Westfälische Wilhelms-Universität Münster

Praktikum zur Mustererkennung

Emotionserkennung

Klassifikation von Action Units anhand von Landmarks

Robin Rexeisen
Matrikelnr. 123456

Johannes Stricker
johannesstricker@gmx.net
Matrikelnr. 383779

Alexander Schlüter
alx.schlueter@gmail.com
Matrikelnr. 409649

Betreuer:
Sören Klemm
soeren.klemm@wwu.de

eingereicht am 16. September 2016

Fachbereich 10
Mathematik und
Informatik

Inhaltsverzeichnis

1. Einleitung

Das Thema der (visuellen) Emotionserkennung durch Computersysteme hat in den letzten Jahren immer mehr an Bedeutung gewonnen. Die Einsatzgebiete sind vielseitig und reichen von Sicherheitsanwendungen, über Robotik, bis hin zu Unterhaltungsmedien. Meist wird versucht, anhand von verschiedenen Merkmalen im Gesicht, diesem eine oder mehrere Emotionen zuzuordnen. Im Rahmen unseres Praktikums war es unsere Aufgabe, ein solches Computersystem zur Erkennung von Emotionen zu entwickeln.

Im folgenden Kapitel werden die Aufgabenstellung sowie die Eingabedaten genauer beschrieben. In Kapitel 2 werden daraufhin die Methodiken vorgestellt, die wir für unser Programm nutzen, woraufhin in Kapitel 3 erläutert wird, wie wir diese implementiert haben. Es folgt die Vorstellung unserer Ergebnisse in Kapitel 4, die wir anschließend in Kapitel 5 diskutieren. Im letzten und 6. Kapitel wird das Ergebnis der Arbeit kurz resümiert.

1.1. Facial Action Code

Der Facial Action Code (kurz FAC) ist ein System zur Unterscheidung von Bewegungen von isolierten Teilen des menschlichen Gesichts, welches 1976 von Paul Ekman und Wallace V. Friesen entwickelt wurde. Es basiert auf sogenannten Action Units (kurz AU), welche eben genau diese Bewegungen beschreiben sollen. Dabei kann eine Action Unit eine Ausprägung zwischen einschließlich 0 und 5 haben, wobei 0 bedeutet, dass keine entsprechende Bewegung vorhanden ist und 5 bedeutet, dass die Bewegung maximal stark ausgeprägt ist [3]. Eine Auflistung der für diese Arbeit relevanten Action Units findet sich im Anhang A.1.

1.2. DISFA Datenbank

Die Denver Intensity of Spontaneous Facial Action Database (kurz DISFA Database) enthält eine Sammlung von Gesichtsbewegungen von insgesamt 27 unterschiedlichen, erwachsenen Probanden. Hierzu wurde von jedem Probanden ein 4-minütiges Video mit je 20 Frames pro Sekunde gedreht. Danach wurde jedes Frame nach dem Facial Action Coding System auf die Ausprägung von 12 Action Units analysiert und gelabelled. Weiterhin enthält jedes Frame 66 Landmark Koordinaten, von markanten Punkten des Gesichtes.

1.3. Die Aufgabenstellung

Die Aufgabenstellung des Praktikums bestand darin, aus einer Auswahl von 10 Videos der DISFA Datenbank einen Klassifikator zu entwickeln, der möglichst präzise in der Lage ist für ein beliebiges Frame aus der Datenbank zu bestimmen, welche Action Units in dem Frame aktiviert sind.

2. Methodik

In diesem Kapitel werden sowohl die von uns verwendeten Methoden zum Training der Klassifikatoren und zum Klassifizieren, als auch die von uns verwendeten Klassifikatoren selbst genauer beschrieben. Außerdem wird erläutert wie die Ergebnisse evaluiert wurden.

2.1. Ground-Truth Daten

Obwohl eine Action Unit eine Ausprägung I zwischen 0 und 5 besitzen kann, wollen wir - um die spätere Klassifikation zu erleichtern - das Problem vereinfachen, indem wir nur in zwei Kategorien klassifizieren: „aktiv“ und „nicht aktiv“. Dazu setzen wir ein Schwellenwert $0 < h \leq 5$ ein. Falls $I < h$, so gehen wir von keiner Aktivierung aus. Falls $h \leq I$, so wollen wir davon ausgehen, dass die Action Unit aktiv ist.

2.2. Vorverarbeitung

2.2.1. Aufbereitung der Eingabedaten

Wie bereits in der Einleitung erwähnt, handelt es sich bei den Eingabedaten um 10 Videos, die der DISFA Datenbank entnommen sind. Die Videos wurden nacheinander aufgenommen und zeigen verschiedene Probanden. Bedingt dadurch sind die Landmarks in den Videos nicht identisch bezüglich Skalierung, Rotation und Position.
Damit die Klassifikation durch diese Störungen nicht beeinträchtigt wird werden die Eingabedaten zunächst normalisiert. Dies geschieht in drei Schritten.

1. Die Landmarks werden um den Koordinaten-Ursprung zentriert. Hierzu berechnen wir einen Vektor vom Mittelpunkt aller Landmarks zum Ursprung und translatieren die gesamte Punktwolke um diesen Vektor.

2. Daraufhin wird die Punktwolke so skaliert, dass die maximale horizontale Distanz alle Landmarks genau 1 beträgt. Hierzu berechen wir diese maximale Distanz und teilen alle Koordinaten der Landmarks durch diese.

3. Um Störungen durch Drehung des Kopfes der Probanden auszugleichen normalisieren wir ebenfalls die Rotation der Punktwolke. Dazu ziehen wir eine Linie zwischen den beiden Augen des Probanden und berechnen den Winkel zur x-Achse. Dasselbe tun wir mit einer Linie zwischen zwei gegenüberliegenden Landmarks an den Gesichtsrändern. Schließlich rotieren wir die gesamte Punktwolke um den Mittelwert der beiden Winkel in entgegengesetzter Richtung (siehe Abbildung 2.1).

Ein weiteres Problem der Eingabedaten besteht darin, dass die Anzahl von true-positives gering ist. True-positives sind die Frames, wo die betrachtete Action Unit aktiv ist. Um mehr Daten zu erzeugen und so auch mehr true-positives, erweitern wir die vorhandenen Daten, indem alle Frames dupliziert und die Landmarks in diesen Frames durch eine leichte, normalverteilte Störung verschoben werden.

