SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering & Analytics

# Optimization of Video-based Facial Data Extraction for Time-series Correlation Analysis using Distributed-Data Processing Framework

Norma Puspitasari

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering & Analytics

# Optimization of Video-based Facial Data Extraction for Time-series Correlation Analysis using Distributed-Data Processing Framework

# Optimierung Videobasierter Gesichtsdatenextraktion zur Zeitreihenkorrelationsanalyse unter Verwendung eines Verteilten

# Datenverarbeitungssystems


| | |
|---|---|
| Author: | Norma Puspitasari |
| Supervisor: | Prof. Gudrun Klinker, PhD |
| Advisor: | Chloe Eghtebas |
| Submission Date: | 15.12.2023 |

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.12.2023 Norma Puspitasari

# Acknowledgments

I would like to express my deepest gratitude to my supervisor Prof. Dr. Gudrun Klinker for providing me with the opportunity to work on such an interesting topic. I also want to thank my advisor Chloe Eghtebas for her thoughtful advice and support. I am really grateful for her willingness to meet with her students regularly, her time to help me when it feels like a dead end and her insightful feedback. I am also grateful to my family and friends for their continuous support and love throughout my studies.

# Abstract

Facial extraction data plays a crucial role in synchrony measurement. There are open source tools available now for extracting Facial Action Units (AUs) data, such as OpenFace and PyFeat. However, it is still difficult to run a user study and have a real-time analysis of Facial Action Units (AUs) data at the same time because the tools are built either as a standalone application or a library for offline usage. Currently, we are building a video-conferencing platform that initiates the implementation of analysis filter for research purposes which still has a lot of rooms for improvement, such as low running fps, mixed frames and data lost. This thesis explores the optimization of facial extraction tool integration to the video-conferencing platform by utilizing OpenFace as a case study. The methodology involves implementing RabbitMQ as a robust queue system to manage and process facial data in real-time. This thesis also introduces a feature that enabled user to have the facial extraction data after running an experiment in the platform called post-processing. Finally, the real-time data was compared with the post-processing one and it showed that they have similar values. The network and memory measurement was also conducted and confirmed that it improved the real-time analysis filter performance.

# Kurzfassung

Gesichtsextraktionsdaten spielen eine entscheidende Rolle bei der Synchronitätsmessung. Es gibt inzwischen Open-Source-Tools für die Extraktion von Facial Action Units (FAU)-Daten, wie z. B. OpenFace und PyFeat. Es ist jedoch nach wie vor schwierig, eine Nutzerstudie durchzuführen und gleichzeitig eine Echtzeitanalyse von FAU-Daten (Facial Action Units) vorzunehmen gleichzeitig durchzuführen, da die Tools entweder als eigenständige Anwendung oder als Bibliothek für die Offline-Nutzung. Derzeit entwickeln wir eine Videokonferenzplattform, die die die Implementierung von Analysefiltern für Forschungszwecke, die noch viel Raum für Verbesserungen bietet, wie z.B. niedrige Bildwiederholrate, gemischte Bilder und Datenverluste. Diese Arbeit untersucht die Optimierung der Integration des Gesichtsextraktionstools in die Videokonferenzplattform anhand von OpenFace als Fallstudie. Die Methodik beinhaltet die Implementierung von RabbitMQ als robustes Warteschlangensystem zur Verwaltung und Verarbeitung von Gesichtsdaten in Echtzeit. Diese Arbeit führt auch eine Funktion ein, die eine Funktion vor, die es dem Benutzer ermöglicht, die Gesichtsextraktionsdaten nach der Durchführung eines Experiments in der Plattform zu erhalten, die so genannte Nachbearbeitung. Schließlich wurden die Echtzeitdaten mit den nachbearbeiteten Daten verglichen und es zeigte sich, dass sie ähnliche Werte aufweisen. Die Netzwerk- und Speichermessungen wurden ebenfalls durchgeführt und bestätigten, dass die Leistung des Echtzeit-Analysefilters verbessert wurde.

# Contents

# 1 Introduction

Video-conferencing platforms have become more than just communication channels. They are also used for conducting user studies and experiments for research and educational purposes. Due to the easy accessability of having the video data from these platforms and the surging of ongoing research about facial extraction technique, Facial Action Units (AUs) data are now possible to obtain by utilizing open source tools, such as OpenFace [1] and PyFeat [2]. There are several interesting applications that will be greatly benefited from the extracted AUs. One of them is synchrony measurement, which lies in both computer science and psychology area.

Obtaining AUs data is pretty straightforward for people with technology background. However, it is not that easy to get them running for non-tech people. Let alone the time for understanding and installing the tool, the extraction process itself also takes quite sometime. The integration of the facial extraction tools into a video-conferencing platform is motivated by a desire to fast-forward that long process and have everyone in any research background could get the AUs data with less hassle. The first step of this integration is already implemented on a video-conferencing platform called experimental-hub [**experimental-hub**], using OpenFace as the facial extraction tool that is run as a real-time filter on the video stream.

As a representative case study, OpenFace serves as a powerful tool in this field because of its capability to extract variety of facial data from image sequences or video. OpenFace is built as a standalone toolbox which consists of multiple facial and landmark detection technology inside so integrating them with the experimental hub is beneficial but also gave us a lot of drawbacks. Running a machine learning project has always been a heavy-computation task for a machine, which OpenFace does the same. Moreover, if we want to apply this real-time filter, we need to run it in multiple instances simultaneously for each user in the experimental hub. It is very desirable to have it running without slowing down the video stream and not losing the frame-by-frame AUs data of the video. Thus, this thesis aims to minimize data lost and run the video stream in a normal fps, so we optimized the existing real-time AUs filter and have the post-processing feature as a backup plan. The case study with OpenFace, coupled with the queue system facilitated by RabbitMQ for the asnychronous exchanging frames process. (add more sentences on why we built the post-processing)

# 2 Related Work

There are four topics that are related to this research. First, I conducted research on the variety of video filters and how they are implemented in the existing video-conferencing platforms. we then explored facial extraction technology options that are available. The next section tells us about how we measure synchrony using facial extraction data. Last section provides an overview of the effectiveness of queue system for real-time data communications.

