Large Scale 3D Reconstruction by Structure from Motion

Devin Guillory
Ziang Xie

CS 331B
7 October 2013

Overview

“*Rome wasn’t built in a day*”

- Overview of SfM
- Building Rome in a Day
- Building Rome on a *Cloudless* Day
  - Differences

Motivation

- Reconstruct cities using hundreds of thousands of online photos
  - Previous reconstructions relied on data from structured sources, e.g. aerial photographs
- Much larger models (1-2 orders of magnitude)
- Commercial applications
  - Photosynth, Google Maps Photo Tours

Example Result

http://grail.cs.washington.edu/rome/

Structure from Motion (SfM)

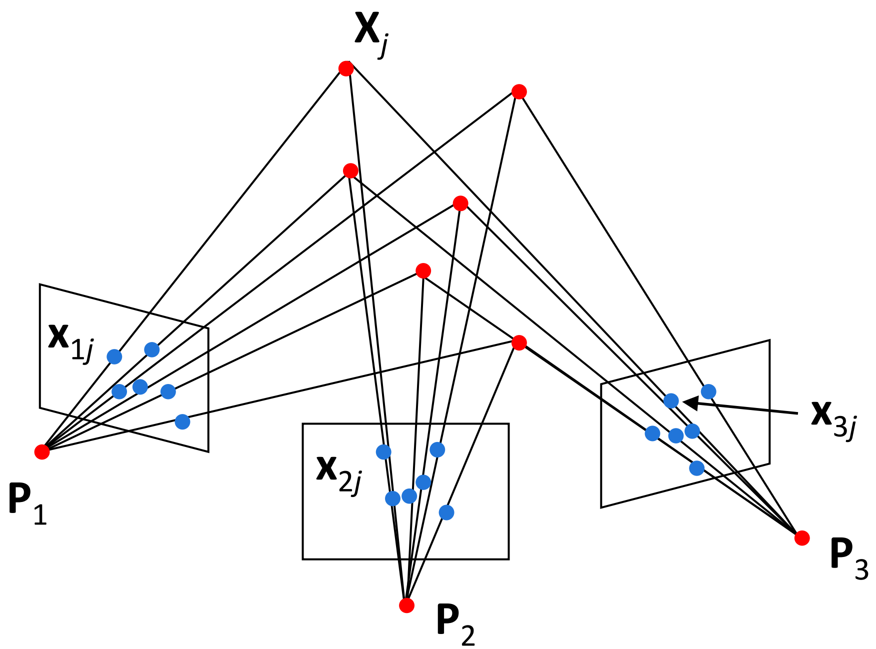
- General Problem Statement
  - Given a set of images with some overlap in views, infer the 3D geometry (structure) and camera poses/matrices (motion)
  - Usually rigid objects
  - Done using image correspondences (texture)
- Can solve approximately using SVD factorization
- Bundle adjustment to refine

SfM: Simplified Case

- Given: *m* images of *n* fixed 3D points

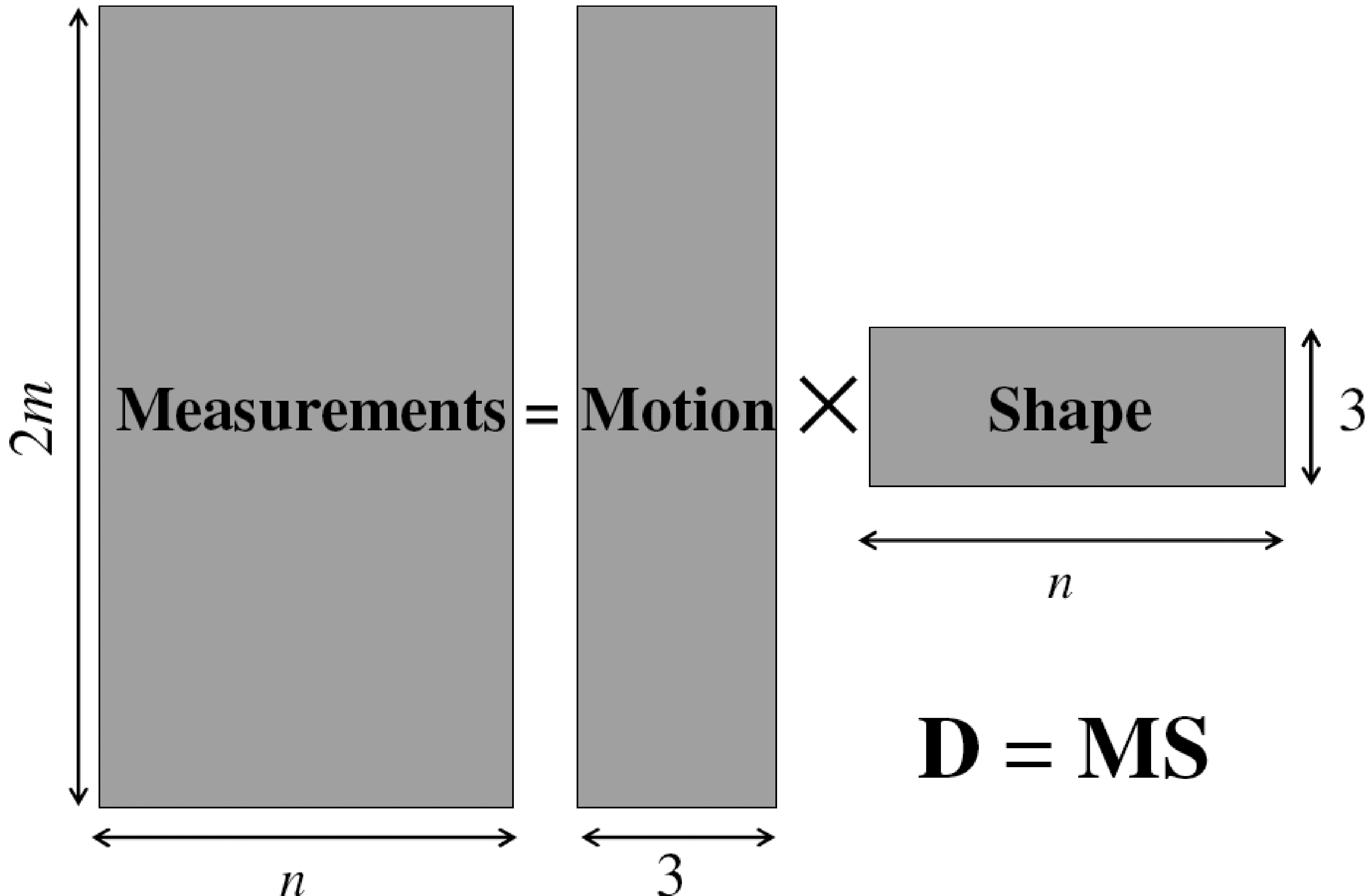
$$\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* correspondences $\mathbf{x}_{ij}$



