Face Detection and **Alignment**

Prof. Xin Yang

HUST
Applications — Virtual Makeup
Applications — Virtual Makeup
Applications — Video Editing
Active Shape Models

• Suppose we have a statistical shape model
  – Trained from sets of examples
• How do we use it to interpret new images?
• Use an “Active Shape Model”
• Iterative method of matching model to image
Building Models

• Require labelled training images
  – landmarks represent correspondences
Building Shape Models

- **Given aligned shapes, \{ \mathbf{x}_i \}**
  For instance, in a 2-D image we can represent the \( n \) landmark points, \{ \((x_i, y_i)\)\}, for a single example as the \( 2n \) element vector, \( \mathbf{x} \), where

\[
\mathbf{x} = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T
\]  

(4.1)

- **Apply PCA**
  \[
  \mathbf{x} = \mathbf{\bar{x}} + \mathbf{Pb}
  \]

- **\( \mathbf{P} \) – First \( t \) eigenvectors of covar. matrix**
- **\( \mathbf{b} \) – Shape model parameters**
From $k$ original variables: $x_1, x_2, \ldots, x_k$:

Produce $k$ new variables: $y_1, y_2, \ldots, y_k$:

\[
y_1 = a_{11} x_1 + a_{12} x_2 + \ldots + a_{1k} x_k
\]
\[
y_2 = a_{21} x_1 + a_{22} x_2 + \ldots + a_{2k} x_k
\]
\[
\vdots
\]
\[
y_k = a_{k1} x_1 + a_{k2} x_2 + \ldots + a_{kk} x_k
\]

such that:

$y_k$'s are uncorrelated (orthogonal)
$y_1$ explains as much as possible of original variance in data set
$y_2$ explains as much as possible of remaining variance
etc.
1st Principal Component, $y_1$

2nd Principal Component, $y_2$
PCA Scores
PCA Eigenvalues

\[ \lambda_1 \quad \lambda_2 \]
From $k$ original variables: $x_1, x_2, \ldots, x_k$:

Produce $k$ new variables: $y_1, y_2, \ldots, y_k$:

$y_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1k}x_k$

$y_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2k}x_k$

... 

$y_k = a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kk}x_k$

such that:

$y_k$'s are uncorrelated (orthogonal)

$y_1$ explains as much as possible of original variance in data set

$y_2$ explains as much as possible of remaining variance

etc.
Principal Components Analysis on:

- **Covariance Matrix:**
  - Variables must be in same units
  - Emphasizes variables with most variance
PCA: General

\{a_{11}, a_{12}, \ldots, a_{1k}\} \text{ is 1st Eigenvector of correlation/covariance matrix, and coefficients of first principal component}

\{a_{21}, a_{22}, \ldots, a_{2k}\} \text{ is 2nd Eigenvector of correlation/covariance matrix, and coefficients of 2nd principal component}

\ldots

\{a_{k1}, a_{k2}, \ldots, a_{kk}\} \text{ is kth Eigenvector of correlation/covariance matrix, and coefficients of kth principal component}
Building Shape Models

- Given aligned shapes, \{ \mathbf{x}_i \}
  
  For instance, in a 2-D image we can represent the \( n \) landmark points, \{ (x_i, y_i) \}, for a single example as the \( 2n \) element vector, \( \mathbf{x} \), where

\[
\mathbf{x} = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T
\]

- Apply PCA

\[
\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Pb}
\]

- \( \mathbf{P} \) – First \( t \) eigenvectors of covar. matrix
- \( \mathbf{b} \) – Shape model parameters
Hand Shape Model

Varying $b_1$  Varying $b_2$  Varying $b_3$
BUILDING THE MODELS

All Faces – Modes of Variation

First mode
Second mode
Third mode
BUILDING THE MODELS

Neutral - Appearance Variation

• Changing the first three modes of variation simultaneously
Active Shape Models

- Match shape model to new image
- Require:
  - Statistical shape model
  - Model of image structure at each point
Placing model in image

• The model points are defined in a model coordinate frame

• Must apply global transformation, $T$, to place in image

$$ x = \overline{x} + Pb $$

$$ T(x; X_c, Y_c, s, \theta) $$

Model Frame

$$ X = T(\overline{x} + Pb) $$

Image
ASM Search Overview

- Local optimisation
- Initialise near target
  - Search along profiles for best match, $X'$
  - Update parameters to match to $X'$.
Local Structure Models

