CS664 Lecture #21: SIFT, object recognition, dynamic programming

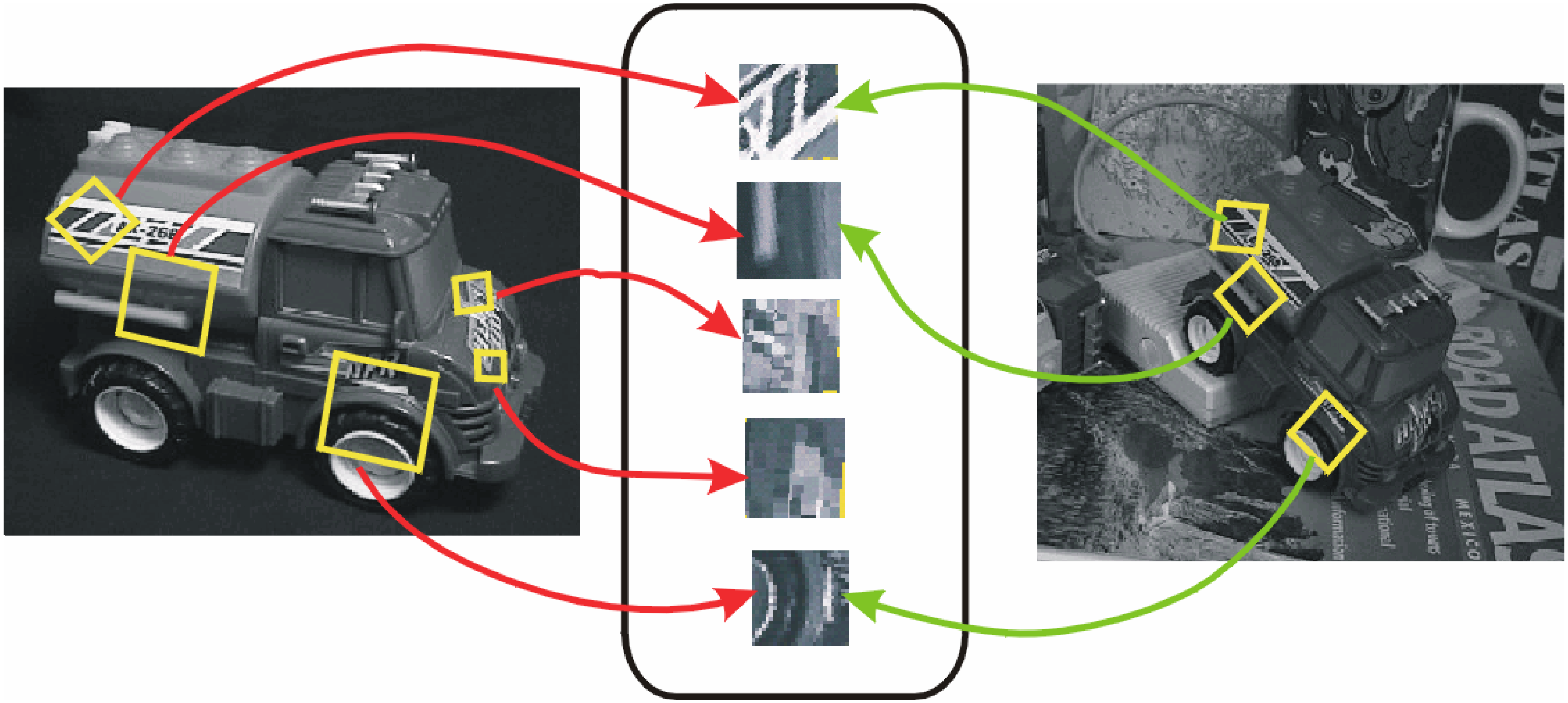
Some material taken from:

- **Sebastian Thrun, Stanford**
  http://cs223b.stanford.edu/
- **Yuri Boykov, Western Ontario**
- **David Lowe, UBC**
  http://www.cs.ubc.ca/~lowe/keypoints/

Announcements

- Paper report due on 11/15
- Next quiz Tuesday 11/15
  – coverage through next lecture
- PS#2 due today (November 8)
  – Code is due today, you can hand in the writeup without penalty until 11:59PM Thursday (November 10)
- There will be a (short) PS3, due on the last day of classes.

