

GLCM TEXTURE: A TUTORIAL

v. 3.0 March 2017 replaces v. 2.8 of August, 2005

v. 3.0 incorporates all corrections up to v. 2.8. It also adds FAQ and discussion created in response to many questions and input from students.

Mryka Hall-Beyer, Ph.D.

Department of Geography
University of Calgary
Calgary, Alberta T2N 1N4 Canada

copyright 2000, 2005, 2017 Mryka Hall-Beyer



GLCM Texture: A Tutorial v. 3.0 by M. Hall-Beyer is licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/).

Statement on peer review and citation: Although this tutorial is not published by a professional journal, it has undergone extensive peer review by third-party reviewers at the request of the author. In addition, many users have discovered minor errors and pointed out areas of improvement that have gone into subsequent versions of the tutorial. When citing, please give the current version and date as displayed in the header above.

Permission is granted to use this tutorial free of charge for educational purposes only, and provided that credit is given.

Please e-mail any comments or corrections to mhallbey@ucalgary.ca

Table of Contents: click on link to take you to that section of the tutorial.

Topics	Examples	Definitions	Equations
What is texture	fill in east matrix framework	co-occurrence	ASM
Information about this tutorial	normalized horizontal GLCM	degree	Contrast
Background information on texture	GLDV	energy	Correlation
Where did the idea come from?	weighted average	entropy	Dissimilarity
Some useful references	image edge problems	horizontal matrix	Energy
Some other approaches besides GLCM	degree	neighbour pixel	Entropy
IF YOU MAINLY WANT AN IN-DEPTH UNDERSTANDING OF THE CONCEPT, USE THIS SECTION:	contrast calculation	normalize	GLCM Mean
the GLCM: definition	orderliness measures	offset	GLCM Std Dev
GLCM calculations	correlation	order	GLCM Variance
Test image	images	orderliness	Homogeneity
Framework for the GLCM		P_{ij}	Normalization
Spatial relationships between two or more pixels	Rgb test image	probability	
Separation between two pixels	Red band of test image	reference pixel	
How to read the framework matrix	Contrast group	spatial relationship	
Properties of the GLCM	CONTRAST	symmetrical matrix	
How to construct a symmetrical matrix	HOMOGENEITY	transpose	
Expressing the GLCM as a probability	DISSIMILARITY	weighted average	

<u>How to normalize the GLCM</u>	<u>Orderliness group</u>		
<u>Summary of steps in creating a symmetrical normalized GLCM</u>	<u>ASM</u>		
<u>Properties of the symmetrical normalized GLCM</u>	<u>ENTROPY</u>	<u>transpose</u>	
<u>The grey level difference vector (GLDV)</u>	<u>Descriptive stats group</u>	<u>weighted average</u>	
IF YOU MAINY WANT TO KNOW HOW TO CALCULATE THE TEXTURES ONCE YOU HAVE THE GLCM, USE THIS SECTION:	<u>GLCM MEAN</u>		
<u>Texture calculations</u>	<u>GLCM VAR</u>		
<u>The texture image</u>	<u>GLCM CORR</u>		
<u>Problems with image edges</u>			
<u>Groups of texture calculations</u>			
<u>Contrast group</u>			
<u>Contrast</u>			
<u>Dissimilarity</u>			
<u>Homogeneity</u>			
<u>Orderliness group</u>			
<u>how weights work in orderliness measures</u>			
<u>Angular Second Moment (ASM), Energy and MAX</u>			
<u>Entropy</u>			
<u>Logarithms mini tutorial</u>			
<u>Descriptive statistics group</u>			
<u>GLCM Mean</u>			
<u>GLCM variance and standard deviation</u>			
<u>GLCM Correlation</u>			
<u>Properties of correlation</u>			
IF YOU ARE USING TEXTURE IN AN IMAGE ANALYSIS CONTEXT, BE SURE TO CONSULT THIS SECTION:			
<u>Practical matters</u>			
<u>Calculations performed on one image channel</u>			
<u>All about neighbourhood windows</u>			

How to choose texture measures			
Correlation of texture measures with one another			
IF YOU MAINLY WANT TO SEE SOME EXAMPLES, USE THIS SECTION:			
Examples of texture on an image			

What is texture?

Everyday texture terms - rough, silky, bumpy - refer to *touch*. They are most easily understood in relation to a topographical surface with high and low points, and a scale compatible with a finger or other.

A texture that is ***rough*** to touch has:



- a large difference between high and low points, as compared to the size of a fingertip, and
- a space between highs and lows approximately the same size as a fingertip.

