

# **Visualisierung multivariater Daten nach Flury Riedwyl**

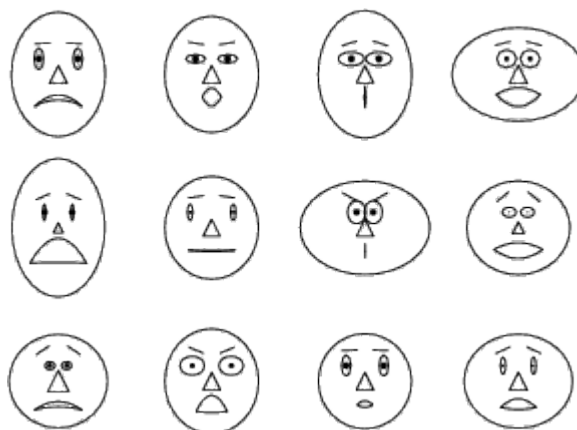
Autor : Stefan Beck  
Dozent : Dr. Axel Hoff  
Fach : Datenanalyse  
Datum : SS 2005

## Inhalt

|   |    |
|---|----|
| Einleitung.....   | 3  |
| Projektbeschreibung.....                                      | 3  |
| Was sind Flury Riedwyl Gesichter ?.....                       | 3  |
| Pro und Contra von Flury Riedwyl Gesichtern.....              | 4  |
| Wo werden Flury Riedwyl Gesichter eingesetzt ?.....           | 4  |
| Wieso Gesichter ?.....  | 5  |
| Was sind multivariate Daten ?.....                            | 7  |
| Die markanten Gesichtsmerkmale.....                           | 8  |
| Welche Merkmale werden benutzt ?.....                         | 9  |
| Umsetzung und verwendete Techniken.....                       | 10 |
| Funktionsweise des Programms.....                             | 10 |
| Tabelle der visualisierten Daten (Studenten – Abi Noten)..... | 12 |
| Die Normalisierung der Daten.....                             | 12 |
| Quellenangabe.....  | 13 |
| Der Quellcode (Face.h).....                                   | 14 |
| Fazit.....  | 16 |

## Einleitung

In der Praxis stellt sich oftmals das Problem, dass man Daten die mehr als drei Merkmale haben grafisch darstellen möchte. Da typischerweise alle Darstellungen die über die dritte Dimension hinausgehen für den Menschen schier unleserlich sind hat man nach neuen Gebieten gesucht um diese Problematik zu beheben. Eine sehr interessante Form davon sind die Flury-Riedwyl Gesichter. Hierbei nutzt man die Fähigkeit des Menschen Gesichter zu unterscheiden um Daten mit vielen Merkmalen zu visualisieren.



Chernoff Gesichter

## Projektbeschreibung

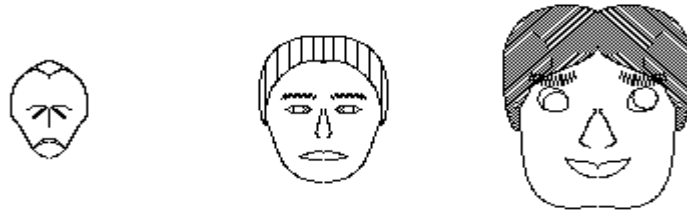
Das vorliegende Programm visualisiert acht Datensätze mit 16 verschiedenen Ausprägungen. Der Algorithmus wurde selbst entwickelt und durch die Flury-Riedwyl Gesichter inspiriert. Da ich typischerweise im Gebiet der Computergrafik aktiv bin war der Reiz ein solches Projekt anzugehen besonders groß.

## Was sind Flury Riedwyl Gesichter ?

Hinter Flury Riedwyl Gesichtern steckt eine stark reduzierte Form von Gesichtern. Diese eignen sich hervorragend zur Darstellung multidimensionale Daten. Flury Riedwyl Gesichter sind eine Weiterentwicklung der Chernoff Gesichter und weisen in der Regel bis zu 36 Merkmalen auf (18 pro Gesichtshälfte) Eine wichtige Tatsache ist, dass man die wichtigsten Merkmale durch die markantesten Gesichtsteilen darstellen sollte. Beispielsweise ist die Krümmung des Mundwinkels

wesentlich markanter als der Winkel der Nase.

Nachfolgendes Bild zeigt von links nach rechts das NULL Gesicht, das Durchschnittsgesicht und das EINS Gesicht. (Flury Riedwyl)



Flury Riedwyl (Null und Eins Gesicht)

### Pro und Contra von Flury Riedwyl Gesichtern

#### **Vorteile :**

- Leichte Erkennung von Trends (wenn genügend Gesichter vorliegen)
- Intuitive Interpretation der Diagramme.

