Karlsruher Institut für Technologie

Analyse von Mikrogesichtsausdrücken

Bachelorarbeit
von

Kay Fleischmann

An der Fakultät für Informatik
Institut für Anthropomatik

Erstgutachter: Hazım Kemal Ekenel
Betreuender Mitarbeiter: Tobias Gehrig

Bearbeitungszeit: 28. Aug 2012 – 27. Dez 2012

Facial Image Processing and Analysis Group
Institut für Anthropomatik
Karlsruher Institut für Technologie
Titel: Analyse von Facial-Microexpressions
Autor: Kay Fleischmann

Kay Fleischmann
Postweg 33
69151 Neckargemünd
kay.fleischmann@gmx.de

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Quellen verwendet zu haben.

Karlsruhe, 27. Dez 2012

.....
(Kay Fleischmann)

Abstrakt

Die zuverlässige Erkennung von Mikrogesichtsausdrücken ist momentan ein offenes Problem. In dieser Arbeit wurde die Merkmalsextraktion von Pfister et al. [10] für die Erkennung von Mikrogesichtsausdrücken in C++, OpenCV und Armadillo implementiert. Mit Hilfe des Temporal Interpolation Models wurden die originalen Bilddaten auf eine einheitliche Länge gebracht, bevor die Merkmalsextraktion durch LBP-TOP durchgeführt wurde. Abhängig von den Parametern der Merkmalsextraktion entstanden sehr unterschiedliche Erkennungsraten. Dabei konnte kein Trend erkannt werden, dass Mikrogesichtsausdrücke, die durch das Temporal Interpolation Model auf die gleiche Länge gebracht wurden, die Erkennungsrate verbessern. Die Erweiterung des Algorithmus durch die Video-Magnification [19], die Bewegungen der Mikrogesichtsausdrücke verstärkt, konnte die Erkennungsraten so stabilisieren, dass diese nicht mehr stark von den Parametern der LBP-TOP Merkmalsextraktion abhängen. Zudem konnte die Erkennungsrate um einige Prozentpunkte verbessert werden.

Inhaltsverzeichnis

1. Einführung

Unter Mikrogesichtsausdrücken versteht man flüchtige, unwillentliche Gesichtsausdrücke, die in einem Bruchteil einer Sekunde ablaufen. Sie sind angeboren und lassen sich bewusst nur schwer kontrollieren. Solche Mikrogesichtsausdrücke sind aufgrund ihrer kurzen Dauer und geringen Intensität in der menschlichen Wahrnehmung kaum zu erkennen und treten hauptsächlich in Situationen erhöhter emotionaler Belastung auf, etwa wenn ein Mensch viel zu gewinnen oder zu verlieren hat. Sie sind zu unterscheiden von länger andauernden Gesichtsausdrücken, die gut erkennbar sind. Positive oder negative Emotionen, beispielsweise die Freude beim Wiedersehen eines befreundeten Menschen, werden durch einen bestimmten, deutlich erkennbaren Ausdruck im Gesicht dargestellt, den man dieser Emotion zuordnen kann.

Mikrogesichtsausdrücke haben eine besondere Bedeutung in Situationen, in denen Menschen ihre Gefühle vor anderen verheimlichen wollen, beispielsweise in Befragungssituationen vor Gericht. In diesen Momenten sind Gesichtsausdrücke Hinweise auf den emotionalen Zustand einer Person, aus dem Schlussfolgerungen abgeleitet werden können. Selbst mit viel Übung ist es nur Wenigen möglich, diese Mikrogesichtsausdrücke vollständig zu unterdrücken. Paul Ekman, einer der Pioniere bei der Erforschung von emotionalen Gesichtsausdrücken [4], ordnet Mikrogesichtsausdrücke sechs Basis-Emotionen zu.

Abbildung 1.1.: Von Links nach rechts: Ärger, Ekel, Furcht, Freude, Traurigkeit, Überraschung. Siehe [6]

Ekman [3] entdeckte während seiner Untersuchungen eine solchen Mikrogesichtsausdruck, als eine seiner Patientinnen ihre Selbstmordgedanken verheimlichte. Ekman hatte zunächst

keine Anzeichen gefunden, die darauf hindeuteten, dass die Frau bei ihren Befragungen die Unwahrheit sagte. Erst als die Patientin die Ärzte darauf hinwies, dass sie gelogen hatte, wurde das Videomaterial erneut ausgiebig analysiert. Hierbei entdeckte Ekman letztlich den Mikrogesichtsausdruck, der auf eine Lüge der Patientin hindeutete. Weitere Experimente bestätigten die Vermutung, dass bei den meisten Menschen solche Mikrogesichtsausdrücke unter erhöhten emotionalen Bedingungen automatisch ablaufen. Weil diese Gesichtsausdrücke unwillentlich passieren und kaum beeinflussbar sind, macht sich der Mensch damit für andere lesbar und er kann seinen Gefühlszustand nur noch eingeschränkt verbergen. Ein maschinelles System, das solche Mikrogesichtsausdrücke zuverlässig erkennen kann, wäre für viele Bereiche des öffentlichen und wirtschaftlichen Lebens von besonderer Bedeutung. Im Beispiel von Paul Ekman und seiner Patientin wäre die Information, dass die Frau immer noch selbstmordgefährdet ist, für weitere Behandlungen entscheidend gewesen.

Abbildung 1.2.: Ausschnitt eines **verächtlichen** Mikrogesichtsausdrucks. Dauer ca. 1/3 Sekunde mit 8 Bildern. Die Bilder stammen aus dem Datensatz von Pfister et al. [10]

1.1. Motivation

Grundsätzlich ist denkbar Mikrogesichtsausdrücke als Bilddaten zu erheben, diese zu labeln und einen Klassifikator damit zu trainieren. Allerdings ist die Erkennungsrate eines Klassifikators stark abhängig von der Qualität der Bilddaten. Paul Ekman zeigte [3], dass Probanden während des Experiments in extreme emotionale Situationen zu bringen sind, um eine realistische Umgebung zu simulieren und damit möglichst gute Trainingsdaten zu generieren. Die Erhebung dieser Daten spielt daher eine besondere Rolle für die Deutung von Mikrogesichtsausdrücken. Wie Pfister et al. [9] in ihrer Arbeit zeigten, macht es einen Unterschied, ob ein Mikrogesichtsausdruck gestellt wird oder spontan entsteht. Diese Anforderungen an die Bilddaten macht die Beschaffung eines ausreichenden Trainingskorpus zu einem besonderen Problem.

Da bis Ende 2011 kein Trainingskorpus veröffentlicht wurde, war es schwer möglich, weiterführende Analysen durchzuführen. Pfister et al. haben mit der Veröffentlichung des SMIC-Korpus und ihren Ergebnissen diese Möglichkeit geschaffen. Der veröffentlichte Trainingskorpus basiert auf psychologisch ausgearbeiteten Experimenten und beinhaltet Bilddaten, die bereits gelabelt sind. Hat man einen ausreichenden Trainingskorpus generiert oder es steht einer zur Verfügung, dann sind im Anschluss aus diesen Daten Merkmale zu extrahieren, um sie in einer anderen noch unbekannten Bildsequenz wiederzuerkennen. Da Mikrogesichtsausdrücke in der Bildsequenz wenige Veränderungen verursachen, ist es wichtig eine geeignete Methode für die Merkmalsextraktion zu wählen. Pfister et al. [10] zeigten in ihren Experimenten, dass die Erkennungsrate stark von der Bildsequenz abhängt und nicht notwendigerweise nachdem die Parameter-Optimierung der Merkmalsextraktion durchgeführt wurde auch für weitere Bildsequenzen gute Ergebnisse liefert. Für weiterführende Analysen soll in dieser Arbeit der Korpus von Pfister et al. verwendet werden.

Hao-Yu Wu et al. [19] entwickelten eine Methode minimale Bewegungen in einem Video so zu verstärken, dass Veränderungen im Bild deutlich erkennbar werden. Mit dieser Methode sollen die Mikrogesichtsausdrücke vorverarbeitet werden, um so die Merkmalsextraktion einer Bildsequenz zu verbessern.

1.2. Forschungen zu Mikrogesichtsausdrücken

Mikrogesichtsausdrücke wurden bereits sehr früh von Psychologen untersucht. William S. Condon veröffentlichte 1996 seine ersten Studien über Mikrosausdrücke. Er erkannte etwa das Schulterzucken einer Frau, während ihr Ehemann die Hand auf sie zu bewegte [18]. John Gottmann erweiterte diese Forschungsergebnisse, als er anhand von Videomaterial die Lebenszeit einer Beziehung prognostizierte [5] [8]. Paul Ekman untersuchte in seinen Arbeiten besonders gesichtsbezogene Mikrogesichtsausdrücke, um Lügner aufgrund ihrer Mikrogesichtsausdrücke zu überführen. Ekman zeigte, dass sechs Basis-Emotionen existieren, die kulturübergreifend gefühlt und verstanden werden. Sie sind bei allen Menschen angeboren [4]. Er legte sich nicht nur auf gesichtsbezogene Mikroausdrücke fest, sondern beschrieb auch Mikroausdrücke, die am Körper auftreten können, wie zum Beispiel ein Schulterzucken [3]. Er trug hauptsächlich dazu bei, zu erkennen, unter welchen Bedingungen Mikrogesichtsausdrücke entstehen und wie man sie bewusst provozieren kann. Für die technische Umsetzung eines maschinellen Erkenners von gesichtsbezogenen Mikroausdrücken veröffentlichten Polikovsky et al. [12] und Shrev et al. [14] [15] bereits erste Arbeiten. Beide verwendeten einen Trainingskorpus mit vorgespielten Mikrogesichtsausdrücken. Den Probanden wurden Mikrogesichstausdrücke gezeigt, die sie vorspielen sollten. Polikovsky et al. sammelte Daten von 10 Studenten. Die Merkmalsextraktion wurde mit Hilfe der Gradient Orientation Histogram-Deskriptoren durchgeführt. Shrev et al. verwendeten 100 vorgespielte Mikroausdrücke in ihrem Korpus und verwendeten Strain Patterns Merkmals-Deskriptoren. Shrev et al. erzielten mit ihrem Ansatz eine Erkennungsrate von 50%.

Pfister et al. [10] veröffentlichten Ende 2011 ihren aufwendig generierten Trainingskorpus zusammen mit ihren Ergebnissen, der keine gestellten Mikrogesichtsausdrücke enthielt, sondern mit Hilfe eines Experiments solche Mikrogesichtsausdrücke bei den Probanden hervorrief. In ihrer Arbeit interpolierten sie alle Mikrogesichtsausdrücke auf eine einheitliche Bildsequenz-Länge, bevor sie eine Merkmalsextraktion durchführten. Eine Merkmalsextraktion ist der Prozess, bei dem charakteristische Merkmale von zu untersuchenden Daten in einen Merkmalsraum transformiert werden. Im Verlauf der Arbeit bezeichnen Daten die Bildsequenzen der Mikrogesichtsausdrücke. Zu jedem Bild entsteht dann ein korrespondierender Merkmalsvektor in diesem Merkmalsraum.

Um Mikrogesichtsausdrücke in anderen noch unbekannten Bildsequenzen wiederzuerkennen, wird ein Klassifikator verwendet, der mit Hilfe von bereits bekannten Mikrogesichtsausdrücken, die als Merkmalsvektoren vorliegen, trainiert wird. Es existieren viele unterschiedliche Klassifikatoren, die abhängig vom Klassifikationsproblem gewählt werden. Pfister et al. verwendeten in ihrer Arbeit drei Klassifikatoren - eine Support Vector Machine (SVM), Random Forest (RF) und Multi-Kernel-Learning (MKL).

1.3. Ziel und Aufbau der Arbeit

Ziel dieser Arbeit ist es, den Algorithmus von Pfister et al. [10] in C++ und OpenCV zu reimplementieren, die Merkmalsextraktion zu beleuchten und zu analysieren, wie sich der Algorithmus von Pfister et al. verhält, wenn man die Bewegungen der Mikrogesichtsausdrücke in der Bildsequenz mit Hilfe der Video-Magnification [19] verstärkt. Diese Arbeit beschränkt sich dabei auf gesichtsbezogene Mikroausdrücke. Der Erkenner überprüft, ob es sich bei einer vorgegebenen Bildsequenz um einen Mikrogesichtsausdruck handelt oder nicht.

Local Binary Pattern TOP (LBP-TOP) wird wie bei Pfister et al. als Merkmalsextraktion dienen. Vor der Merkmalsextraktion werden alle Bildsequenzen auf die gleiche Länge gebracht. Für die Evaluation der Ergebnisse beschränkt sich diese Arbeit auf das Training einer Support Vector Machine. Anschließend wird der Algorithmus durch eine Kreuzvalidierung ausgewertet. Die Arbeit wird zeigen, dass die Erkennungsrate von Parametern der LBP-TOP Merkmalsextraktion abhängt und daher die Parameter sorgfältig gewählt werden müssen. Im Anschluss wird der Algorithmus durch die Video-Magnification [19] erweitert. Hier werden die Bilddaten vor der TIM-Normalisierung ein weiteres Mal vorverarbeitet, indem bestimmte Bewegungen innerhalb eines Videos durch einen Bandpass gefiltert und verstärkt werden. Ein Bandpass-Filter erhält Frequenzen eines bestimmten Frequenzbereichs. Bei diesem Prozess werden minimale Bewegungen sichtbar, die vorher schwer oder kaum sichtbar waren. Dieser Prozess wird dann den Algorithmus weiter verbessern und darüber hinaus macht es den Algorithmus weniger anfällig für die Parametrisierung der LBP-TOP Merkmalsextraktion.

