テニスの試合分析のための記録支援アプリの提案

髙岡 弦世 [1]　新谷虎松 [2*]

概要: 本研究では、ソフトテニスの試合における記録作業を効率化し、分析と指導に役立てることを目的として、Python を用いた記録支援アプリを開発した。従来、試合記録は紙媒体で行われてきたが、リアルタイムでの記録は労力が大きく、データの蓄積や活用が難しいという課題があった。本システムは、試合のスコアやエース、ミスなどを GUI 操作で簡便に記録できる機能に加え、保存・修正機能、コート上へのマークアップ機能を備えている。また、ChatGPT を用いた推論機能を実装し、記録データからプレイヤーの長所・短所を自動分析し、練習方法の提案を行う点に新規性がある。評価実験では、テニス経験者・未経験者の双方に使用してもらい、操作性、応答の質、分析の有用性について検証した。その結果、UI や記録のしやすさについて高評価を得た一方で、推論結果の具体性やユーザビリティに改善の余地があることも明らかになった。本研究は、競技者の成長支援や戦術立案をサポートする新たなアプローチを提示し、スポーツ記録システムの発展に寄与するものである。

キーワード： ソフトテニス, 記録支援システム, ChatGPT, 試合分析・練習提案

Design and Implementation of a Recording Support Application for Tennis Match Analysis

Gensei Takaoka[1]　Toramatsu Shintani[2*]

Abstract: This study proposes and develops a recording support application for soft tennis match analysis, aiming to enhance the efficiency of recording and facilitate post-match feedback. Traditionally, match records have been taken manually on paper, which imposes a heavy burden during games and makes systematic data accumulation difficult. The proposed system, implemented in Python, provides a GUI-based interface that allows users to easily record scores, aces, and errors in real time, as well as to revise and store match data. Furthermore, it introduces novel features such as on-court markup and an inference function powered by ChatGPT, which analyzes recorded data to identify players' strengths and weaknesses and suggests tailored training methods. Evaluation experiments with both experienced and inexperienced players demonstrated the system's usability and the usefulness of its analysis features, while also indicating areas for improvement in terms of the specificity of AI-generated feedback and overall user interface design. This research highlights the potential of integrating generative AI into sports recording systems, contributing to player development and tactical decision-making support.

Keywords: Soft Tennis, Match Recording Support System, ChatGPT, Match Analysis and Training Recommendation

1.はじめに

近年、スポーツにおけるデータ活用は、選手のパフォーマンス向上や戦術の高度化において不可欠な要素となっている。特にテニスやソフトテニスのようなラケットスポーツでは、得点状況、ラリー展開、プレー選択の傾向などを記録・分析することが、試合内容を客観的に振り返るうえで極めて有効である。しかし現状では、ソフトテニスの試合記録は依然として紙媒体に依存している場合が多く、記録者は試合進行に追われながらメモを取らねばならず、記録精度や網羅性が十分に担保されないことが少なくない。また、記録用紙は紛失や劣化のリスクを抱え、長期的なデータ蓄積やチーム内での共有・活用が困難である。これらは競技力向上の観点から大きな制約となっている。

1 岐阜聖徳学園大学（当時）
2 岐阜聖徳学園大学
Gifu Shotoku Gakuen University
*責任著者(Corresponding author): tora@gifu.shotoku.ac.jp

従来、スポーツデータの収集や分析を支援するシステムはいくつか提案されてきたが、多くは野球やサッカーなどメジャースポーツを対象としたものであり、ソフトテニスに特化した記録支援システムは十分に整備されていない。特に、試合中に簡便かつ直感的に操作でき、かつ試合後にデータを分析可能なシステムは未だ限定的である。この点において、本研究は「ソフトテニスに焦点を当てた記録支援」と「記録データを活用した学習支援」という二つの側面を統合的に実現する点に独自性を持つ。

本研究では、プログラミング言語 Python を基盤とし、GUI ライブラリ Tkinter を用いて記録係がスコアやプレー内容をリアルタイムで入力できるシステムを構築する。GUI による操作は、従来の紙メモと比較して迅速かつ正確な記録を可能にし、さらにデータの保存・修正・検索を容易にする。また、記録内容を蓄積することで、長期的なパフォーマンス分析や戦術改善に資するだけでなく、教育現場における教材や研究データとしても活用可能である。

加えて、本システムの最大の特徴は、生成 AI である ChatGPT を組み込み、試合データから自動的に選手の長所・短所を抽出し、改善に向けた練習メニューを提示できる点にある。従来の記録支援システムは「データを残すこと」に主眼を置いていたが、本研究は「データを活用し、学習支援につなげる」という新たな役割を付与することで、記録から実践的な指導・練習支援までを一貫して行えるシステムへと発展させている。これは単なる利便性の向上にとどまらず、データ駆動型のスポーツ教育・指導の可能性を拡張するものである。

本論文ではまず、関連研究を整理し、既存のスポーツ記録支援システムや生成 AI 活用の動向を概観する。その上で、本研究におけるシステム設計と実装手法を詳細に説明し、実際に利用者による評価実験を通じて有用性と課題を検証する。最後に、得られた知見を踏まえ、スポーツデータ活用と生成 AI 技術の融合がもたらす新しい方向性について考察を行う。。

2. 関連研究

試合記録システムを構築する上で重要なのは、従来の紙媒体による記録方式に代わり、正確性と柔軟性を兼ね備えたデジタル記録の仕組みを実現することである。近年では様々な競技において記録システムが実装されており、その有用性が報告されている。 坂東ら[1]は、Firebase を活用したクラウドベースの記録システムを開発している。このシステムは、リアルタイムで試合の進行状況を追跡し、その結果を関係者間で広く共有できる点を特徴とする。具体的には、Firebase の認証機能・ホスティングサービス・リアルタイムデータベースを組み合わせることで、スケーラビリティとデータセキュリティを確保しつつ、インストール不要で誰でも容易にアクセス可能な仕組みを実現した。これにより、試合運営者・コーチ・選手などがリアルタイムで試合状況を把握でき、試合後の分析や戦略立案に有効活用できるとしている。 さらに、全国規模の大会では、各コートに配置された主任がタブレット端末を用いて本システムを利用することで、競技進行の効率化や運営の円滑化が期待されている。こうした研究は、試合運営の改善に直結するだけでなく、選手や指導者が競技記録を容易に活用できる環境を整える基盤となる。したがって、ソフトテニスにおいても、より直感的かつ効率的に試合を記録できるシステムの開発が強く求められている。

