クロスプラットフォームアイトラッキングデータ解析ソフトウェア ganca の紹介

Introducing ganca; a cross-platform eye-tracking data analysis solution.

22N3100023C　桑原　志門（応用認知脳科学研究室）
Simon H. Kuwahara / Applied Cognitive Neuroscience Lab.

Key Words : *eye-tracking, C++, Python, SWIG, software development*

1. 背景

(1) アイトラッキングの普及

アイトラッキング，または視線解析は眼球の位置と向きを計測することで視線を知ることができる技術である．視線解析は視覚的注意を測る手段として 50 年以上前から使用されている [1]．近年は HTC Vive Pro Eye（HTC Corporation，台湾新北市），Apple Vision Pro（Apple Inc.，アメリカ合衆国カリフォルニア州）に代表される、アイトラッカーが組み込まれた VR（Vritrual Rality）や AR（Augmented Reality）のハードウェアが出現するようになった。歴史上初めてアイトラッカーがコンシューマー向け製品を通して身近な存在となり，アイトラッキングの利用は今後増加していくと予想される [2]．また，学術的研究におけるアイトラッキングについても研究用アイトラッカーの API を使用したオープンソースの制御ソフトウェアの開発が急速に進み [3][4][5][6]，より緻密な制御を要求する複雑な計測を伴う研究を行う土俵が整備された．

(2) 解析用ソフトウェアの現状

アイトラッカーを用いて計測を行うためのソフトウェア及びハードウェア両面のインフラストラクチャーは整備されてきたのに対し，取得したデータを解析するソフトウェアの開発はまだ途上である．

VR/AR の開発では Unity（Unity Software Inc.，アメリカ合衆国カリフォルニア州）や Unreal Engine（Epic Games Inc.，アメリカ合衆国ノースカロライナ州）といったゲームエンジンが用いられるが，2024 年 1 月現在，ゲームエンジンに対応したアイトラッキングデータ解析用のソフトウェアは確認されていない．これらのゲームエンジンでは C++で書かれた解析用ライブラリが必要である．

学術的研究におけるアイトラッキングも解析用ソフトウェアの課題を抱えている．アイトラッカーの製造会社は自社のアイトラッカー向けにソフトウェアを開発しているが，これらのプロプライエタリソフトウェアは拡張性やユーザーによる開発プラットフォームとしての利用を考慮した設計になっていない．よって，研究者によるオープンソースの解析ソフトウェアが開発されてきた [7][8][9]が，多くの解析ソフトウェアが以下のような問題を抱えている．

1. 特定のアイトラッカーのメーカーや機種にしか対応していない．
2. R，Python，MATLABなど特定の言語で書かれており，他のソフトウェアとの互換性がない．

アイトラッキングの研究コミュニティ内の限られた開発リソースが異なるベンダー，開発言語で分散してしまい，大きな問題となっている．そもそも，研究者の業務は研究であり，ソフトウェア開発ではないため，実在するプログラムにも次のような問題を抱えている場合が多い．

1. 実装されている解析手法が限定的である．
2. 公式ドキュメントが存在しない．
3. ソフトウェアの保守が放棄されている．

コミュニティにより継続的に保守されている成功例はいくつかあるものの，それらに実装されている解析手法の種類は限定的である．高度な解析手法を用いる研究プロジェクトでは既存のソフトウェアでは対応できなくなり，研究者が自らプロジェクトに特化した解析プログラムを書く事態となっている．毎回新しいプログラムを書くのは労力が大きく，論文が成果物として評価される研究者は高品質なプログラムを書くインセンティブが無い．故に，非機能要件が考慮されない品質の悪いコーディングに繋がる[10]．これは研究の生産性低下のみならず，結果の信頼性にも影響する．

また，Tobii Pro Spectrum (Tobii AB，スウェーデン王国ストックホルム) に代表される、アイトラッカーのハイエンド機種は1000Hz以上のサンプリングレートを誇り，データサイズが増加している．解析ソフトウェアにもパフォーマンスが要求されるが，最適化にはデータ形式やアルゴリズムなどの高度な知識が要求され，高度な人材と開発工数が必要である．

