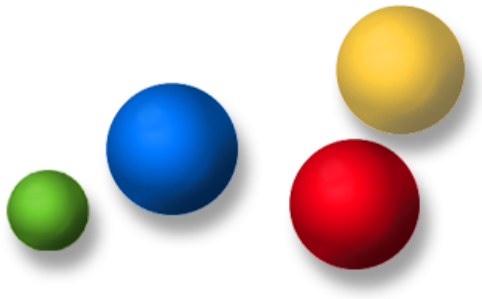
Large-Scale Face Manifold Learning

Sanjiv Kumar

Google Research
New York, NY



* Joint work with A. Talwalkar, H. Rowley and M. Mohri

Face Manifold Learning

50 x 50 pixel faces

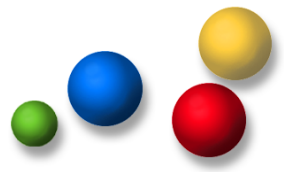
$\rightarrow \Re^{2500}$

50 x 50 pixel random images

Space of face images significantly smaller than 256^{2500}

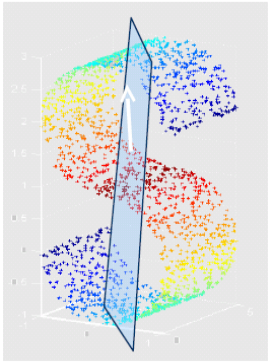
Want to recover the underlying (possibly nonlinear) space !

(Dimensionality Reduction)

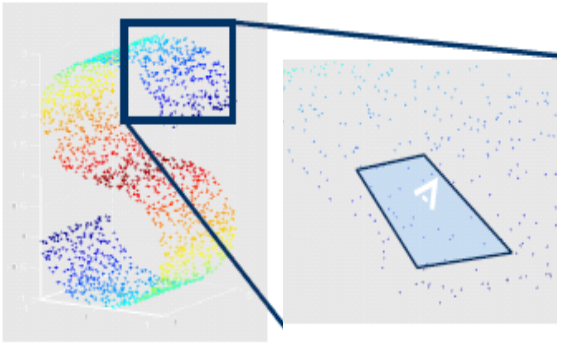


Dimensionality Reduction

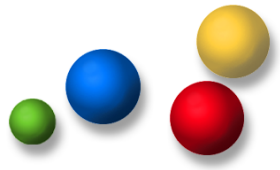
- **Linear Techniques**
  - **PCA, Classical MDS**
  - **Assume data lies in a subspace**
  - **Directions of maximum variance**



- **Nonlinear Techniques**
  - **Manifold learning methods**
    - LLE [Roweis & Saul '00]
    - ISOMAP [Tenanbaum et al. '00]
    - Laplacian Eigenmaps [Belkin & Niyogi '01]
  - **Assume local linearity of data**
  - **Need densely sampled data as input**

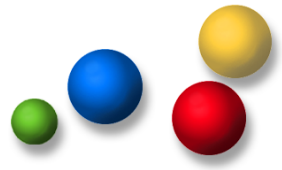


Bottleneck: Computational Complexity ≈ $O(n^3)$!



Outline

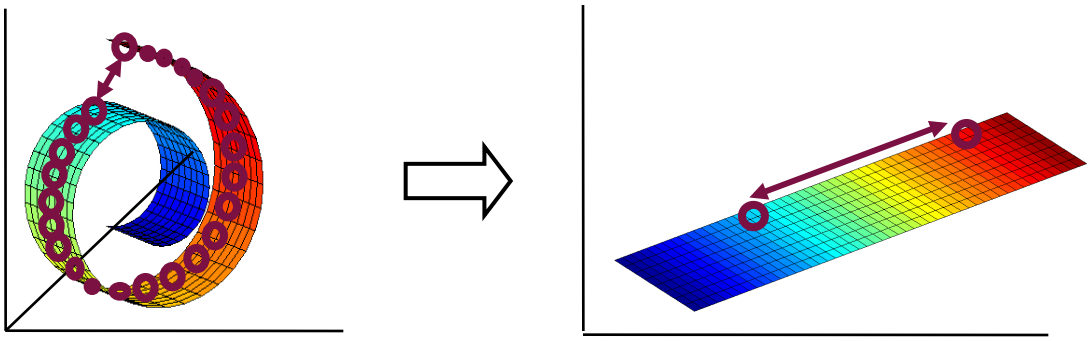
- **Manifold Learning**
  - **ISOMAP**
- **Approximate Spectral Decomposition**
  - **Nystrom and Column-Sampling approximations**
- **Large-scale Manifold learning**
  - **18M face images from the web**
  - **Largest study so far ~270 K points**
- **People Hopper – A Social Application on Orkut**

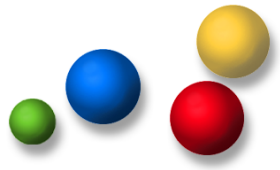


ISOMAP

[Tanenbaum et al., ’00]

- **Find the low-dimensional representation that best preserves geodesic distances between points**

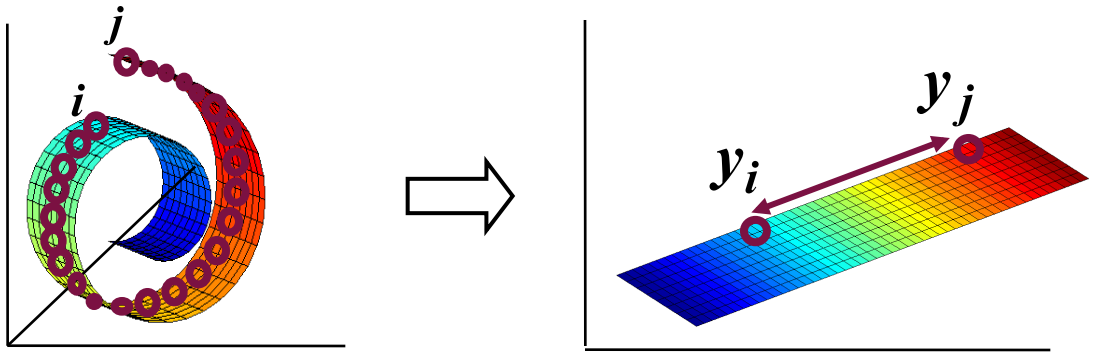




ISOMAP

[Tanenbaum et al., '00]

- **Find the low-dimensional representation that best preserves geodesic distances between points**



$$Y^* = \arg\min_{Y} \sum_{i,j} \left(\left\| y_i - y_j \right\|_2 - \Delta_{ij} \right)^2$$

Output co-ordinates

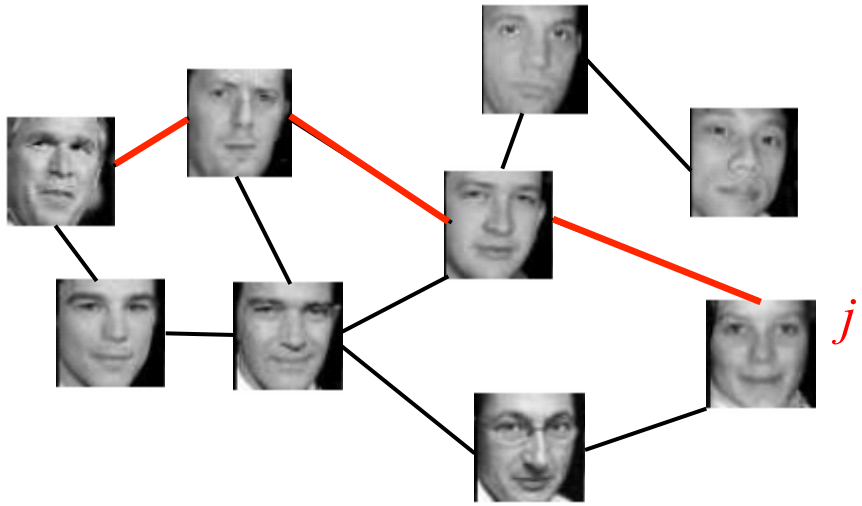
Geodesic distance

Recovers true manifold asymptotically !