Invariant local features



SIFT Features

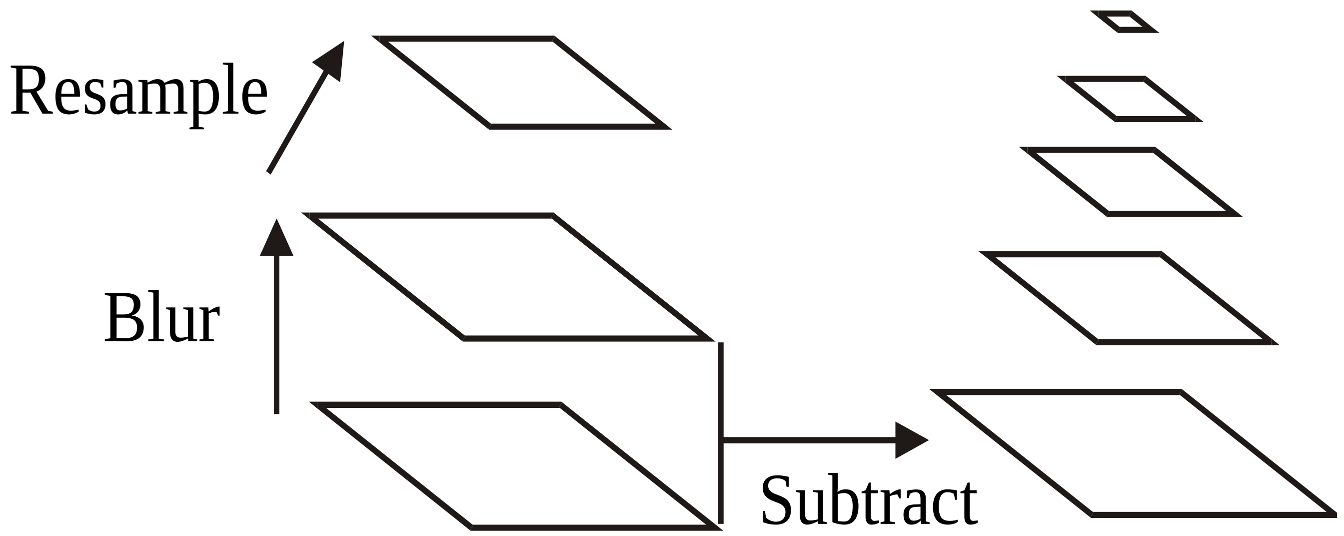
– Invariant to affine transformations, or changes in camera gain and bias