Im Kapitel 2 werden technische Grundlagen erläutert, die wichtig für das Verständnis in Kapitel 3 (Implementierung) und 4 (Evaluation) sind. In Kapitel 2.1 wird das Temporal Interpolation Model besprochen, wodurch die Mikrogesichtsausdrücke auf eine einheitliche Bildsequenz-Länge hoch- beziehungsweise runter skaliert werden. Dazu wird das Temporal Interpolation Model verwendet. Kapitel 2.2 beschäftigt sich dann mit Local Binary Patterns (LBP) für Bilder und der Erweiterung LBP-TOP für ganze Bildseqzenzen, eine Merkmalsextraktion für die Bildsequenzen. Die mathematischen Grundlagen der Support Vector Machine werden im Anschluss erläutert. Abschließend zu Kapitel 2 wird die Methode der Video-Magnification vorgestellt. In Kapitel 3 wird aufbauend die entwickelte Software erläutert und beschrieben, wie die Video-Magnification in den Algorithmus eingebaut wird. In Kapitel 4 werden Experimente vorgestellt und ihre Ergebnisse erläutert. Im Anhang befindet sich eine komplette Implementierung des Temporal Interpolation Models sowie eine kurze Beschreibung der Konfigurationsdatei des Programms.

2. Grundlagen

Für die weitere Analyse von den im ersten Kapitel vorgestellten Mikrogesichtsausdrücken, sollen in diesem Kapitel Grundlagen erläutert werden. Dieses Kapitel wird besonders technische Grundlagen erläutern, die in Kapitel 3 und 4 verwendet werden. Es beschreibt das Temporal Interpolation Model und weitere wichtige Methoden wie Merkmalsextraktion, Klassifikation und beschreibt den Trainingskorpus von Pfister et al. [10].

2.1. Normalisierung der Daten durch das Temporal Interpolation Model

Jeder entstehende Mikrogesichtsausdruck ist unterschiedlich. Die Ausdrücke variieren in ihrer Länge und sind stark abhängig vom Probanden. Mit Hilfe des Temporal Interpolation Models (TIM) [22] [10] lassen sich Bildsequenzen auf eine gemeinsame Länge bringen. Dies hat den Vorteil, die Bildsequenzen besser vergleichbar zu machen und ermöglicht bei sehr kurzen Mikrogesichtsausdrücken sogar eine Skalierung, dass auch bei wenigen Bildern eine Merkmalsextraktion möglich ist. Für die Local Binary Patterns ist es wichtig eine Mindestanzahl von Bildern zu haben, um überhaupt eine Merkmalsextraktion durchführen zu können. Genauso kann die Länge der Bildsequenz auf eine beliebige Bilderanzahl reduziert werden. Ausgangspunkt ist eine beliebige Bildsequenz von einem Mikrogesichtsausdruck. Angenommen man würde diese Bildsequenz auf einen Pfad P_n legen mit einem festen Abstand zwischen den Bildern, dann ergibt dieser Pfad einen Graph, der genauso viele Knoten enthält, wie es Bilder gibt. Zwei Knoten bzw. zwei Bilder sind genau dann miteinander verbunden, wenn sie in der Bildreihenfolge nebeneinander liegen. Dieser Graph lässt sich als Adjazenzmatrix darstellen. Sei $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ diese Adjazenzmatrix zu P_n, sodass gilt $\boldsymbol{W}_{i,j} = 1$ für $|i - j| = 1$. Alle anderen Einträge der Matrix $\boldsymbol{W}$ sind gleich Null. Genauso kann man für diesen Graph eine Laplacematrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ formulieren, wobei $\boldsymbol{D}_{ii} = \sum_{j=1}^{n} \boldsymbol{W}_{ij}$ die Anzahl der Kanten für den Knoten i ist [1]. Ein Beispiel für solch einen Pfad ist in Abbildung 2.1 zu sehen.

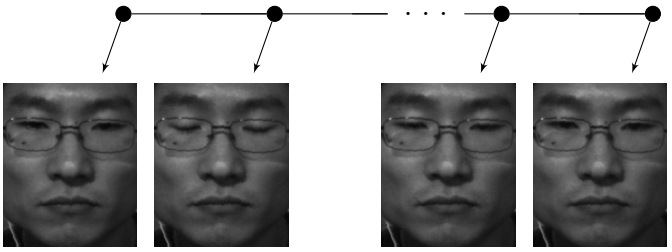


Abbildung 2.1.: Darstellung eines Mikrogesichtsausdrucks, der entlang eines Pfades gelegt wird mit einem festen Abstand zwischen den Bildern.

2.1.1. Die Transformation vom Pfad auf die Kurve

In Abbildung 2.1 ist ein Graph dargestellt, dessen Knoten die Bildsequenz und die Kanten der Übergänge zwischen den Bildern darstellt.

Angenommen es gibt eine kontinuierliche Kurve, die den Ablauf einer Bildsequenz beschreibt, und alle Knotenpunkte werden auf diese Kurve abgebildet, dann wäre es möglich Punkte auf der kontinuierlichen Kurve zu finden, die zwischen den Knoten liegen. Für diese Punkte könnte man eine Rücktransformation in den Bildbereich berechnen, sodass Bilder rekonstruiert werden, die vorher nicht sichtbar waren. Zhou et al. [23] zeigten, dass dies möglich ist.

Seien im Folgenden $y = (\boldsymbol{y_1}, \boldsymbol{y_2}, .., \boldsymbol{y_n})$ die korrespondierenden Punkte des Pfads P_n auf der Kurve, dann versucht man genau die Kurve zu finden, bei der die Distanz zwischen den benachbarten Punkten $\boldsymbol{y_i}$ minimal ist, sodass die Struktur des Graphs erhalten bleibt [7].

$$\arg\min \quad \sum_{i,j} ||\boldsymbol{y_i} - \boldsymbol{y_j}||^2 \boldsymbol{W_{ij}} \tag{2.1}$$

Für die Berechnung der Transformation verwendet man nicht direkt die vektorisierten Bilder, sondern subtrahiert von allen den Mittelwert. Dieser Mittelwert wird später bei der Bildrekonstruktion wieder aufaddiert. Seien weiter $\{\boldsymbol{\xi_i} \in \mathbb{R}^m\}_{i=1}^n, n \ll m$ die originalen vektorisierten Bilder aus der Bildsequenz des Mikrogesichtsausdrucks und $\overline{\boldsymbol{\xi}}$ der Mittelwert aller Bilder, dann erhält man für jedes Bild einen neuen repräsentativen Vektor $\boldsymbol{x_i} = \boldsymbol{\xi_i} - \overline{\boldsymbol{\xi}}$. Im weiteren Verlauf soll die Matrix $\boldsymbol{X}$ alle Bilder $\boldsymbol{x_i}$ spaltenweise in geordneter Reihenfolge enthalten. Sei nun $\boldsymbol{y_i} = \boldsymbol{w}^T \boldsymbol{x_i}$ die Transformation eines beliebigen Punktes des Pfads auf die Kurve, dann kann Gleichung 2.1 umgeformt werden in

$$\arg\min_{\boldsymbol{w}} \quad \sum_{i,j} ||\boldsymbol{w}^T \boldsymbol{x_i} - \boldsymbol{w}^T \boldsymbol{x_j}||^2 \ \boldsymbol{W_{ij}} \qquad i, j = 1, 2, ..., n. \tag{2.2}$$

Dieser Vektor $\boldsymbol{w}$ soll die Bilder aus dem Bildbereich in den Funktionsbereich transformieren. Ist dieser Vektor bekannt, kann man eine Rücktransformation eines beliebigen Punktes von der Kurve zurück in den Bildbereich durchführen, sodass man Bilder erhält, die vorher nicht explizit existierten. Zhou et al. [23] zeigten, dass sich jedes beliebige Bild auf dieser Kurve durch eine Synthese-Gleichung berechnen lässt.

M. Belkin et al. [20] zeigten weiter, dass das Problem aus Gleichung 2.1 umformuliert werden kann in ein neues Minimierugnsproblem:

$$\arg\min \ \boldsymbol{y}^T L\boldsymbol{y} \quad \text{m.d.N.B.} \ \ \boldsymbol{y}^T\boldsymbol{y} = d \tag{2.3}$$

Indem $\boldsymbol{y}^T\boldsymbol{L}\boldsymbol{y} = \boldsymbol{y}^T(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{y}$ ausmultipliziert wird, entsteht Gleichung 2.1 Sei nun $\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{w}$ die Transformation der Punkte des Pfads P_n auf die Kurve, dann gilt für die Nebenbedingung $\boldsymbol{y}^T\boldsymbol{y} = (\boldsymbol{X}^T\boldsymbol{w})^T(\boldsymbol{X}^T\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w} = d$ und es folgt für das Minimierungsproblem

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \ \boldsymbol{w}^T\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}\boldsymbol{w} \quad \text{m.d.N.B.} \ \ \boldsymbol{w}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w} = d. \tag{2.4}$$

2.1.2. Herleitung der Transformation $\boldsymbol{w}$

In Kapitel 2.1.1 wurde die Notwendigkeit der Transformation $\boldsymbol{w}$ für den Übergang vom Bildbereich in den Funktionsbereich dargestellt. Im Folgenden soll aus Gleichung 2.3 das generalisierte Eigenwertproblem hergeleitet werden, das für die Berechnung der Transformation $\boldsymbol{w}$ essentiell ist. Die Gleichung 2.3 lässt sich durch die Methode der Lagrange-Multiplikatorenregel lösen.

$$L(\boldsymbol{w}, \lambda) = \boldsymbol{w}^T\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T\boldsymbol{w} - \lambda(\boldsymbol{w}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w} - d) \tag{2.5}$$

$$\frac{\partial L(\boldsymbol{w}, \lambda)}{\partial \boldsymbol{w}} = 2\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T\boldsymbol{w} - \lambda\ 2\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w} = 0 \iff \boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T\boldsymbol{w} = \lambda\ \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w} \tag{2.6}$$

Damit ist die Lösung für die Transformation

$$\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T\boldsymbol{w} = \lambda\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w}. \tag{2.7}$$

2.1.3. Eigenvektoren der Laplacematrix

Bei der Laplacematrix sind bereits die Eigenvektoren bekannt. Seien $\lambda_1 < \lambda_2 < ... < \lambda_{n-1}$ die zur Laplacematrix gehörigen Eigenwerte, dann gilt für die Eigenvektoren $\boldsymbol{y_k}(u) = sin(\pi kun + \pi(n-k)/n), k = 1, ..., n-1$. Diese Vektoren befinden sich auf der Kurve, die gesucht wird und die Vermutung liegt nahe, dass die Menge der Eigenvektoren die Kurve beschreiben, wie auch Zhou et al. [23] in ihrer Arbeit zeigten.

Ersetzt man u durch $t = u/n$, dann erhält man in Abhängigkeit der Zeit die Funktionen $f_k^n(t) = sin(\pi kt + \pi(n-k)/n), t \in [0, 1]$. Mit der folgenden Funktion $\boldsymbol{\mathcal{F}}^n$ lassen sich dann alle Punkte des Pfads auf der Kurve beschreiben.

$$[1/n, 1] \to \mathbb{R}, \qquad \boldsymbol{\mathcal{F}}^n(t) = \begin{bmatrix} f_1^n(t) \\ f_2^n(t) \\ \vdots \\ f_{n-1}^n(t) \end{bmatrix} \tag{2.8}$$

2.1.4. Synthese-Gleichung für Bilder

Zhou et al. [23] haben das generalisierte Eigenwertproblem aus Gleichung 2.7 mit Hilfe der Singulärwertzerlegung $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ gelöst, sodass eine Synthese-Gleichung für ein beliebiges Bild auf der Kurve ensteht.

$$\boldsymbol{\xi}^{\boldsymbol{syn}} = \boldsymbol{U}\ (\boldsymbol{\Upsilon}^{-1})^T\ \boldsymbol{M}\ \boldsymbol{\mathcal{F}}^n(t) + \overline{\boldsymbol{\xi}} \tag{2.9}$$

Für die geschickte Wahl von $\boldsymbol{A} = (\boldsymbol{Q}\boldsymbol{Q}^T)^{-1}(\boldsymbol{Q}\boldsymbol{L}\boldsymbol{Q}^T)$ mit $\boldsymbol{Q} = \boldsymbol{\Sigma}\boldsymbol{V}^T$ seien $\boldsymbol{v_1}, ..., \boldsymbol{v_{n-1}}$ die Eigenvektoren von $\boldsymbol{A}$. $\boldsymbol{\Upsilon} = [\boldsymbol{v_1}, ..., \boldsymbol{v_{n-1}}]$, $\boldsymbol{Y} = [\boldsymbol{y_1}, ..., \boldsymbol{y_{n-1}}]$ sind die bekannten Eigenvektoren von $\boldsymbol{L}$ und $\boldsymbol{M}_{kk} = m_k$ für $\boldsymbol{Q}^T\boldsymbol{v_k} = m_k\boldsymbol{y_k}$.

