Designing Applications that See
Lecture 7: Object Recognition

Dan Maynes-Aminzade

29 January 2008

Reminders

- Pick up graded Assignment #1
- Assignment #2 released today
- Fill out the interim course evaluation
- CS247L workshop on Flash tomorrow night, 6-8PM in Wallenberg 332
- Next class: OpenCV tutorial
  - Bring your webcams
  - We will be using Windows with Visual Studio (you can try to follow along on your own machine, but you will need to figure out things like library and include paths on your system)

Today's Goals

- Understand the challenges of object recognition
- Learn about some algorithms that attempt to recognize particular objects (and object classes) from their basic constituent features

Outline

- Object recognition: general problem description
- Boosted tree classifiers
- SIFT algorithm

Fast, Accurate, General Recognition

(courtesy of G. Bradski)

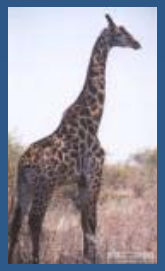
Quick, Spot the Mullets!

(courtesy of G. Bradski)

What Makes a Car a Car?

CARS

NOT CARS

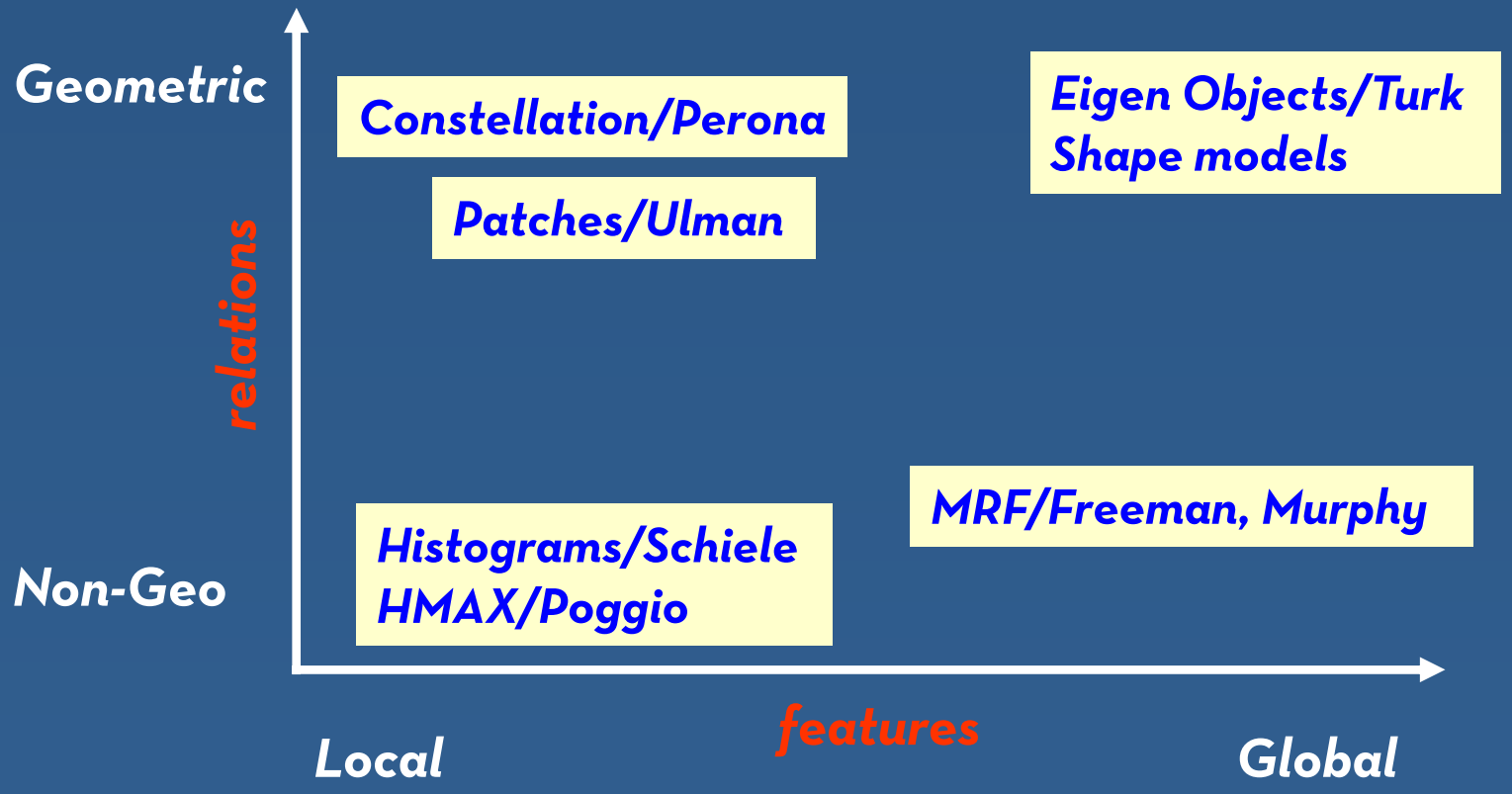


Object Detection

- Goal: find an object of a pre-defined class in a static image or video frame.
- Approach
  - Extract certain image features, such as edges, color regions, textures, contours, etc.