(3) バックエンドの共通化

この状況を解決するためには開発リソースの削減と網羅的な解析手法の実装を両立する必要がある．そこで，様々な解析ソフトウェアが共通のバックエンドと使えるようなAPIを開発することで冗長な開発工程を削減できる．このようなAPIをC++で開発することができ

れば，都合よくVR/AR開発のニーズと研究用途のニーズを両方同時に満たすソフトウェアを実現することができる．

そこでこのコンセプトをオープンソースのアイトラッキング用データ解析ソフトウェア「gaze analysis and compute application (ganca)」として開発を開始した．これはアルファ版リリースに関する報告である．

2. アルファ版の概要

(1) 概要

アルファ版はテストと評価のみを目的とした ganca 初の開発用リリースで現在 GitHub で公開されている (https://github.com/qwasium/ganca/releases)．このアルファ版は SWIG[11] を用いて C++をラップした Linux x64 用 Python ライブラリのみが含まれている．C++ライブラリはソースからビルドできるが， リリースには含まれていない．このリリースは Ubuntu22.04，C++ 17，GCC 11.2.0，Make 4.3-4.1build1，CMake 3.22.1，SWIG 4.0.2 でビルドされており，Python 3.10.12 による単体テストを通過している．アルファ版の開発目標は次の通りである．

1. SWIGを用いたラッピングを採用し、ビルド言語の増加を念頭に置いたビルドシステムの設計をする．
2. 正式リリースを見据えてモジュール数の増加にも対応できるソースツリーの設計をする．
3. 検証のために既存の解析手法のアルゴリズムと同一な構造を保ちつつ，最小限の実装を行う．
4. リリースまでの開発を通して、開発方針の評価，再検討を行う．

また，アルファ版は評価用の最小規模のリリースという性質から，以下のような制約がある．

1. Linux x64のみに対応している
2. Python3のみに対応している．
3. $\mathbb{R}^2$ 上に定義される刺激空間のみに対応している．
4. 並列処理には対応していない．
5. 極めて限定的な機能のみを実装している．

(2) モジュール

アルファ版では4つのモジュール，“aoihit.pyAOIHit”,, “fixationfilter.pyFixationFilter”， “heatmap.pyHeatmap”， “helper.pyHelper”を実装している．詳細はリリースに添付されている公式ドキュメントに記載されている．以降示される時間計算量と空間計算量はxとyをそれぞれディスプレイの横と縦の解像度，nを時系列データの大きさとしたランダウのbig-O表記を用いる．

モジュール“aoihit.pyAOIHit”は関数“cntRasterAOI()”を有している．このモジュールはArea Of Interest (AOI) を用いた分析を目的としている．AOIとは刺激空間の中で解析対象として興味のある領域として設定される．視線がAOIに対してどのような挙動を取るかを観察することで視覚的注意に関する情報を得る手法である[12]．この関数ではディスプレイの解像度と同じ大きさの二次元配列について、AOIの領域には1，それ以外の領域を0としたAOIマスクを用いてAOIを指定する．与えられた空間座標の時系列データの要素を一つずつAOIマスクに対して照会し，AOI内であるかどうかを返す．時間計算量は$O(xy+n)$，空間計算量は$O(xy+n)$ である．

モジュール“fixationfilter.pyFixationFilter” はクラス“FixationFilter”を有している．このクラスは眼球運動の分類を扱っている．人間の眼球は常に動いており，性質の異なる複数の眼球運動に分類されている．視線データを異なる眼球運動の変遷として理解することで視覚的注意の指標として解釈することができる[13]．数多くの眼球運動分類手法は視線の時系列データに対してループする共通構造となっているのでオブジェクト指向プログラミング(OOP)が採用されている．アルファ版ではメソッド“IVTfilter()”によってIdentification by Velocity Threshold(I-VT)フィルター[14]の最も単純な形式の実装がなされている．これはロジックの構造に対する評価のみを目的とした簡略化である．I-VTは時系列の隣接する2計測点の視線ベクトルから角速度を計算し，それが与えられた基準値を下回れば視線が一点付近にとどまる運動であるFixaion，上回れば注視点を移動する際の高速な運動であるSaccadeとして分類される．このメソッドの時間計算量は$O(n)$，空間計算量は$O(n)$ である．

