Clustering Images

John Burkardt (ARC/ICAM)
Virginia Tech

.....

Math/CS 4414:
"Clustering Images"
http://people.sc.fsu.edu/~jburkardt/presentations/
clustering_images.pdf

.....

ARC: Advanced Research Computing
ICAM: Interdisciplinary Center for Applied Mathematics

23 September 2009

VirginiaTech

Clustering Images

- **Chain Letter Clustering Results**
- What's in an Image File?
- Working with an Image
- Clustering Colors
- Clustering Functions

VirginiaTech

Chain Letter Clustering Results

I collected the estimates for the distances between pairs of chain letters into a single distance matrix. Some rows were estimated by more than one person, and row A was *very* popular.

As I was entering the data, more was submitted, so I decided that I needed to set this calculation up in a way that would be easy to correct and modify...you guess it, more MATLAB!

I needed a **dist** matrix, and a **count** vector which would keep track of how many people estimated the same row, so I could average.

I thought it would also be good to average the matrix and its transpose, so that it was symmetric.

VirginiaTech

Chain Letter Clustering Results: Accumulating

```
a=1;b=2;c=3;d=4;e=5;f=6;g=7;h=8;i=9;j=10;k=11;
dist = zeros(11,11);
count = zeros(11,1);
count(g) = count(g) + 1;
dist(g,1:11) = dist(g,1:11) ...
  + [ 12, 18, 14, 12, 16, 11,  0, 16, 11,  9, 12 ];
count(a) = count(a) + 1;
dist(a,1:11) = dist(a,1:11) ...
  + [  0,  9,  9, 10,  9,  7, 16,  9, 14, 13, 11 ];
count(c) = count(c) + 1;
dist(c,1:11) = dist(c,1:11) ...
  + [ 12,  9,  0, 11,  9,  7, 17, 11, 14, 15, 11 ];
 (and so on)
```

VirginiaTech

Chain Letter Clustering Results: Cleaning up

```
% Average the rows
  for row = a : k
    if ( 0 < count(row) )
      dist(row,a:k) = dist(row,a:k) / count(row);
    end
  end
% Make the matrix symmetric
% (I could have done this with one command.  What is it?)
  for row = a : k
    for col = a : row - 1
      rc = dist(row,col);
      cr = dist(col,row);
      dist(row,col) = ( rc + cr ) / 2.0;
      dist(col,row) = ( rc + cr ) / 2.0;
    end
```

VirginiaTech

Chain Letter Clustering Results: The Table

	A	B	C	D	E	F	G	H	I	J	K
A	0	9	9	9	7	7	13	10	14	12	9
B	9	0	9	7	10	6	18	11	15	12	8
C	9	9	0	8	9	7	15	11	14	14	9
D	9	7	8	0	8	7	14	9	14	13	0
E	7	10	9	8	0	9	16	11	13	13	8
F	7	6	7	7	9	0	14	11	12	12	8
G	13	18	15	14	16	14	0	16	12	11	13
H	10	11	11	9	11	11	16	0	15	16	9
I	14	15	14	14	13	12	12	15	0	9	12
J	12	12	14	13	13	12	11	16	9	0	11
K	9	8	9	0	8	8	13	9	12	11	0

VirginiaTech

Chain Letter Clustering Results: Observations

By chance, the chain letters are grouped in such a way that the distance matrix suggest that G, H, I and J are "far away" from the rest of the letters.