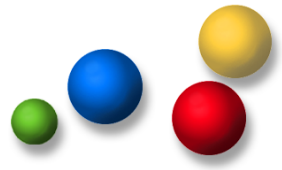
ISOMAP [Tanenbaum et al., '00]

Given n input images:

- Find t nearest neighbors for each image : $O(n^2)$
- Find shortest path distance for every (i, j), Δ_{ij} : $O(n^2 \log n)$
- Construct $n \times n$ matrix G with entries as centered Δ_{ij}^2
  - $G \sim$ 18M x 18M dense matrix
- Optimal k reduced dims: $U_k \Sigma_k^{1/2}$



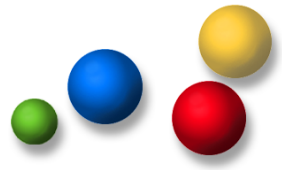
Eigenvectors Eigenvalues $O(n^3)$!



Spectral Decomposition

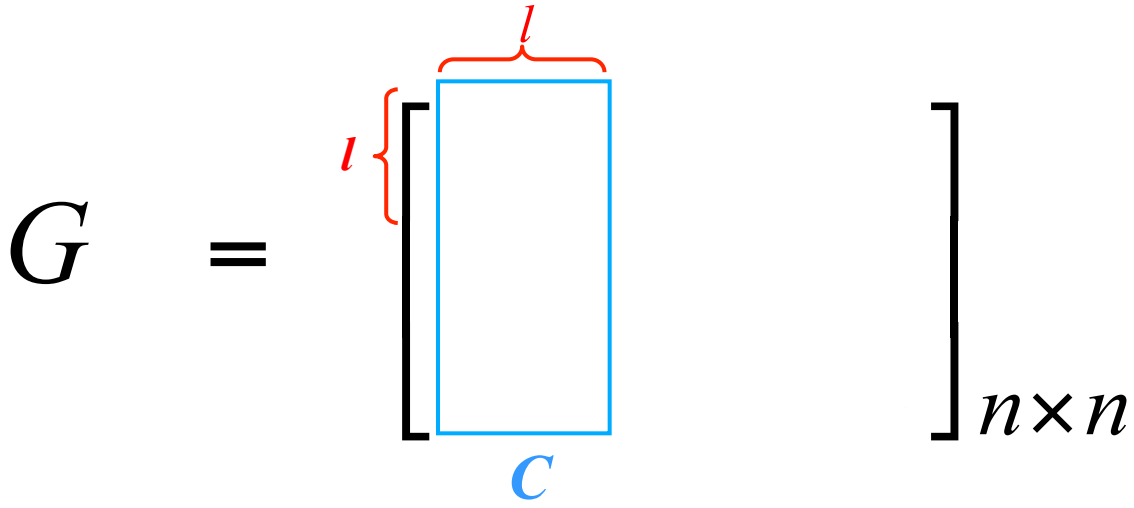
- **Need to do eigen-decomposition of symmetric positive semi-definite matrix** $[G]_{n\times n}$ **O(n^3)**
- **For** $n = 18\text{M}$, $G \approx$ **1300 TB**
  - **~100,000 x 12GB RAM machines**
- **Iterative methods**
  - **Jacobi, Arnoldi, Hebbian** [Golub & Loan, '83][Gorell, '06]
  - **Need matrix-vector products and several passes over data**
  - **Not suitable for large dense matrices**
- **Sampling-based methods**
  - **Column-Sampling Approximation** [Frieze et al., '98]
  - **Nystrom Approximation** [Williams & Seeger, '00]

Relationship and comparative performance?



Approximate Spectral Decomposition

- **Sample l columns randomly without replacement**

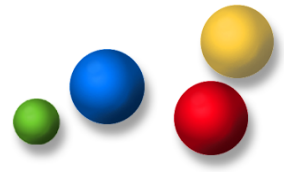


- **Column-Sampling Approximation – SVD of C**

[Frieze et al., '98]

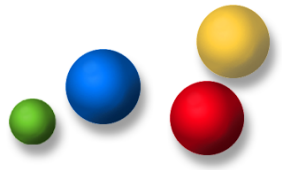
- **Nystrom Approximation – SVD of W**

[Williams & Seeger, '00][Drineas & Mahony, '05]



Column-Sampling Approximation

$$C = U_c \sum_c V_c^T$$

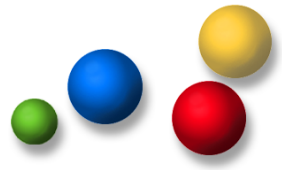


Column-Sampling Approximation

$$C = U_c \sum_c V_c^T$$

$$\tilde{U}_G = U_c = CV_c \sum_c^{-1}$$

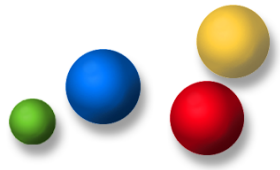
$$\tilde{\sum}_G = \sqrt{\frac{n}{l}} \sum_c$$



Column-Sampling Approximation

$$\underset{[n \times l]}{C} = U_c \sum_c V_c^T \qquad \mathbf{O}(nl^2)\ \mathbf{!}$$

$$\left.\begin{array}{l} \boxed{\tilde{U}_G = U_c = C V_c \sum_c^{-1}} \\ \boxed{\tilde{\sum}_G = \sqrt{\frac{n}{l}} \sum_c} \end{array}\right\} \underset{[l \times l]}{C^T C} = V_c \sum_c^2 V_c^T \qquad \mathbf{O}(l^3)\ \mathbf{!}$$

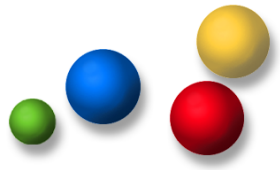


Nystrom Approximation

$$G = \begin{bmatrix} W & G_{21}^T \\ G_{21} & G_{22} \end{bmatrix}_{n \times n}$$

(W: $l \times l$; C: $n \times l$)

$$G \approx \tilde{G} = CW^{-1}C^T$$



Nystrom Approximation

$$G = \begin{bmatrix} W & G_{21}^T \\ G_{21} & G_{22} \end{bmatrix}_{n \times n}$$

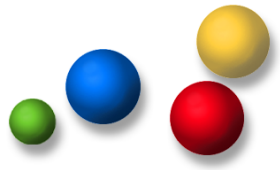
(W: $l \times l$; C: $n \times l$)

$$G \approx \tilde{G} = C W^{-1} C^T$$

$$W = U_W \Sigma_W U_W^T \qquad \mathrm{O}(l^3)\ !$$

$$\tilde{\Sigma}_G = \frac{n}{l} \Sigma_W$$

$$\tilde{U}_G = \sqrt{\frac{l}{n}} C U_W \Sigma_W^{-1}$$



Nystrom Approximation

$$G = \begin{bmatrix} W & G_{21}^T \\ G_{21} & G_{22} \end{bmatrix}_{n \times n}$$