Rough textures can of course occur at any spatial scale, not just what we could touch. The illustration above would be considered rough whether it represents 1 cm or 100 km in horizontal dimension. To probe it with anything except our scale-informed eyes, however, we would have to adapt the “fingertip” used to the appropriate scale.

Silky or ***smooth*** has



- little difference between high and low points, and
- the differences would be spaced very close together relative to fingertip size.

Image texture works in the same way, except the highs and lows are brightness values (also called grey levels, GL, or digital numbers, DN) instead of elevation changes. Instead of probing a

fingertip over the surface, a "window" - a square box defining the size of the probe - is used. And, of course, the scale (pixel size) is not necessarily that of a fingertip, but may be defined however is convenient for the image data available.

Textures in images quantify:

- **Grey Level differences (contrast)**
- **Defined size of area where change occurs (neighbourhood, defined by a window size)**
- **Directionality, or lack of it (omnidirectional)**

Information about this tutorial

This document concerns the most commonly used texture measures, those derived from the **Grey Level Co-occurrence Matrix (GLCM)**. The essence is understanding the calculations and how to do them. The GLCM and "texture measures" derived from it are *descriptive statistics*, though rather complicated ones. Their values, and the use of those values to answer research or application questions, depend on the data for which they are calculated. Just like an average as a raw number is of no particular use unless the user understands the dataset from which it is derived, so the texture measures are of little use unless the user understands the way they can be used and the characteristics of the dataset from which they are derived.

Understanding the calculations involves

- **Defining** a Grey Level Co-occurrence Matrix (GLCM)
- **Creating** a GLCM
- Using it to **calculate** texture measures (in the exercises).
- Understanding how calculations are used to build up a **texture image**

There are exercises to perform. When done, click on the answer link to see the details of the calculations.

Also, extended sections in italics within the exercise section contain information that is supplementary and is not necessary the first time through. You may skip these and return to them later.

Material in blue italics is also supplementary, and addresses common misconceptions and questions related to material in that section. Some are labelled FAQ.

Additional information related to practical use of texture is provided after the calculations section. [Click here](#) to go directly to it.

BACKGROUND INFORMATION ABOUT TEXTURE

Who originated this idea?

Most of the GLCM texture calculations used in remote sensing were systematized in a series of papers by Robert Haralick and co-authors in the 1970s. He built on the work of several researchers concerned with mathematical pattern (spatial) analysis. There is a series of "standard" textures called [Brodatz Textures](#) that texture algorithms are tested on, to see if the algorithms can tell these textures apart. They include such things as leather, grass, woven cloth, felt, and the like. Anys and He (1995) systematized the terminology, and proposed a way to choose the best textures to use in an image.

[Haralick et al. \(1973\)](#) proposed 14 different measures. They include the ones detailed here, and also variance calculated on the sum of adjacent pixels; variance on the difference between adjacent pixels; entropy on the sum and on the difference; correlation involving entropies, and the maximum correlation coefficient. These have not been generally adopted. Haralick also suggested calculating each of the 14 measures for the four directions, then taking the mean and range of each measure and using those values in classification. Again, this has not been widely adopted. Indeed, directionality has not been widely used, even when it is available on software. Instead, the option is almost always adopted of a single, "invariant" spatial direction that is the average of the four directions. This is perhaps because of widespread lack of understanding of what Haralick's texture measures can do. It would seem likely that many ground covers would exhibit a textural directionality that would make this a useful tool in identifying classes: orchards spring to mind. This tutorial is intended to make the idea of GLCM texture more accessible, while software makes the calculations painless. With experience, this difficulty might disappear, or at least be better constrained. Nevertheless, the directionality will only be addressed here in the calculations section where horizontal and vertical GLCMs will both be shown.

Personal interest I became aware of the general utility of including some measure of spatial pattern in a general spectral classification scheme, while classifying grassland ecological units in the early 1990s as part of my doctoral research. The topic was not texture in itself, rather seeking for information derivable from the image itself that would increase classification accuracy in grasslands – a notoriously “textureful” class. Pragmatically, I used the software available at the time, and I did demonstrate that the spatial information improved the classification accuracy to a statistically significant extent. However, my advisors, supervisory and examining committee, and audience members at conferences kept asking “why?” It took me several years to realize that they were not quizzing the new student or colleague to see if I knew the answer, but rather none of them knew themselves how it worked, and hoped that I would explain it! Thus this tutorial was born, after I had been teaching remote sensing at an advanced level for a few years. It is intended to set out what I had to learn the hard way, as well as to help future students and researchers apply texture intelligently.

