Forming and using patch dictionaries

D.A. Forsyth

University of Illinois at Urbana Champaign

Forming and using patch dictionaries

- Idea:
  - other patches in an image are good for denoising
  - why not use other patches in other images?

- Issues:
  - how to find the right patch in a very large number of patches?
  - redundancy
  - many patches are like one another, and so just make work

Desirable outcome

- Procedure to
  - accept a patch
  - produce a patch/some patches that are similar
- Built out of very large number of patches

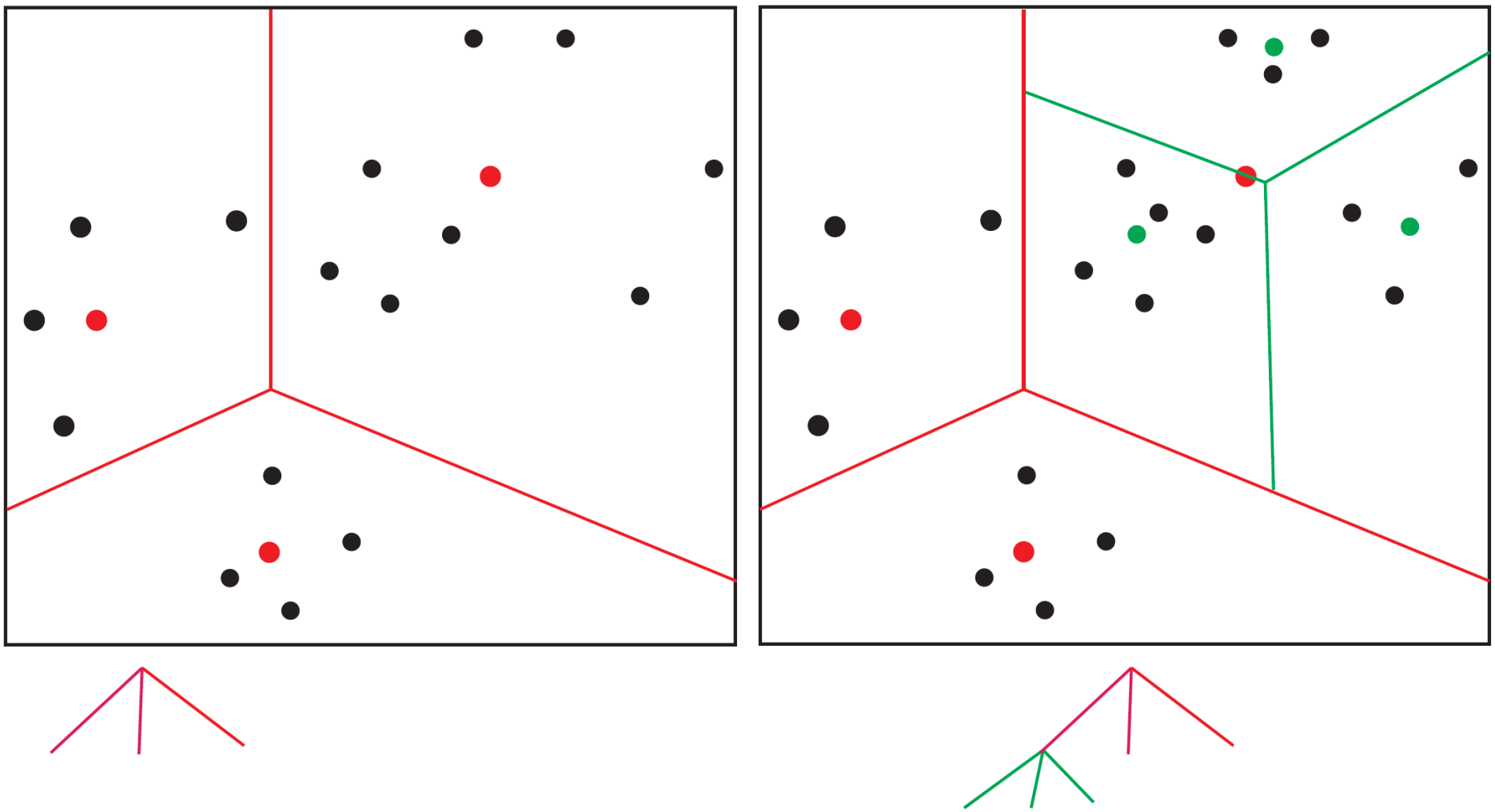
Why K-means is useful

- Take input patch
- Find its cluster center
- Use patches in the cluster or the cluster center itself to denoise the image

Hierarchical K means

- But what if there are too many data points?
  - too many clusters, finding cluster center is hard

- IDEA:
  - Cluster data using K means
  - Each cluster is now a dataset
  - Cluster each dataset using K means



- Notice this is a simple tree; you could make it deeper.