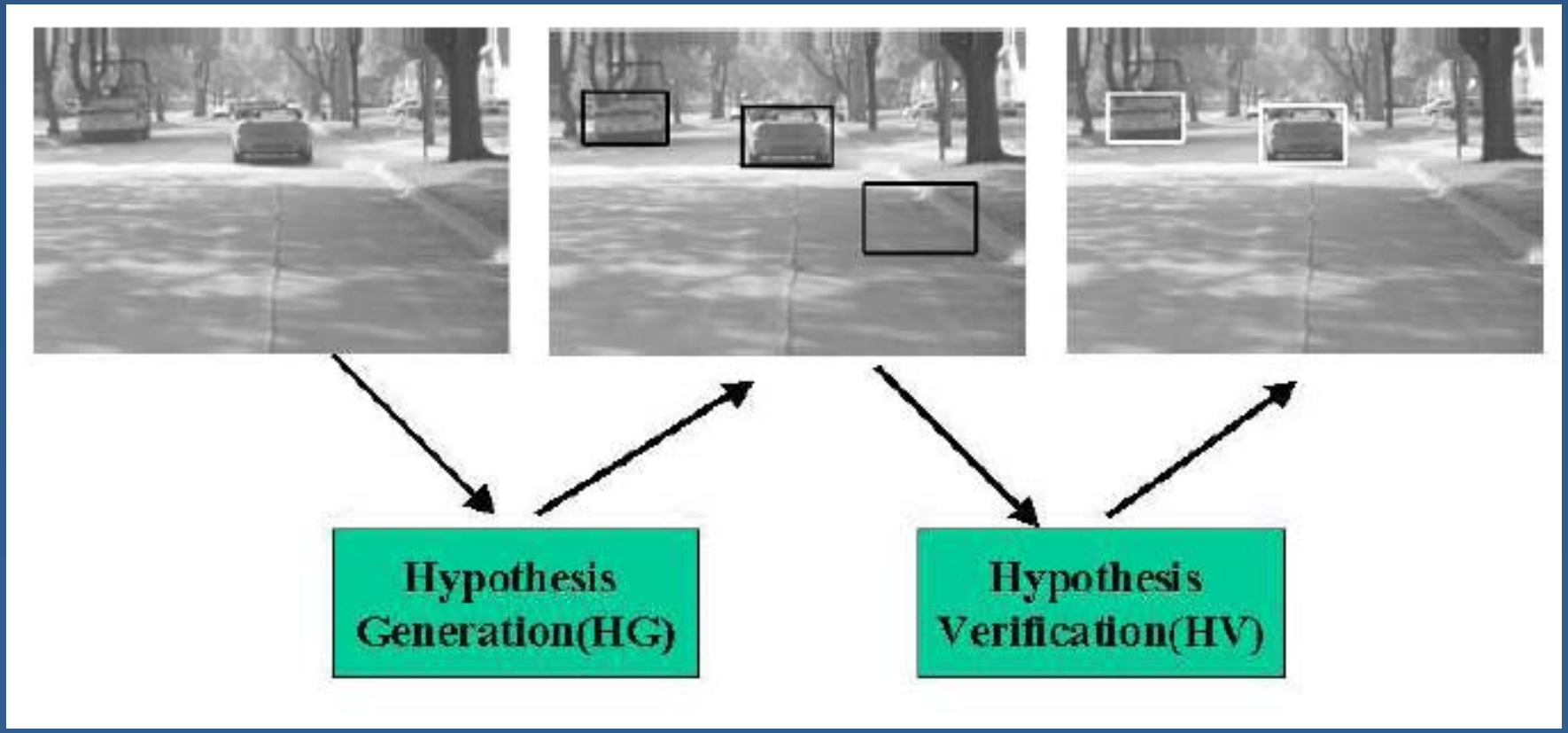
  - Use some heuristics to find configurations and/or combinations of those features specific to the object of interest

Many Approaches to Recognition



(courtesy of G. Bradski)

A Common 2-Step Strategy



Possible Things to Look For

- Symmetry
- Color
- Shadow
- Corners
- Edges
- Texture
- Taillights

Symmetry

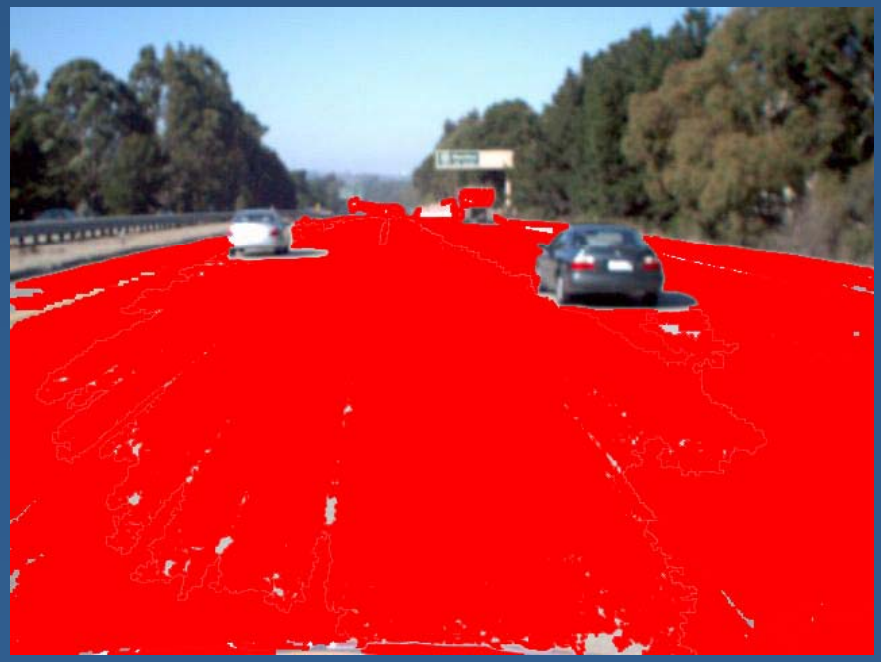
- Observed from rear view, a car generally has vertical symmetry

- Problems:
  - Symmetry estimation is sensitive to noise
  - Prone to false detections, such as symmetrical background objects
  - Doesn't work for partly occluded vehicles

Color

- Road is a fairly constant color
- Non-road regions within a road area are potential vehicles

- Problems:
  - Color of an object depends on illumination, reflectance properties of the object, viewing geometry, and camera properties
  - Color of an object can be very different during different times of the day, under different weather conditions, and under different poses



Shadow

- Area underneath a vehicle is distinctly darker than any other areas on an asphalt paved road.

- Problem:
  - Doesn‘t work in rain, under bad illumination (under a bridge for example)
  - Intensity of the shadow depends on illumination of the image: how to choose appropriate threshold values?