Lazebnik

Factorizing the measurement matrix



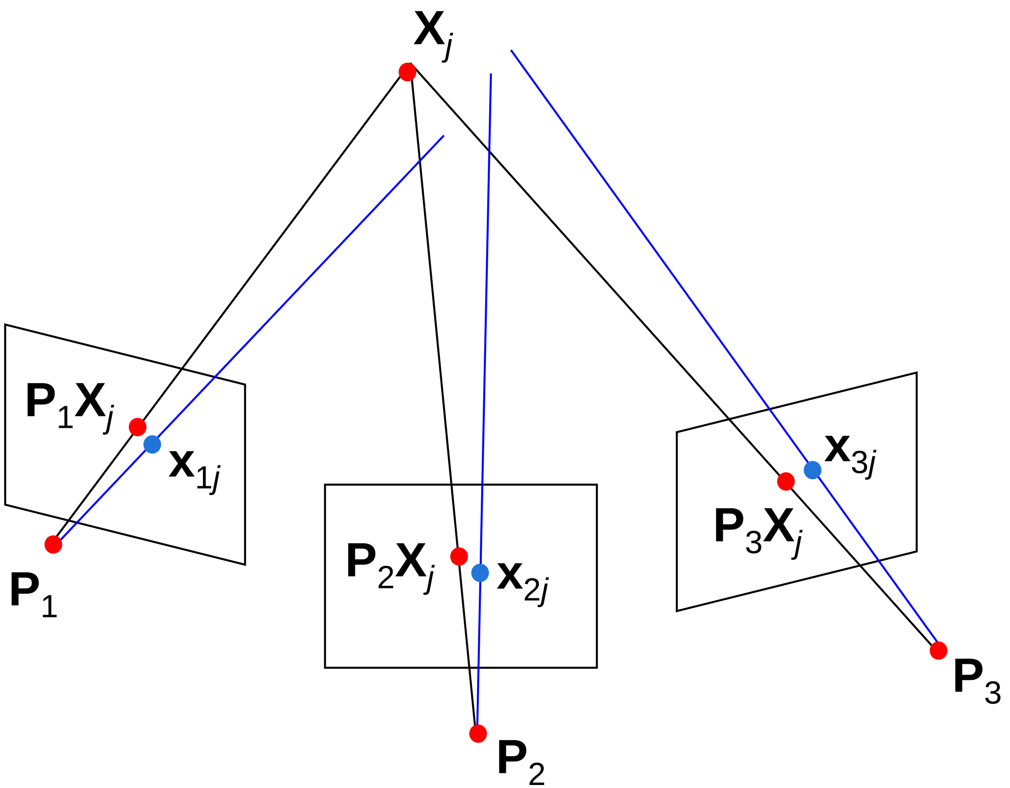
$$\mathbf{D} = \mathbf{MS}$$

Source: M. Hebert

Bundle adjustment

- Non-linear method for refining structure and motion
- Minimizing reprojection error

