Why extract features from camera images?

- Motivation: understanding images is really hard!
  - Lots of data
  - Some parts of the image are “boring”

- Idea: extract “good” features
  - From 1M pixels to 100s of features
  - Can make features robust

Corner Detectors

- Intuitively, corners are a good feature.
  - Relatively easy to find
  - Trackable

- But what is a corner?
  - We're processing from bottom-up
  - No idea (yet) about objects
    - a corner != object corner

- What isn't a corner?
  - Uniform areas
  - Edges/lines

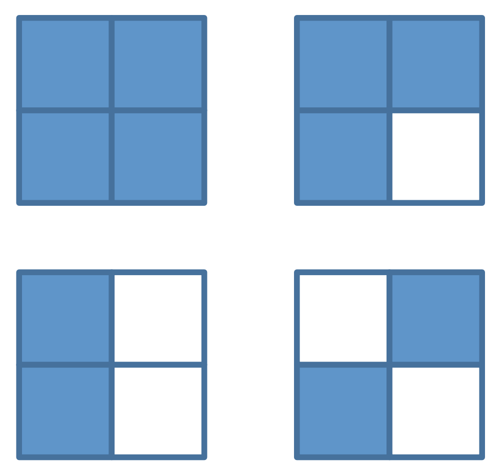
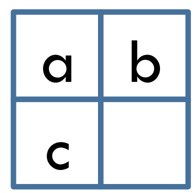


Image Gradients

- Idea: let's look at gradients of a patch of pixels
  - Gradient at pixel a is (b-a, c-a)



- Compute gradients for 2x2 area
  - We need 3x3 input…

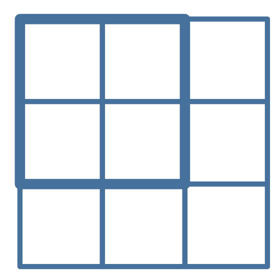


Image Gradients

- Are these good corners?
  - (What are the gradients?)

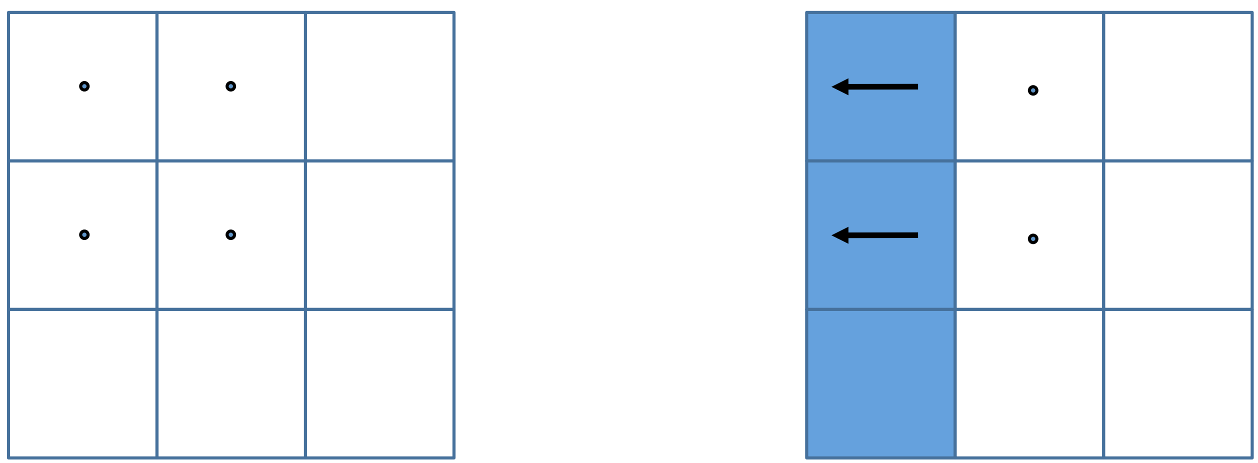
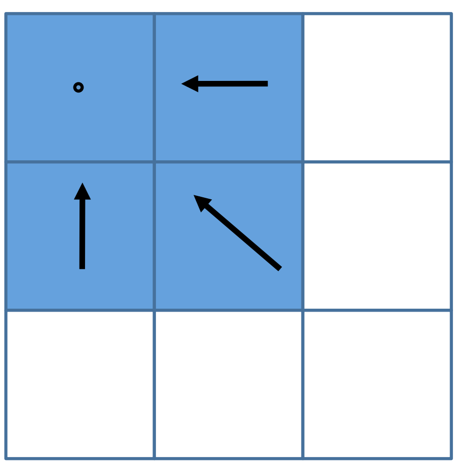
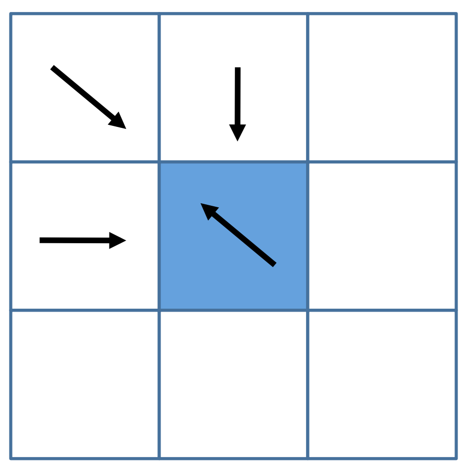
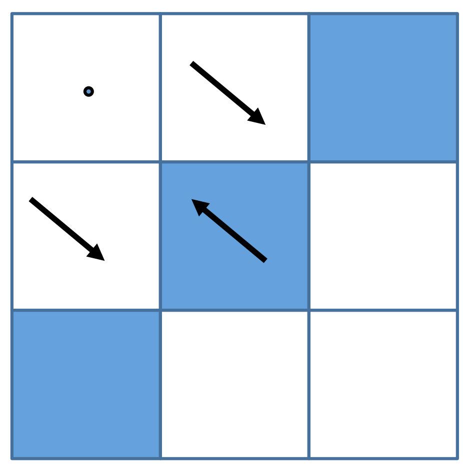


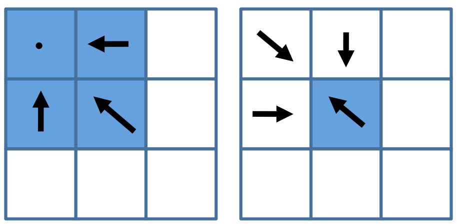
Image Gradients

- Are these good corners?
  - (What are the gradients?)

