前回の報告[３]では、HSV 色空間を用いた方法を提案したが、今回は、それに加えて縞模様の検出を用いた方法を提案した。有効突きの判定の詳細については、3 章で行う。

４）有効突き判定装置の試作機

図 3 は、作成した有効突き判定装置の試作機の写真である。鍔は、モデリングを行なった後に 3D プリンターで作成した。剣の上部には、赤色 LED テープを設置しており、剣で突いた直後や有効突き判定後に発光する。カメラは鍔の下部に固定されており、剣先の黄色テープを常に画角に捉えている。なお、黄色テープは白色 LED で常に照らされており、はっきりとカメラで撮影することができる。中央制御部には、Raspberry Pi Zero を使用している。Raspberry Pi Zero は、マイクロコンピュータの Raspberry Pi シリーズの一種である。通常、Raspberry Pi 3 を用いられることが多いが、Raspberry Pi Zero は、サイズ(65mm×30mm×9mm)がとても小さく、またとても軽量(22g)であり、今回は鍔を可能な限りコンパクトに設計するために使用する。但し、処理性能は Raspberry Pi 3 と比較して非常に低いため、アルゴリズムの最適化を検討する必要がある。

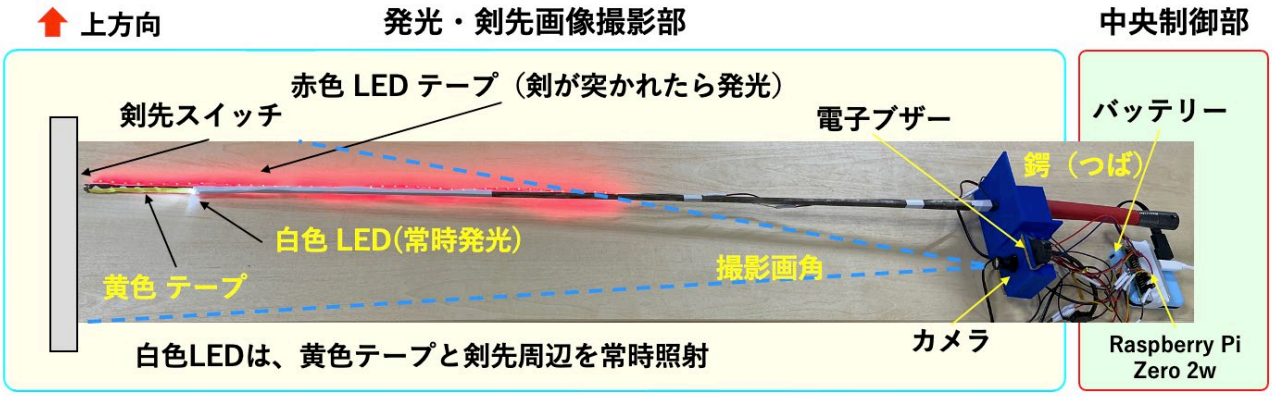

図３　有効突き判定装置の試作機の写真

2．２　剣先周辺画像取得部

剣先周辺画像取得部の動作について、図４、図５を用いて説明する。なお、画像処理は、プログラミング言語 Python と画像処理ライブラリーである OpenCV を用いて行

った。

図４は、突いた瞬間の剣先画像からの剣先座標と剣先周辺画像の取得方法を示す概要図であり、

図４(A)は、突いた瞬間の剣先画像を示し、

図４(B)は、剣先座標の取得方法を示し、

図４(C)は、剣先周辺画像の取得方法を示している。

図４(A)の突いた瞬間の剣先画像には、剣先でメタルジャケットを突いた画像が表れる。カメラの解像度は 640×340 で撮影を行った。この画像から、図４（B）剣先座標の取得方法に示すように、剣先画像領域を取得した後、剣先位置を特定し、剣先座標を取得する。その後、図４（C）剣先周辺画像の取得に示すように、剣先位置の上部の位置から、円形の剣先周辺画像を取得する。

次に、剣先座標の取得と剣先周辺画像の取得について、図５を用いて説明する。図５(A)は、剣先座標と剣先周辺画像の取得方法の処理の流れを示すフローチャートである。剣先座標の取得では、まず処理 1-1）で、剣先画像に含まれる黄色テープの画像画素を用いて、その HSV 分布を取得する。

次に処理 1-2)で、取得した HSV 分布を用いたマスクを作成し、剣先画像のマスキング処理を行い、マスキング画像を出力する図５(B)。マスキング処理については、文献[４]を参考にした。処理 1-3)では、領域同士がつながるようにマスキング画像の膨張処理を行った上で、輪郭を取得する。輪郭で囲まれた領域としては、通常、剣先以外のノイズ等に起因する複数の領域が検出されるので、処理 1-4)では、ノイズを除くために、それら複数の領域から面積が最大の領域を一つ選び、その領域内の点の y 座標が最大の点の位置座標を剣先座標として取得し、処理 1-3)と処理 1-4)の結果の画像を出力する図５(C)。膨張処理に関しては、文献[５]を参考にした。また、輪郭の取得方法に関しては、文献[５]を参考にした。画像領域の上部の白い点が剣先位置である。その後、処理 2)で、剣先周辺画像を、剣先座標の円形の領域のマスクを用いて取得する。円の半径はパラメータとして変更できるようにする。処理 2)の結果の画像を出力する図５(D)。