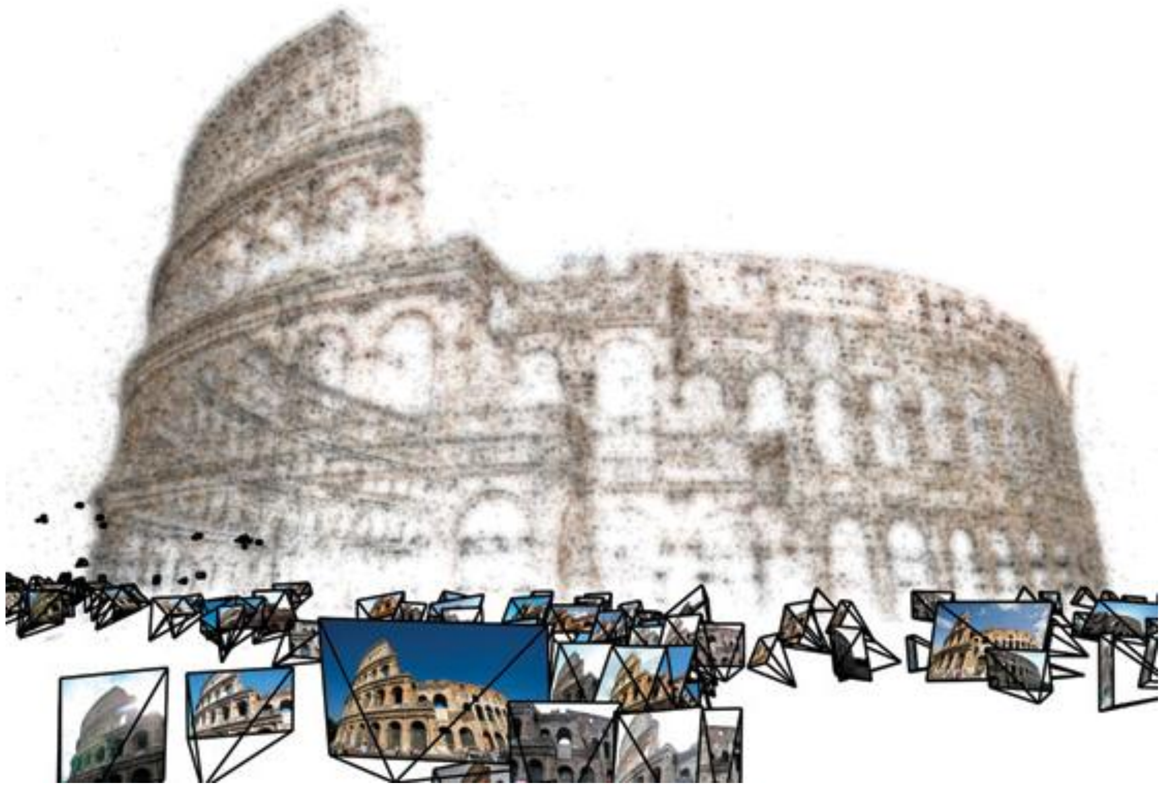
$$E(\mathbf{P}, \mathbf{X}) = \sum_{i=1}^{m} \sum_{j=1}^{n} D\left(\mathbf{x}_{ij}, \mathbf{P}_i \mathbf{X}_j\right)^2$$



Lazebnik

More Detailed Treatments of SfM

- http://www.cs.illinois.edu/~slazebni/spring13/lec18_sfm.pdf
- http://courses.cs.washington.edu/courses/cse576/08sp/lectures/Sfm.pdf
- Multiple View Geometry in Computer Vision, Richard Hartley and Andrew Zisserman, Cambridge University Press, 2004



Building Rome in a Day

S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. Seitz and R. Szeliski. Communications of the ACM, Vol. 54, No. 10, Pages 105-112, October 2011

