Recommender systems and Structure from Motion
or
Neat tricks with SVD's

D.A. Forsyth

SVD in information retrieval

- Recall: D is word by document table
- Take an SVD of D to get $\mathcal{D} = \mathcal{U}\Sigma\mathcal{V}^T$
  - cols of U are an orthogonal basis for the cols of D
  - cols of V are an orthogonal basis for the rows of D (notice V^T!)
  - Sigma is diagonal; sort the diagonal to get largest at top
  - Notice
    - cols of U span word count vectors (cols of D)
    - cols of U corresponding to big singular values are common types of word count
    - cols of U corresponding to small singular values are uncommon types of word count

The SVD

- Important notions:
  - there are good algorithms (efficient, accurate, etc.)
  - column rank of a matrix = number of linearly independent columns
  - row rank of a matrix = number of linearly independent rows

- Approximation:
  - start with $\mathcal{D} = \mathcal{U}\Sigma\mathcal{V}^T$
  - write $\sum_k$ for matrix obtained by taking $\sum$ and setting all but the k largest singular values to zero
  - the matrix $\mathcal{D}_k = \mathcal{U}\Sigma_k\mathcal{V}^T$ is the best approximation to D with col rank (row rank) k

The SVD

- Important trick:
  - assume we know that D should have rank k
  - we measure D; the measured D will generally have higher rank (noise)
  - Best estimate of D is then D_k from SVD, previous page

- Variant:
  - assume we know that D factors
    - into (tall+thin) x (short+fat)
    - with known dimensions, hence known rank
  - we measure D; the measured D will generally have higher rank (noise)
  - Best estimate of D is then D_k from SVD, previous page
    - get the factors from SVD

Latent Semantic Analysis - II

- (we used SVD to smooth word counts)
- Recall: SVD of D is $\mathcal{D} = \mathcal{U}\Sigma\mathcal{V}^T$
- Strategy for smoothing word counts:
  - take word count vector c
  - expand on some of U's cols corresponding to large singular values
  - yields new, smoothed count vector
    - eg if many "elephant" documents contain "pachyderm", then smoothed "pachyderm" count will be non-zero for all elephant documents.
- Obtain a smoothed word document matrix hat(D) like this

Write $\mathcal{U}_k$ for the matrix consisting of the first k columns of $\mathcal{U}$, $\mathcal{V}_k$ for the matrix consisting of the first k columns of $\mathcal{V}$, Σ_k for Σ with all but the k largest singular values set to be zero, and write $\hat{\mathcal{D}} = \mathcal{U}_k \Sigma_k \mathcal{V}_k^T$.

Recommender systems: SVD as clustering

- Assume we have a (movie x viewer) table of scores
  - large number=liked it
  - small number = didn’t like
  - for the moment, assume all entries are known
- Viewer model
  - there are “types” of viewer
    - i.e. columns tend to be repeated
    - eg likes horror films vs likes romances
  - viewers could be a “mixture” of types
    - eg likes scary romances (?)
  - suggests that column rank may be low

Recommender systems - II

- Assume we have a (movie x viewer) table of scores
  - large number=liked it
  - small number = didn't like
  - for the moment, assume all entries are known
- Movie model
  - there are "types" of movie
    - i.e. rows tend to be repeated
    - eg appeals to people who like horror films vs appeals to people who like romances
  - viewers could be a "mixture" of types
    - eg appeals to people who like scary romances (?)
  - suggests that row rank may be low
    - (which is good, cause it should be the same as column rank)

Recommender systems: - III

- If we knew all the entries, and the rank, SVD yields
  - viewer types (or, at least, a basis)
  - movie types (or, at least, a basis)
- don't know rank - > search
- BUT
  - we don't know all the entries
  - NETFLIX prize - predict the missing entries in this table
    - because that allows you to suggest movies to viewers

Factors and the SVD

- Assume the true matrix M has the property $\mathcal{M} = \mathcal{T}\mathcal{S}$
  - where T is (tall+thin), S is (short+fat)
  - inner dimension known, k, so rank of M is k
- We observe D
  - rank is usually much higher
  - SVD guarantees that D_k is the closest rank k matrix to D
- Factors
  - we can estimate S and T from SVD, but not uniquely

Recommender systems and Factors

- Write D for the data matrix, W for a mask matrix
  - W_ij=0 if that entry of D is unknown, =1 if it is known
- Strategy:
  - choose S, T to minimize

$$\sum_{i,j} W_{ij}(D_{ij} - \sum_k T_{ik} S_{kj})^2$$

  - now multiply these S, T - the result is the whole of D
    - i.e. holes are filled in
  - we expect this to work even if D has many holes in it because
    - there are few parameters in S, T