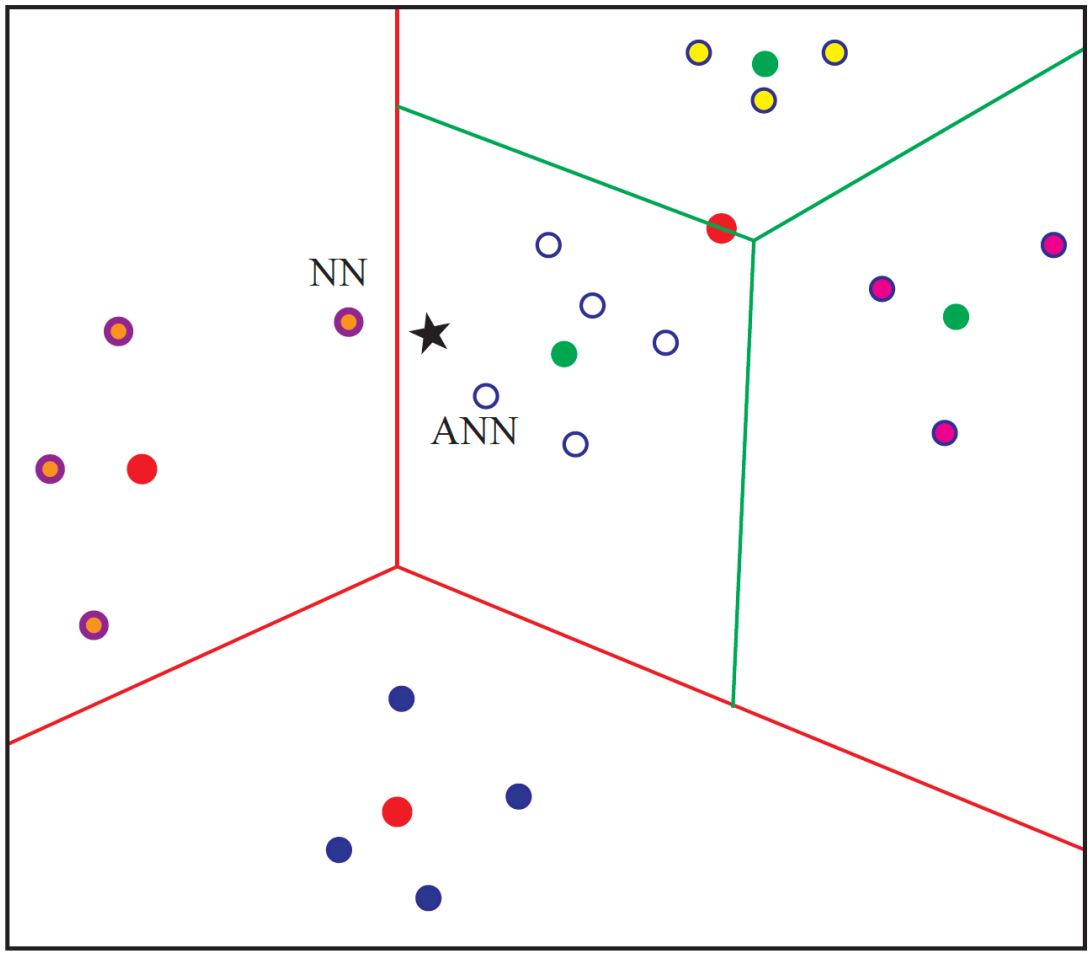
Nearest neighbors

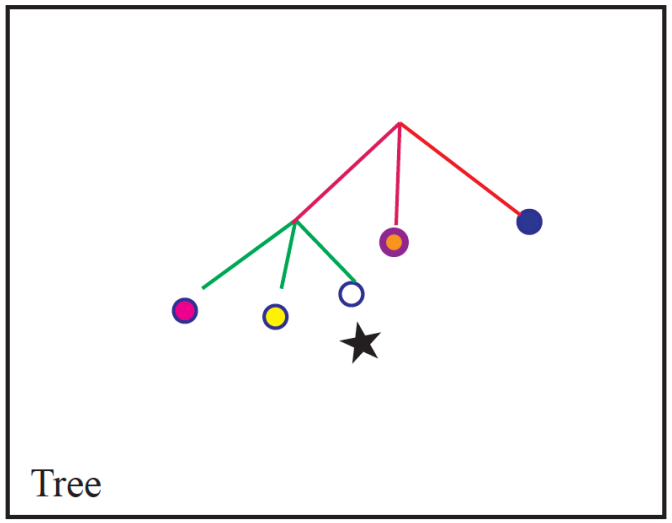
- Idea:
  - Find the patch that is closest to query patch, denoise (etc) with that
- Issue:
  - How to do this fast? (v. hard)