Corners

- Vehicles in general have a rectangular shape
- Can use four templates one for each corner, to detect all corners in image, and then use a search method to find valid configurations (matching corners)

Edges

- Rear views of cars contain many horizontal and vertical structures (rear-window, bumper, etc.)
- One possibility: horizontal edge detector on the image (such as Sobel operator), then sum response in each column to locate horizontal position (should be at the peaks)
- Problem:
  - Lots of parameters: threshold values for the edge detectors, the threshold values for picking the most important vertical and horizontal edges, and the threshold values for choosing the best maxima in profile image

Texture

- Presence of a car causes local intensity changes
- General similarities among all vehicles means that the intensity changes may follow a certain pattern
- Problem: in most environments the background contains lots of texture as well

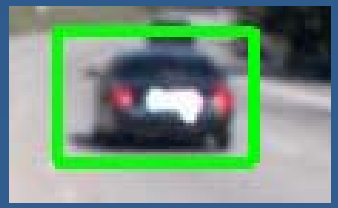
Taillights

- Fairly salient feature of all vehicles
- Problem:
  - A little different on every car
  - Not that bright during the daytime; probably would work only at night.

Other Factors to Consider

- Perspective:
  This is not a likely position / size

- Shape:
  Trace along the outer contour



General Problem

- For complex objects, such as vehicles, it is hard to find features and heuristics that will handle the huge variety of instances of the object class:
  - May be rotated in any direction  
  - Lots of different kinds of cars in different colors
  - May be a truck 
  - May have a missing headlight, bumper stickers, etc.
  - May be half in light, half in shadow 

Statistical Model Training

- Training Set
  - Positive Samples
  - Negative Samples






- Different features are extracted from the training samples and distinctive features that can be used to classify the object are selected.
- This information is “compressed” into the statistical model parameters.
- Each time the trained classifier does not detect an object (misses the object) or mistakenly detects the absent object (gives a false alarm), model is adjusted.

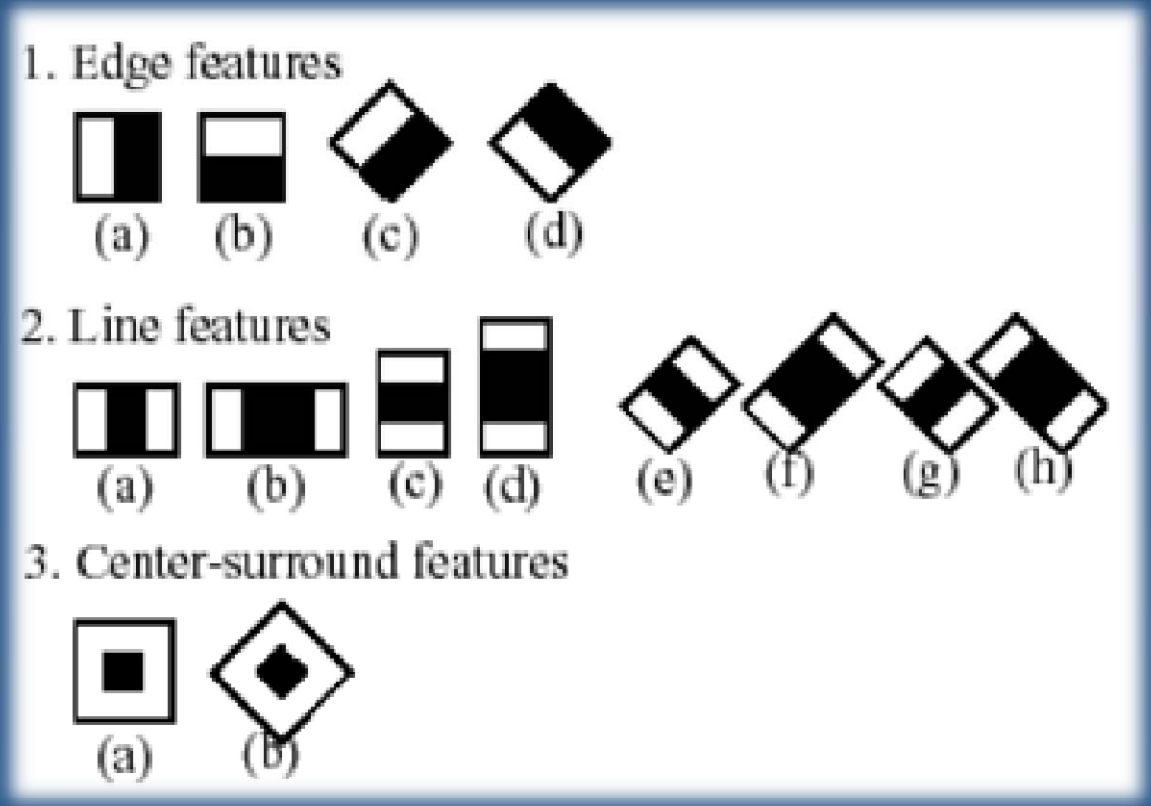
Training in OpenCV

- Uses simple features and a cascade of boosted tree classifiers as a statistical model.
- Paul Viola and Michael J. Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features*. IEEE CVPR, 2001.
- Rainer Lienhart and Jochen Maydt. *An Extended Set of Haar-like Features for Rapid Object Detection*. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

Approach Summary

- Classifier is trained on images of fixed size (Viola uses 24x24)
- Detection is done by sliding a search window of that size through the image and checking whether an image region at a certain location “looks like a car” or not.
- Image (or classifier) can be scaled to detect objects of different sizes.
- A very large set of very simple “weak” classifiers that use a single feature to classify the image region as car or non-car.
- Each feature is described by the template (shape of the feature), its coordinate relative to the search window origin and the size (scale factor) of the feature.