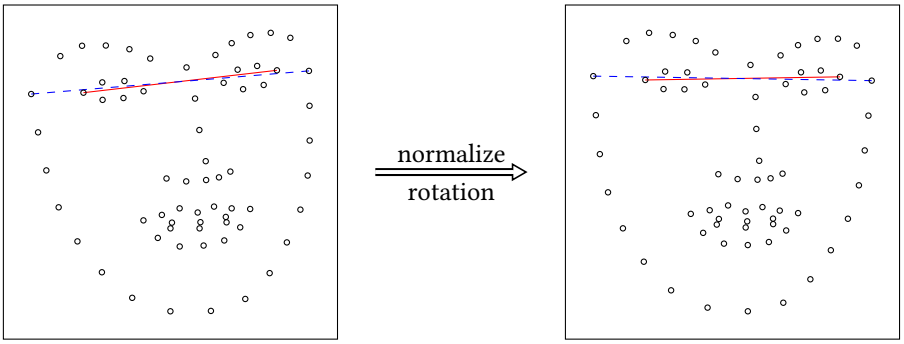

Abbildung 2.1.: Normalisierung der Rotation durch einen Vektor zwischen den Augen des Probanden

2.2.2. Feature Extraction

Bei der (visuellen) Emotionserkennung wird versucht anhand von einem oder mehreren, verschiedenen Merkmalen (engl. Features) einem Gesicht eine oder mehrere Emotionen zuzuordnen. Je mehr Aussagekraft die Kombination dieser Merkmale über die jeweiligen Emotionen haben, desto besser können diese klassifiziert werden. Die Aussagekraft ist aber a priori nicht bekannt. Deshalb extrahieren wir aus den Eingabedaten, also den Videos mit je 66 Landmarks pro Frame, verschiedene Features, um sie in verschiedenen Kombinationen miteinander zu testen. Es folgt eine Beschreibung der von uns verwendeten Features. Die Methode, wie diese erzeugt werden, werden wir im Folgenden als Feature-Extraction bezeichnen.

Statische Features

Mit statischen Features bezeichnen wir solche, die mit Hilfe der Landmarks von genau einem Frame erzeugt werden.

1. **X-/Y-Koordinaten (XY)**: die Koordinaten der Landmarks werden als Merkmale genutzt. Da in der Menge der Koordinaten sowohl Informationen über die individuellen Punkte liegen, als auch Informationen über ihre Relation zueinander, ist es sinnvoll dieses Feature zu testen.
2. **Paarweise Orientierung (Orientation)**: es werden jeweils alle Paare von je zwei unterschiedlichen Landmarks betrachtet und die Rotation des Vektors zwischen den beiden Punkten als Merkmal genutzt. Weil sich bei verschiedenen Mimiken meist die Position markanter Punkte im Gesicht zueinander ändert, erscheint es sinnvoll Features zu nutzen, die die Landmarks untereinander explizit in Relation setzen.
3. **Paarweise Euklidische Distanz (EuclidianDist.)**: auch hier werden jeweils alle Paare unterschiedlicher Landmarks betrachtet und die euklidische Distanz zwischen den beiden Punkten als Merkmal benutzt. Dieses Feature erscheint ebenfalls sinnvoll, weil es Informationen über die Relation von Landmarks untereinander hat.
4. **Orientierung relativ zum Mittelpunkt der Landmarks (CenterOrient.)**: bei diesem Feature wird die Orientierung jedes Landmarks relativ zum Mittelpunkt aller Landmarks betrachtet, das heißt es wird die Rotation des Vektors zwischen Mittelpunkt und Landmarks als Merkmal genutzt. Dieses Feature enthält Informationen darüber, wie die Position der Landmarks relativ zum gesamten Gesicht ist. Dies erscheint für viele Gesichtsausdrücke sinnvoll.
5. **Euklidische Distanz zum Mittelpunkt der Landmarks (CenterDist.)**: dieses Feature betrachtet die euklidische Distanz jedes Landmarks zum Mittelpunkt aller Landmarks. Dieses Feature sagt ebenfalls etwas über die Relation der einzelnen Landmarks zum gesamten Gesicht aus.

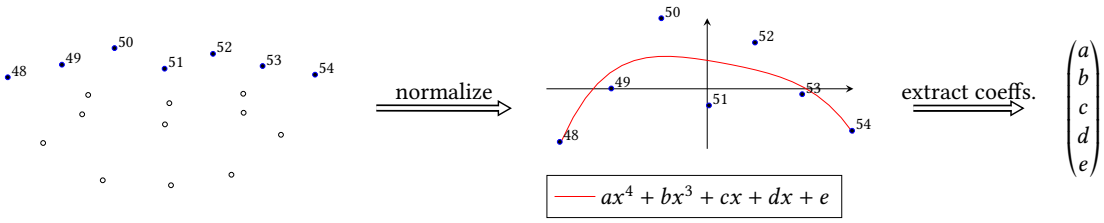

Abbildung 2.2.: Die Koeffizienten eines interpolierenden Polynoms als Feature

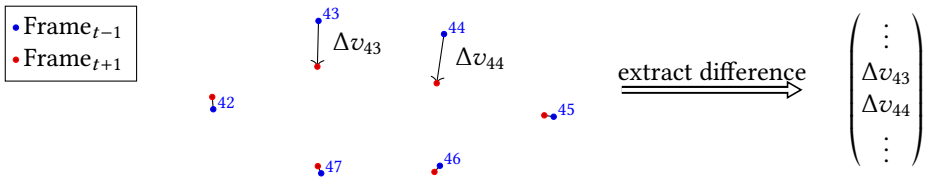

Abbildung 2.3.: Zeitliche Ableitung der Landmarks als Feature

6. **Polynominterpolation (Interpolation)**: hierbei werden verschiedenen Bereiche eines Gesichts einzeln betrachtet (z.B. die Oberlippe, siehe Abbildung 2.2). Diese Landmarks werden zuerst normiert. Dazu verschiebt man die dortigen Punkte, so dass der Mittelpunkt der Koordinaten-Ursprung ist. Danach werden sie so skaliert, dass die maximale Entfernung in x-Richtung gleich 1 ist. Nach dieser Normierung werden diese Punkte durch ein Polynom interpoliert. Hierzu wird die Methode der kleinsten Quadrate angewendet. Die dadurch berechneten Koeffizienten des Polynoms werden nun verwendet. Sie ergeben nämlich mit den Polynomkoeffizienten der anderen Gesichtsbereiche die Features.
   Dieses Verfahren hat folgende Vorteile: Erstens fließt ein eventuelles Rauschen durch Interpolation anhand mehrerer Punkte weniger in die resultierenden Features mit ein. Und zweitens wird der Feature-Raum verringert, da mehrere Punkte durch wenige Koeffizienten repräsentiert werden.