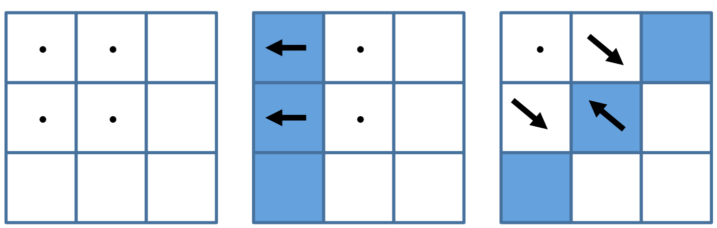




Good and Bad Corners

- Good Corners

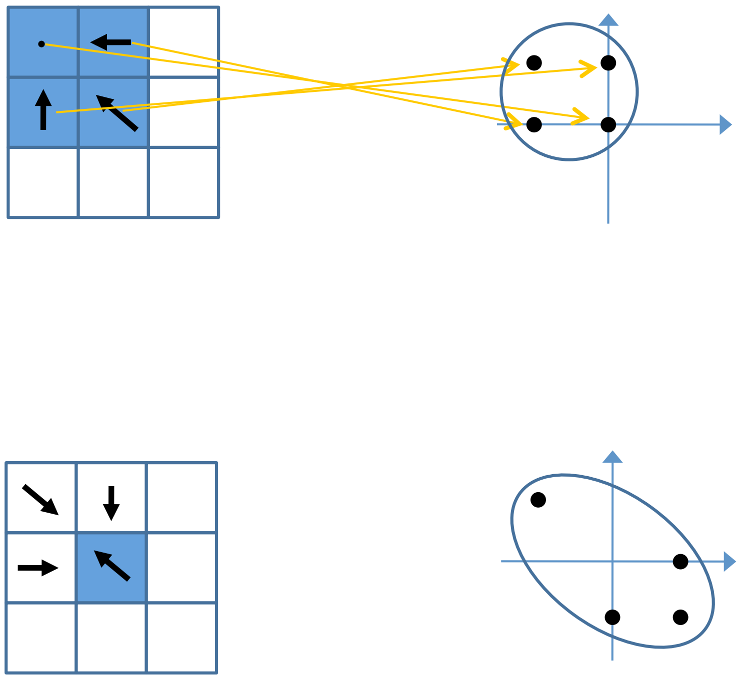


- Bad Corners

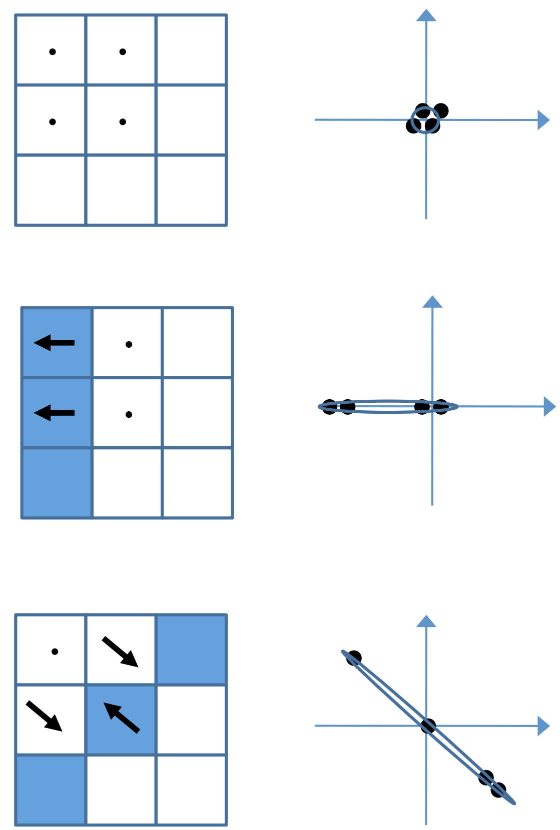


- What do good/bad corners have in common?

Corners in gradient space



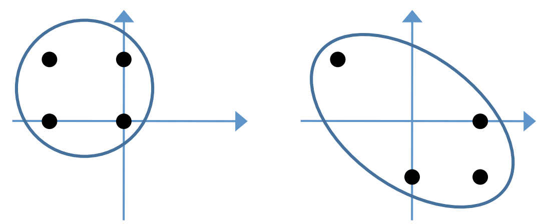
Good Corners



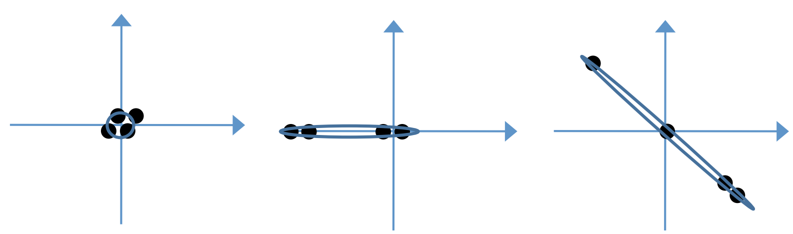
Bad Corners

Harris Corner Detector

- Good Corners



- Bad Corners



- Idea: The "fatness" of the covariance ellipse of the gradient directions is a measure of "cornerness"
    - How do we compute this?