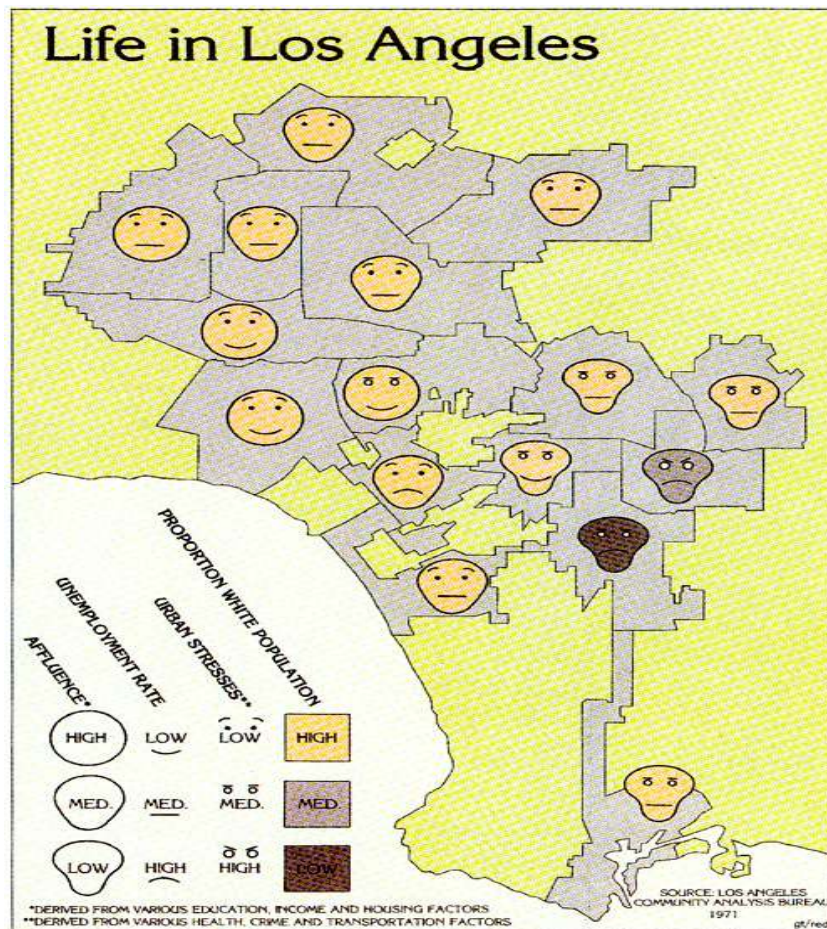
#### **Nachteile :**

- Durch die freie Zuordnung von Kennzahlen zu Merkmalen des Gesichtes kann der Gesamteindruck manipuliert werden.
- Unterschiedliche Gesichter können auf Personen verschieden wirken.
- Die absolute Darstellung der Ausprägungen geht verloren.

### Wo werden Flury Riedwyl Gesichter eingesetzt ?

Überall wo Datensätze mit mehreren Merkmalen Ausgewertet werden müssen.

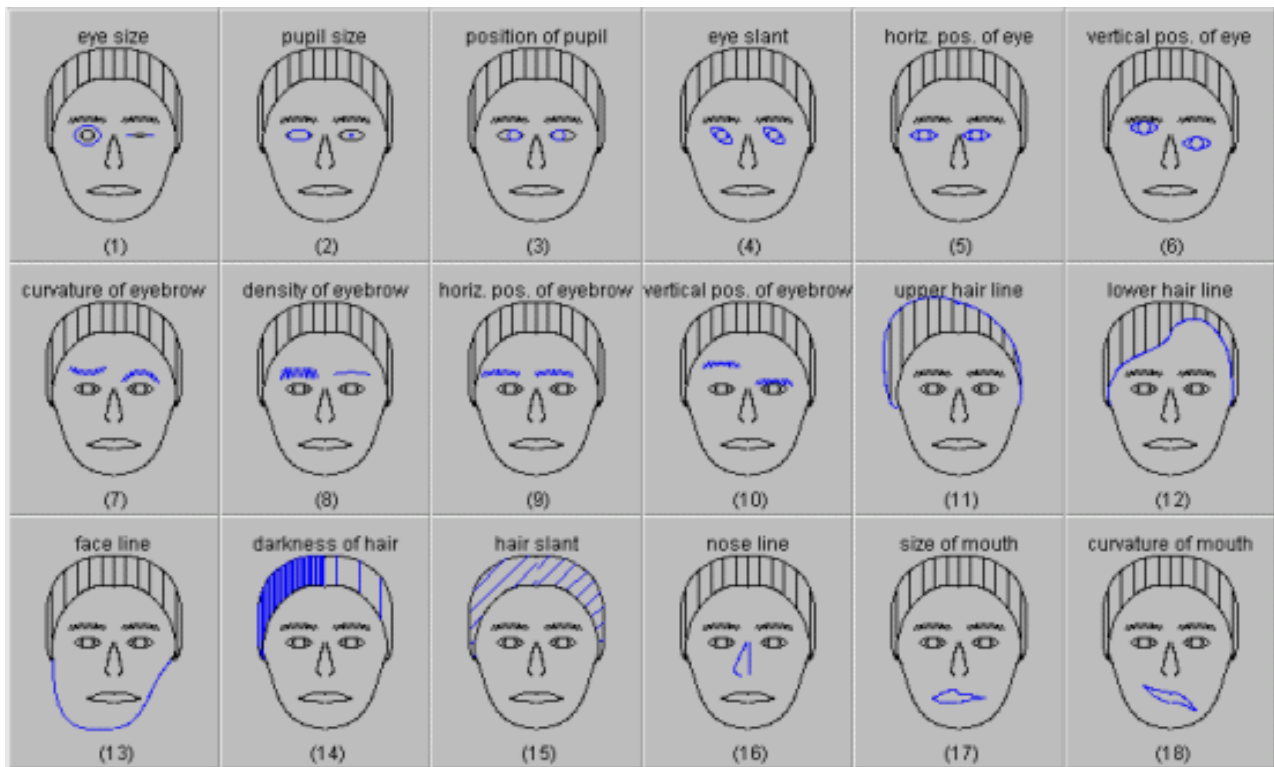
- Marktsegmentierung (anhand von Alter, Beruf, Bildung, ...)
- Privatkredit Vergabe (Kunden Bonitäts Prüfung)
- Produkt Image
- Ranking von z.B. Städten (Durchschnittsalter, Grünfläche, Kriminalitätsrate, ...)
- Ländervergleiche (Wirtschaftsdaten, Bevölkerung, ...)