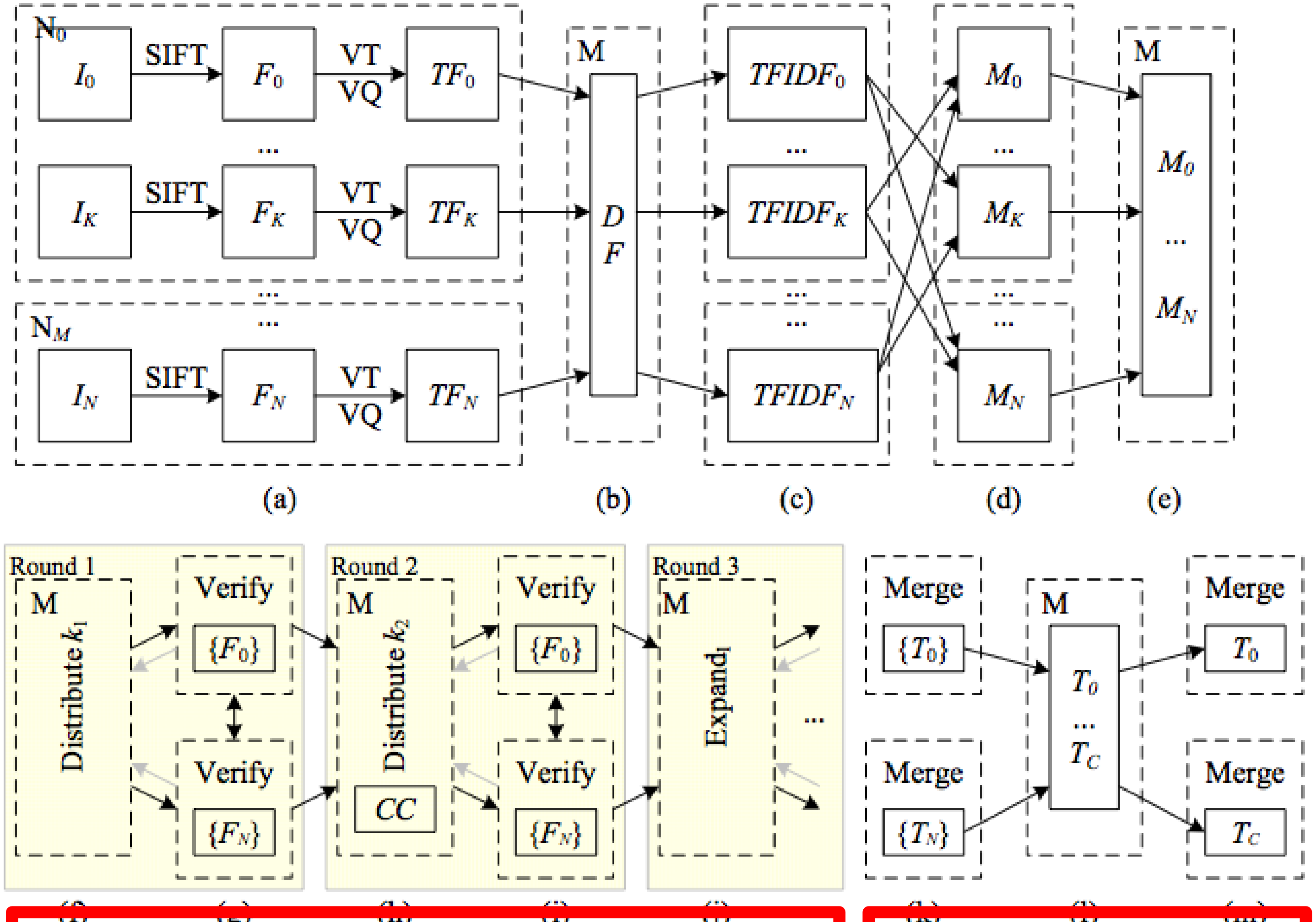
Outline

- Image preprocessing
- Matching / finding correspondences
- SfM
- System / implementation details throughout
  - Master node / worker node architecture
  - Main contributions of paper

System Overview

Preprocessing

Computing pairwise matches



Building the match graph

Merging points

Image Preprocessing

- Images on a central store, distributed to cluster nodes on demand (load balanced)
- On each node
  - For each image on the node
    - Extract focal length metadata (if available)
    - Downsample
    - Extract SIFT descriptors

Matching: Pairwise Case

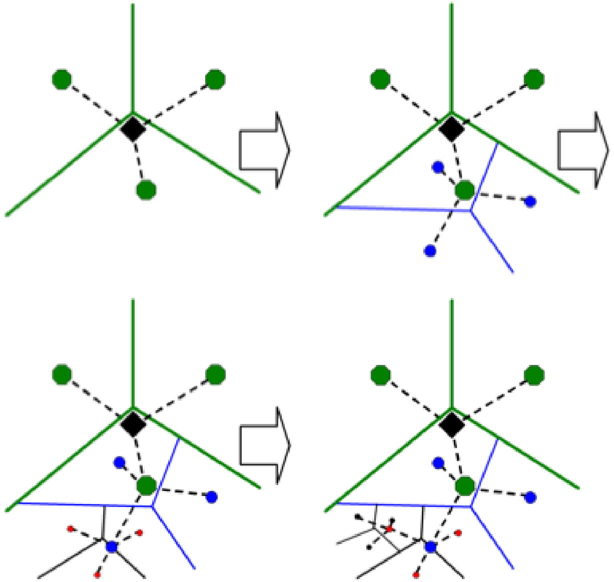
- Match SIFT features using approximate nearest neighbors library
- Features of one image put in k-d tree, features from other used as queries
  - Priority queue w/ 200 bin max
- For each query, take 2 nearest neighbors, if $d1/d2 < r \rightarrow$ accept (points correspond)

Matching: Optimized Scheme

- Pairwise matching too expensive (for 100K images, need to perform about 5 billion comparisons), and wasteful, since most images don’t match
- Heuristic based on whole image similarity to generate candidate image pairs
- Matching scheme alternates between proposal and verification steps

Matching: Vocabulary Tree Proposal (1)

- First proposal uses vocabulary trees as proposed by Nister and Stewenius
- images → descriptors → hierarchical k-means
- Tree defines quantization
  - Built offline with 20K images of Rome



Matching: Vocabulary Tree Proposal (2)

- Given quantizations of descriptors, then form bag-of-words representations of images
- Also use TF-IDF (term frequency inverse document frequency)

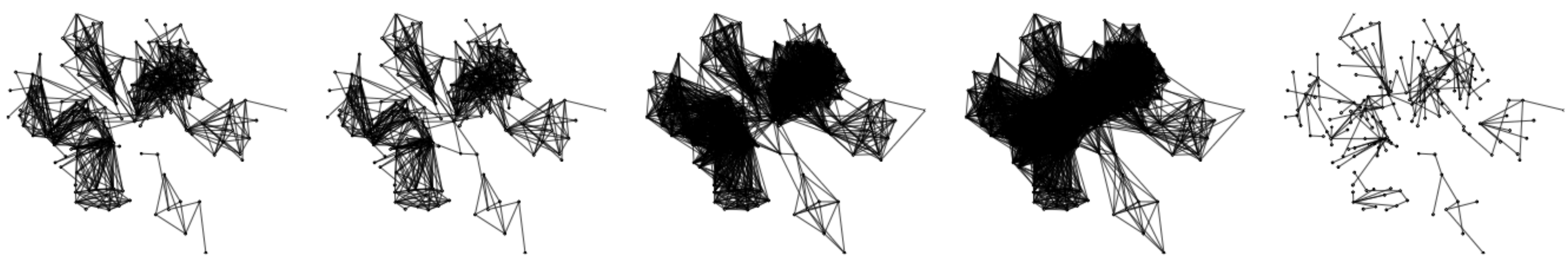
Matching: TF-IDF

Matching: Vocabulary Tree Proposal (2)