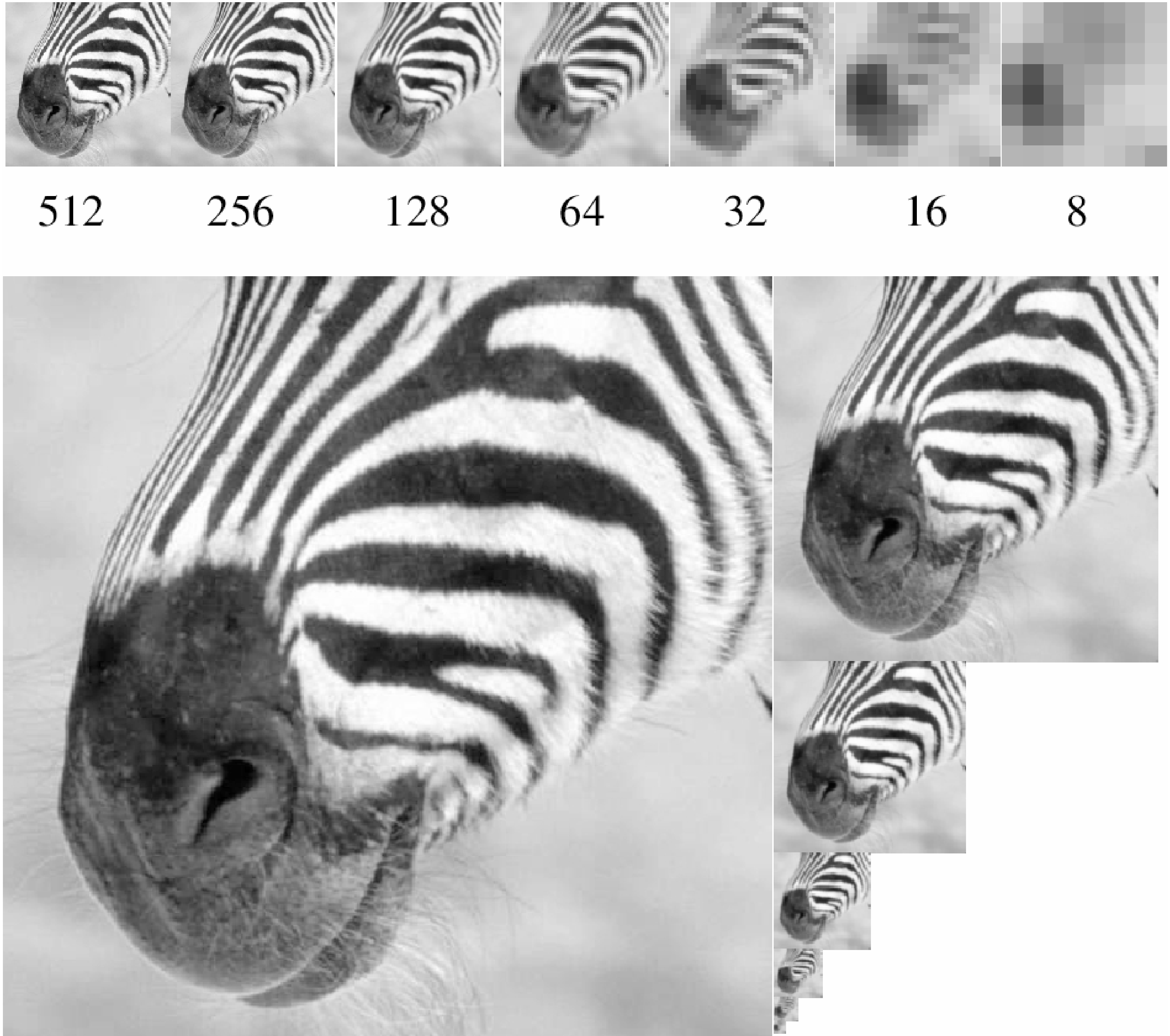
Keypoint detection

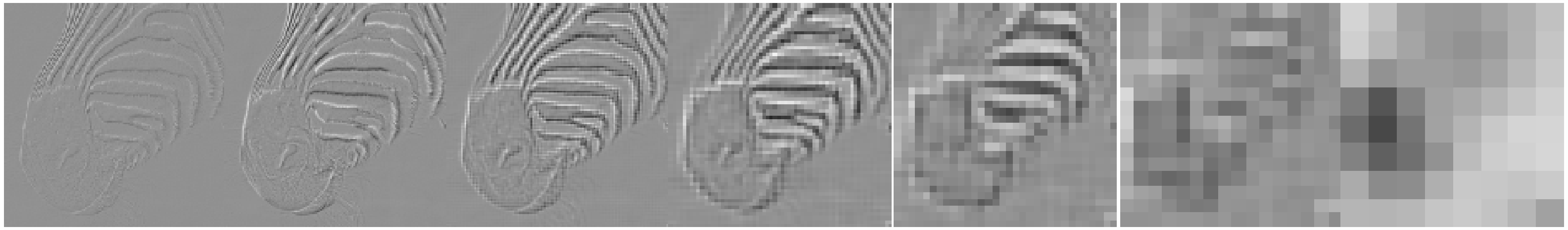
- Laplacian is a center-surround filter
  - Very high response at dark point surrounded by bright stuff
    - Very low response at the opposite
- In practice, often computed as difference of Gaussians (DOG) filter:
  - $(I \star h_{\sigma 1})-(I \star h_{\sigma 2})$, where $\sigma 1/\sigma 2$ is around 2
  - Scale parameter σ is important
- Keypoints are maxima (minima) of DOG that occur at multiple scales