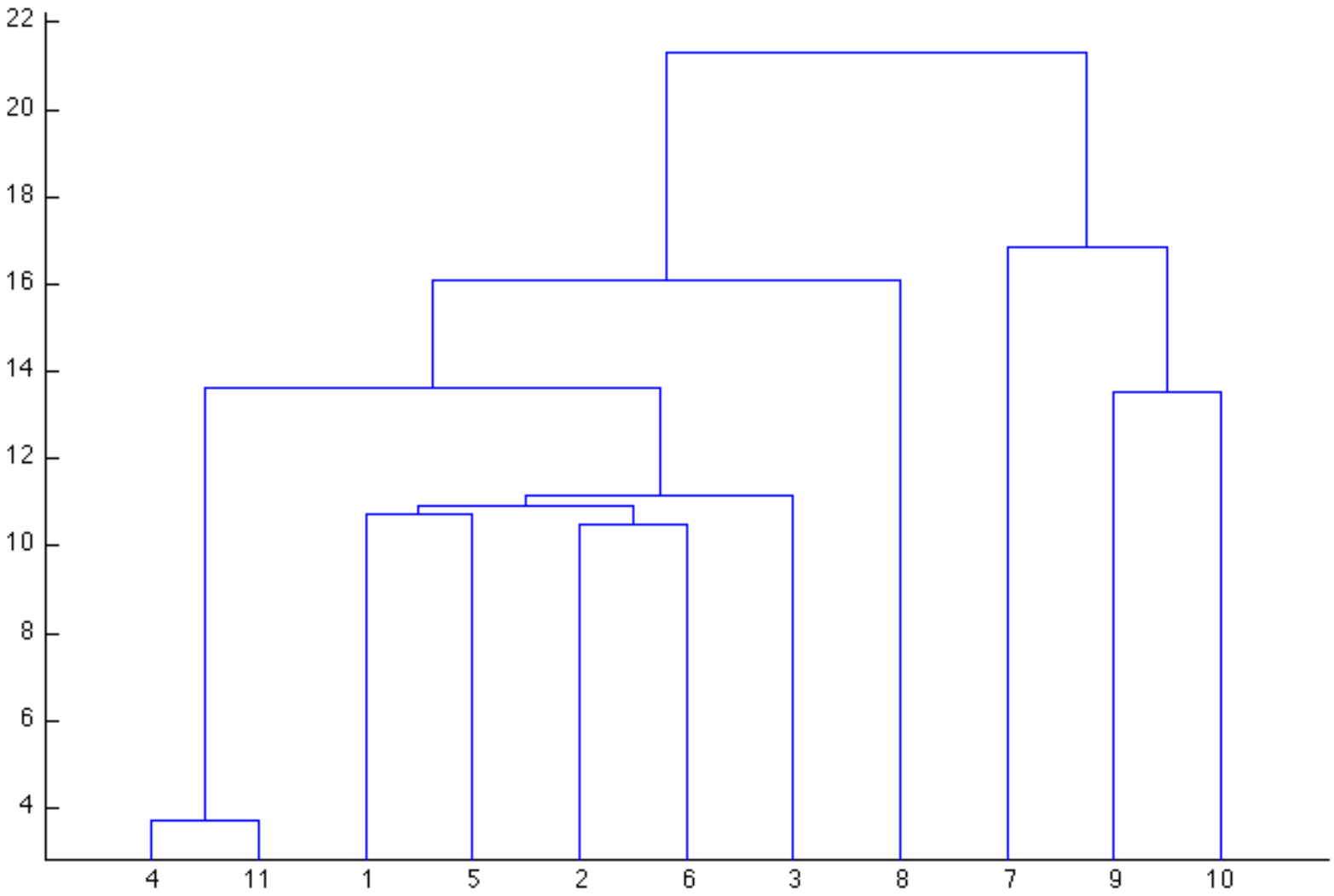
If you are patient, you will also notice that D and K are apparently 0 units apart, and, correspondingly, their rows are almost identical.

But to get any more information, we need to use MATLAB:

```
dv = pdist ( dist );
dl = linkage ( dv, 'single' );
dendrogram ( dl )
```

VirginiaTech

Chain Letter Clustering Results: Dendrogram



Clustering Images

- Chain Letter Clustering Results
- **What's in an Image File?**
- Working with an Image
- Clustering Colors
- Clustering Functions

VirginiaTech

What's in an Image File?