- Given quantizations of descriptors, then form bag-of-words representations of images
- Also use TF-IDF (term frequency inverse document frequency)
- TF vector computed at nodes for each image, and DF vector for each node broadcast at master
- Master then broadcasts combined DF for entire set of images

Matching: Vocabulary Tree Proposal (3)

- TF-IDF matrices broadcast across network so each node can compute dot product of TF-IDF vector of each of its images w/ all other images
- For each image, the first k_1 closest images are verified to create a sparsely connected match graph
- The next k_2 images are then used to join the connected components of the graph (consider those in intersection of separate components)



Initial Matches CC Merge Query Expansion 1 Query Expansion 4 Skeletal Set

Matching: Verification (1)

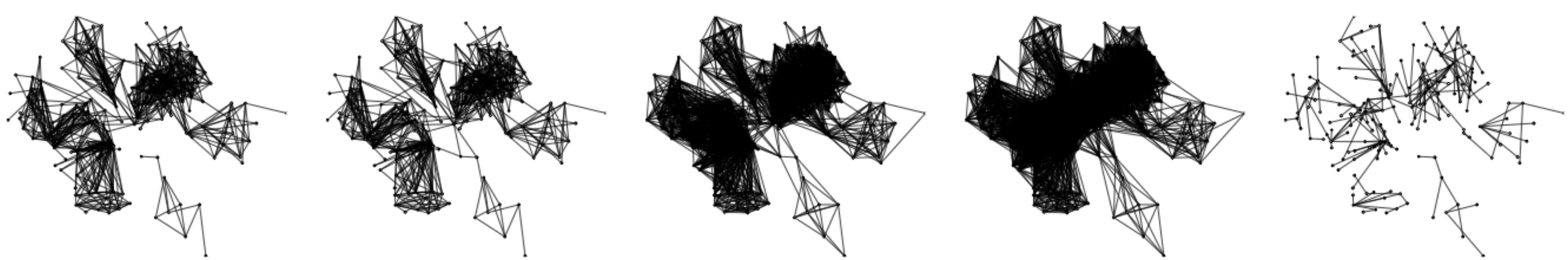
- To try and minimize network transfer of image features, use a greedy bin-packing algorithm
- When node asks for work (to perform verification), master node first chooses image-pairs with features on that node
- Then assigns image to node with most associated pairs until bin full

Matching: Verification (2)

- Verification then done in 3 steps
  - Match descriptors as previously described
  - Estimate essential/fundamental matrix
- If matrix estimation succeeds, sufficient overlap in views, and matches > threshold, do stereo reconstruction and store it

Match Graph: Query Expansion

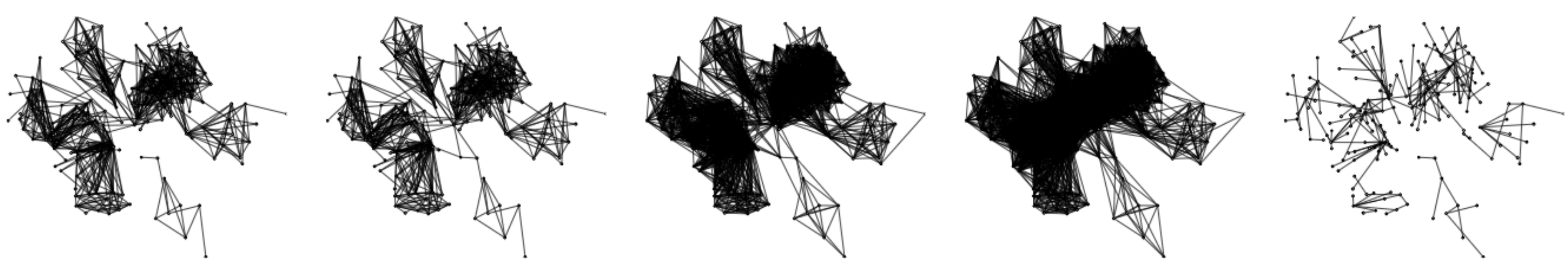
- Vocabulary tree gave proposal images for each image queried
- Now query using proposal images
- Finding all vertices within 2 steps of initial query vertex and add to match graph
- Repeat query expansion 4 times



Initial Matches   CC Merge   Query Expansion 1   Query Expansion 4   Skeletal Set

Match Graph: Skeletal Sets

- Want to first find and reconstruct minimal subset of photographs that capture basic connectivity and scene geometry
- Use method by Snavely et. al. (2008) to generate skeletal sets
  - Tries to find minimal set of images that spans reconstruction while bounding uncertainty between images
- Also gives significant speedup by removing redundancies



Initial Matches CC Merge Query Expansion 1 Query Expansion 4 Skeletal Set

Merging: Track Generation

- Up till now only considered image pairs
- Want to estimate 3D points and merge from more views
- Generate feature *tracks* for each CC
  - First generate tracks on each node
  - Passed to master node, which assigns each worker a CC to stitch tracks together for

SfM: Incremental Approach

- First done on CCs of skeletal set
- Uses incremental approach of Snavely et. al. (2006)
  - Pick pair w/ largest number of matches
  - Next add camera with largest number of tracks
  - Repeat 2nd step
- Run final bundle adjustment

Experiments

- Run on 62 node cluster, dual quad core processors, 32GB RAM, 1TB disk space (SSD?), 1GB/sec ethernet

Table 1. Matching and SfM statistics for the three cities.