Das Temporal Interpolation Model berechnet aus einer Menge von Bildern ein Modell (Gleichung 2.8), das als Parameter einen Zeitpunkt $t \in [0, 1]$ (Position auf der Kurve) erwartet, um ein beliebiges Bild zwischen Anfangsbild ($t = 0$) und Endbild ($t = 1$) zu generieren.

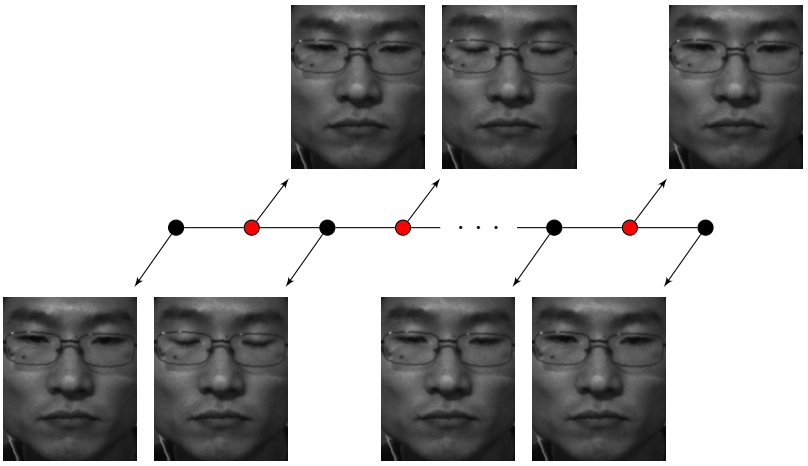


Abbildung 2.2.: Darstellung eines Mikrogesichtsausdrucks entlang eines Pfades. Die roten Punkte auf dem Pfad sind Bilder, die nicht explizit gegeben sind und können durch das Temporal Interpolation Model berechnet werden.

In Abbildung 2.3 ist eine Bildsequenz von einem originalen Mikrogesichtsausdruck mit 28 Bildern, im Vergleich zu einer Bildsequenz, die auf 15 Bilder reduziert wurde.

Abbildung 2.3.: Originaler Mikrogesichtsausdruck mit 28 Bildern aus dem Korpus von Pfister et al.[10], aufgenommen mit einer Hochgeschwindigkeitskamera

Abbildung 2.4.: Mikrogesichtsausdruck aus dem Korpus von Pfister et al., der durch das Temporal Interpolation Model auf 15 Bilder interpoliert wurde.

2.2. Merkmalsextraktion - Local Binary Patterns

Nach Zhao et al. [21] lässt sich aus einer Bildsequenz ein Merkmalsvektor extrahieren, der besonders robust gegen geringe Lichtintensitäten ist. Ein Video besteht aus einer Sequenz von Bildern mit einer festen Höhe und Breite. Diese Bildsequenz spannt einen Raum im $\mathbb{R}^3$ auf. In der X-Y Ebene liegen die einzelnen Bilder hintereinander und in Richtung der Z-Ebene ist die Veränderung der Zeit gegeben. Die Z-Achse wird hier auch als T-Achse bezeichnet. Betrachtet man die X-T Ebene sieht man, wie sich Zeilen in Abhängigkeit der Zeit verändern. Die Y-T Ebene beschreibt die Veränderung der Spalte in Abhängigkeit der Zeit. Jeder Pixel hat damit nicht mehr nur zwei Koordinaten (x, y), sondern auch noch eine Zeit-Komponente, die Bild-Position (x, y, t).

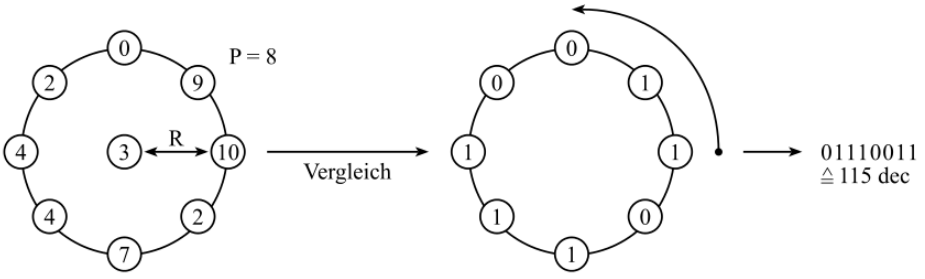


Abbildung 2.5.: Darstellung der LBP-Code Berechnung. Diese Berechnung ist abhängig von einem vorher festgelegten Radius.
(Siehe [16]).

Der Ansatz der Local Binary Patterns (LBP) wurde ursprünglich für ein 2-dimensionales Graustufen Bild entwickelt und wurde später erst durch die Zeit-Achse erweitert. Zu jedem Pixel wird ein charakterisierender Binärwert berechnet, der angibt, wie sich die Umgebung um einen zentralen Pixel verhält. Wie viele Pixel in der Umgebung in diese Berechnung mit einfließen ist von der Anwendung abhängig (z.B. 3×3 Nachbarschaft). Mit dem Radius R kann man angeben, welche Pixel in der Umgebung in die Berechnung mit einfließen. Für jeden Pixel wird dann der Grauwert des zentralen Pixels abgezogen - Werte kleiner oder gleich Null werden 0 (dunkel) und alle anderen werden 1 (hell). Dies macht den Deskriptor weniger anfällig für geringe Hell-Dunkel Stufen.

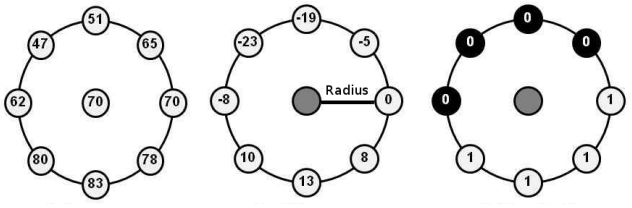


Abbildung 2.6.: Berechnung eines LBP-Codes eines Pixels in einer Umgebung mit dem Radius R. Siehe [13].

Nachdem die Pixel binärisiert sind, wird die Summe über die Zweierpotenzen gebildet, sodass die Potenz nur dann in die Summe eingeht, wenn der Pixel in der Umgebung hell, also 1 ist. Das Ergebnis ist dann der LBP-Code zu diesem Pixel.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^P \qquad s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & sonst. \end{cases} \tag{2.10}$$

Für das obige Beispiel würde man als Ergebnis für den Pixel einen Wert von $1 \cdot 1 + 1 \cdot 2 + 1 \cdot 4 + 1 \cdot 8 + 0 \cdot 16 + 0 \cdot 32 + 0 \cdot 64 + 0 \cdot 128 = 15$ erhalten. Bei welchem Pixel man anfängt zu zählen spielt an dieser Stelle keine Rolle. Durch die Berechnung dieser charakterisierenden LBP-Codes erhält jeder Pixel einen neuen Wert $f_l(x, y)$. Aus diesen Codes wird ein Histogramm gebildet, das als Merkmalsvektor genutzt werden kann. Um eine bessere Auflösung des Merkmalsvektors zu erhalten, kann man das Bild vorher in 4×5 oder 8×8 Blöcke aufsplitten und einzeln die Histogramme H pro Block berechnen, die dann zusammen wieder den gesamten Merkmalsvektor ergeben. Durch die Aufteilung in Blöcke erhält man auch regionale Informationen über Helligkeitsintensitäten. Wie man die Bilder in Blöcke aufteilt ist wieder anwendungsabhängig. Im Folgenden bezeichnet H_i die Anzahl der vorkommenden LBP-Codes i. Die Histogramme werden zuletzt noch durch die Anzahl der LBP-Codes dividiert und damit normalisiert, um diese mit anderen Bildern vergleichbar zu machen. N_i ist dann die normalisierte Beschreibung für die LBP-Codes.

$$H_i = \sum_{x,y} I\{f_l(x, y) = i\}, \quad i = 0, ..., n-1 \qquad N_i = \frac{H_i}{\sum_j H_j} \tag{2.11}$$

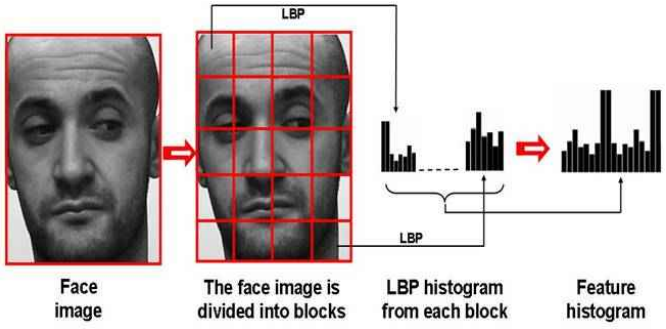


Abbildung 2.7.: Local Binary Patterns Merkmalsextraktion. Ein Bild wird in Blöcke aufgeteilt und es wird jeweils ein Histogramm berechnet. Diese Histogramme werden aneinander gehängt und bilden einen Merkmalsvektor. Siehe [13].

Spatiotemporal LBP bzw. Spatiotemporal Local Texture Descriptors (SLTD) sind Raum-Zeit Merkmals-Deskriptoren und erweitern den LBP Algorithmus durch die Zeit-Komponente. Bei der Erweiterung des Local Binary Patterns wird nicht nur das Histogramm in der X-Y Raum-Ebene berechnet, sondern auch jeweils für die X-T und Y-T Ebene. Um den Aufwand für die Berechnung dieser LBP-Codes zu reduzieren, entstand die LBP-TOP (Local Binary Patterns Three Orthogonal Planes) Erweiterung. Bei dieser Merkmalsextraktion werden zur Berechnung der LBP-Codes nur die Pixel einbezogen, die in einer orthogonalen Ebene zueinander liegen. Zu jedem Block wird dann wieder ein Histogramm berechnet, und insgesamt ergibt sich aus allen Block-Histrogrammen der gesamt Merkmalsvektor.

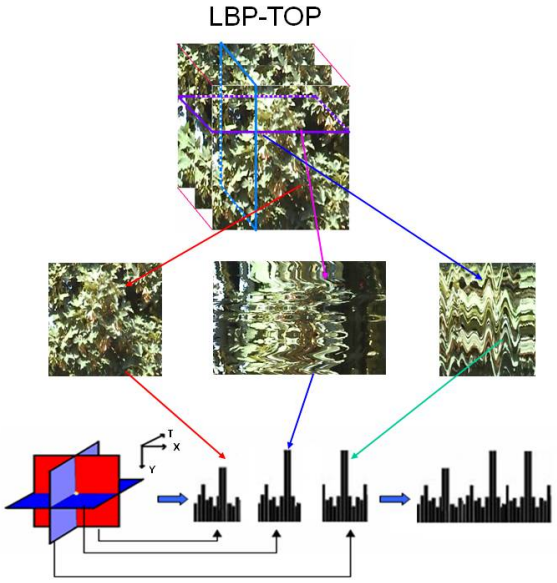


Abbildung 2.8.: LBP-TOP (Drei-Orthogonale-Ebenen) Erweiterung. XY Bilddarstellung, XT - Veränderungen der Zeilen, YT - Veränderungen der Spalten. Siehe [13].

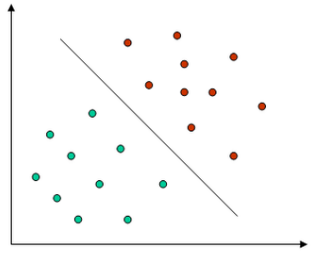
Im folgenden ist der Algorithmus nochmal Schritt für Schritt aufgelistet.

Algorithm 1 LBP-TOP Merkmalsextraktion

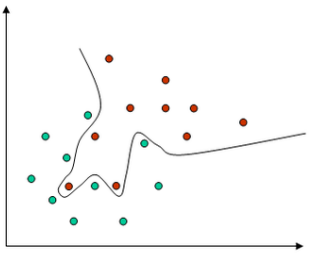
1: Die Textur wird in ein Graustufen-Bild konvertiert.
2: Für jeden Pixel in einem Block wird z.B. durch die 8-Nachbarschaft ein 8-Bit LBP-Code berechnet. Wenn die Pixelintensität größer ist als die des Zentrums, schreibe: 1, ansonsten 0.
3: Dann wird die Textur in Blöcke (zb. 8×8 oder 16×16) aufgeteilt.
4: Berechne für jeden Block ein Histogramm. Gezählt werden die Häufigkeiten der LBP-Codes.
5: Optional: Normalisiere das Histogramm (aber essentiell, wenn man zwei Bilder unterschiedlicher Größe vergleichen möchte).
6: Der Merkmalsvektor ist dann die Verkettung aller Block-Histogramme.