スポーツ・体育の分野において、競技データの収集と分析は選手の技術向上や戦術立案に直結する重要な要素である。近年では、大規模言語モデルである ChatGPT を推論エンジンとして活用し、既存データの分析や指導支援に役立てる取り組みが注目されている。例えば、ラリーに関する指導方法に教材を組み込み、ラリー技術の向上を目指す研究[2]や、ダブルスの試合においてサービスやネットプレーで得点を重ねることが勝利に直結することを示した研究[3]がある。これらの知見を ChatGPT に組み込み、プロンプト設計によって適切に指示することで、多角的な観点からの推論を実現できる。また、アウトやネットといったコントロール面でのミスに関しても、単なる技術的要因だけでなく、ラケットのガットテンションがコントロール低下に影響する可能性が指摘されている[4]。こうした知見を入力情報として与えることで、ChatGPT の推論はより信頼性と正確性を備えた結果を導き出すことができる。さらに、矢野ら[5]は ChatAPI を用いた自動採点付き学習アプリを開発し、ChatGPT に対するプロンプトを精緻化することで、回答の正誤判定を一定の精度で実現できることを示している。このように、プロンプト設計の工夫は ChatGPT の推論精度を大きく左右する要素であり、スポーツ分野におけるデータ分析への応用可能性を拡大する鍵となる。以上のような先行研究から、ChatGPT はスポーツにおけるデータ分析と推論の両面で有効に機能し得ることが示唆されている。本研究では、この知見を応用し、ソフトテニスの試合データを ChatGPT によって分析・解釈させ、選手の課題抽出や練習提案に結び付ける点に独自性がある。

システムを構築する上で重要な要件の一つは、初めて利用するユーザーが直感的に「どのようなシステムであるか」を理解できることである。特に教育・スポーツ支援システムにおいては、ボタンの配置や配色といったデザイン要素が、利用者の操作性や学習効率に直接的な影響を与える。機能的に優れたシステムであっても、インターフェースが煩雑であったり、配色が視覚的に負担を与えたりする場合、ユーザーにとって「使いにくいシステム」と認識されてしまう。鍋田ら[6]は、黒板に近い深緑色の板書における配色効果について研究を行い、基本的には白色のチョークが視認性に優れているが、強調すべき情報に色の変化を加えることで理解度が向上することを示した。この知見はシステム設計にも応用でき、例えば「データを変更・操作するボタン」を他のボタンと色で区別することで、ユーザーに注意喚起を行い、誤操作を防ぐといった効果が期待できる。また、ユーザビリティを高めるには、開発者視点ではなく利用者視点での設計が欠かせない。ユーザーからのヒアリングを通じて「どの位置にボタンを配置すべきか」「結果を表示するエリアはどのようなレイアウトが適切か」「どのような配色が快適でわかりやすいか」といった要素を取り入れることが、システムの受容性を高めるために重要である[7]。このように、視認性とユーザビリティに関する研究は、単なるインターフェース設計にとどまらず、ユーザーの認知負荷を軽減し、学習や競技記録といった目的達成を支援する重要な基盤技術である。本研究においても、ソフトテニスの試合記録支援システムをより多く

の利用者に受け入れられるものとするために、GUI の配色・ボタン配置・操作動線に配慮した設計を導入する。

本研究の位置付けは、ソフトテニスにおけるプレイヤーの技術力向上を支援するために、試合記録をリアルタイムで収集・活用できる記録支援システムを実現する点にある。従来の紙媒体による記録では、試合の進行速度に対応することが難しく、また記録内容の再利用性や分析の幅が限られていた。本研究では、この課題を克服するために「直感的な操作によるリアルタイム記録」「AI による分析・提案」「データの保存と比較」 を統合したシステムを構築する。

具体的には、GUI を用いた操作により、試合中のプレー内容や得点要素をボタン一つで記録できる仕組みを実装することで、ソフトテニス特有の試合進行の速さにも対応できるようにした。さらに、記録されたデータを ChatGPT に入力し、プレイヤーの長所や短所を自動的に抽出し、それに基づいた指導方法や練習メニューを提示する推論機能を備える。これにより、従来の「記録するだけの仕組み」から「記録を活用して成長を促す仕組み」へと発展させている点に新規性がある。

また、記録データには保存機能を設け、過去のデータと最新のデータを比較できるようにすることで、技術力の変化を可視化し、試合戦略の立案に活用可能とした。これにより、プレイヤー自身やコーチが客観的なデータに基づいた成長度の確認や課題抽出を行える環境を提供する。本研究は、記録支援・AI 分析・データ比較を一体化したソフトテニス特化型システムを提案する点において、従来研究と一線を画すものである。

3. システムの実装

3.1 ソフトテニス試合記録システムの概要

試合記録システムのインタフェースを図 3.1 に示す。本システムでは、まず日付・対戦相手・一言メモなどを入力するテキスト欄を設けている。入力欄にはプレースホルダー（入力の目安を示す薄い文字）を表示しており、利用者がどの情報を記入すべきか直感的に理解できるようにした。保存時には入力された日付や対戦相手名を自動的にファイル名に反映させることで、後から「いつ」「誰との試合」であるかを容易に判別できる仕組みとなっている。

画面下部には試合内容を記録するための操作エリアを配置した。ここでは、レシーブ、ネットプレー、エースショットなどのプレー内容をプルダウンメニューから選択し、続いてトグルボタンによって「ミス」か「エース」かを切り替えることが可能である。その後、対象となるプレイヤーを指定すると、該当する記録が中央のテーブルに自動的に反映される。これにより、プレーの種類・結果・選手を一貫して簡便に入力できるよう設計した。