(W: $l \times l$; C: $n \times l$)

$$G \approx \tilde{G} = C W^{-1} C^T$$

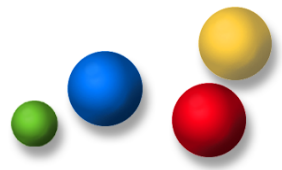
$$W = U_W \Sigma_W U_W^T \qquad O(l^3)\ !$$

$$\tilde{\Sigma}_G = \frac{n}{l} \Sigma_W$$

$$\tilde{U}_G = \sqrt{\frac{l}{n}} C U_W \Sigma_W^{-1}$$

Not Orthonormal !

$$\tilde{U}_G^T \tilde{U}_G \neq I$$



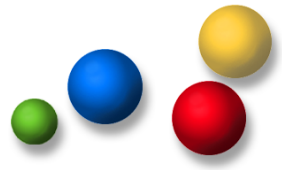
Nystrom Vs Column-Sampling

$$\tilde{G}_{nys} = CW^{-1}C^T$$

$$\tilde{G}_{col} = C\left(\left[\frac{l}{n}C^TC\right]^{1/2}\right)^{-1}C^T$$

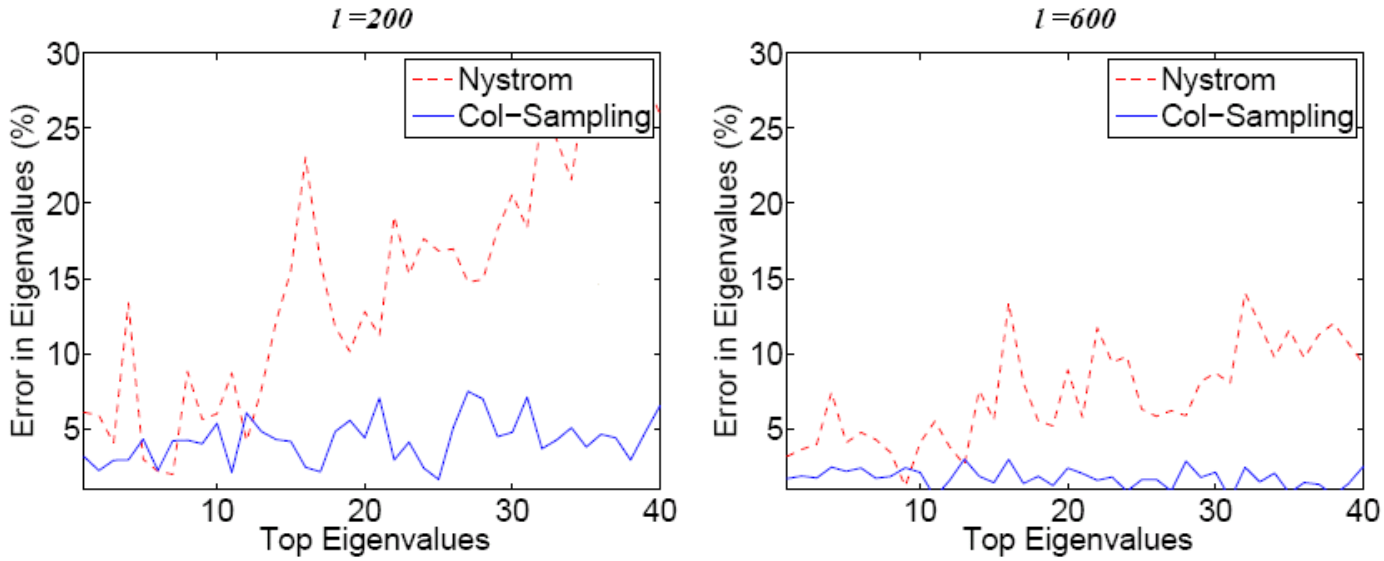
- **Experimental Comparison**
  - **A random set of 7K face images**
  - **Eigenvalues, eigenvectors, and low-rank approximations**

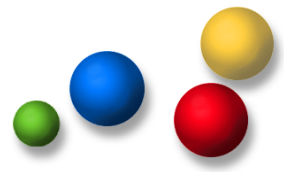
[Kumar, Mohri & Talwalkar, ICML '09]