Computing corner response

- Compute S matrix
  - Covariance of the gradients

$$S = \sum_i g_i g_i^T$$

- Compute eigen-values
  - Corner response = *smallest* eigen value
  - How bad (computationally) is this?
- Identify pixels with "good" corner responses
  - Thresholding

- A handful of practical issues!
- Noise in original image
  - Creates false positives
  - Apply low-pass filter *first*
    - Also improves isotropicity of response
- Local maximum suppression
- How big a patch to compute gradients over?

Actually, I lied.

- There are two very closely related corner detectors
  - Kanade-Tomasi
    - what we described (uses eigenvalues)
  - Harris
    - identical, except uses (bad) approximation of eigenvalue.

$$\mathrm{trace(S)} = \lambda_1 + \lambda_2$$
$$\mathrm{det(S)} = \lambda_1 \lambda_2$$
$$M = \det(S) - \kappa \mathrm{trace}(S)^2$$

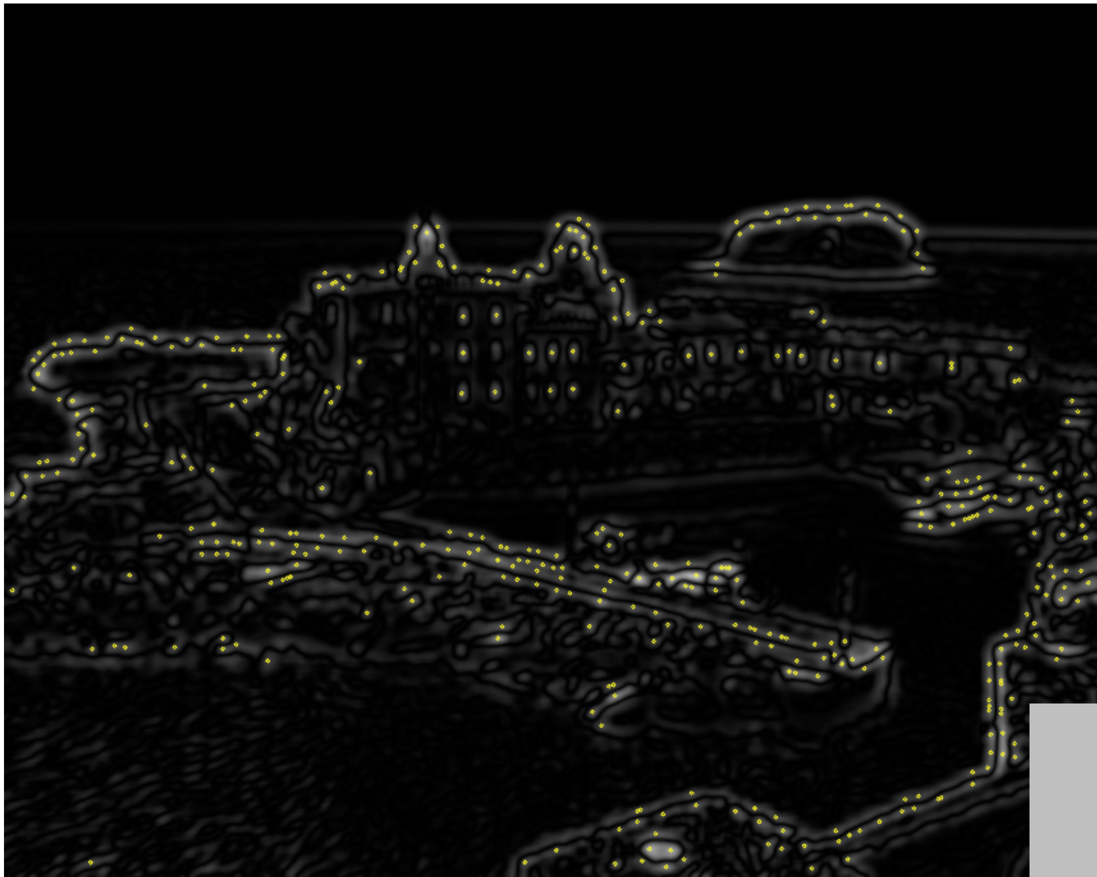
"Area of ellipse, minus a penalty for those that are highly eccentric."

Difference of Gaussians

- Another way of looking at corner detectors
  - Look for areas with high frequency in both directions

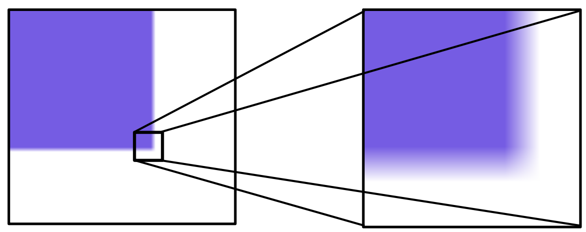
- What frequencies to look for?
  - We want a band pass
    - not too high (it's noise!)
    - not too low (it's not a corner!)

- Filter an image with two different Gaussians
  - Each corresponds to a low-pass filter
  - Difference corresponds to a band-pass filter




Multiple Scales

- Corners of high-resolution images are often blurry or noisy at fine levels of detail



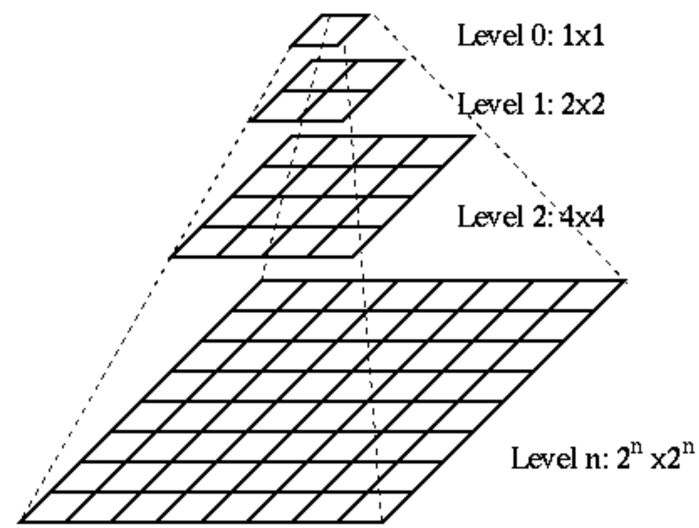
Corner indistinct at high resolution: no corner extracted

- Idea: run Harris corner detector on down-sampled versions of the image
  - Extract corners, blur, decimate, repeat.