記録内容の削除や修正、保存は画面右上のボタンから実行できる。また、本システムには記録機能に加えて、ChatGPT を用いた推論機能や「コート表示」機能も実装している。前者は蓄積された記録を基に選手の長所・短所を抽出し、改善に向けた指導法や練習法を提案するものであり、従来の記録支援システムには見られない新規性を有する。後者の「コート表示」機能は、試合の状況を視覚的に再現し、位置関係やプレーの流れを直感的に把握できる補助的な機能である。

以降の節では、これらの機能の詳細および実行手順について順に述べる。

図 3.1 試合記録システムのインタフェース

3.2 主な機能

3.2.1 試合の記録方法

本節では、本システムにおける試合記録の具体的な手順について述べる。例として、プレイヤー1 がレシーブでミスをした場合を考える。まず、画面に設けられたプルダウンメニューから「レシーブミス」を選択する。次に、その右側に配置されたトグルボタンを「ミス」に切り替える。最後に「プレイヤー1 の記録」ボタンを押下することで、プレイヤー1 がレシーブミスを犯したという情報が中央のテーブルに自動的に反映され、記録が完了する。

このように、本システムでは「プレーの種類」「結果（エース／ミス）」「対象プレイヤー」を順に指定するだけで記録が完結するよう設計されているため、試合の進行スピ

ードに対応しつつ、記録者が直感的に操作できることが特徴である。記録が完了した状態を図 3.2 に示す。

図 3.2 プレイヤー 1 のミスが記録されている状態

テニスのスコアリングシステムにおいては、デュース後に 2 ポイントを連取したプレイヤーがゲームを獲得するという特有のルールが存在する。この仕組みを実装するためには、特定条件に基づいたポイント差の判定ロジックが不可欠である。その実装例を図 3.3 に示す。

本システムでは、ゲームカウントが 3–3（または 6–6）となった際に、ポイント数が 7 以上かつ相手との差が 2 ポイント以上である場合、そのプレイヤーをゲーム獲得者として処理する。具体的には、if self.game_count_1_2 == 3 and self.game_count_3_4 == 3: によりデュース条件を判定し、さらに if self.point_3_4 >= 7 and self.point_3_4 - self.point_1_2 >= 2: の条件式によって勝敗を確定させている。

勝敗が確定すると、以下の処理が実行される。まず、self.tree.insert("", tk.END, values=("ゲームカウント", f"{self.game_count_1_2} - {self.game_count_3_4}", "")) によりスコアテーブルへ最新のゲームカウントを挿入する。同時に、self.point_1_2 = 0 および self.point_3_4 = 0 により両プレイヤーのポイントをリセットし、次のゲームがスムーズに開始できるよう設計されている。この一連の処理により、デュース時の展開を正確に反映し、試合進行を途切れなく記録することが可能となっている。

さらに、ゲーム獲得時には現在のゲームカウントが画面上に逐次更新される。図 3.4 にその表示例を示す。if game_count_changed: で状態を確認した後、再び self.tree.insert("", tk.END, values=("ゲームカウント", f"{self.game_count_1_2} - {self.game_count_3_4}", "")) を実行することで、ゲームカウントがリアルタイムに反映される。これにより、記録者や観戦者は常に最新のゲーム状況を把握でき、スコアリングの整合性を維持できる。

本処理により、デュースを含む試合展開を自動的かつ正確に管理できる点が、本システムの大きな特徴である。

```
# 3-3の場合は7ポイント先取でゲームカウントを更新
if self.game_count_1_2 == 3 and self.game_count_3_4 == 3:
    if self.point_1_2 >= 7 and self.point_1_2 - self.point_3_4 >= 2:
        game_count_changed = True
        score = f"{self.point_1_2}-{self.point_3_4}"
        self.tree.insert("", tk.END, values=(miss_1 if player == "プレイヤー1" else "", miss_2 if player == "プレイヤー2" else "", score))
        self.records.append((miss_1, miss_2, score, player))
        self.game_count_1_2 += 1
    elif self.point_3_4 >= 7 and self.point_3_4 - self.point_1_2 >= 2:
        game_count_changed = True
        score = f"{self.point_1_2}-{self.point_3_4}"
        self.tree.insert("", tk.END, values=(miss_1 if player == "プレイヤー1" else "", miss_2 if player == "プレイヤー2" else "", score))
        self.records.append((miss_1, miss_2, score, player))
        self.game_count_3_4 += 1
```

図 3.3 デュースの際のプログラム例

```
if game_count_changed:
    self.tree.insert("", tk.END, values=("ゲームカウント", f"{self.game_count_1_2} - {self.game_count_3_4}", ""))
    self.point_1_2 = 0
    self.point_3_4 = 0
```

図 3.4 ゲームカウントを表示するためのコード

図 3.5 デュースを終えてゲームカウントが表示されている様子

3.2.2 試合の記録修正に関する説明

試合記録の入力においては、記録者の操作ミスや入力の誤りにより誤記録が発生する可能性がある。誤記録を放置したまま分析に用いると、結果の信頼性が大きく損なわれる恐れがある。そのため本システムでは、記録修正を可能とする「一項目削除」機能と「リセット」機能を実装した。対応するコードを図 3.6 に示す。

一項目削除機能は、ユーザーが選択した特定の記録を 1 行単位で削除するものである。試合進行中に誤記録が判明した際、この機能を用いて対象データを削除すると、テーブルから該当項目が消去され、さらに自動的に再計算処理が行われるため、修正前後のデータに不整合が生じないよう調整される（図 3.7 参照）。

リセット機能は、テーブル内のすべての記録を削除し、試合記録を初期化するものである。誤操作防止のため、実行時には確認メッセージボックスが表示され、利用者に最終確認を求める仕様とした（図 3.8 参照）。