Approximate nearest neighbors

- Idea:
  - “Walk” input patch down hierarchical k-means tree
  - Find closest patch from patches in the leaf cluster
  - Use that instead of nearest neighbor

ANN and k-means



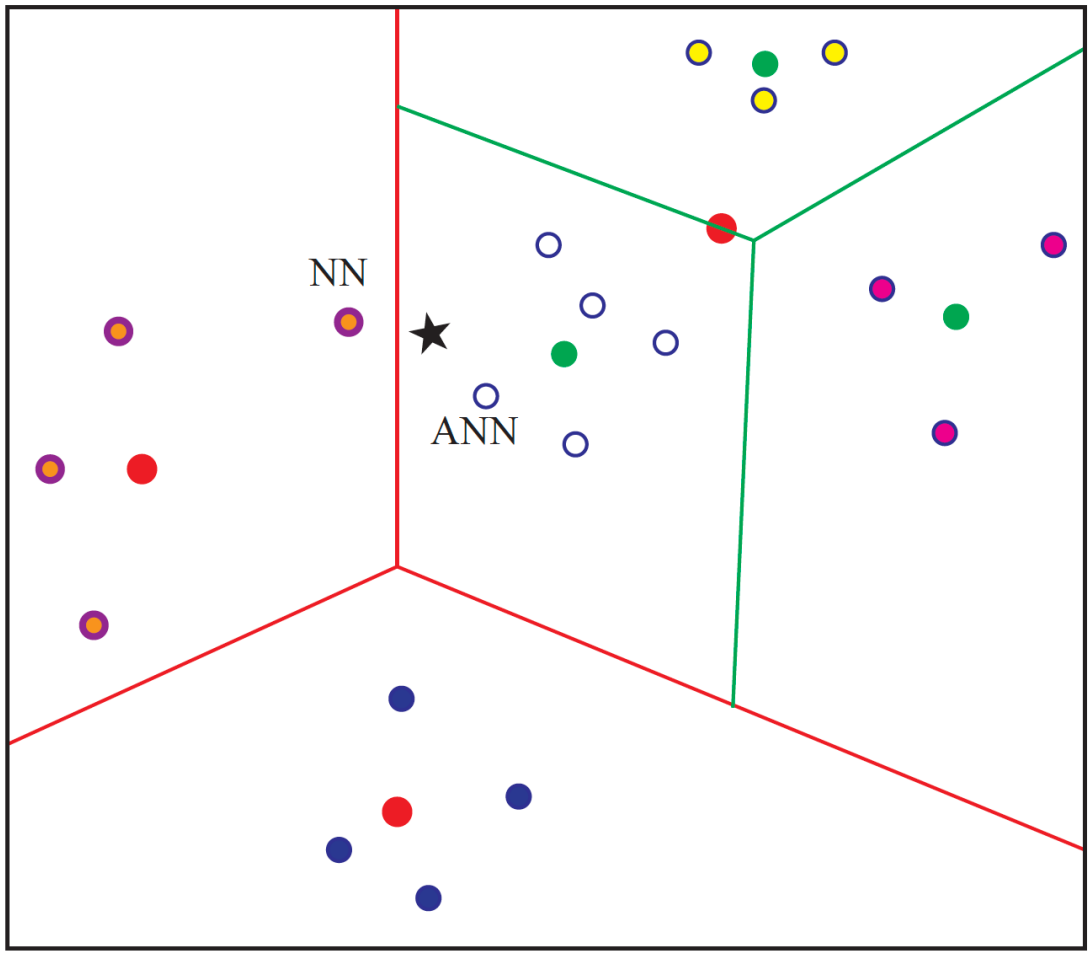


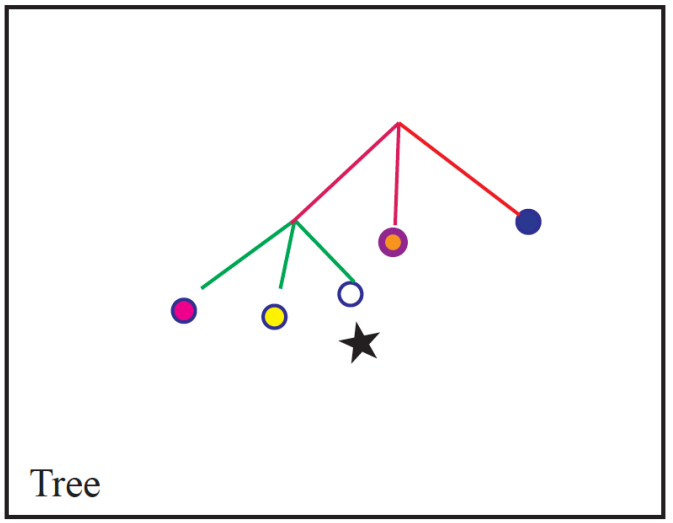
Query ★

ANN and k-means

- Idea:
  - “Walk” input patch down hierarchical k-means tree
  - Find closest patch from patches in the leaf cluster
  - Use that instead of nearest neighbor
- Fact: not nearest neighbor; with high probability nearly as close
- Fact: hard to find a closer neighbor by walking tree

ANN and k means





Query ★

Denoising with a patch dictionary

- Procedure:
  - Divide noisy image into patches, which could overlap
  - Match each patch using a dictionary (ANN using HKM)
  - Reconstruct
    - if the patches don’t overlap, easy
    - if they do, average

Options when matching

- Walk tree, find patch in leaf that is closest to query patch

- Walk tree, use mean of all patches in leaf