It remains true today, just as in the 1990s, that adding *some* texture measure – *any* measure! - to a classification usually improves the accuracy. This is because texture measures spatial relationships, whereas spectral data measures, basically, chemical properties of the ground

objects. These are very different object properties, so spectral and spatial are very likely to be independent data and so complement one another. This remains true as remote sensing moves into higher spatial resolution imagery, where we can access a signal at much higher detail, but also create much more noise as well, compounding the difficulty in telling them apart. Spatial helps!

A complete review of texture today would have to include in-depth analysis of things like landscape metrics (patch analysis), wavelets, semivariograms, fractal analysis, and many other ideas. *Texture is important in many areas besides remote sensing, such as computer graphics. Given the huge amount of new research in many areas, sometimes the wheel gets reinvented, and given different names. This makes talking about GLCM in a remote sensing context somewhat complicated. If you are reading this as an expert in another area, please be aware that this tutorial is intended for remote sensing students and practitioners, and uses their vocabulary and assumes knowledge common to remote sensing. If you find yourself getting confused by terms such as “wavelength band” or “PCA component”, please consult a reputable online source or remote sensing textbook for definitions.*

Despite these complications, in terms of software accessibility, ease of use, and even intuitive grasp of the texture measures, GLCM remains a primary go-to tool, mainly because it is able to measure roughness, coarseness and directionality in one calculation. I have recently shown ([Hall-Beyer 2017](#)) that the choice of measures can be simplified by using certain rules of thumb. I hope it will encourage image analysts to not ignore spatial information out of fear of misuse.

Some useful references: this list is not exhaustive! As texture is a fairly basic concept, older references are often still useful. There are many articles applying texture to particular ground classes or with particular images. Those included below will give an idea of how texture is used; more examples can be had fairly easily with searches using “GLCM” or “co-occurrence” or “texture” in the keywords. Additional comments are provided below under some references. There is extensive literature on the application of texture in synthetic aperture radar (SAR) images, and in applications fields such as sea ice, cloud analysis and forestry. There is an increasing literature comparing GLCM texture measures to other spatial approaches in terms of classification accuracy resulting from their use.

Akono, A.; Tonyé, E.; Nyoungui, A. N.; Rudant, J.-P. 2003. Nouvelle méthodologie d'évaluation des paramètres de texture d'ordre trois. Int. J. remote Sensing vol. 24 no. 9 pp. 1957-1967. *This looks at texture involving the co-occurrence of three pixels instead of two. It is interesting, but to my knowledge has not been picked up as practical. GLCM is complicated enough to interpret, and adding additional dimensions has not proved useful. I would compare this to the relative utility of standard deviation vs. kurtosis in statistics.*

Anys, H. & D-C. He 1995. Evaluation of textural and multipolarization radar features for crop classification. IEEE Trans. on Geosci. And Rem. Sens. Vol. 33 no. 5 pp

Clausi, D. A. 2002. An analysis of co-occurrence texture statistics as a function of grey-level quantization. Canadian Journal of remote sensing vol. 28 no. 1 pp. 45-62. *Important for quantization levels: as a result of this article one software added the option for 5-bit quantization (bit depth); most use only 4-bit.*

Coburn, C. A. and A. C. B. Roberts. 2004. A multiscale texture analysis procedure for improved forest stand classification. International Journal of Remote Sensing. vol. 25 no. 20 pp. 4287-4308.

Ferro, C. J. S. and T. A. Warner 2002. Scale and texture in digital image classification. Photog. Eng. and Rem. Sens. vol. 68 no. 1, pp. 51-63.

Gotlieb, Calvin C. and Herbert E. Kreyszig. 1990. Texture Descriptors based on co-occurrence matrices. Computer Vision, Graphics, and Image Processing, Volume 51, No. 1, pp. 70-86. *Tests discriminatory power in computer vision setting using Brodatz textures.*

Hall-Beyer, M. 2017. [Practical guidelines for choosing GLCM textures to use in landscape classification tasks over a range of moderate spatial scales](#). International Journal of Remote Sensing, 38"1312-1338.

Hann, D. B.; Smith, A. M. S.; Powell, A. K. 2003. Classification of off-diagonal points in a co-occurrence matrix. Int. Journal of Remote Sensing. vol. 24 no. 9 pp. 1949-1956.