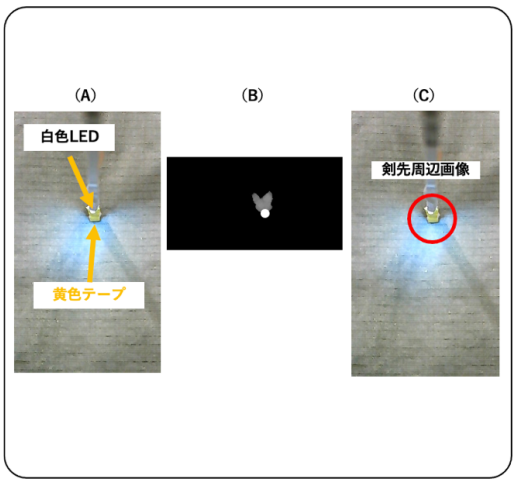

図 4 剣先座標と剣先周辺画像の取得方法

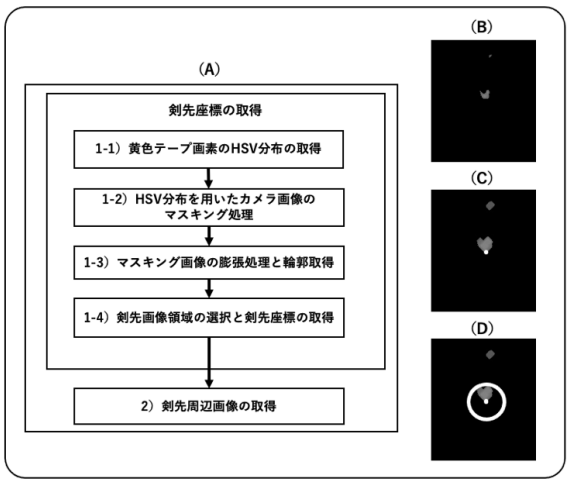

図 5 剣先座標と剣先周辺画像の取得方法のフローチャート

3．有効突き判定

3．1　HSV 色空間を用いた有効突き判定

1 ）HSV 色空間を用いた有効突き判定アルゴリズム

図６、図７を用いて、有効突きの判定を行うアルゴリズムの検討のための分析結果について説明する。図６は、剣先周辺画像を取得から HSV 取得までを示した図である。図７は、剣先周辺画像の HSV 画像分布のヒストグラムを、メタルジャケットとワイン色ジャージで比較した図である。剣先周辺画像を用いて、画像の RGB 値を HSV 色空間に変換した上で、色相(H)、彩度(S)、明度(V)毎に、画素数頻度のヒストグラムを取得した。試合の際は、通常メタルジャケットの下にユニフォームを着るが、本アルゴリズムは、フェンシング競技の未経験者向けて、フェンシング用具を所持していない方を想定しているため、ユニフォームではなくワイン色のジャージを使うこととした。この手法は、長袖シャツとベストなどメタルジャケットとユニフォームと似た形の服、かつ色が異なっているものであれば、フェンシング競技フルーレ種目の判定をすることが可能である。

図７(A)は、メタルジャケット上の剣先周辺画像の HSV 画像分布のヒストグラムであり、図７(B)は、ワイン色のジャージ上の剣先周辺画像の HSV 画素分布のヒストグラムである。横軸が HSV 値で、縦軸が画素頻度である。H、S、V それぞれのヒストグラムを示している。メタルジャケット上の剣先周辺領域、及びジャージ上の剣先周辺領域の画像のヒストグラムを示している。H、S、V 共に、メタルジャケットとワイン色のジャージ上の位置で、異なるピーク位置の単独のピークが得られており、特に H のピークがシャープであることがわかる。S と V の分布は、色の明るさに起因しており、周囲の明るさによって分布が変化してしまうため、有効突きの判定には使わないこととした。

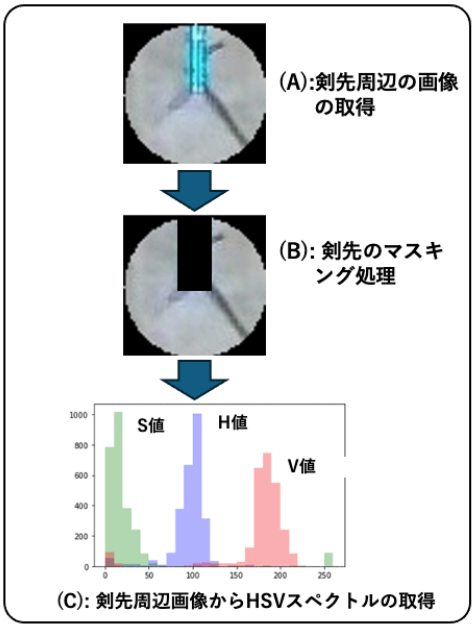

図６　剣先周辺画像のマスキング処理と HSV スペクトルの取得

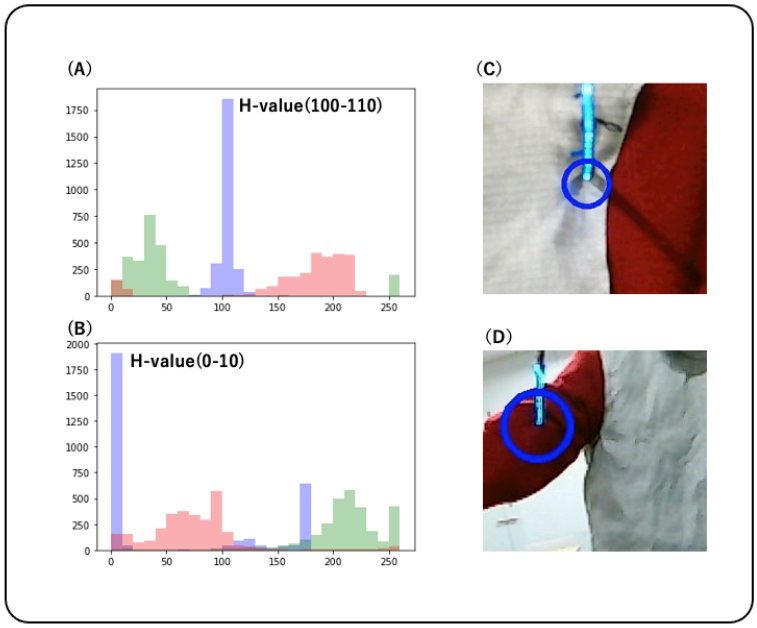

図７　HSV 色画像分布のヒストグラムを、メタルジャケットとワイン色ジャージで比較した図

2 ）HSV 色空間を用いた有効突き判定フロー

① HSV 色空間を用いた有効突き判定の前処理

先ず、図８(A)有効突き判定の前処理について説明する。この処理では、メタルジャケットとワイン色ジャージの H 分布の分割閾値の取得処理を行う。最初に、処理(A-1)では、メタルジャケット上(1)とワイン色ジャージ 上(2)で剣先周辺画像を取得し、HSV 色空間への変換を行う。次に、処理(A-2)では、(1)、及び(2)で、HSV 色空間のうち、H のヒストグラムを取得する。更に、処理(A-3)では、(1) の H ヒストグラムにおいて、H 分布のある値 Ht を変化させながら､Ht 以下の頻度の積算値 SHLm(Ht)の計算を行い､(2)の H ヒストグラムにおいて、H 分布のある値 Ht を変化させながら、Ht 以上の頻度の積算値 SHUw(Ht)を計算する。処理(A-4)では、SHLm(Ht)≒SHUw(Ht)となるような、H 分布の値 Ht=Ht0（分割閾値と呼ぶ）を求めて、有効突き判定の前処理を終了する。

ヒストグラムの全積算値は領域の画素数であり、測定ごとに同一である。従って、メタルジャケットとワイン色ジャージのどちらの領域を突いたかを判断する際に、ここでは、H のピーク強度ではなく、画素数の積算値を用いることとした。