Types of Features



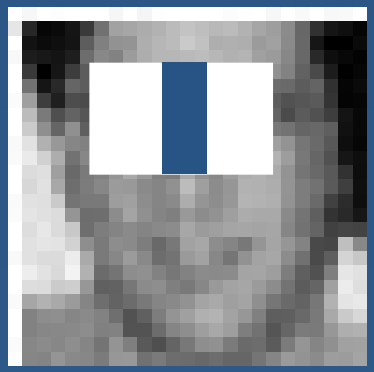
- Feature's value is a weighted sum of two components:
  - Pixel sum over the black rectangle
  - Sum over the whole feature area

Weak Classifier

- Computed feature value is used as input to a very simple decision tree classifier with 2 terminal nodes

$$f_i = \begin{cases} +1, & x_i \geq t_i \\ -1, & x_i < t_i \end{cases}$$

- 1 means "car" and -1 means "non-car"



Bar detector works well for "nose," a face detecting stump.

It doesn't work well for cars.

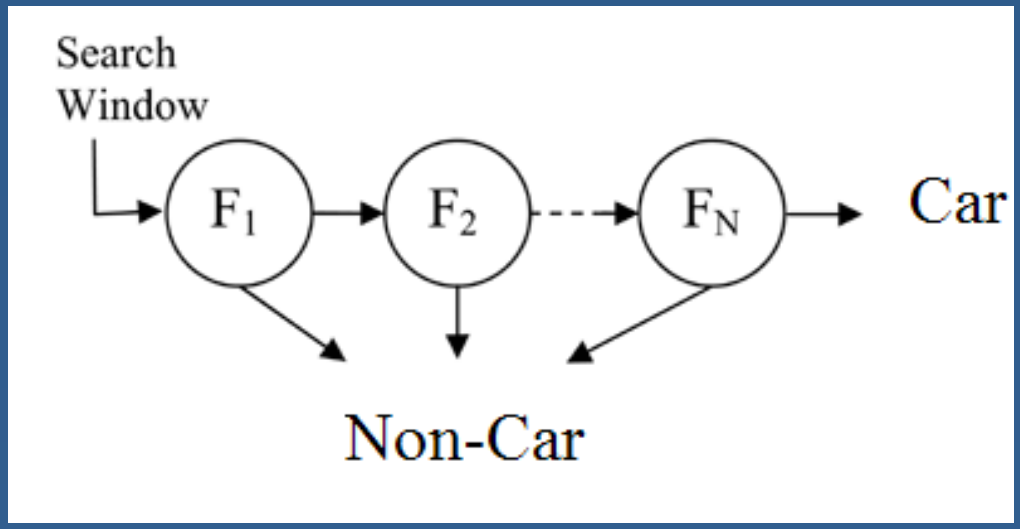
Boosted Classifier

- Complex and robust classifier is built out of multiple weak classifiers using a procedure called boosting.
- The boosted classifier is built iteratively as a weighted sum of weak classifiers:

  $F = sign(c_1 f_1 + c_2 f_2 + \ldots + c_n f_n)$
- On each iteration, a new weak classifier f_i is trained and added to the sum. The smaller the error f_i gives on the training set, the larger is the coefficient c_i that is assigned to it.

Cascade of Boosted Classifiers

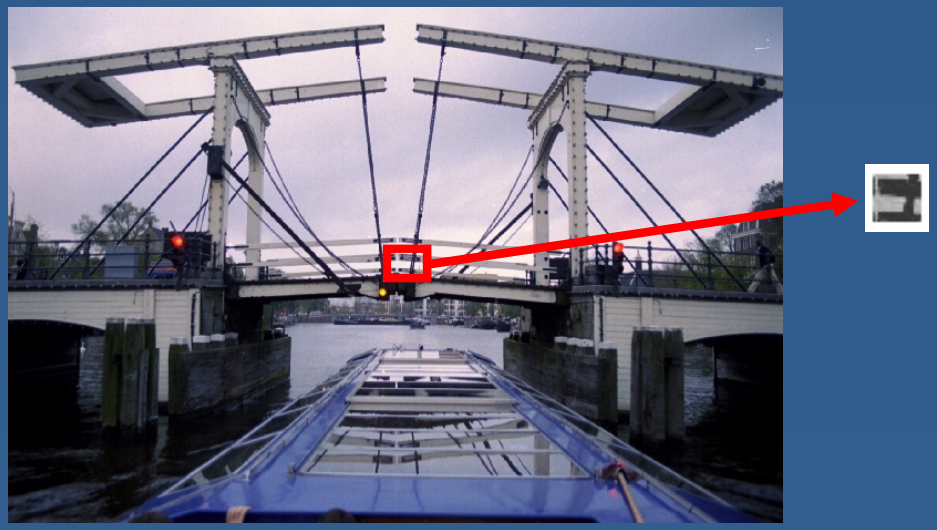
- Sequence of boosted classifiers with constantly increasing complexity
- Chained into a cascade with the simpler classifiers going first.



Classifier Cascade Demo

Identifying a Particular Object

- Correlation-based template matching?
  - Computationally infeasible when object rotation, scale, illumination and 3D pose vary
  - Even more infeasible with partial occlusion