Haralick, R.M. 1979. Statistical and Structural Approaches to Texture. Proceedings of the IEEE, vol. 67 pp.786-804. *This is the basic reference that should generally be cited when working with GLCM texture.*

Haralick, R.M., K. Shanmugam and I. Dinstein. 1973. Textural Features for Image Classification. IEEE Transactions on Systems, Man and Cybernetics. SMC vol. 3 no. 6 pp.610-620. *The original publication, although it is in a hard to access volume.*

He, D.-C. and L. Wang. 1990 Texture Unit, Texture Spectrum and Texture Analysis. IEEE Trans. On Geoscience and Remote Sensing, vol 28 no. 4 pp. 509-512. *A novel non-GLCM approach to texture that will be helpful in understanding the basic concepts of what texture is.*

[He, D.-C., L. Wang and J. Guibert. 1988](#). Texture Discrimination based on an optimal utilization of texture features. Pattern recognition vol. 21 no. 2 pp. 141-146.

Jensen, J.R. 2015. Introductory Digital Image Processing: A Remote Sensing Perspective. 4th ed. Upper Saddle River, NJ: Prentice-Hall. *A standard textbook with excellent examples.*

Kourgli, A. and A. Belhadj-Aissa. 1997. Approche structurale de génération d'images de texture. International Journal of Remote Sensing vol. 18 no 17, pp. 3611-3627. *Even if you don't read French this possesses an excellent bibliography up to 1997.*

Pearlstine, L., K. M. Portier and S. e. Smith. 2005. Textural discrimination of an invasive plant, *Schinus terebinthifolius*, from low altitude aerial digital imagery. *Photogrammetric Engineering and Remote Sensing*. 7vol. 1 no. 3 pp. 289-298. *An approach using high spatial resolution imagery*.

van de Sanden, J. J. and D. H. Hoekman. 2005. Review of relationships between grey-tone co-occurrence, semivariance and autocorrelation based image texture analysis approaches. *Canadian Journal of Remote Sensing* vol. 38 no. 3 pp 207-213. *A comparative article and review up to 2005*.

Wulder, M. and B. Boots. 2001 Local Spatial Autocorrelation Characteristics of Landsat TM Imagery of a Managed Forest Area. *Cdn. J of Remote Sensing*, vol. 27 no. 1 pp. 67-75

Other approaches to texture:

The GLCM explained here is not the only texture measure that has been proposed. However, it is the most commonly implemented one. See Jensen (1996) for the idea of a "**texture spectrum**" ([He & Wang 1990](#)). This is an interesting approach to characterizing image classes, but it has the disadvantage of requiring a very large number of pixels within each class to be useful as a classification tool.

Much work is now being done to characterize texture using semivariograms (spatial autocorrelation using Moran's I or Geary's C statistics), fractal dimension, wavelet analysis, lacunarity, and others. There is an extensive literature on each that is not covered here. [Van der Sanden and Hoekman \(2005\)](#) have demonstrated that GLCM CON is identical to semivariance, and GLCM COR provides almost identical information as provided by autocorrelation methods. [Pearlstine et al. \(2005\)](#) suggest other textures, such as statistics (mean, median, std deviation) of density of edges following edge-enhancing filtering of various kinds.

A good deal of other texture research is on the intersection of landscape metrics with texture, and relates the concepts to a particular ground phenomenon. Landscape metrics are generally developed for use in vector-based systems, using irregular patches, defined for a certain ground cover, as the basis for defining texture. Texture uses the arbitrarily-defined square pixel as the base. The research field of GEOBIA (geographically-oriented object analysis) can bring these two together, in two ways. One investigates the use of texture images to aid in drawing meaningful patch boundaries using raster-based data inputs. The other uses textures as ways of distinguishing these objects from one another and characterising them. The latter is more developed as of this writing (2017), as it can use texture images derived from other software within the object-oriented classification. The possibility of deriving textures from within pre-defined objects is not yet well developed, and this is where it may overlap with landscape (patch) metrics.

The **Grey Level Co-occurrence Matrix, GLCM** (also rarely called the **Grey Tone Spatial Dependency Matrix**)

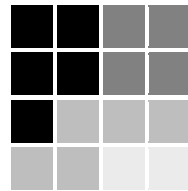
Definition: The GLCM is a tabulation of how often different combinations of pixel brightness values (grey levels) occur in an image.