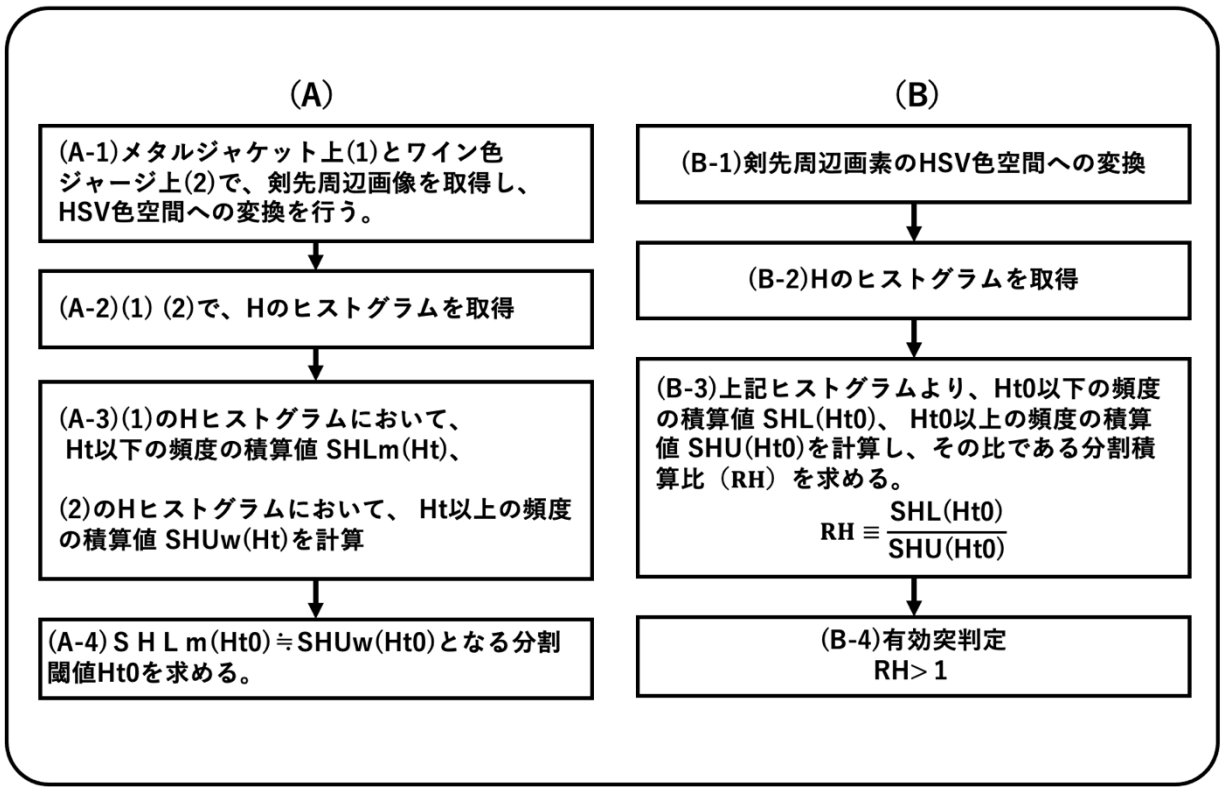

図８　有効突判定の前処理と有効突判定処理の方法のフローチャート

(A)は有効突判定の前処理として行う、メタルジャケットとワイン色ジャージの H 分布の分割閾値の取得処理
(B)は有効突判定処理として行う、H 分布の分割積算比（RH）を用いた判定処理

図９は、メタルジャケットとワイン色ジャージの H ヒストグラムから求めた Ht=Ht0 による H の分割の様子を示しており、図９(A)は、突き位置がメタルジャケットの場合の、HSV 分布上の Ht0 を示し、図９(B)、は突き位置がワイン色ジャージの場合の、HSV 分布上の Ht0 を示している。図９(A)のピーク位置の H と図９(B)のピーク位置の H の間の位置に、Ht=Ht0 が位置しており、Ht0 値以下の(A) の分布の積算値 SHLm(Ht0)と、Ht0

値以上の(B)の分布の積算値 SHUw(Ht0)が等しくなっていることが分かる。本処理は、競技の前に行う。この結果は、無線通信を用いて外部 PC からみられるようにして、Ht0 の取得結果を H ヒストグラムと共に可視化できるようにしておく。これによって、Ht0 の取得処理の結果の確認とマニュアルによる修正が可能になる。

②HSV 色空間を用いた有効突き判定処理

次に、図８（B）有効突き判定処理の手順を説明する。本処理は、競技中の処理である。本処理は、処理(A-4)で求めた Ht0 を境界として、それ以上と以下で H スペクトルの画素頻度の積算を取り、どちらの画素数が多いかを判定する。先ず、処理(B-1)で、剣先周辺画素の HSV 色空間への変換を行う。次に、処理(B-2)で、HSV 色空間のうち、H 値のヒストグラムを取得する。更に、処理(B-3)で、取得したヒストグラムから、Ht0 以下の画素頻度の積算値 SHL(Ht0)、Ht0 以上の画素頻度の積算値 SHU(Ht0) を計算し、数式(1 に示すように、その比である H 分布の分割積算比 RH を求める。

$$\mathbf{RH} \equiv \frac{\mathrm{SHL(Ht0)}}{\mathrm{SHU(Ht0)}} \quad \cdots (1)$$

最後に、処理(B-4)で、処理(B-3)で求めた RH が 1 以上の場合を有効突とし、1 未満の場合を無効突きとして、判定を終了する。この判定処理を、メタルジャケットとワイン色ジャージの境界領域へ適用した例を図９に示す。

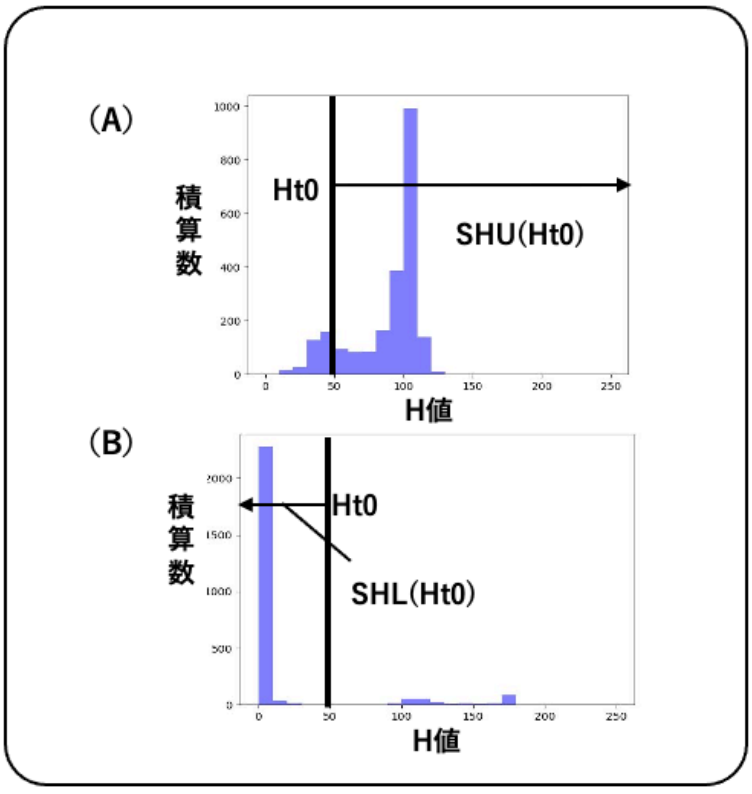

図９　メタルジャケットとワイン色ジャージの H ヒストグラムから求めた Ht=Ht0 による H の分割の様子

(A)は突き位置(メタルジャケット)の HSV 分布上の Ht0
(B)は突き位置(ワイン色ジャージ)の HSV 分布上の Ht0

③有効突判定処理の判定処理の一例

図 10 は、有効突き判定処理の H 分布の分割積算比(RH) を用いた判定処理の 1 例を示した図である。図 10(A1)は、境界付近のメタルジャケット側を突いた場合、図 10(B1)は、境界付近のジャージ側を突いた場合である。図 10(A2)は、図 10(A1)の剣先周辺領域の H 分布と、分割積算比(RH)の値を示している。同様に、図 10(B2)は、図 11(B1)の剣先周辺領域の H 分布と、分割積算比(RH) の値を示している。