Zeitliche Features

Bisher haben wir den zeitlichen Aspekt bei der Berechnung der Features außer Acht gelassen. Die oben aufgelisteten statischen Features werden nur anhand eines Frames berechnet. Man kann jedoch eine statische Featureberechnung um einen zeitlichen Aspekt erweitern, indem man dessen Veränderung über die Zeit als Feature benutzt (diese Methode bezeichnen wir als TimeDiff.). Um die zeitliche Ableitung zu einem bestimmten Zeitpunkt zu berechnen, wird numerische Differentiation benutzt. Es bezeichne $t \in \mathbb{N}$ den aktuellen Zeitpunkt und $f(t) \in \mathbb{R}^d,\ d \in \mathbb{N} \setminus \{0\}$ den Feature-Vektor eines statischen Features. Nun ergibt sich das zeitliche Feature durch die numerische Ableitung $\frac{f(t+\delta)-f(t-\delta)}{2}$. Üblicherweise nimmt man $\delta = 1$ an, doch da sich eventuell ein aussagekräftigeres Resultat ergibt, wenn man einen größeren zeitlichen Abstand benutzt, wollen wir hier $\delta \in \mathbb{N} \setminus \{0\}$ beliebig voraussetzen.

Als Beispiel nehmen wir die zeitliche Ableitung der X-/Y-Koordinaten (siehe dazu Abbildung 2.3). Hier werden zum Zeitpunkt $t+1$ und $t-1$ die Koordinaten voneinander subtrahiert und halbiert, zum Beispiel $\Delta v_{43} := \frac{v_{43}(t+1)-v_{43}(t-1)}{2} \in \mathbb{R}^2$ (wobei $v_j(t)$ die Koordinate des Punktes $j \in \{0, \ldots, 65\}$ zum Zeitpunkt t ist). Diese sich so ergebene Differenz dient nun als neues Feature.

Nun können jedoch nicht mehr die Intensitäten der Action Units direkt als Klassifikationsziel genutzt werden. Um das zu veranschaulichen, gehen wir von einem Zeitraum aus, während dem eine Action Unit aktiv ist, aber ihre Intensität sich nicht ändert. Da keine Änderung passiert, wird sich auch in den errechneten Features kaum etwas ändern. Die zeitliche Ableitung wird also nahe 0 sein, müsste aber als aktiv klassifiziert

werden. Die Ableitung ist aber auch nahe 0, wenn sich kaum etwas ändert und die Action Unit nicht aktiv ist. Um dieses Problem zu lösen, betrachtet man ebenfalls die zeitliche Differenz der Intensität der Action Unit. Wir subtrahieren hierfür die Intensitäten zu den Zeitpunkten $t + 1$ und $t - 1$. Falls die absolute zeitliche Differenz einen Schwellenwert überschreitet, gehen wir von einer Aktivierung aus. Somit bezeichnen wir nun nur das Abklingen bzw. Aufkommen einer Action Unit als aktiv.

Featureverarbeitung

Mit den bis hierher extrahierten Features wäre es möglich einen Klassifikator zu trainieren. Dies kann jedoch effizienter geschehen, wenn die Features zuvor noch einmal vorverarbeitet werden. Wir haben die Features folgendermaßen verarbeitet:

- **Normalisierung**: Obwohl unsere Eingabedaten bereits normalisiert sind, ist unter Umständen eine erneute Normalisierung der Features notwendig. Die extrahierten Features haben unterschiedliche Wertebereiche. Bei den Orientierungsfeatures können beispielsweise Werte zwischen 0 und 360 angenommen werden. Die Distanzfeatures hingegeben, liegen ungefähr im Bereich von 0 bis 1. Damit der Klassifikator dadurch nicht beeinträchtigt wird, ist es wichtig die Features durch erneute Normalisierung in einen einheitlichen Wertebereich zu projezieren. Dazu werden die Mittelwerte der Features auf 0 und die Varianzen auf 1 gesetzt.
- **Shufflen**: Aufgrund der Struktur der Videos sind die Frames mit aktiver Action zeitlich nahe bei einander. Zwischen den Beginn und Ende einer Action ist sie die ganze Zeit aktiv. Dies ist jedoch eine Struktur, die der Klassifikator nach Möglichkeit nicht lernen soll. Darum kann man die Features aus mehreren Frame mischen, damit diese zeitliche lokale Eigenschaft nicht mehr zur Geltung kommt.
- **PCA**: Da die Anzahl der extrahierten Features teilweise sehr hoch ist (zum Beispiel 2145 Features für die paarweise Extraktion alleine), scheint es sinnvoll die Dimension der Features durch eine Principal Component Analysis (PCA) zu reduzieren. Dadurch wird die benötigte Trainingszeit zum Teil enorm verringert.

Klassifikatoren

Wir nutzen zwei verschiedene Klassifikatoren. Zum einen eine Support Vektor Maschine (SVM) und zum anderen einen Random Forest (RF). Diese können mit verschiedenen Parametern trainiert werden. Auch in diesem Fall sind die optimalen Parameter zuvor allerdings unbekannt, weshalb verschiedene Kombination dieser getestet werden müssen.
Für die SVM haben wir verschiedene Kernelfunktionen mit unterschiedlichen Werten für θ und den Polynomgrad d angewandt:

- Linearer Kernel: $K(x_i, x_j) = x_i^T x_j$
- Polynomieller Kernel: $K(x_i, x_j) = (y x_i^T x_j + \theta)^d$
- Radial Basis Kernel: $K(x_i, x_j) = e^{-y\|x_i - x_j\|^2}$
- Sigmoid Kernel: $K(x_i, x_j) = \tanh(y x_i^T x_j + \theta)$

2.3. Training und Evaluierung

2.3.1. Aufteilung der Mengen

Die im vorherigen Abschnitt beschrieben Methoden zur Feature Extraction, Verarbeitung und Klassifikation sollen in verschiedenen Kombinationen evaluiert werden. Der erste Datensatz aus 10 Personen wird dazu aufgeteilt in 60% Trainingsmenge und 40% Validierungsmenge. Hier ist die Entscheidung zu treffen, wie die Personen auf die Mengen aufgeteilt werden:

1. Erst die Frames durchmischen, dann aufteilen: Dies ist sinnvoll, wenn der Klassifikator nur verwendet werden soll, um Action Units in neuen Frames von schon bekannten Personen zu erkennen. Es wird nicht getestet, wie gut der Klassifikator auf neue Personen generalisiert, da Frames von allen Personen zum Training verwendet werden.
2. 6 Personen nur im Training, 4 nur in der Validierung verwenden: Die Performance auf der Validierungsmenge ist repräsentativ dafür, wie gut der Klassifikator Action Units bei bisher unbekannten Personen erkennt.