## Wieso Gesichter ?

Gesichter bieten eine hervorragende Möglichkeit multivariate Daten darzustellen. Da ein Gesicht mehrere Merkmale aufweist (Grösse der Augen, Dicke der Augenbrauen, Krümmung des Mundes) und jedes dieser Merkmale in unterschiedlichsten Ausprägungen auftreten kann, gibt es nahezu unendlich viele verschiedene Gesichtstypen.

Stellen wir uns ein abstraktes Gesicht vor welches jeweils zehn unterschiedliche Ausprägungen für Augen, Ohren und Mund besitzt. Mit diesen „jeweils“ zehn unterschiedlichen Ausprägungen lassen sich insgesamt 1000 verschiedene Gesichter darstellen. ( $10 \text{ Augen} * 10 \text{ Ohren} * 10 \text{ Münder}$ ). Natürlich gibt es weit mehr als die drei Merkmale Augen, Ohren und Mund. Ebenso haben diese Merkmale, in der Praxis, auch weit mehr als zehn Ausprägungen. Mit einem abstrakten Gesicht welches 50 Merkmale besitzt die jeweils 100 Ausprägungsstufen haben können, könnte man  $50^{100}$  Gesichter darstellen. Die Zahl die hierbei heraus kommt hat 170 Stellen.



Obenstehende Grafik zeigt das Flury-Riedwyl Gesicht mit verschiedenen Merkmalen. Natürlich ist das Gesicht stark abstrahiert. In der Realität gibt es bei weitem mehr Merkmale. Eine gute Aufgabe sich darüber bewußt zu werden „wie“ viele Merkmale und Ausprägungen es in der Realität gibt, ist einfach mal zu schauen wie viele Menschen es gibt die ein wirklich „gleiches“ Gesicht haben. Zwillinge natürlich außen vor gelassen.

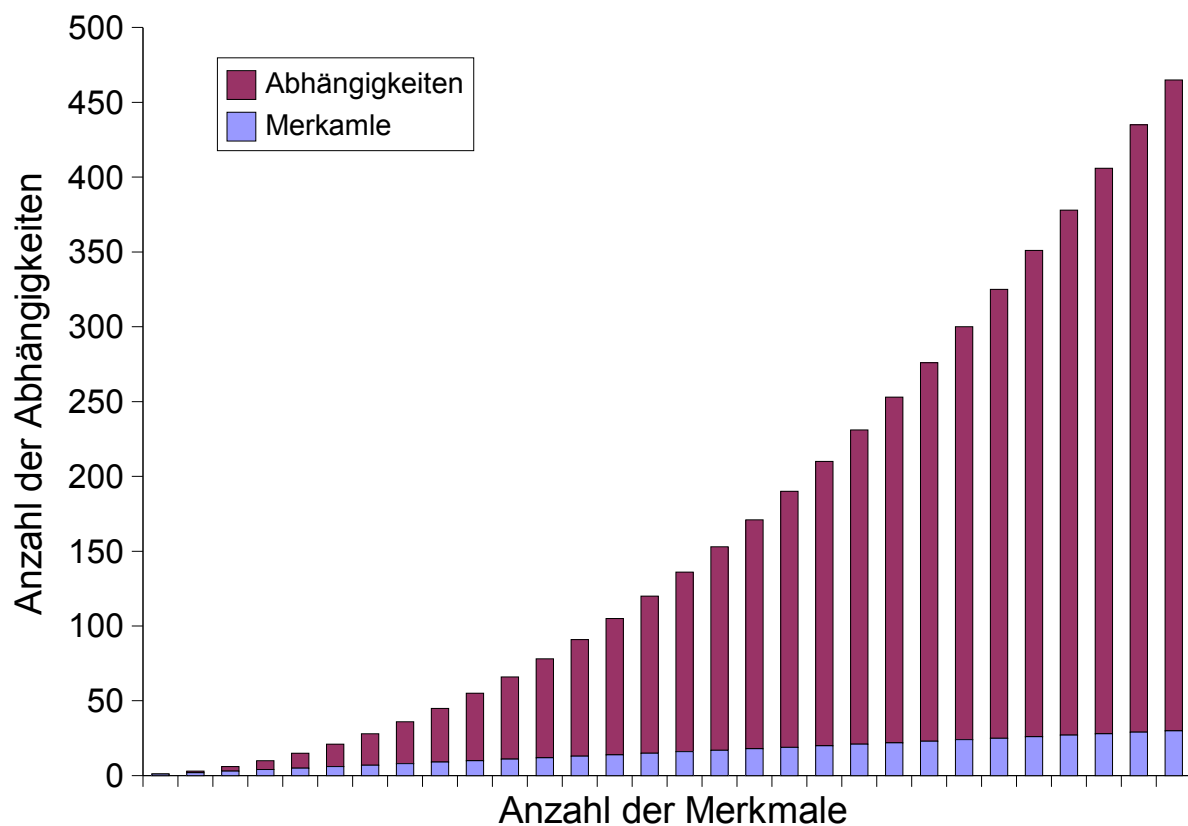
Ein anderer Grund der für die Benutzung von Gesichtern spricht ist auf die Fähigkeit zurück zu führen, das der Mensch ein Experte im Erkennen von Gesichtern ist. Diese Tatsache hat weitestgehend damit zu tun das wir unsere Artgenossen wieder erkennen. Das Erkennen von Gesichtern können übrigens auch andere Spezien. Versuche haben ergeben das Schafe durchaus in der Lage sind unter tausenden von anderen Schafen Ihre Artgenossen anhand der Gesichter wieder zu erkennen. Für einen Menschen eine schier unlösbare Aufgabe. Dafür ist der Mensch auf die Erkennung „seiner“ Artgenossen spezialisiert.