図 10(A1)では、剣先周辺領域の中のメタルジャケットの方が、図 10(B1)では、剣先周辺領域の中のジャージ側の方が、面積が大きいことが分かる。このことを反映して、図 10(A2)、図 10(B2)に示すように、図 10(A1)での RH 値は 2.3、図 10(B1)での RH 値は 0.7 となる。従って、有効突き判定は、図 10(A1)が有効、図 10(B1)が無効となり、正確に判定していることが分かる。

上記の判定方法は、以下のように解釈することができる。すなわち、積算値 SHLm(Ht0)が有効面画素の積算値であり、積算値 SHUw(Ht0)が、非有効面画素の積算値である。有効突き判定部は、剣先周辺画像に含まれる有効面画素と非有効面画素の各々の画素数をカウントし、有効面画素の画素数が該非有効面画素の画素数に比べて大きいときに、有効突きと判定する。

④有効突判定処理において、判別境界が直線でない場合に対応したアルゴリズムの 1 例

上記の方法では、剣先周辺画像内のメタルジャケットとワイン色ジャージの境界が直線でない場合、判定誤差が大きくなるものと考えられる。誤差を減らすためには、メタルジャケットとワイン色ジャージの境界の形を認識する必要があると考えられる。そのための具体的な方法の例として、以下の方法を示す。

(1) 剣先周辺画像内の各画素の周辺の一定範囲のウィンドウ領域画像を考え、ウィンドウ領域毎に上記で示した有効突き判定処理を行い、その結果を画素毎の 1 (有効突きの場合)、0 (有効突きでない場合) で出力する。
(2) 画素毎の有効突き判定結果を、ウィンドウ領域がとれる全ての画素について集めて、剣先周辺画像上の 1、0 のパターンを得る。
(3) このパターンを 2 次元平面上の 1、0 の散布図とみなし、非線形 SVM などの判別分析方法を用いて、1、0 の判別境界曲線を求める。
(4) 有効突き判定は、判別境界曲線によって分けられた領域のどちら側に、剣先座標が属するかによって、有効か無効かを判定する。

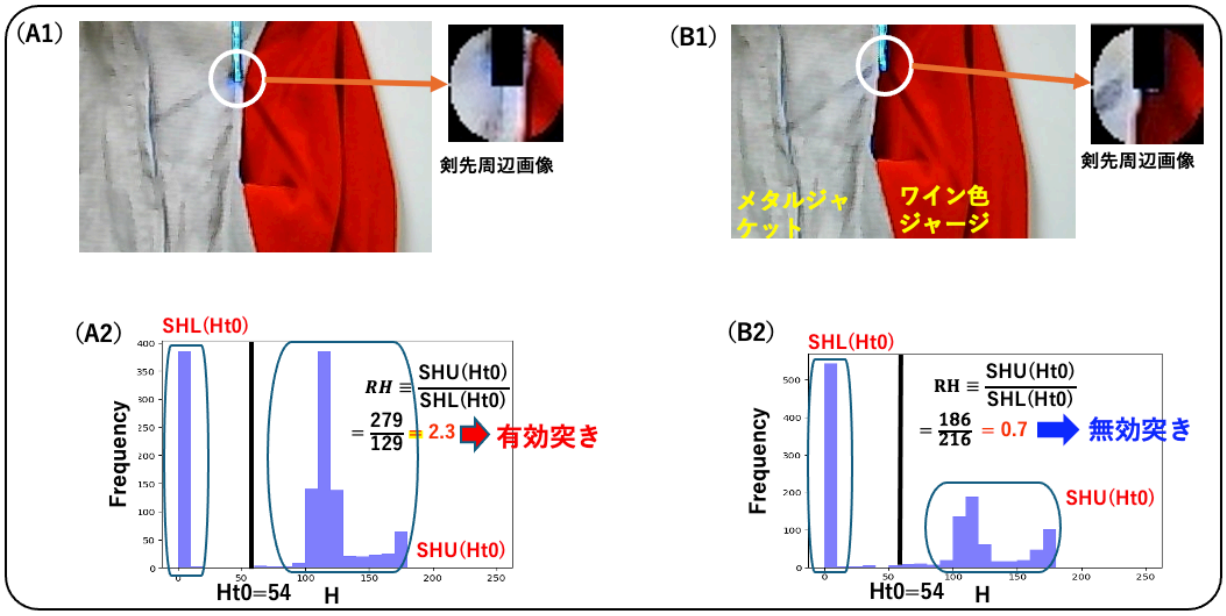

図　10 は、有効突判定処理の H 分布の分割積算比（RH）を用いた判定処理の一例を示した図

(A1)は、境界付近のメタルジャケット側を突いた場合
(B1)は、境界付近のジャージ側を突いた場合
(A2)は、図(A1)における突き位置の HSV 分布
(B2)は、図(B1)における突き位置の HSV 分布

3 ）HSV 色空間を用いた有効突き判定の高速化の検討

剣先検出、及び有効突き判定の処理方法について、それぞれの処理時間に関するアルゴリズムの最適化の検討を行った。表 1 に、各種条件における処理時間を示す。

マイクロコンピュータの Raspberry Pi シリーズでは、Raspberry Pi 3 が用いられることが多いが、今回は、有効突き判定装置をよりコンパクトに設計するために、Raspberry Pi 3 の半分以下の重さ（表 1）の Raspberry Pi Zero を使用する。しかし、Raspberry Pi Zero は、Raspberry Pi 3 と比較して情報処理性能が低いことが知られている。そこで、まず有効突き判定時間を、高速化の検討なしで、Raspberry Pi 3 と Raspberry Pi Zero を用いて計測し、比較を行った。

撮影した画像の全画素に対して処理を行った結果を表 11 の No.1、No.2 を用いて説明する。Raspberry Pi 3 で実行した場合、合計処理時間は 0.25 秒であったのに対して、Raspberry Pi Zero では、合計処理時間は 4.3 秒を要していた。

