HANDOUT’s OBJECTIVES:
• familiarize student with fundamental steps in image analysis
• introduce image-analysis terminology & its practical applications
• introduce student to basic image filtering and analysis techniques

3 steps in image analysis:
1) preprocessing raw data from satellite: removing systematic errors
2) image enhancement: removing random errors
3) image classification: pattern extraction

We draw these distinctions without hoping to make them rigorously for every satellite image. In fact, it’s more useful to think of these steps as simply the means to our end of extracting meaningful oceanographic data from satellite images.

We begin with some image analysis terminology: each element of a digital satellite picture is called a pixel (for “picture element”). Each pixel corresponds to the smallest resolvable element in the satellite image, such as this $n$-by-$m$ array of pixels for downtown Washington, DC.

Systems analogous to a satellite’s sensors are the photodetectors in the human eye and those in a TV camera. This analogy implies that a sensor element measures
radiances, not irradiances. In a detector designed to yield images, this must be the case. Name one theoretical and one practical reason why this must be so.

The satellite photodetectors receive radiances, but they do not produce radiance values directly. In digital imaging, the physically continuous quantity radiance must first be converted into discrete, discontinuous integers.

We make this jump from real (mathematically speaking) numbers to integers in order to accommodate the way that digital computers operate. In so doing, we lose some resolution in the data, but the payback is that computers can now assimilate the flood of numbers from the satellite.

Frequently, the range of radiances is separated into 256 gray levels, or integer divisions from 0 to 255. This resolution requires 8 bits (for “binary digits”) of data at each pixel, because $256 = 2^8$. These integer gray levels are also called DNs, for “digital numbers.”

How much information is contained in the modest 275-by-320 pixel image of Washington above? It’s $(275 \times 320)$ pixels $\times$ 8 bits/pixel $\sim 7.04 \times 10^5$ bits, or more than 22 million possible pixel combinations in a single monochrome image. (Remember that each 8-bit pixel can encode 256 gray levels, so that $275 \times 320 \times 256 \sim 22 \times 10^6$ possible images.) If we now use several monochrome images to define a color image, we will use at least 3 different spectral bands, thereby tripling our storage requirements.

It’s often convenient to define a particular number of bits per pixel, and we use the term byte to do so. In our example above, we were using 8-bit bytes per pixel. We can also use 8-bit bytes to define a set of characters, such as the English alphabet and Arabic numbers, with lots of extra room for special symbols.

Computer programs are usually stored in this way, and obviously we can store other kinds of text as well. Assuming that a each volume of a 26-volume Encyclopedia Britannica has 1200 pages, and that each page contains 1400 seven-letter words, how many 8-bit bytes would be needed to store the entire encyclopedia?

1) image preprocessing

   a) correcting systematic radiometric errors

   As noted above, real radiances are converted into an arbitrary number of integers. If we knew the details of that conversion perfectly at each image pixel, radiometric calibration would be simple: just measure the detector’s analog response to a known $L_\lambda$ (say, $R = \frac{10^{-4} \text{ volts}}{W \text{ m}^{-2} \text{ sr}^{-1} \text{ nm}^{-1}}$), and divide all future analog responses (in volts) by $R$ in order to get the new, unknown $L_\lambda$. 
To complete the translation, we need to specify how the analog response (in $\text{W m}^{-2} \text{sr}^{-1} \text{nm}^{-1}$) is translated into integer pixel values via a \textbf{lookup table} or LUT.

\begin{center}
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
\textbf{pixel values} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \ldots \\
\hline
\textbf{radiances} & 0.01 & 0.0135 & 0.015 & 0.02 & 0.022 & 0.026 & 0.0285 & \ldots \\
\textbf{(W m}^{-2} \text{sr}^{-1}\text{nm}^{-1}) & & & & & & & & \\
\hline
\end{tabular}
\end{center}

Although we can design a linear relationship between detector response and pixel values, the relationship between physically significant quantities (radiance here) and pixel values usually is \textit{not linear}. For example, doubling received radiances won’t, in general, double the corresponding pixel values in the received image.

In principle, we should also assume a different LUT at \textit{each} pixel, because $R$ can vary across a satellite’s array of sensor elements. In addition, $R$ will vary with sensor age, often in unknown ways. The solution to both problems is to periodically recalibrate the sensor array against a source of known radiances. What advantages would a sun-synchronous satellite have for solving this recalibration problem?

We earlier discussed a separate class of radiometric errors for nadir-viewing satellites attempting to measure oceanographic parameters: the variable transmissivity and absorptivity of the atmosphere. The histogram below illustrates how atmospheric scattering affects the downtown Washington Landsat scene shown above.