GLCM Calculations

The test image:

Here is a simple "**test image**" for working out examples. The values are image grey levels (GLs). It will be used throughout this tutorial. As is the case with remote sensing images, the lower the GL (digital number) the darker the image appears. This uses 2-bit data (2^2 or 4 possible values, namely 0, 1, 2 and 3). This will be discussed more later; for now we will keep it simple.

The image as it appears:



The GL (digital numbers) associated with each pixel:

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Definition: Order:

The GLCM described here is used for a series of "**second order**" texture calculations.

Second order means they consider the relationship between **groups of two pixels** in the original image. *Notice that this is not the same thing as "second order equations" which would mean equations with some variables squared.*

First order texture measures are statistics calculated from the original image values, like variance, and **do not consider pixel relationships**.

Third and higher **order** textures (considering the **relationships among three or more pixels**) are theoretically possible but not implemented due to calculation time and interpretation difficulty.

Framework for the GLCM:

Spatial relationship between two pixels:

GLCM texture considers the relation between two pixels at a time, called the **reference** and the **neighbour** pixel. In the illustration below, the neighbour pixel is chosen to be the one to the east (right) of each reference pixel. This can also be expressed as a (1,0) relation: 1 pixel in the x direction, 0 pixels in the y direction. Each pixel within the window becomes the reference pixel in turn, starting in the upper left corner and proceeding to the lower right. Pixels along the right edge have no right hand neighbour, so they are not used for this count. The illustration below shows one such relationship: the pixel value shown in red are reference pixels and the pixels shown in blue are neighbour pixels in a (1,0) relationship to their reference pixel. *This shows examples only; all pixels can serve as reference and neighbour pixels. If you are about to object that there is a problem with the left and right edges, you are right: we will show how this is accounted for later in the document.*

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

To see images of the effect of different spatial relationships in addition to the (1,0) shown here, [click here](#).

Separation between two pixels:

All the examples in this tutorial use 1 pixel **offset** (a reference pixel and its immediate neighbour). If the window is large enough, using a larger offset is perfectly possible. There is no difference in the counting method. The sum of all the entries in the GLCM (i.e. the number of pixel combinations) will just be smaller for a given window size.

Now we will look at how the GLCM matrix is actually constructed, by counting and tabulating the number of pixel pairs that show a combination of all possible GL value pairs. Let's take the first pair of reference and neighbour pixels as shown above, in red (reference) and blue (neighbour). Since the reference pixel has the GL value of 0, and the neighbour pixel the value of 1, we count this as one entry in a table of frequencies.

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

neighbour pixel value ->	0	1	2	3
ref pixel value: ↓				
0		<input checked="" type="checkbox"/>		
1				
2				
3				

To completely fill in the table, each pixel in the original image will take its turn as reference pixel, and a check box will be produced for every reference pixel. Don't worry, computer software will actually do the counting for you: what is important here is to understand how it is done (and why).

To generalize from this example: The table below shows the combinations of the grey levels that are possible for the test image, and their position in the matrix. We already saw in the example that a reference pixel with value 0 next to a neighbour pixel in the (1,0) relationship would lead to an entry in the (0,1) box:

neighbour pixel value ->	0	1	2	3
ref pixel value: ↓				
0	0,0	0,1	0,2	0,3
1	1,0	1,1	1,2	1,3
2	2,0	2,1	2,2	2,3
3	3,0	3,1	3,2	3,3

How to read the matrix framework:

The top left cell will be filled with the number of times the combination 0,0 occurs, i.e. how many times within the image area a pixel with grey level 0 (neighbour pixel) falls to the right of another pixel with grey level 0 (reference pixel). Each cell is read in this pattern with appropriate changes in numbers.

A different co-occurrence matrix exists for each spatial relationship. The one produced above was for the (1,0) relationship: a pixel and its neighbour to the right (east). Other possible relationships are above (0,1), next to the west (-1,0), diagonal (1,1) or (-1,-1). [More information: click here.](#)

Example: Actually making the count for the test image we are using: **Fill in the matrix framework for the east (1,0) spatial relationship:**

Original image:

```

0 0 1 1
0 0 1 1
0 2 2 2
2 2 3 3

```

The filled in east (1,0) spatial relationship GLCM. The entries in each cell are the sum of the counts (check boxes) as described above. The entries in colour refer to the three examples in the three matrices below.