フェンシングのプレー中の突きの認識を考えた場合、突きが成立してから長くても 1 秒以内での判定が必要とされると考えられる。Raspberry Pi 3 の 0.25 秒という時間は、これよりも短く、実用になると考えられる。一方、Raspberry Pi Zero での 4.3 秒という時間は、時間が掛かりすぎてとても実用的とは言えない。そこで、ここでは、Raspberry Pi 3 と同程度の処理時間を達成することを目標として、アルゴリズムの最適化を検討した。

表 1 の No. 3 は高速化の検討を行った結果を示している。剣先座標の検索範囲の最適化、有効突き判定を行う際の剣先周辺画像の間引きの導入やサイズの検討を行った結果、合計処理時間が 0.087 秒と Raspberry Pi 3 よりも高速に処理をすることが可能になった。高速化の検討内容の詳細に関しては、次の①と②で示す。

表 1 有効突きの判定処理時間

No. 1 は、 Raspberry Pi 3 で、高速化処理を行わずに実行した結果
No. 2 は、 Raspberry Pi Zero で、高速化処理を行わずに実行した結果
No. 3 は、 Raspberry Pi Zero で、高速化処理を行い実行した結果

No.	CPU	Size	高速化	剣先検出(秒)	有効突き判定(秒)	合計処理時間(秒)
1	RaspberryPi 3	86 (W) x 57 (D) x 17 (H) mm, 50g	なし 全画素に対して処理	0.10	0.15	0.25
2	RaspberryPi Zero	65 (W) ×30 (D) ×9 (H) mm, 22g	なし 全画素に対して処理	0.90	3.4	4.3
3	RaspberryPi Zero	65 (W) ×30 (D) ×9 (H) mm, 22g	あり 部分的な画素に対して処理	0.047	0.04	0.087

① 剣先座標の検索範囲の検討

図 11 は、取得した剣先画像内で、剣先を検出した座標の分布を示している。この図から、剣先を検出する座標は集中しているため、青い矩形で囲った範囲から剣先の検索を行えば良いことが分かる。剣先の検索範囲と検索時間の変化について、図 12 を用いて説明する。図 12(A)は、剣先画像の y 軸の大きさは変更せずに、x 軸を 640 ピクセルから 90 ピクセルまで変化させたときの検索時間の変化を示した図である。図 12(B)は、剣先画像の x 軸の大きさを 90 ピクセルに固定し、y 軸の大きさを 360 ピクセルから 110 ピクセルまで変化させたときの検索時間の変化を示した図である。剣先画像全体から剣先の検索をした場合、剣先の検出には、0.9 秒であったのに対して、検索場所を狭めた場合、約 20 分の 1 の時間である 0.047 秒まで減少させることができた。

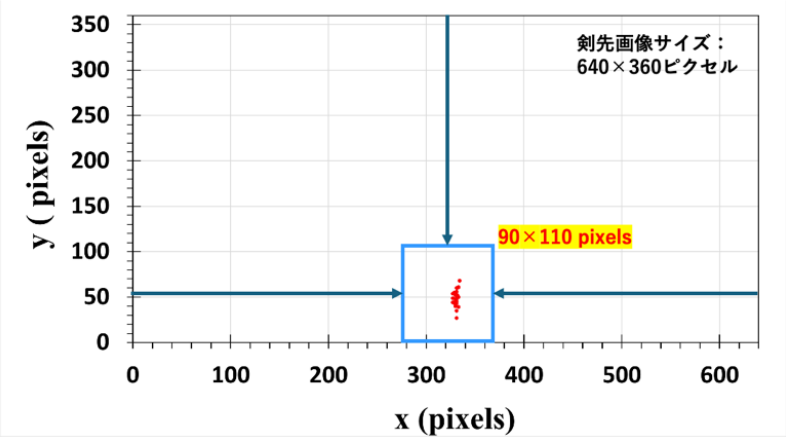

図 11 様々な突きにおける剣先座標の分布

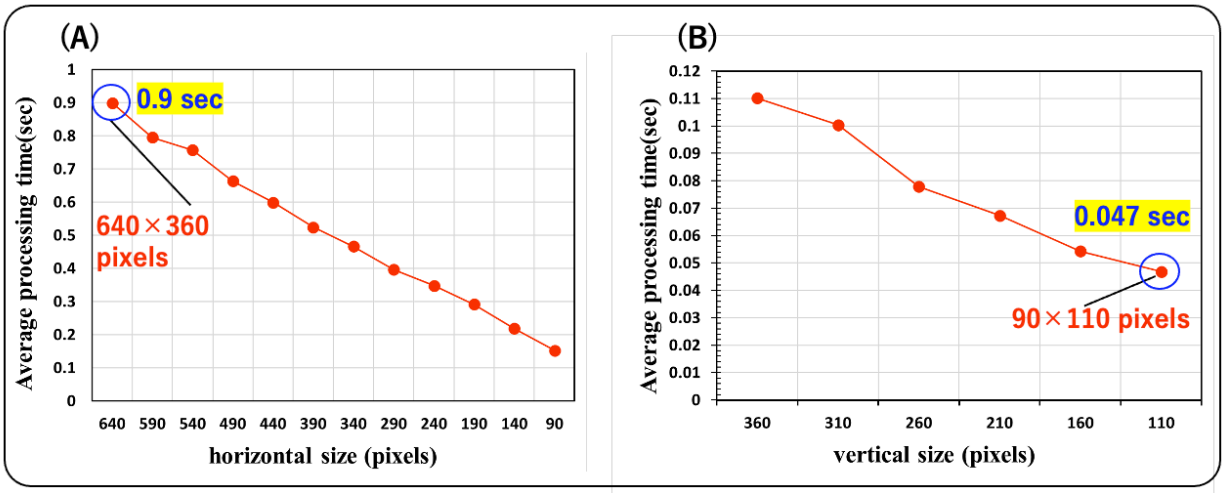

図 12 剣先座標検出時間の剣先画像のサイズへの依存性

(A)は、 y 軸方向のサイズを 360px に固定させ、x 軸座標方向のサイズを 640px から 90px まで縮小させた際の剣先座標検出時間の変化
(B)は、x 軸方向のサイズを 90px に固定させ、y 軸方向のサイズを 360px から 110px まで縮小させた際の剣先座標検出時間の変化

② 剣先周辺画像の間引きとサイズの検討

剣先周辺画像は、比較的均一であるため、画像のピクセルを間引いて有効突きの判定を行うことが可能である。図 13 は、間引き処理を表すイメージ図である。また、剣先周辺画像のサイズは精度が落ちない範囲で小さく設定することができる。

図 14(A)に、間引き回数と有効突き判定に要する平均時間を示す。有効突き判定に要する時間は間引き回数と依存関係にあることが分かる。間引き処理を行わない場合と比較し、間引き回数が 2 回、すなわち 3 ピクセルおきに画像処理を行うことで、処理時間を 1/9 にすることが可能になった。

