Facial Expression Tracking based on statistical methods and neural networks -
Automatische Erkennung von Gesichtsausdrücken auf der Basis statistischer Methoden und neuronaler Netze

Arne Arnold
Agenda

• Introduction
  – What’s behind the topic?
  – What’s already out there?
  – What’s isn’t there, what’s new?

• What my work is based on:
  – The Active Shape and Active Appearance Model
  – The Fast Artificial Neural Network library
  – The Visual something Libraries (VxL)

• Implementation
• Experiments & Results
• Conclusion
What’s behind the topic?

• We see facial expressions as a subset of affective computing / sensing.
• As humans are able to interact using only facial expressions, we want to have computer system behaving the same.
• A system should be able to automatically track facial expressions and classify them into a given number of emotions.

[1]
Indeed, there are similar approaches, but…

- They are using huge ANNs as feature-detector.
- They require additional knowledge for constraints.
- They are using connected sensors as a source of information.
- They require special mark-ups in the video frames.
What’s isn’t there, what’s new?

Our approach therefore is:

- Automatically recognize the facial expressions subject to only visual information.
- Using a flexible but restricted statistical Active Shape / Active Appearance Model to track the facial features.
- Using an artificial neural network to classify the resulting Eigenvalues.
Agenda

• Introduction
  – What’s behind the topic?
  – What’s already out there?
  – What’s isn’t there, what’s new?

• What my work is based on:
  – The Active Shape and Active Appearance Model
  – The Fast Artificial Neural Network library
  – The Visual something Libraries (VxL)

• Implementation
• Experiments & Results
• Conclusion
The main advantage of the Active Shape Model (ASM) and Active Appearance Model (AAM) described by Cootes et al. [6, 7] is, that they can only deform to fit data in ways consistent with a training set.

In order to model such a set of shapes, each shape must be prior labelled and aligned in scale, rotation and translation.

Any unseen image can then be spanned by taking the meanshape and adding a linear combination of Eigenvectors.

\[ x_i = (x_{i1}, y_{i1}, x_{i2}, y_{i2}, \ldots x_{in}, y_{in})^T \]

\[ \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i, \quad x = \bar{x} + \Phi b \]
Edge-detection using ASM

- Finding a new shape means to minimize the difference between found points in the image and a current instance of the model.
- This is an iterative approach, where a better position for each model point is searched and afterwards the whole shape is trimmed to the allowable shape domain.
#include "fann.h"

int main()
{

    struct fann *ann = fann_create(1, 0.7, 3, 26, 13, 3);

    fann_train_on_file(ann, "freq.data", 200, 10, 0.0001);

    fann_save(ann, "lang_classify.net");

    fann_destroy(ann);

    return 0;
}

- FANN is an Open-Source ANN Library, which can be used from C, C++, PHP, Python, Delphi and Mathematica.

- It is created and actively maintained by Steffen Nissen (Denmark).

- As a function approximator like an ANN can be viewed as a black box, this is pretty much what FANN is (or can be).
Visual *something* Libraries (VxL)

- VxL is a collection of C++ libraries designed for computer vision research and implementation. It covers:
  - Numerical containers and algorithms
  - Loading, saving and manipulating images
  - Multi-platform user-interface using OpenGL
  - ...

```cpp
// compute eigenvalues
vtrX = vnl_vector<double>(n, 0.0);
mtxL = vnl_matrix<double>(m, n, 0.0);
vnl_symmetric_eigensystem_compute( mtxA, mtxL, vtrX );

// sort eigenvectors corresponding to decreasing eigenvalues
vtrX.flip();
mtxL.fliplr();
```
Agenda

• Introduction
  – What’s behind the topic?
  – What’s already out there?
  – What’s isn’t there, what’s new?

• What my work is based on:
  – The Active Shape and Active Appearance Model
  – The Fast Artificial Neural Network library
  – The Visual something Libraries (VxL)

• Implementation
• Experiments & Results
• Conclusion
Facial Expression Tracking Application - architecture (static perspective)
FETA – dataflow (dynamic perspective)
Facial Expression Tracking Application (FETA) written by Arne Arnold (as part of the Masterthesis) at the National ICT Australia / Australian National University, Canberra, 2005
The imagination driving Australia’s ICT future.

FETA – in operation

result for ../test/surpr_095.jpg: 0 0 1 (0.26 seconds)
result for ../test/surpr_096.jpg: 0 0 1 (0.27 seconds)
result for ../test/surpr_098.jpg: 0 0.992747 0.9569 (0.26 seconds)
result for ../test/surpr_099.jpg: 0 1 0.087709 (0.26 seconds)
result for ../test/happy_100.jpg: 0 1 0 (0.27 seconds)
result for ../test/happy_101.jpg: 0 1 0 (0.25 seconds)
TOTAL number of Pictures proceeded: 203
AVERAGE time per Image: 0.258128 seconds
TOTAL elapsed time: 52.4 seconds
Experiments & Results

- **example with intrinsic data:** 392 (120/165/46/61)* Training img., 354 (87/178/64/124)* Test images and four different facial expressions a recognition rate of 72% could be reached.

- **example with extrinsic data:** 60 (30/30)** Training images, 20 (10/10)** Test images and two different facial expressions the model was able to recognize 80% of all Testimages correct.

<table>
<thead>
<tr>
<th>Tabelle 7.2: Confusion matrix for FGnet DB</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>neutr.</td>
<td>happy</td>
<td>surpr.</td>
<td>disgs.</td>
</tr>
<tr>
<td>neutral</td>
<td>46</td>
<td>1</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>happy</td>
<td>6</td>
<td>67</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>surprise</td>
<td>3</td>
<td>-</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>disgust</td>
<td>19</td>
<td>4</td>
<td>1</td>
<td>69</td>
</tr>
<tr>
<td>undefined</td>
<td>13</td>
<td>7</td>
<td>11</td>
<td>33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tabelle 7.4: Confusion matrix for IMM FaceDB</th>
<th>neutr.</th>
<th>happy</th>
</tr>
</thead>
<tbody>
<tr>
<td>neutral</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>happy</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tabelle 7.5: Recognition rates for all tested imagesets</th>
<th>FGnet DB</th>
<th>IMM FaceDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>neutral</td>
<td>53%</td>
<td>90%</td>
</tr>
<tr>
<td>happy</td>
<td>38%</td>
<td>70%</td>
</tr>
<tr>
<td>surpr.</td>
<td>75%</td>
<td>-</td>
</tr>
<tr>
<td>disgs.</td>
<td>56%</td>
<td>-</td>
</tr>
<tr>
<td>total</td>
<td>72%</td>
<td>80%</td>
</tr>
</tbody>
</table>

* (neutr./happy/surpr./disgs.)
** (neutr./happy)
Conclusion

- It was shown, that the combination of ASM/AAM and AAN can achieve similar recognition rates, as other approaches, using only ANNs.
- The advantage is, that it is much more robust and particularly suitable for working with occluded image data.
- Furthermore, there is no need of special knowledge, eg. on biological constraints, because all constraints are learned automatic within the training phase.
- There are still many options, where the model can be optimized, so the shown results are only the bottom line of what seems to be possible.
Agenda

• Introduction
  – What’s behind the topic?
  – What’s already out there?
  – What’s isn’t there, what’s new?

• What I’d to deal with
  – The Active Shape and Active Appearance Model
  – The Fast Artificial Neural Network library
  – The Visual something Libraries (VxL)

• Implementation
• Experiments & Results
• Conclusion
Bibliography