これらの処理は delete_selected_item_and_recalculate メソッドにより実現されている。まず、self.tree.selection() を用いてユーザーが選択した項目を取得し、そのインデックスを self.tree.index(selected_item) で特定する。削除処理後、self.records.pop(index) により対象データを記録リストから除去する。その後 recalculate_scores メソッドが呼び出され、ゲームカウントおよびポイントを初期化した上で、残りの記録をループ処理により再計算し、新しいスコアをツリーに再挿入する。この再計算過程では、各記録がどのプレイヤーに属するか、また「エース」が含まれるかを判定し、正確に得点へ反映させている。

これにより、誤って入力された記録を修正しても、試合全体の進行や整合性を損なうことなく、正確かつ一貫性のある試合データを保持できる。例えば、カウント「2–1」でのサービスエースが誤入力であった場合でも、該当項目を選択して一項目削除を実行すれば、データは即座に修正され、以降の記録との整合性が保たれる。これにより、今後の試合展開においても記録のずれや誤分析のリスクを大幅に軽減することが可能となる。

3.2.3 試合の記録保存に関する説明

本システムでは、試合記録を保存する機能を実装している。保存機能により、実際に入力された試合内容を長期的に保持することが可能となり、選手の成長や技術的進歩を継続的に振り返る基盤を提供する。これにより、過去のデータを参照して現在の状況と比較することで、必要に応じたトレーニング方法の調整や戦術の見直しが可能となる。

さらに、記録を電子的に保存することで、従来の紙媒体に依存したメモ記録に比べ、紛失や劣化といったリスクを大幅に低減できる。データは一元的に管理されるため、長期間にわたる安定的な利用が可能となり、チームや指導者との共有にも適している。

図 3.9 に、記録を保存した際の画面例および関連コードを示す。本機能の実装により、試合記録の「即時性」と「持続性」を両立させ、選手育成や戦術研究に資する有効なデータ基盤を構築することができた。

```
def delete_selected_item_and_recalculate(self):
    selected_item = self.tree.selection()
    if selected_item:
        index = self.tree.index(selected_item)
        self.tree.delete(selected_item)
        self.records.pop(index)
        self.recalculate_scores()

def recalculate_scores(self):
    self.game_count_1_2 = 0
    self.game_count_3_4 = 0
    self.point_1_2 = 0
    self.point_3_4 = 0
    self.tree.delete(*self.tree.get_children())

    for record in self.records:
        miss_1, miss_2, score, player = record
        if player == "プレイヤー1":
            if "エース" in miss_1:
                self.point_1_2 += 1
            else:
                self.point_3_4 += 1
        elif player == "プレイヤー2":
            if "エース" in miss_2:
                self.point_1_2 += 1
            else:
                self.point_3_4 += 1

        game_count_changed = False
        if self.point_1_2 >= 4 and self.point_1_2 - self.point_3_4 >= 2:
            self.game_count_1_2 += 1
            game_count_changed = True
        elif self.point_3_4 >= 4 and self.point_3_4 - self.point_1_2 >= 2:
            self.game_count_3_4 += 1
            game_count_changed = True
        elif self.game_count_1_2 == 3 and self.game_count_3_4 == 3:
            if self.point_1_2 >= 7 and self.point_1_2 - self.point_3_4 >= 2:
                self.game_count_1_2 += 1
                game_count_changed = True
            elif self.point_3_4 >= 7 and self.point_3_4 - self.point_1_2 >= 2:
                self.game_count_3_4 += 1
                game_count_changed = True

        if game_count_changed:
            self.tree.insert("", tk.END, values=("ゲームカウント", f"{self.game_count_1_2} - {self.game_count_3_4}", ""))
            self.point_1_2 = 0
            self.point_3_4 = 0
        else:
            score = f"{self.point_1_2}-{self.point_3_4}"
            self.tree.insert("", tk.END, values=(miss_1 if player == "プレイヤー1" else "",
                                                 miss_2 if player == "プレイヤー2" else "",
                                                 score))
```

図 3.6 試合記録を修正するためのコード

図 3.7 スコアの修正前と修正後の図

図 3.8 スコアリセットの確認画面の図

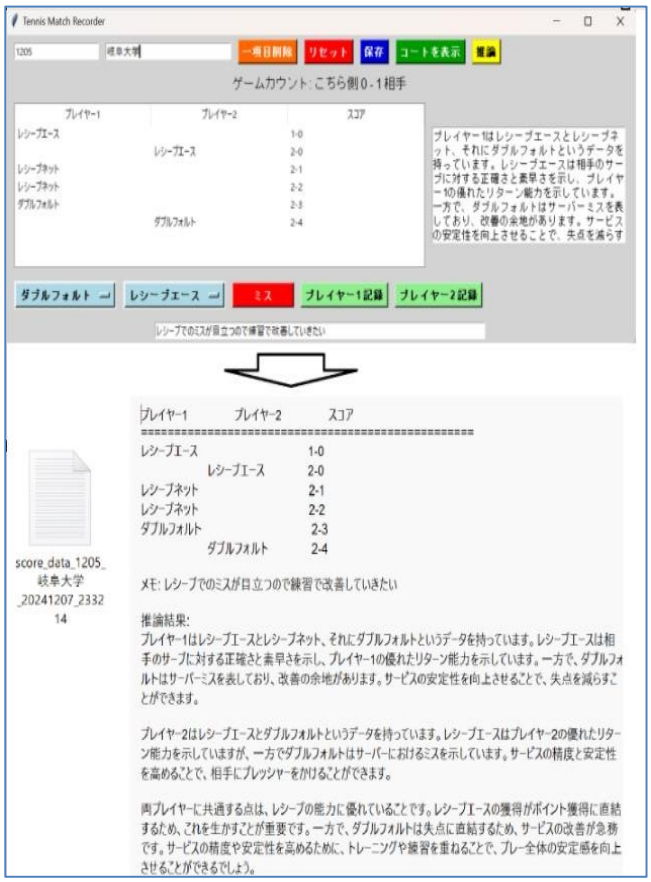

図 3.9 保存された状態を表す図

図 3.9 は、本システムにおける試合記録の保存結果を示している。左側のアプリケーション画面では「プレイヤー1」と「プレイヤー2」のスコアがポイント進行に応じて正確に更新され、試合中の得点推移が視覚的に把握できるよう整理されている。一方、右側の保存されたテキストデータでは、「Player1」「Player2」のラベルに続き、実際の試合進行に沿った得点が簡潔に整列されて記録されている。さらに、ユーザーが入力した「メモ」が追加情報として保存されており、試合状況に関する補足やコメントを残すことが可能となっている。ファイル名には日付や対戦相手の情報が組み込まれるため、記録内容を一目で識別できる点も特徴である。