						Time (h)		
Data set	**Images**	**Cores**	**Registered**	**Pairs verified**	**Pairs found**	**Matching**	**Skeletal sets**	**SfM**
Dubrovnik	57,845	352	11,868	2,658,264	498,982	5	1	16.5
Rome	150,000	496	36,658	8,825,256	2,712,301	13	1	7
Venice	250,000	496	47,925	35,465,029	6,119,207	27	21.5	16.5

Table 2. Reconstruction statistics for the largest connected components in the three data sets.

Data set	CC1	CC2	Skeletal set	Reconstructed
Dubrovnik	6,076	4,619	977	4,585
Rome	7,518	2,106	254	2,097
Venice	20,542	14,079	1,801	13,699

CC1 is the size of the largest connected component after matching, CC2 is the size of the largest component after skeletal sets. The last column lists the number of images in the final reconstruction.

More Results (1)

More Results (2)

Limitations

- Matching still takes significant amount of time
  - Depends a lot on initial distribution of images across nodes
- Track generation, skeletal sets, and SfM dominated by few largest components
- Skeletal sets helps a lot

Software

- SIFT++ (Now VLFeat)
  - http://vlfeat.org
- Multicore Bundle Adjustment (Newer)
  - http://grail.cs.washington.edu/projects/mcba/
- Bundler (SfM)
  - http://www.cs.cornell.edu/~snavely/bundler/

Building Rome on a Cloudless Day

J. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. "Building Rome on a Cloudless Day". ECCV 2010

Demo

Cloud vs Cloudless Rome

- Order of magnitude greater dataset
- Single Computer vs 62 “equivalent” nodes
- Comparable computation time
- GPU-Acceleration
- Varied Optimization Pipeline

Cloud vs Cloudless Rome

- SIFT Vocabulary tree vs GIST Features
- Iconic Images
- Skeletal Graph vs Local Iconic Scene Graph
- Geo-Tags

Pipeline

Clustering
- Appearance-Based (GIST)

Verification
- ARRSAC (SIFT)

Scene Reconstruction
- Local Iconic Image

Clustering

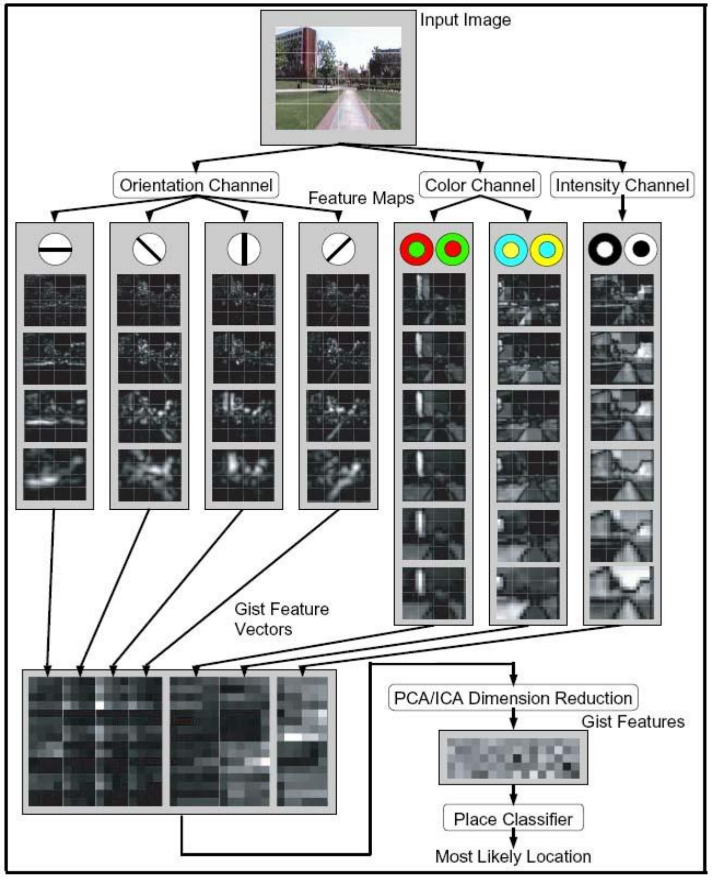
Appearance-Based Clustering Model

- GIST Descriptor + Subsampled RGB Image
- Locality Sensitive Binary Code
- K-medoids clustering by Hamming Distance