- Idea: repeatedly compute DoG, increasing both sigma1 and sigma2
  - Look for successively lower frequency corners
  - Better yet, once we've band-limited "enough", we can decimate the image!

Image Pyramids

- Look for features on multiple scales
  - Just repeat image processing algorithm on successively lower-resolution images
  - Must produce lower-resolution images



- Avoiding aliasing requires low-pass filters
  - Ideal low-pass filter?
  - Don't create new features when filtering
    - Avoid ringing!
    - Want a monotonic filter

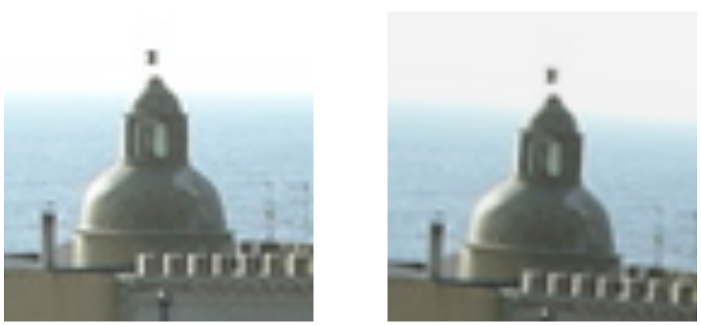
Feature Tracking

- We often want to track (or match) features across two frames.
  - Which corners in image A match those in image B?
  - i.e., data association

- Can we use *more* information?
  - Why not use the local appearance?

Image patch patching

- Consider the pixel patch around a feature
  - Sum of absolute/squared (SAD/SSE) differences/errors
- How robust is this to small alignment errors/rotations/ changes in viewpoint/etc.?



These will probably match

These probably won’t

Invariances

- Our goal: detect distinctive features, maximizing repeatability
  - Transform pixel patch into a space where a simple comparison (SAD/SSE) *is* effective.
- Scale invariance
  - Robust to changes in distance
- Rotation invariance
  - Robust to rotations of camera
- Affine invariance
  - Robust to tilting of camera
- Brightness invariance
  - Robust to minor changes in illumination

SIFT: Scale-Invariant Feature Transform

- David Lowe (Univ. British Columbia)

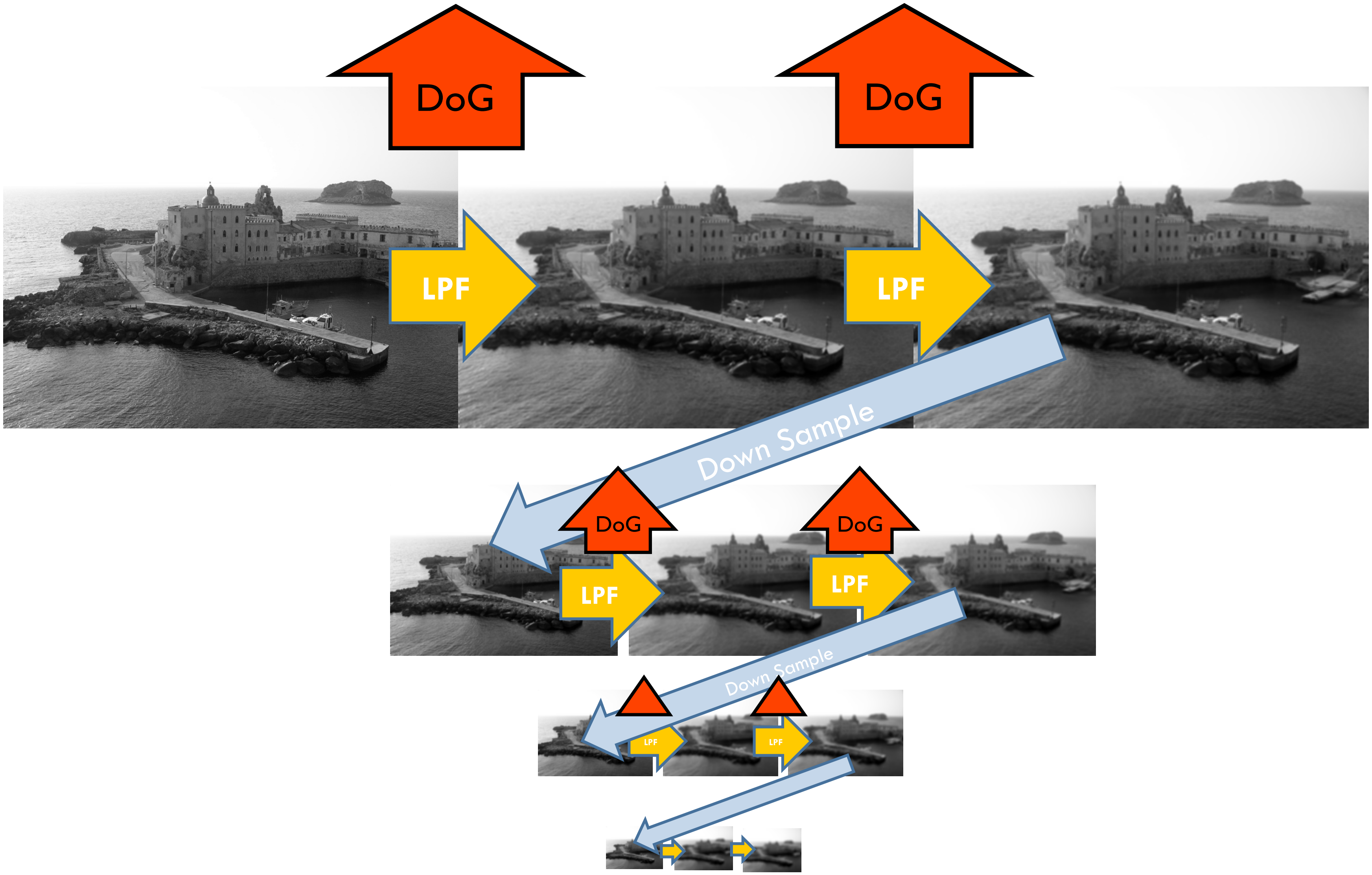
- Probably the single most commonly used tool in computer vision
  - For better or for worse... often used “reflexively” even if it’s not a good choice!