2.3. Klassifikation - Support Vector Maschine

In dieser Arbeit wird für die Evaluation eine Support Vector Machine (SVM) verwendet. Die Herleitungen stammen aus [2]. Eine Support Vector Machine versucht beim Training eine optimale Hyperebene zu finden, welche die Trainingsdaten in zwei Klassen trennt. Im Beispiel von Abbildung 2.9 (a) sind beide Klassen linear trennbar. Mit einer SVM können auch nichtlinear trennbare Daten klassifiziert werden, indem der sogenannte Kernel-Trick angewendet wird (siehe Abbildung 2.9 (b)). Für das Training einer SVM müssen bereits gelabelte Daten vorliegen. Das Training besteht darin, geeignete Stützvektoren zu finden. Das sind Datenpunkte, die den kleinsten Abstand zu dieser Hyperebene haben. Für die Datenpunkte soll nun eine Trennebene bestimmt werden, die zu allen Stützvektoren einen möglichst großen Abstand haben.



(a) Linear separierbares Problem im $\mathbb{R}^2$, die Hyperebene ist eine Gerade.



(b) Nichtlinear separierbares Problem.

Abbildung 2.9.: Bei nichtlinear separierbaren Problemen bilde in einen höher dimensionalen Raum ab, damit das Problem separierbar ist. Siehe [17]

Ist das Problem nichtlinear, so muss man eine nichtlineare Transformation finden, um die Datenpunkte in einen hochdimensionalen Featureraum zu transformieren. In diesem Raum kann dann eine lineare Trennung erfolgen. Im Folgenden seien $\boldsymbol{x_1}, ..., \boldsymbol{x_n}$ die Datenpunkte und $t_1, ..., t_n$ die zu den Datenpunkten gehörenden Zielwerte (Labels). Für den linearen Fall benötigt man für die Hyperebene zwei Parameter $\boldsymbol{w}$ und b für $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$. Der Abstand zwischen Hyperebene und Datenpunkt ist gegeben durch $\frac{t_n y(\boldsymbol{x_n})}{||\boldsymbol{w}||}$. Für t_n wird ein Zielwert -1 oder 1 festgelegt.

Man erhält somit ein Maximierungs- und Minimierungsproblem, das es zu lösen gilt.

$$(\boldsymbol{w}^{opt}, b^{opt}) = \arg\max_{\boldsymbol{w},b} \left[\frac{1}{||\boldsymbol{w}||} \min_n [t_n(\boldsymbol{w}^T\boldsymbol{x_n} + b)] \right] \tag{2.12}$$

Dieses Problem scheint auf den ersten Blick sehr komplex zu wirken. Mit der künstlichen Nebenbedingung $\min_n[t_n(\boldsymbol{w}^T\boldsymbol{x_n} + b)] = 1$ lässt sich der Ausdruck vereinfachen in

$$(\boldsymbol{w}^{opt}, b^{opt}) = \arg\max_{\boldsymbol{w},b} \frac{1}{||\boldsymbol{w}||} = \arg\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2 \tag{2.13}$$

und es folgt automatisch $t_n(\boldsymbol{w}^T\boldsymbol{x_n} + b) \geq 1$ für alle $\boldsymbol{x_n}$. Mit Hilfe von n Lagrange Multiplikatoren kann nun das vereinfachte Problem gelöst werden.

$$L(\boldsymbol{w}, b, a) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{n=1}^{N} a_n[t_n(\boldsymbol{w}^T\boldsymbol{x_n} + b) - 1] \tag{2.14}$$

Die Ableitung nach $\boldsymbol{w}$ und b ergeben $\boldsymbol{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{x_n}$ und $0 = \sum_{n=1}^{N} a_n t_n$. Eingesetzt in die Lagrange-Funktion ergibt sich als Lösung des Gesamtproblems:

$$L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \underbrace{k(\boldsymbol{x_n}, \boldsymbol{x_m})}_{Kernel-Funktion} \tag{2.15}$$

Man sieht an dieser Stelle, dass die Lösung abhängig von einer Beziehung zwischen den Datenpunkten ist. Das bedeutet, für das Training der SVM muss die Transformation in einen höherdimensionalen Raum nicht explizit gegeben sein, sondern es reicht aus, eine geeignete Kernel-Funktion zu wählen, die genau diese Beziehung berechnet. Dies wird auch als Kernel-Trick bezeichnet. In Abbildung 2.10 ist ein Beispiel gegeben, bei dem durch eine Transformation in einen höherdimensionalen Raum die Daten linear trennbar werden.

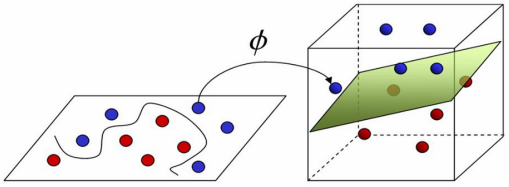


Abbildung 2.10.: Darstellung einer Support Vector Machine. Links ist die Trennung im originalen Merkmalsraum nichtlinear. Rechts, nach der Transformation in einen hochdimensionalen Raum, ist die Trennung linear durch eine Hyperebene durchführbar. Die Transformation Φ ist dabei nicht explizit gegeben, sondern es genügt eine Kernelfunktion zu verwenden, welche die Beziehung im hochdimensioanlen Raum zwischen zwei Merkmalsvektoren beschreibt [11].

2.4. Algorithmus von Pfister et al.

2.4.1. SMIC Korpus

Pfister et al. [10] generierten für ihre Arbeit einen eigenen Korpus, den Spontaneous Micro-expression Corpus (SMIC), da vorher kein geeigneter Korpus zur Verfügung gestellt wurde. Für die Experimente wurde eine 100fps Kamera mit einer Auflösung von 640x480 Pixel verwendet. Der Datensatz umfasste insgesamt 6 Probanden (3 Frauen, 3 Männer) mit insgesamt 77 spontan hervorgerufenen Mikrogesichtsausdrücken. Von den Probanden trugen vier eine Brille.
Die Aufgabe bestand darin, 16 Filmclips anzusehen, die vorher so ausgewählt wurden, dass sie die Probanden in einen emotional erregten Zustand versetzten. Die Probanden sollten dabei wie in einem Verhör versuchen, eine sichtbare emotionale Reaktion zu unterdrücken. Ein Beobachter, dem nicht bekannt war, welcher Film gerade zu sehen ist, sollte aufgrund der Gesichtsausdrücke erkennen, welcher Film abgespielt wird. Um eine hohe Motivation beim Probanden zu erzeugen, sichtbare Emotionen zu unterdrücken, wurde dem Probanden gedroht, im Fall des Erkennens des abgespielten Films durch den Beobachter, einen Fragebogen mit 500 Fragen ausfüllen zu müssen. Dabei entstanden insgesamt 210 Minuten Videomaterial mit 1.260.000 Bildern. Ekman entwickelte einen Prozess, um Mikrogesichtsausdrücke möglichst gut erkennen und labeln zu können. Danach wurde das Material zunächst Bild für Bild angeschaut und dann mit immer schneller werdender Geschwindigkeit, um die Mikrogesichtsausdrücke sichtbar zu machen. Die durchschnittliche Länge eines Mikrogesichtsausdrucks lag bei 29 Bildern bzw. 3/10 Sekunde. Alle Mikrogesichtsausdrücke wurden mit fünf von Ekman ausgewählten Basisemotionen gelabelt: Ärger, Angst, Freude, Traurigkeit, Ekel.

Abbildung 2.11.: Probanden aus dem Spontaneous Micro-expression Corpus (SMIC) [10].

Abbildung 2.12.: Ausgeschnittene Gesichter, die für Merkmals-Generierung verwendet wurden. Siehe [10].

Im Folgenden ist Schritt für Schritt aufgelistet, wie Pfister et al. [10] die Bilddaten der Mikrogesichtsausdrücke in den Merkmalsraum transformierten. Die Bilddaten nach dem Verarbeitungsschritt 6 wurden von Pfister et al. veröffentlicht und für diese Arbeit verwendet.

Algorithm 2 Merkmalsextraktion von Pfister et al

1: Initialisiere LBP-TOP Parameter Γ=8x8x1, 5x5x1, 8x8x2, 5x5x2
und T=10,15,20,30 für das Temporal Interpolation Model (TIM).
2: Erkenne Gesicht F_i im ersten Bild $p_{i,1}$.
3: Berechne h Merkmale $\Psi = \{(a_1, b_1)..(a_h, b_h)\}$ mittels Active-Shape-Model.
4: Normalisiere Gesicht F_i an das Modell, durch LWM Transformation $\zeta = LWM(\Psi, \omega, p_{i,1})$.
5: Erkenne Augen-Position $\{(x_{i,l}, y_{i,l}), (x_{i,r}, y_{i,r})\}$ und berechne die Distanz der Augen $\delta_i. = \sqrt{(\mathbf{x_{i,l}}, \mathbf{y_{i,l}}), (\mathbf{x_{i,r}}, \mathbf{y_{i,r}})}$.
6: Berechne Eckpunkte des Gesichts und schneide dieses aus. Setze
Oben-links=$(x_{i,l}, y_{i,l}) + 0.4 * (y_{i,l} - y_{i,r}) - 0.6(x_{i,r} - x_{i,l})$;
Breite=$1.8\delta_i$;
Höhe=$2.2\delta_i$.
7: Für alle $\theta \in T$ berechne TIM Bildsequenz $\boldsymbol{\xi^{syn}} = \boldsymbol{U}\ (\boldsymbol{\Upsilon}^{-1})^T\ \boldsymbol{M}\ \boldsymbol{\mathcal{F}}^n(t) + \overline{\boldsymbol{\xi}}$.
8: Für alle $p \in \Gamma, \theta \in T$ berechne $\mu_{i,p,\theta}(\xi_{i,\theta})$ LBP-TOP Merkmalsvektor.

2.5. Video-Magnification

Hao-Yu Wu et al. [19] entwickelten eine Methode kleine Bewegungen in einem Video zu verstärken. Dadurch werden Bewegungen in Videos sichtbar, die sonst für das menschliche Auge kaum sichtbar sind. In Kapitel 2.5.1 wird die Bewegungsverstärkung eines 1-dimensionalen Signals hergeleitet. In Kapitel 2.5.2 wird dann aufbauend auf die Herleitung eine Bewegungsverstärkung für ganze Videos beschrieben.

2.5.1. Herleitungen der Bewegungsverstärkung

Bei der Video-Magnification wird zunächst das originale Video in verschiedene Frequenzbänder zerlegt, auf die jeweils der gleiche Zeit-Bandpassfilter pixelweise angewendet wird. Hao-Yu Wu et al. [19] haben gezeigt, dass dadurch ein neues Video entsteht, indem die Bewegungen verstärkt sind. Es genügt den Fall eines 1-dimensionalen Signals darzustellen, da er auf ein mehrdimensionales Signal übertragbar ist. Betrachtet man nun eine Bewegung in einer Bildsequenz und sei $I(x,t)$ die Intensität eines Pixels am Ort x zum Zeitpunkt t, dann lassen sich die Intensitäten der Pixel zum Zeitpunkt $t = 0$ als $I(x,0) = f(x)$ beschreiben. Dieser Fall lässt sich durch eine Verschiebungsfunktion $\delta(t)$ auf

$$I(x,t) = f(x + \delta(t)) \tag{2.16}$$

verallgemeinern. Man erhält nun eine Bewegungsfunktion der Pixel-Intensitäten in Abhängigkeit der Zeit. Das Ziel ist im Folgenden die Bewegung $\delta(t)$ um den Faktor α zu verstärken, sodass gilt

$$\widehat{I}(x,t) = f(x + (1+\alpha)\delta(t)). \tag{2.17}$$

Diese Gleichung wird im Folgenden aus dem ursprünglichen Signal $I(x,t)$ hergeleitet. Angenommen, man könnte die Bewegungs-Gleichung 2.13 durch eine Taylor Reihe ersten Grades annähern, dann gilt

$$I(x,t) \approx f(x) + \underbrace{\delta(t)\frac{\partial f(x)}{\partial x}}_{B(x,t)} \tag{2.18}$$

Man hat jetzt die Bewegung durch $B(x,t)$ dargestellt. Wenn man nun einen Zeit-Bandpass Filter anwendet, der alle Bewegungen $B(x,t)$ durchlässt und den Rest $f(x)$ nicht, dann könnte man genau diese Bewegungen $B(x,t)$ um einen Faktor α verstärken und das verstärkte Signal dem Ursprungsbild zurückführen. Man erhält damit

$$\widehat{I}(x,t) = I(x,t) + \alpha B(x,t). \tag{2.19}$$

Zusammengefasst ergibt sich eine neue Gleichung, die genau diese Verstärkung um den Faktor α bewirkt.

$$\widehat{I}(x,t) \approx f(x) + \delta(t)\frac{\partial f(x)}{\partial x} + \alpha\delta(t)\frac{\partial f(x)}{\partial x} \tag{2.20}$$

$$\widehat{I}(x,t) \approx f(x) + (1+\alpha)\delta(t)\frac{\partial f(x)}{\partial x} \tag{2.21}$$

Wieder angenommen die Taylor-Reihen Entwicklung genügt für diese Berechnung, dann folgt die Behauptung aus Gleichung 2.14 als Approximation

$$\widehat{I}(x,t) \approx f(x + (1+\alpha)\delta(t)). \tag{2.22}$$

Damit erhält jede Bewegung eine Verstärkung um den Faktor α. An dieser Stelle sei noch erwähnt, dass die Taylor-Reihen Annäherung nicht notwendigerweise bei allen Raum-Frequenzen ausreichend ist, wie auch Hao-Yu Wu et al. [19] in ihrer Arbeit erwähnen, jedoch im allgemeinen für geringe Frequenzen eine gute Approximation ist.