![Histogram of atmospheric scattering effects](image.png)

SO431 — Image analysis basics (10-29-2009)
In the graph above, Band 1 DNs (0.45-0.52 µm) form the right peak and Band 3 DNs (0.63-0.69 µm) form the left peak. At street level, Washington doesn’t look especially blue, so why does the satellite see generally higher reflectances in the blue?

b) *Correcting systematic geometric errors*

We specify pixel coordinates by image row and column; clearly these numeric abstractions don’t tell us anything about a scene’s physical dimensions. To get these, we must know the satellite sensor’s angular field-of-view (FOV), usually expressed in degrees, as well as the satellite’s altitude at any point in its orbit. FOV is usually a constant (although it need not be), but obviously altitude is not.

Many satellites use conventional lenses to focus scenes, and many of these lenses map radiances within their FOV onto a tangent plane. In tangent-plane mapping (the kind used in “normal” camera lenses), a surface feature of lateral size $dx$ seen from an altitude $A$ subtends an incremental FOV angle $d\theta$ according to:

$$d\theta = \arctan\left(\frac{dx}{A}\right), \quad \text{(Eq. 1)}$$

Now $d\theta \approx \frac{dx}{A}$ if $d\theta$ is small, as it will be for most pixels in a narrow FOV nadir-viewing satellite sensor array. However, if we look tens of degrees from the nadir, or look across a large $d\theta$, then $d\theta \approx \frac{dx}{A}$ may be a bad assumption.

For example, compare $d\theta = 1^\circ$ at the nadir and $25^\circ$ from it and note how $dx$ changes. If the satellite’s altitude $A$ is 400 km in the nadir direction $\psi (\psi = 0^\circ)$, it will be about $\frac{A}{\cos(25^\circ)} = 1.1A = 441$ km at $\psi = 25^\circ$. What happens to $dx$ here? If $d\theta = 1^\circ$, the change in the effective value of $A$ means that $dx$ increases from ~7 km for $\psi = 0^\circ$ to $dx = 7.7$ km at $\psi = 25^\circ$. The error in assuming that constant $d\theta$ means constant $dx$ is even worse if we take into account the sphericity of the earth.

Other problems of scene geometry arise when we try to piece together separate images into a mosaic, or when we compare images taken from different altitudes or vantage points. **Ground control points** (GCPs) are often used in mathematically transforming the slightly different scenes into a common cartographic frame of reference. GCPs can be either natural or manmade landmarks (such as a river confluence or a large structure) that are clearly visible in the image.

At a minimum, we assume that the slightly overlapping images can be made congruent by a combination of: 1) rotation, 2) differential x, y scaling, 3) translation.

By definition, oceonic GCPs are scarce, and so a satellite’s image’s location could be based on a knowledge of the satellite’s orbital parameters and the time when the image was acquired.
2) **image enhancement:** contrast modification and spatial filtering

   a) **contrast modification**

   Contrast in an image area is defined in many different ways, but each definition is a variation on the **radiance dynamic range** \((L_{\text{max}}/L_{\text{min}})\), or this ratio’s log). In principle, the simple contrast ratio will be undefined (or \(\infty\), depending on your taste) for \(L_{\text{min}} = 0\), so a better definition of **contrast** \(C\) is:

   \[
   C = \frac{L_{\text{max}} - L_{\text{min}}}{L_{\text{max}} + L_{\text{min}}}
   \]  

   (Eq. 2).

   Clearly we could substitute the corresponding maximum and minimum gray-level values \(D_{\text{Nmax}}\) and \(D_{\text{Nmin}}\) in Eq. 2, although in general this new \(C\) won’t be linearly related to \(C\) derived from radiances.

   Satellite sensors can be designed to accommodate a wide range of radiances, although the full dynamic range doesn’t occur in many scenes. So by definition, many scenes’ gray-level contrast will be low. In other words, because these scenes generate DNs that span only a fraction of the system’s radiance resolution, the difference between \(L_{\text{max}}\) and \(L_{\text{min}}\) will be relatively small.

   For mathematical purposes, low \(C\) is unimportant in an image. However, low \(C\) is important for humans, because their ability to see detail requires \(C \geq 0.02\). So we distinguish between **image** \(C\) and **display** \(C\), and we improve the latter by increasing the gray-level dynamic range throughout an image. To improve contrast, we write a LUT that maps a low-contrast image’s \(D_{\text{Nim}}\) into a new, wider range of display \(D_{\text{Ndis}}\). In the example below, \(60 \leq D_{\text{Nim}} \leq 160\) (range = 100) is linearly mapped into \(0 \leq D_{\text{Ndis}} \leq 255\), the entire dynamic range of an 8-bit display device.