It's easy to look at a JPG file, but it's hard to figure out exactly what is going on and how the bits in the file turn into a picture.

A simple model of a digital image would assume that it mostly contains information about the color of each pixel.

It must also contain the height and width of the image in pixels.

We don't know the order of the data (left to right, probably, but is it top to bottom or the other way?)

In the simplest case, color is assumed to be a triple (R,G,B), with each value an unsigned integer between 0 and 255. (0,0,0) is black, (255,255,255) is white.

This means that we have, theoretically 256x256x256 = 16 million colors available.

VirginiaTech

What's in an Image File? - Dealing With Color...or Not.

Color is a complicated subject for many reasons. We can make our life simpler by skipping color for now.

Luckily, there are many "black and white" images available. Actually, these are **gray scale** images. The color of each pixel can be described by a single number, which again we can assume goes from 0 to 255. Now we just have 256 colors to worry about, and more importantly, they are linearly ordered, and easy to understand!

A gray scale image can be stored as a color image. To be sure we really just have one color value, we can look for graphics files in the **PGM** format ("**P**ortable **G**ray **M**ap").

PGM files come in two formats: binary and ASCII.

VirginiaTech

What's in an Image File? - What's in an ASCII PGM File?

An example we will work with is called *gator.pgm*. The image is 633 rows by 621 columns, a total of 393,093 pixels.

The ASCII version of this file takes 1,833,666 bytes. (Between 4 and 5 times the number of pixels. Can you explain why?)

We can print out the beginning of the file:

```
P2
# gator.ascii.pgm created by PGMA_IO::PGMA_WRITE.
633  621
255
200  200  198  198  198  198  200  200  197  197  196  196
196  197  198  198  195  183  216  225  216  222  212  208
210  213  217  218  217  216  216  216  212  214  214  212
209  209  213  218  215  213  211  210  210  211  213  214
```

What's in an Image File? - What's in a Binary PGM File?

The binary version of the file, *gator.pgm* takes 393,131 bytes.

That's almost exactly the number of pixels. Since it's so close to "perfection", where did those extra 38 bytes come from?

You can peek into the beginning of binary file. It actually begins with a very short bit of ASCII text. If we count these characters, including carriage returns, we get:

```
 3  "P5(cr)"
23  "#Created with The GIMP(cr)"
 8  "633 621(cr)"
 4  "255(cr)"
--
38 bytes
```

VirginiaTech

What's in an Image File?: Junior the Alligator

Here is the black and white picture we will be looking at:

VirginiaTech

Clustering Images

- Chain Letter Clustering Results
- What's in an Image File?
- **Working with an Image**
- Clustering Colors
- Clustering Functions

VirginiaTech

Working with an Image - Cracking the Shell

I hope I have convinced you that an image file is, in some sense, just a big "paint by numbers" table.

Now we're going to use MATLAB to crack open the file and look at the numbers.

MATLAB has a function called **imread()** which reads a graphics file, interprets it based on its file extension, and, if it's a PGM file, returns a single matrix of the gray scale values, that is, numbers between 0 and 255.

```
a = imread ( 'gator.pgm' );
```

VirginiaTech

Working with an Image - Cracking the Shell

Now **a** is just a MATLAB matrix. And one thing we can do with a matrix is ask for its dimensions:

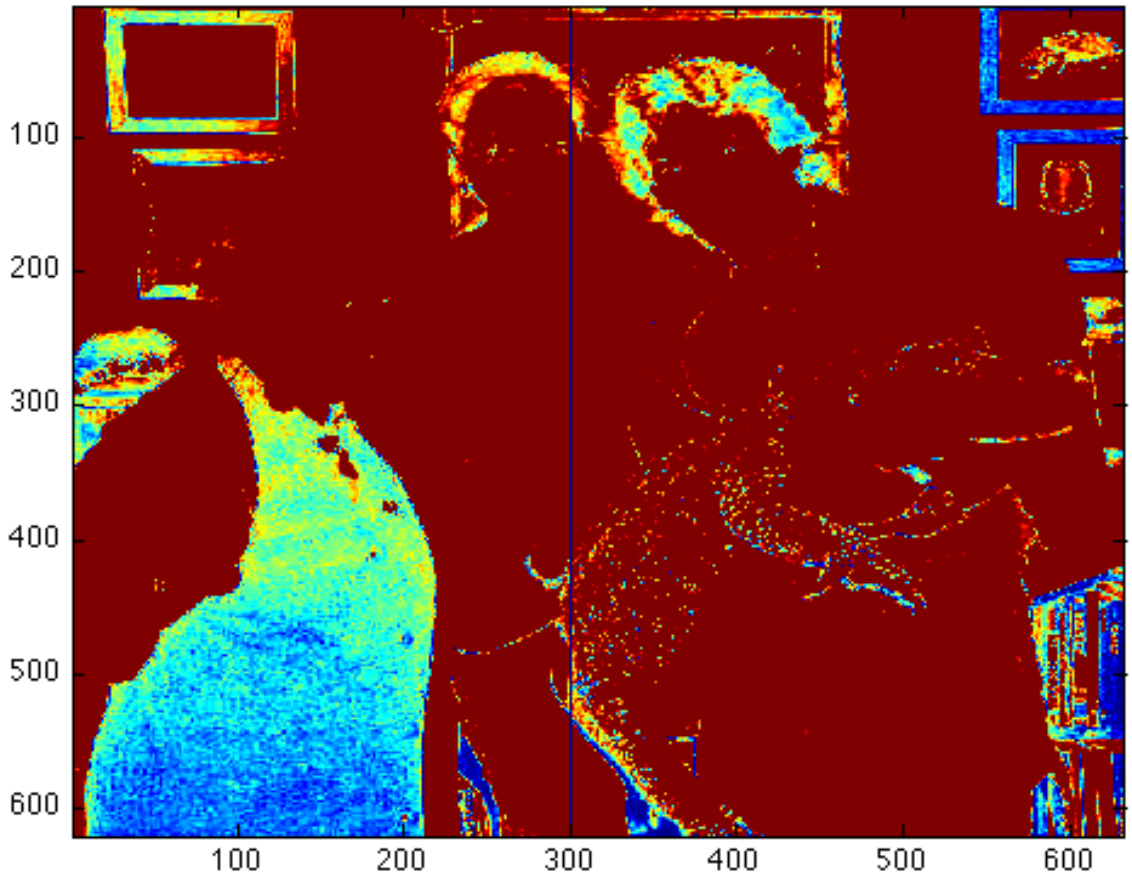
```
size ( a )
ans =

   621   633
```

So you see that MATLAB has reversed the storage (621 rows by 633 columns) that was used by the PGM file.

Working with an Image - Cracking the Shell

We should be able to have MATLAB display the image using the **image()** command, but when we do so, something horrible happens.



giniaTech

Working with an Image - Gray Colormap

The problem is that MATLAB has a default color map, which doesn't automatically change just because you read in a gray scale image. Let's notify MATLAB that for a while, we want to use grays only (you can undo this by typing