• Need to search for local match for each point

• Model
  – Strongest edge
  – Statistical model of profile
Computing Normal to Boundary

Tangent \((t_x, t_y)\)

Normal \((n_x, n_y) = (-t_y, t_x)\)

\((X_{i-1}, Y_{i-1})\) \hspace{2cm} \((X_{i+1}, Y_{i+1})\)

\begin{align*}
(t_x, t_y) &\approx \frac{(d_x, d_y)}{\sqrt{d_x^2 + d_y^2}} \\
d_x &= X_{i+1} - X_{i-1} \\
d_y &= Y_{i+1} - Y_{i-1}
\end{align*}

(Unit vector)
Sampling along profiles

Model boundary

Model point

$$(X,Y)$$

Profile normal to boundary

Interpolate at these points

$$(X,Y) + i(s_n n_x, s_n n_y)$$

$$i = \ldots -2, -1, 0, 1, 2, \ldots$$

Take steps of length $$s_n$$ along $$(n_x, n_y)$$

$$s_n$$

$$s_n n_x$$

$$s_n n_y$$
Noise reduction

• In noisy images, average orthogonal to profile
  – Improves signal-to-noise along profile

Use \( g_i = 0.25 g_{i1} + 0.5 g_{i2} + 0.25 g_{i3} \)

Sampled profile is

\( g = (\ldots, g_{-2}, g_{-1}, g_0, g_1, g_2, \ldots) \)
Searching for strong edges

$\frac{dg(x)}{dx} = 0.5(g(x + 1) - g(x - 1))$

Select point along profile at strongest edge
Profile Models

- Sometimes true point not on strongest edge

- Model local structure to help locate the point
Statistical Profile Models

- Estimate p.d.f. for sample on profile
- Normalise to allow for global lighting variations
- From training set learn $p(g)$
Profile Models

• For each point in model
  – For each training image
    • Sample values along profile
    • Normalise
  – Build statistical model
    • eg Gaussian PDF using eigen-model approach

By applying PCA to the normalised data we obtain a linear model:

\[ g = \bar{g} + P_g b_g \]  \hspace{1cm} (5.3)

where \( \bar{g} \) is the mean normalised grey-level vector, \( P_g \) is a set of orthogonal modes of variation and \( b_g \) is a set of grey-level parameters.
Searching Along Profiles

- During search we look along a normal for the best match for each profile

\[ g(x) \]

Form vector from samples about \( x \)

\[ p(g(x)) \]
Search algorithm

- Search along profile
- Update global transformation, $T$, and parameters, $b$, to minimise

$$| X - T (\bar{x} + Pb) |^2$$
Updating parameters

- Find pose and model parameters to minimise

\[ f (b, X_c, Y_c, s, \theta) = |X - T (\bar{x} + Pb; X_c, Y_c, s, \theta) |^2 \]

- Either
  - Put into general optimiser
  - Use two stage iterative approach
Updating Parameters

\[
    f (b, X_c, Y_c, s, \theta) = \left| X - T (\bar{x} + Pb ; X_c, Y_c, s, \theta) \right|^2
\]

Repeat until convergence:

Fix \( b \) and find \( (X_c, Y_c, s, \theta) \) which minimise \( \left| X - T (\bar{x} + Pb) \right|^2 \)

Analytic solution exists (see notes)

Fix \( (X_c, Y_c, s, \theta) \) and find \( b \) which minimises \( \left| X - T (\bar{x} + Pb) \right|^2 \)

\[
    b = P^T (T^{-1}(X) - \bar{x})
\]
Update step

- Hard constraints

\[
\text{Minimise} \quad |X - T(x + Pb)|^2 \quad \text{subject to} \quad p(b) < p_t
\]

\[\text{e.g.}\ |b_i| \leq 3\sqrt{\lambda_i}\]
Multi-Resolution Search

• Train models at each level of pyramid
  – Gaussian pyramid with step size 2
  – Use same points but different local models

• Start search at coarse resolution
  – Refine at finer resolution
Gaussian Pyramids