![Diagram showing linear contrast stretch between \(D_{\text{Nim}}\) and \(D_{\text{Ndis}}\)](image)

\(D_{\text{Nim}} = 60 \rightarrow D_{\text{Ndis}} = 0\)

\(D_{\text{Nim}} = 160 \rightarrow D_{\text{Ndis}} = 255\)
At each target pixel, \[
C(L_{\text{target}}) = \frac{L_{\text{target}} - L_{\text{surr}}}{L_{\text{surr}}},
\]

\[
L_{\text{surr}} = \frac{L_1 + L_2 + L_3 + L_4 + L_6 + L_7 + L_8 + L_9}{8}.
\]
At each target pixel, 

\[ C(L_{\text{target}}) = \frac{L_{\text{target}} - L_{\text{surr}}}{L_{\text{surr}}} \],

\[ L_{\text{surr}} = \frac{L_1 + L_2 + L_3 + L_4 + L_6 + L_7 + L_8 + L_9}{8} \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L_1</td>
<td>L_2</td>
<td>L_3</td>
</tr>
<tr>
<td>L_4</td>
<td>L_{\text{target}}</td>
<td>L_6</td>
</tr>
<tr>
<td>L_7</td>
<td>L_8</td>
<td>L_9</td>
</tr>
</tbody>
</table>

**image contrast, Morro Bay, CA**

{19 Nov 1984, bands 1-3}

- **Red line**: fogged Morro Bay contrast
- **Blue line**: original Morro Bay contrast

Surf zone

Offshore water

Nominal threshold contrast
Another way of changing contrast is histogram equalization. Here we transform the frequency distribution of displayed pixels \( f(DN) \) so that \textit{in principle} \( f(DN) \) is constant at all DN. Histogram equalization’s advantage is that contrast is increased for the largest number of pixels throughout the entire image. However, we may not want this “best average” contrast throughout the image, instead preferring poor contrast in some areas and much better contrast in others.

To equalize an image’s histogram we first calculate its cumulative histogram; \textit{i.e.}, we form a new histogram whose entries are the total \( f(DN) \) at all gray levels below the current one. The cumulative image histogram (see curve with circles) can be saved in a 256-element array we’ll call \( CH \). By definition, \( CH \)’s \( f(DN) \) values always increase or stay the same with increasing DN.

For the image analyzed above, \( CH \)’s maximum value \( X = 364080 \), and we create a DN transfer function by multiplying \( CH \) by the scaling factor \( 255/X \) (here, this factor = 0.0007). This scaled version of \( CH \) is our transfer function or DN LUT: in scaled \( CH \)’s column 53 (\( DN_{im} = 53 \)), the entry is \( DN_{dis} = 16 \) (see table below). Thus the original image’s \( DN_{im} = 53 \) are all replaced by \( DN_{dis} = 16 \), \( DN_{im} = 59 \) are replaced by \( DN_{dis} = 28 \), etc. What does this mapping do to \( \Delta DN \) in the transformed image?

<table>
<thead>
<tr>
<th>( DN_{im} ) (column # in ( CH ))</th>
<th>53</th>
<th>54</th>
<th>55</th>
<th>56</th>
<th>57</th>
<th>58</th>
<th>59</th>
<th>60</th>
<th>61</th>
</tr>
</thead>
<tbody>
<tr>
<td>( DN_{dis} ) (scaled ( CH ) value)</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>25</td>
<td>28</td>
<td>31</td>
<td>34</td>
</tr>
</tbody>
</table>

State College, PA

clear sky image (Feb. 1987)
Note that the continuous run of $DN_{im}$ integers is replaced by the discontinuous run of $DN_{dis}$. In general, equalized images have histograms with many gaps (for clarity, gaps are omitted in the graph above), and those gaps are due to: (1) aliasing caused by rounding in the scaled CH, (2) mapping a (possibly) limited range of $DN_{im}$ values into the range $DN_{dis} = 0 \rightarrow 255$.

b) random noise suppression

Many factors can cause errors in moving from real radiances to integer gray levels, among them random sensor errors and errors in the analog-to-digital conversion itself. One way of reducing this random-error noise is to replace each pixel with an average of itself and its neighbors. Will this work accurately if we change the original image matrix as we proceed with the averaging?

One simple averaging technique is the moving 9-point average $W = \begin{pmatrix} 1 & 1 & 1 \\ 9 & 9 & 9 \\ 1 & 1 & 1 \end{pmatrix}$ which is applied to the pixel $p$ in the neighborhood $\begin{pmatrix} * & * & * \\ * & p & * \\ * & * & * \end{pmatrix}$. Obviously we can replace the uniform weights $W$ with ones biased for or against the central pixel.