Wir haben uns für Methode 2 entschieden, weil die Generalisierung auf neue Personen das interessantere Problem ist: In Anwendungsfällen ist es wünschenswert, für neue Personen nicht erst mehrere tausend Frames manuell labeln zu müssen, um den Klassifikator auf dieser Person zu trainieren.

Wir haben eine zweite Menge an Daten von 5 Personen, die als Testmenge dient.

2.3.2. Performanzberechnung

Aufgrund der geringen Anzahl positiver Samples (Frames, in denen die Action Unit aktiviert ist), ist die Accuracy keine zuverlässige Statistik. Ein Klassifikator, der die Action Unit immer als „nicht aktiv“ klassifiziert, könnte sehr hohe Accuracy erreichen, ohne tatsächlich etwas über die Action Unit gelernt zu haben. Stattdessen evaluieren wir die Klassifikatoren anhand von

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \qquad \text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Hierbei ist TP die Anzahl von true-positives, FP die Anzahl false-postitives und FN die Anzahl false-negatives. Der F1 score ist das harmonische Mittel zwischen Precision und Recall. Da wir von der Aufgabenstellung her keine Präferenz für hohe Precision / hohen Recall haben, nutzen wir den F1 score als erste Zahl zum Vergleich der Klassifikatoren.

2.3.3. Zustandekommen der Ergebnisse

Wir wollen hier nochmal zusammenfassen, wie unsere Ergebnisse zustande kommen: Zuerst wird die Trainingsmenge und Validierungsmenge gebildet. Die Videos der Trainingsmenge werden durch Rauschen künstlich erweitert und bei beiden Mengen die Landmarks normalisiert. Danach werden die Features aus den resultierenden Landmarks extrahiert. Es folgt eine Normalisierung der Features anhand ihres Mittelwerts und ihrer Varianz. Für eine Reduktion des Feature-Raumes auf eine verbleibenden Varianz von 97% wurde dann der PCA-Algorithmus angewendet. Eine Ausnahme ist hierbei die Interpolation: Da dieses Feature schon den Raum verkleinert hat, wurde auf eine weitere Reduktion durch PCA verzichtet. Schlussendlich wurden die Features für das Training dann zufällig geshuffled. Für zeitlichen Features haben wir uns nur auf X-/Y-Koordinaten und die Interpolationskoeffizienten konzentriert. Für die Ableitung wurde $\delta = 1$ gesetzt.

Wir lernen dann mehrere SVM mit verschiedenen Kernelfunktionen und mehrere RF mit variierender Anzahl an Bäumen und maximaler Tiefe für jede der genannten Feature Extractions. Wir lernen auf der Trainingsmenge und versuchen die Qualität des Klassifikator-Feature Paares für jede Action Unit anhand der Validierungsmenge und des dortigen F1 scores zu beurteilen. Da pro Action Unit ca. 160 Kombinationen evaluiert werden, kann es durch diese Auswahl der besten fünf zu einem Overfitting gegen die Validierungsmenge kommen. Um realistische Zahlen für die Performance zu bekommen, werden deshalb die besten fünf nochmal auf der Testmenge evaluiert. Diese besteht aus fünf bisher unbekannten Personen aus einem zweiten Datensatz.

3. Implementierung

Das Projekt wurde in C++11 umgesetzt. Für die Erstellung einer Benutzeroberfläche wurde die QT5 Bibliothek verwendet [4]. Für verschiedene Algorithmen aus der Mustererkennung wurde die OpenCV2 Bibliothek genutzt [1]. Außerdem verwenden wir CMake als plattformunabhängiges Build-System [2]. Für das Projekt haben wir zwei separate Anwendungen erstellt, den QViewer und das Auto-Train Programm.

Die im nächsten Kapitel vorgestellten Ergebnisse wurden am Computer mit dem Betriebssystem Archlinux erstellt, der den Linux-Kernel in Version 4.7.3 verwendet. Weiter wurde OpenCV in der Version 2.4.13, der C++ Compiler GCC in Version 6.2, CMake in der Version 3.6.2 und Qt in der Version 5.7 benutzt. Der Rechner besitzt einen 4-Kern Prozessor von AMD mit jeweils 2.8 GHz an Kapazität und einen Arbeitsspeicher von 8 GB.

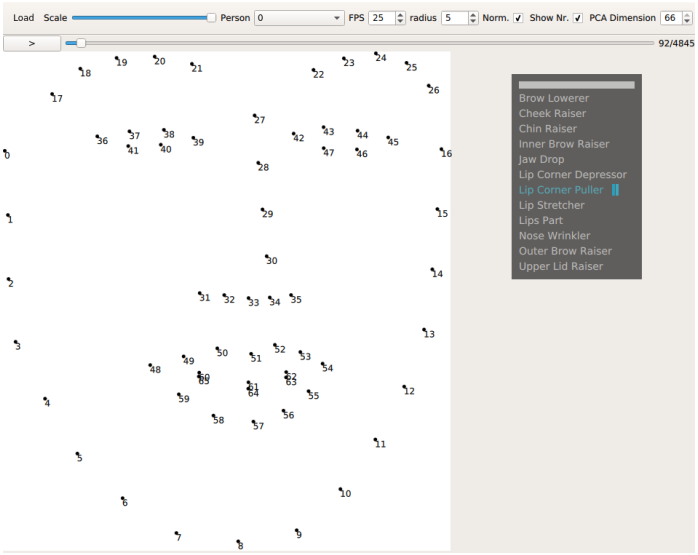

Abbildung 3.1.: Screenshot der QViewer Anwendung

3.1. QViewer

Der QViewer ermöglicht uns eine Visualisierung der Eingabedaten. Es ist eine in Qt geschrieben grafische Oberfläche, in der die Landmark und Action Unit Dateien geladen werden können. Ein Nutzer kann dann eine Person auswählen und dessen Landmarks als Videos abspielen. Dabei werden dann die Intensitäten der Action Units angezeigt. Auf Abbildung 3.1 ist die Oberfläche des Programmes zu erkennen. Zusätzlich können wir die Landmarks normalisieren. Weiter kann man durch PCA die Punkte auf einen neuen Raum mit weniger Dimensionen projezieren und dann wieder zurückprojezieren. Dadurch ist es möglich den Effekt von PCA zu veranschaulichen. Der QViewer ist also insgesamt eine Hilfe um Landmarks, Action-Unit und einige hier beschriebenen Verfahren zu verstehen.