図 14(B)に、剣先周辺画像の 1 辺のサイズと有効突き判定に要する時間の関係を示す。青いプロットは間引き処理を行わない場合(間引き回数が 2 回)、赤いプロットは間引き処理を行った場合、かつ、剣先周辺画像の一辺のサイズを変更した際の処理時間を示している。先ほど示したように、剣先周辺画像の一辺のサイズが 60 ピクセルの場合、間引き処理の有無に関する比較をすると、間引き処理を行わない場合、有効突き判定時間が 3.3 秒だったのに対して、間引き処理を行う場合、有効突き判定処理は 0.37 秒であった。ここから、処理時間が約 1/9 になっていることが確認できる。

剣先座標の剣先範囲や剣先周辺画像の間引きとサイズの最適化を行ったことで、合計処理時間が 0.087 秒に短縮することができた。この結果は、高速化処理を行わずに Raspberry Pi 3 を用いて実行した結果よりも高速に処理を行うことが可能になっており、実用的な判定処理時間であると言える。

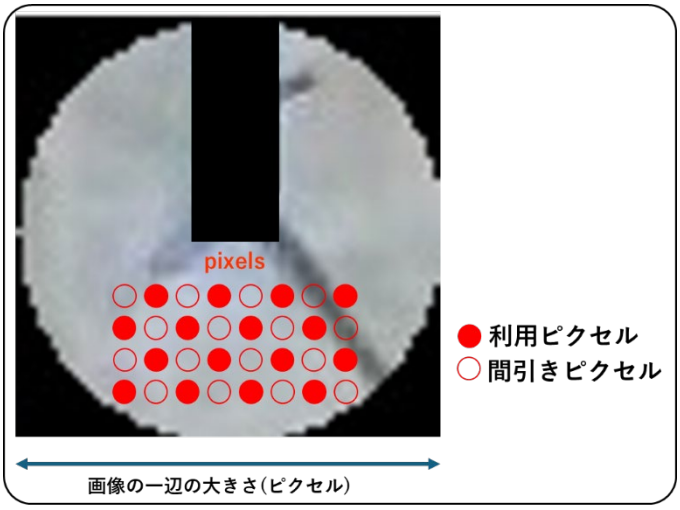

図 13 剣先周辺画像の間引きとサイズのイメージ

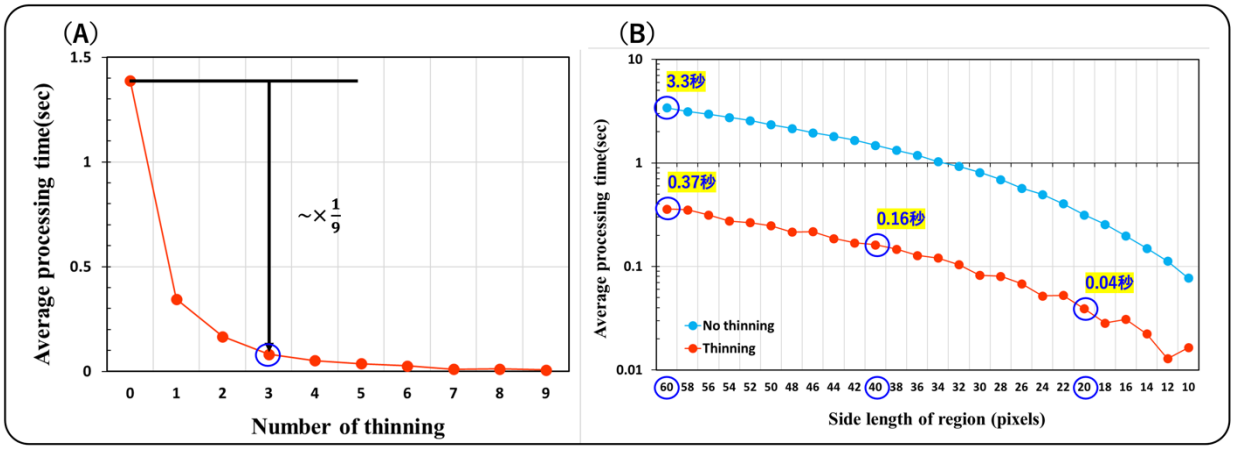

図 14 平均判定処理時間の剣先周辺画像のサイズと間引き間隔への依存性

(A) は、平均判定処理時間の間引き間隔の依存性
(B) は、平均判定処理時間の剣先周辺画像のサイズ依存性

3．2　縞模様の検出を用いた有効突き判定の検討

1 ）判定アルゴリズムについて

図 15 を用いて、縞模様の検出を用いた有効突きの判定方法について説明する。

図 15(1)は剣先でメタルジャケットを突いた際の剣先周辺画像からグレースケール化処理を行った画像である。この画像から、メタルジャケットには、うっすらと縞模様が存在することが確認できる。このメタルジャケット特有の縞模様の検出を画像処理を用いて行い、有効突きの判定を行う方法を提案する。

図 16(2)に、メタルジャケット画像のヒストグラム平坦化処理を行った結果を示す。ヒストグラム平坦化処理を行うことで、画像のコントラストが増大され、メタルジャケットの縞模様がはっきりと確認できるようになる。図 17(3)は、ヒストグラム平坦化処理を行った画像に対して、2 次元のフーリエ変換処理を行った結果である。2 次元フーリエ変換の結果から、メタルジャケットの縞模様に起因した、右斜めの等間

隔の明るい点を確認することができる。このパターンの検出によって、縞模様の検出を行った。図 15(3)の赤色矩形のようにパターンが現れる箇所を囲う範囲で画像のトリミングを行った。その後、画像中の x 軸方向に 5 ピクセルずつ積算を行いながら、y 軸方向の信号を取得し、図 15(4)に示すように縦方向の信号強度スペクトルのグラフを作図した。作図したグラフから、Python の関数である find_peaks を用いて、ピーク検出を行った。このように、点のパターンに対応したピークが現れることが分かる。

ヒストグラム平坦化の方法については、文献[７]、2 次元のフーリエ変換処理方法については、文献[８]を参考にした。また、ピーク検出の方法については、文献[９]を参考にした。