## Was sind multivariate Daten ?

Im Gegensatz zur **bivariaten** ( $bi=2$ ) Daten, beschäftigt sich die **multivariate** Datenanalyse mit Daten die mehrere (multi) Merkmale aufweisen. Bei einer Analyse der Daten müssen je nach Anzahl der Merkmale sehr viele Zusammenhänge untersucht werden.

**n Merkmale verlangen nach  $(n*(n-1)) / 2$  Tests.**

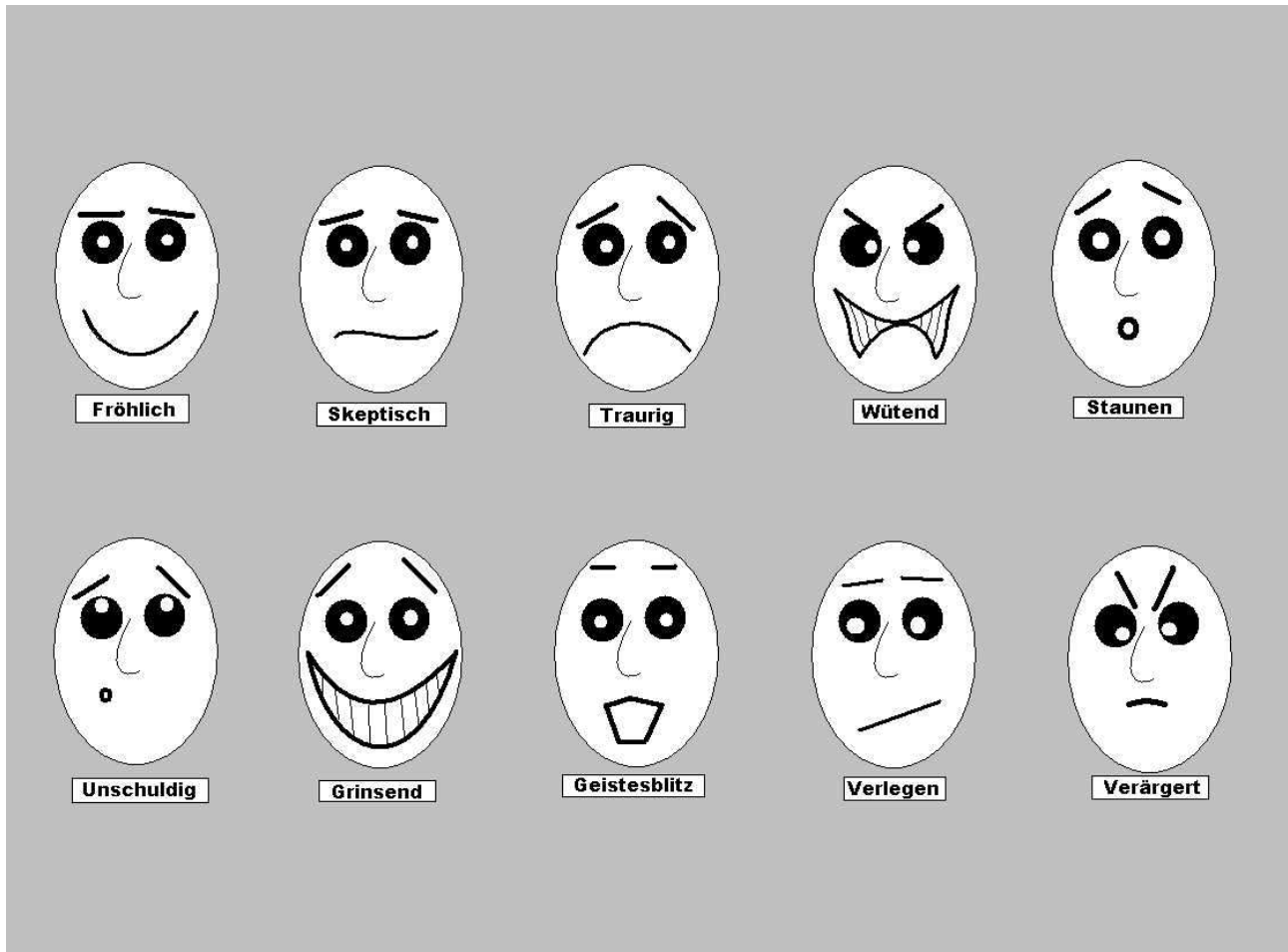
### Abhängigkeiten bei multivariaten Daten



