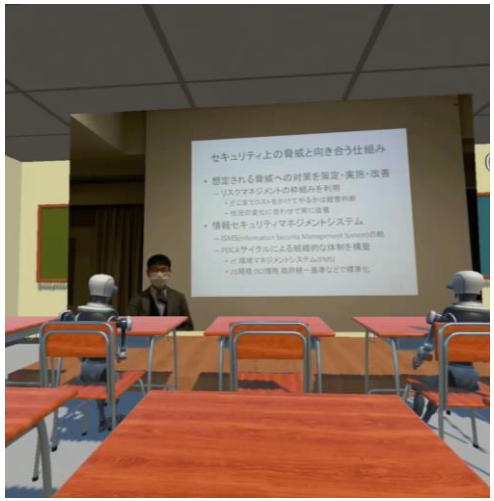
ビデオ視聴のマルチモーダル学習分析に向けたOpencastへのARグラスIMU計測機能の実装

京都工芸繊維大学 情報基盤センター
〇永井 孝幸

2026/3/4 CLE48@NII

はじめに

- 講義視聴状況（うなずき）の共有
  - Takayuki Nagai and Takuma Shinohara, *Shared Nods, Shared Presence: Enhancing Engagement in VR On-Demand Lectures,* 1st International Conference on Learning Evidence and Analytics(ICLEA),2025



- 長時間の視聴・計測に適さない
- 計測用（視聴用）アプリが必要

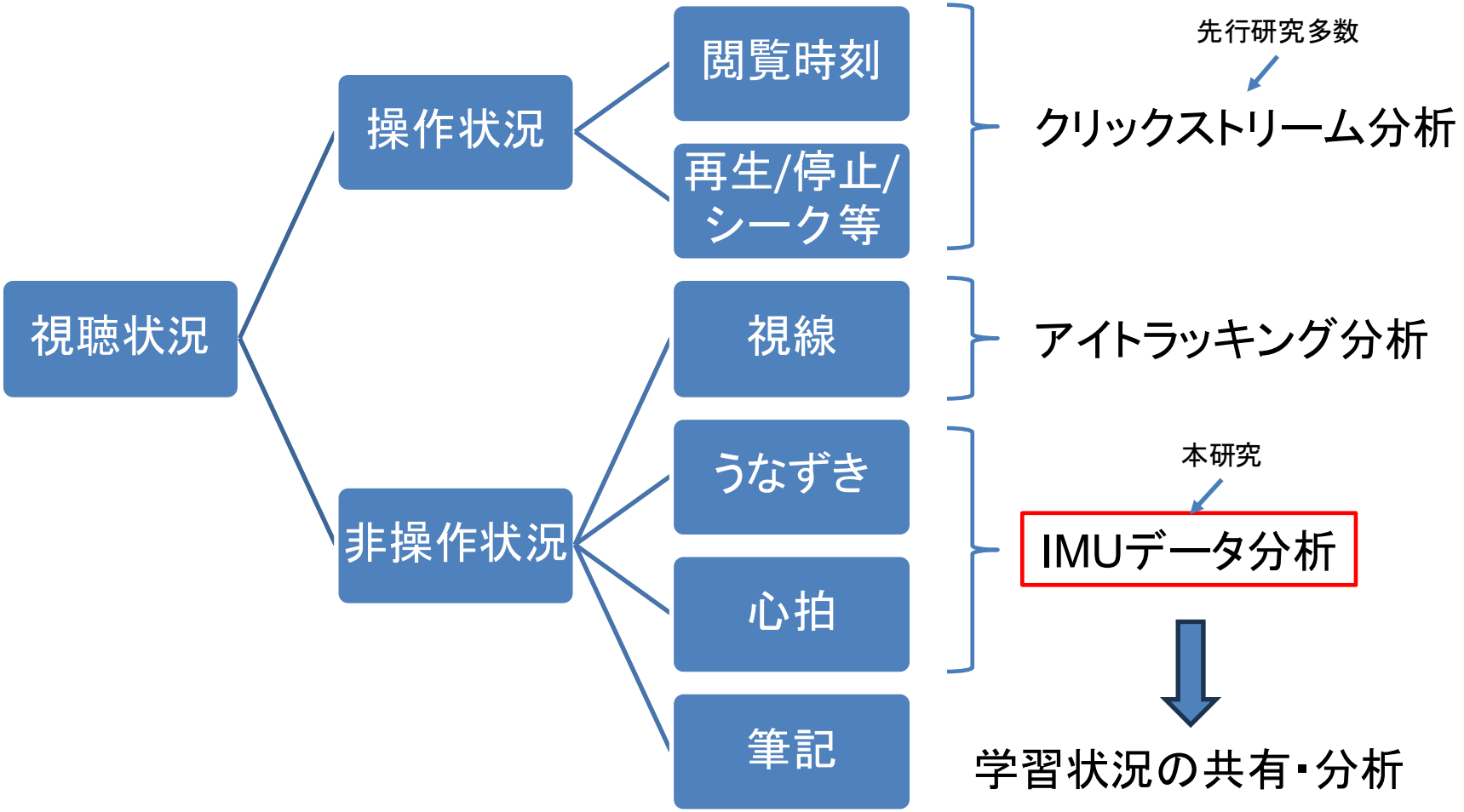
⟹ Opencast/Moodle + ARグラス(IMU計測)

頭部IMUデータの取得

- IMU(慣性計測ユニット)データ
  - 3軸角速度(ヨー・ピッチ・ロール)＋3軸加速度
- 利用デバイス：ARグラス(頭部IMU)
  - 「AR」ではなく「IMUセンサー搭載メガネ」が本質

XRealOne Pro
- 高頻度(1000Hz)IMUデータ
- 公式SDKあり(Unity用)
- 通信ライブラリ自作(Python)