Scale-space pyramid

- All scales must be examined to identify scale-invariant features
- DOG pyramid (Burt & Adelson, 1983)



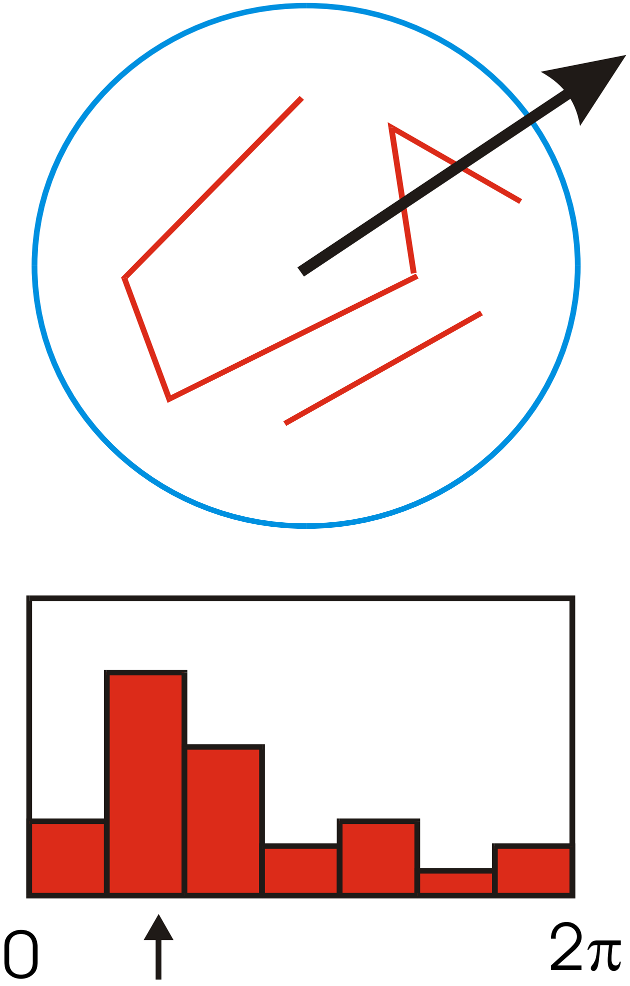

512
256
128
64
32
16
8



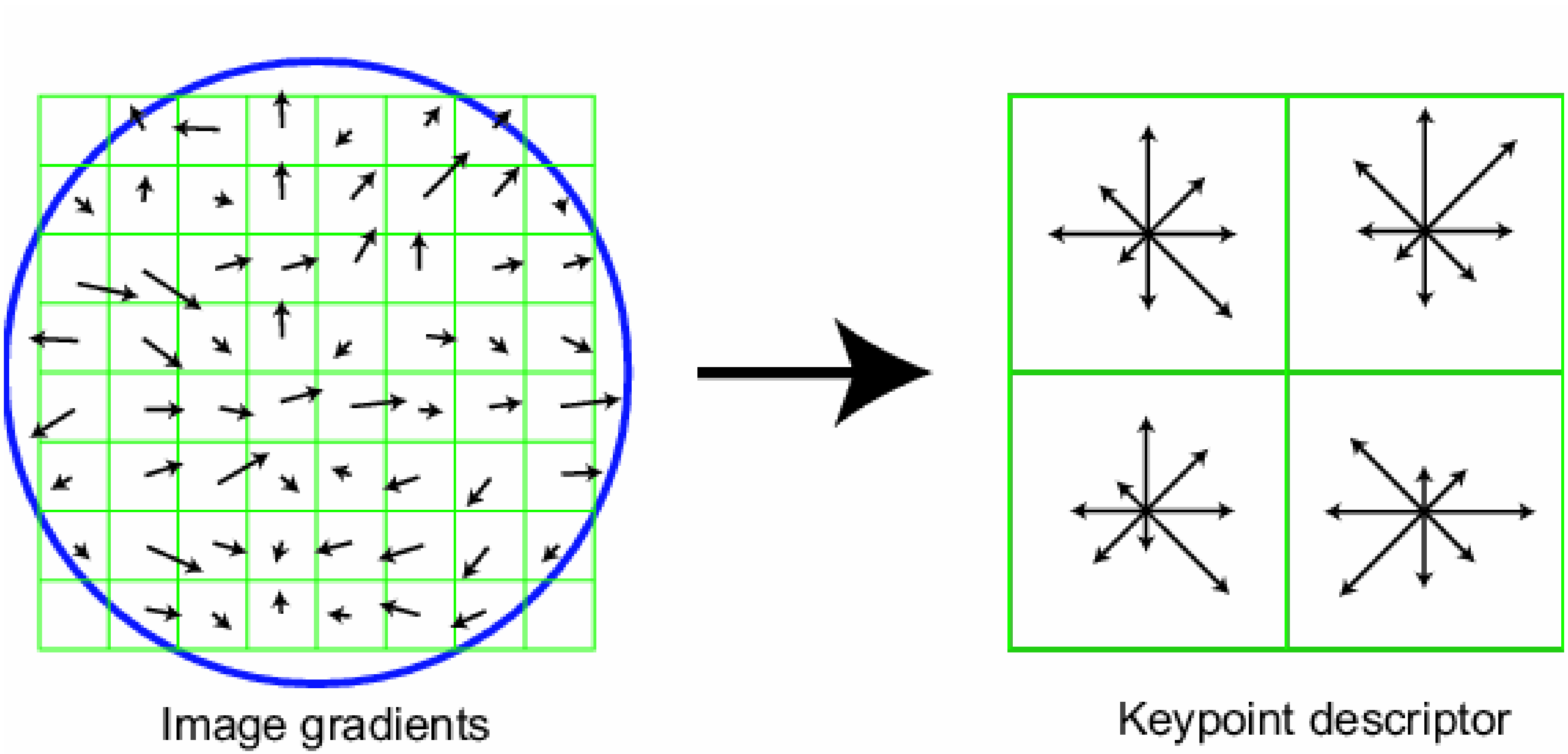
512 256 128 64 32 16 8