Local Image Features

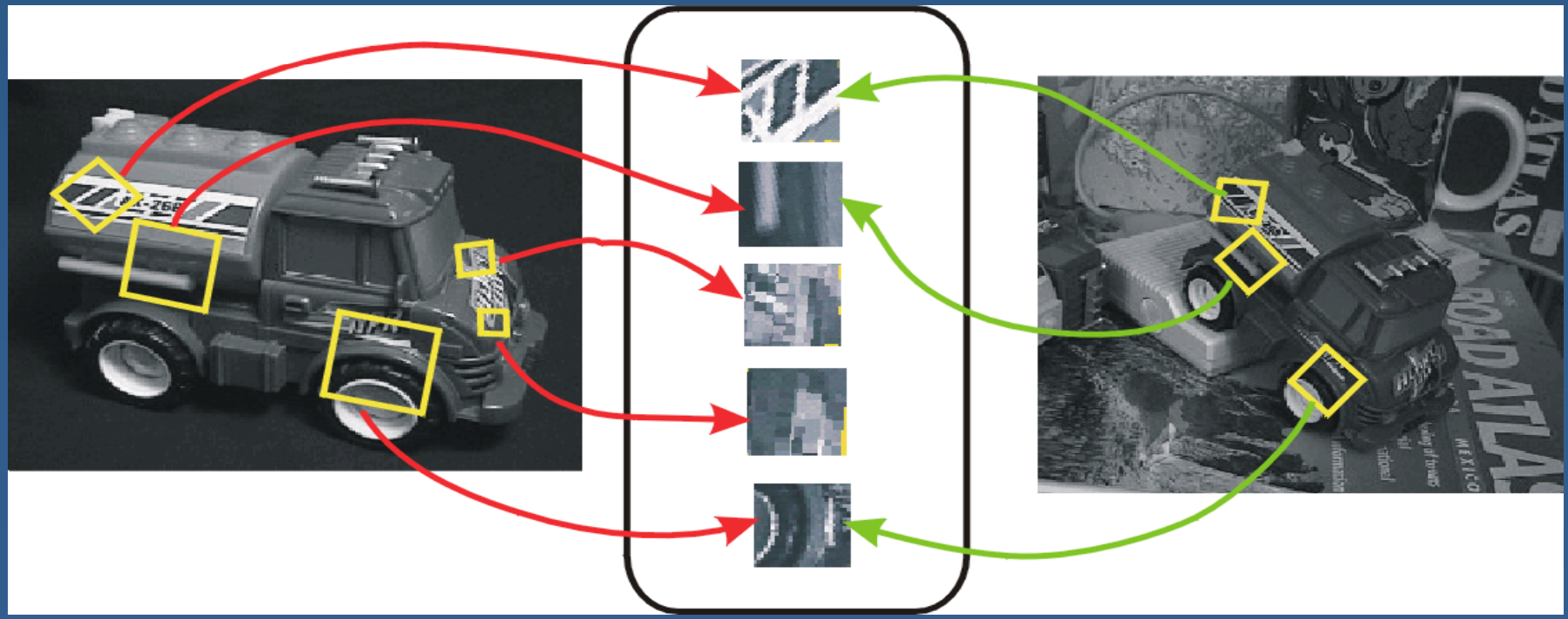
- We need to learn a set of local object features that are unaffected by
  - Nearby clutter
  - Partial occlusion
- … and invariant to
  - Illumination
  - 3D projective transforms
  - Common object variations
- …but, at the same time, sufficiently distinctive to identify specific objects among many alternatives!

What Features to Use?

- Line segments, edges and region grouping?
  - Detection not good enough for reliable recognition
- Peaks detection in local image variations?
  - Example: Harris corner detector
  - Problem: image examined at only a single scale
  - Different key locations as the image scale changes

Invariant Local Features

- Goal: Transform image content into local feature coordinates that are invariant to translation, rotation, and scaling



SIFT Method

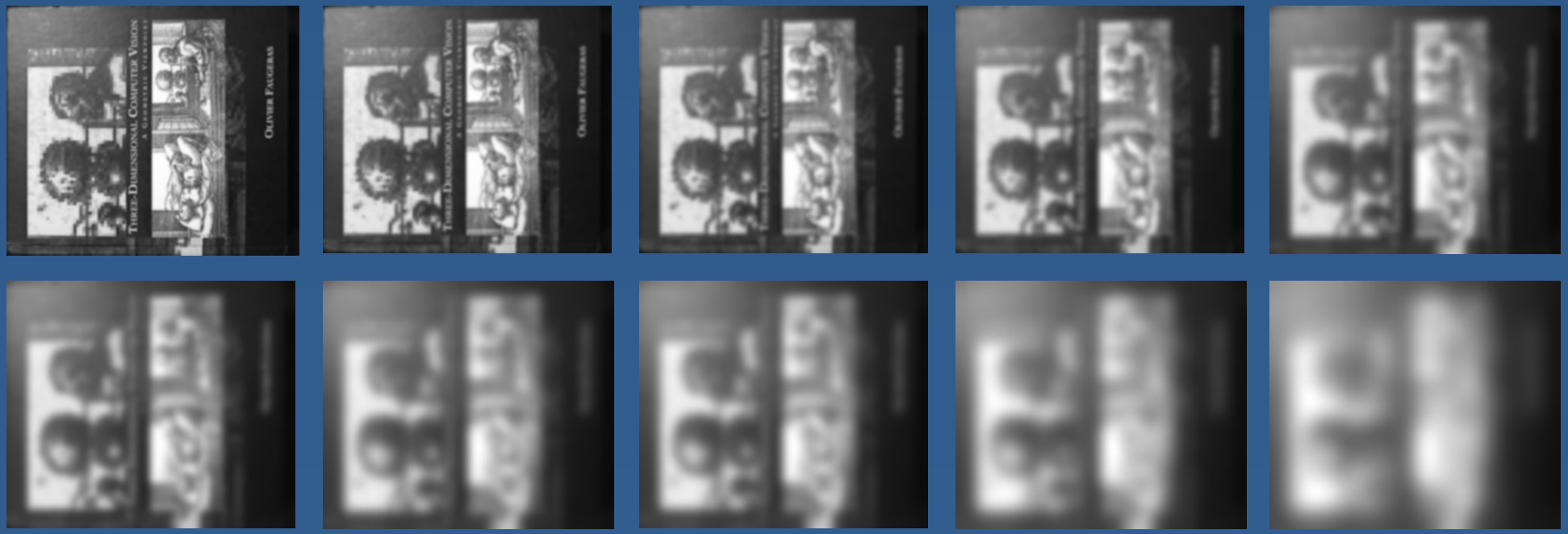
- Scale Invariant Feature Transform (SIFT)
- Staged filtering approach
  - Identifies stable points (called image “keys”)
- Computation time less than 2 seconds
- Local features:
  - Invariant to image translation, scaling, rotation
  - Partially invariant to illumination changes and 3D projection (up to 20° of rotation)
  - Minimally affected by noise
  - Similar properties with neurons in inferior temporal cortex used for object recognition in primate vision