ビデオ視聴状況の分析



京都工芸繊維大学
KYOTO INSTITUTE OF TECHNOLOGY

本発表の内容

Moodle/Opencast にIMU計測機能を実装

- システム構成,実装方法

IMUデータとビデオ再生時刻の対応付け

- 再生速度変更、シーク操作への対応

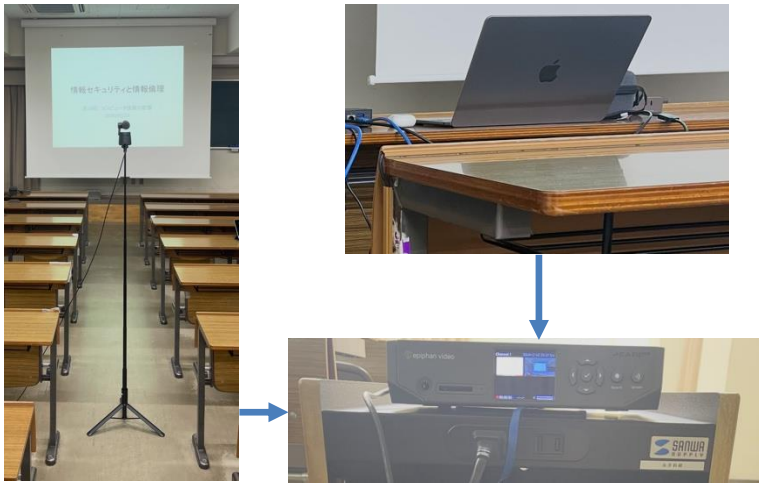
IMU計測機能の動作検証

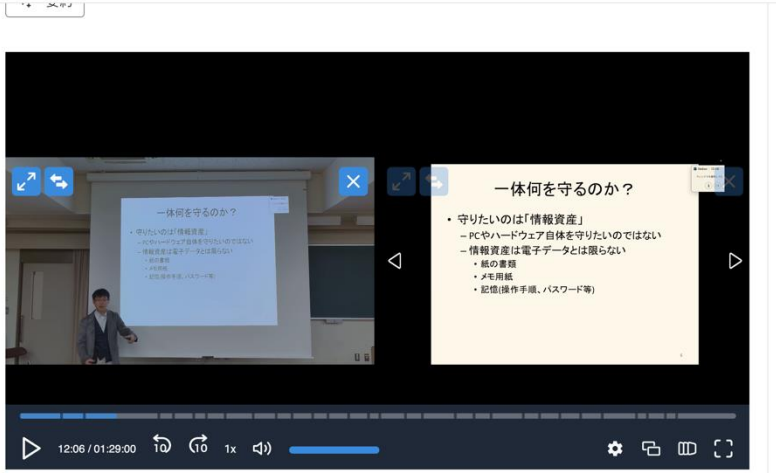
- 再生速度変更条件での計測結果

今後の応用

講義集録配信システムの構成

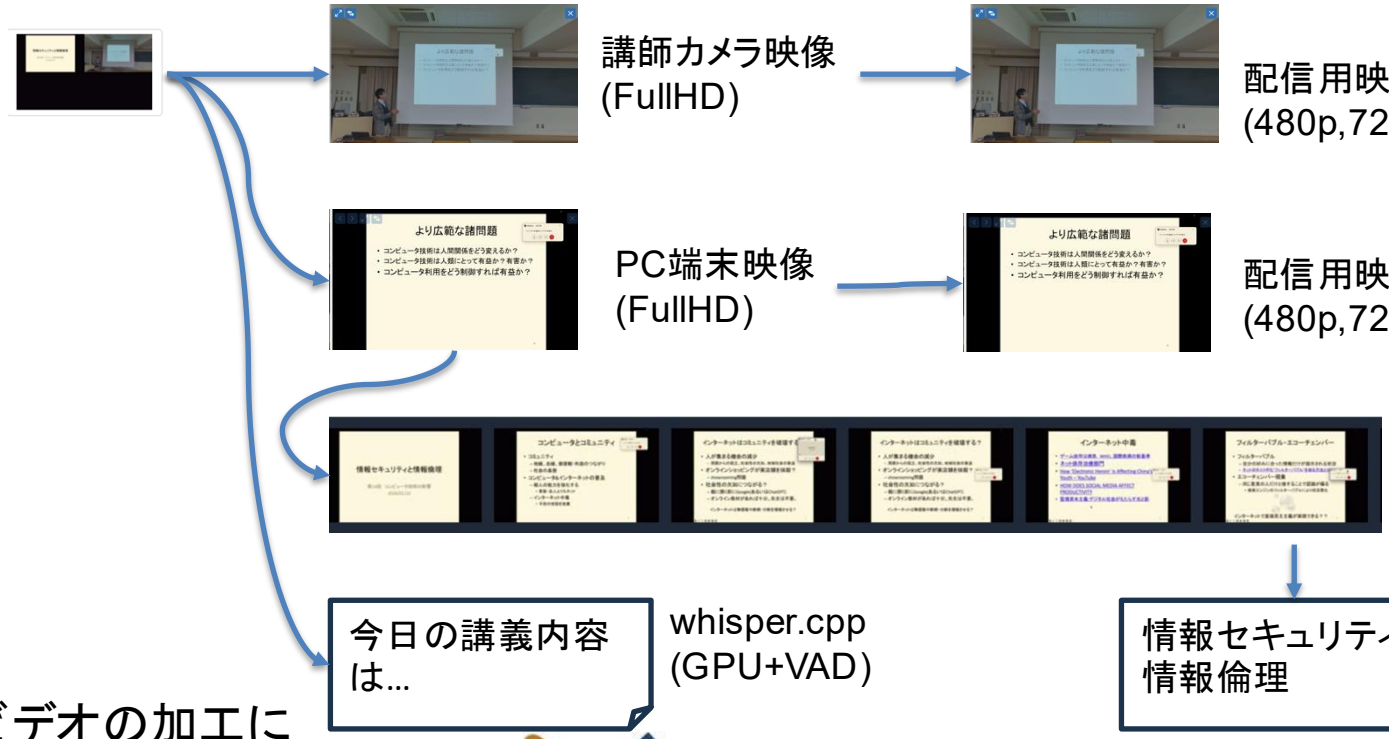
- Opencast + Moodle
  - All-in-One コンテナイメージを使用
  - 収録にepiphan Pearl Nano(2入力,1出力)を使用
  - Opencast Video Providerプラグインを利用





