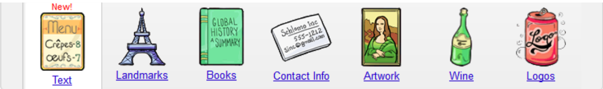
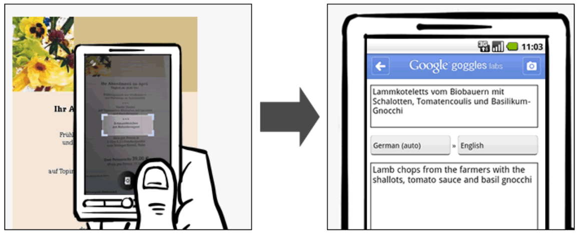
Image Search for Object Recognition and 3D Scene Reconstruction

Google Goggles



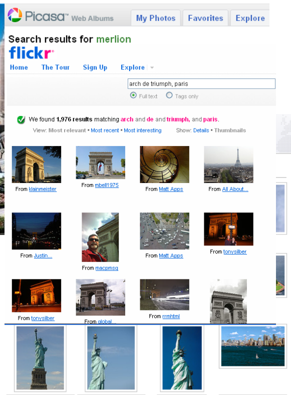


Lammkoteletts vom Biobauern mit Schalotten, Tomatencoulis und Basilikum-Gnocchi

German (auto) » English

Lamb chops from the farmers with the shallots, tomato sauce and basil gnocchi

Landmark Recognition



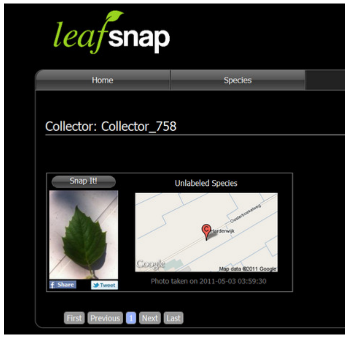
Search results for merlion

arch de triumph, paris

We found 1,976 results matching arch and de and triumph, and paris.