Rotation invariance

- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)



SIFT feature vector



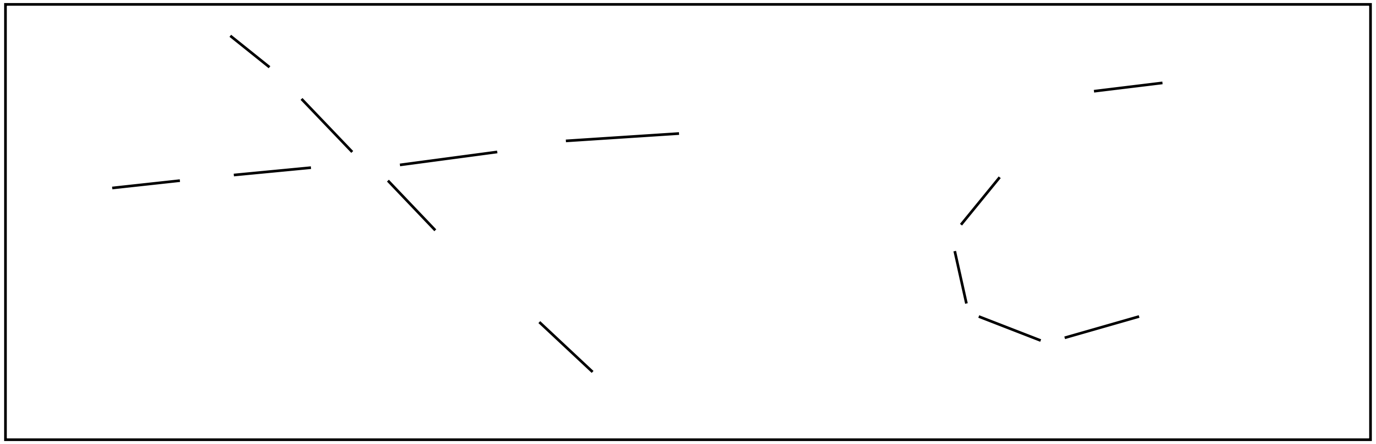
- Note: this is somewhat simplified; there are a number of somewhat *ad hoc* steps, but the whole thing works pretty well