Eigenvalues Comparison

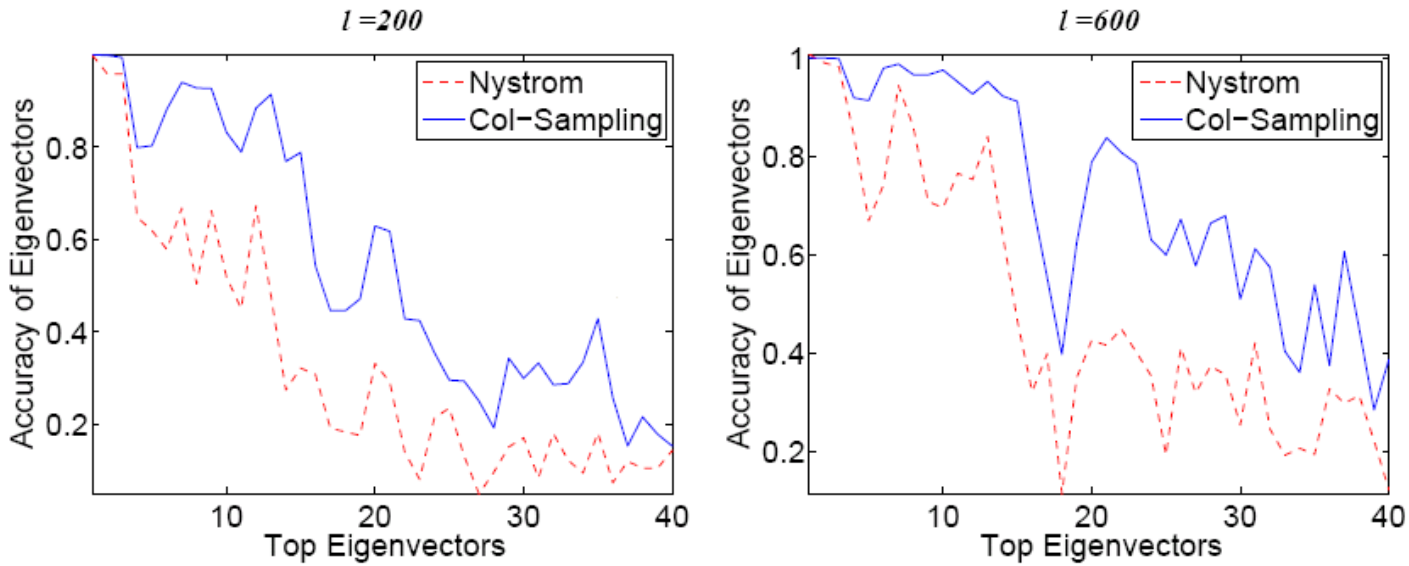
% deviation from exact

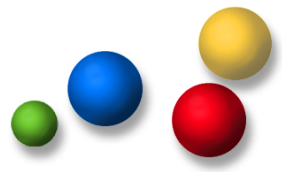




Eigenvectors Comparison

Principal angle with exact

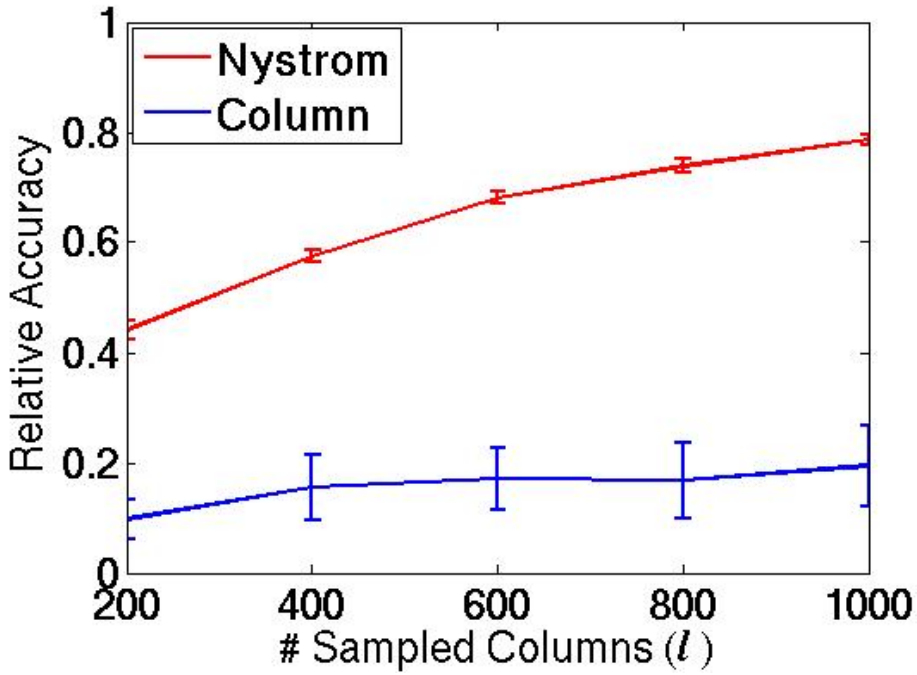




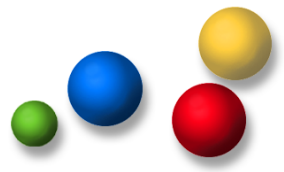
Low-Rank Approximations

$$\tilde{G}_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{U}_k^T$$

$$\frac{\left\| G - \hat{G}_k \right\|_F}{\left\| G - \tilde{G}_k \right\|_F}$$



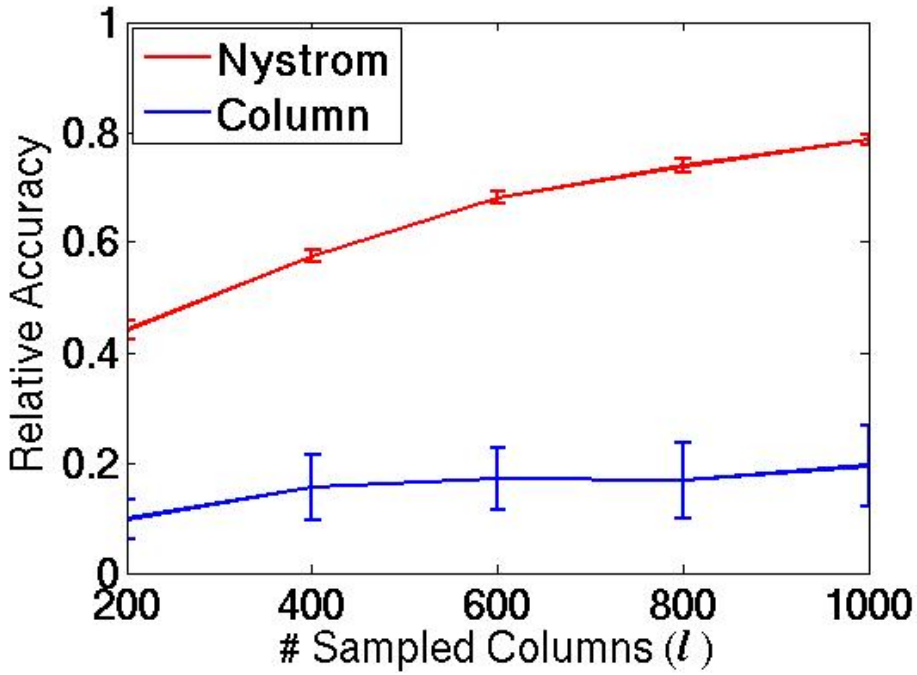
Nystrom gives better reconstruction than Col-Sampling !



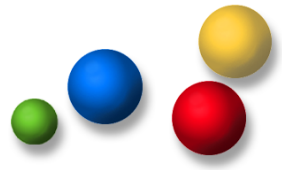
Low-Rank Approximations

$$\tilde{G}_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{U}_k^T$$

$$\frac{\left\| G - \hat{G}_k \right\|_F}{\left\| G - \tilde{G}_k \right\|_F}$$



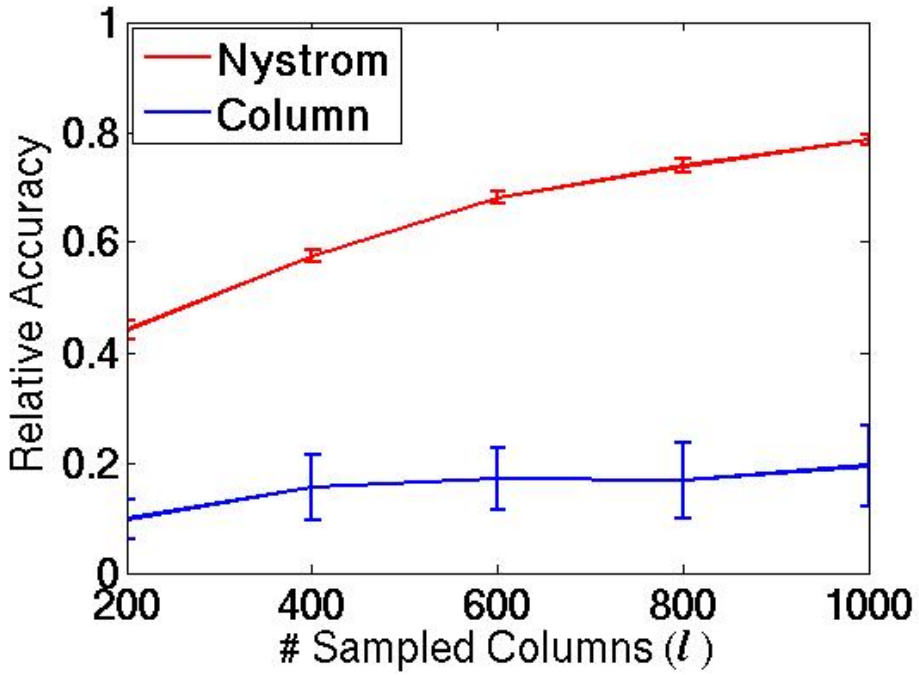
$$\left.\begin{aligned} \tilde{U}_{col} &= C V_c \Sigma_c^{-1} \\ \tilde{\Sigma}_{col} &= \sqrt{\frac{n}{l}} \Sigma_c \end{aligned}\right\} \tilde{G}_{col} = C \left(\left[\frac{l}{n} C^T C \right]^{1/2} \right)^{-1} C^T$$