3.2. Auto-Train

Wie bereits in Kapitel 2 erwähnt, ist ein guter Klassifikator abhängig von der Wahl verschiedener Variablen, wie zum Beispiel von der Auswahl der Features oder von den Parametern mit denen der Klassifikator trainiert wird.

Wünschenswert wäre ein Programm, das automatisch alle möglichen Kombinationen mitsamt der Vorverarbeitung und Evaluierung vornimmt und dabei alle nötigen Informationen, wie die Eigenvektoren bei PCA oder die gelernten Klassifikatoren, automatisch speichert.

Hier kommt das Programm Auto-Train ins Spiel. Das Programm liest aus einer JSON-Datei, wie sie im Anhang A.2 zu finden ist, die gewünschte Konfiguration. Dazu zählen unter anderem die Schritte zur Vorverarbeitung der Landmarks und der extrahierten Features, als auch die zu verwendeten Feature-Extractoren.

Zusätzlich zum Training kann das Programm, mithilfe einer etwas anderen JSON-Datei, dann gelernte Klassifikatoren und Parameter (wie z.B. Eigenvektoren der PCA) verwenden und die Performanz für eine Testmenge berechnen. Der Einfachheit halber, wird die dafür benötigte JSON-Datei beim Training automatisch erstellt. Die Datei bedarf lediglich eines manuellen Eintrages für die zu evaluierenden Daten.

4. Ergebnis

Die hier vorgestellten Ergebnisse wurden mit Hilfe von Auto-Train und der JSON Konfiguration im Anhang A.2 berechnet. Das Training der Action Units Lip Stretcher und Jaw Drop wurde konnte im vorgegebenen Zeitrahmen nicht beendet und musste vorzeitig abgebrochen werden.

AU	Bester Klassifikator					
	F1 Val	F1 Test	Prec. Test	Recall Test	Klassifikator	Features
Lips Part	0.656	0.575	0.598	0.554	RF	CenterDist.
Upper Lid Raiser	0.394	–	0	0	SVM Polyn.	Interpolation
Outer Brow Raiser	0.391	–	0	0	SVM Polyn.	EuclidianDist.
Lip Corner Puller	0.37	0.431	0.317	0.674	SVM Lin.	XY
Cheek Raiser	0.277	0.18	0.102	0.769	SVM Polyn.	XY
Nose Wrinkler	0.242	0.03	0.015	0.62	SVM Polyn.	EuclidianDist.
Inner Brow Raiser	0.2	0.084	0.064	0.125	SVM Polyn.	EuclidianDist.
Chin Raiser	0.191	0.03	0.163	0.017	SVM Polyn.	CenterDist.
Brow Lowerer	0.163	0.2	0.659	0.118	RF	EuclidianDist.
Lip Corner Depressor	0.022	–	0	0	SVM Polyn.	XY

Tabelle 4.1.: F1 scores und Testergebnisse des besten Klassifikators pro Action Unit

In Tabelle 4.1 sind die Ergebnisse des besten Klassifikators (ausgewählt nach F1 score auf der Validierungsmenge) pro Action Unit zu sehen. Gute Klassifikation auf unbekannten Personen ist möglich für Lips Part: Der hohe F1 score 0.656 in der Validierung bestätigt sich auch auf der Testmenge. Akzeptable Performance liefert der beste Klassifikator für Lip Corner Puller. Dieser verbessert sich sogar von einem F1 score von 0.37 in der Validierung auf 0.431 im Test.

Die Klassifikatoren für Outer Brow Raiser und Upper Lid Raiser scheinen in der Validierung akzeptabel, erkennen jedoch im Test überhaupt keine Aktivierung der Action Units mehr. Die übrigen Action Units werden mit keiner unserer Feature Extraction-Methoden an neuen Personen befriedigend klassifiziert.

Abb. 4.1 zeigt, dass die verschiedenen Kombinationen aus Feature Extraction, Klassifikator und Parametern zu stark variierender Performance auf der Validierungsmenge führen. Der Tradeoff zwischen Precision und Recall ist deutlich zu sehen. Man sieht eine Trennung zwischen SVM und Random Forest Klassifikatoren: erstere tendieren dazu, zu viele negative Frames (ohne Aktivierung der Action Unit) als positiv zu klassifizieren, was zu schlechter Precision führt. Die Random Forests neigen hingegen zu vielen false-negatives. Sie schneiden bezogen auf Lips Part besser ab, bei anderen Action Units ist die Performance zwischen den Klassifikatoren ausgeglichen. Aus Platzgründen werden hier nur Graphen für ausgewählte Action Units gezeigt. Unsere gesamten Ergebnisse sind aber auf einer von uns erstellten Internetseite veröffentlicht[1].

Die Dominanz der Random Forests für Lips Part ist auch in Tabelle 4.3 zu sehen. Die beste SVM taucht mit einem F1 score von 0.356 in den Top 5 nicht mehr auf. Lip Corner Puller wird dagegen von einer SVM am besten erkannt, allerdings dominieren wieder Random Forests in den Top 5.

[1] URL: http://alexschlueter.github.io/detect-emotion

.