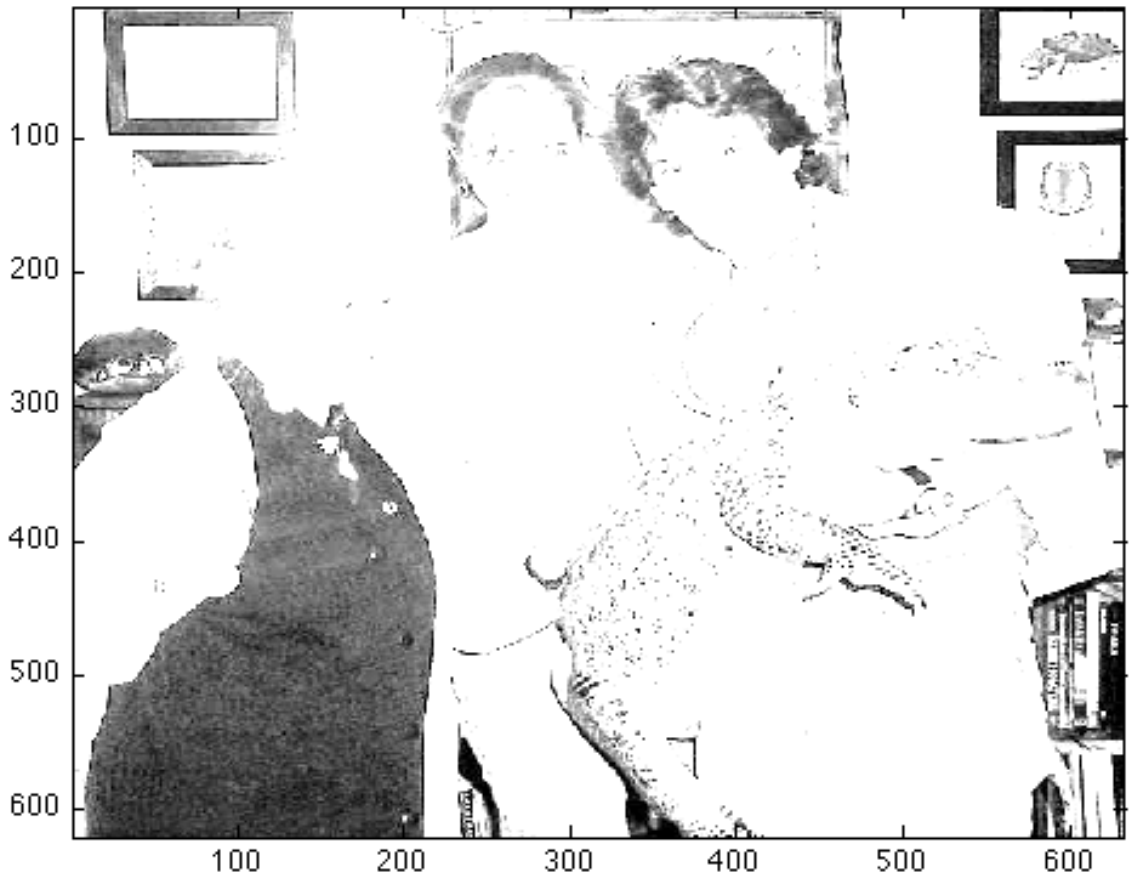
```
colormap ( 'gray' )
image ( a )
```

If you later want to do other plotting tasks, you might want to undo this colormap by the command **colormap ('default')**.

VirginiaTech

Working with an Image - The Gray Scale is Too Bright

Well, who turned on the bright lights here?



VirginiaTech

Working with an Image - "Turn Down" the Data

And now the problem is that MATLAB's gray scale is too bright. We can't change MATLAB, so we have to change our data, by darkening it.

Here, I'm essentially using the trick that a number between 0 and 1 gets smaller when you square it, even smaller if you cube it.

```
b = double ( a ) / 255;   <-- Make B between 0 and 1.
b = b^3;                  <-- The cubes are "darker".
c = uint8 ( 255 * b );    <-- Put darker grays into "c".
image ( c )               <-- Any better?
```

We have just done some "image processing"!

VirginiaTech

Working with an Image - A Picture We Can Recognize

VirginiaTech

Working with an Image - Messing Up the Data

Now I'm confused that the PGM data was 633 by 621, but the MATLAB array is 621 by 633. Does a column in MATLAB mean a vertical or horizontal strip of the picture?

Let's make a copy of **c**, change 6 "columns" to 0, and then display the new image.

```
d = c;
d(1:621,300:305) = 0;
image ( d )
```

VirginiaTech

Working with an Image - Messing up the Data

Now I have some confidence that array columns are picture columns!

giniaTech

Working with an Image - Image Analysis

Now let's consider whether this picture is suitable for clustering.