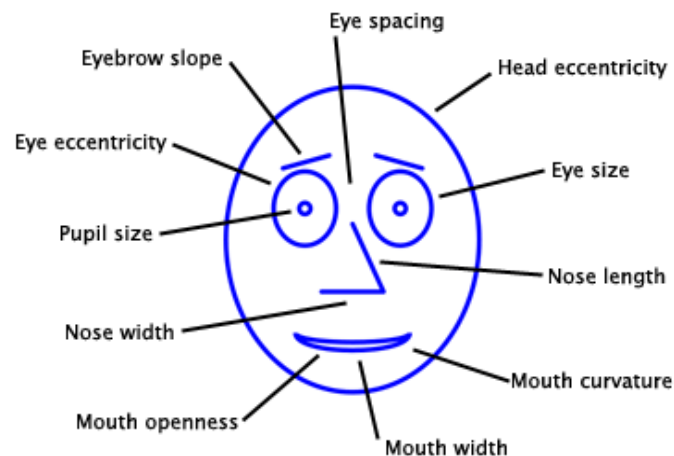
Das Diagramm zeigt den Anstieg der Abhängigkeiten. Auffallend ist dass bei einem linearen Anstieg der Merkmale die Abhängigkeiten fast exponentiell ansteigen.

## Die markanten Gesichtsmerkmale

Eine frühere Studie über Merkmale von Gesichtern, die verschiedene Emotionen Ausdrücken, hat mit zur Inspiration gedient Gesichts Merkmale zu erruieren.



Wie unschwer zu erkennen ist spielt der Mund sowie die Anordnung der Augenbrauen eine wesentliche Rolle beim Ausdruck von Emotionen. In diesem frühen Beispiel sind viele Merkmale noch unbeachtet. Die Anzahl der Merkmale des Gesichtes, legt den Grad der Dimension fest die ein Datensatz besitzen darf, bzw. den Grad der Visualisiert werden kann. Das heisst um so höher die Anzahl der Merkmale sind, um so höher-dimensionale Sachverhalte können visualisiert werden. Nächstes Bild könnte z.B. Elf Dimensionen beschreiben.



### Welche Merkmale werden benutzt?

Nun sollen die Merkmale die ein von mir konstruiertes Gesicht aufweisen, aufgezählt werden. Die Eingangsdaten die typischerweise unnormiert vorliegen werden auf einen Wert zwischen 0 und 1 normiert. Also so zu sagen vorgefiltert.

- Kopfform (2) : Höhe, Breite
- Augenbrauen (4) : Höhen Position Y, Abstand, Winkel, Dicke,
- Augen (3) : Höhen Position Y, Höhe, Abstand,
- Nase (2) : Höhen Position Y, Breite,
- Mund (4) : Höhen Position Y, Höhe, Breite, Dicke

Das sind 15 verschiedene Merkmale. Somit lässt sich mit einem Gesicht praktisch ein 15 dimensionales System darstellen. Allerdings ist zu beachten daß einzelne Merkmale sich in einem kleineren Bereich bewegen als die Daten die man visualisieren möchte. Somit kann es schnell zu Quantisierungsfehlern kommen.

Theoretisch liesse sich die Merkmalsanzahl auf 32 anheben indem man jedes doppelt vorkommende Gesichtsmerkmal (Augen, Augenbrauen ...) einzeln parametrieren könnte.

## Umsetzung und verwendete Techniken

Das Programm zur Visualisierung der Gesichter erfolgte unter Verwendung von

- C++ (Visual Studio .Net 2003)
- API OpenGL (Open Graphics Library) Version 1.2
- Open Office (Sun Microsystems)
- Photoshop (Adobe)
- Paint (Microsoft)
- Windows XP Professional (Microsoft)

Die Primitiven die zur Darstellung des Gesichtes verwendet wurden sind Bestandteil einer von mir entwickelten 3D Engine.