開発したビデオ加工フロー

- 4K合成動画を講師映像・PC端末映像に分離
  - 文字起こし,スライド抽出,スライドOCRも実施

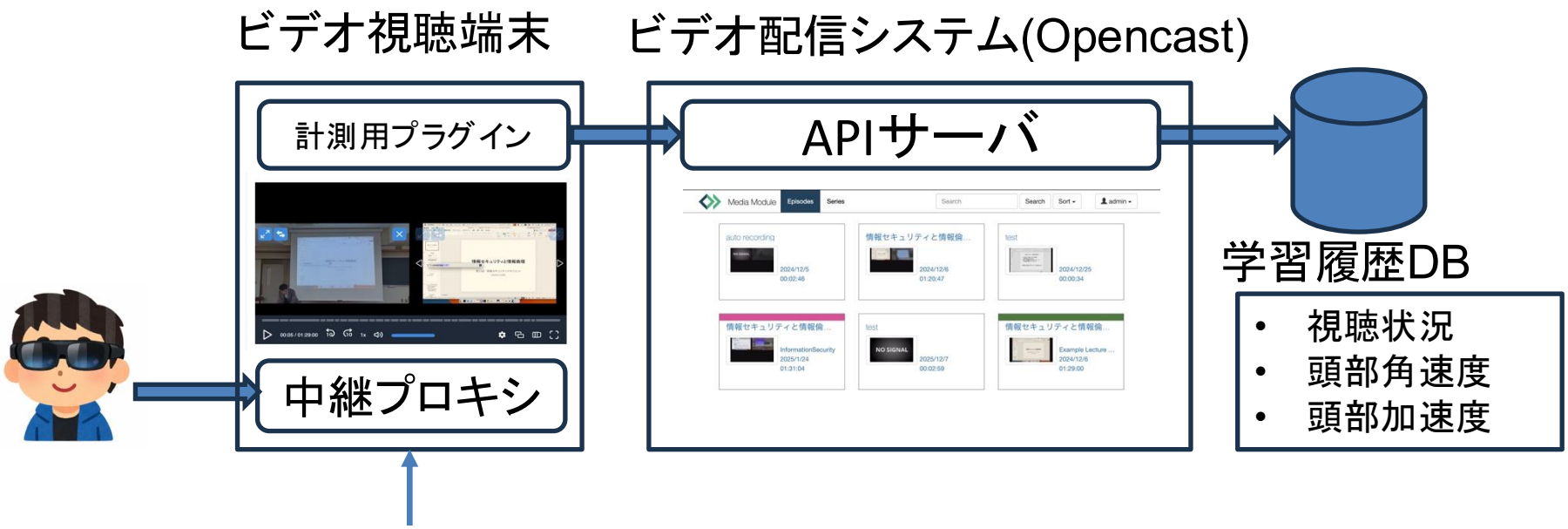


90分ビデオの加工に約84分

京都工芸繊維大学 KYOTO INSTITUTE OF TECHNOLOGY

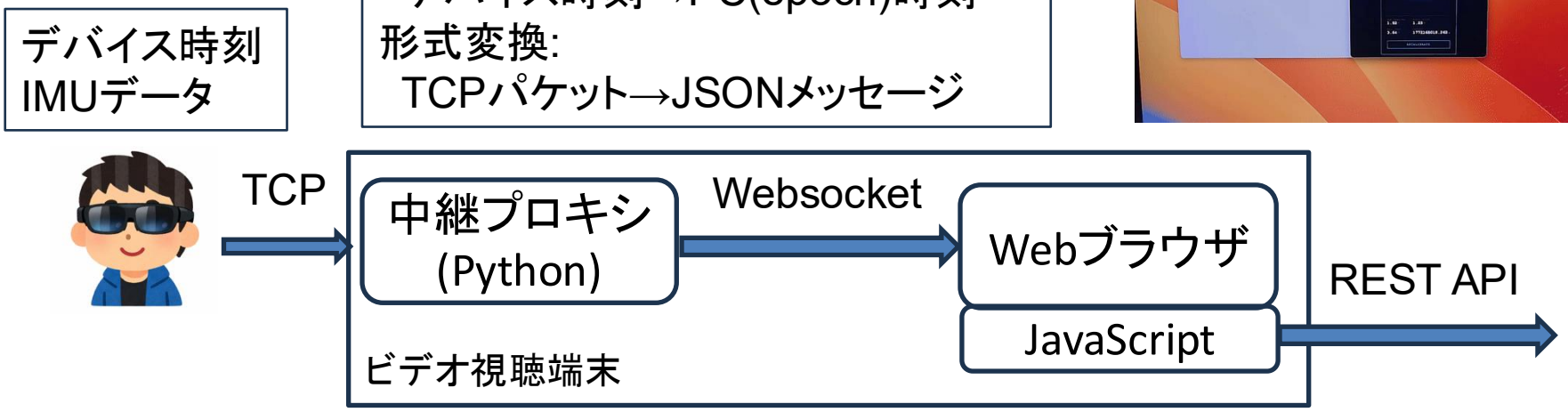
マルチモーダル学習分析への拡張

- Paella Player：IMU計測用プラグインを開発
- Opencast: APIサーバ機能を追加



ARグラスを用いたIMU計測

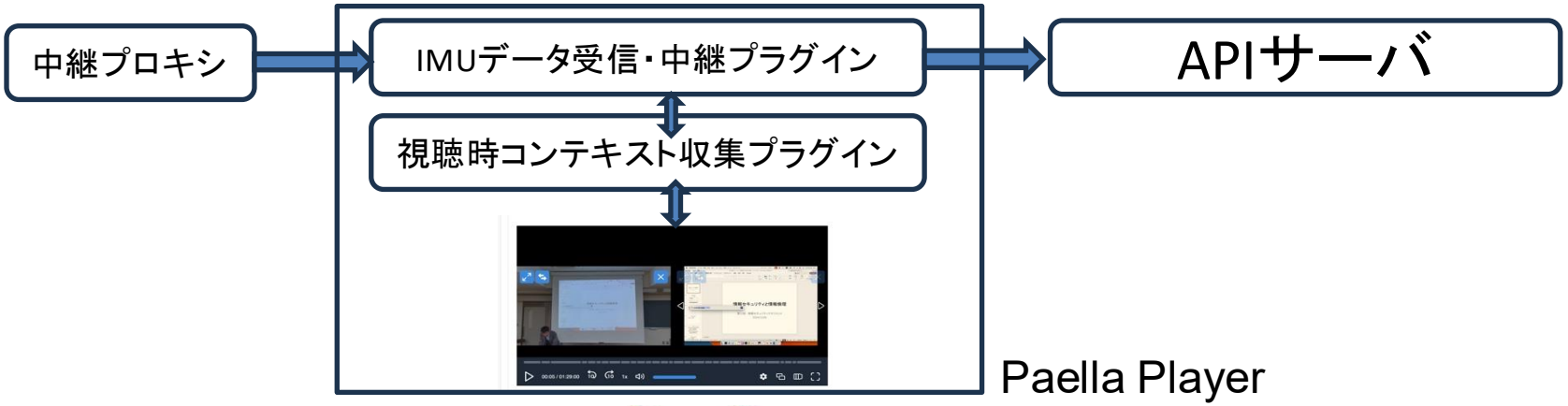
- 視聴端末上で中継プロキシを実行