Clustering

GIST Feature Descriptor

- Similar to SIFT Descriptor, Global



Locality Sensitive Binary Code

Binary Coding size reduction

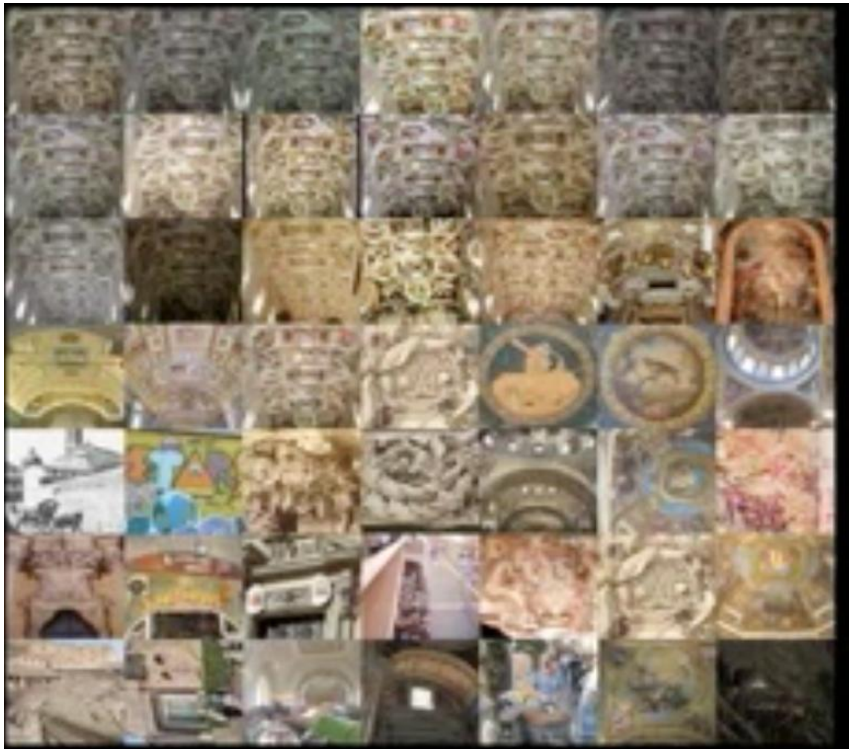
- 11,778 Bytes to 64 bytes

Enabled GPU Implemented Hamming distance/
K-mediods calculation

K-Medoids Clustering

- Similar to K-Means, uses datapoints instead of averages
- Utilizing Geo-tags to initialize clusters
- 100,000 clusters, $K = K_{geo} + K_{rand}$

Clusters



Geometric Verification

- SIFT Feature Extraction – GPU Implementation
- Putative Feature matches computed on GPU
- ARRSAC Verification of matches

RANdom Sample Consensus (RANSAC)

1. Select Random Inlier Hypotheses set model

2. Test model against other points

3. Keep set of inliers if greater than threshold

4. Re-estimate model, repeat 1

ARRSAC

- Optimized for real-time computation
- Partial depth-first search
- Improved initial hypotheses selection SPRT
- Checked by Geo-Tags

Iconic Image

- N top images closest to mediod determine Iconic Image
- Iconic image represents cluster, most inliers with n-1 top images
- Remaining cluster images verified with respect to top image

Iconic Image

Verified Clusters

- Checks for N verified images or discards cluster
- Discards images with less than M inliers from cluster
- N = 4, M = 18

Verified Cluster

Verified Cluster

Discards all images without at least N inliers

Scene Reconstruction

- Local Iconic Scene Graph Reconstruction
  - Geo-location Candidate Pairs of Iconic Images
  - KNN Candidate Pairs using GIST Features
- Geometric Verification
- 3D point Cloud Generation

Local Graph Initialization

- All iconics within *s* distance set as Candidate Pairs

- K-Nearest Neighbor iconics set as Candidate Pairs

Geometric Verification

- Candidates are verified according to previous step

- ARRSAC Verified Iconics are connected

- Stored a *local iconic scene graph,* -> distinct geographic site

Local Iconic Scene Graph

Colosseum Local iconic Scene Graph

Local Iconic Scene Graphs

Trevi Fountain

Incremental 3D Point Cloud Generation

- Per Local Iconic Scene Graph
- Choose highest inlier pair in local graph
- Obtain two-view metric reconstruction, using EXIF tags or estimates
- Iterate compute 3D sub-model