A uniform $W$ has the advantage of not changing average scene radiance, but has the distinct disadvantage of blurring clearly defined lines. What happens if we apply $W$ to the pixel neighborhood $\begin{pmatrix} 0 & 0 & 0 & 255 & 0 & 0 & 0 \\ 0 & 0 & 0 & 255 & 0 & 0 & 0 \\ 0 & 0 & 0 & 255 & 0 & 0 & 0 \\ 0 & 0 & 0 & 255 & 0 & 0 & 0 \end{pmatrix}$, which is a bright vertical line one pixel wide? Remember, $W$ is being applied as a moving filter (i.e., element-by-element multiplication), not as we would in regular matrix multiplication.

We could change the weights so that $W = \begin{pmatrix} 1 & 1 & 1 \\ 32 & 8 & 32 \\ 1 & 3 & 1 \\ 8 & 8 & 8 \\ 1 & 1 & 1 \\ 32 & 8 & 32 \end{pmatrix}$, thus giving extra weight to the central pixel and its “nearest neighbors.” (Note that this $W$, like the earlier one, sums to 1, so that it does not change the image’s average radiance.)
new $\mathcal{W}$ has the advantage of reducing the blurring seen above while still smoothing the image. In fact, we can define a spatial filter $\mathcal{W}$ of any size and arrangement that we like. However, our choice of the elements $\mathcal{W}_{i,j}$ will not always result in smoothing, or a **low-pass spatial frequency filter** (see Rees, Fig. 10.8 for image effects resulting from a smoothing filter).

c) **spatial filtering**

We can apply spatial filters either directly to the images themselves or apply their Fourier transform to the images’ Fourier transforms. Here we will only concern ourselves with direct image manipulations.

One possible definition of an image feature’s spatial frequency $q$ is that

$$q = \left( \frac{\text{total # image pixels}}{\text{# pixels in the feature}} \right)^{1/2},$$

meaning that $q$ ranges between 1 (low spatial frequency; the feature occupies the whole image) and $\to \infty$ (high spatial frequency; a subpixel-sized feature).

Suppose that we want to emphasize high-spatial-frequency portions of an image (i.e., small-scale features). Here we would use a **high-pass spatial frequency filter** such as that shown in Rees’ Figure 10.10 where $\mathcal{W} = \begin{pmatrix} -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \\
-\frac{1}{9} & -\frac{8}{9} & -\frac{1}{9} \\
-\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \end{pmatrix}$. This $\mathcal{W}$ looks similar to the averaging (low-pass) filters, and it does have little effect on the contrast within uniform image areas. However, it sharpens pixel brightness gradients along boundaries (see Rees, Fig. 10.11 for an illustration).

A more visually appealing filter is the **high-boost spatial frequency filter** like the one illustrated in Rees, Figure 10.13. This filter gives high-frequency features more contrast, while not affecting low-frequency features. Qualitatively, we would say that the image’s “sharpness” has been improved. Here $\mathcal{W} = \begin{pmatrix} -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \\
-\frac{1}{9} & -\frac{17}{9} & -\frac{1}{9} \\
-\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \end{pmatrix}$ which, mathematically speaking, looks little different from the high-pass filter. Visually, however, the two are starkly different.
As we might expect, there is no end to the kinds of spatial, spectral, and other filters that we can apply to images. However, using these filters subtly moves us from preparing an image for analysis to analyzing it proper.

**Recipes for Using Spatial Filters**

Given the original subscene:

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>O</th>
<th>L.</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>83</td>
<td>132</td>
<td>63</td>
<td>150</td>
</tr>
<tr>
<td>2</td>
<td>114</td>
<td><strong>92</strong></td>
<td>140</td>
<td>136</td>
</tr>
<tr>
<td>3</td>
<td>133</td>
<td>135</td>
<td>55</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>137</td>
<td>115</td>
<td>80</td>
</tr>
</tbody>
</table>

First apply uniform 1/9-weight 3-x-3 filter ⇒ **smoothed** subscene:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>105</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>110</td>
<td></td>
</tr>
</tbody>
</table>

{at ROW 2, COL. 2, filtered DN = \(\frac{1}{9}(83 + 132 + 63 + 114 + 92 + 140 + 133 + 135 + 55)\) = 105, rounded}

Now apply a **high-boost** filter:

<table>
<thead>
<tr>
<th></th>
<th>-1/9</th>
<th>-1/9</th>
<th>-1/9</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1/9</td>
<td>17/9</td>
<td>-1/9</td>
<td></td>
</tr>
<tr>
<td>-1/9</td>
<td>-1/9</td>
<td>-1/9</td>
<td></td>
</tr>
</tbody>
</table>

which ⇒ **sharpened** subscene:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>156</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

{at ROW 2, COL. 2, filtered DN = \(-\frac{1}{9}(83 + 132 + 63 + 114 + 140 + 133 + 135 + 55) + \frac{17}{9}*92\) = 79, rounded}
d) Fourier image analysis

What does it do and why go to the trouble?