モジュール“heatmap.pyHeatmap”は関数“heatmapGauss()”を有している．このモジュールはヒートマップに関連した解析を目的としている．ヒートマップは刺激空間に対して視覚的注意の確率密度を視覚的に表現する手法である[15]．混合ガウス分布がヒートマップの手法として最も一般的であり[16]，この関数でも採用している．フィクセーションデータの時系列を入力とする場合は各フィクセーションの座標を位置パラメータ，継続時間に調整用の定数パラメータを乗した値をスケールパラメータとした正規分布の確率密度を配列の各要素について加算していき，最後にすべての要素の値が[0, 1]に収まるように正則化する．低レイヤーのAPIという特性上，返り値は二次元配列である．PythonのMatplotlib[17]やRのggplot2[18]を用いると簡単にレンダリングできる．時間計算量は$O(xyn)$，空間計算量は$O(xy+n)$ である．

モジュール“helper.pyHelper”はヘルパー関数のモジュールでパフォーマンス評価の対象からは除外している．

3. パフォーマンス評価

パフォーマンス評価をするためにベンチマークとしてgancaのソースコードと全く同じロジックをPythonで書き，実行時間を直接比較した．

テストデータはTobii Pro Spectrumを用いてシステム検証

用に計測されたデータを用いた．生データおよびI2MC[19]で眼球運動を分類したデータによって構成されており，テストデータはテスト関数とともにGitHubレポジトリにて公開されている．このデータはパフォーマンス評価用のデータとしてのみ存在する．

テスト環境はPython 3.10.12，ipykernel 6.29.0，VSCode 1.85.2，Ubuntu 22.04，Intel Core i7-6700K，DDR4 32GB．テストはJupyter Notebookの%%timeitマジックセルコマンドを用いてVSCode上で実行された．セルを10回実行する処理を1試行として100試行が行われ，1試行あたりの平均実行時間と標準偏差が取得される．10回の実行をまとめて1試行とする理由は，ランタイムにオーバーヘッドが存在するので一回の実行では本来の実行時間は計測できないからである．

4. 結果

AOI関数は平均で4.74倍，FixationFilterクラスは平均で10.3倍，heatmap関数は平均で42.5倍gancaの方が実行時間が短かった(表1).

5. 考察

(1) 結果の解釈

結果から，Python及びJupyter Notebookのオーバーヘッドのある環境で実行する場合においてもC++のラッパーコードはパフォーマンスを大きく改善することが確認できた．結果を考察するうえで，パラメータに対する計算量と並列化による高速化の2点は今後の開発に大きな影響を及ぼす重要な要素である。

生の視線データにしろ，フィクセーションデータにしろ，入力する時系列データの大きさ n はソフトウェアの観点ではプロジェクトの生産性に最も影響のあるパラメータ特性である．アイトラッキングで用いられる多くのアルゴリズムは時系列データの要素をループする操作である．今回のアルファ版のように全てシングルスレッド処理だった場合，C++ラッパー関数とPythonの実行時間比は一回のループの中の命令の数に依存している．ループ内の処理が単純な場合はSingle Instruction Multiple Data (SIMD)を用いた並列処理を用いてnに対して効率的に並列化できる．

刺激空間の大きさ x 及び y はディスプレイを用いている場合は離散空間かつ、視線解析で用いられるディスプレイには現実的な上限があるので生産性に及ぼす影響は限定的である．しかし，VR/AR開発では刺激空間は$\Re^3$上で連続的に定義されるため，時間的及び空間的計算時間両方に非常に大きな影響を与えることとなる．