頭部姿勢推定は自分で行う必要がある
→今回はimufusion(AHRS法の実装)を利用

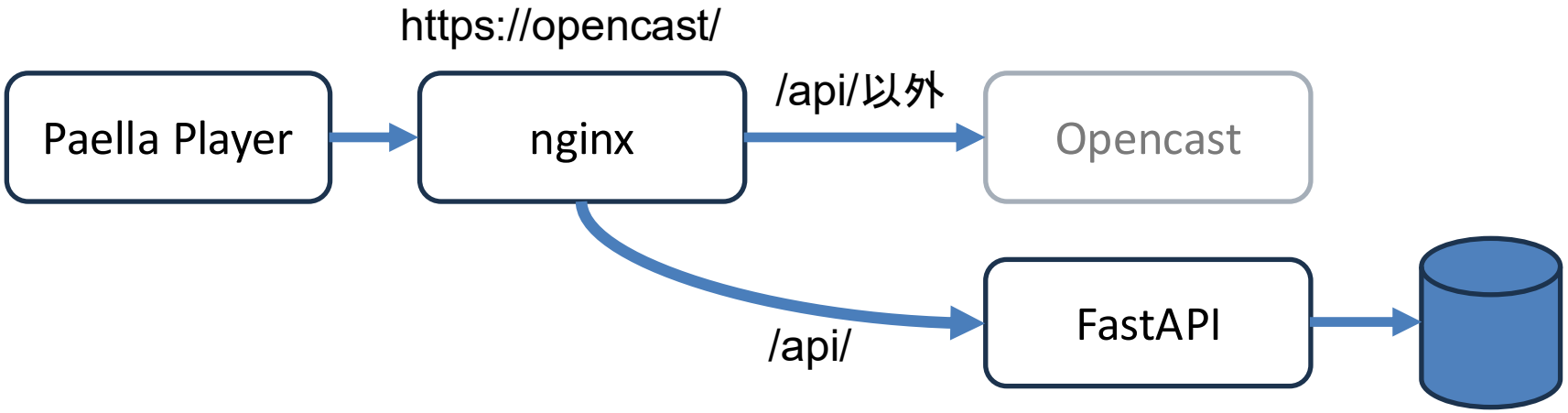
Paella Playerの拡張

1. 視聴時コンテキスト収集プラグイン
   - 再生状態(再生/一時停止,速度,再生トラック等)
2. IMUデータ受信・中継プラグイン
   - 時刻対応づけ用データの取得(数十ms間隔)
   - IMUデータ/視聴コンテクストをAPIサーバに中継



OpencastのREST API拡張

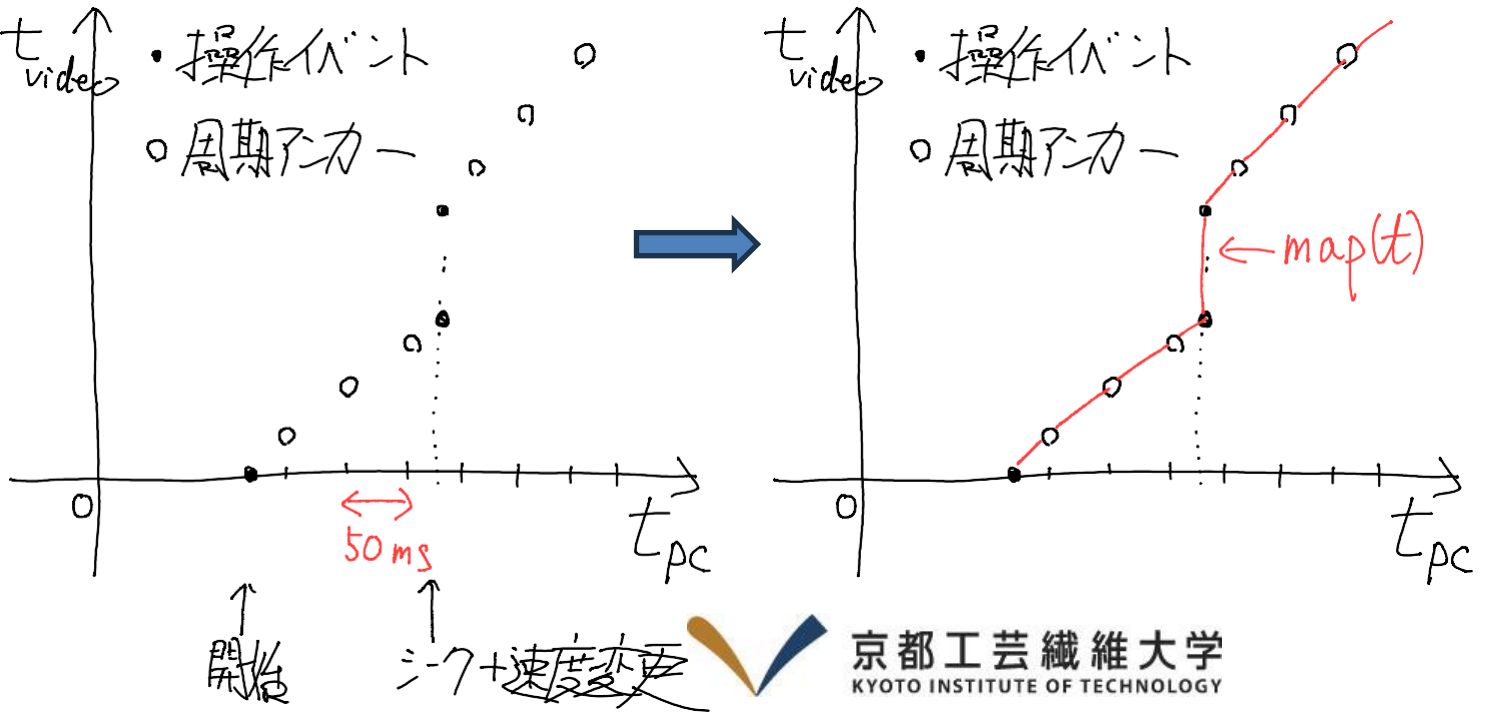
- サイドカー方式でREST API拡張
  - CORSの問題が起きない,通信路を一本化
  - Opencastと独立に開発・更新できる