SIFT: Algorithm Outline

- Detect extrema across a “scale-space”
  - Uses a difference-of-Gaussians function
- Localize keypoints
  - Find sub-pixel location and scale to fit a model
- Assign orientations to keypoints
  - 1 or more for each keypoint
- Build keypoint descriptor
  - Created from local image gradients

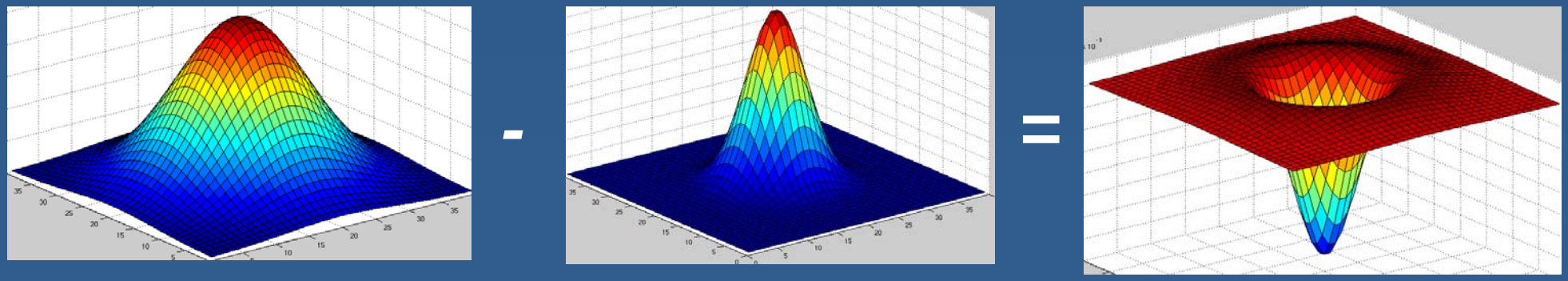
Scale Space

- Build a pyramid of images
  - Images are difference-of-Gaussian functions
  - Resampling between each level



Finding Keypoints

- Basic idea: find strong corners in the image, but in a scale invariant fashion
  - Run a linear filter (difference of Gaussians) at different resolutions on image pyramid



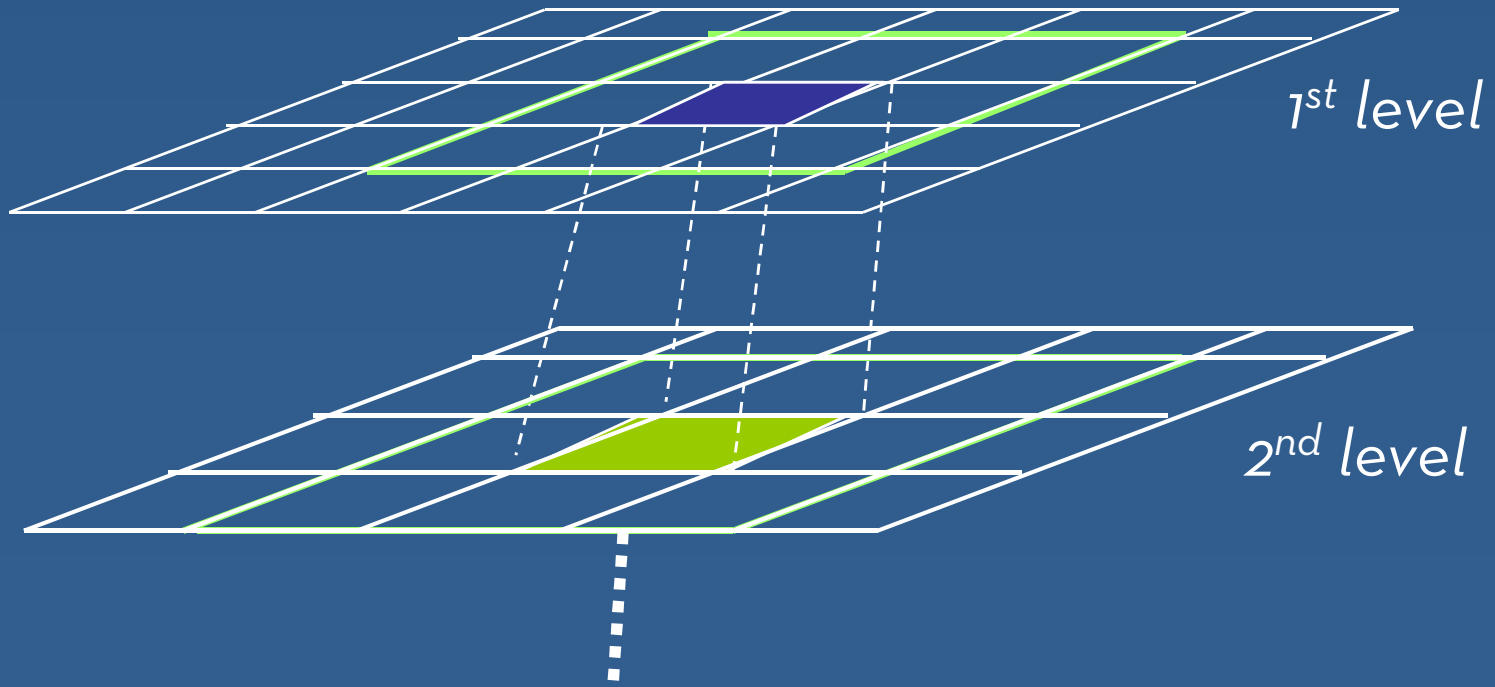
Difference of Gaussians

Finding Keypoints

- Keypoints are the extreme values in the difference of Gaussians function across the scale space