- Watch out!
  - Patented, commercial use restricted

SIFT

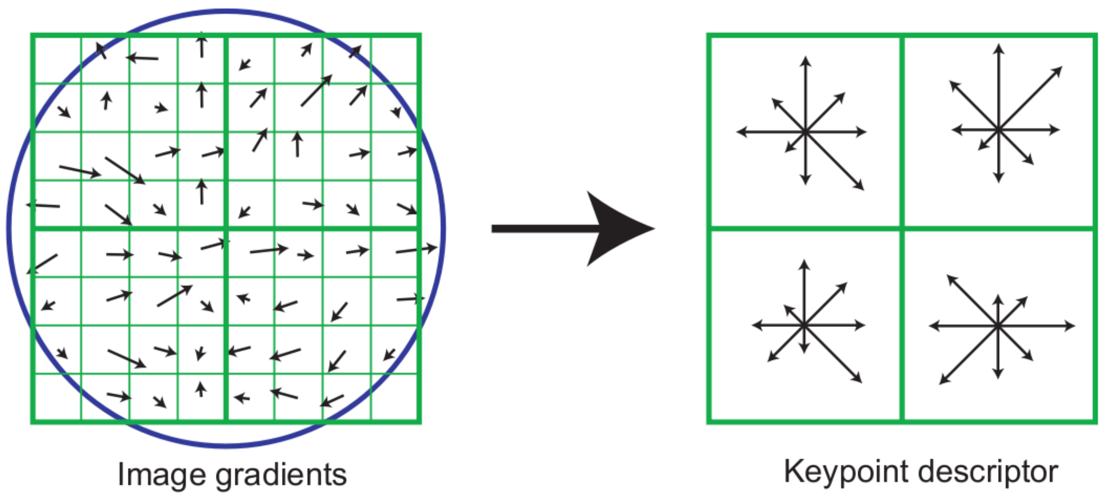
- Detect interest points
  - Image pyramid using DoG "corners"
  - Output: corners *and* scale (which level of the pyramid?)
- Output a "descriptor"
  - Consider pixel match around corner
  - Compute a histogram of the gradient directions
  - "Rotate" the histogram so that the dominant direction is first.

Sub-Octave Image Pyramids



SIFT Descriptor

- Histogram of gradients gives good information about a pixel patch
  - But building just one histogram loses a lot of spatial information.
  - Idea: For a given interest point, compute a set of histograms; output each.
  - Shift histograms so dominant direction is first in histogram ==> rotational invariance.



- "Official" SIFT uses 16x16 pixel patches, 4x4 bins, 8 histogram buckets
- How many degrees of freedom in SIFT descriptor?
  - # bins * # histogram buckets = 4*4*8 = 128