エンドポイント:
- 視聴セッション管理
- IMUデータ保存

IMUデータとビデオ再生時刻の対応付け

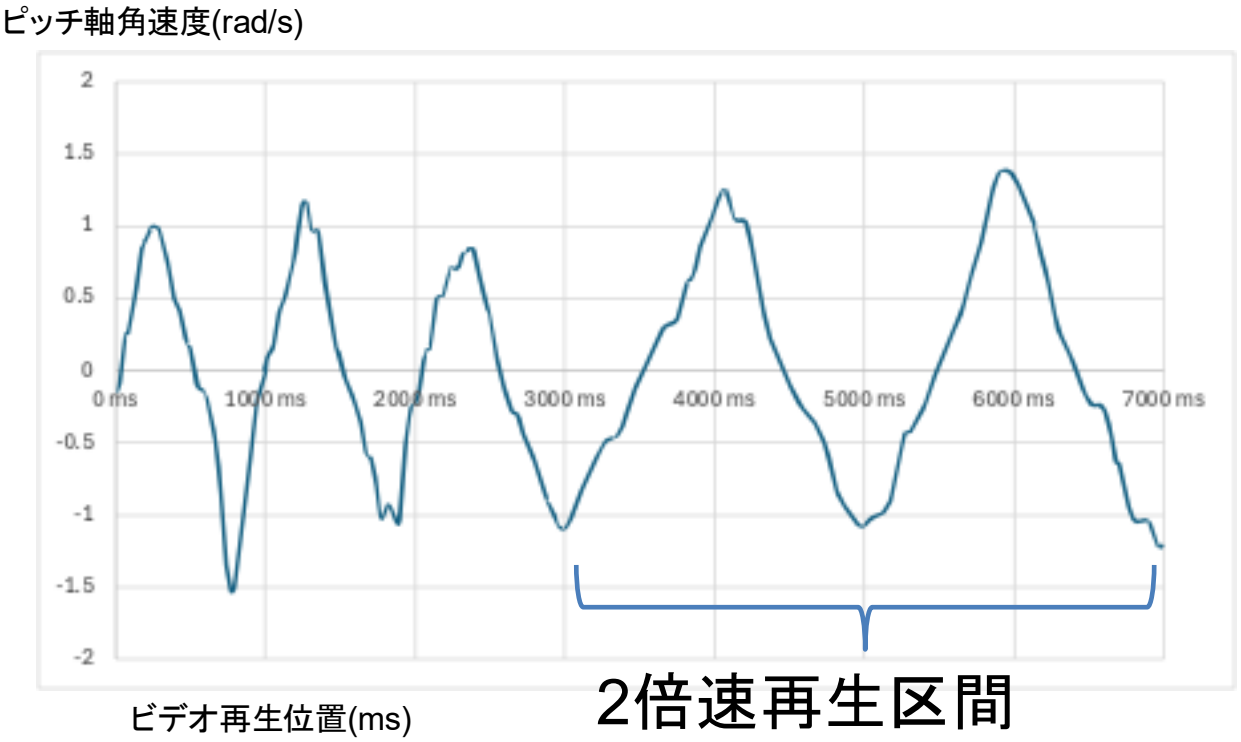
- 対応付け関数：PC時刻→ビデオ再生位置
  - IMUデータ毎(1000Hz)にビデオ再生位置を取得？
  - 周期アンカー点＋操作イベント点から再構成