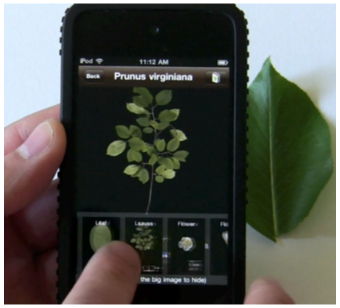
Leafsnap

- Leaf recognition

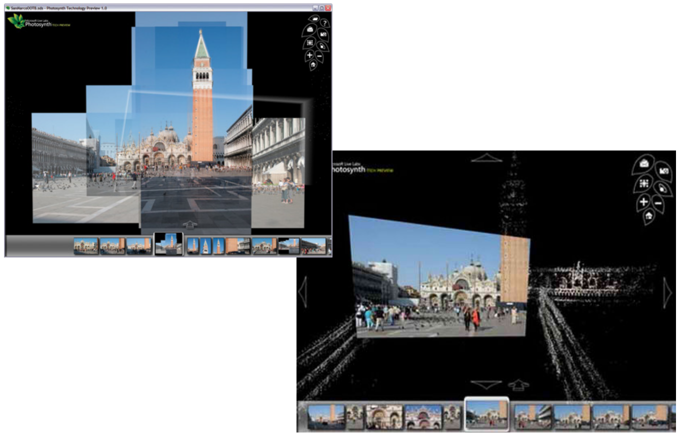


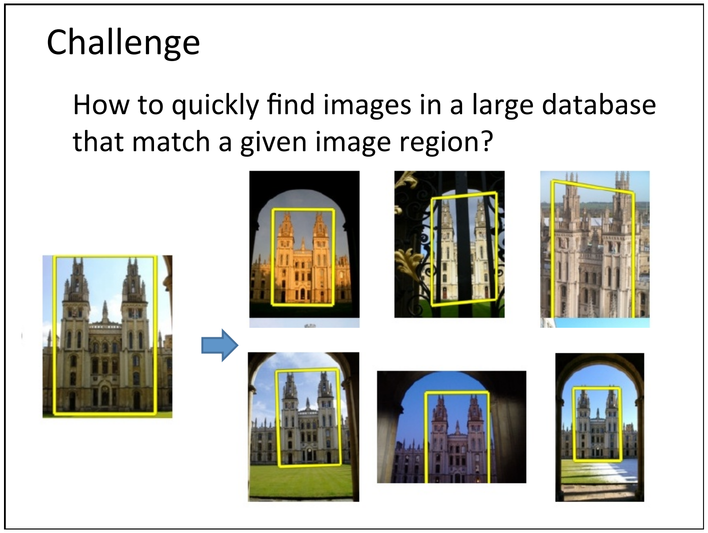
Collector: Collector_758

Photo taken on 2011-05-03 03:59:30



Microsoft Photosynth for 3D Reconstructdion





Challenge

How to quickly find images in a large database that match a given image region?

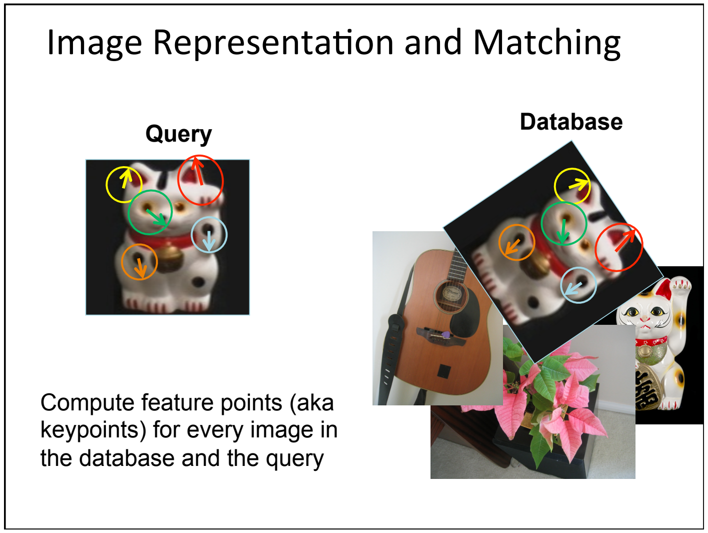


Image Representation and Matching

Query

Database

Compute feature points (aka keypoints) for every image in the database and the query

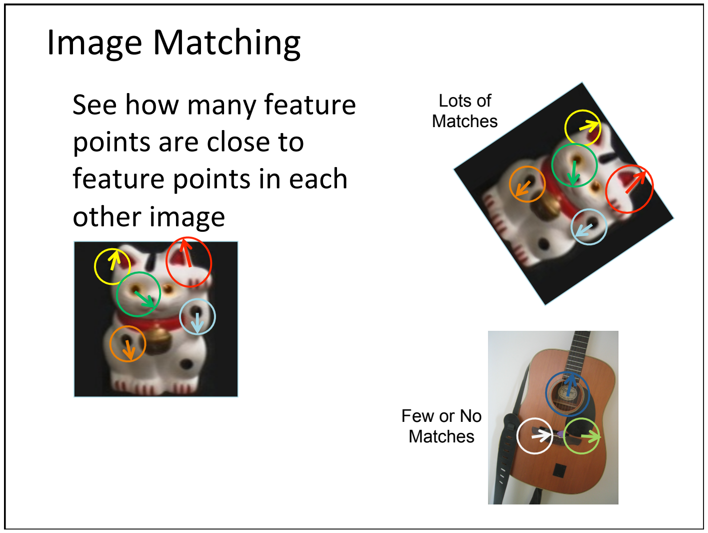
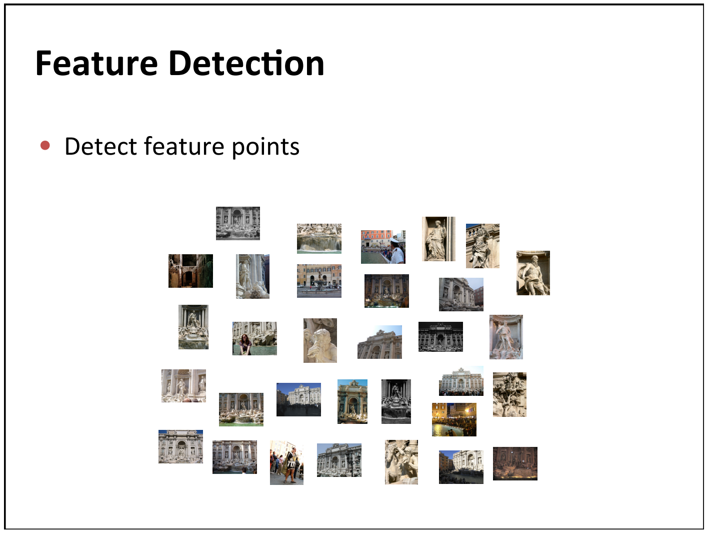