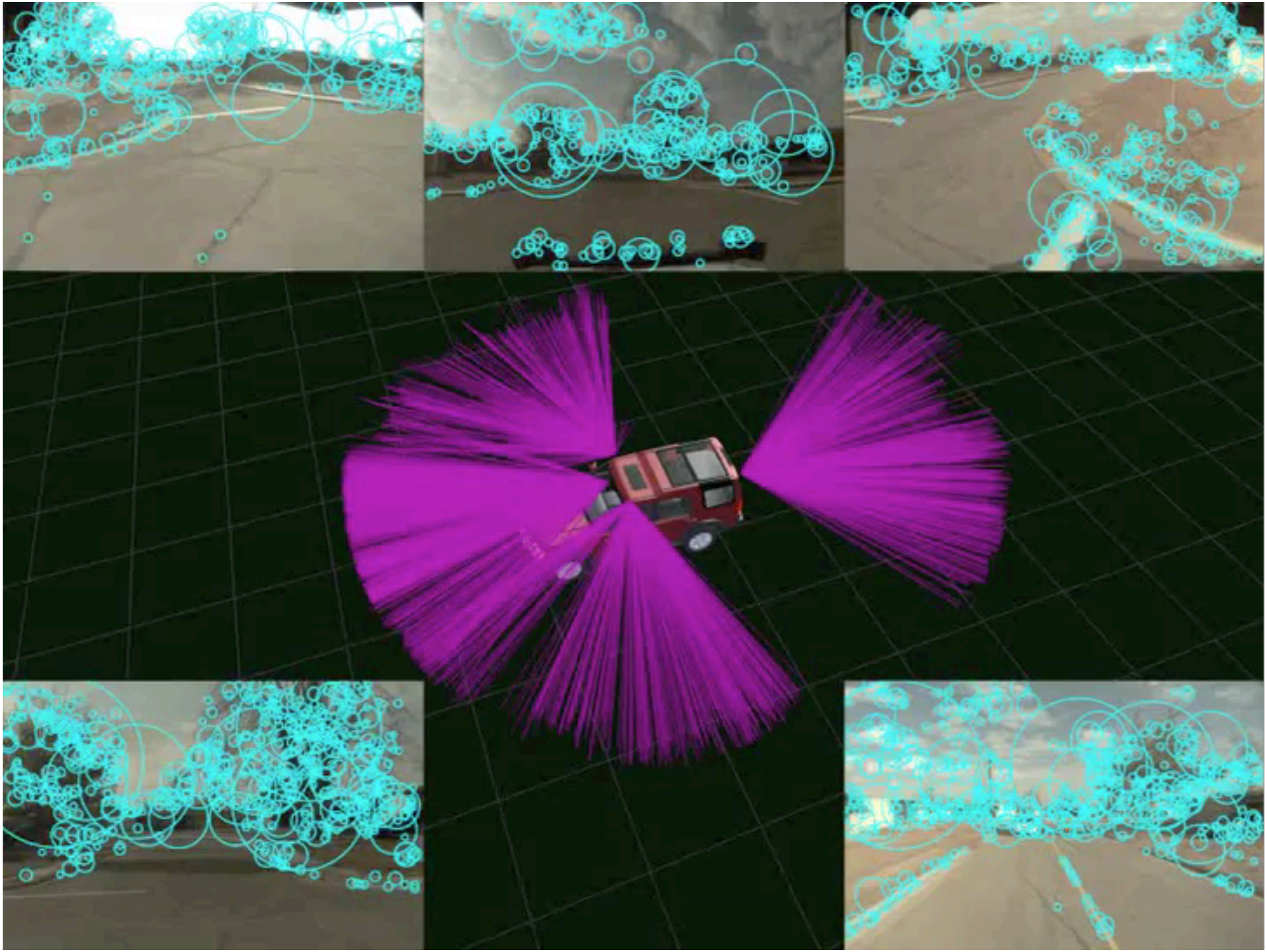
Matching SIFT Descriptors

- Each SIFT feature:
  - (x,y,scale) (ignore scale if you want scale invariance!)
  - descriptor[128]

- Two descriptors can be compared using Euclidean distance…
  - Small distances = similar descriptors
  - What if same/similar feature appears more than once? nearest neighbor may not be good enough

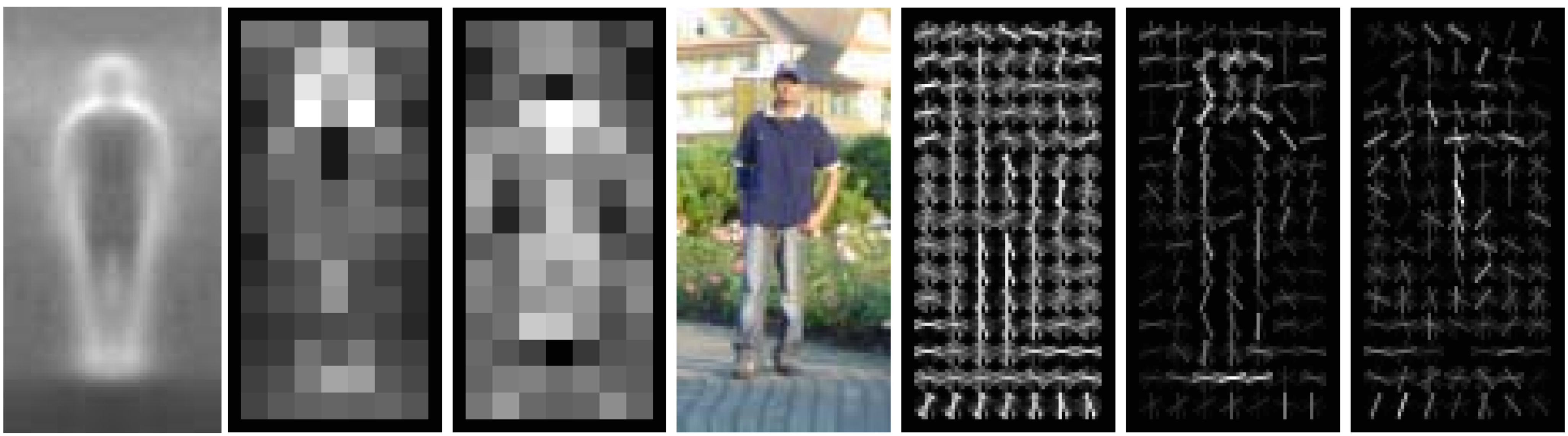
- Common approach:
  - Suppose best match for Ai is Bj (with dij).
  - Suppose next best match for Ai is Bk (with dik).
  - Require dij < alpha dik. (alpha typically 0.8).

- "Marriage" constraint: Ai and Bj match only if Bj is the best feature for Ai *and vice versa*.



Histogram of Oriented Gradients

- What all the cool kids are doing these days.
  - Basically the same descriptor as in SIFT, without rotation invariance.
  - Often evaluated densely, used with templates, SVMs for object detection



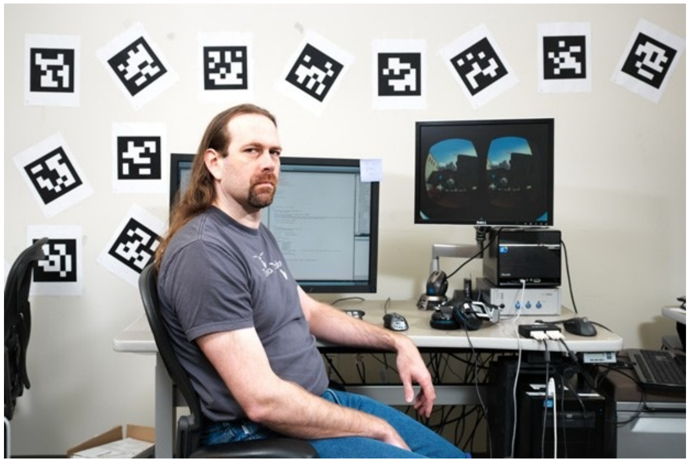
(a) (b) (c) (d) (e) (f) (g)