2.5.2. Anwendung auf Videos

Übertragen auf Videos bzw. Bildsequenzen, zerlegt die Methode von Hao-Yu Wu et al. [19] das originale Video in verschiedene Raum-Frequenzbänder (in Abbildung 2.13 *Spatial Decomposition*) und wendet auf jedes Frequenzband pixelweise einen Zeit-Bandpassfilter an (in Abbildung 2.13 *Temporal Processing*). Für die Wahl eines Frequenzbereichs, entsteht bei diesem Prozess das Signal $B(x,t)$, das aus $I(x,t)$ gefiltert wird. $B(x,t)$ wird um den Faktor α verstärkt und die verstärkte Bewegung $\alpha B(x,t)$ wird auf das originale Signal aufaddiert, wodurch das neue Signal $\widehat{I}(x,t)$ entsteht. Im letzten Schritt werden alle Raum-Frequenzbänder wieder zu einem einzigen Video zusammengeführt.

Welche Frequenzen verstärkt werden ist daher abhängig von einem geeigneten Frequenzbereich des Zeit-Bandpassfilters und muss vorher gut gewählt werden, sodass auch nur Bewegungen verstärkt werden, die für die Anwendung interessant sind.

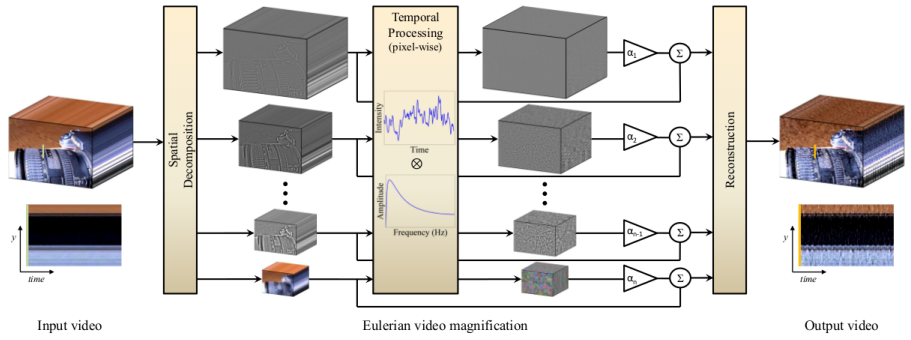


Abbildung 2.13.: Video-Magnification Prozess nach [19]. Das originale Video wird im ersten Schritt in Raum-Frequenzbänder (*Spatial Decomposition*) zerlegt. Im zweiten Schritt wird auf jedes Raum-Frequenzband dann pixelweise ein Zeit-Bandpassfilter *Temporal Processing* angewendet. Das durch den Zeitpass gefilterte Signal wird dann um den Faktor α verstärkt und dem originalen Signal zurückgeführt. Zuletzt werden dann wieder alle Raum-Frequenzbänder zu einem Video wieder zusammengeführt.

3. Implementierung

In diesem Kapitel werden Details zur Implementierung der Arbeit besprochen. Zuerst gibt es eine kurze Übersicht aller Bibliotheken, die für diese Arbeit verwendet werden. In Kapitel 3.2 wird der Aufbau der Software erläutert und die Reihenfolge beschrieben, in der die Methoden aus dem Grundlagen Kapitel 2 angewendet werden.

3.1. Verwendete Bibliotheken

OpenCV ist eine Open Source Computer Vision Bibliothek für C und C++. Sie unterstützt Methoden und Datenstrukturen für die Bildverarbeitung, maschinelles Lernen und lineare Algebra. Die Bibliothek kommt häufig in den Gebieten der Gesichtserkennung, Gestenerkennung, Objekterkennung und Kamerakalibrierung zum Einsatz. Von OpenCV werden sämtliche Bildverarbeitungsmethoden sowie auch die Implementierung einer Support Vector Machine verwendet.

Für die Arbeit werden hauptsächlich die mathematischen Operationen von Armadillo genutzt, weil die Singulärwertzerlegung von OpenCV für die Lösung von großen nichtquadratischen Matrizen zu lange dauert (nach ungefähr einer Stunde gab es noch kein Ergebnis). Armadillo löst das *economical* SVD Problem innerhalb weniger Millisekunden, auch für quadratische Matrizen. Armadillo ist eine reine lineare Algebra Programmbibliothek für C++ und unterstützt alle gängigen Methoden für Matrizen-Berechnung.

3.2. Programmaufbau

Die entwickelte Software besteht aus insgesamt 2 Teilen. Zunächst die Bildvorverarbeitung, die für die TIM-Generierung und später für die Video-Magnification zuständig ist (siehe Abbildung 3.1 links), und weiter die Evaluation (Abbildung 3.1 Rechts), in der die Erkennungsrate gemessen wird. Die Bildvorverarbeitung ist durch eine Konfigurationsdatei parametrisierbar, die beim Programmaufruf übergeben werden kann. Bei der Bildvorverarbeitung werden durch das Temporal Interpolation Model alle Bildsequenzen auf

jeweils fünf unterschiedliche Längen skaliert - Original, 10, 15, 20, 30 Bilder. Während der TIM-Generierung wird direkt die Merkmalsextraktion durchgeführt, sodass der Merkmalsvektor, der durch LBP-TOP berechnet wird, in einer Datei zeilenweise abgespeichert wird. Für jede Bildsequenzlänge wird eine separate Datei erstellt, in der zeilenweise alle Merkmalsvektoren enthalten sind. Ein Merkmalsvektor ist das Ergebnis aus der LBP-TOP Merkmalsextraktion von einer Bildsequenz. Da jeweils ein Merkmalsvektor (Zeile) zu einem bestimmten Probanden gehört, wird auch eine Index-Datei generiert, die zeilenweise den dazugehörigen Probanden 1,2,3,4,5,6 referenziert.

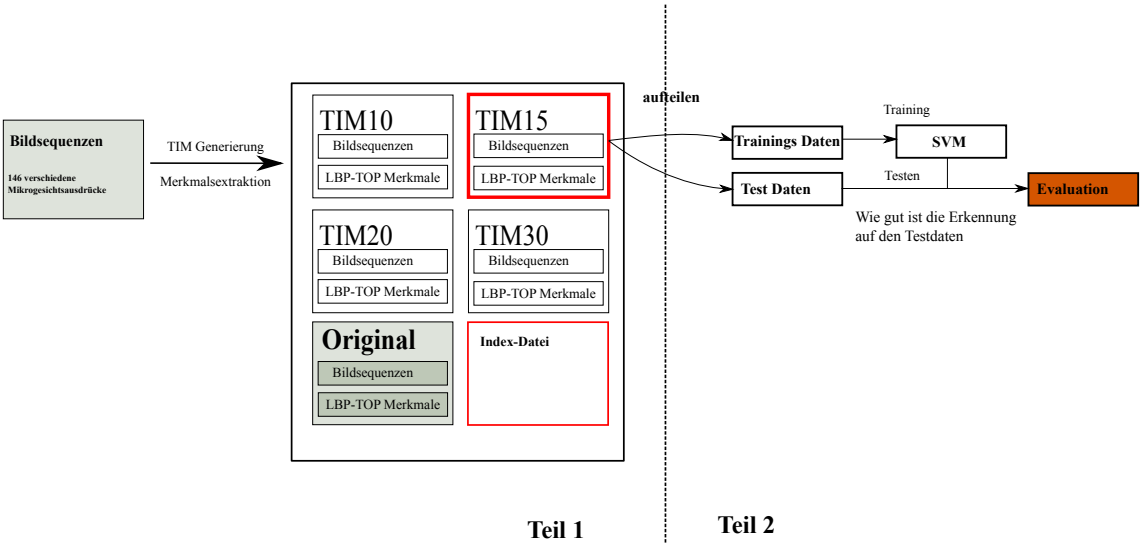


Abbildung 3.1.: Aufbau der entwickelten Software. Links die Merkmalsextraktion und rechts die Evaluation der generierten Daten.

3.2.1. TIM-Generierung

Bei diesem Teil der Arbeit wird Punkt 8 (die Interpolation) und Punkt 9 (die Merkmalsextraktion) in C++ mit Hilfe der Bibliotheken Armadillo und OpenCV umgesetzt. Als Eingabe wird der veröffentlichte SMIC Korpus von Pfister et al. [10] verwendet. Die Daten sind bereits gelabelt und die Gesichtsextraktion von Punkt 2-7 liegt bereits vor. Aus insgesamt 146 Mikrogesichtsausdrücken, die alle jeweils in unterschiedlichen Längen vorliegen, werden einheitliche Mikrogesichtsausdrücke auf die Länge 10,15,20 und 30 Bilder gebracht (TIM10, TIM15, TIM20, TIM30).

3.2.2. Merkmalsextraktion

Die Merkmalsextraktion wird genauso wie bei Pfister et al. [10] mit LBP-TOP realisiert. Um zu prüfen wie sich die Qualität des Algorithmus verhält, werden Experimente für die Blockgrößen $4 \times 4 \times 1$, $8 \times 8 \times 1$, $16 \times 16 \times 1$ durchgeführt. Für diese Experimente wird die Blockaufteilung im Zeitbereich konstant auf 1 belassen. Weitere Parameter sind, wie in Kapitel 2.2 bereits erläutert, der Radius für den Raum X-Y und Zeitdomäne T. Für die einzelnen Bildsequenzlängen Original, TIM10, TIM15, TIM20 und TIM30 wird eine separate Datei angelegt, die alle Merkmalsvektoren für das Training der Support Vector

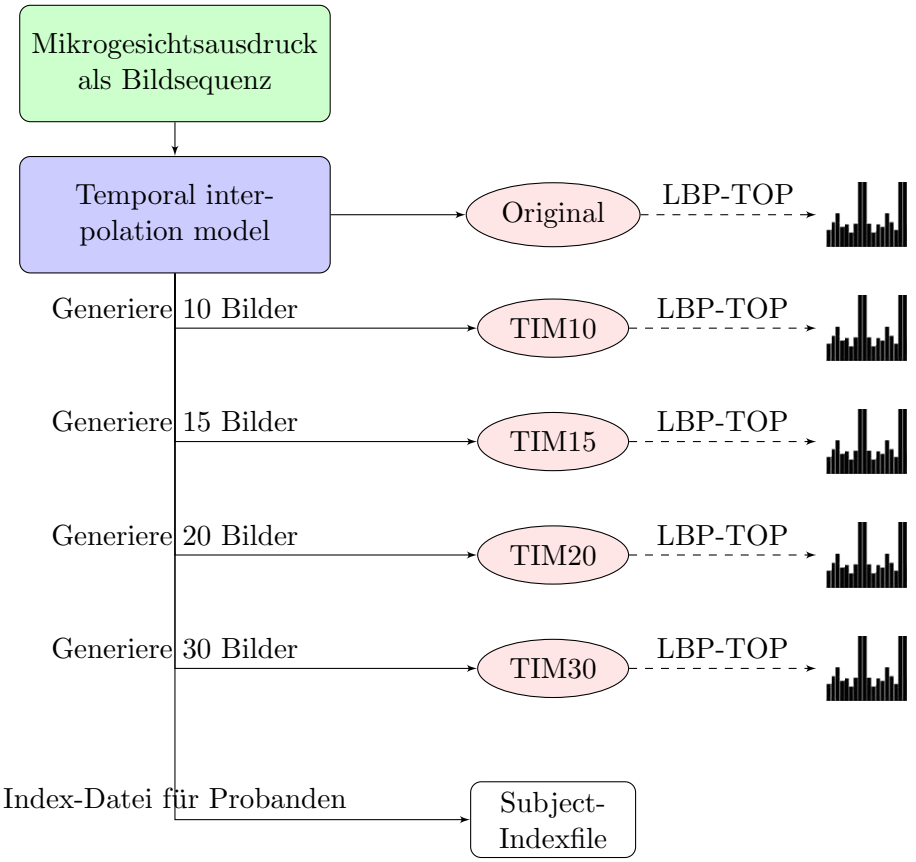


Abbildung 3.2.: Aufbau der Merkmalsextraktion. Die Mikrogesichtsausdrücke aus dem SMIC Korpus werden auf 4 feste Bildsequenzlängen interpoliert 10,15,20 und 30, die originale Bildsequenzlänge bleibt erhalten. Danach wird direkt eine LBP-TOP Merkmalsextraktion durchgeführt und in einer separaten Datei abgespeichert.