Image Matching

See how many feature points are close to feature points in each other image

Lots of Matches

Few or No Matches



Feature Detection

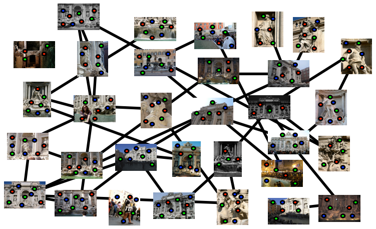
- Detect feature points

Feature Detection

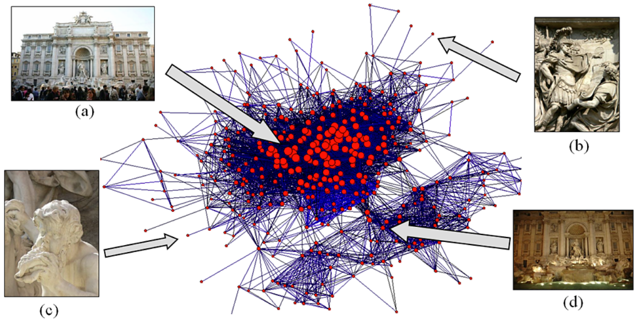
- Detect feature points

Pairwise Feature Matching

- Match features between each pair of images
  - Use a “descriptor” (i.e., a feature vector) for each feature point
  - For each feature point in image *I*, find 2 closest points in each other image, *J*, and accept if $d1/d2 < 0.6$

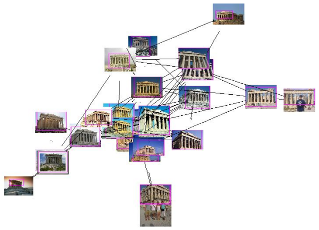


Construct Image Connectivity Graph

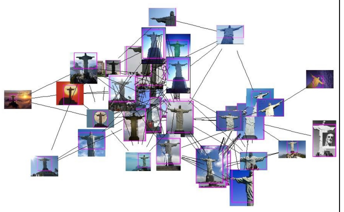


Each image is a node; an edge is present if there are enough matching feature points between an image pair

Visual Clusters



Acropolis, Athens, Greece



Corcovado, Rio de Janeiro, Brazil

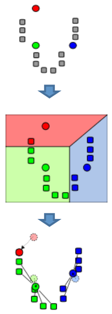
How to find Matches Fast?

Key idea 1: "Visual Words"

- Cluster the feature point descriptors

Key idea 1: "Visual Words"

K-Means Clustering algorithm



1. Randomly select K centers

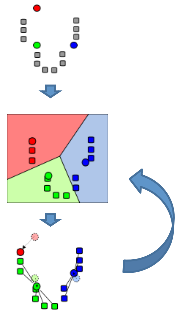
2. Assign each point to nearest center

3. Compute new center (mean) for each cluster

Illustration: http://en.wikipedia.org/wiki/K-means_clustering

Key idea 1: "Visual Words"

K-Means Clustering algorithm



1. Randomly select K centers

2. Assign each point to nearest center

3. Compute new center (mean) for each cluster

Back to 2

Illustration: http://en.wikipedia.org/wiki/K-means_clustering

Key idea 1: "Visual Words"