map(t)で
ビデオ再生位置に
変換

うなずき時のIMU計測結果

- メトロノームに合わせて毎秒1回うなずき
  - ビデオ再生速度を1倍→2倍に変更



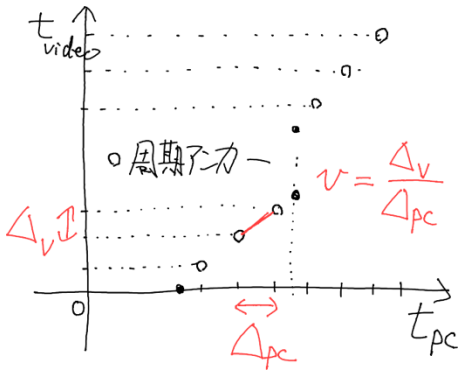
波形が引き延ばされる
→時刻対応付けOK

アーティファクトを観測

計測精度について

- 対応付け関数の傾きが15%程度ずれる
  - map点を50ms間隔で取得した場合
    - MacBook Air M2, Chromeブラウザ

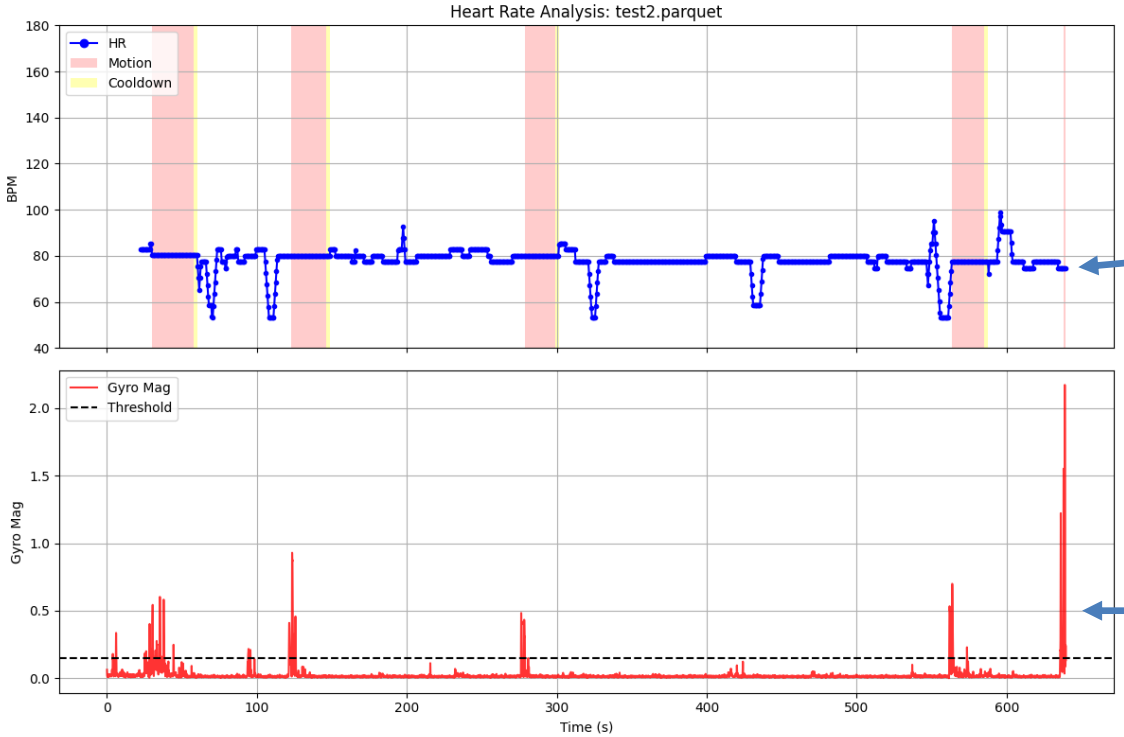
再生速度	再生位置計測誤差
x1.0	7ms程度
X1.5	11ms程度
X2.0	14ms程度



- うなずき(数百ms～数秒)をとらえるには十分
- 映像(30fps)より高精度

応用可能性の検討:心拍推定

- IMU3軸加速度をPCAで1次元化
  - HPS法で基本周波数推定(FFTではうまくいかず)



フィットネストラッカーの値と概ね一致

推定可能性あり

頭部加速度ノルム

運用上の狙い

- 分析の基礎データをOpencastに集約
  - 講義動画(講師/PC),字幕,スライド,OCR結果
- 日常システム(LMS)をマルチモーダル化
  - 学習者はMoodle上で視聴
    - IMU中継プロキシを動かす必要はある
- 照明の影響を受けない

今後の課題

技術面

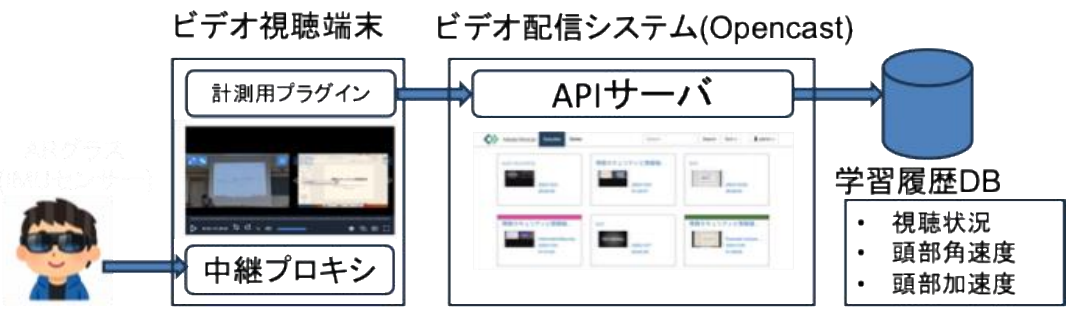
- APIサーバの増強：多人数同時利用
- 時刻対応付け手法の改良
- 分析に必要なデータ精度の見極め
- 学習行動抽出処理・ダッシュボードの実装
- 他IMUデバイスへの対応

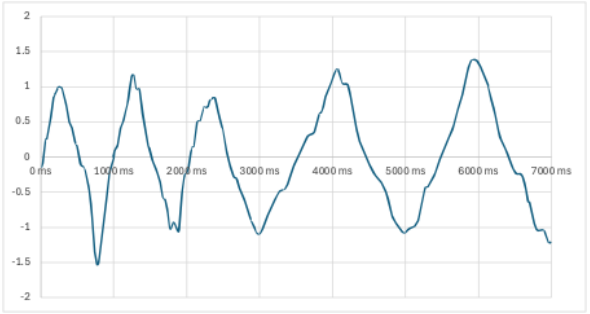
学習分析

- 実データの収集
- IMUデータ分析手法
  - うなずき、首かしげ
  - 心拍、集中度、etc.
- 学習状況(ex.うなずき)共有
- 講義状況との対応付け
- プライバシー保護
  - 利用規程
  - 二次利用

まとめ

- Moodle/Opencast 上の講義動画視聴に、ARグラスIMUを統合する基盤を実装
  - IMU時刻とビデオ再生位置を対応付けて保存
  - 再生速度変更を含む条件で動作検証し、応用可能性を示した





- *本研究の一部はJSPS 科研費JP22K12313 の助成を受けたものである*

付録

視聴セッションの管理

- 1セッションの定義
  - 開始:Playerのロード完了/(2回目以降の)再生
  - 終了:末尾まで再生 or 再生停止操作
- 保持する情報
  - セッションID(UUID)
  - ビデオID(UUID)
  - 認証方式(cookie/basic認証)
  - ユーザ情報
  - クライアント情報(IPアドレス,User-agent等)

APIサーバの設計

- IMUは高頻度(最大1000hz想定)
  - クライアント側で一時バッファ
  - バッチPOSTで平準化
    - コンパクトJSON形式を利用
- 非同期DB書き込みで取りこぼしを減らす