Hough transform

- Motivation: find global features

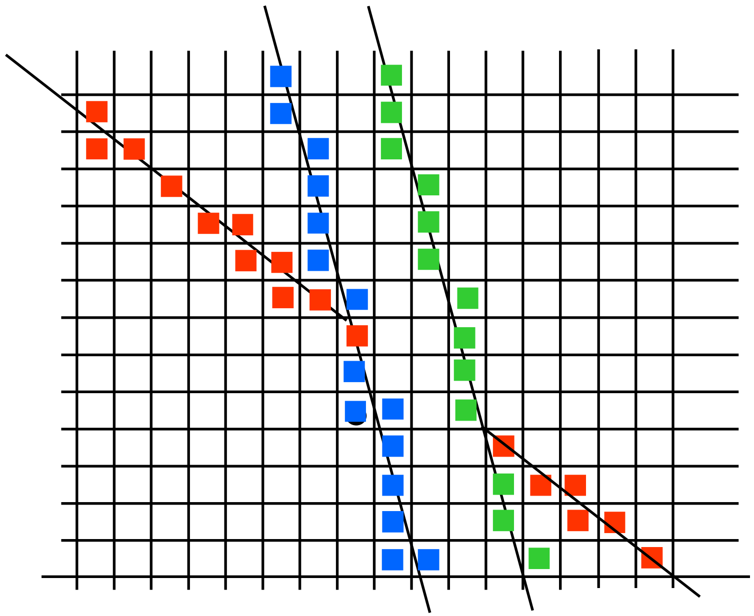
Example: vanishing points

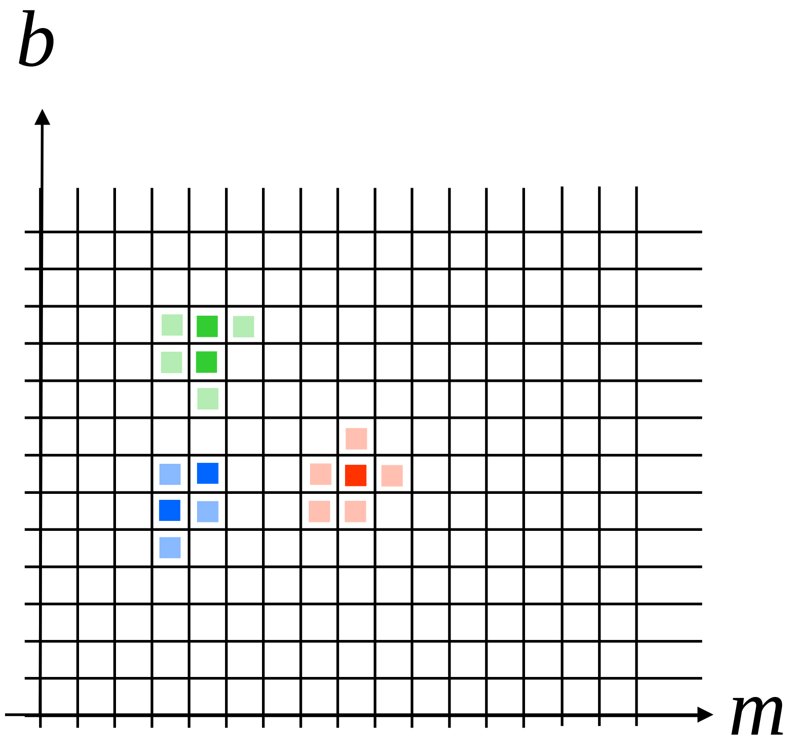
From edges to lines



- An edge should “vote” for all lines that go (roughly) through it
  - Find the line with lots of votes
  - A line is parameterized by m and b
    - This is actually a lousy choice, as it turns out