- To generate image at level L
  - Smooth image at level L-1 with gaussian filter (eg (1 5 8 5 1)/20)
  - Sub-sample every other pixel

Each level half the size of the one below
Multi-Resolution Search

• Start at coarse resolution

• For each resolution
  – Search along profiles for best matches
  – Update parameters to fit matches
  – (Apply constraints to parameters)
  – Until converge at this resolution
ASM Fitting Process
ASM Fitting Process
Face Tracking Framework: Active Shape Model

Start Shape on the ROI

Feature Extraction & Search

Shape Realignment on the ROI

Output the Shape

Landmarks of Active Shape Model
Algorithm

1. Face Detection
2. Start Shape on the ROI
3. Feature Extraction & Search
4. Shape Realignment on the ROI
5. Output the Shape
Algorithm

1. Face Detection
2. Start Shape on the ROI
3. Feature Extraction & Search
4. Shape Realignment on the ROI
5. Output the Shape
Algorithm

1. Face Detection
2. Start Shape on the ROI
3. Feature Extraction & Search
4. Shape Realignment on the ROI
5. Output the Shape
Algorithm

1. Face Detection
2. Start Shape on the ROI
3. Feature Extraction & Search
4. Shape Realignment on the ROI
5. Output the Shape
Algorithm:

1. Face Detection
2. Start Shape on the ROI
3. Feature Extraction & Search
4. Shape Realignment on the ROI
5. Output the Shape
Algorithm

1. Face Detection
2. Start Shape on the ROI
3. Feature Extraction & Search
4. Shape Realignment on the ROI
5. Output the Shape
Experiment

• Data: 1000 frames, 800x600 (hy1.avi)

• Intel(R) Core(TM) i3-2350 CPU @ 2.30 GHz
  – LDB-ASM: 15.503s (64fps)
  – HAT-ASM: 65.123s (15fps)

• Mobile Platform (Qualcomm 8974)
  – LDB-ASM: 00:01:22m (12fps)
  – HAT-ASM: 00:08:13m (2fps)
LDB-ASM

HAT-ASM
Active Shape Models

• Advantages
  – Fast, simple, accurate
  – Efficient to extend to 3D

• Disadvantages
  – Only sparse use of image information
  – Treat local models as independent
Cascade Shape Regression Framework

Stage $t = 0$  $t = 3$  $t = 5$

$S^t = S^{t-1} + R^t(I, S^{t-1})$

Cascaded pose regression, Dollar et. al., CVPR 2010

Regressor $R^t(I, S^{t-1})$ is learnt to minimize the shape residual on training data

$$R^t = \arg\min_R \sum_i |\Delta \hat{S}_i - R(I_i, S_i^{t-1})|$$

$\Delta \hat{S} = \hat{S} - S^{t-1}$: ground truth shape residual
Face Alignment based on LBF-Regression

• Tree Induced Local Binary Features
  – learned from data
  – global optimization
    • much stronger than previous regression trees
  – efficient training / testing

• Best accuracy on challenging benchmarks

• 3,000 FPS on desktop, or 300 FPS on mobile
  – first face tracking method on mobile
Face Alignment based on LBF-Regression

\[ \Delta S^t = W^t \Phi^t (I, S^{t-1}) \]

where \( I \) is the input image, \( S^{t-1} \) is the shape from the previous stage, \( \Phi^t \) is a feature mapping function, and \( W^t \) is a linear regression matrix. Note that \( \Phi^t \) depends on both \( I \) and \( S^{t-1} \). The feature learned in this way is referred to as **Shape-index Feature**
Face Alignment based on LBF-Regression

• A simple form
  – sum of a large number of regression trees

\[ R^t(I, S^{t-1}) = \sum_{k=1}^{K} \text{reg\_tree}_k (I, S^{t-1}) \]

• Novel two step learning
  1. Local learning of tree structure
     • learn an easier task and better features
  2. Global optimization of tree output
     • enforce dependence between points and reduce local estimation errors
Local Learning of Tree Structure