  - - this should be better than you might think
    - cause all the patches should be quite similar

Denoising with a patch dictionary - overlapping

Original

Patch (31.2) Mean (24.5)

3x3, dictionary 1

Patch (25.2) Mean (21.0)

5x5, dictionary 1

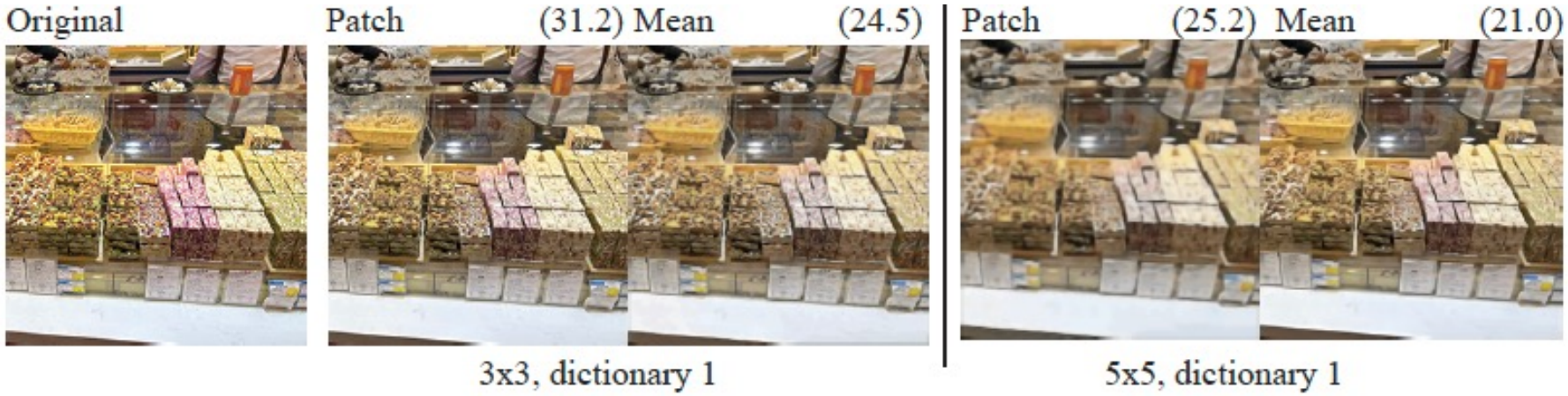


FIGURE 10.4: *Patch based reconstructions using patches from large image dictionaries can be very successful. On the* **left**, *the original image. Others show reconstructions using the approximate nearest neighbor (***Patch***) or the mean of the query cell (***Mean***) using patches of the size indicated. For these reconstructions, the patches overlap and are averaged (the stride is always* 1*). The PSNR of the reconstruction is shown in parentheses for reference. In each case, the tree contained 1e7 patches, taken from approximately 200, 000 images in the ImageNet test dataset (Section* ??*), and contains no patches from this image.* Image credit: *Figure shows my photograph of a sweetshop in Beijing.*