So far, all of our image manipulations have been in the spatial domain — we treat images like rectangular maps and simply do arithmetic on those maps’ individual pixels, and often we only want to compare adjacent pixels. As more-or-less intuitively obvious as spatial operations are, they analyze only part of an image’s content.

We can get very different information from an image — and make very different improvements to it — if we shift to the frequency domain, where maps of pixel DNs are regarded as sums of simple functions, say, cosines and sines.

- BOTTOM LINE: frequency analysis and filtering of images ⇒ results that are difficult, if not impossible, to get by spatial analysis alone

Fourier Analysis in Theory and Practice

1) \( f(x) \) is a real 1D spatial function of a real variable \( x \) — say, \( f(x) \) is the horizontal gradient of radiances \( L \) that underlie a satellite image, or \( f(x) = \frac{\partial L}{\partial x} \)

2) Substitute discrete DNs & pixels for the continuous \( L \) & \( x \) — so now \( f(x) \sim \frac{\Delta DN}{\Delta x} \)

3) Fourier’s theorem says \( f(x) \) can be approximated as accurately as necessary by adding a series of sine & cosine terms of increasing frequency — practical factors decide how many terms we add up.

4) The 1D Fourier transform of \( f(x) \) is written \( F(u) \), where \( u \) = a real (or even discrete integer) variable that determines frequency & \( F(u) \) = a complex function of \( u \). Now recall that:
   a) the imaginary unit \( i = (-1)^{1/2} \), and
   b) \( \exp(-2\pi i ux) = \cos(2\pi ux) - i \sin(2\pi ux) \) and \( \exp(+2\pi i ux) = \cos(2\pi ux) + i \sin(2\pi ux) \). (Euler’s formula)

5) As a complex function, \( F(u) \) is written in polar form as:
   \[
   F(u) = |F(u)| \exp(i \phi(u))
   \]
   where \( |F(u)| \) is a real function called \( F(u) \)’s magnitude and \( \phi(u) \) is its real phase. \( |F(u)|^2 \) is a function called the power spectrum (or spectral density) of \( f(x) \).

SO431 — Image analysis basics (10-29-2009)
6) For discrete functions like a satellite image’s integer DNs, we can write:

<table>
<thead>
<tr>
<th>discrete Fourier transform</th>
<th>inverse Fourier transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \exp\left(-\frac{i 2\pi ux}{N}\right) ] (Eq. 3)</td>
<td>[ f(x) = \sum_{u=0}^{N-1} F(u) \exp\left(\frac{i 2\pi ux}{N}\right) ] (Eq. 4)</td>
</tr>
</tbody>
</table>

Note that \( F(u) \) and \( f(x) \) are inverse functions — knowing \( F(u) \) lets us reconstruct our original \( f(x) \) and vice versa. Our original function \( f(x) \) can now be identified as just the inverse of its transformed self, the complex \( F(u) \).

\( N \) is the number of sampled points along \( f(x) \) (for us, \( f(x) \) is the DN-variations along a row of pixels in a digital image). Normally \( N = 1/2 \) the image dimension being sampled (here = 1/2 image width) in order to be able to define the highest-frequency feature in the image row (see p. 10 above for a frequency definition).

7) The simplest way of moving from 1D line functions to 2D “image functions,” is to do a 1D Fourier transform on each image row, then do a separate transform on each image column. Together, these ⇒ the desired 2D transform.

8) The 2D transform \( F(u,v) \) is a 2D array whose complex #s we can display as an image. It’s customary to display only \( F(u,v) \)'s magnitudes when doing such transforms as the FFT (fast Fourier transform). While we need the phase data to actually calculate the inverse transforms, that data has been politely called “difficult or impossible to interpret visually.” (Russ 1995, p. 289)
Although visually interpreting the FFT magnitude array is no picnic either, it’s important to understand it. First, note that the FFT algorithm requires square images of size $= 2^n$, where $n$ = an integer. This limits FFT images to sizes such as 256-x-256 or 512-x-512, etc. Because we can rescale and/or crop images that don’t fit these square dimensions, the $2^n$ size restriction is only a minor inconvenience.

In the YP686 FFT array above, note that:

a) the darker a FFT array feature is, the more pixels in the original image have the corresponding orientation and frequencies

b) spatial frequency increases radially outward from FFT array center ($\text{radius} = \rho$)
So the original image’s mean gray level is represented by the center black dot ($\rho = 0$), and the highest-frequency features (e.g., waves; $\rho \Rightarrow 255$) are represented along the FFT edges. Frequency can be displayed in other ways, but this is the one we’ll use.

c) large, linear features in the original spatial image appear in the FFT array as lines rotated 90° w.r.t. the original features (orientation angle $= \theta$) Note that purely sinusoidal features with only 1 frequency are dots. This rotation is inherent in the frequency transform $F(u,v)$, but displaying the 90° rotation is an arbitrary choice.

d) Because two features with $\theta = \theta_1$ and $\theta_1 + 180^\circ$ produce the same FFT, half of the square displayed FFT array is redundant (whether top vs. bottom or left vs. right). So if we flip the FFT’s right half vertically & then horizontally, it will appear exactly the same as the left half.