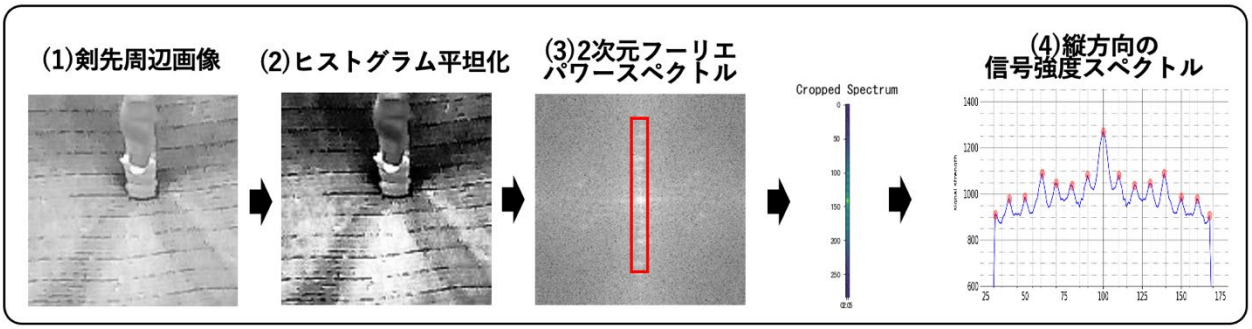

図 15 メタルジャケットの縞模様の検出の流れ

2 ）画像角度のスキャンによる最大ピーク数及び縞模様の同定条件について

ブレードが斜めに傾いた状況で剣先を撮影した場合、図 16(1)のように、斜め方向に縞模様のパターンが現れる。この場合は、縞模様に対応するピークが殆ど検出できなくなる。このような斜め方向のパターンにも対応するために、画像を-45 度から 45 度まで 1 度ずつ回転させて、トリミング領域に縞模様のパターンが平行に重なる角度を検索する。図 16(1)の画像を出発点として、画像角度を 28 度回転させたときに、図 16(2)に示すように、トリミング領域が縞模様のパターンと重なり、縞模様に対応する多くのピークが検出されることが分かる。この場合に、最大ピーク数 13 が得られることがわかる。

図 16(3)は、検出のピーク数の角度依存性を表している。この図から、角度に依存してピーク数が変化しており、トリミング領域がパターン位置に重なる 28 度付近でピーク数が最大値で 13 であることが確認できる。これは、他のデータでも同様にパターン検出が可能であった。

ここでは、このグラフの特徴を利用して、ピーク数の最大値が一定の値を超えたら、縞模様が確認できたと判定する方式を提案した。今回、縞模様のパターンの同定条件を、最大ピーク数が 9 以上とした。このしきい値の統計的な意味づけは今後の課題である。

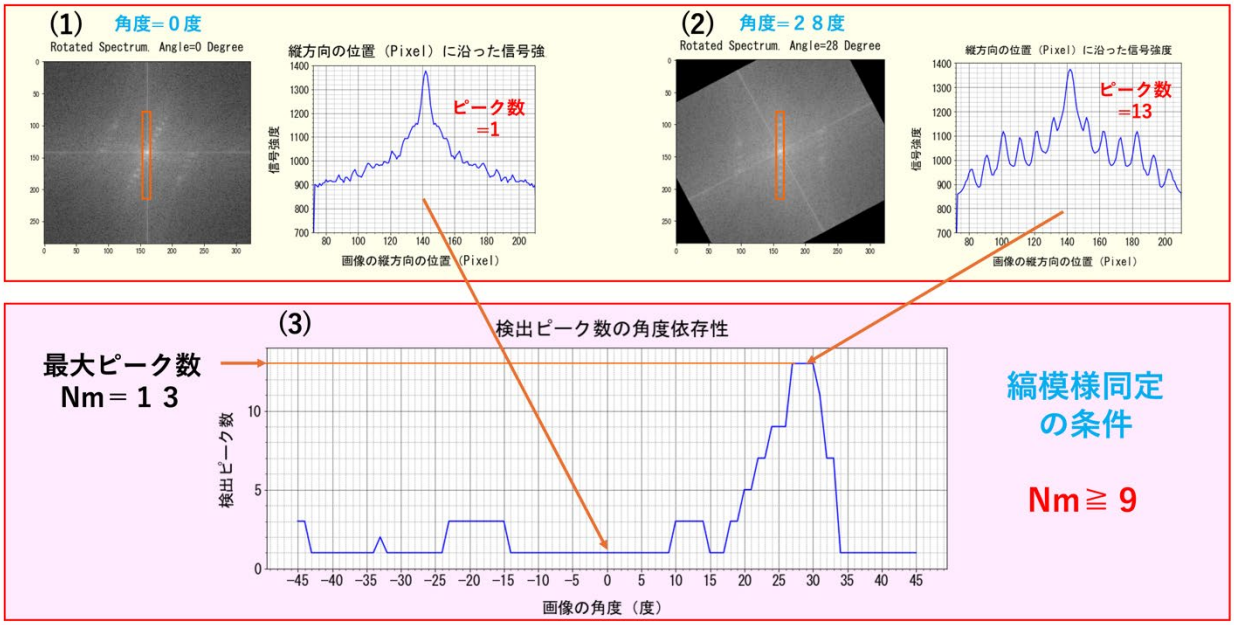

図 16 画像角度のスキャンによる検出ピーク数の角度依存性

3 ）ピーク検出間の距離を用いた判定方法について

図 17 を用いて、ユニフォームとメタルジャケットとの識別を行うための判定条件について説明する。図 17(A)はメタルジャケットを剣先で突いた場合、図 17(B)はユニフォームを剣先で突いた場合を示している。メタルジャケットとユニフォームそれぞれを剣先で突いた画像から上記の方法でピーク検出を行った結果、メタルジャケットとユニフォームを突いた画像のどちらも多数のピークを検出した。これは、スペクトル画像上のむらに起因すると考えられた。

そこで、メタルジャケットとユニフォームを識別するために、ピーク間の間隔に注目した。メタルジャケットの場合のピークは、等間隔の縞に起因しているので、ほぼ等間隔で、９ピクセル以上となっている。一方、ユニフォームの場合は、むらに起因しているので、間隔がばらついており、このように、５ピクセルと小さい場合がある。この傾向は、他のデータの場合でも同様であったため、有効突きの判定条件として、ピーク間間隔の閾値を７以上とした。多くのデータを用いた統計的な分析によるしきい値の検討は、今後の課題である。