コンパクトJSON（IMUバッチ）

```
{
    "version": 2,
    "eventId": "<uuid>",
    "viewingSessionId": "<uuid>",
    "clientTimestampMs": <int>,
    "samplerateHz": <int>,
    "batchSeq": <int>,
    "isFinal": <bool>,
    "timebase": "xreal.imu.ts_ms",
    "mappingIntervalMs": <int>,
    "map": {
      "tPerfUs": [<int>, <int>, ...],
      "videoTimeMs": [<int>, <int>, ...]
    },
    "contextAtFlush": {
      "playerState": "playing" | "paused",
      "playbackRate": <number>,
      "track": "<string>",
      "fullscreen": <bool>
    },
    "events": [],
    "flushMeta": {
      "reason": "batchsize" | "<string>",
      "boundary": <bool>
    },
    "samples": {
      "t0PerfUs": <int>,
      "dPerfUs": [0, <int>, <int>, ...],
      "gx": [<number>, <number>, ...],
      "gy": [<number>, <number>, ...],
      "gz": [<number>, <number>, ...],
      "ax": [<number>, <number>, ...],
      "ay": [<number>, <number>, ...],
      "az": [<number>, <number>, ...]
    }
}
```

4～8サンプル

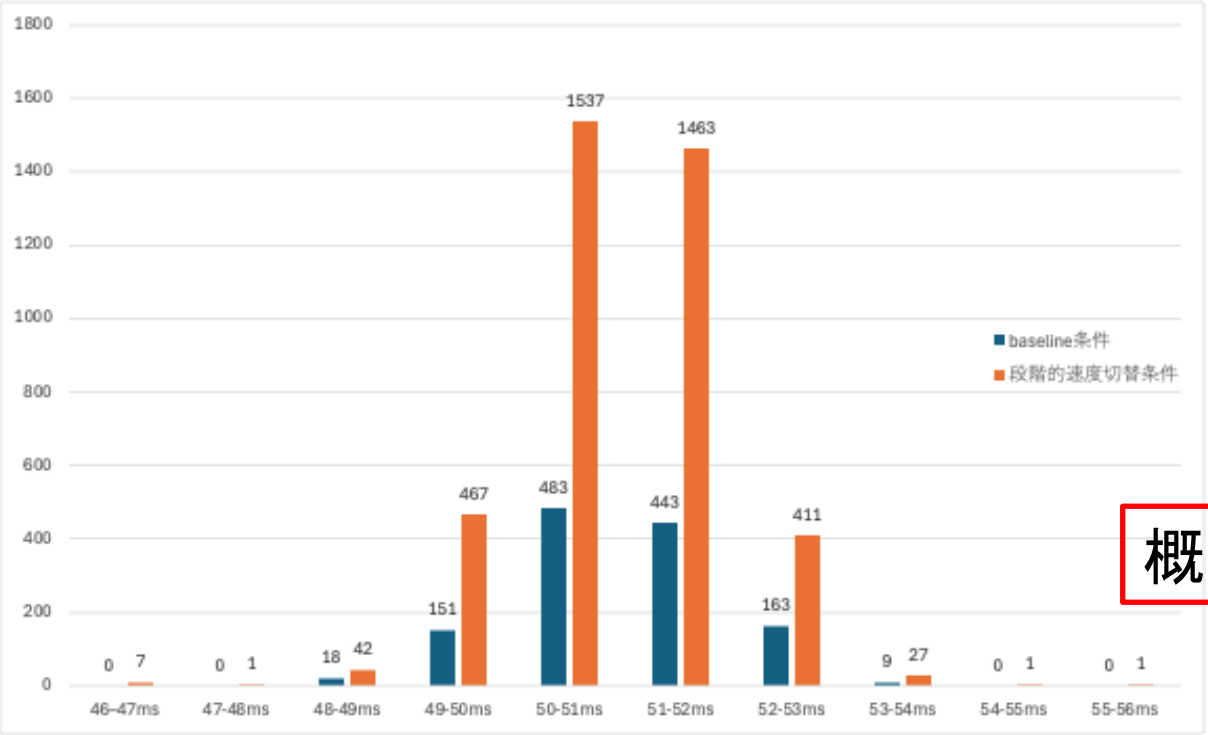
200サンプル
(200ms分)