Hough transform for lines

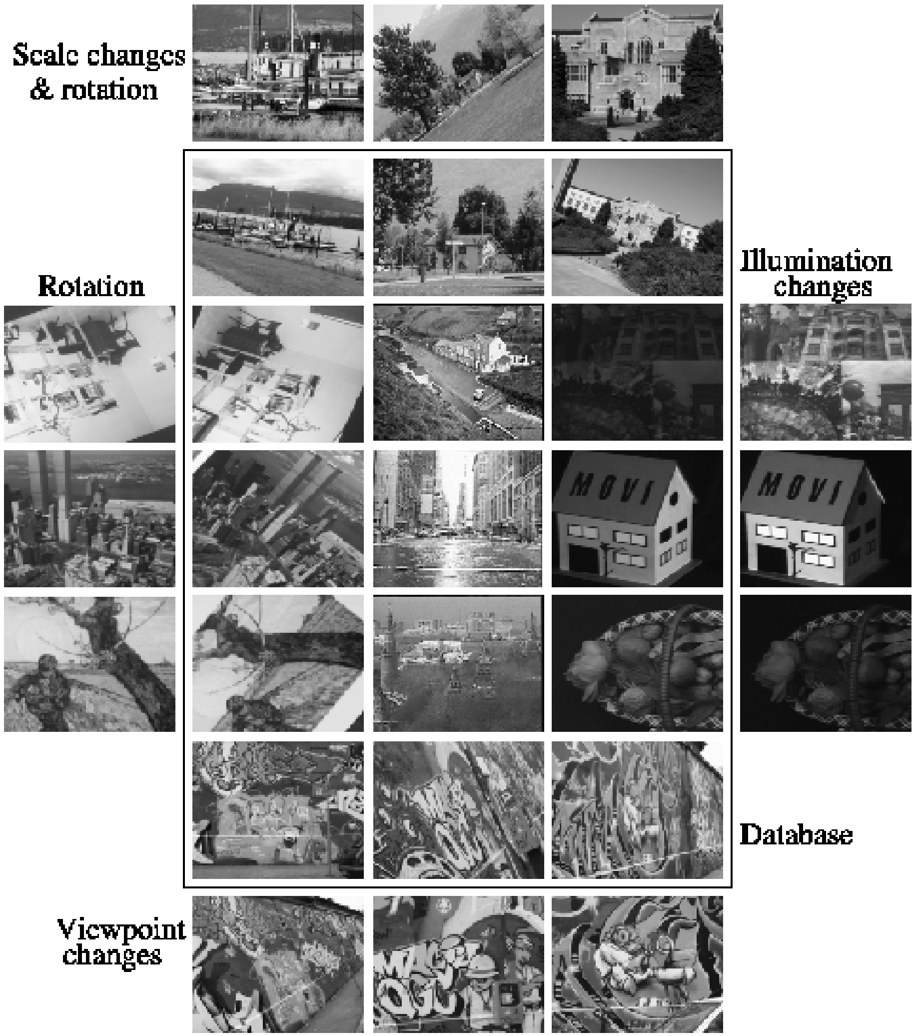




SIFT-based recognition

- Given: a database of features
  - Computed from model library
- We want to probe for the features we see in the image
- Use approximate nearest-neighbor scheme

SIFT is quite robust



SIFT DEMO!