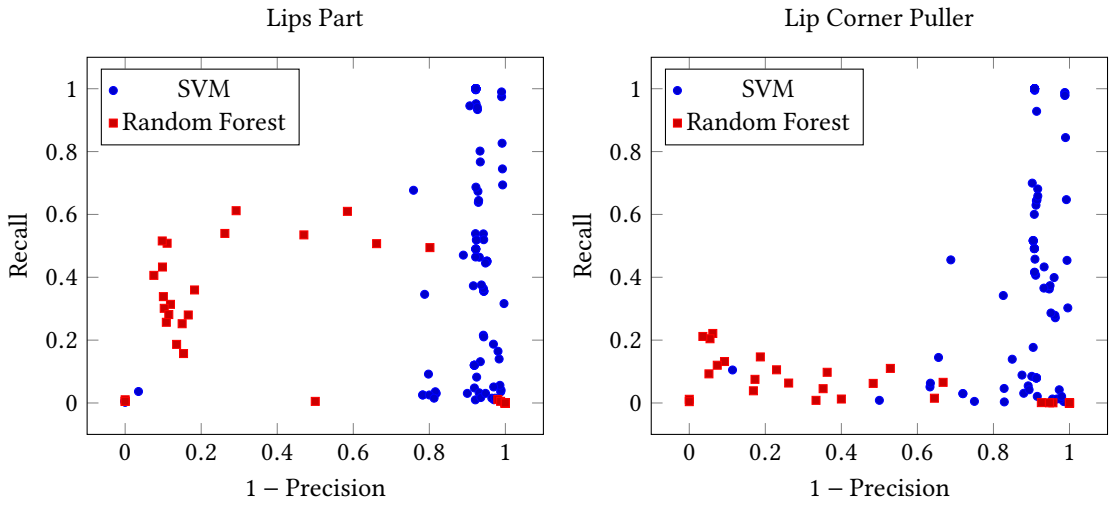

Abbildung 4.1.: Validierungsergebnisse für Lips Part und Lip Corner Puller. Jeder Punkt steht für eine Kombination aus Feature Extraction, Klassifikator und Parametern.

F1 Val	F1 Test	Prec. Test	Recall Test	Klassif.	Parameter	Features
0.656	0.575	0.598	0.554	RF	#trees = 20, maxDepth = 20	CenterDist.
0.656	0.594	0.639	0.555	RF	#trees = 50, maxDepth = 20	CenterDist.
0.647	0.655	0.809	0.550	RF	#trees = 20, maxDepth = 10	CenterDist.
0.623	0.616	0.966	0.452	RF	#trees = 50, maxDepth = 10	Interpolation
0.585	0.732	0.752	0.713	RF	#trees = 20, maxDepth = 20	XY

Tabelle 4.2.: F1 scores und Testergebnisse der Top 5 Klassifikatoren für Lips Part

F1 Val	F1 Test	Prec. Test	Recall Test	Klassif.	Parameter	Features
0.370	0.431	0.317	0.674	SVM Lin.	–	XY
0.358	0.514	0.675	0.415	RF	#trees = 10, maxDepth = 4	Interpolation
0.347	0.389	0.752	0.262	RF	#trees = 50, maxDepth = 20	Interpolation
0.336	0.536	0.733	0.423	RF	#trees = 20, maxDepth = 10	Interpolation
0.248	0.426	0.774	0.293	RF	#trees = 20, maxDepth = 10	EuclidianDist.

Tabelle 4.3.: F1 scores und Testergebnisse der Top 5 Klassifikatoren für Lip Corner Puller

Features	Bester F1 score	Dimensionen	Dim. nach PCA
CenterDistance	0.656	66	17
Interpolation	0.623	37	–
XY	0.585	132	28
EuclidianDistance	0.5	2,145	30
Orientation	0.144	2,145	2,145
CenterOrientation	0.137	66	2
TimeDiff_XY	0.126	132	117
TimeDiff_Interpolation	0.06	37	–

Tabelle 4.4.: Vergleich der verschiedenen Features für Lips Part. Auf Interpolation-Features wurde keine PCA angewandt.

Schließlich liefert Tabelle 4.4 einen Vergleich der verschiedenen Features am Beispiel von Lips Part. Der Vergleichswert ist ein F1 score von 0.585, wenn man die Koordinaten aller Landmarks als Features übernimmt („XY“ in der Tabelle). Besser schneiden die Features „CenterDistance“ und „Interpolation“ ab. Es fällt auf, dass die PCA die Dimensionen von „Orientation“ überhaupt nicht und von „CenterOrientation“ fast vollständig reduziert. Die zeitbasierten Features bilden das Schlusslicht in den F1-Wertungen, mehr dazu in der Diskussion.

5. Diskussion

In den Ergebnissen haben wir gesehen, dass Lips Part und Lip Corner Puller relativ gut, die anderen Action Units eher schlecht erkannt werden. Lips Part ist verhältnismäßig leichter zu erkennen, da nur die Abstände der 6 Landmarks in der Mitte des Mundes relevant sind (vgl. Abb. 5.1). Dies erklärt auch, warum die eher simple

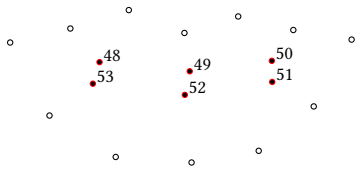

Abbildung 5.1.: Einfach Klassifikation von Lips Part

CenterDist.-Extraction erfolgreich ist: Ein Vergleich der Abstände von Landmarks 49 und 52 zum Mittelpunkt liefert ein einfaches Kriterium, für die Aktivierung der Action Unit. In den Top 5 zu Lip Corner Puller taucht häufig die Interpolation-Extraction auf. Wie in Abb. 2.2 dargestellt, kann durch Polynominterpolation gut die Form und Öffnungsrichtung von Ober- und Unterlippe extrahiert werden.

Eine weitere Erklärung in Bezug auf die Performanceunterschiede für verschiedene Action Units liefern Abb. 5.2 und Tabelle 5.1. Die Anzahl Frames, in denen die Action Unit aktiviert ist, variiert stark zwischen den Action Units. Nicht überraschend sind Lips Part und Lip Corner Puller in vergleichsweise vielen Frames aktiv. Upper Lid Raiser und Outer Brow Raiser lieferten zwar gute F1 scores in der Validierung, aber auf der Testmenge war die Performance schlecht (vlg. Tabelle 4.1). Auch dies stimmt mit einer geringen Anzahl positiver Frames überein. Möglicherweise wurde durch die Auswahl des besten Klassifikators aus 160 die Validierungsmenge overfitted.

Die Action Unit Brow Lowerer fällt hier aus der Reihe, da sie sehr schlecht klassifiziert wurde, obwohl viele positive Frames vorhanden waren. Wahrscheinlich haben wir hierfür nicht die richtigen Features gefunden.