- Cluster the feature point descriptors
- Assign each descriptor to a cluster number
  - What does this buy us?
  - Each descriptor was 128 dimensional floating point, now is 1 integer (easy to match!)
  - Is there a catch?
    - Need **a lot** of clusters (e.g., 1 million) if we want points in the same cluster to be very similar
    - Points that really are similar might end up in different clusters

Key idea 1: "Visual Words"

- Cluster the feature point descriptors
- Assign each descriptor to a cluster number
- Represent an image region with a count of these "visual words"

Key idea 1: "Visual Words"

- Cluster the feature point descriptors
- Assign each descriptor to a cluster number
- Represent an image region with a count of these "visual words"
- An image is a good match if it has a lot of the same visual words as the query region

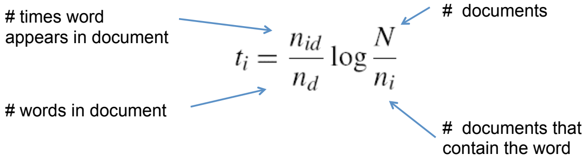
Naïve matching is still too slow

- Imagine matching 1,000,000 images, each with 1,000 feature points

Key Idea 2: Inverse Document File

- Like a book index: keep a list of all the words (feature points) and all the pages (images) that contain them
- Rank database images based on tf-idf measure

tf-idf: Term Frequency – Inverse Document Frequency



# times word appears in document

# words in document

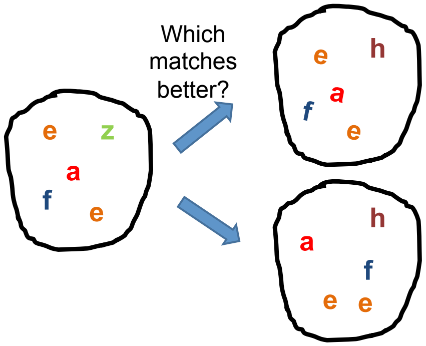
$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

# documents

# documents that contain the word

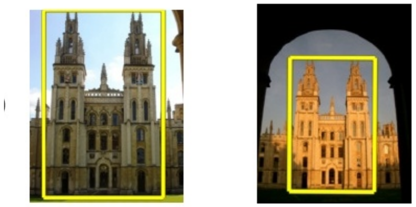
Can we be more accurate?

So far, we treat each image as containing a "bag of words", with no spatial information


Which matches better?

Can we be more accurate?

So far, we treat each image as containing a "bag of words", with no spatial information


Real objects have consistent geometry

Final key idea: geometric verification

- Goal: Given a set of possible keypoint matches, figure out which ones are geometrically consistent

How can we do this?

Final key idea: geometric verification

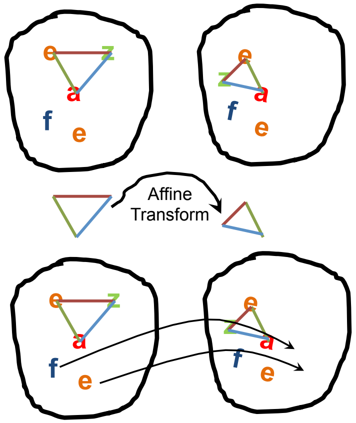
RANSAC for affine transform

Repeat N times:

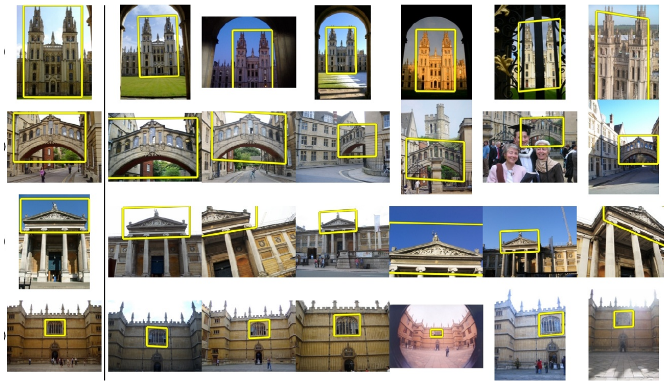
Randomly choose 3 matching pairs

Estimate transformation

Predict remaining points and count "inliers"