図 3.10 に示す保存機能に関するコードは、試合データを適切なフォーマットで外部ファイルに保存することを目的としている。具体的には、save_data 関数内で日時データを datetime モジュールにより取得し、入力された日付および対戦相手名と組み合わせて一意のファイル名を生成する仕組みを採用した。スコアはループ処理により行ごとに抽出され、Player1 と Player2 の得点状況がそれぞれ 20 文字幅に整列されて出力されるため、保存データの可読性が確保されている。

さらに、記録データの末尾にはユーザーが入力したメモや ChatGPT による推論結果も追加される設計とした。これにより、単なる得点経過の保存にとどまらず、試合状況の解釈や振り返りを支援する情報も併せて保存できる。本保存機能は、記録内容の正確性と再現性を保持しながら、簡潔かつ効率的に試合進行を外部ファイルに残す信頼性の高い仕組みを提供している。

```
def save_data(self):
    date_text = self.date_entry.get().replace(" ", "_")
    opponent_text = self.opponent_entry.get().replace(" ", "_")
    timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"score_data_{date_text}_{opponent_text}_{timestamp}.txt"

    with open(filename, "w") as f:
        # ヘッダーを書き込む
        f.write(f"{'Player1':<20}{'Player2':<20}{'Score':<10}\n")
        f.write("="*50 + "\n")

        for row in self.tree.get_children():
            values = self.tree.item(row)["values"]
            player1_text = values[0] if values[0] else ""
            player2_text = values[1] if values[1] else ""
            score_text = values[2] if values[2] else ""
            f.write(f"{player1_text:<20}{player2_text:<20}{score_text:<10}\n")

        # 一言メモをファイルに保存
        f.write("\nメモ: " + self.memo_entry.get() + "\n")
```

図 3.10 保存機能を実装するためのコードの図

3.2.4 コートへのマークアップに関する説明

本研究では、コートの平面図上に自由に書き込みを行える「マークアップ機能」を実装した。図 3.1 に示した記録システムの全体画面において、保存ボタンの右側に配置された緑色の「コート表示」ボタンを押下することで、本機能を呼び出すことができる。

この機能の最大の利点は、試合や練習中のプレーの流れや位置情報を直感的かつ視覚的に記録できる点にある。具体的には、選手やコーチがショットの位置、ミスの発生箇所、得点パターン、あるいはポジショニングの提案などをコート図上に直接描画できる。これにより、言語だけでは十分に表現できない状況を正確に補足でき、後の分析においてプレーの再現性が大幅に高まる。

さらに、マークアップによって記録された情報は、戦術的な改善点の発見や選手の動きの傾向を把握する上で極めて有効である。特に試合後や練習後のフィードバック時に活用することで、選手は自身のプレーを客観的に振り返りやすくなり、自己理解の深化や戦略的修正につなげることが可能となる。この点は、単なるスコア記録を超えて、戦術分析と教育的支援を統合的に行えるシステムとして本研究の独自性を示すものである。

図 3.11 に、本機能を実装した画面例および対応するコード(図 3.12)を示す。

```
def draw_tennis_court(self, canvas):
    canvas.create_rectangle(55, 80, 350, 420, outline="green", width=2) # 外枠
    canvas.create_line(45, 250, 370, 250, fill="green", width=2) # ベースラインの間
    canvas.create_line(105, 180, 305, 180, fill="green", width=2) # サービスライン1
    canvas.create_line(105, 320, 305, 320, fill="green", width=2) # サービスライン2
    canvas.create_line(105, 80, 105, 420, fill="green", width=2) # サイドライン1
    canvas.create_line(305, 80, 305, 420, fill="green", width=2) # サイドライン2
    canvas.create_line(205, 180, 205, 320, fill="green", width=2) # センターライン
```

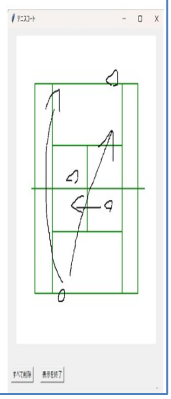

図 3.11 コートを描画するためのコードと実際に描画したコートの図

```
# コートを表示する関数
def show_court(self):
    court_window = tk.Toplevel(self.root)
    court_window.title("テニスコート")
    court_window.geometry("450x600")

    canvas = tk.Canvas(court_window, width=400, height=500, bg='white')
    canvas.pack(side="top", padx=10, pady=10)

    # コート描画
    self.draw_tennis_court(canvas)

    # マウスドラッグで自由に描画するための関数
    def start_draw(event):
        canvas.old_coords = event.x, event.y

    def draw(event):
        x, y = event.x, event.y
        old_x, old_y = canvas.old_coords
        canvas.create_line(old_x, old_y, x, y, width=2, fill="black")
        canvas.old_coords = x, y

    def end_draw(event):
        canvas.old_coords = None

    # 描画イベントをバインド
    canvas.bind("<Button-1>", start_draw)
    canvas.bind("<B1-Motion>", draw)
    canvas.bind("<ButtonRelease-1>", end_draw)
```

図 3.12 コートへのマークアップを実装するためのコード

図 3.11 は、キャンバス上にテニスコートを描画した例を示している。右側のコードでは、外枠の長方形、ベースライン、サービスライン、サイドライン、センターサービスラインを緑色で描画し、試合コートの基本構造を正確に再現している。

