# 36-350: Data Mining

**Handout 7**
**September 17, 2003**

---

Some uses of data partitioning

**Image database browsing**—Partitioning a large set of images into groups that a user can browse down into.

In the paper by Chen et al., the images are represented by color counts, texture counts, and edge counts. "Texture" is represented by how different a pixel is from its neighbors. Thus areas of uniform color have zero "texture" and a wavy pattern would have high "texture". If the difference is high along one direction only, then the pixel is considered an "edge" and the angle of the edge is counted. These measures are insensitive to shifting objects in the image but they are sensitive to scaling and rotation.

The images are clustered into a binary tree by a variant of Ward's method. Some of the interior nodes in this tree are then removed to make it a quadtree, where each node has four children instead of two. The top level of the tree is laid out on the screen, with one image representing each group (the image closest to the mean of the group). When the user clicks on an image, it expands into four images representing the four subgroups. The images are automatically arranged so that similar images are nearby on the display.

The Chen et al. paper also has a nice discussion of various clustering methods, including K-means and Ward's (section 3).

**Image segmentation**—Partitioning an image into regions of similar color. This allows foreground/background and various objects to be separated, so that we can search for parts of images instead of entire images. Each pixel is a vector of 3 numbers (RGB or some other representation). Similar pixels will be close in Euclidean distance, therefore we can use Ward's method to cluster them. Ward's method even gives a hierarchy of possible segmentations.

A drawback of Ward's method for image segmentation is that pixels in the same "color cluster" may be far apart in the image. In the paper by Nock, Ward's method is modified so that pixels cannot be merged unless they are adjacent in the image. This forces all clusters to be spatially contiguous.

**Video segmentation**—Partitioning a stream of video into similar shots. A video can be considered a **volume** of pixels, organized in space $(x, y)$ and time $t$. The video is initially divided into **cuts**, when it abruptly switches to another scene or viewpoint. The sequence of images between two cuts is called a **shot**.

In the paper by Ngo et al., a shot is described by color counts and motion counts. Colors are quantized into 64 bins, and all pixels in the shot are counted. The motion of each pixel is also computed, by matching pixels in consecutive frames. Motion is a 2D vector. The motions are quantized into 10 directions, and counted across the shot. The counts for each shot are normalized and then compared by Euclidean distance.

Shots are then clustered hierarchically, in a top-down scheme. First all shots are clustered by K-means based on color only. This automatically partitions the basketball video into audience shots, player close-ups, full court shots, and the logo sequence. The shots in each color group are further partitioned with K-means on the motion counts. This automatically separates four different kinds of shots: still camera/still players, still camera/moving players, camera panning across still players, and camera tracking moving players. With these divisions, it is straightforward to find all penalty shots, shots of the coach, etc.

For the soccer video, the top level partitions it into shots of team A, shots of team B, overhead views, medium shots, the sky, and the intro sequence. The audience wears the same color as their team, so shots of the audience are not separated at the top level. The second level divides the audience and player shots, and by camera/player motion as in basketball. With these divisions, it is straightforward to find close-ups of goal kicks, for example.

In shots with a lot of motion, foreground and background can be automatically separated by finding the most common motion vector in an image. All pixels with this motion are likely to be background; the rest are foreground. This crude separation can be cleaned up by counting the colors in the background and foreground pixels and iteratively re-classifying the pixels based on color and motion. The player in the foreground can then be automatically classified as team A or team B based on color.

## References

[1] Jau-Yuen Chen, Charles A. Bouman, and John C. Dalton. Hierarchical Browsing and Search of Large Image Databases. *IEEE Transactions on Image Processing* 9(3): 442-455, March 2000. `http://citeseer.nj.nec.com/175002.html`

[2] Richard Nock. Fast and Reliable Color Region Merging Inspired by Decision Tree Pruning. *Proceedings of CVPR*, 2001. `http://www.univ-ag.fr/ rnock/Articles/CVPR01/`

[3] Chong-Wah Ngo, Ting-Chuen Pong, Hong-Jiang Zhang. On Clustering and Retrieval of Video Shots. *ACM Multimedia*: 51–60, 2001. `http://citeseer.nj.nec.com/503450.html`