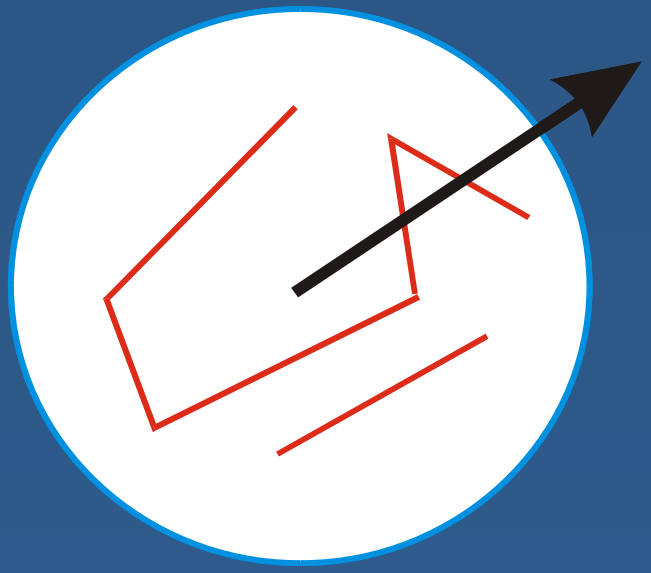
Key Localization

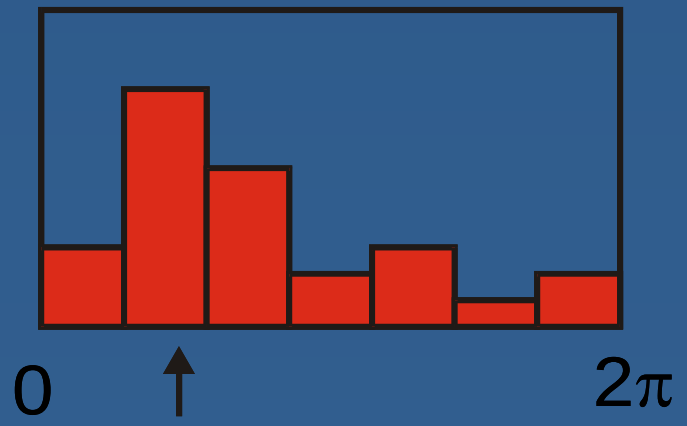
- Find maxima and minima, and extract image gradient and orientation at each level to build an orientation histogram



Select Canonical Orientation

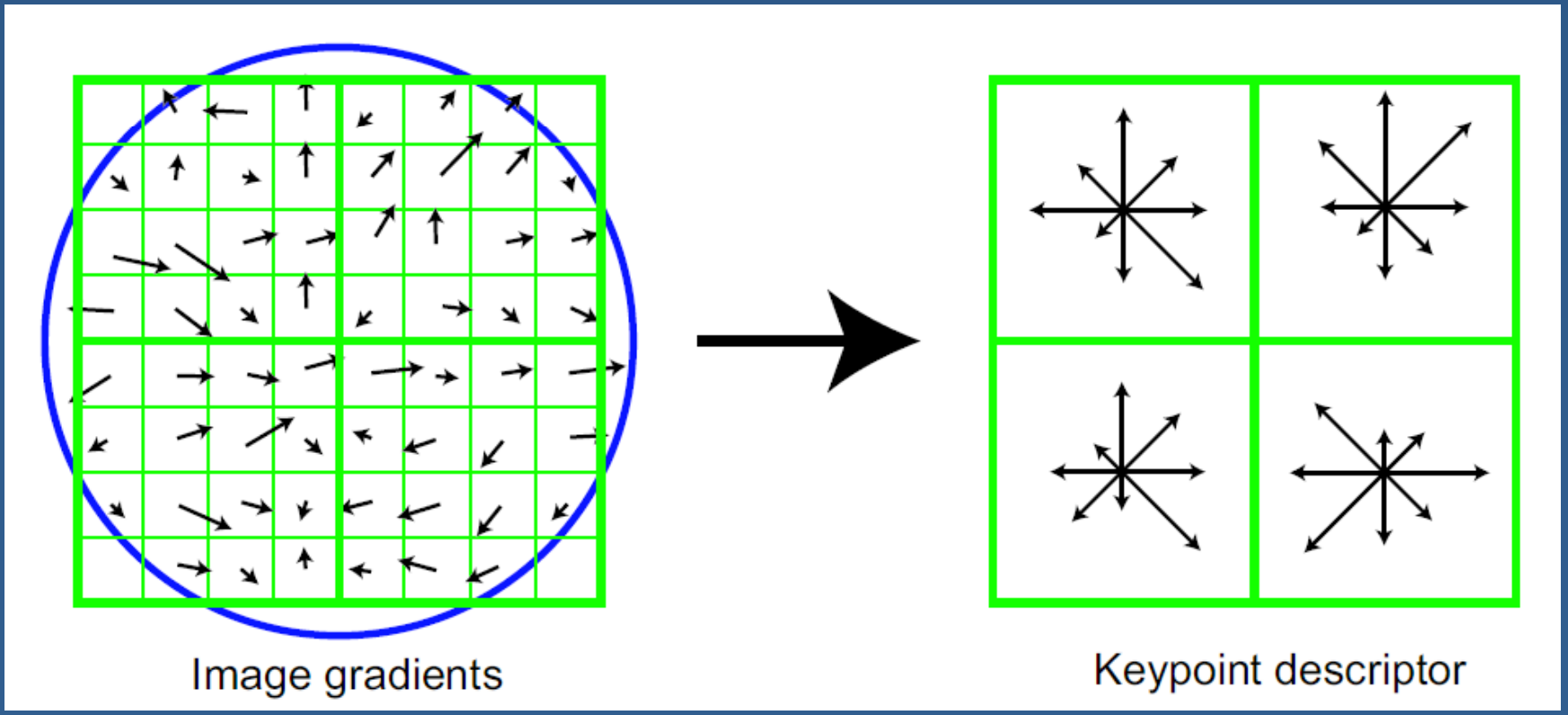
- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram





Orientation Histogram

- Stored relative to the canonical orientation for the keypoint, to achieve rotation invariance