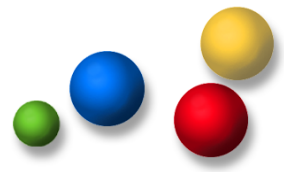
Low-Rank Approximations

$$\tilde{G}_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{U}_k^T$$

$$\frac{\left\| G - \hat{G}_k \right\|_F}{\left\| G - \tilde{G}_k \right\|_F}$$



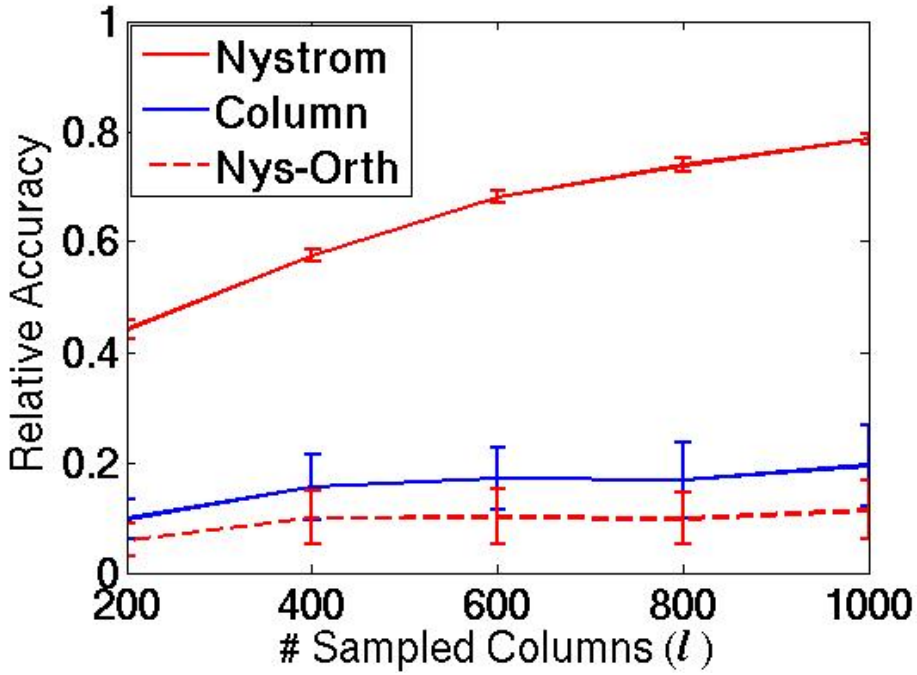
$$\left.\begin{aligned} \tilde{U}_{col} &= C V_c \Sigma_c^{-1} \\ \tilde{\Sigma}_{col} &= \sqrt{\frac{n}{l}} \Sigma_c \end{aligned}\right\} \tilde{G}_{col} = C\left(\left[\frac{l}{n} C^T C\right]^{1/2}\right)^{-1} C^T \qquad \left.\begin{aligned} \tilde{U}_{nys} &= \sqrt{\frac{l}{n}} C U_W \Sigma_W^{-1} \\ \tilde{\Sigma}_{nys} &= \frac{n}{l} \Sigma_W \end{aligned}\right\} \tilde{G}_{nys} = C W^{-1} C^T$$



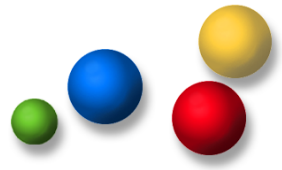
Orthogonalized Nystrom

$$\tilde{G}_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{U}_k^T$$

$$\frac{\left\| G - \hat{G}_k \right\|_F}{\left\| G - \tilde{G}_k \right\|_F}$$

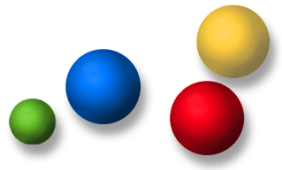


Nystrom-orthogonal gives worse reconstruction than Nystrom !



Low-Rank Approximations
Matrix Projection

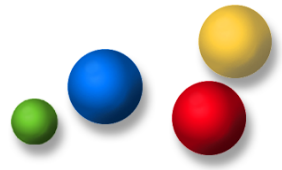
$$G_k = U_k \sum_k U_k^T$$



Low-Rank Approximations
Matrix Projection

$$G_k = U_k \sum_k U_k^T = U_k U_k^T G = G U_k U_k^T$$

$$\tilde{G}_k = \tilde{U}_k \tilde{U}_k^T G \neq \tilde{U}_k \tilde{\sum}_k \tilde{U}_k^T$$



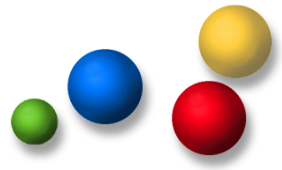
Low-Rank Approximations
Matrix Projection

$$G_k = U_k \sum_k U_k^T = U_k U_k^T G = G U_k U_k^T$$

$$\tilde{G}_k = \tilde{U}_k \tilde{U}_k^T G \neq \tilde{U}_k \tilde{\sum}_k \tilde{U}_k^T$$

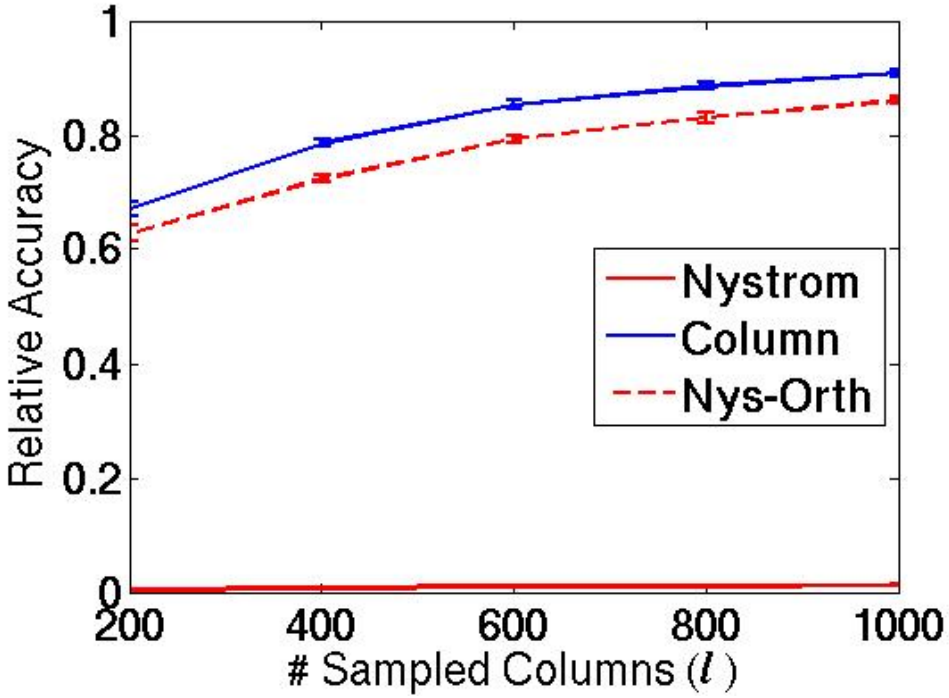
$$\tilde{G}_{col} = C\left(C^T C\right)^{-1} C^T G$$

$$\tilde{G}_{nys} = C\left(\frac{l}{n} W^{-2}\right) C^T G$$



Low-Rank Approximations
Matrix Projection

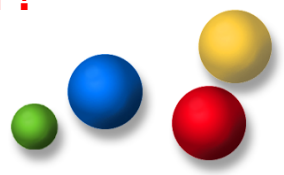
$$\widetilde{G}_k = \widetilde{U}_k \widetilde{U}_k^T G$$



Col-Sampling gives better Reconstruction than Nystrom !