AU	F1 Val	Positives in Training	
		#	%
Lips Part	0.656	12,516	21.5
Upper Lid Raiser	0.394	166	0.3
Outer Brow Raiser	0.391	2,712	4.7
Lip Corner Puller	0.37	6,454	11.1
Cheek Raiser	0.277	5,870	10.1
Nose Wrinkler	0.242	3,432	5.9
Inner Brow Raiser	0.2	3,522	6.1
Chin Raiser	0.191	2,104	3.6
Brow Lowerer	0.163	7,804	13.4
Lip Corner Depressor	0.022	350	0.6

Tabelle 5.1.: Gegenüberstellung der F1 scores und der Häufigkeit positiver Trainingsbeispiele

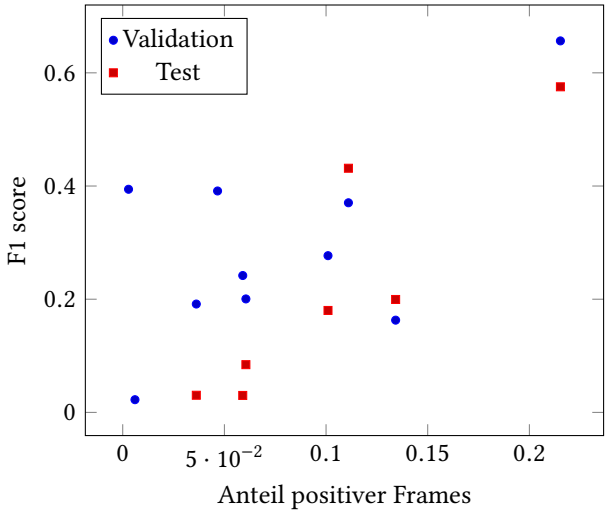

Abbildung 5.2.: F1 score des besten Klassifikators jeder Action Unit gegen Anteil positiver Frames

Die zeitbasierte Klassifikation war wenig erfolgreich. Auch hier ist ein Blick auf das Verhältnis positiver Frames aufschlussreich:

AU	Positives in Training	
	#	%
Lips Part	938	1.6
Brow Lowerer	584	1
Lip Corner Puller	536	0.9
Chin Raiser	434	0.7
Inner Brow Raiser	248	0.4
Outer Brow Raiser	200	0.3
Nose Wrinkler	196	0.3
Lip Corner Depressor	96	0.2
Upper Lid Raiser	80	0.1

Tabelle 5.2.: Positive Trainingsbeispiele für zeitbasierte Features

Aus den vorherigen Ergebnissen geht hervor, dass schon die Klassifikation der *aktuellen* Action Unit im Frame schwer ist. Deshalb ist es nicht verwunderlich, dass die Vorhersage, ob sich die Action Unit gerade ändert, noch schwerer zu treffen ist.

Insgesamt ist festzuhalten, dass die Generalisierung auf Personen, die nicht im Training gesehen wurden, eine schwierige Aufgabe ist. Wie Tabelle 5.3 zeigt, wurden während des Testlaufs für jede Action Unit mehrere Klassifikatoren gefunden, die auf dem Trainingsdatensatz fast perfekte Performance erreichen. Allerdings hat sich diese Performance nur in den wenigen bereits diskutierten Fällen auf die neuen Personen in Validierungs- und Testmenge übertragen. Die Klassifikatoren scheinen sich also die individuellen Formen und Abstände in den Gesichtern der Trainingspersonen gemerkt zu haben (Overfitting). Da RF im Allgemeinen resistenter gegenüber Overfitting ist, lässt sich auch seine Überlegenheit in dieser Arbeit damit erklären.

Es lässt sich schließen, dass die sechs Personen in der Trainingsmenge nicht genügend Varianz bieten, um den Klassifikator zur Abstraktion zu zwingen.

AU	Bester Klassifikator im Training		
	Falsch klassif. nach Training	Precision (Val)	Recall (Val)
Lip Corner Puller	8	1	0.012
Outer Brow Raiser	0	0	0
Lip Corner Depressor	0	0	0
Upper Lid Raiser	0	0	0
Inner Brow Raiser	3	0	0
Cheek Raiser	3	0	0
Lips Part	23	0	0
Brow Lowerer	23	0	0
Chin Raiser	7	0	0
Nose Wrinkler	5	0	0

Tabelle 5.3.: Gute Klassifikation im Training überträgt sich nicht auf die neuen Personen in der Validierungsmenge.

6. Fazit

6.1. Zusammenfassung

Die Aufgabe des Praktikums war es, Action Units in Gesichtern von Personen mittels der vorgegebenen Landmarks zu klassifizieren. Wir haben uns zusätzlich die Frage gestellt: Wie gut lässt sich ein Klassifikator trainieren, der Action Units bei unbekannten Personen erkennen kann?

Dazu haben wir uns sieben verschiedene Methoden überlegt, um aus den Landmarks aussagekräftige Features zu extrahieren. Diese reichen von eher simplen Features (Distanz zum Mittelpunkt), zu komplexeren wie die Polynominterpolation und die zeitliche Ableitung. Wir haben eine Pipeline gebaut, die es ermöglicht, viele Kombinationen aus Features, Klassifikationsalgorithmen und Parametern auf allen Action Units zu testen. Zur Evaluierung haben wir sinnvolle Performancestatistiken ausgesucht, die Klassifikatoren verglichen und die Ergebnisse visualisiert und diskutiert.

Wir stellen fest, dass die Verallgemeinerung auf im Training nicht gesehene Personen nur bei zwei Action Units befriedigend funktioniert hat. Dies liegt unserer Meinung nach an der marginalen Anzahl verschiedener Personen in der Trainingsmenge sowie der geringen Anzahl von Frames, in denen manche Action Units aktiviert waren. Erfolgreich war die Erkennung von Lips Part und Lip Corner Puller. Hier haben auch die von uns entwickelten Features (Distanz zum Mittelpunkt bzw. Interpolation) gute Ergebnisse erziehlt.

6.2. Ausblick

Wir wollen uns hier die Frage stellen, wie unser Ergebnis weiter verbessert werden könnte.

Um auch bei anderen Action Units bessere Ergebnisse zu erzielen, muss für größere Varianz in der Trainingsmenge gesorgt werden. Im Praktikum standen nur 15 Videos (davon nur 10 in der ersten Phase) zur Verfügung, was dazu geführt hat, dass in der Trainingsmenge nur sechs Personen vertreten waren. Die volle DISFA Datenbank bietet Videos von 27 Personen. Hier gibt es also mit mehr Daten noch Verbesserungspotential.

Im Praktikum wurden nur die Landmarks als Rohdaten benutzt. Allerdings enthalten die vollen Videos mehr Informationen, die sich mit Methoden der Computer Vision (Gabor Filter etc.) extrahieren ließen. Diese sind teilweise schon mit der DISFA mitgeliefert.

Die Klassifikation anhand von weiteren zeitbasierten Features könnte noch ausgebaut werden. Dies ist aus Zeitgründen nicht geschehen. Allerdings müsste dazu die Einordnung eines Frames als aktiv bzw. nicht aktiv überdacht werden, damit es mehr positive Frames gibt.