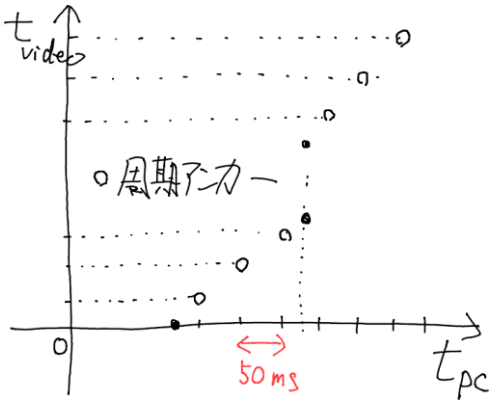
共通メタデータを外出しして転送量を削減

- サンプル時刻は差分配列で表現
- 文脈（再生状態/速度/トラック等）と同梱
- map点（epoch↔video）も同一バッチに格納

動作検証:タイムスタンプの計測精度

- map点の取得間隔は一定になっているか？
  - 50ms間隔で取得する設定で実行



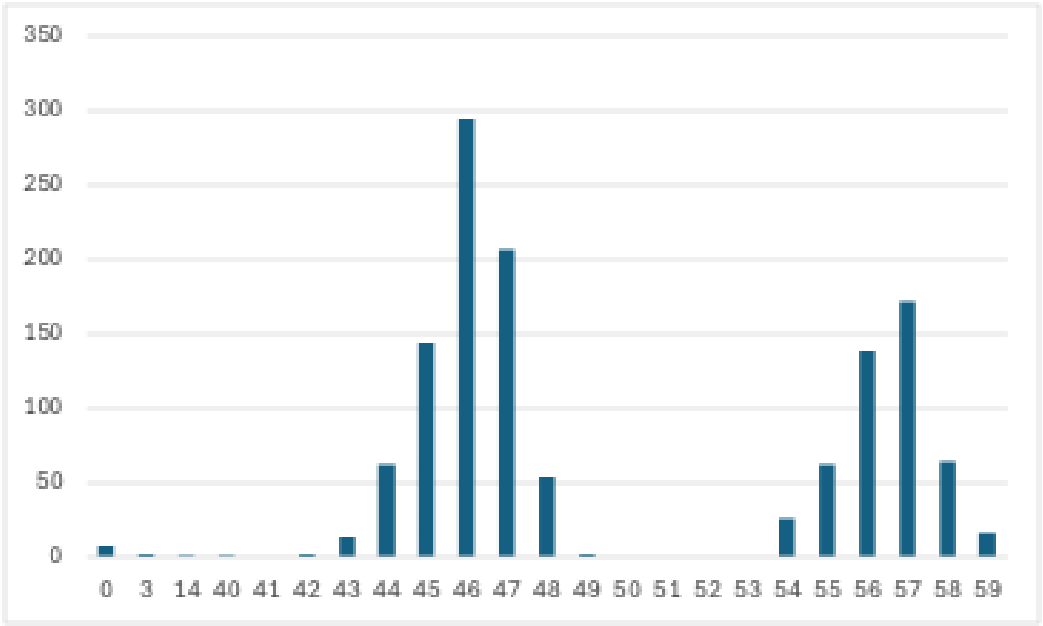


概ね50ms間隔で取得

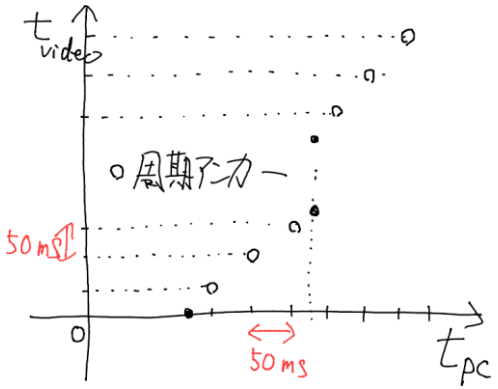
京都工芸繊維大学
KYOTO INSTITUTE OF TECHNOLOGY

検証:ビデオ再生位置の計測精度

- 1倍速再生時の再生位置取得間隔は？
  - 50msごとに計測→再生位置の差分も50ms？



46msと57msにピーク

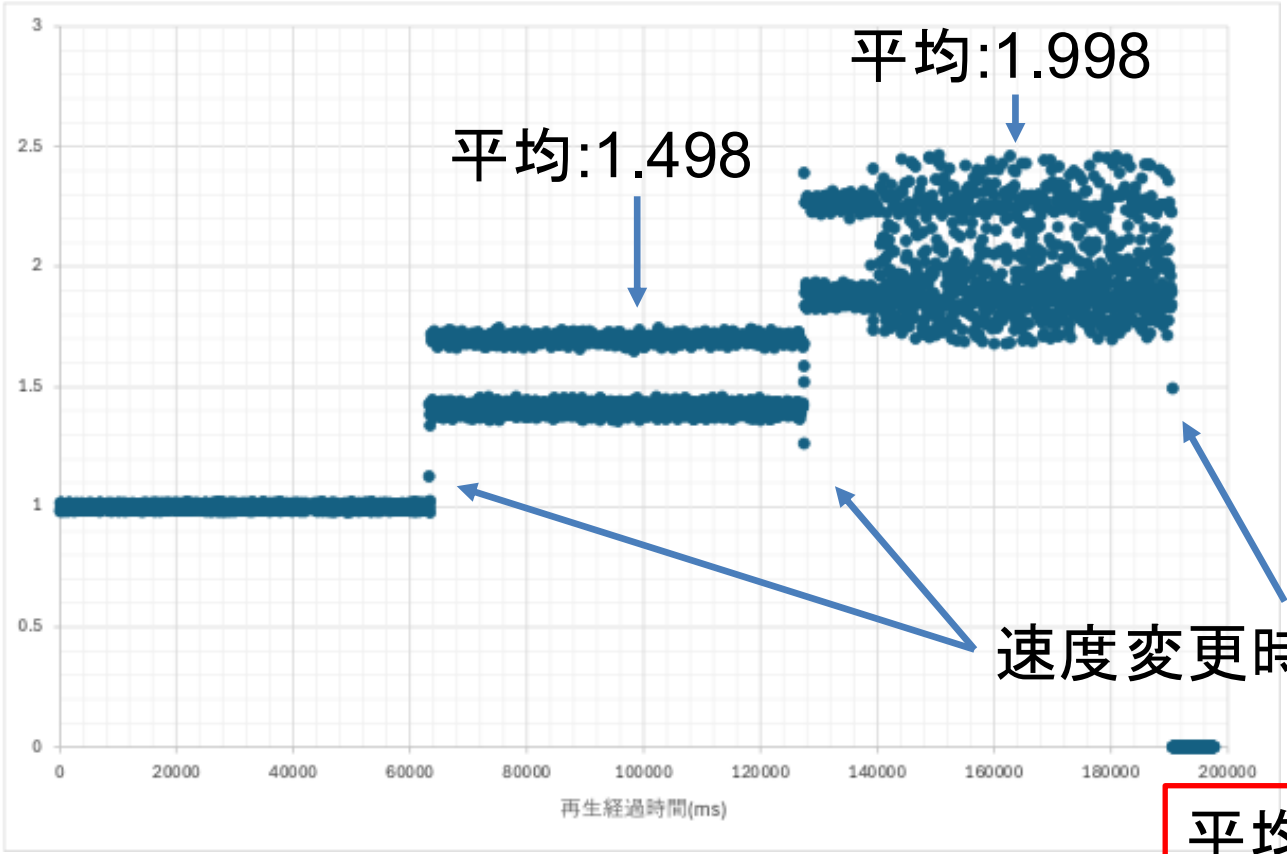


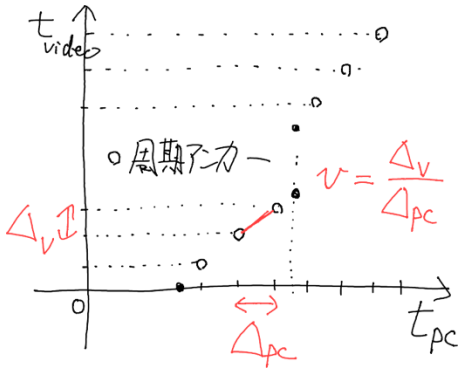
平均としては50msだが
±15%程度ずれる

プレイヤー側精度に課題

検証:ビデオ再生速度の計測精度

- 1倍速→1.5倍速→2倍速→一時停止





平均再生速度は追従

ビデオ再生位置取得方法

- 現在の実装
  - Paella Playerのasync currentTime()を呼び出し
  - JavaScriptで実装が完結
- 他の実装案
  - video要素のrequestVideoFrameCallback()を利用
    - フレーム描画毎にコールバック呼び出し(ex. 30fps)
    - 古いブラウザは対応していない
  - JavaScript側からvideo要素を見つける必要あり