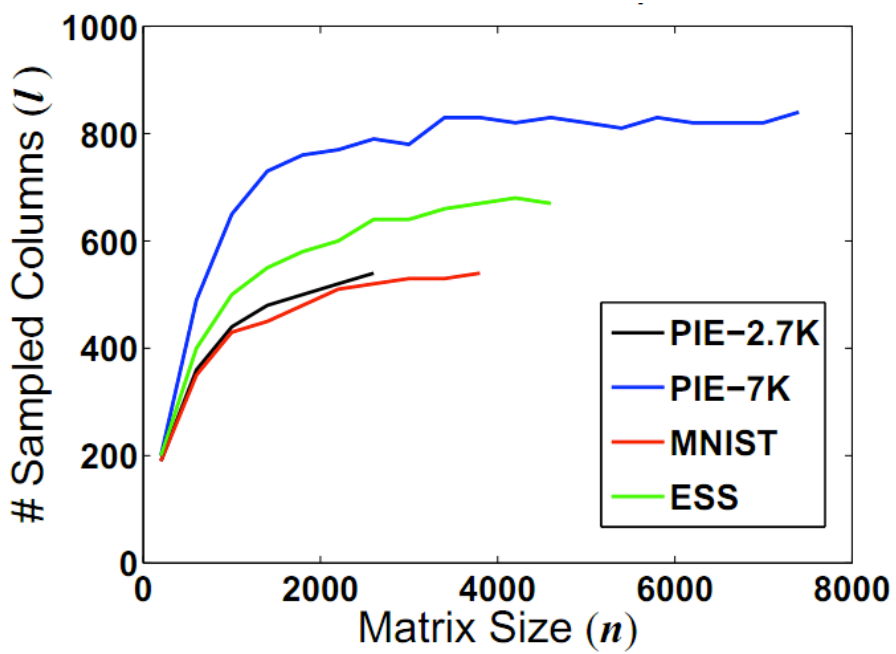
– **Theoretical guarantees in special cases**

[Kumar et al., ICML '09]

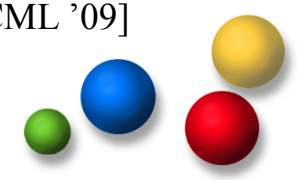


How many columns are needed?

Columns needed to get 75% relative accuracy

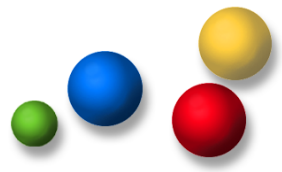


- **Sampling Methods**
  - **Theoretical analysis of uniform sampling method** [Kumar et al., AISTATS '09]
  - **Adaptive sampling methods** [Deshpande et al. FOCS '06] [Kumar et al., ICML '09]
  - **Ensemble sampling methods** [Kumar et al., NIPS '09]



So Far …

- Manifold Learning
  - ISOMAP
- Approximate Spectral Decomposition
  - Nystrom and Column-Sampling approximations
- **Large-scale Face Manifold learning**
  - **18 M face images from the web**
- **People Hopper – A Social Application on Orkut**



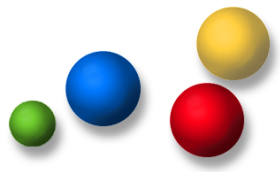
Large-Scale Face Manifold Learning

[Talwalkar, Kumar & Rowley, CVPR '08]

- **Construct Web dataset**
  - **Extracted 18M faces from 2.5B internet images**
  - **~15 hours on 500 machines**
  - **Faces normalized to zero mean and unit variance**
- **Graph construction**
  - **Exact search ~3 months (on 500 machines)**
  - **Approx Nearest Neighbor – Spill Trees (5 NN, ~2 days)**
  - **New methods for hashing based kNN search**
  - **Less than 5 hours!**

[Liu et al., '04]

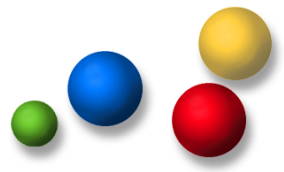
[CVPR '10] [ICML '10] [ICML '11]



Neighborhood Graph Construction

- Connect each node (face) with its neighbors
- Is the graph connected?
  - Depth-First-Search to find largest connected component
  - 10 minutes on a single machine
  - Largest component depends on number of NN (t)

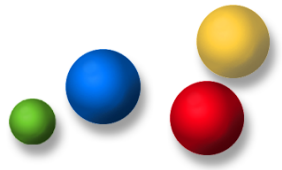
t	# Comp	% Largest
1	4.3M	0.03 %
2	285K	80.1 %
3	277K	82.2 %
5	275K	83.1 %



Samples from connected components

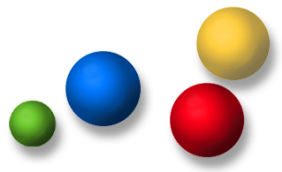
From Largest Component

From Smaller Components

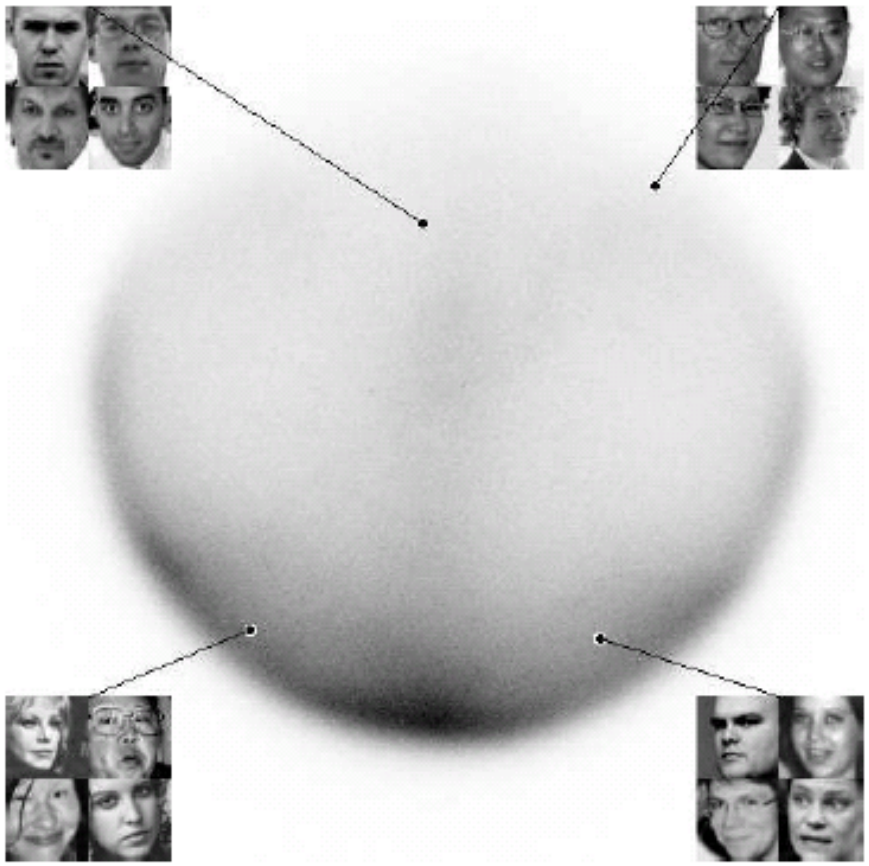


Graph Manipulation

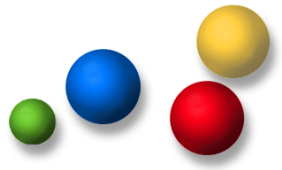
- **Approximating Geodesics**
  - Shortest paths between pairs of face images
  - Computing for all pairs infeasible $O(n^2 \log n)$!
- **Key Idea: Need only a few columns of G for sampling-based decomposition**
  - require shortest paths between a few (l) nodes and all other nodes
  - 1 hour on 500 machines (l = 10K)
- **Computing Embeddings (k = 100)**
  - Nystrom: 1.5 hours, 500 machine
  - Col-Sampling: 6 hours, 500 machines
  - Projections: 15 mins, 500 machines