Zusätzlich gibt es noch die Möglichkeit, mehrere Features und Klassifikatoren zu kombinieren, um verlässlichere Ergebnisse zu erzielen. Unser Programm erlaubt viele Einstellungsmöglichkeiten, denen wir aus zeitlichen Gründen nicht weiter nachgegangen sind. Zum Beispiel könnte man die Reihenfolge von Normalisierung der Features und Reduktion mit Hilfe von PCA vertauschen. Zudem ist die von der PCA zu erhaltene Varianz konsequent auf 97% gestellt worden. Versuche mit mehreren Werten könnten unsere Ergebnisse vielleicht ebenfalls verbessern.

A. Anhang

A.1. Auflistung der relevanten Action Units

1 Inner Brow Raiser

2 Outer Brow Raiser

4 Brow Lowerer

5 Upper Lid Raiser

6 Cheek Raiser

9 Nose Wrinkler

12 Lip Corner Puller

15 Lip Corner Depressor

17 Chin Raiser

20 Lip Stretcher

25 Lips Part

26 Jaw Drop

A.2. Konfiguration für den Trainingslauf

```
{
  "landmark_dir": "/path/to/landmarks",
  "action_dir": "/path/to/actionunits",
  "action_threshold": 1,
  "action_names": [
    "Lips Part",
    "Inner Brow Raiser",
    "Outer Brow Raiser",
    "Brow Lowerer",
    "Upper Lid Raiser",
    "Cheek Raiser",
    "Nose Wrinkler",
    "Lip Corner Puller",
    "Lip Corner Depressor",
    "Chin Raiser",
    "Lip Stretcher",
    "Jaw Drop"
  ],
  "training_percent": 0.6,
  "classifier": [
    {"name": "svm", "type": "linear"},
    {"name": "svm", "type": "rbf","gamma": 1 },
    {"name": "svm", "type": "rbf","gamma": 2 },
    {"name": "svm", "type": "rbf","gamma": 5 },
    {"name": "svm", "type": "rbf","gamma": 0.4 },
    {"name": "svm", "type": "sigmoid",
      "gamma": 1, "coef0": 0},
    {"name": "svm", "type": "sigmoid",
      "gamma": 2, "coef0": 0},
    {"name": "svm", "type": "sigmoid",
      "gamma": 5, "coef0": 0},
    {"name": "svm", "type": "sigmoid",
      "gamma": 0.4, "coef0": 0},
    {"name": "svm", "type": "polynom",
      "gamma": 1, "coef0": 0, "degree": 2 },
    {"name": "svm", "type": "polynom",
      "gamma": 5, "coef0": 0, "degree": 2 },
    {"name": "svm", "type": "polynom",
      "gamma": 1, "coef0": 0, "degree": 10 },
    {"name": "svm", "type": "polynom",
      "gamma": 2, "coef0": 0, "degree": 10 },
    {"name": "svm", "type": "polynom",
      "gamma": 5, "coef0": 0, "degree": 10 },
    {"name": "svm", "type": "polynom",
      "gamma": 0.3, "coef0": 0, "degree": 10 },
    {"name": "randomforest", "max_depth": 4,
      "rand_subset_size": 0,
      "number_of_trees": 10},
    {"name": "randomforest", "max_depth": 10,
      "rand_subset_size": 0,
      "number_of_trees": 20},
    {"name": "randomforest", "max_depth": 20,
      "rand_subset_size": 0,
      "number_of_trees": 20},
    {"name": "randomforest", "max_depth": 10,
      "rand_subset_size": 0,
      "number_of_trees": 50},
    {"name": "randomforest", "max_depth": 20,
```

```
    "rand_subset_size": 0,
    "number_of_trees": 50}
],
"cloud_processor": [
{"name": "randomJitterExpander",
  "meanx": 0, "meany":0,
  "stdx": 1, "stdy": 1},
  {"name": "PointCloudNormalization"}
],
"frame_features": [
  {
    "name": "xy",
    "processors":[
      {"name": "stdmeannormalize"},
      {"name": "pcaReducer",
        "retain_variance": 0.97},
      {"name": "shuffle"}
    ]
  },
  {
    "name": "interpolation",
    "processors":[
      {"name": "stdmeannormalize"},
      {"name": "shuffle"}
    ]
  },
  {
    "name": "orientation",
    "processors":[
      {"name": "stdmeannormalize"},
      {"name": "pcaReducer",
        "retain_variance": 0.97},
      {"name": "shuffle"}
    ]
  },
  {
    "name": "euclideandistance",
    "processors":[
      {"name": "stdmeannormalize"},
      {"name": "pcaReducer",
        "retain_variance": 0.97},
      {"name": "shuffle"}
    ]
  },
  {
    "name": "centerdistance",
    "processors":[
      {"name": "stdmeannormalize"},
      {"name": "pcaReducer",
        "retain_variance": 0.97},
      {"name": "shuffle"}
    ]
  },
  {
    "name": "centerorientation",
    "processors":[
      {"name": "stdmeannormalize"},
      {"name": "pcaReducer","retain_variance": 0.97},
      {"name": "shuffle"}
    ]
  }
],
"time_features": [
  {
    "name": "differential",
    "base": {"name": "xy"},
    "truth_threshold": 1,
    "processors": [
      {"name": "stdmeannormalize"},
      {"name": "pcaReducer", "retain_variance": 0.97},
      {"name": "shuffle"}
    ]
  },
  {
    "name": "differential",
    "base": {"name": "interpolation"},
    "truth_threshold": 1,
    "processors": [
      {"name": "stdmeannormalize"},
      {"name": "shuffle"}
    ]
  }
],
"time_frame_step": 2
}
```

Literatur

[1] G. Bradski. In: *Dr. Dobb's Journal of Software Tools* (2000). URL: http://opencv.org/ (besucht am 14. 09. 2016).

[2] *CMake*. URL: http://cmake.org/ (besucht am 14. 09. 2016).

[3] Paul Ekman und Wallace V. Friesen. „Measuring Facial Movement". In: *Environmental Psychology and Nonverbal Behavior* (1976).

[4] *QT5 Bibliothek*. URL: https://www.qt.io/ (besucht am 14. 09. 2016).

[5] Robin Rexeisen, Johannes Stricker und Alexander Schlüter. *Ergebnisse für alle Action Units*. 2016. URL: http://alexschlueter.github.io/detect-emotion (besucht am 14. 09. 2016).