The first question we can ask is, how are the grays used in the picture? The easiest way to answer that is to take all the pixels as a single vector, and create a histogram.

We have to **reshape()** the **c** matrix into a vector to do this:

```
d = reshape ( c, 621*633, 1 );  <-- D is a vector.
hist ( d, 256 );                <-- Using 256 bins.
```

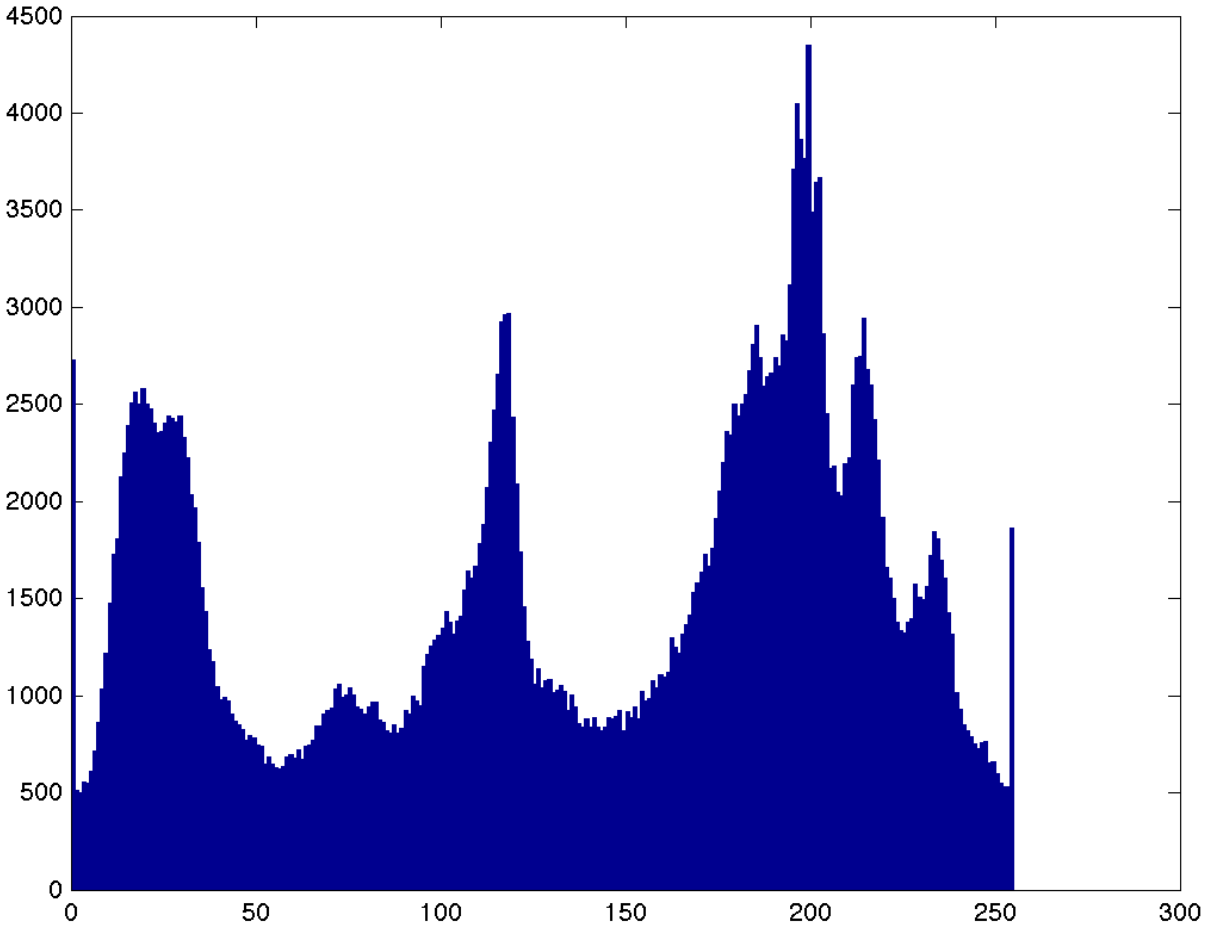
VirginiaTech

Clustering Images

- Chain Letter Clustering Results
- What's in an Image File?
- Working with an Image
- **Clustering Colors**
- Clustering Functions