Object recognition

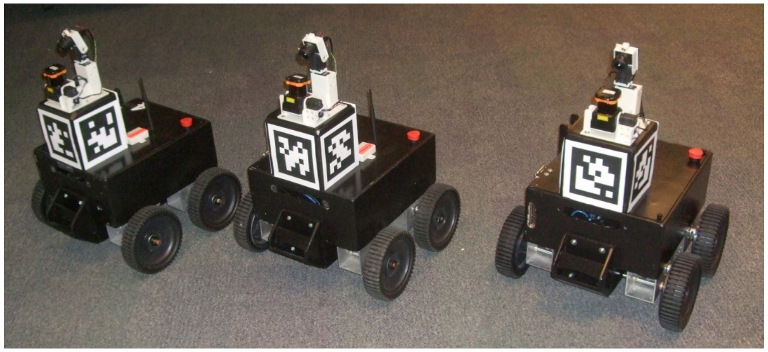
- SIFT also used to build object recognition systems

Artificial Features

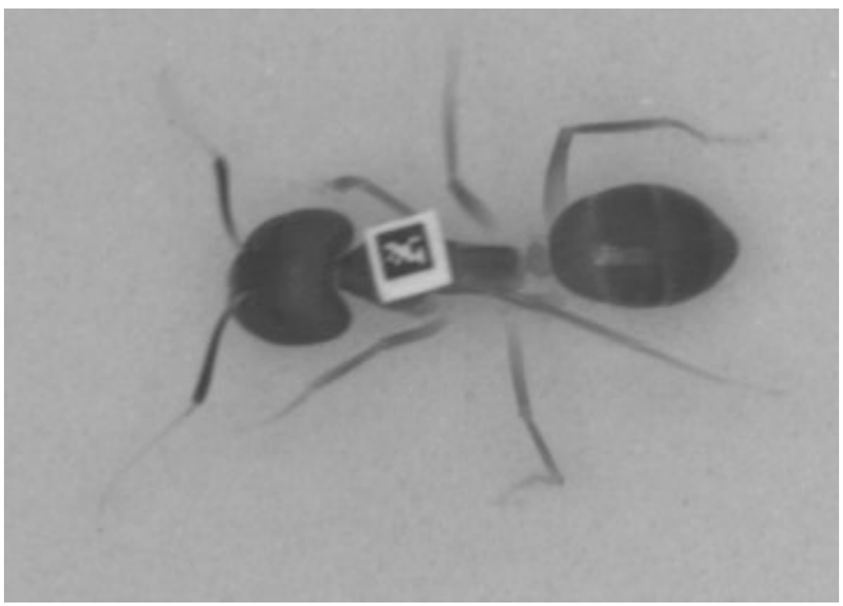
Applications

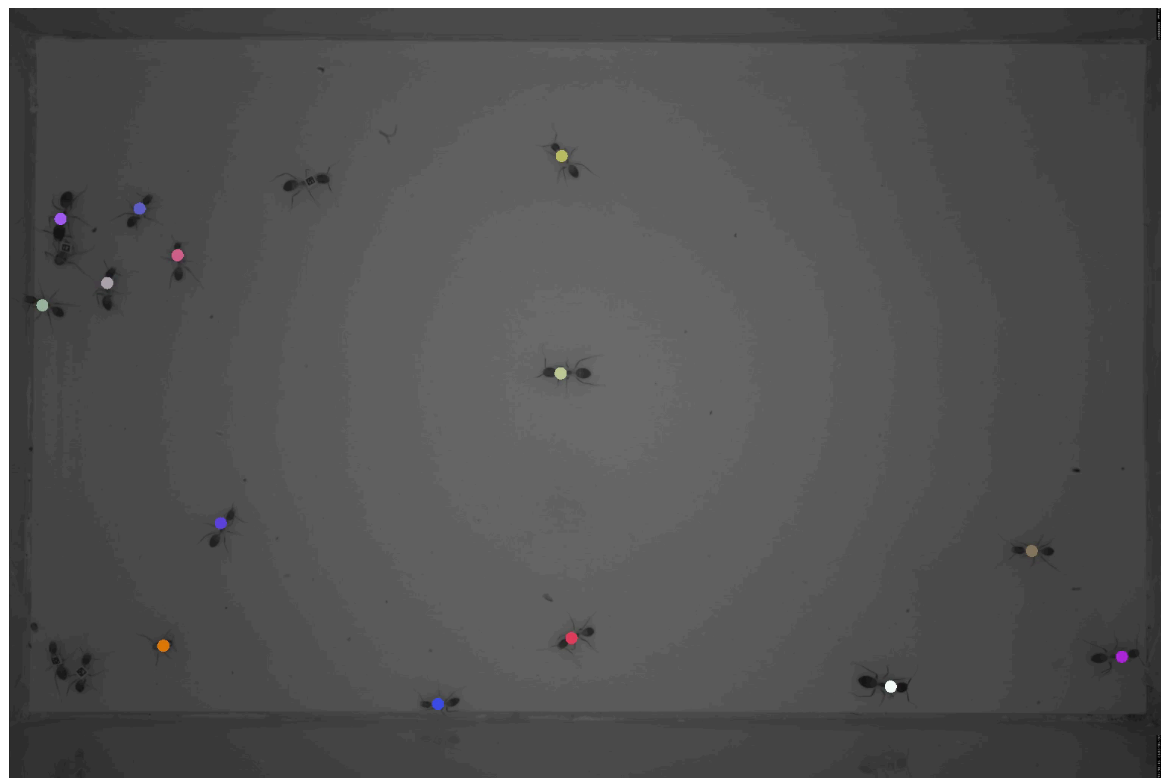


- Ground truthing
- Recognizing robots
- Commanding robots
- Education
  - Often useful to bypass open-ended perception problems