## Funktionsweise des Programms

Das Beispielprogramm zeigt in der obersten Reihe ein Gesicht welches kontinuierlich zwischen Null und Eins Gesicht schwankt. Nebenan ist das Null und das Eins Gesicht zu sehen.

In der Zweiten Zeile sieht man acht unterschiedlich parametrisierte Gesichter. Die Daten kommen aus einer Untersuchung, bei der man herausfinden wollte inwiefern die Punkte, einzelner Fächer, beim Abitur Einfluss auf die zukünftige Studienwahl haben. Um es an einem kleinen Beispiel zu verdeutlichen: Studenten von Naturwissenschaftlichen Fächern hatten eher gute Leistungen in Mathematik und Physik gezeigt, als Studenten der Richtung Philosophie.

Darunter sind zum Vergleich die original Flury Riedwyl Gesichter als Bild zu sehen. Mit der Taste "i" kann man sich die Bedeutung der einzelnen Parameter auflisten. Ausserdem gibt es die Möglichkeit sich die Grafikkarte und deren "Flags" anzusehen als einen kleinen "Credits" Bildschirm.

Nach drücken der Taste "Enter" kommt man in einen viereckigen Würfel wo eine zufällige 3D Kurve erscheint. Diese soll nur als Demonstration dienen wie unanschaulich eine Darstellung im 3D Raum mit nur drei Parametern ist.

Dieser Screenshot gibt einen ersten Einblick in das Programm.

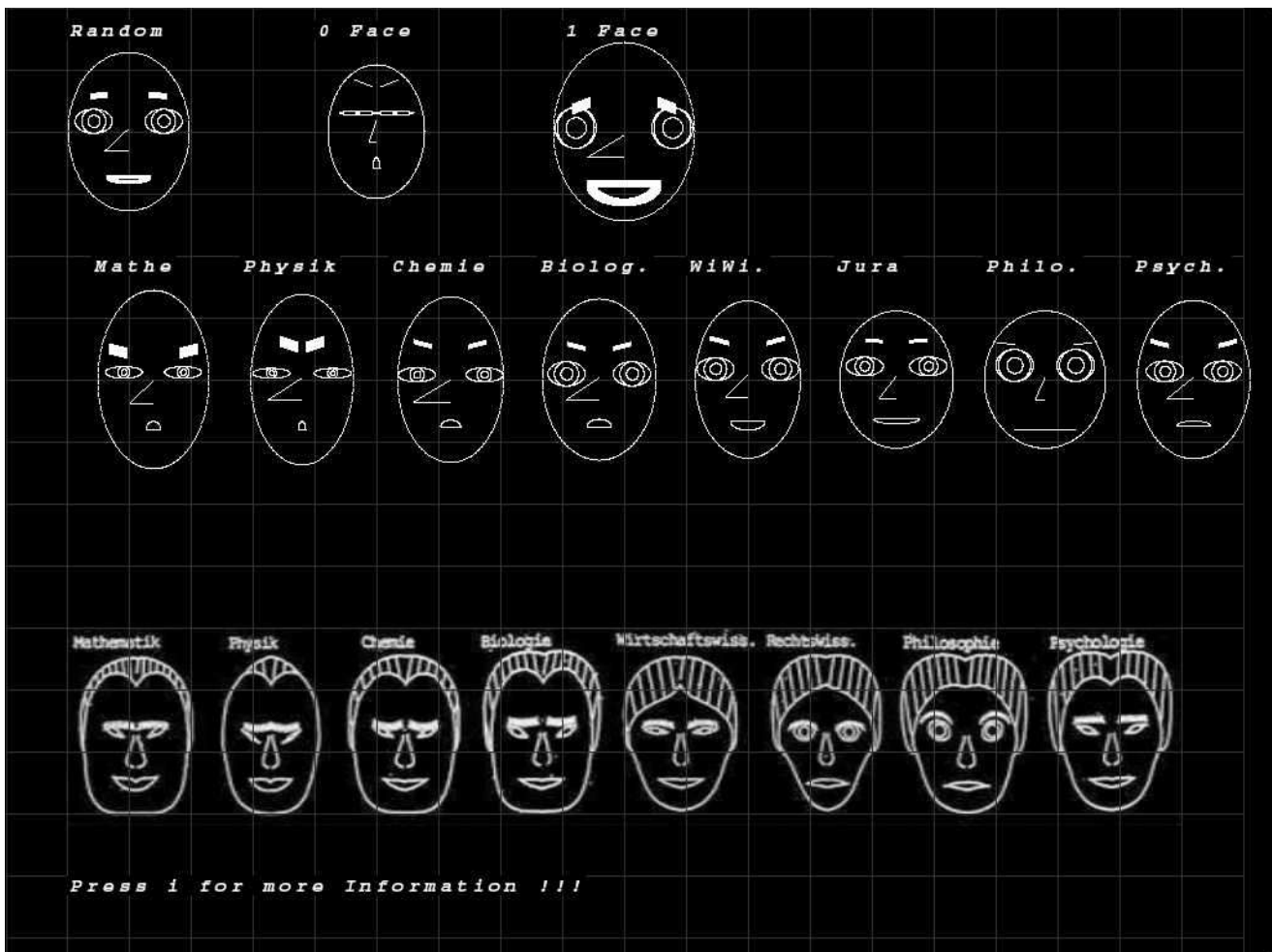


Tabelle der visualisierten Daten (Studenten – Abi Noten)