Machine umfasst. In einer Zeile befindet sich ein Merkmalsvektor, der aus der Bildsequenz durch die LBP-TOP Merkmalsextraktion ensteht.

Jede Bildsequenz wird beispielsweise in $8 \times 8 \times 1$ Blöcke aufgeteilt. Für jeden der 16 Blöcke wird jeweils zu jedem Pixel ein LBP-Code berechnet. Daraus wird dann im Anschluss ein Histogramm gebildet. Das Ergebnis aller Histrogramme verkettet ergibt einen langen Merkmalsvektor, der abhängig von der Bildsequenzlänge in der dazugehörigen Datei abgespeichert wird. In jeder Datei befinden sich damit insgesamt 146 Zeilen. Um später dann die Kreuzvalidierung zur Messung der Ergebnisse durchführen zu können, wird parallel ein Index für die Probanden (Subject-Index Datei) generiert. In dieser Datei steht, zu welchen Probanden (1 bis 6) der Datensatz gehört. Anhand dieser Index-Datei werden in einem weiteren Validierungsprozess alle Datensätze in Trainings- und Testdaten aufgeteilt, um die SVM zu trainieren und zu testen. Damit existieren nach der TIM-Generierung und der Merkmalsextraktion insgesamt 5 Dateien, welche jeweils die Merkmale zu den Bildsequenzen enthalten und dazu noch die Subject-Index Datei.

Im Folgenden ist ein Ausschnitt einer Trainingsdatei *train_data_tim10* mit den ersten Komponenten der Merkmalsvektoren zu sehen. Die voranstehende Zahl beschreibt die Klasse

der Mikrogesichtsausdrücke. Bei der Klasse 1 handelt es sich um Merkmalsvektoren, die zu Mikrogesichtsausdrücken gehören und -1, wenn es sich bei dem Merkmalsvektor um keinen Mikrogesichtsausdruck handelt. Jeder Komponente des Merkmalsvektors ist ein Index vorangestellt. Da die Histogramme normiert sind, liegen die Werte zwischen 0 und 1.

```
 1 1:0.0115546 2:0.00577731 3:0.00262605 4:0.0105042 5:0.0052521 ...
-1 1:0.012605 2:0.00682773 3:0.00472689 4:0.00892857 5:0.00472689 ...
-1 1:0.0162815 2:0.00630252 3:0.00420168 4:0.00997899 5:0.00262605 ...
...
```

Im Folgenden ist ein Ausschnitt der dazugehörigen Subject-Index Datei.

```
5
5
3
...
```

3.2.3. Parametrisierung der Merkmalsextraktion

Die TIM-Generierung bzw. die Merkmalsextraktion ist über eine externe Datei konfigurierbar. Sie wird dem Programm als Parameter übergeben. Folgende Einstellungen können unter anderem vorgenommen werden.

- Verzeichnis, für die neu generierten TIM Bildsequenzen
- Verzeichnis, aus dem die Bildsequenzen für die Klasse c_i geladen werden sollen
- Das zur Klasse c_i gehörende Label für die SVM. Für diese Arbeit -1 oder 1
- Angabe der TIM Sequenzen, die aus den Eingabedaten generiert werden sollen, mit Komma getrennt
- Blockaufteilung der Merkmalsextraktion
- Radius für die LBP-TOP Merkmalsextraktion

Im Anhang befindet sich ein Auszug A aus der Konfigurationsdatei und auch weitere Details zu den Parametern.

3.2.4. Training und Testverfahren

Das Trainings- und Testverfahren ist der Prozess, bei dem die Erkennungsrate gemessen und damit die Qualität des Klassifikators festgestellt wird. Beim Training lernt der Klassifikator anhand der Trainingsdaten, zwischen Mikrogesichtsausdrücken und ¬Mikrogesichtsausdrücken zu unterscheiden. Auf den Testdaten wird dann die Erkennungsrate des Klassifikators gemessen. Der Klassifikator versucht anhand der Daten die Klasse des Mikrogesichtsausdrucks vorherzusagen. Hierbei ist es wichtig, dass der Klassifikator diese Daten noch nicht kennt.

Wenn viele Daten zur Verfügung stehen, können diese in separate Trainings- und Testdaten aufgeteilt werden. Für dieses Verfahren sollten jedoch genug Daten vorliegen, um repräsentative Datenmengen in den Trainings- und Testdaten zu haben. Für den Fall, dass nicht genug Daten vorliegen, kann man eine Kreuzvalidierung durchführen, wie es auch Pfister et al. [10] getan haben. Dies wird nach dem *Leave-One-Subject-Out* Schema durchgeführt. Hierbei bestehen die Trainingsdaten aus dem gesamten Korpus bis auf die Datensätze eines beliebigen Probanden. An den Datensätzen dieses Probanden wird die Erkennungsrate gemessen. Dies passiert genauso oft wie es insgesamt Probanden gibt. Der Mittelwert aus diesen Werten ergibt die Gesamt-Erkennungsrate. In diesem Fall ist $n = 6$ die Probanden-Zahl. Insgesamt gibt es 146 Mikrogesichtsausdrücke, die zu diesen 6 Probanden gehören. Damit ergeben sich 6 Messwerte, aus denen der Mittelwert gebildet wird.

4. Evaluation

In diesem Kapitel werden die Ergebnisse der Experimente aus Kapitel 3 erläutert. Dabei soll diskutiert werden, wie sich die Ergebnisse der originalen Mikrogesichtsausdrücke im Vergleich zu den durch die TIM-Generierung erstellten Mikrogesichtsausdrücke unterscheiden. In Kapitel 4.1 wird erläutert, wie sich die Erkennungsrate verhält, wenn man den Radius der LBP-TOP Merkmalsextraktion verändert. Im anschließenden Kapitel wird die Erkennungsrate im Verhältnis zur Blockaufteilung der Bildsequenz untersucht. In Kapitel 4.2 wird dann die Video-Magnification eingeführt und mit den Ergebnissen aus Kapitel 4.1 verglichen. Die Video-Magnification wird der TIM-Generierung und damit der Merkmalsextraktion vorangestellt. Für die Evaluation beschränkt sich diese Arbeit auf eine Support Vector Machine, um die Veränderung des Algorithmus durch die Einführung der Video-Magnification zu zeigen. Die Support Vector Machine ist derzeit ein weit verbreiteter Klassifikator, der anhand geeigneter Kernel-Funktionen auf beliebigen Daten trainiert werden kann. Pfister et al. haben in ihrer Arbeit bestmögliche Ergebnisse mit anderen Konfigurationen erhalten, was in Kapitel 4.1.4 kurz diskutiert wird.

4.1. Evaluation von LBP-TOP

Zunächst soll untersucht werden, wie sich die Erkennungsrate im Bezug auf den Radius und der Blockaufteilung der LBP-TOP Merkmalsextraktion verhält.

4.1.1. Variation des LBP-TOP Radius

Bei der Blockaufteilung wird das Bild in $B \times H$ Blöcke aufgeteilt, für die jeweils ein Histogramm berechnet wird. Pfister et al. [10] wählten für ihre SVM Tests eine Blockaufteilung von 8x8, was auch in dieser Arbeit weiter untersucht werden soll. Für die Konfiguration des LBP-TOP Radius $R = 1$ wurden in allen beschriebenen Experimenten die besten Ergebnisse erzielt. Daher wird in allen folgenden Experimenten auch diese Einstellung verwendet. Tabelle 4.1 enthält die Ergebnisse, wie sich die Erkennungsrate abhängig zum Radius R verhält.

Bei diesem Experiment lässt sich kein Trend erkennen, wonach durch die TIM-Generierung eine Verbesserung der Erkennungsrate erzielt wird. Die Erkennungsrate schwankt stark

Interpolation	Erkennungsrate(%) für R=3	Erkennungsrate(%) für R=2	Erkennungsrate(%) für R=1
Original	49.3	49.3	63.7
TIM10	**57.5**	50.0	**64.4**
TIM15	51.4	**52.7**	63.0
TIM20	50.7	52.1	63.0
TIM30	49.3	49.3	62.3

Tabelle 4.1.: TIM steht für Temporal Interpolation Model. Die Zahl hinter TIM steht für die Anzahl der Bilder. Vergleich verschiedener LBP-TOP Radii und TIM Konfigurationen. Trainiert wurde eine Support Vector Machine.

zwischen 50% und 65% und erzielt für den Parameter $R = 1$ die besten Ergebnisse. Pfister et al. [10] beschreiben auch in ihrer Arbeit, dass die Performance stark von den Eingabedaten und dem Klassifikationsverfahren abhängt. Das Ergebnis zeigt, dass die Merkmalsextraktion ein wichtiger Faktor für die Erkennungsrate darstellt.

4.1.2. Variation der LBP-TOP Blockgrößen

Das nächste Experiment zeigt, wie sich die Erkennungsrate verändert, in dem die Blockaufteilung variiert wird. Die Ergebnisse des Experiments sind in Tabelle 4.2 aufgeführt.

Interpolation	4x4x1, R=1	5x5x1,R=1	6x6x1,R=1	8x8x1,R=1	10x10x1,R=1
Original	65.8	64.4	**64.3**	63.7	54.8
TIM10	**67.1**	**65.0**	61.6	**64.4**	52.7
TIM15	62.3	61.6	63.0	63.0	54.8
TIM20	65.7	64.4	63.0	63.0	53.4
TIM30	64.4	64.4	62.3	62.3	54.8

Tabelle 4.2.: TIM steht für Temporal Interpolation Model. Die Zahl hinter TIM steht für die Anzahl der Bilder. Vergleich verschiedener LBP-TOP Blockgrößen und TIM Konfigurationen. Trainiert wurde eine Support Vector Machine.

Die Performance des Erkenners sinkt mit wachsender Blockaufteilung. Das Temporal Interpolation Model liefert in den meisten Fällen eine Verbesserung von 1-2 Prozentpunkte. Die besten Ergebnisse wurden bei der Blockaufteilung $4 \times 4 \times 1$, $5 \times 5 \times 1$, $6 \times 6 \times 1$, $8 \times 8 \times 1$ erzielt. Versuche zeigten, dass bei einer Blockaufteilung bei mehr als $10 \times 10 \times 1$, die Erkennungsrate weiter sinkt. Die durch das Temporal Interpolation Model generierten Daten scheinen im Vergleich zur Blockaufteilung weniger Einfluss auf die Qualität der Erkennungsrate zu haben.

In Kapitel 4.2 sollen durch einen neuen Ansatz, die Video-Magnification, die Bildsequenzen in der Art verändern, sodass die LBP-TOP Merkmalsextraktion beeinflusst wird. Die Video-Magnification verstärkt kleine Bewegungen innerhalb der Bildsequenz. Diese Verstärkungen führen dazu, dass sich die Weißintensitäten in den Bereich verändern, in denen Bewegungen auftreten. Dies wirkt sich dann auf die LBP-TOP Merkmalsextraktion aus.

4.1.3. Vergleich mit den Ergebnissen von Pfister et al.

In der Tabelle 4.3 sind die Ergebnisse für den SMIC Korpus von Pfister et al. [10] zusammengefasst. Die Daten stammen von einer 100fps Kamera.

Ergebnisse	Klassen	Methode	Erkennungsrate (%)
von Pfister	Mikro/¬Mikro	RF+TIM15	67.7
	Mikro/¬Mikro	SVM	70.3
	Mikro/¬Mikro	RF+TIM20	70.3
	Mikro/¬Mikro	MKL	71.4
	Mikro/¬Mikro	RF+TIM10	74.3
von dieser Arbeit	Mikro/¬Mikro	SVM+TIM10 (4x4x1)	67.1
	Mikro/¬Mikro	SVM+TIM10 (5x5x1)	65.0
	Mikro/¬Mikro	SVM+TIM10 (8x8x1)	65.0

Tabelle 4.3.: Vergleich der Ergebnisse von Pfister et al. und den Ergebnissen dieser Arbeit. Support Vector Maschine (SVM), Random Forst (RF), Multi-Kernel-Learning (MKL) sind Klassifikatoren. TIMn steht für die Interpolation der Eingabedaten auf eine Bildsequenz Länge von n Bildern. Bei der Klassifikation Mikro/¬Mikro geht es darum einen Mikrogesichtsausdruck von einem ¬Mikrogesichtsausdruck zu unterscheiden.

Pfister et al. haben ihre besten Ergebnisse mit der Konfiguration $R = 3$ erhalten. Sie haben auch in ihrem Paper andere Erkennungsraten für das SVM-Training erhalten. Dies liegt vermutlich daran, dass Pfister et al. eine andere Support Vector Machine Implementierung verwendeten.