- learn standard random forests for each local point
  - standard regression tree using pixel difference features
- only use pixels in the local patch around the point
  - regularization of feature selection

```
We use a standard regression random forest [2] to learn each local mapping function $\phi^t_i$. The split nodes in the trees are trained using the pixel-difference feature [5, 3]. To train each split node, we test 500 randomly sampled features and pick the feature that gives rise to maximum variance reduction. Testing more features results in only marginal improvement in our experiment. After training, each leaf node stores a 2D offset vector that is the average of all the training samples in the leaf.
```
From Local to Global

Fix tree structures and optimize tree leave’s output
Global Optimization of Tree Output

- Estimated Shape $S^t$
- Ground Truth Shape $\hat{S}$

Regression Target

Feature Mapping Function
Global Optimization of Tree Output
Tree Induced Binary Features

• Each leave is a binary indicator function
  - 1 if the image sample arrives at the leaf
  - 0 otherwise

• Trees -> high dimension sparse binary features

• Learning global linear regression $\mathbf{W}^t$

$$\min_{W^t} \sum_{i=1}^{N} \| \Delta \hat{S}^t_i - W^t \Phi^t (I_i, S_i^{t-1}) \|_2^2 + \lambda \| W^t \|_2^2,$$

where the first term is the regression target, the second term is a L2 regularization on $W^t$, and $\lambda$ controls the regularization strength.
Experiments

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#landmarks</th>
<th>#training images</th>
<th>#testing images</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFPW</td>
<td>29</td>
<td>717</td>
<td>249</td>
</tr>
<tr>
<td>Helen</td>
<td>194</td>
<td>2000</td>
<td>330</td>
</tr>
<tr>
<td>300-W</td>
<td>68</td>
<td>3149</td>
<td>689</td>
</tr>
</tbody>
</table>

- Two variants of our method
  - Accurate: \( LBF \) 1200 trees with depth 7
  - Fast: \( LBF \) \textit{fast} 300 trees with depth 5
<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>3.99</td>
<td>≈ 1</td>
</tr>
<tr>
<td>ESR [2]</td>
<td>3.47</td>
<td>220</td>
</tr>
<tr>
<td>RCPR [3]</td>
<td>3.50</td>
<td>-</td>
</tr>
<tr>
<td>SDM [4]</td>
<td>3.49</td>
<td>160</td>
</tr>
<tr>
<td>EGM [5]</td>
<td>3.98</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>LBF</td>
<td>3.35</td>
<td>460</td>
</tr>
<tr>
<td>LBF fast</td>
<td>3.35</td>
<td>4200</td>
</tr>
</tbody>
</table>

### Helen (194 landmarks)

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>STASM [6]</td>
<td>11.1</td>
<td>-</td>
</tr>
<tr>
<td>CompASM [7]</td>
<td>9.10</td>
<td>-</td>
</tr>
<tr>
<td>ESR [2]</td>
<td>5.70</td>
<td>70</td>
</tr>
<tr>
<td>PCPR [3]</td>
<td>6.50</td>
<td>-</td>
</tr>
<tr>
<td>SDM [4]</td>
<td>5.85</td>
<td>21</td>
</tr>
<tr>
<td>LBF</td>
<td>5.41</td>
<td>200</td>
</tr>
<tr>
<td>LBF fast</td>
<td>5.80</td>
<td>1500</td>
</tr>
</tbody>
</table>

### 300-W (68 landmarks)

<table>
<thead>
<tr>
<th>Method</th>
<th>Fullset</th>
<th>Common Subset</th>
<th>Challenging Subset</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESR [2]</td>
<td>7.58</td>
<td>5.28</td>
<td>17.00</td>
<td>120</td>
</tr>
<tr>
<td>SDM [4]</td>
<td>7.52</td>
<td>5.60</td>
<td>15.40</td>
<td>70</td>
</tr>
<tr>
<td>LBF</td>
<td>6.32</td>
<td>4.95</td>
<td>11.98</td>
<td>320</td>
</tr>
<tr>
<td>LBF fast</td>
<td>7.37</td>
<td>5.38</td>
<td>15.50</td>
<td>3100</td>
</tr>
</tbody>
</table>

LBF is much more accurate and a few times faster
LBF fast is slightly more accurate and dozens of times faster
Face Alignment at 3000 FPS
Summary

• State-of-the-art face alignment

• Best accuracy on challenging benchmarks

• Dozens of times faster than previous methods
  – faster than real time face tracking on mobile