| (Abiturleistung)             | Mathe | Phys. | Chem. | Bio  | Wiwi | Jura | Philo. | Psych |
|------------------------------|-------|-------|-------|------|------|------|--------|-------|
| <b>Note Deutsch</b>          | 9.5   | 8.2   | 8.7   | 9.9  | 8.6  | 9.8  | 11.0   | 9.7   |
| <b>Ant. Deutsch</b>          | 0.15  | 0.09  | 0.14  | 0.17 | 0.14 | 0.35 | 0.38   | 0.15  |
| <b>Note Mathe</b>            | 14.0  | 12.8  | 12.1  | 11.4 | 10.8 | 7.8  | 7.8    | 11.0  |
| <b>Ant. Mathe</b>            | 0.85  | 0.86  | 0.38  | 0.35 | 0.32 | 0.17 | 0.04   | 0.35  |
| <b>Note Natur Wissensch.</b> | 12.4  | 12.8  | 13.5  | 13.1 | 10.9 | 9.7  | 9.3    | 12.0  |
| <b>Ant. Natur Wissensch.</b> | 0.5   | 0.82  | 0.9   | 0.78 | 0.45 | 0.3  | 0.08   | 0.6   |
| <b>Note Geschichte</b>       | 9.8   | 10.0  | 10.7  | 10.5 | 10.4 | 10.3 | 10.0   | 9.6   |
| <b>Ant. Geschichte</b>       | 0.1   | 0.09  | 0.14  | 0.09 | 0.6  | 0.48 | 0.33   | 0.2   |
| <b>Note neue Sprachen</b>    | 8.7   | 8.4   | 9.2   | 11.0 | 10.7 | 10.0 | 12.4   | 10.4  |
| <b>Ant. neue Sprachen</b>    | 0.1   | 0.0   | 0.19  | 0.26 | 0.41 | 0.57 | 0.8    | 0.4   |
| <b>Note alte Sprachen</b>    | 12.4  | 7.3   | 10.5  | 9.5  | 10.7 | 9.3  | 13.1   | 12.1  |
| <b>Ant. alte Sprachen</b>    | 0.05  | 0.0   | 0.0   | 0.04 | 0.05 | 0.04 | 0.21   | 0.1   |
| <b>Note Musik</b>            | 12.0  | 10.4  | 10.5  | 11.0 | 9.7  | 10.1 | 10.8   | 10.3  |
| <b>Note Sport</b>            | 12.9  | 11.6  | 12.4  | 11.5 | 10.1 | 10.7 | 11.4   | 11.2  |
| <b>Note Technik</b>          | 12.9  | 13.2  | 11.8  | 12.1 | 9.4  | 8.4  | 13.0   | 11.3  |

Die Normalisierung der Daten

Da für interne Zwecke eine Darstellung zwischen 0-1 besser geeignet ist, als beliebige Werte müssen Die Daten erst normalisiert werden. (Alle Eingänge der Gesichts-Funktion erwarten Werte zwischen 0-1). Die normalisierung gestaltet sich denkbar einfach. Wenn man Daten hat die so skaliert werden sollen dass der kleinste Wert Null entspricht und der grösste Wert 1 so sucht man den Minimalen und den Maximalen Wert aus den Daten. Als nächstes stellt man diese Formel auf:

$$(-\text{Min} + x) / (\text{Max} - \text{Min}) = \text{skaliertes Wert}$$

x entspricht hierbei einem beliebigen Wert aus dem unskalierten Datenbereich.

## Quellenangabe

Institut für mathematische Statistik und Versicherungslehre

[http://www.imsv.unibe.ch/html/faceplot/Faceplot\\_README.html](http://www.imsv.unibe.ch/html/faceplot/Faceplot_README.html)

SPRINT-Ansatz zur prozessbegleitenden Interpretation und Visualisierung von Messdaten in der Software Entwicklung ( Silke Bötzel Mai 2000 )

<http://www.wagse.informatik.uni-kl.de/pubs/repository/boetzel00a/DA-Boetzel-2000.pdf>

Lehrstuhl für Statistik und Ökonomie Wirtschaftswissenschaftliche Universität Potsdam (SS 2005 Knut Bartels)

[http://stat.wiso.uni-potsdam.de/documents/mvstat/mvstat05\\_folien\\_01.pdf](http://stat.wiso.uni-potsdam.de/documents/mvstat/mvstat05_folien_01.pdf)

Wolfram Research

<http://mathworld.wolfram.com/ChernoffFace.html>

Script zur Datenanalyse von Dr. Axel Hoff (Steinbeis Transferzentrum) an der Naturwissenschaftlich Technischen Akademie NTA Prof. Dr Grübler gGmbH