## 2.1 Video Filters in Video-Conferencing Platforms

Video filters are created for a wide-range of user needs, such as beautification like smoothing the skin, privacy like blurring the background or facial recognition for observing user emotions. By looking at these purposes, we then divided the video filters into two categories:

### 2.1.1 Manipulative Filter

Manipulative filters are used for manipulating facial features, environments or applying augmented reality elements onto the video stream. These kind of filters are quite well-known from social media users. They use them for gaining self-acceptance and engagement with their online audiences [**ar-filter-on-social-media**]. Professional and privacy concerns are also raised during video calls in a virtual environment so blur background filter are built to make the users feel secure of their surroundings [**blur-privacy-ar**]. There are ton options of manipulative filters that augment the video streams using visual effects or simply just changing the video's color, even we are actually possible to implement a customized filter, e.g. applying AR filter on top of another filter.

### 2.1.2 Analysis Filter

Analysis filters are mostly related to facial and landmark detection of users and used for research and educational purposes. These filters are rarely found in commercial video-conferencing platforms, like Zoom, Skype, etc. On the other side, open source

tools provide analysis filter that can detect faces and landmarks but we can't do it within the video-conferencing platform. Most of them are standalone toolbox that we need to run separately [**tracked**] [1].

## 2.2 Facial Extraction Technology

Face recognition has gained a lot of attentions in research and business world. Many practical applications could benefit from it, such as identification, access control and human-computer interactions which increase the availability of the relevant technologies. tbc.

## 2.3 Queue Systems

explain the use of queue systems for real time data applications. list down the available queue systems. why we use rabbitmq instead of other tools.

## 2.4 Synchrony Measurement

should I add this part?

# 3 Requirements Evaluation

In this chapter, I define the

## 3.1 Technical Requirements

- fps before and after using queue system (should be higher, because no response time is needed) - memory usage - network traffic - total frames acquired

## 3.2 Data Validation

- real-time openface has similar values as post-processing one

# 4 Implementation

## 4.1 Section

# 5 Method

## 5.1 Section

# 6 Results

## 6.1 Section

# 7 Discussion

## 7.1 Section

Citation test [1].

Acronyms must be added in `main.tex` and are referenced using macros. The first occurrence is automatically replaced with the long version of the acronym, while all subsequent usages use the abbreviation.

E.g. `\ac{TUM}`, `\ac{TUM}` ⇒ **TUM!** (**TUM!**), **TUM!**

For more details, see the documentation of the `acronym` package[1].

### 7.1.1 Subsection

See **??**, **??**, **??**, **??**.

Table 7.1: An example for a simple table.

| A | B | C | D |
|---|---|---|---|
| 1 | 2 | 1 | 2 |
| 2 | 3 | 2 | 3 |

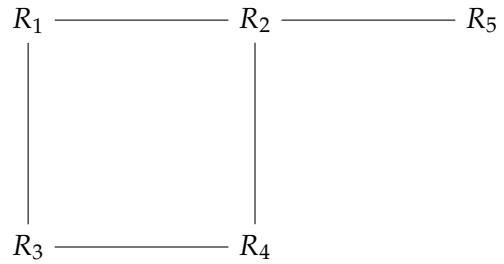


Figure 7.1: An example for a simple drawing.

[1] `https://ctan.org/pkg/acronym`

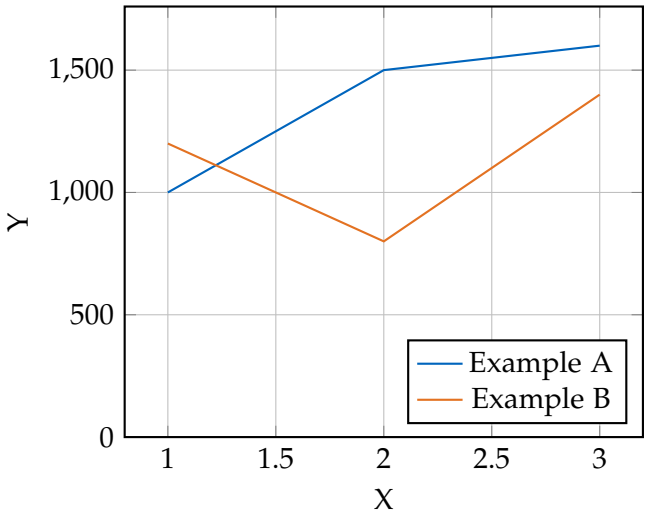


Figure 7.2: An example for a simple plot.

```
SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 7.3: An example for a source code listing.

# Abbreviations

# List of Figures

# List of Tables

# Bibliography

[1] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L.-P. Morency. "OpenFace 2.0: Facial Behavior Analysis Toolkit." In: *2018 13th IEEE International Conference on Automatic Face  Gesture Recognition (FG 2018)*. 2018, pp. 59–66. DOI: 10.1109/FG.2018.00019.

[2] J. H. Cheong, T. Xie, S. Byrne, and L. J. Chang. "Py-Feat: Python Facial Expression Analysis Toolbox." In: *CoRR* abs/2104.03509 (2021). arXiv: 2104.03509.

[3] L. Lamport. *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*. Addison-Wesley Professional, 1994.