モジュール“aoihit.pyAOIHit”は x と y の二重ループと n のループで構成されており，小さい実行時間差からも示唆されるようにそれぞれのループ内の処理は少ない．全てのループに対して並列化は容易だが，他のアルゴリズムと比較すると最適化の余地は少なく，開発優先順位としては低い．

モジュール“fixationfilter.pyFixationFilter”はnのループの中で処理が多いため，実行時間比が大きくなっている．アルファ版の実装では極めて簡略化されているが，実用を想定した眼球運動判別アルゴリズムではループの内部の処理はループ内の処理が格段に多く[20]，実質的な実行時間比は今回のテスト結果と比べて著しく増大することは容易に推測できる．その反面，複雑な構造故に効率性を維持した並列化は難しい．すなわち，C++によるラッピングは非常に効果的であるが，それ以上のパフォーマンス最適化には高度な知識を要求する．実際の解析作業における眼球運動判別の重要性と既存の

表-1 実行時間計測の結果(実行回数10回の数値)

モジュール	テストデータ	実装	平均(s)	標準偏差(s)	対ganca時間比
aoihit.pyAOIHit	data 1	ganca	3.73 * 10e-2	4.59 * 10e-3	4.63
		Python	1.69 * 10e-1	2.46 * 10e-3	
	data 2	ganca	3.65 * 10e-2	2.75 * 10e-3	4.85
		Python	1.81 * 10e-1	1.19 * 10e-2	
fixationfilter. pyFixationFilter	data 1 left eye	ganca	9.21 * 10e-4	4.76 * 10e-5	1.27 * 10
		Python	1.17 * 10e-2	1.77 * 10e-3	
	data 1 right eye	ganca	9.44 * 10e-4	5.75 * 10e-5	9.52
		Python	8.99 * 10e-3	7.58 * 10e-4	
	data 2 left eye	ganca	9.24 * 10e-4	3.72 * 10e-5	9.72
		Python	8.99 * 10e-3	9.00 * 10e-4	
	data 2 right eye	ganca	9.63 * 10e-4	1.66 * 10e-4	9.32
		Python	8.98 * 10e-3	6.69 * 10e-4	
heatmap. pyHeatmap	data 1	ganca	4.81 * 10e-1	1.25 * 10e-2	4.39 * 10
		Python	1.98 * 10e1	6.37 * 10e-1	
	data 2	ganca	4.63 * 10e-1	1.45 * 10e-2	4.10 * 10
		Python	1.90 * 10e1	3.16 * 10e-2	

ソルトウェアが要する計算時間を考慮すると最も開発優先順位が高いモジュールである．

モジュール“heatmap.pyHeatmap”は3重ループの構造になっている．実行時間比も大きいものの，テストデータを用いても実行時間を要している．しかし、ヒートマップの生成は実質的にはレンダリングと同等の処理を行っているため[21]，既存のグラフィックス処理のフレームワークとGPUを用いることで比較的低い開発負荷で大幅な高速化が期待できる[22]．

⑵　今後の開発方針

開発負荷の削減が現在最大の問題である．アルファ版ではC++のラッピングのフレームワークとしてSWIGを選択した．SWIGでビルドできる言語は多いが，その分開発難度が高い．これは，ビルド対象言語を増やすと指数関数的に開発負荷が増加していくことと，C++の言語仕様が極めて複雑であること[23]に由来する．そこで，アルファ版のフィードバックを生かし，gancaの開発方針を転換する．今後はラッパーフレームワークを依存関係を限定することで開発をコストの削減に適したPyBind11に変更し，C++11とPythonのみに開発リソースを専念してベータ版のリリースを目指す．他の言語に関してはオープンソースコミュニティに放任することとする．