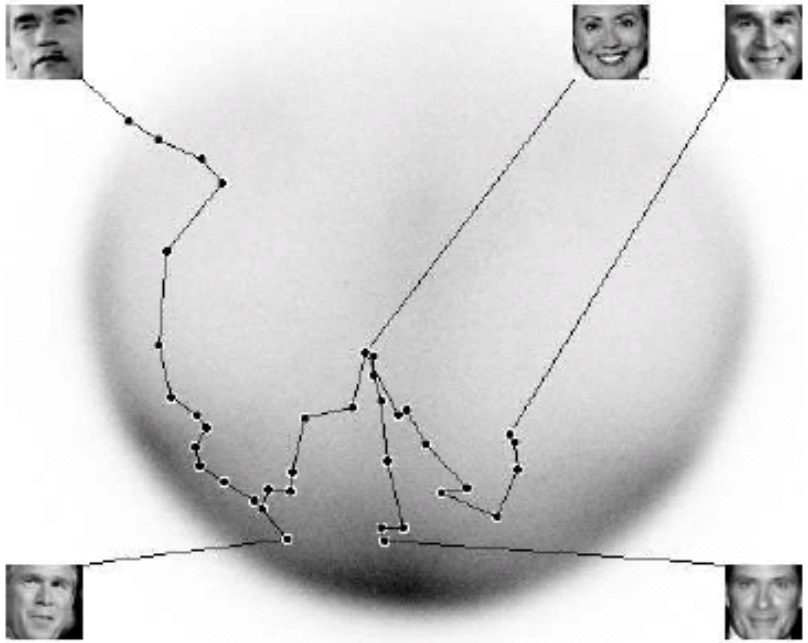
18M-Manifold in 2D



Nystrom Isomap



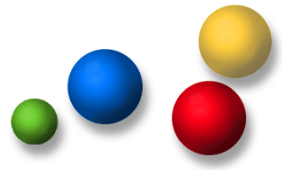
Shortest Paths on Manifold



18M samples not enough!

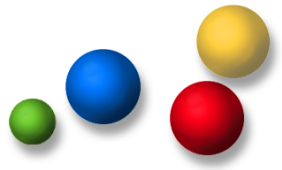
Summary

- Large-scale nonlinear dimensionality reduction using manifold learning on 18M face images
- Fast approximate SVD based on sampling methods
- Open Questions
  - Does a manifold really exist or data may form clusters in low dimensional subspaces?
  - How much data is really enough?



People Hopper

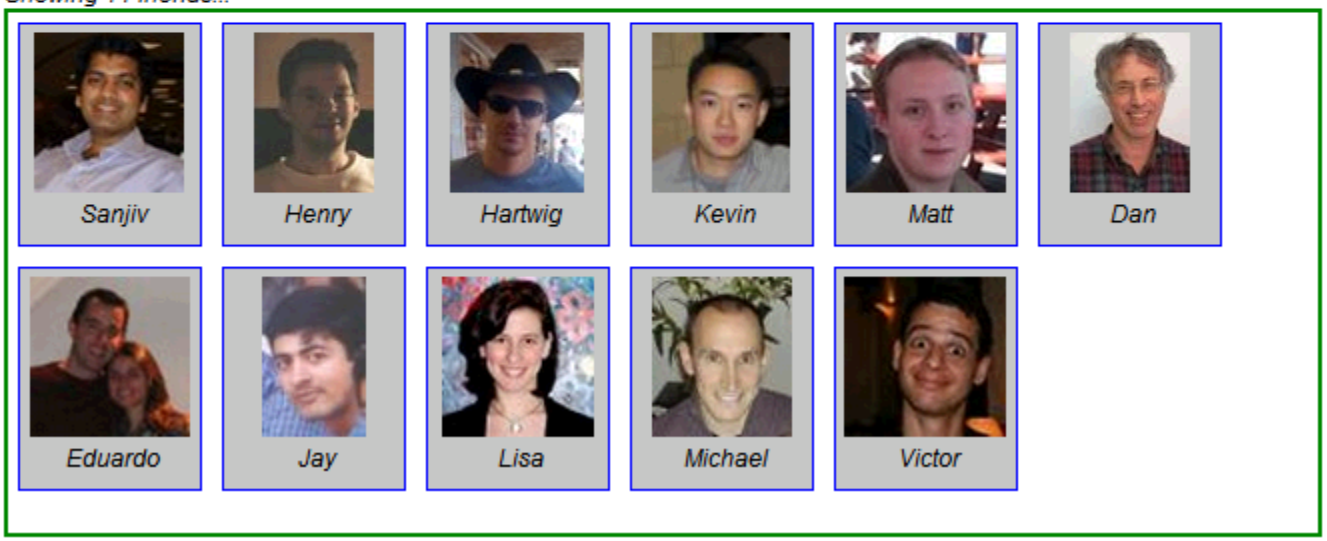
- A fun social application on Orkut
- Face manifold constructed with Orkut database
  - Extracted 13M faces from about 146M profile images
  - ~3 days on 50 machines
  - Color face image (40x48 pixels) → 5760-dim vector
  - Faces normalized to zero mean and unit variance in intensity space
- Shortest path search using bidirectional Dijkstra
- Users can opt-out – Daily incremental graph update



People Hopper Interface

Cancel Path

Showing 11 friends...



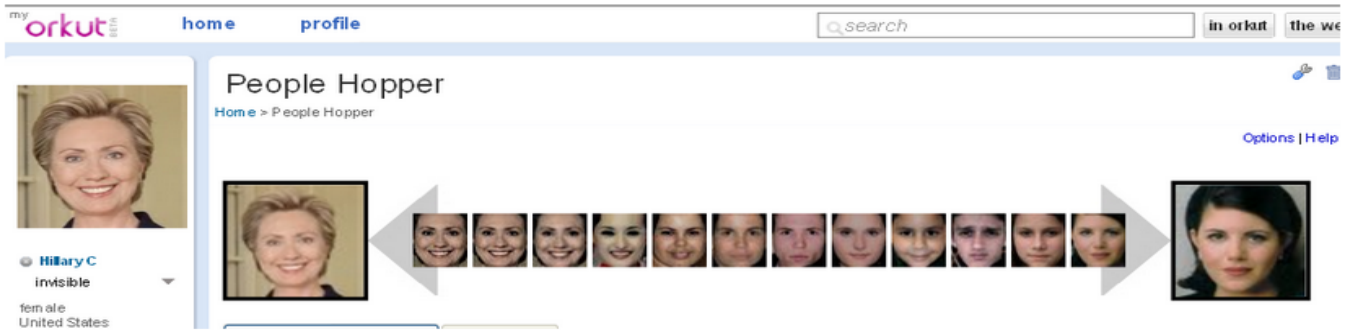
From the Blogs

Google Labs Presents: People Hopper

The mad scientists at Google unleashed their latest Google Labs experiment, People Hopper, to the world this last Wednesday on January 27th, 2010. So what is **People Hopper** you ask? Basically it is an application for Google's social networking site Orkut, which allows members to take their profile picture and morph it into a friend's profile picture. After you install the application you will be able to compare two of your friends (they must be Orkut members) using the People Hooper application. Simply drag your friends' photo into the application and within a couple of seconds, People Hooper will display a clickable "path" that transitions your image to that of a friends'. Google Labs use image matching technology to power their People Hopper application.