Now identify some image features in the FFT array shown above.

SO431 — Image analysis basics (10-29-2009)
3) **image classification**

Many classification schemes are possible; we mention four here.

*a) density slicing*

When we examine single-band (*i.e.*, monochrome) images, it may be useful to sort DNs into different categories, a technique called **density slicing**. For example, we may find that vegetation in a near-IR image has DNs ranging from 100-150. If **only** vegetation fell within this range, we would have a useful tool for predicting its presence. In the thermal IR, where exitance depends only on temperature, single-band density slicing can be quite useful.

However qualitatively useful density slicing is, most natural scenes are too complex for it to be **quantitatively** useful. More quantitative information can be extracted by using several wavelength bands at once, a technique called “multispectral classification.”

*b) multispectral classification*

If we compare the same image DNs in two different spectral regions, we can construct **scattergrams of the cross-correlations** between corresponding pixels. Below is a scattergram for Bands 2 & 3 of our Morro Bay, CA Landsat image. Bright areas in the scattergram indicate more occurrences of a given DN$_2$, DN$_3$ pairing.
A Recipe For Making Multispectral Scattergrams

Consider a small subscene within a 24-bit color, aerial image \( C \) of a vivid volcanic twilight.

### RED color plane (~ Landsat Band 3)

<table>
<thead>
<tr>
<th>ROW #</th>
<th>C [338]</th>
<th>O [339]</th>
<th>L [340]</th>
<th># [341]</th>
</tr>
</thead>
<tbody>
<tr>
<td>311</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>312</td>
<td>225</td>
<td>225</td>
<td>224</td>
<td>225</td>
</tr>
<tr>
<td>313</td>
<td>197</td>
<td>196</td>
<td>195</td>
<td>194</td>
</tr>
<tr>
<td>314</td>
<td>172</td>
<td>171</td>
<td>170</td>
<td>169</td>
</tr>
</tbody>
</table>

### GREEN color plane (~ Landsat Band 2)

<table>
<thead>
<tr>
<th>ROW #</th>
<th>C [338]</th>
<th>O [339]</th>
<th>L [340]</th>
<th># [341]</th>
</tr>
</thead>
<tbody>
<tr>
<td>311</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>312</td>
<td>39</td>
<td>39</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>313</td>
<td>37</td>
<td>36</td>
<td>36</td>
<td>35</td>
</tr>
<tr>
<td>314</td>
<td>37</td>
<td>36</td>
<td>35</td>
<td>35</td>
</tr>
</tbody>
</table>

### BLUE color plane (~ Landsat Band 1)

<table>
<thead>
<tr>
<th>ROW #</th>
<th>C [338]</th>
<th>O [339]</th>
<th>L [340]</th>
<th># [341]</th>
</tr>
</thead>
<tbody>
<tr>
<td>311</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>312</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>48</td>
</tr>
<tr>
<td>313</td>
<td>43</td>
<td>42</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>314</td>
<td>44</td>
<td>43</td>
<td>42</td>
<td>41</td>
</tr>
</tbody>
</table>

To build this part of \( C \)’s red/green multispectral scattergram \( M \):

1) For 8-bit DNs, create a new 256-by-256 integer array \( M \) and initialize (i.e., fill) it with zeros. (At worst, one element of the completed \( M \) could = the total number of pixels in \( C \).)

2) At column 338, row 311 in \( C \), the red DN = 255 and the green DN = 45, which is 1 case of this red/green pairing.

3) Now go to \( M \)’s row 255 and column 45 and add 1 to the value found there. This addition indicates 1 more case of red DN = 255 and green DN = 45 at the same pixel location in \( C \). Since \( M \)’s initial values were all zero, the new value of \( M(255,45) = 1 \).
4) At column 339, row 311 in \( C \), once again the \textbf{red DN} = 255 and the \textbf{green DN} = 45. This is the second such red/green pairing in \( C \), so adding 1 to \( M(255,45) \) makes \( M(255,45) = 2 \).

5) Do this for all rows and columns in \( C \). When done, the scattergram \( M \) will be a kind of 2D histogram of all pairings of red and green in \( C \). In fact, \( M \)'s values show the \textbf{degree of cross-correlation} between all red and green DNs in \( C \) — where \( M \) values are large, the cross-correlation is high.

6) Plot \( M \) as a square gray-scale map, with \( M \)'s largest element displayed as white and its zero elements (of which there will be many) displayed as black. Display intermediate-value \( M \) elements as different grays.