In dieser Arbeit wurden die besten Ergebnisse mit einem Radius von $R = 1$ und einer Blockaufteilung von $4 \times 4 \times 1$ erhalten. Ähnlich gute Ergebnisse lieferten die Blockaufteilungen von $5 \times 5 \times 1$ und $8 \times 8 \times 1$.

4.2. Evaluation der Video-Magnification

Idee ist, die Video-Magnification als weiteren Vorverarbeitungsschritt zu verwenden, sodass Bewegungen, die durch Mikrogesichtsausdrücke verursacht wurden, verstärkt werden. In Kapitel 4.2.1 wird anhand eines Beispiels gezeigt, wie sich ein spezieller Mikrogesichtsausdruck durch die Video-Magnification verändert und welchen Einfluss dies auf die Merkmalsextraktion hat. In Kapitel 4.2.2 werden dann Seiteneffekte der Video-Magnification erläutert und abschließend die Auswirkung auf die LBP-TOP Merkmalsextraktion diskutiert.

4.2.1. Anwendung auf Mikrogesichtsausdrücke

Die Video-Magnification soll zur Vorverarbeitung des gesamten SMIC Korpus von Pfister et al. [10] verwendet werden, um die Bewegungen zu verstärken. Für diesen Prozess musste zuerst festgestellt werden, in welchem Frequenzbereich die Mikrogesichtsausdrücke liegen. Bewegungen innerhalb eines Videos sind Intensitätsveränderungen, die durch Frequenzen

beschrieben werden können. Diese Frequenzen erhält man, indem eine Transformation in den Fourier-Raum durchgeführt und anschließend ein Bandpassfilter angewendet wird. Die Hauptaufgabe bestand darin geeignete Parameter zu finden, die Mikrogesichtsausdrücke in ihrer Ausprägung sinnvoll verstärken, sodass nach der Anwendung der Video-Magnification die Datensätze zum Training der Support Vector Machine weiter verwendet werden können. Durch Modulation der Parameter $\alpha, \lambda_c, \omega_l$ und ω_h ergab sich eine bestimmte Konfiguration, bei der sich die meisten Mikrogesichtsausdrücke in der Form veränderten, sodass Bewegungen nach der Video-Magnification sichtbarer wurden. Abbildung 4.2 zeigt ein Beispiel eines Mikrogesichtsausdrucks nach der Anwendung der Video-Magnification im Vergleich zum originalen Mikrogesichtsausdruck in Abbildung 4.1

Abbildung 4.1.: Orignialer Mikrogesichtsausdruck aus dem SMIC Korpus von Pfister et al. [10]

Abbildung 4.2.: Mikrogesichtsausdruck, der durch die Video-Magnification verstärkt wurde. Die Einstellungen waren ein Temporal IIR Filter mit $\alpha = 10$, $\lambda_c = 16$, $\omega_l = 0.4$, $\omega_h = 0.05$ und $chromAttenuation = 0.1$. Siehe [19] für weitere Details.

Nach der Anwendung der Video-Magnification auf alle Mikrogesichtsausdrücke wurde der veränderte Korpus erneut zum Training der Support Vector Machine verwendet. Die Ergebnisse sind in Tabelle 4.4 aufgeführt.

Interpolation	6x6x1,R=1	8x8x1,R=1	10x10x1,R=1
Original	**67.1**	68.8	**66.4**
TIM10	63.7	66.4	64.4
TIM15	66.4	67.8	63.7
TIM20	65.1	**69.1**	65.1
TIM30	62.3	67.1	63.7

Tabelle 4.4.: In dieser Tabelle werden verschiedene LBP-TOP Blockgrößen und TIM Konfigurationen bei Nutzung der Video-Magnification verglichen. TIM steht für Temporal Interpolation Model. Die Zahl hinter TIM steht für die Anzahl der Bilder pro Mikrogesichtsausdruck.

Im Vergleich zum Versuch aus Kapitel 4.1.2, bei dem eine Blockaufteilung von $10 \times 10 \times 1$ zu einer Erkennungsrate von nur 50% führte, fällt auf, dass die Erkennungsrate jetzt auch bei größeren Blockaufteilungen konstant bleibt. Die Erkennungsrate liegt für $8 \times 8 \times 1$ bei allen Testfällen bei über 65% im Vergleich zum Versuch ohne Video-Magnification. Die Ergebnisse deuten darauf hin, dass die Video-Magnification in der Lage sein kann, das

SVM Training zu stabilisieren, sodass die Erkennungsrate unanfälliger für die Parameter der LBP-TOP Merkmalsextraktion ist. Weiterhin zeigt das Ergebnis mit 69.1% TIM20, dass sogar der Algorithmus durch eine Bildvorverarbeitung weiter verbessert werden kann.

4.2.2. Seiteneffekte bei der Video-Magnification

Insgesamt wirken die Ergebnisse vielversprechend. Jedoch gibt es Probleme, sobald sich im Bild nicht nur der Mikrogesichtsausdruck bewegt, sondern sich auch der Kopf selbst bewegt. Diese Bewegungen werden dann genauso verstärkt, wie es beispielsweise bei dem folgenden Mikrogesichtsausdruck der Fall ist.

Abbildung 4.3.: Mikrogesichtsausdruck, der durch die Video-Magnification verstärkt wurde. In diesem Beispiel hat sich der Kopf während dem Mikrogesichtsausdruck auch bewegt. Die Einstellungen waren ein Temporal IIR Filter mit $\alpha = 10$, $\lambda_c = 16$, $\omega_l = 0.4$, $\omega_h = 0.05$ und $chromAttenuation = 0.1$. Siehe [19] für weitere Detais.

Die Video-Magnification fängt an, die Bewegung des Kopfes ab dem zweiten Bild zu verstärken, sodass der gesamte Kopf während der Bildsequenz leuchtet. Zur Mitte der gesamten Bildsequenz des Mikrogesichtsausdrucks klingt die Kopfbewegung ab und es bleibt eine Bewegungsverstärkung der Nase zu erkennen.

Eine Lösung für das Problem könnte sein, die Kopfbewegung mitzuverfolgen und das Gesicht abhängig der Bewegung auszuschneiden. Man könnte auch das Gesicht auf ein 3D-Modell projizieren das ohne Kopfbewegung als gerenderte Bildsequenz verwendet wird. Der Algorithmus von Pfister et al. generiert in Schritt 6 der Merkmalsextraktion (siehe Algorithmus 2) durch das Active-Shape-Model solche Ausschnitte des Gesichts. Das Experiment der Video-Magnification deutet darauf hin, dass diese generierten Gesichtsausschnitte nicht sehr genau sind. Genauso wäre denkbar, Bewegungen des Kopfes und die Bewegungen von Mikrogesichtsausdrücken zu analysieren, sodass die Kopfbewegung eben nicht durch die Video-Magnification verstärkt wird, sondern nur der Mikrogesichtsausdruck selbst.

Die Video-Magnification scheint Bewegungen in der Bildsequenz so zu verstärkten, dass sich der Weiß-Anteil stärker ausprägt. Im nächsten Beispiel des folgenden Mikrogesichtsausdrucks handelt es sich um ein Blinzeln zusammen mit einem Zucken des linken Auges.

In Abbildung 4.5 ist die Veränderung der Weiß-Intensität während der Augenbewegung in Bild 6-10 deutlich zu erkennen. Genauso wird die Bewegung des linken Auges in Bild 2-6 durch die Video-Magnification durch eine erhöhte Weiß-Intensität sichtbar. Die LBP-TOP Merkmalsextraktion wird durch solche Veränderungen der Weiß-Intensitäten stark beeinflusst, weil solche Effekte die Berechnung der LBP-Codes verändern. Diese LBP-Codes werden dann im Histogramm für den Klassifikator sichtbar. Im Vergleich zu den

Abbildung 4.4.: Originaler Mikrogesichtsausdruck aus dem SMIC-Korpus, vor der Video-Magnification.

Abbildung 4.5.: Mikrogesichtsausdruck, die durch die Video-Magnification verstärkt wurde. Die Einstellungen waren ein Temporal IIR Filter mit $\alpha = 10$, $\lambda_c = 16$, $\omega_l = 0.4$, $\omega_h = 0.05$ und $chromAttenuation = 0.1$. Siehe [19] für weitere Details.

originalen Bilddaten in Abbildung 4.4 tragen die Bilder 6-10 besonders stark zur LBP-Codes Berechnung bei, was insgesamt die Ergebnisse aus Tabelle 4.4 erklären könnte.

5. Zusammenfassung und Ausblick

In Kapitel 1 wurden kurz Mikrogesichtsausdrücke und ihre Bedeutung dargestellt. Paul Ekman trug maßgeblich zur Erforschung von Mikrogesichtsausdrücken bei, indem er insbesondere die Details zu den psychologischen Rahmenbedingungen untersuchte und unter welchen Bedingungen sie auftreten. Dies ermöglichte Pfister et al. [10] einen realistischeren Korpus zu generieren, den Spontaneous Micro-expression Corpus (SMIC).

In Kapitel 2 wurden dann die technischen Grundlagen erläutert, wie Pfister et al. [10] die Bildsequenzlänge von Mikrogesichtsausdrücken auf eine einheitliche Länge brachten, indem sie das Temporal Interpolation Model verwendeten. In ihrer Arbeit haben sie alle Mikrogesichtsausdrücke auf eine bestimmte Bilderanzahl vereinheitlicht – 10 Bilder (TIM10), 15 Bilder (TIM15), 20 Bilder (TIM20) und 30 Bilder (TIM30) –, bevor sie diese für Merkmalsextraktion und die Klassifikation verwendeten. Die gleichen Konfigurationen wurden auch für diese Arbeit verwendet.

Im Anschluss wurden die Grundlagen zur Local Binary Pattern und LBP-TOP Mermalsextraktion beschrieben. Für die Klassifikation der Mikrogesichtsausdrücke wurde eine Support Vector Machine verwendet, deren Grundlagen in Kapitel 2.3 näher erläutert wurde. Abschließend wurden im Kapitel 2 noch wichtige Gleichungen für die Video-Magnification hergeleitet, um Bewegungen innerhalb eines Videos zu verstärken. Diese Methode wurde verwendet, um die Mikrogesichtsausdrücke vorzuverarbeiten und dadurch die minimalen Bewegungen der Mikrogesichtsausdrücke in den Bildern sichtbar zu machen.

In Kapitel 3 wurde weiter der Aufbau der entwickelten Software erläutert, um den Algorithmus von Pfister et al. zu reimplementieren. Die besten Ergebnisse wurden mit $R = 1$ für die LBP-TOP Merkmalsextraktion und einer Blockaufteilung von $8 \times 8 \times 1$ erzielt. Die Performance des Klassifikators war stark abhängig von den Eingabedaten und erzielte je nach Konfiguration quantitativ stark schwankende Ergebnisse. Abschließend wurde in Kapitel 3 der Algorithmus von Pfister et al. durch die Methode der Video-Magnification erweitert [19]. Angewendet wurde die Methode auf den gesamten SMIC Korpus, sodass bei allen Mikrogesichtsausdrücken Bewegungen für einen bestimmten Frequenzbereich verstärkt wurden. Hierbei hat sich herausgestellt, dass bei diesem Prozess Bewegungen nicht

nur in ihrer räumlichen Ausprägung, sondern auch in ihrer Lichtintensität verstärkt werden. Genau diesen Effekt haben Hao-Yu Wu et al. [19] in ihrer Arbeit beschrieben, dass eine Farbverstärkung auch eine Bewegungsverstärkung sein kann. Diese Veränderungen der Lichtintensitäten in der Bildsequenz führten dazu, dass bei der LBP-TOP Merkmalsextraktion die LBP-Codes verändert wurden, die maßgeblich zur Bildung der Merkmalsvektoren beitrugen. Mittels der durch die Video-Magnification veränderten Daten konnte die Erkennungsrate nochmal um einige Prozentpunkte auf 69% verbessert werden. Besonders bemerkenswerte Ergebnisse lieferte die Video-Magnification bei der Variation der Blockaufteilung. Die Erkennungsrate blieb für die untersuchten Blockaufteilungen nahezu konstant bei 66-68% während ohne Video-Magnification die Erkennugsrate bei einer Blockaufteilung von $10 \times 10 \times 1$ auf 50% zurückging. Die Video-Magnification führte allerdings auch zu nicht vorgesehenen Ergebnissen, wie beispielsweise die Verstärkung von Kopfbewegungen in der Bildfolge. Weil sich Bewegungen in einer Bildsequenz auch durch verstärkte Weiß-Intensitäten bemerkbar machen, wurde der gesamte Kopf so verstärkt, dass der Mikrogesichtsausdruck während der Kopfbewegung nicht mehr zu erkennen war. Zur Lösung dieses Problems fehlt noch eine geeignete Methode.