Application: Large-Scale Retrieval

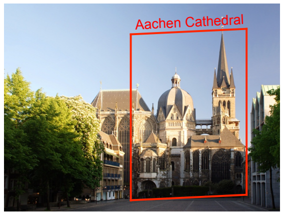


Query

Results on 5K (demo available for 100K)

[Philbin CVPR'07]

Example Applications



Mobile tourist guide

Self-localization
Object/building recognition
Photo/video augmentation

Video Google System

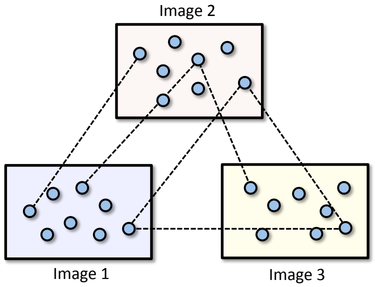
1. Collect all words within query region
2. Inverted file index to find relevant frames
3. Compare word counts
4. Spatial verification

[Sivic & Zisserman, ICCV 2003]

Demo online at
http://www.robots.ox.ac.uk/~vgg/research/vgoogle/index.html

From Pairwise Matches to Tracks

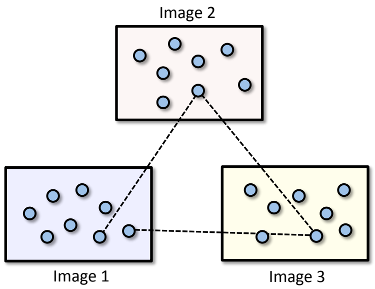
- Given pairwise matches, next step is to link up matches to form *"tracks"*



- Each track is a connected component of the pairwise feature match graph
- Each track will eventually grow up to become a 3D point

From Pairwise Matches to Tracks

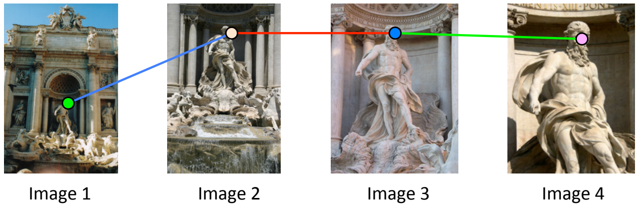
- Given pairwise matches, next step is to link up matches to form *"tracks"*



- Some tracks might be *inconsistent*
- We remove these features from the troublesome images

Correspondence Estimation

- Link up pairwise matches to form connected "tracks" of matching feature points across several images



The Power of Transitivity

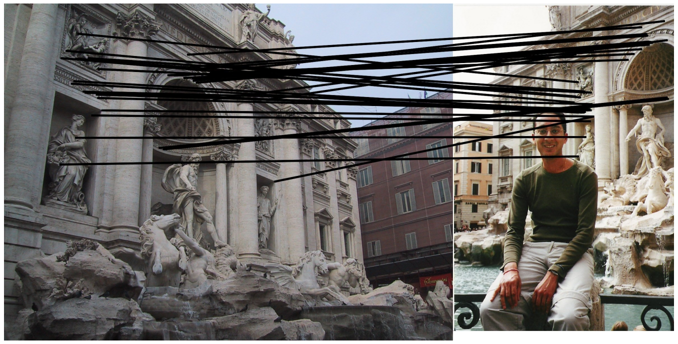
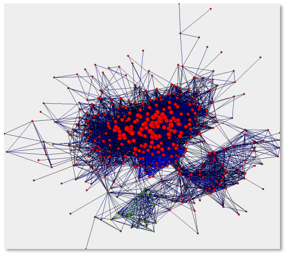