SIFT Keypoints

- Feature vector describing the local image region sampled relative to its scale-space coordinate frame
- Represents blurred image gradient locations in multiple orientations planes and at multiple scales

SIFT: Experimental Results

Image transformation	Location and scale match	Orientation match
Decrease constrast by 1.2	89.0 %	86.6 %
Decrease intensity by 0.2	88.5 %	85.9 %
Rotate by 20°	85.4 %	81.0 %
Scale by 0.7	85.1 %	80.3 %
Stretch by 1.2	83.5 %	76.1 %
Stretch by 1.5	77.7 %	65.0 %
Add 10% pixel noise	90.3 %	88.4 %
All previous	78.6 %	71.8 %

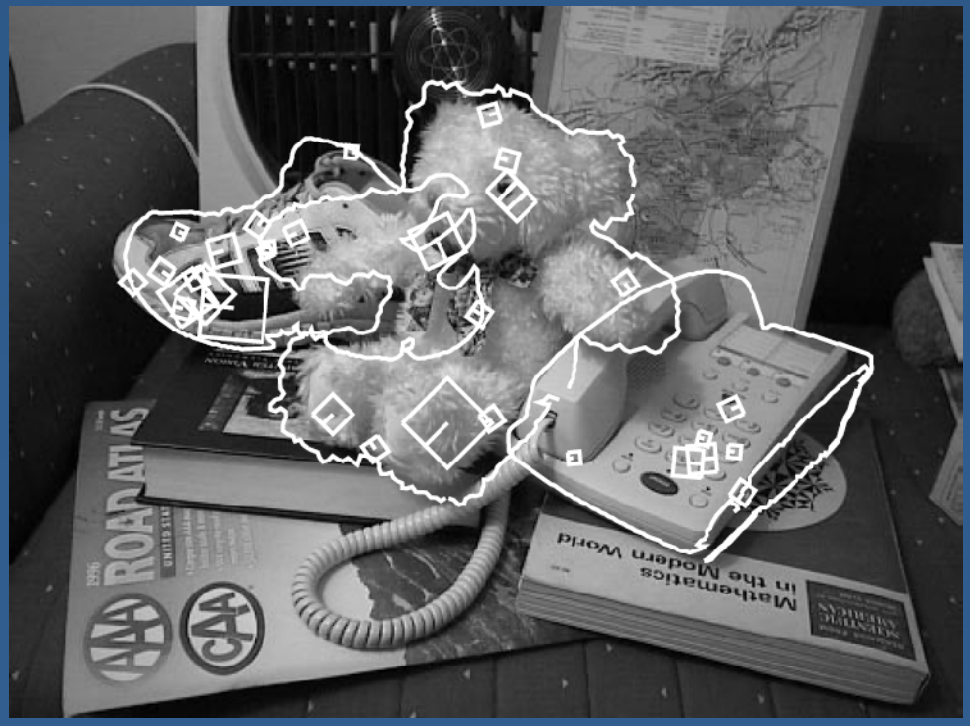
SIFT: Finding Matches

- Goal: identify candidate object matches
  - The best candidate match is the nearest neighbor (i.e., minimum Euclidean distance between descriptor vectors)
  - The exact solution for high dimensional vectors has high complexity, so SIFT uses approximate Best-Bin-First (BBF) search method (Beis and Lowe)
- Goal: Final verification
  - Low-residual least-squares fit
  - Solution of a linear system: x = [ATA]-1ATb
  - When at least 3 keys agree with low residual, there is strong evidence for the presence of the object
  - Since there are dozens of keys in the image, this still works with partial occlusion

SIFT Example

(courtesy of David Lowe)

Partial Occlusion Example



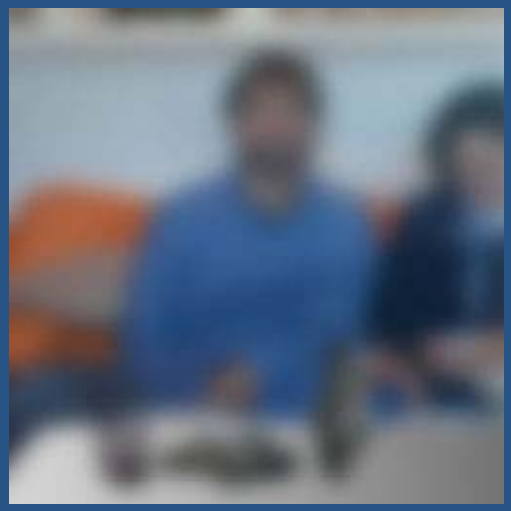
(courtesy of David Lowe)

SIFT Demo

Context Focus?

(courtesy of Kevin Murphy)

Importance of Context

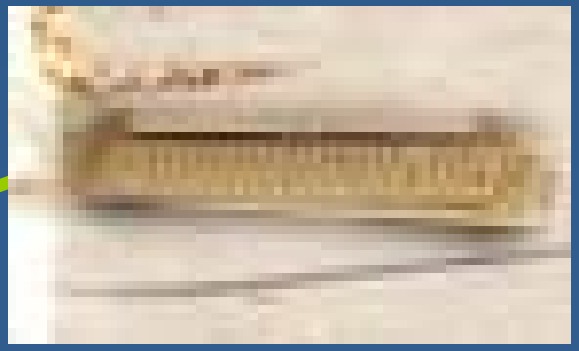


(courtesy of Kevin Murphy)

Importance of Context

We know there is a keyboard present in this scene even if we cannot see it clearly.

We know there is no keyboard present in this scene...



... even if there is one indeed.

(courtesy of Kevin Murphy)

Summary

- SIFT
  - Very robust recognition of a specific object, even with messy backgrounds and varying illumination conditions
  - Very quick and easy to train
  - A little too slow to run at interactive frame rates (if the background is complex)
- Classifier Cascades
  - Can recognize a general class of objects (rather than a specific object)
  - Less robust (more false positives / missed detections)
  - Slow to train, requires many examples
  - Runs well at interactive frame rates