さらに、図 3.12 はコート上にマークアップを可能とするコードを示している。本機能は、ユーザーが自由にコー

ト上へ戦術的な書き込みやメモを行えるよう設計されており、Python の Canvas ウィジェットを用いて実装されている。ユーザーはマウス操作によって自由に線を描画することができ、具体的な戦術の流れやプレー位置を直感的に表現することが可能である。

本コードでは、次の 3 つの主要な関数を用いてマウスイベントを処理し、描画を実現している。

1. start_draw 関数：ユーザーがコート上でマウスボタンを押した際に、その位置 (event.x, event.y) を記録する。これにより描画の開始点が特定され、以降の処理に利用される。

2. draw 関数：マウスをドラッグしている間に呼び出され、canvas.create_line を用いて前回位置 (old_x, old_y) と現在位置 (x, y) の間に線を描画する。この処理を繰り返すことで、ユーザーの動きに沿った連続的な線が生成される。

3. end_draw 関数：マウスボタンを離した際に描画を終了し、開始点の記録をリセットすることで次回描画に備える。

これらの関数は Canvas ウィジェットに対して bind メソッドを用いて <Button-1>（クリック）、<B1-Motion>（ドラッグ）、<ButtonRelease-1>（リリース）の各マウスイベントと関連付けられている。その結果、ユーザーはコート上に自由にラインやマークを描画でき、特定の場面における戦略やプレーの流れを視覚的に表現することが可能となる。本機能により、数値やテキスト情報だけでは捉えにくい戦術的要素を直感的に可視化でき、試合後の振り返りや戦略検討に有効な手がかりを提供する。

3.2.5 推論機能

本機能は、記録テーブルに蓄積されたプレイヤー別データを入力として ChatGPT による推論を行い、各プレイヤーの長所・短所の抽出と、それに適合した練習提案を自動生成するものである。推論結果の表示例を図 3.13、図 3.14 に、プロンプト設計を図 3.15 に、実装コードを図 3.16 に示す。

まず self.records に格納された試合記録から、プレイヤー1・プレイヤー2 の各アクション（例：レシーブ、ネットプレー、エース、ミス等）を集計し、エース＝長所、ミス＝短所という評価軸を明示した要約を生成する。次に、この要約を ChatGPT に入力し、(1) 選手ごとの強み・弱み、(2) 頻出パターンと共通課題、(3) 改善に向けた具体的な練習メニュー（回数・時間など定量指示を含む）を出力させる。記録量が増えるほど文脈が豊かになり、提案の一貫性（具体性・網羅性）が高まる。

実装は openai.ChatCompletion.create を用い、model="gpt-3.5-turbo" を指定する。プロンプトは システムメッセージで「テニス試合分析のエキスパート」としての役割・出力形式（箇条書き、数値指定など）を定義し、ユーザーメッセージでプレイヤー別の集計データを与える。API 応答は response['choices'][0]['message']['content'] から抽出し、テキストボックス（self.inference_textbox）に表示する前に delete で初期化してから insert で挿入する。API 呼び出し失敗時は try–except によりエラーメッセージを提示することで、運用時の堅牢性を確保した。

さらに、両者に共通する傾向と相違点を並記し、チーム全体の戦術設計に活かせる視点を明示するよう促している。保存機能と連携して推論結果を記録へ併載できるため、推論→実践→再記録→再推論のループを通じて、練習計画の PDCA をデータ駆動で回すことが可能となる。

図 3.13 推論結果の表示

推論結果:
プレイヤー1はレシーブエースとレシーブネット、それにダブルフォルトというデータを持っています。レシーブエースは相手のサーブに対する正確さと素早さを示し、プレイヤー1の優れたリターン能力を示しています。一方で、ダブルフォルトはサーバーミスを表しており、改善の余地があります。サービスの安定性を向上させることで、失点を減らすことができます。

プレイヤー2はレシーブエースとダブルフォルトというデータを持っています。レシーブエースはプレイヤー2の優れたリターン能力を示していますが、一方でダブルフォルトはサーバーにおけるミスを示しています。サービスの精度と安定性を高めることで、相手にプレッシャーをかけることができます。

両プレイヤーに共通する点は、レシーブの能力に優れていることです。レシーブエースの獲得がポイント獲得に直結するため、これを生かすことが重要です。一方で、ダブルフォルトは失点に直結するため、サービスの改善が急務です。サービスの精度や安定性を高めるために、トレーニングや練習を重ねることで、プレー全体の安定感を向上させることができるでしょう。

図 3.14 ChatGPT による推論結果(全文)

あなたはテニスの試合の分析、それに応じた指導をすることが出来るエキスパートです。以下のプレイヤー1のデータ{player1_data}とプレイヤー2のデータ{player2_data}から、それぞれの得意なプレイとミスが多いパターンを分析し、それに応じたアドバイスを３００文字以内でしてください。

図 3.15 プロンプト設計の例

あなたはテニスの試合の分析、それに応じた指導をすることが出来るエキスパートです。以下のプレイヤー1のデータ{player1_data}とプレイヤー2のデータ{player2_data}から、それぞれの得意なプレイとミスが多いパターンを分析し、それに応じたアドバイスを３００文字以内でしてください。

図 3.16 実装コード

3.3 実装環境

本研究で開発した試合記録システムの実装環境を表 3.1 に示す。開発言語には Python を採用し、統合開発環境として Visual Studio Code を用いた。実行環境は Windows PC 上で構築されている。 本システムの主要なモジュールは以下の通りである。まず、GUI の構築には Tkinter を利用し、ユーザーが直感的に操作できるインターフェースを実現した。試合データの保存処理では datetime モジュールを用いて日付を自動取得し、ファイル名として利用することで記録の管理性を高めている。また、推論機能の実装には OpenAI モジュールを使用し、外部 API を通じて GPT-3.5-turbo モデルを呼び出す仕組みを導入した。 なお、Python から ChatGPT を利用するためには OpenAI API のクレジット登録が必要となる。本モデルの利用料金は 1,000 トークンあたり 0.0015 ドルと比較的低廉であり、教育現場や研究利用においても十分に実用的なコストで運用可能である。 本実装環境の構成により、GUI ベースの記録支援機能と AI を活用した推論機能を統合した、実用性の高い試合記録システムの開発が可能となった。