7) If we compare any color plane of \( C \) (or a gray-scale satellite image) with a copy of itself, all we actually get is the ordinary 1D histogram. In this case, our gray-scale map of \( C \) will be a line 1 pixel wide (and of varying gray level) that stretches from \( M(0,0) \) to \( M(255,255) \). This map makes sense, because any single image is perfectly correlated with itself.

If the image DNs from the two bands form distinct \textbf{clusters} on the scattergram, we have identified (presumably) unique spectral classes that arise from spectral reflectances or emissivities that are unique to different materials within the image. While this assumption is wrong in principle (see “metamerism” below), it’s right in practice often enough that we can routinely use it successfully.

\textbf{Band ratios} are a one-dimensional form of multispectral classification. For spectral bands 1 and 2 we calculate the ratio \( \frac{\text{DN} \_2}{\text{DN} \_1} \) or \( \arctan \left( \frac{\text{DN} \_2}{\text{DN} \_1} \right) \), thereby effectively eliminating radiance changes within an image that are due to the effects of nadir viewing angle \( \psi \) on: a) reflected and transmitted radiances from the surface and atmosphere, b) the sensor optics (called \textbf{vignetting}).

Band ratios’ disadvantage is that they cannot yield the kind of two-dimensional clustering that we hope will give us additional physical insight from our data. Various manual and automatic (\textit{i.e.}, statistical) approaches to identifying pixel clusters are used.

\textbf{Unsupervised} (automated) \textbf{classification} is clearly the only practical way of assessing large data sets. However, once the various DN clusters are identified, a human analyst faces the problem of deciding whether they correspond to identifiable physical materials (say, vegetation or clear ocean water). Now we have \textbf{supervised classification} in the sense that, although the computer algorithm sorts the pixel data into clusters, the clusters’ physical significance must first be identified manually. The supervision consists of the analyst instructing the computer to remember the (presumably) narrow range of pixel gray levels found within homogeneous \textbf{training areas} in an image.
Now, all other occurrences of those gray levels within the image are likely to be caused by the same physical features. More accurately, it's the coincidence of the same ranges of gray levels in 2 (or more) spectral regions of a scene that identifies features. But does this coincidence guarantee that we have found the same materials?

No, because **metamerism** says that broad wavelength-band data such as satellite images cannot uniquely specify data that has more detail, such as a reflectance spectrum.
Consider an illustration. Above are two spectral reflectance curves for a light blue pigment. Obviously the spectra are not (and cannot be made) congruent, or overlapping each other. If they did coincide, it makes sense that they would have the same color. Yet surprisingly these two very dissimilar reflectance spectra, under one particular kind of illumination, have identical colors. Why?

Remember that our visual systems, like satellite spectral channels, are broad-band sensors of spectral radiances $L_\lambda$. We have three such sensors to span the visible (~380-700 nm); a satellite often has the same number. However, a detailed picture of the visible spectrum might require 30-60 separate $L_\lambda$.

It makes sense that a high-resolution spectrum containing 10-20 times as much information as we can process will have spectral details to which we (or any broad-band sensor) will be oblivious. We can only respond to three different sets of radiances, each of which as been spectrally integrated across a different (and wide) section of the visible spectrum.

If we cannot resolve small-scale spectral detail, then within limits, we will not notice changes in those spectral details. We respond only to the ratios of the different photodetectors’ responses. In similar fashion, broad-band multispectral analysis of satellite images can only make distinctions within an image if DN-ratios change.

Imagine that our sample metameric reflectance spectra (p. 13) were of, say, silty water and blue flowers. If we looked at large areas of both from a satellite, we could not distinguish between them. Note that the reflectance spectra crisscross in the visible, with first one, then the other being more reflective. That kind of behavior is typical of metameric spectra; can you see why?

c) principal components analysis

When we divide visible-$\lambda$ natural scenes into three broad spectral bands, the different images contain very similar, but not identical, information. One proof that the images from different satellite bands are largely redundant is that we can overlay them to form clear, plausible colored images. If there were little correlation among the bands, the overlay would be a blur.

We can illustrate the visible bands’ high correlation by comparing, say, bands 2 and 3 (0.52-0.60 $\mu$m and 0.63-0.69 $\mu$m) with the thermal infrared’s band 6 (10.4-12.5 $\mu$m) from the Landsat image of Morro Bay, California. Note how much less distinct the IR image is in comparison to the visible ones.

Principal components analysis exploits the redundancy that exists in most satellite images by: (a) mathematically removing the correlation between different spectral
bands, (b) using this transformed version of the original pixel data to account for a large fraction of the image’s gray-level variance.