Recognition

- Classical recognition (Roberts, 1962)
  - http://www.packet.cc/files/mach-per-3D-solids.html
    - Influenced by J. J. Gibson
- Given: set of objects of known fixed shape
- Find: position and pose (“placement”)
- Match model features to image features
- Models and/or image can be 2D or 3D
  - 2D to 2D example: OCR
  - Common case is 3D model, 2D image

Face recognition

- Extensively studied special case
- Approaches: intensities or features
  - Intensities: SSD (L_2 distance) or variants
  - Features: extract eyes, nose, chin, etc.
- Intensities seem to work more reliably
  - Images need to be registered
  - Famous application of PCA: eigenfaces
- Nothing really works with serious changes in lighting, profile, appearance
  - FERET database has good evaluation metrics

Combinatorial search

- Possible formulation of recognition: match each model feature to an image feature
  - Some model features can be occluded
- This leads to an intractable problem with lots of backtracking
  - “Interpretation tree” search
  - Especially bad with unreliable features
- The methods that work tend to avoid explicit search over matchings
  - Robust to feature unreliability

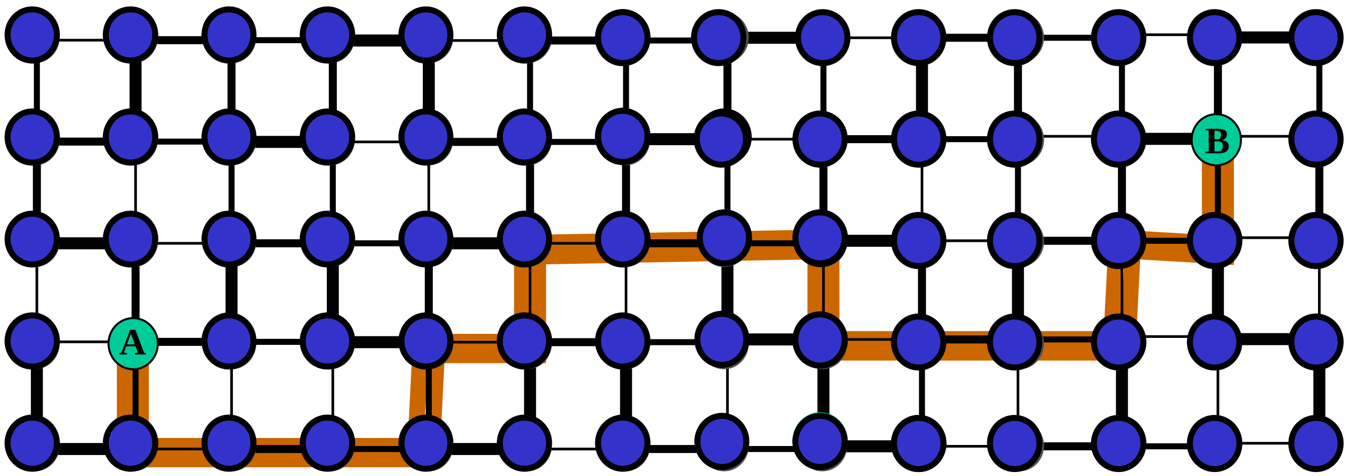
Distance-based matching

- Intuition: all points (features) in model should be close to some point in image
  - We will assume binary features, usually edges
  - All points assumption means no occlusions
  - Many image points will be unmatched
- Naively posed, this is very hard
  - For each point in the model, find the distance to the nearest point in the image
  - Do this for each placement of the model
  - How can we make this fast?

Dynamic programming

- General technique to speed up computations by re-using results
  - Many successful applications in vision
- Canonical examples:
  - Shortest paths (Dijkstra's algorithm)
    - Many applications in vision (curves)
  - Integral images
    - Efficiently compute the sum of any quantity over an arbitrary rectangle
    - Useful for image smoothing, stereo, face detection, etc.

Shortest paths via DP



- processed nodes (distance to A is known)
- active nodes (front)
- active node with the smallest distance value

Dijkstra's algorithm

Integral images via DP

- Suppose we want to compute the sum in D
  - At each pixel (x,y), compute the sum in the rectangle [(0,0),(x,y)]
  - Gives: A+C,A+B,A+B+C+D
  - (A+B+C+D) – (A+C) – (A+B) + A = D
  - Can compute rectangle sums by same trick
    - Row major scan