表 3.1 実装環境

OS環境	Windows11
開発環境	Visual Studio Code(Ver1.95.3)
使用言語	python(Ver3.11.9)
使用モジュール	Tkinter、datetime、messagebox、openai

3.4 システム構成図

本システムの全体構成を図 3.17 に示す。ユーザーは試合中のデータを入力し、これらの情報はプレイヤーごとに集約される。集約されたデータは ChatGPT に送られ、各プレイヤーにおけるエースやミスの傾向を分析し、その結果をテキストとして出力する仕組みである。

さらに、入力された記録および生成された推論結果は、指定されたファイルに保存される。これにより、単なるスコアの記録にとどまらず、AI による分析と保存を組み合わせた一貫したデータ管理が可能となり、試合後の振り返りや戦術検討を効率的に行うための基盤を提供する。

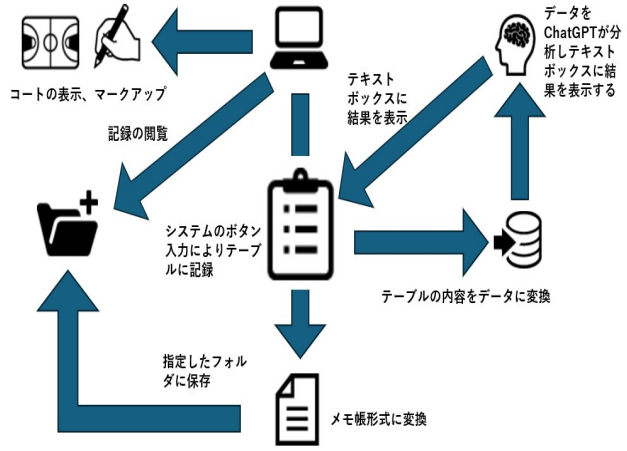

図 3.17 システム構成図

4. 評価実験

4.1 実験結果

本システムは、ソフトテニスにおける試合データを効率的に収集・分析することを目的として設計されている。その有効性を検証するため、まず開発者自身が過去の試合映像を用いてリアルタイムで記録を行い、操作性および分析に必要なデータの収集精度を確認した。続いて、ソフトテニス経験者 5 名および未経験者 5 名を対象に試用を依頼し、UI の見やすさ、操作感、推論機能の有用性について評価を得た。

その結果、図 4.1 および図 4.2 に示すように、一試合を通して「誰がどのようなミスをしたか」「どのようにエースを決めたか」といった情報を正確に取得できることが確認された。また、図 4.3 に示す ChatGPT による推論結果からは、プレイヤーごとの特徴や課題、改善に向けた練習方法が提示され、一定の有効性を示した。これにより、データを基に「どのカウントでミスが出やすいか」などの傾向を分析し、次回の試合に向けた具体的な練習方針を立案できる可能性が示唆された。さらに保存機能により、過去

の試合データと比較して成長の度合いや課題の変化を把握できる点も有用であった。

ユーザ評価の結果、未経験者からは「推論機能」や「コートへのマークアップ機能」といった他の収集システムには見られない機能が便利であるとの意見や、「ボタンが色分けされており視認性が高い」といった肯定的な評価が得られた。一方で、記録用ボタンの数が多く、初めて使用する者にとっては操作が分かりにくい点が課題として指摘され、UI 配置やユーザビリティの改善が必要であることが分かった。

経験者からは、システムの全体像は容易に理解できたものの、ChatGPT による推論結果の内容が抽象的であり、戦術や練習方法をより具体的に提示する必要があるという指摘を受けた。これらの結果から、本システムはデータ収集・可視化機能において有効である一方、UI の操作性改善および推論の具体性向上が今後の課題であるといえる。

Player1	Player2	Score
ダブルフォルト		0-1
レシーブエース		1-1
レシーブアウト		1-2
レシーブアウト		1-3
レシーブアウト		1-4
ゲームカウント	0 - 1	
	ボレーエース	1-0
レシーブエース		2-0
	ボレーエース	3-0
レシーブネット		3-1
	ボレーエース	4-1
ゲームカウント	1 - 1	
サービスエース		1-0
レシーブアウト		1-1
レシーブエース		2-1
レシーブエース		3-2
レシーブエース		4-2
ゲームカウント	2 - 1	
レシーブアウト		0-1
	レシーブアウト	0-2
	スマッシュエース	1-2
レシーブエース		2-2
レシーブネット		2-3
レシーブネット		2-4
ゲームカウント	2 - 2	

図 4.1 実際に行った試合の記録の内訳（前編）

4.2 今後の展望

本システムは、スコア記録、エラーやエースの管理、ChatGPT による推論、さらにコートへのマークアップ機能を統合した点に特徴がある。これにより、試合の進行を視覚的かつ正確に把握できるという利点を備えている。

ゲームカウント	2 - 2	
サービスエース		1-0
レシーブエース		2-0
レシーブアウト		2-1
	レシーブネット	2-2
レシーブエース		3-2
ダブルフォルト		3-3
レシーブネット		3-4
レシーブエース		4-4
レシーブアウト		4-5
レシーブネット		4-6
ゲームカウント	2 - 3	
	ボレーミス	0-1
レシーブエース		1-1
	ボレーエース	2-1
レシーブネット		2-2
レシーブエース		3-2
	ボレーミス	3-3
レシーブエース		4-3
	レシーブエース	5-3
ゲームカウント	3 - 3	
レシーブエース		1-0
	ボレーエース	2-0
	スマッシュエース	3-0
レシーブアウト		3-1
	ボレーエース	4-1
	ボレーミス	4-2
レシーブアウト		4-3
レシーブアウト		4-4
レシーブアウト		4-5
レシーブエース		5-5
	レシーブエース	6-5
	ボレーエース	7-5
ゲームカウント	4 - 3	