TagTest
smoothing sigma (segmentation)
smoothing sigma (sampling)
magnitude threshold
theta threshold (radians)
debug





Related Work

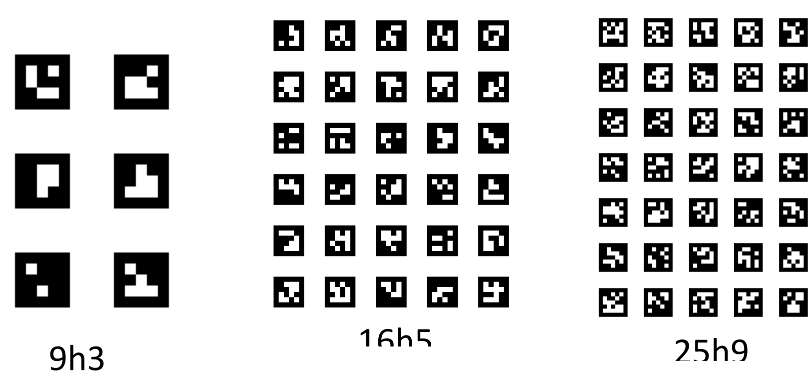
- ARToolkit
  - Widely used
  - Primitive binarization scheme => high failure rate in unstructured environments
  - Weak coding system
  - Freely available
- ARTag (Fiala, 2005)
  - Seems to address many shortcomings in ARToolkit
  - Methods are not well-documented
  - Source code not available

- Bokode (Mohan et al, 2009)
- Fourier codes (Sattar et al, 2007)
- Quick Response (QR) Tags

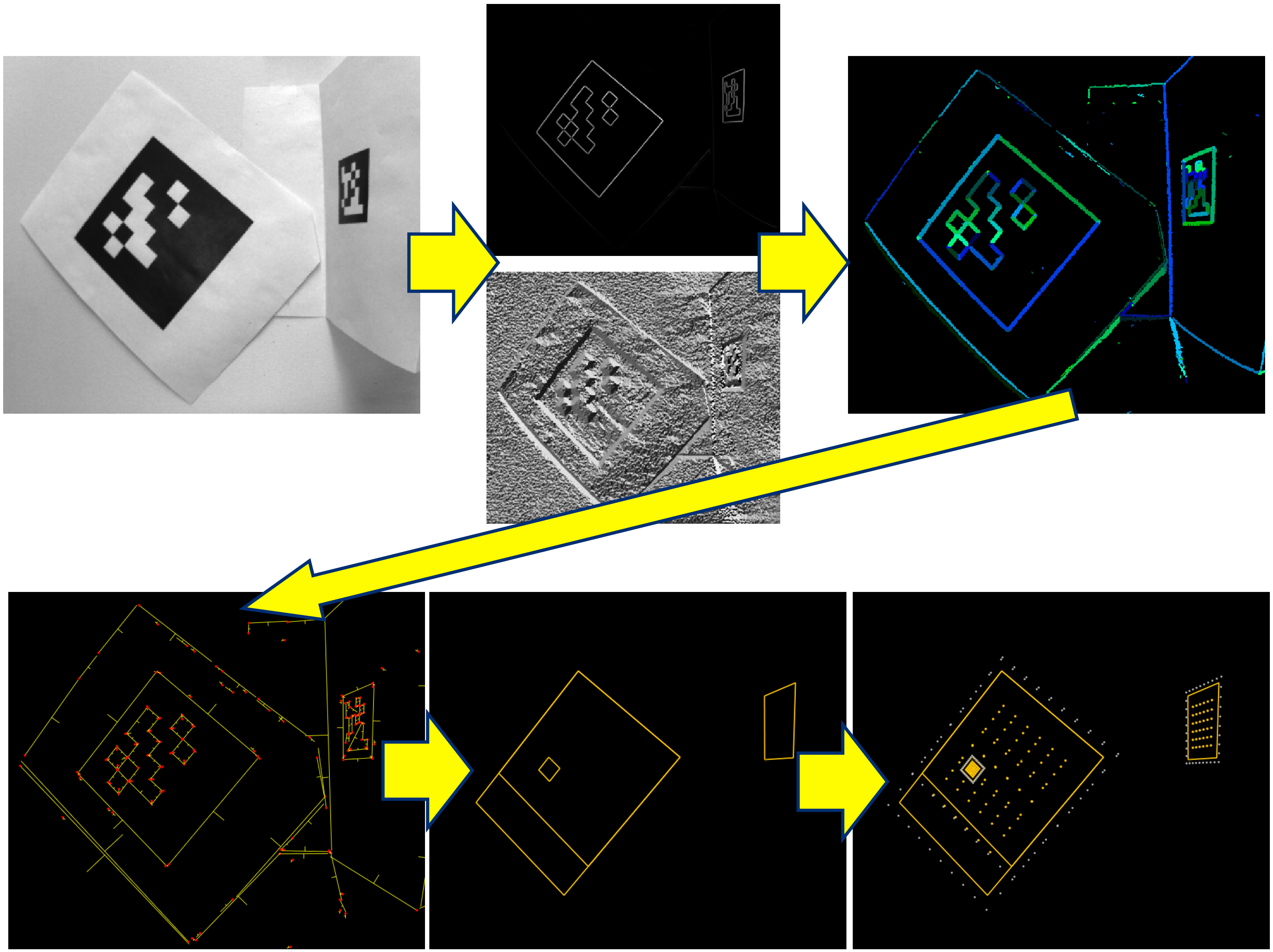
(a) Layout (b) ID=6 (6 bits) (c) ID=625518 (36 bits)

AprilTags

- Robust detection
  - Not based on threshold-based binarization scheme
  - Works better in unstructured environments
  - Accurate localization
- Strong coding system
  - Low false positive rate
- Parameterizable
  - Pick your own tag family



Detection Approach




id 0
(err=0, rot=3)