VirginiaTech

Clustering Colors: Color Usage Histogram



Clustering Colors - Image Analysis

The histogram suggests, by its 3 big peaks, that there is a lot of information in the picture that is associated with three colors.

It almost seems worth trying to use just 3 colors for the image. This would be the next best thing to a true black and white image!

It's easy to pick off 3 good colors from the histogram, but then we have to somehow tell all the pixels which color they should change to.

It's very tempting to have the K-Means program do this for us!

VirginiaTech

Clustering Colors - Using K-Means

Suppose we asked K-Means to cluster the data into 3 groups. We'll need to use the vector **d** which contains the grays in one long vector. Actually, we'll also have to make a copy of **d** that is of type **double** so we can do arithmetic on it!

```
dim = 1;
n = 621 * 633;
k = 3;
e = double ( d );
[ c, ptoc ] = km ( dim, n, e, k );
```

VirginiaTech

Clustering Colors - Using K-Means

Once we have the cluster information, we just want to replace each pixel by the color of its cluster center. This is amazingly simple:

```
f = c(ptoc);
f = uint8 ( f );
f = reshape ( f, 621, 633 );
image ( f );
```

VirginiaTech

Clustering Colors - KM is TOO SLOW

Unfortunately, when I tried this, my **km.m** code ran forever. It works OK for small problems only. I did not optimize it. Luckily, MATLAB includes a program called **kmeans** which does the job. The usage is a little different:

```
k = 3;
e = double ( d );
[ ptoc, c ] = kmeans ( e, k );
```

VirginiaTech

Clustering Colors: Using Only 3 Grays

VirginiaTech

Clustering Colors - Using K-Means

Suppose we asked K-Means to cluster the data into 3 groups. We'll need to use the vector **d** which contains the grays in one long vector. Actually, we'll also have to make a copy of **d** that is of type **double** so we can do arithmetic on it!

```
dim = 1;
n = 621 * 633;
k = 3;
e = double ( d );
[ c, ptoc ] = km ( dim, n, e, k );
```

VirginiaTech

Clustering Colors - Saving the Image

The MATLAB Image Processing Toolbox includes a function **imwrite()**, which can take image data and save it to a file.

The file extension will tell MATLAB what kind of image file we want to write, including **bmp**, **jpg**, **pgm**, **png**, **tiff**.

```
imwrite ( e, 'gator_3grays.pgm' );
```

VirginiaTech

Clustering Colors - Looking Back

Today, we've gone through the steps necessary to work with an image.

We worked with a gray-scale image, to try to avoid some of the complications associated with color (so believe me, it could have been worse!).

In our example, we were able to take a 256 "color" picture and, using clustering, find 3 good colors (cluster centers) and a way to replace each pixel color by one of the 3 colors (cluster assignment), so that the processed image still was recognizable.

Along the way, I hope I showed you some of the surprises you can run into, and ways of trying to deal with them.

VirginiaTech

Clustering Images

- Chain Letter Clustering Results
- What's in an Image File?
- Working with an Image
- Clustering Colors
- **Clustering Functions**

VirginiaTech

Clustering Functions

When we compare a random clustering to a clustering produced by K-Means, we can tell that the second clustering is "better". But a computer program needs a numerical formula by which to determine if one clustering is better. For K-Means clustering, this numerical formula involves the *cluster variance* that we talked about last time.

It's possible to choose other formulas that evaluate a clustering. In Chapter 11, two functions are suggested, called $\mathcal{D}$ and $\mathcal{R}$. Let us investigate the definitions and uses of these other functions.