Recommender systems and Factors

- How to minimize? set the gradient to zero
- gradient with respect to T_uv is

$$2\sum_j W_{uj}(D_{uj} - \sum_k T_{uk}S_{kj})S_{vj}$$

- gradient with respect to S_uv is

$$2\sum_i W_{iv}(D_{iv} - \sum_k T_{ik}S_{kv})T_{iu}$$

Recommender systems and Factors

- We have two linear systems
  - one in S, one in T
  - can solve by matrix methods
    - reshape into vectors
    - write A(T) S=b, C(S) T=d for the systems
- Strategy:
  - chose S^(0), T^(0)
  - iterate
    - A(T^(n-1)) S^(n)=b
    - C(S^(n))T^(n)=d
  - possibly changing order
  - this tends to converge

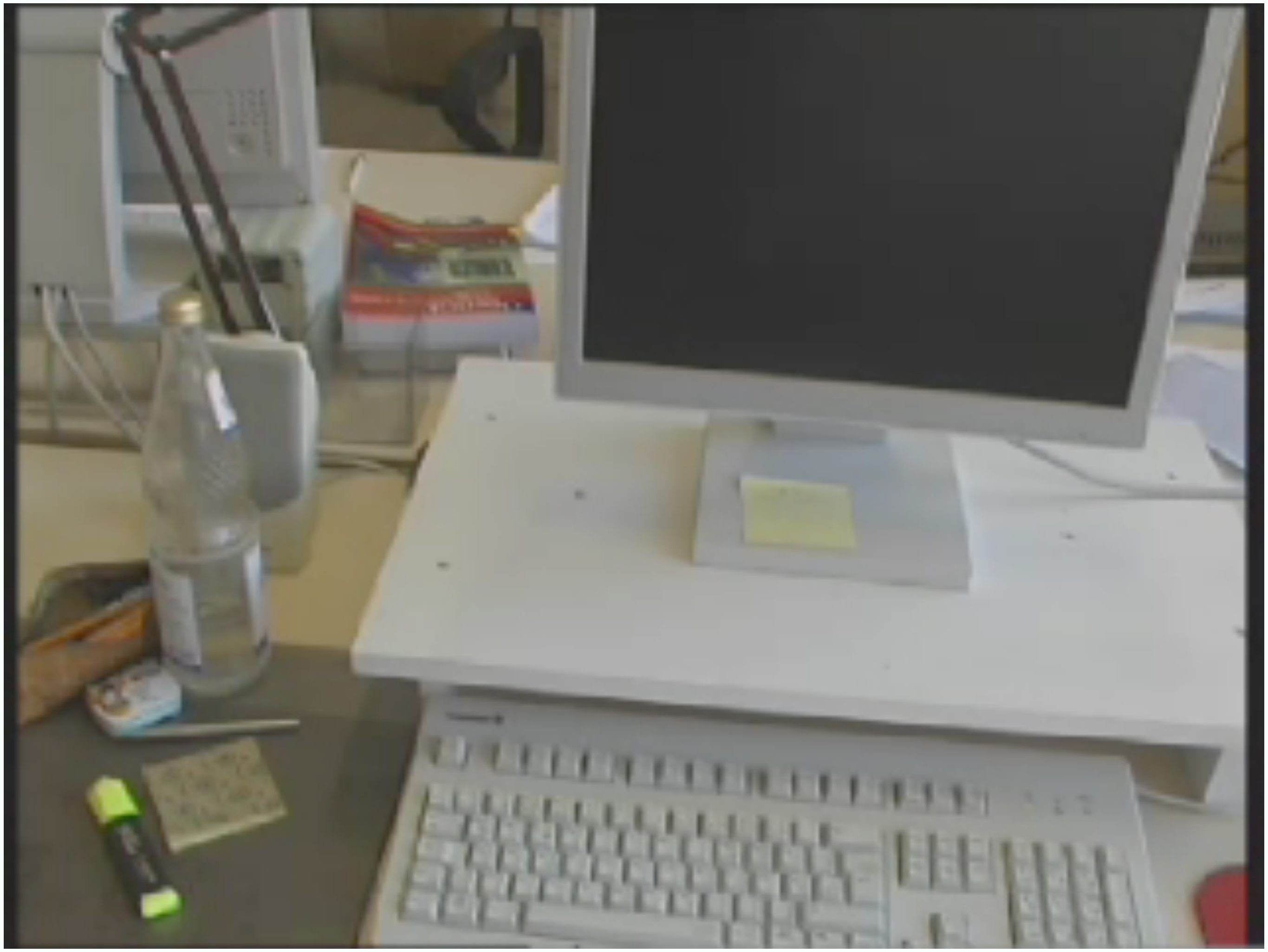
Camera and structure from motion

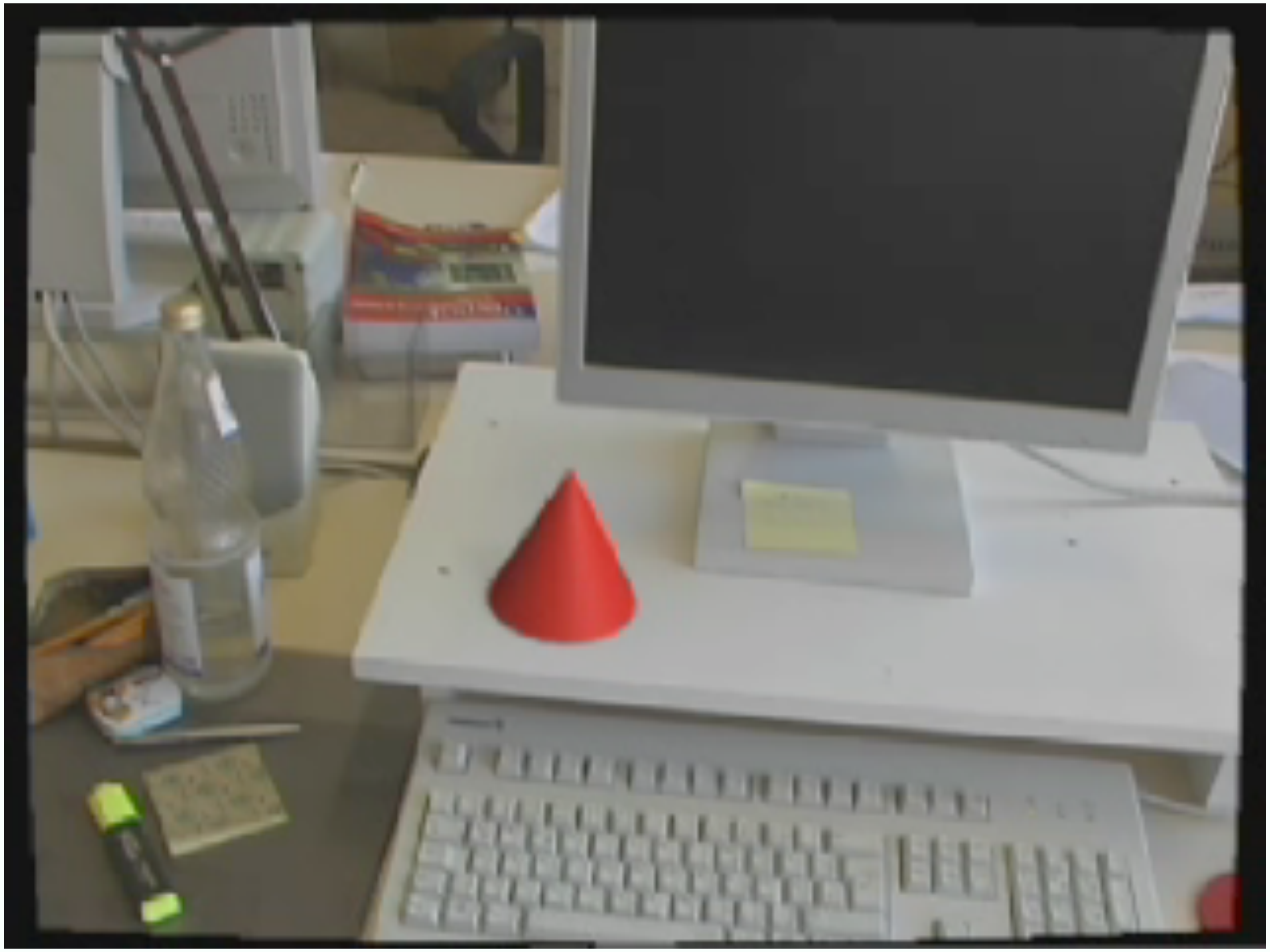
- Assume:
  - a moving camera views a static scene
  - the camera is orthographic (explanation coming)
- Can get:
  - the positions of all points in the scene
  - the configuration of each camera
- Applications
  - Reconstruction: Build a 3D model out of the reconstructed points
  - Mapping: Use the camera information to figure out where you went (robotics)
  - Object insertion: Render a 3D model using the cameras, then composite the videos