メモ: レシーブアウトが目立ったので改善が必要

図 4.2 実際に行った試合の記録の内訳（後編）

一方で、初心者を含め誰もが容易に操作できるような直感的 UI の改善が求められる。特に、リアルタイムでスコアやミスを入力する場面では、シンプルで分かりやすいボタン配置やラベル表示が重要である。また、入力の手間を最小化し、誤操作を防ぐための設計も不可欠となる。

さらに、記録ミスが発生した場合に即座に修正できるリアルタイム修正機能の強化も課題である。試合の進行を止めることなく、誤った入力を訂正できる仕組みや、過去の履歴を参照して修正できる機能を実装することで、信頼性の高い記録が可能となる。

コートへのマークアップ機能についても改良の余地がある。描画自由度が高すぎると整理が困難になり、誤操作によって情報が重なりすぎることがある。そこで、描画履歴の管理や消去機能の追加、さらにマークアップを画像として保存する機能を実装する必要がある。例えば Python の Pillow ライブラリを活用することで、マークアップを画像ファイルとして出力し、後の分析に利用できるだろう。

推論機能に関しては、一定の水準で有用な回答を得ることはできたが、抽象的な提案にとどまる場合もあった。今後は、プロンプト設計を見直し、より具体的な数値や練習メニューを含む回答を促す必要がある。また、ラケットや戦術要因を考慮した多角的なアプローチを組み込むことで、実際のプレイヤーの特性に合致した提案が可能になると考えられる。

以上を踏まえ、今後の展望としては、① UI と修正機能の改善、② マークアップ機能の整理性と保存機能の追加、③ 推論機能の精度と具体性の向上、の三点を優先的に進めることが重要である。これらの改良を通じて、試合記録から分析、振り返りまでを一貫して支援できる、実用性の高いシステムへ発展させていきたい。

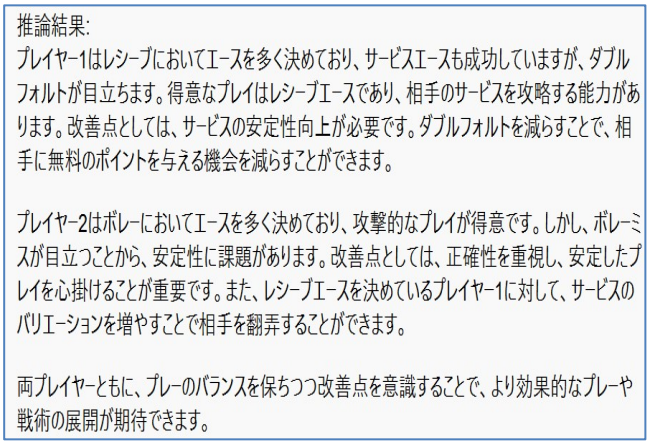

図 4.3 推論結果

5. おわりに

本研究では、ソフトテニスの試合を効率的に記録・分析するための試合記録システムを開発した。Python を用いて構築された本システムは、リアルタイムでのスコア記録、データの保存、コート上のマークアップ、そして ChatGPT を活用した推論機能を統合している。従来、紙媒体で行われていた作業をデジタル化することで、記録の効率化と保存性の向上を実現し、試合後の分析や指導に役立つ基盤を提供した。

システム実装では、デュース処理を含むスコア管理、日付や対戦相手を付与した保存機能、直感的なコートマークアップ機能を導入した。また推論機能により、プレイヤーごとの強み・弱みを自動的に分析し、練習方針の検討に活用できる仕組みを実現した。評価実験の結果、試合の流れに沿った記録やテキスト保存の有効性が確認され、分析や振り返りに活用可能であることが示された。一方で、リアルタイム入力時にボタン数の多さが操作負担となり、UI の簡素化や誤操作防止の仕組みが今後の課題であることも明らかになった。

本研究の成果として、①紙媒体での記録に伴う手間や紛失リスクの低減、②デジタルデータによる長期的な蓄積と分析の容易化、③直感的なマークアップによる戦術的理解の支援、④AI 推論による強み・弱みの可視化、を挙げることができる。これらを統合した本システムは、試合記録と分析の効率化に寄与し、ソフトテニスにおける競技力向上と指導支援に資する有用なツールとなる。

参考文献

[1]坂東将光、齊藤彰, クラウドベースのソフトテニス試合記録システムの開発, 近畿大学工業高等専門学校近畿大学工業高等専門学校研究紀要 (17),pp.55-58,2024-03-15

[2]塩見 一成, テニスにおけるラリーの継続を目指した初心者指導に関する実践的研究-グラウンドストロークに着目して, 富山短期大学紀要 59 巻(2023.3)

[3]髙橋仁大, 中村和樹, 岡村修平, 大澤啓亮, 柏木涼吾, 村上俊祐, テニスの女子ダブルスにおける地方学生選手の課題を探るためのゲームパフォーマンス分析, 日本スポーツパフォーマンス学会スポーツパフォーマンス研究, 16, pp.1-9, 2024

[4]松江拓, 前田明, ソフトテニスラケットにおけるストリングテンションの違いが打球速度および打球コントロールに及ぼす影響, スポーツパフォーマンス研究,15, pp.186-192, 2023

[5]矢野浩二朗 OpenAIChatAPI を用いた自動採点付き学習アプリの開発と授業実践, 特集「生成 AI を活用した新しい教育への展望」2023 年 12 月 01 日コンピュータエデュケーション 55、pp.25-31

[6]鍋田歩美, 相羽大輔, 配色の違いが板書の視認性に及ぼす効果板書に対する正常色覚者の視認性評価の比較, 愛知教育大学学術情リポジトリ障害者教育・福祉学研究第 20 巻 pp.117～124（2024）

[7]谷川由紀子 福住伸一, 情報システム開発現場におけるユーザビリティ向上にための人間中心プロセス支援環境：開発と強化, 理化学研究所 AIP センターヒューマンインタフェース学会論文誌 25(1),pp.65-76,2023-02-25