Derzeit beinhaltet der SMIC Korpus wohl noch zu wenige Mikrogesichtsausdrücke, um aussagekräftige Ergebnisse zu liefern. Dementsprechend wäre ein größerer Korpus mit Mikrogesichtsausdrücke von mehreren Probanden wünschenswert. Denkbar wäre einen neuen Korpus zu entwickeln unter Nutzung von gelabelten Pokerfilmen. Poker ist eine Disziplin, die die Rahmenbedingungen von Ekman gewährleisten könnte, sodass potenziell Mikrogesichtsausdrücke beobachtet werden können. Dies bedarf aber einer weiteren Untersuchung.

Weil bei der Video-Magnification nur Bewegungen für bestimmte Frequenzen verstärkt werden und jede Bewegung eine andere Frequenz aufweist, müsste man weiter nach Frequenzbereiche suchen, um gezielt Bewegungen von Mikrogesichtsausdrücken anhand ihrer charakterisierenden Frequenzen zu verstärken. Hieraus könnte sich ein Testverfahren für unterschiedliche Arten von Mikrogesichtsausdrücken entwickelt werden.

Literaturverzeichnis

[1] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *Neural Information Processing Systems*, pages 585–591, 2001.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] Paul Ekman. *Telling Lies: Clues to Deceit in the Marketplace, Politics, and Marriage*. Norton, New York, 1985.

[4] Paul Ekman. Basic emotions. In T. Dalgleish and T. Power (Eds.) The handbook of cognition and emotion. pages 45–60, 1999.

[5] John Gottman. FAQ About the Gottman study. `http://www.gottman.com/49853/Research-FAQs.html`. [Online; accessed 14-Dezember-2012].

[6] Paul Ekman Group. F.a.c.e. training. `http://face.paulekman.com`, 2012. [Online; accessed 14-Dezember-2012].

[7] Xiaofei He, Deng Cai, Shuicheng Yan, and HongJiang Zhang. Neighborhood Preserving Embedding. In *International Conference on Computer Vision*, pages 1208–1213, 2005.

[8] John Mordechai Gottman, Robert Wayne Levenson. A Two-Factor Model for Predicting When a Couple Will Divorce: Exploratory Analyses Using 14-Year Longitudinal Data. *Family process*, 41(1):83–96, 2002.

[9] Tomas Pfister, Xiaobai Li, Guoying Zhao, and Matti Pietikäinen. Differentiating spontaneous from posed facial expressions within a generic facial expression recognition framework. In *International Conference on Computer Vision Workshops*, pages 868–875, 2011.

[10] Tomas Pfister, Xiaobai Li, Guoying Zhao, and Matti Pietikäinen. Recognising spontaneous facial micro-expressions. In *International Conference on Computer Vision*, pages 1449–1456, 2011.

[11] Rakesh Kaundal, Amar S. Kapoor and Gajendra P.S. Raghava. Machine learning techniques in disease forecasting: a case study on rice blast prediction. `http://www.imtech.res.in/raghava/rbpred/algorithm.html`.

[12] S. Polisovsky, Y. Kameda, Y. Ohta. Facial Micro-Expressions Recognition Using High Speed Camera and 3D-Gradients Descriptor. In *Imaging Crime Detection and Prevention*, pages 1–6, 2009.

[13] Scholarpedia. Local Binary Patterns. `http://www.scholarpedia.org/article/Local_Binary_Patterns`, 2012. [Online; accessed 14-Dezember-2012].

[14] Matthew Shreve, Sridhar Godavarthy, Dmitry B. Goldgof, and Sudeep Sarkar. Macro- and micro-expression spotting in long videos using spatio-temporal strain. In *Automatic Face and Gesture Recognition and Workshops*, pages 51–56, 2011.

[15] Matthew Shreve, Sridhar Godavarthy, Vasant Manohar, Dmitry B. Goldgof, and Sudeep Sarkar. Towards macro- and micro-expression spotting in video using strain patterns. In *Applications of Computer Vision*, pages 1–6, 2009.

[16] Jascha Wetzel. Local Binary Patterns Ausarbeitung für Hauptseminar Medizinische Bildverarbeitung. In *Aachener Schriften zur Medizinischen Informatik*, volume 2, 2007.

[17] Wikipedia. Support vector maschine (de). `http://de.wikipedia.org/wiki/Support_Vector_Machine`. [Online; accessed 14-December-2012].

[18] W. D. William S. Condon, OGSTON. Sound Film Analysis of Normal and Pathological Behavior Patterns. *Journal of Nervous and Mental Disease*, 143(4).

[19] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John V. Guttag, Frédo Durand, and William T. Freeman. Eulerian video magnification for revealing subtle changes in the world. *Association for Computing Machinery Trans. Graph.*, 31(4):65, 2012.

[20] Shuicheng Yan, Dong Xu, Benyu Zhang, and HongJiang Zhang. Graph Embedding: A General Framework for Dimensionality Reduction. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 830–837, 2005.

[21] Guoying Zhao and Matti Pietikäinen. Dynamic Texture Recognition Using Local Binary Patterns with an Application to Facial Expressions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):915–928, 2007.

[22] Ziheng Zhou, Guoying Zhao, and Matti Pietikäinen. Synthesizing a talking mouth. In *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 211–218, 2010.

[23] Ziheng Zhou, Guoying Zhao, and Matti Pietikäinen. Towards a practical lipreading system. In *Conference on Computer Vision and Pattern Recognition*, pages 137–144, 2011.

Anhang

A. Implementierung - Temporal Interpolation Model

```
namespace microexp {

  struct TIM_model_t {
    // list of eigenvektoren T=(v_k) = v1|v2|v2|..
    // L(Q'*v_k) = lambda_k*(Q'*v_k)

    arma::mat T; // sorted eigenvectors (based on eigenvalue sortion)
    arma::mat U; // X=U*S*V'
    arma::mat M; // scalar matrix (for
    arma::mat mu;  // median of inputframes
    arma::mat E;
    int n;     // frame count
    bool single_frame;
    bool valid;
  };


  class TIM
  {
    private:
       TIM_model_t model;

       arma::mat project( float t);
    public:
      TIM();
      ~TIM();

      bool createModel( arma::mat imv );
      arma::mat synthesize( vector<float> t);


      static vector<float> genTimeSequence(int from, int to, int n);
  }; // class TIM


} // namespace microexp
#endif
```

```
bool microexp::TIM::createModel( arma::mat imv )
{
  model.single_frame = false;
  model.valid = false;


  int n = imv.n_cols;
  int img_length = imv.n_rows;

  arma::mat W = arma::zeros(n,n);
  arma::mat D = arma::zeros(n,n);
  arma::mat L = arma::zeros(n,n);

  // compute mean vector mu of vectorised images
  arma::mat mu = arma::zeros(img_length,1);
  for( int j=0; j < img_length; j++ ){
    for( int i=0; i < n; i++ ){
      mu.at(j,0) += imv.at(j,i);
    }
    mu.at(j,0) /= n;
  }

  // X_ = imv - mean
  arma::mat X = arma::zeros(img_length,n);
  for( int i=0; i < n; i++ ){
    for( int j=0; j < img_length; j++ ){
      X.at(j,i) = imv.at(j,i)- mu.at(j,0);
    }//for
  }//for

  // define W
  for( int i=1; i < n; i++ ){
    W.at( i,i-1 ) = 1;
    W.at( i-1,i ) = 1;
  }//for

  // define D
  for( int i=0; i < n; i++ ){
    for( int j=0; j < n; j++ ){
      D.at(i,i) += W.at(i,j);
    }//for
  }//for
  // compute laplacian matrix L
  L = D-W;

  bool is_zero=true;
  for( int x=0; x < X.n_cols; x++ ){
    for( int y=0; y < X.n_rows; y++ ){
      if( X.at(y,x) != 0 ) { is_zero=false;break;}
    }//for
    if(is_zero==false){break;}
  }//for
```

```
if( is_zero ){
  model.single_frame = true;
  model.mu = mu;
  model.valid = true;
  return true;
}

arma::mat U, V;
arma::vec s;

// SVD decomposition X = U*s*V'
arma::svd_econ(U, s, V, X );
arma::mat S = arma::diagmat(s);

U.resize( U.n_rows, U.n_cols-1 );
S.resize( S.n_rows-1, S.n_cols-1 );
V.resize( V.n_rows, V.n_cols-1 );

arma::mat Q = S*V.t();

if( arma::det( (Q*Q.t())*(Q*L*Q.t()) ) == 0){
  printf("\n: error image sequence, determinate == 0 ");
  return false;
}

arma::mat A = arma::inv(Q*Q.t())*(Q*L*Q.t());

arma::mat V_;
arma::mat lambda_;

arma::cx_vec vx_eigval;
arma::cx_mat vx_eigvec;

arma::eig_gen( vx_eigval, vx_eigvec, A );
arma::vec eigval = arma::conv_to<arma::vec>::from(vx_eigval);
arma::mat eigvec = arma::conv_to<arma::mat>::from(vx_eigvec);

arma::uvec eigval_sorted = arma::sort_index(eigval);

arma::mat T(eigvec.n_rows, eigvec.n_cols);
for( int i=0; i < eigval_sorted.n_rows; i++ ){
  T.col(i) = eigvec.col(eigval_sorted.at(i));
}//for

arma::mat y(T.n_rows, 1);
for( int k=0; k < y.n_rows; k++) {
  y.at(k,0) = sin(M_PI*(double)(k+1)*(double)1/n + M_PI*(n-(double)(k+1))/(
      double)(2*n) );
}//for

arma::mat M = (Q.col(0).t()*T).t() / y;

model.U = U;
```

```
  model.T = T;
  model.M = M;
  model.mu = mu;
  model.n = n;

  model.valid = true;

  return true;
}

arma::mat microexp::TIM::project( float t){
  arma::mat Fn(model.n-1,1);
  for( int k=0; k < model.n-1; k++ ){
    Fn.at(k,0) = sin(M_PI*(double)(k+1)*t + M_PI*((double)model.n-(double)(k+1))
        /((double)2*model.n) );
  }
  return Fn;
}

arma::mat microexp::TIM::synthesize( vector<float> t)
{
  if( false == model.single_frame ){
    arma::mat imseq;
    imseq.set_size(model.mu.n_rows, (int)t.size() );

    arma::mat trans=(model.U*arma::inv(model.T).t()*arma::diagmat(model.M));
    for( int frame=0; frame < t.size(); frame++ ){
      imseq.col(frame) = trans*project(t.at(frame))+model.mu;
    }//for
    return imseq;

  } else {
    arma::mat imseq;
    imseq.set_size(model.mu.n_rows, (int)t.size() );
    for( int frame=0; frame < t.size(); frame++ ){
      imseq.col(frame) = model.mu;
    }//for
    return imseq;

  }
}


vector<float> microexp::TIM::genTimeSequence(int from, int to, int n){
  vector<float> t;
  for( int i=from; i <= to; i++ ){
    float time = (float)i/(float)n;
    t.push_back( time );
  }//for
  return t;
}
```

```
arma::mat imv = loadImages(src, width, height, channels);
if( imv.n_cols > 0 ){
  TIM tim;

  // create model for image sequence
  tim.createModel(imv);

  // compute new image sequence with 30 images
  synim = tim.synthesize( TIM::genTimeSequence(1, 30, 30) );

  saveImages(synim, "/imageroot/", "frame%i.bmp", width, height );
}
```

B. Konfiguration - TIM-Generierung

```
[general]
train_dir=../pfad-zu-TIM-Generierung

[classify]
class1_dir=../pfad-zu-klasse1
class2_dir=../pfad-zu-klasse2

[labels]
class1_label=1
class2_label=-1

[TIM]
tim_sequences = 0,10,15,20,30

[LBP_TOP]
blocks_x = 8
blocky_y = 8
blocks_t = 1
radius_x=1
radius_y=1
border_length=1
interval_t=1
time_length=1
```

Parameter	**Beschreibung**
train_dir	Verzeichnis, für die neu generierten TIM Bildsequenzen
classX_dir	Verzeichnis, aus dem die Bildsequenzen für die Klasse X geladen werden sollen.
classX_label	Das zur Klasse X gehörende Label für die SVM. Für diese Arbeit -1 oder 1.
tim_sequences	Angabe der TIM-Sequenzen, die aus den Eingabedaten generiert werden sollen. Mit Komma getrennt. TIM0 bezeichnet die originale Eingabesequenz
LBP_TOP	
blocks_x	Horizontale Aufteilung des Bildes in Blöcke für die LBP Merkmalsextraktion
blocky_y	Vertikale Aufteilung des Bildes in Blöcke für die LBP Merkmalsextraktion
blocks_t	Zeitliche Aufteilung des Bildes in Blöcke. Ist auf 1 gesetzt und wurde für diese Arbeit nicht weiter verwerdet.
radius_x	Radius in X-Richtung, um einen Pixel für die LBP-Code Berechnung
radius_y	Radius in Y-Richtung, um einen Pixel für die LBP-Code Berechnung
border_length	Äußerer Abstand in der X-Y Ebene für die LBP-TOP Berechnung
interval_t	Äußerer Abstand in der Zeitebene für die LBP-TOP Berechnung
time_length	Anzahl Blöck in der Zeitebene, ist für diese Arbeit auf 1 gesetzt.