VirginiaTech

Clustering Functions - Distance to the Nearest Center

Each point $\mathbf{p_i}$ in a clustering wants to be as close as possible to the center of some cluster. From $\mathbf{p_i}$'s point of view, we should try to minimize the function $\mathbf{d_i}$:

$$\mathrm{d_i} = \min_{1<=j<=k} \|p_i - c_j\|$$

Of course, *every* point wants us to minimize its distance, so the function we would end up trying to minimize would be

$$\mathcal{D}1 = \sum_{i=1}^{n} \mathrm{d_i}$$

or, perhaps

$$\mathcal{D}2 = \sum_{i=1}^{n} \mathrm{d_i}^2$$

Clustering Functions - Distance to the Furthest Point

If we look at clustering from the cluster center's point of view, it wants all of the points it contains to be close to it. One way to measure that for a particular cluster $\mathbf{c_j}$ is to compute the cluster radius, that is, the distance to the furthest point in the cluster:

$$r_j = \max_{p \in c_j} \|p_i - c_j\|$$

Now every cluster center will want to minimize this quantity, so to do a good job of clustering, we would try to minimize

$$\mathcal{R}1 = \sum_{j=1}^{k} r_j$$

or, perhaps

$$\mathcal{R}2 = \sum_{j=1}^{k} {r_j}^2$$

Clustering Functions - Distance to the Furthest Point

The variance and the functions $\mathcal{D}1$, $\mathcal{D}2$, $\mathcal{R}1$, and $\mathcal{R}2$ are all reasonable functions to work with. In general, the optimal clustering for each function will be different, and it is up to the user to decide which function to use.

If the K-Means algorithm is used for clustering, then the variance is automatically the function that is minimized.

If a different clustering function is used, then the user must work with some kind of minimization program, such as MATLAB's **fminunc()**, to try to minimize the function, and thus produce a clustering.

VirginiaTech

Clustering Functions - Using FMINUNC

MATLAB's **fminunc()** can seek a number or vector **x** which minimizes a function **f(x)**.

The user supplies a starting estimate **x0** and the name of an M-file that evaluates the function to be minimized.

For the clustering problem, our input data will be the location of the centers, which we might call **c**. So calling **fminunc()** looks easy:

```
c0 = rand(dim,k);          <-- K centers, DIM dimensional.
c = fminunc ( @f, c0 );
```

VirginiaTech

Clustering Functions - Using FMINUNC

The tricky part involves writing the function **f.m**. Somehow, **f** needs to know the value of our data points **p**. Let's assume we can fix that somehow. Then **f** might look like this:

```
function d1 = f ( c )
  k = length ( c );
  [ dim, n ] = size ( p );
  ptoc = kmeans_update_clusters ( dim, n, p, k, c );
  d1 = 0.0;
  for i = 1 : n
    j = ptoc(i);
    d1 = d1 + norm ( p(1:dim,i) - c(1:dim,j) );
  end
  return
end
```

VirginiaTech

Clustering Functions - Using FMINUNC

We said that **f** needs to know the value of our data points **p**. It can't read them from a file every time **f** is called (well it could, but that would take forever). Instead, we can do some slightly dangerous programming and use **global** variables. This allows two MATLAB functions to share a variable without having to pass it explicitly. So our main program might look like this:

```
global p
p = rand(3,100);        <-- 100 points, 3 dimensional
c0 = rand(3,5);         <-- K centers, 3 dimensional.
c = fminunc ( @f, c0 );
```

and **f** would be modified to:

```
function d1 = f ( c )
  global p
```

VirginiaTech

Clustering Functions - Using FMINUNC

On Friday we may return to these issues of using **fminunc** to do our clustering by choosing a particular clustering function.

In particular, we can actually do a K-Means clustering by calling **fminunc**, if we come up with the correct clustering function.

VirginiaTech