M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, R. Koch, Visual modeling with a hand-held camera, International Journal of Computer Vision 59(3), 207-232, 2004

Rendering and compositing

- Rendering:
  - take camera model, object model, lighting model, make a picture
  - very highly developed and well understood subject
  - many renderers available; tend to take a lot of skill to use (Luxrender)
- Compositing:
  - place two images on top of one another
  - new picture using some pixels from one, some from the other
  - example:
    - green screening
      - take non-green pixels from background, non-bg pixels from top





Orthographic cameras

- Standard model of a camera
  - Imagine a film plane on the x-y plane
  - Point (x, y, z) makes a mark at (s x, s y)
    - here s is a scale (eg pixels/meter)
- What about a camera in general position?
  - the camera film plane has
    - two axes, u and v
    - an origin, at (tx, ty)
  - they are at right angles
  - they are the same length
  - point in 3D is $(x, y, z) = \mathbf{x}$
  - equation:

$$\mathbf{x} \rightarrow (\mathbf{u} \cdot \mathbf{x} + t_x, \mathbf{v} \cdot \mathbf{x} + t_y)$$

Simplify

- Place the 3D origin at center of gravity of points
  - ie mean of x over all points is zero, mean of y is zero, mean of z is zero
- Camera origin at center of gravity of image points
  - we see all of them, so we can compute this
  - this is the projection of 3D center of gravity
- Now camera becomes

$$\mathbf{x} \to (\mathbf{u} \cdot \mathbf{x}, \mathbf{v} \cdot \mathbf{x})$$

- Index for points, views

$$\mathbf{x}_j \to (\mathbf{u}_i \cdot \mathbf{x}_j, \mathbf{v}_i \cdot \mathbf{x}_j)$$

Multiple views

- More notation:
  - write $x_{i,j}$ for the first (x) coordinate of the i'th picture of the j'th point
  - write $y_{i,j}$ for the second (y) coordinate of the i'th picture of the j'th point
- We had: $\mathbf{x}_j \to \left(\mathbf{u}_i \cdot \mathbf{x}_j, \mathbf{v}_i \cdot \mathbf{x}_j\right)$
- Rewrite:

$$\begin{pmatrix} x_{i,j} \\ y_{i,j} \end{pmatrix} = \begin{pmatrix} \mathbf{u}_i^T \\ \mathbf{v}_i^T \end{pmatrix} \mathbf{x}_j$$

Multiple views

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \dots & & & \\ y_{m,1} & y_{m,2} & \dots & y_{m,n} \\ y_{1,1} & y_{1,2} & \dots & y_{1,n} \\ y_{2,1} & y_{2,2} & \dots & y_{2,n} \\ \dots & & & \\ y_{m,1} & y_{m,2} & \dots & y_{m,n} \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \dots \\ \mathbf{u}_m^T \\ \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \dots \\ \mathbf{v}_m^T \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \end{pmatrix}$$

$$\mathcal{D} = \mathcal{V}\mathcal{X}$$

Data - observed!

Multiple views

- The data matrix has rank 3!
  - so we can factor it into an mx3 factor and a 3xn factor
  - (tall+thin)x(short+fat)
  - so we know what to do; SVD -> factors
- These factors are not unique
  - assume A is 3x3 with rank 3, we get symmetry below

$$\mathcal{D} = \mathcal{T}\mathcal{S} = (\mathcal{T}\mathcal{A})(\mathcal{A}^{-1}\mathcal{S})$$

Camera and reconstruction

- Can choose factors uniquely
  - recall v_i, u_i are
    - at right angles
    - same length
- Algorithm
  - form D
  - factor
  - now choose A so that v_i, u_i are at right angles, same length
    - by numerical optimization
- What if there are missing points?
  - no problems, dealt with this already

Software

- Look up the Voodoo camera tracker

  http://www.digilab.uni-hannover.de/docs/manual.html

Summary

- Getting (tall+thin)x(short+fat) factors of a matrix is easy
  - and quite accurate
  - can do it without knowing all the matrix
- Numerous problems take this form
- It's (rather loosely) a form of clustering

[redacted]