Image Connectivity Post-Track Generation



Raw image matches

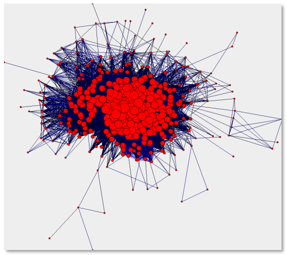
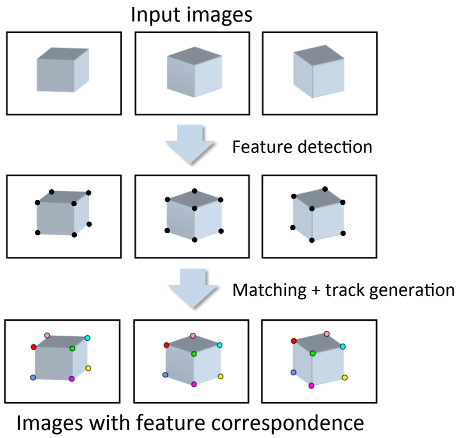
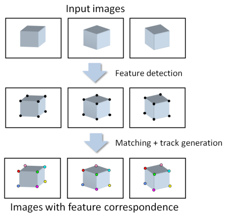


Image matches after track generation

The Story so far…


Input images
Feature detection
Matching + track generation
Images with feature correspondence

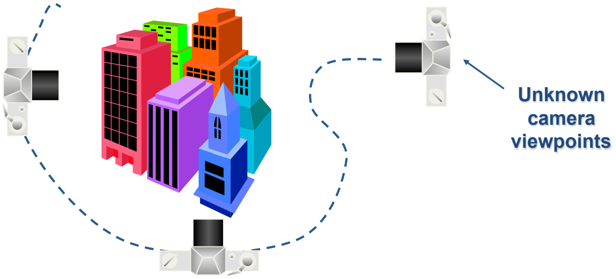
The Story so far…


Input images
Feature detection
Matching + track generation
Images with feature correspondence

Next step:

- Use "**structure from motion**" to solve for geometry (cameras and 3D points)

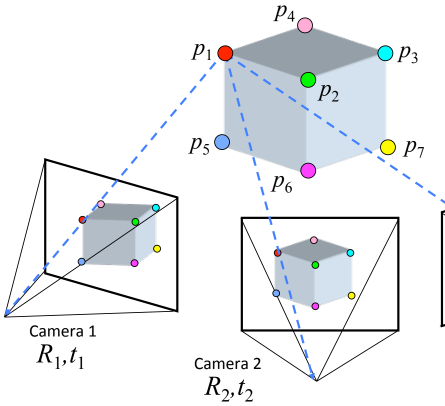
Structure from Motion (SfM)



Estimate

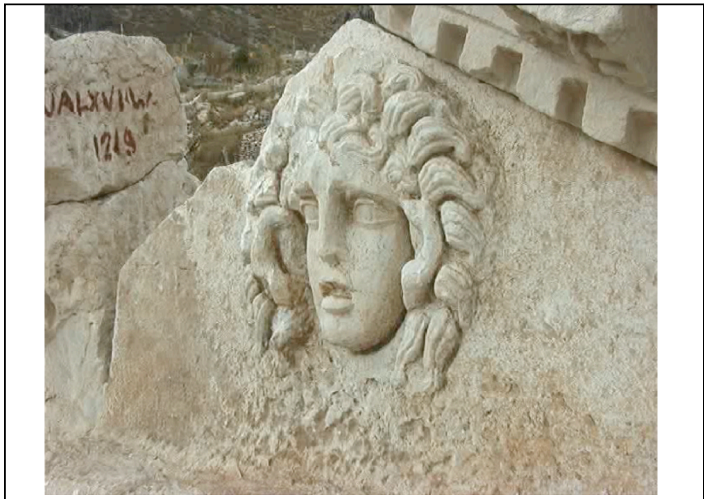
- Scene geometry (3D coordinates for each point)
- Camera extrinsic and intrinsic parameters (3D relative position and orientation, focal length, lens radial distortion)

Recover Structure from Motion



minimize

$$\mathrm{f}\,(\mathbf{R},\mathbf{T},\mathbf{P})$$



M. Pollefeys and L. Van Gool