参考文献

1) Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. Psychological Bulletin, 124(3), 372.
2) Clay, V., König, P., & Koenig, S. (2019). Eye tracking in virtual reality. Journal of Eye Movement Research, 12(1). https://doi.org/10.16910/jemr.12.1.3
3) Dalmaijer, E. S., Mathôt, S., & Van der Stigchel, S. (2014). PyGaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. Behavior Research Methods, 46(4), 913–921. https://doi.org/10.3758/s13428-013-0422-2
4) Niehorster, D. C., Hessels, R. S., & Benjamins, J. S. (2020). GlassesViewer: Open-source software for viewing and analyzing data from the Tobii Pro Glasses 2 eye tracker. Behavior Research Methods, 52, 1244–1253. https://doi.org/10.3758/s13428-019-01314-1
5) Niehorster, D. C., & Nyström, M. (2020). SMITE: A toolbox for creating Psychophysics Toolbox and PsychoPy experiments with SMI eye trackers. Behavior Research Methods, 52(1), 295–304. https://doi.org/10.3758/s13428-019-01226-0
6) Niehorster, D. C., Andersson, R., & Nyström, M. (2020). Titta: A toolbox for creating PsychToolbox and Psychopy experiments with Tobii eye trackers. Behavior Research Methods, 52(5), 1970–1979. https://doi.org/10.3758/s13428-020-01358-8
7) Voßkühler, A. (2009). OGAMA description (for Version 2.5). Berlin, Germany: Freie Universität Berlin, Fachbereich Physik.
8) Ghose, U., Srinivasan, A. A., Boyce, W. P., Xu, H., & Chng, E. S. (2020). PyTrack: An end-to-end analysis toolkit for eye tracking. Behavior Research Methods, 52(6), 2588–2603. https://doi.org/10.3758/s13428-020-01392-6
9) Dink, J., & Ferguson, B. (2016). Performing a window analysis using eyetrackingR. Performing a Window Analysis Using Eyetrackingr.
10) Austin, R. D. (2001). The effects of time pressure on quality in software development: An agency model. Information Systems Research, 12(2), 195–207. https://doi.org/10.1287/isre.12.2.195.9699
11) Beazley, D. M. (2003). Automated scientific software scripting with SWIG. Future Generation Computer Systems, 19(5), 599–609. https://doi.org/10.1016/S0167-739X(02)00171-1
12) Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., & Van de Weijer, J. (2011). Eye tracking: A comprehensive guide to methods and measures. OUP Oxford.
13) Rayner, K. (2009). The 35th Sir Frederick Bartlett Lecture: Eye movements and attention in reading, scene perception, and visual search. Quarterly Journal of Experimental Psychology, 62(8), 1457–1506. https://doi.org/10.1080/17470210902816461
14) Olsen, A. (2012). The Tobii I-VT fixation filter. Tobii Technology, 21.
15) Pomplun, M., Ritter, H., & Velichkovsky, B. (1996). Disambiguating complex visual information: Towards communication of personal views of a scene. Perception, 25(8), 931–948. https://doi.org/10.1068/p250931
16) Wooding, D. S. (2002). Fixation maps: Quantifying eye-movement traces. 31–36.
17) Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(03), 90–95. https://doi.org/10.1109/MCSE.2007.55
18) Wickham, H., & Wickham, H. (2016). Getting Started with ggplot2. Ggplot2: Elegant Graphics for Data Analysis, 11–31.
19) Hessels, R. S., Niehorster, D. C., Kemner, C., & Hooge, I. T. (2017). Noise-robust fixation detection in eye movement data: Identification by two-means clustering (I2MC). Behavior Research Methods, 49(5), 1802–1823. https://doi.org/10.3758/s13428-016-0822-1
20) Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. 71–78. https://doi.org/10.1145/355017.355028
21) Shum, H., & Kang, S. B. (2000). Review of image-based rendering techniques. 4067, 2–13.
22) Neider, J., Davis, T., & Woo, M. (1993). OpenGL programming guide (Vol. 478). Addison-Wesley Reading, MA.
23) Balogun, M. (2022). Comparative Analysis of Complexity of C++ and Python Programming Languages. Asian J. Soc. Sci. Manag. Technol, 4, 1–12.