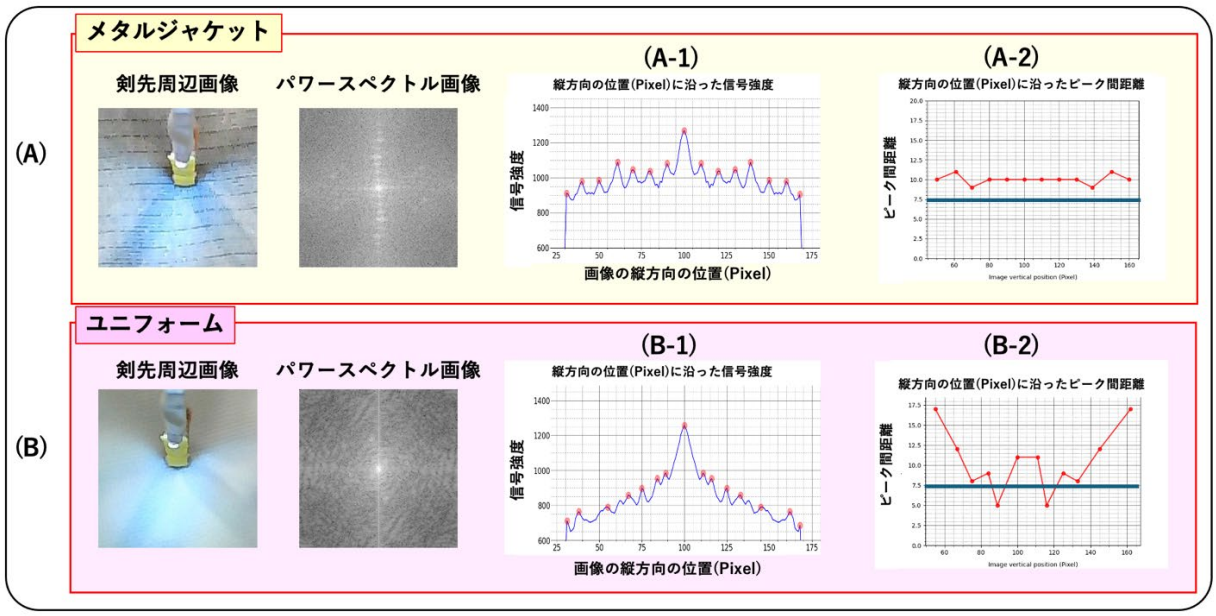

図 17 メタルジャケットとユニフォームのピーク数及びピーク間距離の比較

4. 判定が正確に行えない場合の対応について

図 18 を用いて、判定を失敗するケースとその対応について説明する。図 18 のように、相手の鍔や腕などで剣先を画像内に収めることができない場合、有効突き判定を正確に行えない。判定が正確かどうかを確認できるよう、剣先周辺画像や画像処理結果を、突き判定後、外部のディスプレイ上に表示する機能を構築した。外部ディスプレイと中央制御部は Wi-Fi 環境下で無線接続されるように設計した。

図 18 剣先画像の撮影失敗例

5. 結論と今後の課題

5. 1　結論

フェンシング競技フルーレ種目での、有効突き判定には電気審判機が用いられているが、１．設置に手間がかかる。２．判定が見難く分かりにくい。という２つの問題が存在している。この問題の解決策として、今回、電気審判機を使用せずに剣先を発光させることによって突きを可視化させ、画像処理を用いた自動判定を行える有効突き判定装置の検討を行った。

有効突き判定方法に関して、以下の結果が得られた。

１）有効突き判定装置の改良

前回、文献[３]では、剣先の上部に設置された１つの青色 LED の発光から、突きの可視化を行い、また、カメラで青色 LED の発光を捉えることで、剣先検出を行っていたが、正確な剣先座標が求められない場合などの課題が存在した。今回は、２つの異なる LED を剣先に設置し、剣先上部に設置された赤色 LED は、突きの可視化のために用い、剣先の下部に設置された白色 LED は、剣先の下部に貼られた黄色テープを常時照射し、その反射光から、剣先座標を求める。この改良によって、LED や環境光の影響を受けにくくなったため、従来のものより剣先座標の検出の精度が向上した。また、LED の常時設置によって、剣が突いた瞬間に剣先画像が取得できるようになり、遅れによる遅延が生じにくくなった。

２）HSV 色空間を用いた有効突き判定

メタルジャケットやユニフォームを使用せずに、フェンシング競技フルーレ種目を行う場合の判定を行う方法として、HSV 色空間を用いた有効突き判定が可能になる見込みが得られた。

３）有効突き判定の高速化の検討

剣先周辺領域において、画像処理を行う画素を少なくする方法によって高速化を実現する検討を行った結果、Raspberry Pi Zero を用いた場合に、0.087 秒という試合中での判定として、十分短い判定時間を達成することができた。

４）縞模様の検出用いた有効突き判定方法の検討

メタルジャケットやユニフォームを使用してフェンシング競技フルーレ種目を行う場合の判定を行う方法として、メタルジャケットの剣先周辺領域の画像が有する縞模様の検出を、2 次元フーリエ変換を用いた方法による有効突き判定方式を提案し、突き判定が有効である見込みが得られた。

5. 2　今後の課題

１）メタルジャケットとユニフォームとの境界付近での突きの判定の精度向上

HSV 色空間を用いた判定方法と縞模様検出を用いた判定方法の両者において、メタルジャケットとユニフォームとの境界付近での突きの判定に関して課題がある。

①　HSV 色空間を用いた判定方法

剣先周辺画像全体のH値ヒストグラムによって、判定処理を行っているため、境界付近での判定では誤判定となる可能性がある。より精度高く判定を行うために、メタルジャケットとワイン色ジャージの境界曲線を求められる機械学習などのアルゴリズムの検討をする必要がある。

② 縞模様検出を用いた判定方法

剣先周辺領域全体で判定を行っているため、境界付近での現在の判定は誤判定となる可能性が高い。境界付近での判定を正確に行うためには、より狭い領域の特徴量を使った判定方式を検討する必要がある。

2）縞模様検出を用いた判定方法における、高速な突きの判定精度向上

縞模様検出を用いた判定方法において、高速な突きの場合に縞模様の映像にぶれが生じて、突きの判定に誤判定が生じる可能性が高くなる。高速な突きに対応するためには、現在の20fps程度の撮影速度のカメラから、60fps程度の撮影速度を有するカメラを導入する必要がある。

6. 文献

1．実用新案出願広報No. 3151105

2．特許 第7073595号（2022年5月13日）

3．Seira Aguni, Tetsuo Nishikawa, Kaito Fujita, Ren Nakanishi, Yumi Asahi. Development of a light-emitting sword tip accompanying thrusts and a device for judging valid thrusts by light spectrum detection without an electric judge in the foil event of fencing competitions. Human Interface and the Management of Information, **23**, (2023), 457-470.

4．PyStyle(2020年8月19日). “OpenCV - マスク画像を利用した画像処理について”. https://pystyle.info/opencv-mask-image/ (2024年3月31日)

5．PyStyle(2020年8月30日). “OpenCV - モルフォロジー演算(膨張、収縮、オープニング、クロージング)”. https://pystyle.info/opencv-morpology-operation/ (2024年3月31日)

6．PyStyle(2020年8月13日). “OpenCV - findContoursで画像から輪郭を抽出する方法)”. https://pystyle.info/opencv-find-contours/ (2024年3月31日)

7．鳥取大学. “ヒストグラム その2: ヒストグラム平坦化”. http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_histograms/py_histogram_equalization/py_histogram_equalization.html (2024年3月31日)

8．鳥取大学. “フーリエ変換”. http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_transforms/py_fourier_transform/py_fourier_transform.html (2024年3月31日)

9．Qiita(2021年10月18日). “PythonでFFTとそのピーク検出をする方法”. https://qiita.com/kuwamaso/items/6a4c8310c0222a72afe0 (2024年3月31日)

（原稿提出：2024 年 3 月 31 日）