Perhaps surprisingly, more than 98% of the original variance among pixel gray-level values can be accounted for in this way (see Rees, pp. 216-218), even when using only a fraction of the transformed data. Thus we can build a low-error approximation to the original data while using very little of that data.

d) texture analysis

Thus far, we have been concerned with spectral analysis of images. Clearly images contain significant spatial information as well. Techniques that exploit images’ spatial data are often called texture analysis.

We can attempt to quantify texture as either: (a) variance of gray-level values about their mean, (b) the shape of an image’s Fourier spatial frequency spectrum (see our earlier mathematical definition of spatial frequency, q). Technique (b) is often used in satellite radar imaging, where data is available from only a single spectral band.

Image texture can tell us, at the most basic level, about surface geometry. Some of the techniques developed include: (a) the semi-variogram, (b) the spatial dependence matrix, and (c) syntactic methods. These last are attempts to emulate the kind of rule-making (and rule-observing) that guides human analysts in categorizing spatial features.

A Recipe For Making Spatial Dependence Matrices (SDMs)

The SDM shows the frequency of all possible gray-level juxtapositions within an image. To make this texture-analysis matrix, we usually operate on 1 band of a satellite image $\mathbb{G}$ (or 1 color plane in a color image):

**BLUE color plane $\mathbb{G}$ (\~ Landsat Band 1)**

<table>
<thead>
<tr>
<th>ROW</th>
<th>C</th>
<th>O</th>
<th>L</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>311</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>312</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>48</td>
</tr>
<tr>
<td>313</td>
<td>43</td>
<td>42</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>314</td>
<td>44</td>
<td>43</td>
<td>42</td>
<td>41</td>
</tr>
</tbody>
</table>

1) For 8-bit DNs, create a new 256-by-256 integer array $\mathbb{S}$ and initialize (i.e., fill) it with zeros. (At worst, one element of the completed $\mathbb{S}$ could = the total number of pixels in $\mathbb{G}$.)

2) Begin with a reference pixel at row 312, column 339 in $\mathbb{G}$. Its DN = 47. By definition, this reference pixel has a relative row, column position of 0,0.
3) Now examine the 4 rightward (0, 1 and 1,1), downward (1,0), & left backward (1, -1) neighbors of pixel 0,0 (absolute row 312, column 339 here).

{Don’t compare 0,0 with itself, or look up one row or back to pixel 0,-1 because that lateral next-neighbor comparison was made when 0,-1 was the reference pixel (= 0,0) and the present 0,0 was 0,1.}

4) When we compare 0,0 with 0,1, we find that both have DN = 47. So we add 1 to $S(47, 47)$, which makes $S(47, 47) = 1$.

When we compare 0,0 with 1,1, one has DN = 47 and the other has DN = 42. So we add 1 to $S(47, 42)$, which makes $S(47, 42) = 1$. The next time that we find a DN 47 adjoining DN 42 anywhere in the image, $S(47, 42)$ becomes $= 2$.

5) Do this for all rows and columns in $G$ — every row and column is reference pixel 0,0 for 4 comparisons made with its adjoining pixels. When done, we have a map of the frequency of gray-level juxtapositions throughout $G$, so the SDM is a map of $G$’s spatial texture.

6) Plot $S$ as a square gray-scale map, with $S$’s largest element displayed as white and its zero elements (of which there will be many) displayed as black. Display intermediate-value $S$ elements as different grays.

-21-

\[
\begin{array}{ccc}
\text{RELATIVE COLUMN (2nd #)} & \text{RELATIVE ROW (1st #)} \\
\{-1, -1\} & \{0, -1\} & \{1, -1\} \\
\{-1, 0\} & 0, 0 & 1, 0 \\
\{-1, +1\} & 0, 1 & 1, 1 \\
\end{array}
\]

Note that #s in { } are NOT compared with 0,0
IMPORTANT CONCEPTS FOR REVIEW

- pixel DNs (or gray levels) & # of possible “images” in an n-x-m array
- bits, bytes, & satellite image storage capacities
- LUTs — converting between radiances & DNs
- physical interpretation of satellite-image band histograms
- FOV & tangent-plane mapping
- use of GCPs in rectifying satellite images
- definition of contrast, radiance dynamic range, & their application to satellite images
- linear contrast stretch — calculation & uses
- histogram equalization in theory & practice
- smoothing (or blurring) filters vs. various sharpening filters — calculation & uses
- Fourier image analysis — describe conversions between frequency vs. spatial domain; interpreting & manipulating FFTs; filtering in frequency space
- density slicing vs. thresholding (from lab exercises)
- multispectral classification — calculation & uses; cross-correlation scattergrams, clusters, band ratios, supervised vs. unsupervised classification
- metamerism & its implications for satellite imagery
- principal components analysis
- texture analysis & spatial dependence matrices — calculation & uses; mapping a satellite image’s frequency of gray-level juxtapositions
- review Eqs. 1 – 4
- review all material in additional handouts, supplementary reading on reserve