Virtual Cinematography: Making The Matrix' Sequels
George Borshukov
VFX Technology Supervisor, ESC Entertainment
Friday, June 4, 2004
1:00 p.m. to 2:30 p.m.
[Pizza lunch will precede the event from noon to 1 p.m.]
Main Auditorium, San Diego Supercomputer Center

The presentation will cover the key technologies that had to be developed and deployed to create the synthetic human sequences in the Matrix sequels including Universal Capture - image-based facial animation, realistic human face rendering, and use of measured BRDF in film production. It will also feature a breakdown of The Superpunch shot (pictured above) from "The Matrix Revolutions" (the bullet time punch that Neo delivers to Agent Smith during the film's last face-off). This difficult, important, expensive, and challenging shot was entirely computer generated and showcased the technological developments of 3.5+ years at their best by showing a full-frame close-up of a known human actor.

Mathematical formulation
[Note change of notation: image coordinates now (x,y), not (u,v)]

\[
I(x,y,t) = \text{brightness at image point (x,y) at time t}
\]

Consider scene (or camera) to be moving, so \(x(t), y(t)\)

\[
\begin{align*}
\frac{\partial I}{\partial t} + \frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} & = 0 \\
\end{align*}
\]

Brightness constancy assumption:

\[
I(x + \frac{dx}{dt}, y + \frac{dy}{dt}, t + \frac{dt}{dt}) = I(x,y,t)
\]

Optical flow constraint equation:

\[
\frac{dl}{dt} = 0
\]

\[
\frac{dl}{dt} = \frac{\partial l}{\partial x} \frac{dx}{dt} + \frac{\partial l}{\partial y} \frac{dy}{dt} + \frac{\partial l}{\partial t} = 0
\]

Optical Flow: Where do pixels move to?

Aperture Problem and Normal Flow
Two ways to get flow

1. Think globally, and regularize over image
2. Look over window and assume constant motion in the window

Recognition

Recognition Challenges

- Within-class variability
  - Different objects within the class have different shapes or different material characteristics
  - Deformable
  - Articulated
  - Compositional
- Pose variability:
  - 2-D Image transformation (translation, rotation, scale)
  - 3-D Pose Variability (perspective, orthographic projection)
- Lighting
  - Direction (multiple sources & type)
  - Color
  - Shadows
- Occlusion – partial
- Clutter in background -> false positives

Object Recognition Issues:

- How general is the problem?
  - 2D vs. 3D
  - range of viewing conditions
  - available context
  - segmentation cues
- What sort of data is best suited to the problem?
  - Whole images
  - Local 2D features (color, texture,
  - 3D (range) data
- What information do we have in the database?
  - Collection of images?
  - 3D models?
  - Learned representation?
  - Learned classifiers?
- How many objects are involved?
  - small: brute force search
  - large ??
A Rough Recognition Spectrum

### Appearance-Based Recognition (Eigenface, Fisherface)
- Shape Contexts
- Geometric Invariants
- Aspect Graphs
- 3-D Model-Based Recognition

Local Features + Spatial Relations → Increasing Generality

### Increasing Generality

#### Appearance-Based Vision:
**A Pattern Classification Viewpoint**
1. Feature Space + Nearest Neighbor
2. Dimensionality Reduction
3. Bayesian Classification
4. Appearance Manifolds

### Sketch of a Pattern Recognition Architecture

- **Image (window)** → **Feature Extraction** → **Classification** → **Object Identity**

### Example: Face Detection
- Scan window over image.
- Classify window as either:
  - Face
  - Non-face

### The Problem of Recognition
Given an image $I$ and a database of $k$ objects and a representation $R_j$ for object $j$ in the database, recognition can be expressed as:

$$i = \arg \min_{j \in [1, \ldots, k]} c(R_j, I)$$

where $c(R_j, I)$ is a function which gives the compatibility or consistency of representation $R_j$ with the image.

### Image as a Feature Vector
- Consider an $n$-pixel image (window) to be a point in an $n$-dimensional space, $x \in \mathbb{R}^n$.
- Each pixel value is a coordinate of $x$. 
Simplest Recognition Scheme

- $R_i$ is an image.
- $c(R_i, I)$ is Euclidean distance.

Nearest Neighbor Classifier

Comments

- Sometimes called “Template Matching”
- Variations on distance function (e.g. $L_1$, robust distances)
- Multiple templates per class - perhaps many training images per class.
- Expensive to compute $k$ distances, especially when each image is big ($N$ dimensional).
- May not generalize well to unseen examples of class.
- Some solutions:
  - Dimensionality reduction
  - Bayesian classification

Eigenfaces: Linear Projection

- An $n$-pixel image $x \in \mathbb{R}^n$ can be projected to a low-dimensional feature space $y \in \mathbb{R}^m$ by
  $$y = Wx$$
  where $W$ is an $m \times n$ matrix.
- Recognition is performed using nearest neighbor in $\mathbb{R}^m$.
- How do we choose a good $W$?

Eigenfaces: Principal Component Analysis (PCA)

Assume we have a set of $n$ feature vectors $x_i (i = 1, \ldots, n)$ in $\mathbb{R}^d$. Write
  $$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$
  $$\Sigma = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T$$

The unit eigenvectors of $\Sigma$ — which we write as $w_1, w_2, \ldots, w_k$, where the order is given by the size of the eigenvalues and $w_1$ has the largest eigenvalue — give a set of features with the following properties:
- They are independent.
- Projection onto the basis $(w_1, \ldots, w_k)$ gives the $k$-dimensional set of linear features that preserves the most variance.

Algorithm 22.8: Principal components analysis identifies a collection of linear features that are independent, and captures as much variance as possible from a dataset.
Some details: Use Singular value decomposition, “trick” described in appendix of text to compute basis when $m << d$.

Eigenfaces

- Modeling
  1. Given a collection of $n$ labeled training images,
  2. Compute mean image and covariance matrix.
  3. Compute $k$ Eigenvectors (note that these are images) of covariance matrix corresponding to $k$ largest Eigenvalues. (Or perform using SVD!!)
  4. Project the training images to the $k$-dimensional Eigenspace.
- Recognition
  1. Given a test image, project to Eigenspace.
  2. Perform classification to the projected training images.
**Eigenfaces: Training Images**

![Eigenfaces: Training Images][1]

[ Turk, Pentland 91]

---

**Eigenfaces**

Mean Image

Basis Images

---

**Basis Images for Variable Lighting**

![Basis Images for Variable Lighting][2]

---

**Projection, and reconstruction**

- An \( n \)-pixel image \( x \in \mathbb{R}^n \) can be projected to a low-dimensional feature space \( y \in \mathbb{R}^m \) by
  \[
  y = Wx
  \]
- From \( y \in \mathbb{R}^m \), the reconstruction of the point is \( W^Ty \)
- The error of the reconstruction is:
  \[
  ||x - W^TWx||
  \]

---

**Reconstruction using Eigenfaces**

- Given image on left, project to Eigenspace, then reconstruct an image (right).

---

**Face detection using “distance to face space”**

- Scan a window \( \omega \) across the image, and classify the window as face/not face as follows:
  - Project window to subspace, and reconstruct as described earlier.
  - Compute distance between \( \omega \) and reconstruction.
  - Local minima of distance over all image locations less than some threshold are taken as locations of faces.
  - Repeat at different scales.
  - Possibly normalize windows intensity so that \( ||\omega|| = 1 \).