3D Point Cloud Generation

- Merge sub-models with 3D matches
- Use ARRSAC to transform sub-model merges

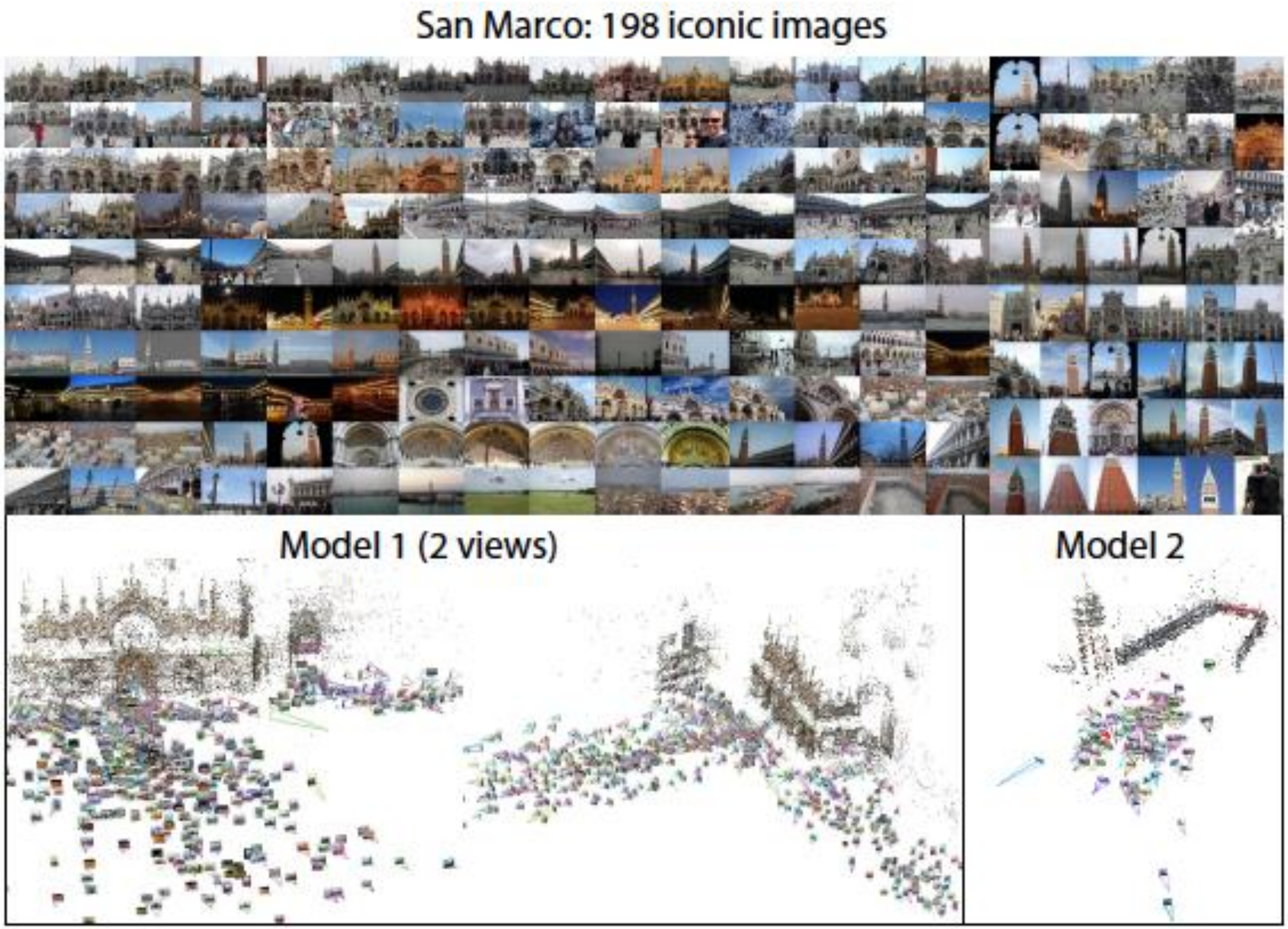
3D Point Cloud Generation

Iconic – Cluster Image matching

- Use 2D matches with Iconic to determine 3D correspondences on model

- ARRSAC determine camera pose non-iconics

3D Point Cloud Generation



3D Point Cloud Generation

Results

Dataset	Gist & Clustering	SIFT & Geom. verification	Local iconic scene graph	Dense	total time
Rome & geo	1:35 hrs	11:36 hrs	8:35 hrs	1:58 hrs	23:53 hrs
Berlin & geo	1:30 hrs	11:46 hrs	7:03 hrs	0:58 hrs	21:58 hrs
San Marco	0:03 hrs	0:24 hrs	0:32 hrs	0:07 hrs	1:06 hrs

Table 1. Computation times (hh:mm hrs) for the photo collection reconstruction for the Rome dataset using geo-tags, the Berlin dataset with geo-tags, and the San Marco dataset without geo-tags.

		LSBC	#images			
Dataset	**total**	clusters	**iconics**	**verified**	**3D models**	**largest model**
Rome & geo	2,884,653	100, 000	21,651	306788	63905	5671
Rome	2,884,653	100, 000	17874	249689	-	-
Berlin & geo	2,771,966	100, 000	14664	124317	31190	3158
San Marco	44, 229	4,429	890	13604	1488	721

Table 2. Image sizes for the the Rome dataset, the Berlin dataset, and the San Marco dataset.

Data set	CC1	CC2	Skeletal set	Reconstructed
Dubrovnik	6,076	4,619	977	4,585
Rome	7,518	2,106	254	2,097
Venice	20,542	14,079	1,801	13,699

Time (h)		
Matching	**Skeletal sets**	**SfM**
5	1	16.5
13	1	7
27	21.5	16.5

Another Example

Questions?

The proposed scheme

- Given: kernel K and desired code length n.
- Produce: a mapping

$$F^n : \mathbb{R}^D \to \{-1,+1\}^n, \qquad \text{where } F^n(\boldsymbol{x}) = (F_1(\boldsymbol{x}), \ldots, F_n(\boldsymbol{x}))$$

- Obtain each bit of the code by composing a random Fourier feature with a random sign mapping:

$$F_i(\boldsymbol{x}) = \operatorname{sgn}\left[\sqrt{2}\cos(\boldsymbol{\omega} \cdot_i \boldsymbol{x} + b_i) + t_i\right], \qquad i = 1, \ldots, n$$

where $\boldsymbol{\omega}_i \sim P_K$, $b_i \sim$ Unif$[0, 2\pi]$, $t_i \sim$ Unif$[-\sqrt{2}, \sqrt{2}]$,
$i = 1, \ldots, n$, are i.i.d.

Normalized Hamming Distance

- Approx: (1 – K(x,y)) / 2, Gaussian Kernel

- $K(x,y) = F_n(x) * F_n(y)$