<http://www.stz-isd.de/datenanalyse/>

Faces 2.0

<http://www.bradandkathy.com/software/faces.html>

Chernoff Faces

[http://kspark.kaist.ac.kr/Human%20Engineering\\_files/Chernoff/Chernoff%20Faces.htm](http://kspark.kaist.ac.kr/Human%20Engineering_files/Chernoff/Chernoff%20Faces.htm)

## Der Quellcode (Face.h)

```
/*
//
//     VERSION : 1.0
//     NAME   :   Face.h
//     DESC   :   several functions for face-primitives (2D)
//     AUTOR  :   Stefan Beck | tolschock@gmx.de
//     DATE   :   19.04.2005
//     UPDATE :   21.07.2005
//     BUGS   :   none
//
*/
void DrawEyebrown(float x, float y, float u, float v, float thick)
{
    // u = abstand der Augenbrauen
    // v = neigung der augenbrauen

    DrawLine(x-u-50, y, x-u, y+v, thick);
    DrawLine(x+u, y+v, x+u+50, y, thick);
}
/*
void DrawEye(float x, float y, float u, float v, float thick)
{
    // v = Höhe des Auges
    // u = Abstand der Augen

    DrawEllipse(-u, y-50, 50, v, thick);
    DrawEllipse(-u, y-50, v, v, thick);
    DrawEllipse(-u+((v)/v), y-50, v/2, v/2, thick);

    DrawEllipse(+u, y-50, 50, v, thick);
    DrawEllipse(+u, y-50, v, v, thick);
    DrawEllipse(+u+((v)/v), y-50, v/2, v/2, thick);
}
/*
void DrawNose(float x, float y, float u, float thick)
{
    DrawLine(x, y-30, x-u, y+30, thick);
    DrawLine(x-u, y+30, x, y+30, thick);
}
*/
```

```

void DrawMouth(float x, float y, /*float u, */float v, float w, float thick)
{
    // u = x-Position des Mundes
    // v = breite des mundes
    // w= höhe des mundes

    glPushMatrix();
        glTranslatef(x, y, 0);
        glRotatef(90, 0, 0, 1);
        //glColor3f(204.0f/255.0f, 95.0f/255.0f, 100.0f/255.0f);           // Mund
innenraum
        //DrawFilledArc(0, 0, v, w, 2, thick);
        DrawArc(0, 0, v, w, 2, thick);
        //glColor3f(1.0f, 0.0f, 0.0f);                                   // Lippenfarbe
        DrawArc(0, 0, v, w, 2, thick);
    glPopMatrix();
}
/*****
/*****
void DrawFace( float x,
               float y,
               float thick,
               float x_head_width,
               float x_head_height,
               float x_eye_y_pos,
               float x_eye_spacing,
               float x_eye_height,
               float x_eyebrow_y_pos,
               float x_eyebrow_spacing,
               float x_eyebrow_angle,
               float x_eyebrow_thick,
               float x_nose_y_pos,
               float x_nose_length,
               float x_mouth_y_pos,
               float x_mouth_height,
               float x_mouth_width,
               float x_mouth_thick
               )
{
// Kopf
DrawEllipse(  x,
              y,
              130 + x_head_width *60, // width of head -20 / 20
              180 + x_head_height *60, // height of head
              1); // border line of head

```

```

DrawEye(    0,
            y      + x_eye_y_pos      *40, // y-position der augen
            50     + x_eye_spacing    *80, // Abstand der Augen
            5      + x_eye_height     *50, // Höhe der Augen
            1); // Linienstärke der Augen

DrawEyebrow(x,
            y-140 + x_eyebrow_y_pos    *80, // y-position der augenbrauen
            10   + x_eyebrow_spacing  *80, // Abstand der Augenbrauen
            20   - x_eyebrow_angle    *40, // Neigung der Augenbrauen
            1     + x_eyebrow_thick    * 8); // Dicke der Augenbrauen

DrawNose(   x,
            y      + x_nose_y_pos *40, // Höhe der Nase relativ zum Kopf
            20     + x_nose_length  *80, // Länge der Nase
            1); // Nasen Dicke

DrawMouth(  x,
            y+100 + x_mouth_y_pos    *40, // y-position
            -30   + x_mouth_height    *80, // höhe des mundes
            10    + x_mouth_width     *80, // breite des mundes
            0     + x_mouth_thick     * 6); // Linienstärke des Mundes
            20);
}
/*****/

```

## Fazit

Interessant an der Analyse multivariater Daten mit Flury Riedwyl Gesichtern ist die Tatsache das man keine komplexe Mathematik wie z.B. bei der Hauptkomponentenanalyse benötigt. Multivariate Daten zu finden gestaltet sich ungleich schwerer als uni oder bivariate Daten. Des weiteren konnte ich meine Engine zur 2D Darstellung ausbauen und um etliche Primitive erweitern. Mögliche Überlegungen wären, das Programm dahingehend auszubauen, das die Daten die derzeit "hardgecodet" im Programm vorliegen über eine Schnittstelle z.B. XML einzulesen.