1 tweet

retweet



We thought it may be interesting using People Hopper to compare Hillary Clinton with Monica Lewinsky… just for fun of course

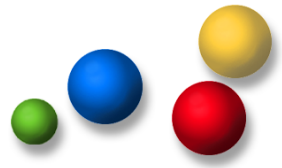
Marisa 4 days ago Mark as spam

Can you develop a "morph" that shows people what they might look like if they smoke two packs of cigaretts a day for 30 years. Or what they would look like if they lost 100 pounds or gained a hundred pounds.

I think my son and daughter would have fun using this tool the way it is - between the two of them. I could pick out the middle morphed picture between them - and they could see what another sibling of theirs might have had the potential to look like.

CMU-PIE Dataset

- 68 people, 13 poses, 43 illuminations, 4 expressions
- 35,247 faces detected by a face detector
- Classification and clustering on poses



Clustering

- **K-means clustering after transformation (k = 100)**
  - **K fixed to be the same as number of classes**
- **Two metrics**

  Purity - points within a cluster come from the same class

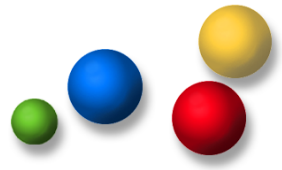
  Accuracy - points from a class form a single cluster

Methods	Purity (%)	Accuracy (%)
PCA	54.6 ($\pm$1.3)	46.8 ($\pm$1.3)
Nyström Isomap	59.9 ($\pm$1.5)	53.7 ($\pm$4.4)
Col-Sampling Isomap	56.5 ($\pm$0.7)	49.4 ($\pm$3.8)
Laplacian Eigenmap	39.3 ($\pm$4.9)	74.7 ($\pm$5.1)

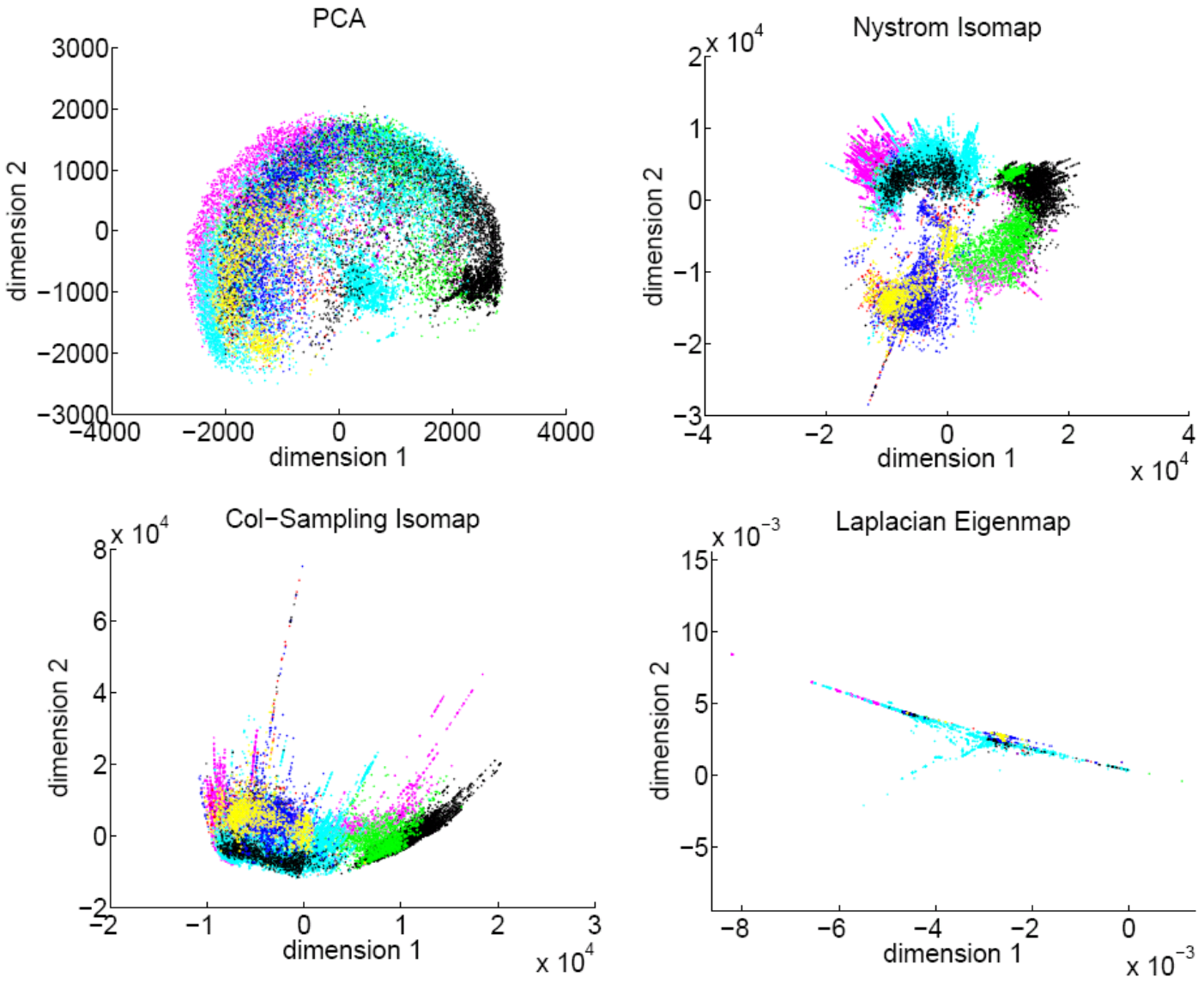
Matrix G is not guaranteed to be positive semi-definite in Isomap !

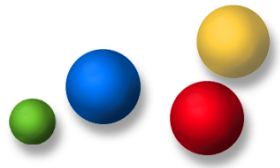
- Nystrom: EVD of W (can ignore negative eigenvalues)

- Col-sampling: SVD of C (signs are lost) !



Optimal 2D embeddings





Laplacian Eigenmaps

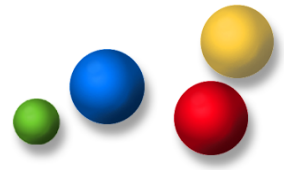
[Belkin & Niyogi, '01]

Minimize weighted distances between neighbors

- **Find t nearest neighbors for each image : $O(n^2)$**
- **Compute weight matrix W:** $$W_{ij} = \begin{cases} \exp(-\|x_i - x_j\|^2 / \sigma^2) & \text{if } i \sim j \\ 0 & \text{otherwise} \end{cases}$$
- **Compute normalized laplacian** $G = I - D^{-1/2} W D^{-1/2}$

  where $D_{ii} = \sum_j W_{ij}$
- **Optimal k reduced dims:** U_k

  ↙ **Bottom eigenvectors of G** $O(n^3)$



Different Sampling Procedures