Denoising with a patch dictionary, no overlap

Original

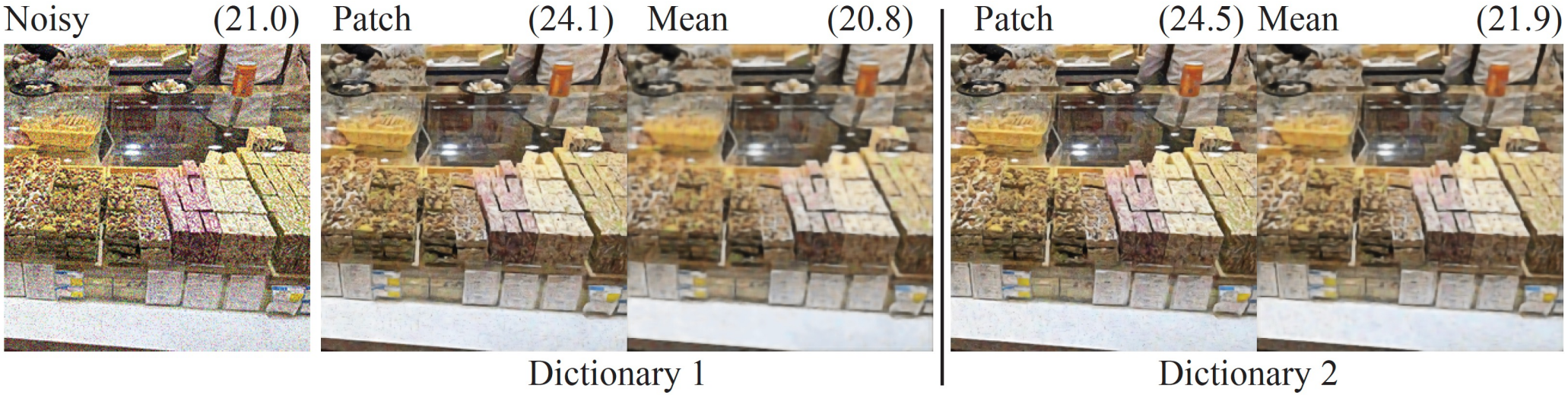
Patch (26.6) Mean (21.6)

3x3, dictionary 1

Patch (22.2) Mean (19.0)

5x5, dictionary 1

Dictionary size has a significant effect



- Dictionary 1 – 1e7 patches, 200, 000 images, 2000 leaves
- Dictionary 2 – 5 e 7 patches, 1e6 images

Vector quantization

- Walk tree, use mean of all patches in leaf
  - - this should be better than you might think
    - cause all the patches should be quite similar

- Notice that you have built a mapping. You could replace the mean with the number of the leaf to get: patch-> leaf

- This has a range of uses – compression, etc.

Building features with VQ

Here is a somewhat older construction that remains useful because ti can be so widely applied. Choose a patch size and some appropriate number of cluster centers. Represent the signal by cutting it into patches of fixed size, and turn these into vectors. Cluster a large set of training vectors. Now use the resulting set of cluster centers to represent a test signal by first vector quantizing each patch in the test signal, then building a histogram of the cluster centers. This histogram has fixed size (the number of cluster centers), and so can be used in a linear classifier.

- Image might have variable size

- Histogram has fixed size

VQ and voting

- Problem:
- you want to classify an image

- Strategy:
- learning:

  get many labelled images

  cut into patches

  build tree

  for each leaf, record the most common label

- classifying:

  cut image into patches
  pass patch down tree
  vote for label

Now old-fashioned, but moderately effective and very good in its time

Denoising from weird noise

- Imagine you have very strange noise, and you can simulate it
- You want to denoise

- Strategy:
  - Take many images, and create (noisy – clean) pairs
  - Carve into patches and build a HKM tree using only the noisy patches
    - (but keep the clean patch – so each data item is (noisy – clean) but the clusters, etc are formed using noisy alone)
- To denoise:
  - carve up noisy image into patches
  - walk the tree with the noisy patches
  - at leaf, substitute the clean version of the best match

Image to image mapping

- You could do:
  - image to edge map (fairly well)
  - image to normal map (rather badly)
  - quantized image to image
- You'd have a lot of trouble with:
  - edge map to image
  - image to depth map