Lecture 3: Data Structures for Image Analysis

Reading

SH&B, Chapter 3

3.1 Introduction

If you're going to analyze the content of an image, you first have to develop ways of representing image information.

3.2 Image Maps

The simplest way to store information about an image is on a per-pixel basis. The idea is to build a one-to-one mapping between information and pixels. Of course, the easiest way to do this is with other images of the same size. So, while you may have an original image that you're trying to analyze, you may also have any number of other images that store information about the corresponding pixel in the original image. Such information may include

- region labels (which object does this picture belong to)
- local geometric information (derivatives, etc.)
- distances (the distance maps we saw in the last lecture)
- etc.

3.3 Chains

One step up from image maps is to store information on a per-region basis. The most basic thing to store about a region is where it is. One could store the location of each pixel in the region, but one can be more efficient by simply storing the border. By traversing the border in a pre-defined direction around the region, one can build a *chain*.

Instead of encoding the actual pixel locations, we really need to only encode their relative relationships: a *chain code*. For 4-connected borders, we can do this with only two bits per pixel. For 8-connected borders, we need three bits per pixel. An example of a chain code is shown in Figure 3.1 of your text.

Later on, we'll talk about other ways of representing chains and using these encodings to extract shape object information.

3.4 Run-Length Encoding

In CS 450, you probably talked about run-length encoding as a way of compressing image content. We can also use run-length encoding to store regions (or images, or image maps). An example run-length encoding is given in Figure 3.2 of your text.

3.5 Hierarchical Image Structures

Another way of storing image information is hierarchically. There are many ways to do this, but we'll talk about two methods here: pyramids and trees.

3.5.1 Pyramids

An image pyramid is a hierarchy of successively lower-resolution images based on the original image. Each stage in building the pyramid involves the following two steps:

1. Blur the image by some resolution-reducing kernel (low-pass filtering).

2. Because the image has now been low-pass filtered, one can reduce the sampling rate accordingly.

Notice that there are no constraints on the type of low-pass filtering done or the reduction in sampling rate other than the contraint imposed by the sampling theorem.

The most common way of building a pyramid is to do a 2-to-1 reduction in the sampling rate. This makes things convenient for storing the image, mapping lower-resolution pixels to higher-resolution pixels, etc. The simplest way to blur the image so as to reduce the resolution by a factor of 2 is with a 2×2 uniform kernel. Done this way, each pixel in the pyramid is simply the average of the corresponding 2×2 region in the next lower-resolution image.

The problem with this type of reduction is that the "footprints" of the lower-resolution pixels don't overlap in the higher-resolution image. This means that this upper levels of the pyramid can be extremely sensitive to single-pixel shifts in the original image. Ideally, we want to build pyramidal representations that are invariant to such translation.

It's usually better then to build pyramids by using footprints that overlap spatially in the original image. One can use small triangles of finite extent, Gaussians, etc.

Pyramids can be used to greatly speed up analysis algorithms. The provide a "divide and conquer" approach to vision algorithms. We'll see various examples of such pyramidal algorithms as the semester progresses.

3.5.2 Trees

Another way of representing images it to store large, coherent regions using a single piece of data. One example of this is the *quadtree*. A quadtree is built by breaking the image into four equal-size pieces. If any one of these pieces is homogeneous (in whatever property you're using the tree to represent), don't subdivide it any further. If the region isn't homogeneous, split it into four equal-size subregions and repeat the process. The resulting representation is a tree of degree 4 where the root of the tree is the entire image, the four child nodes are the four initial subregions, their children (if any) are their subregions, etc. The leaves of the tree correspond to homogeneous regions. An example of a quadtree is shown in Figure 3.6 of your text.

Quadtrees have been used for compression, rapid searching, or other applications where it is useful to stop processing homogeneous regions.

Notice that although built differently (bottom-up vs. top-down), reduce-by-two pyramids and quadtrees are almost the same. One just uses images and the other uses trees. Pyramids have the advantage that it is easy to do spatial operations at any level of the hierarchy; trees have the advantage that they don't store redundant information for homogeneous regions.

3.6 Relation Graphs

Once you've segmented an image into regions that (you hope) correspond to objects, you may want to know the spatial relationships between the regions. By recording which regions are next to which other regions, you can build a graph that describes these relationships. Such a graph is called a *region adjacency graph*. An example such a graph is in Figure 3.3 of your text. Notice that nodes of degree 1 are inside the region that they're adjacent to. (It's common to use a node to represent everything outside the image.)

3.7 Co-occurrence Matrices

Suppose that you want to record how often certain transitions occur as you go from one pixel to another. Define a spatial relationship r such as "to the left of", "above", etc. The co-occurrence matrix C_r for this relationship r counts the number of times that a pixel with value i occurs with relationship r with a pixel with value j. Co-occurrence matrices are mainly used to describe region texture (and we'll come back to them then), but they can also be used on image maps to measure how often pixels with certain labels occur with certain relationships to other labels.

Vocabulary

- image map
- chain
- run length encoding
- pyramid
- quadtree
- region adjacency graphs