<pre> 0 0 1 1 0 0 1 1 0 2 2 2 2 2 3 3 </pre>	<pre> 0 0 1 1 0 0 1 1 0 2 2 2 2 2 3 3 </pre>	<pre> 0 0 1 1 0 0 1 1 0 2 2 2 2 2 3 3 </pre>
--	--	--

neighbour pixel value ->	0	1	2	3
ref pixel value:				
0	2	2	1	0
1	0	2	0	0
2	0	0	3	1
3	0	0	0	1

How to read the east matrix:

Twice in the test image the reference pixel has the value of 0 and its eastern neighbour is also 0. **Twice** the reference pixel is 0 and its eastern neighbour is 1. **Three times** the reference pixel is 2 and its neighbour is also 2. And so on.

Make sure you understand the difference between the original image and its pixel values, and the entries in the GLCM, which are counts of frequencies of the neighbouring pairs of image pixel values. The fact that we have low count numbers, and are using 2-bit image data, means that it might be less clear what

“0” means at first glance: trust me, it would be worse if we didn’t do this (I’ve tried!).

From here on in, we will not be using original pixel values very much, relying on the computer to do that counting work for us in a real situation. So make sure now you know that the GLCM is a table of frequencies, not an image, and you will do fine!

Properties of the GLC Matrix:

Now that we know how to construct the GLCM we can list some generalizations about it that will help us to go to the next step.

1. It is square:

The reference pixels have the same range of possible values as the neighbour pixels, so the values along the top are identical to the values along the side.

2. It has the same number of rows and columns as the quantization level of the image:

The test image is 2-bit has four (2^2) grey level values (0,1,2 and 3). Eight bit data has 256 possible values (2^8), so would yield a 256 x 256 square matrix, with 65,536 cells. 16 bit data would give a matrix of size 65536 x 65536 = 429,496,720 cells!

FAQ: Isn't that too much to handle, even for a computer?

Yes - even for 8-bit data. Most operational programs rescale the image values into 4 bit (16 x 16 matrix with 256 cells) or 5 bit (32x32 matrix with 1024 cells). The rescaling algorithms vary from one software to another, and are usually proprietary, meaning they do not say precisely how they do it.

Until about 2007, almost all image data was 8-bit. Now image data is often in 16-bit or some other value. The same problem exists, and the same solution!

There is another reason for compressing the data into 4 or 5 bit. If all 256 x 256 (or more) cells were used, there would be many cells filled with 0's (because that combination of grey levels simply does not occur on the image). The GLCM approximates the joint probability distribution of two pixels. Having many 0's in cells makes this a very bad approximation. If the number of grey levels

is reduced, the number of 0's is reduced, and the statistical validity is greatly improved. Because users often have no choice (unless writing their own algorithms), the question of the effects of quantization level is often overlooked. In practice, some statistics calculated from the GLCM don't help classification very much when a large number of grey levels are used. Other statistics don't degrade as much. See [Clausi 2002](#) for a discussion.

3. We want the GLCM to be symmetrical around the diagonal:

A symmetrical matrix means that the same values occur in cells on opposite sides of the diagonal. For example, the value in cell 3,2 would be the same as the value in cell 2,3. The matrix we calculated above is not symmetrical. However, texture calculations are best performed on a symmetrical matrix.

The matrix above counted each reference pixel with the neighbour to its right (east). If counting is done this way, using one direction only, then the number of times the combination 2,3 occurs is not the same as the number of times the combination 3,2 occurs (for example 3 may be to the right of 2 three times, but to the left of 2 only once). However, symmetry will be achieved if each pixel pair is counted twice: once "forwards" and once "backwards" (interchanging reference and neighbour pixels for the second count).

Example: A reference pixel of 3 and its neighbour of 2 would contribute one count to the matrix element 3,2 and one count to the matrix element 2,3.

Symmetry also means that when considering an eastern (1,0) relation, a western (-1,0) relation is also counted. This could now be called a "**horizontal**" matrix.

Making a matrix symmetrical in this way also neatly gets us around the problem of the window edge pixels: remember the ones on the left could never be a neighbour pixel in an east relationship, and the ones on the right could never be a reference pixel in an east relationship. But in a horizontal relationship, making the symmetrical matrix, each pixel gets to be a reference and neighbour pixel, no matter where it is in the window.

How to make the matrix symmetrical - an easier way than double counting:

Any matrix can be made symmetrical by adding it to its **transpose**. The transpose is created by interchanging the rows and columns of the original matrix.

The transpose of the eastern (1,0) matrix above could be called a "western" (-1,0) matrix. It is:

2	0	0	0
2	2	0	0
1	0	3	0
0	0	1	1

Add the original matrix to its transpose (by adding each element in the 2 matrices) for this "horizontal" result. It is symmetrical.

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

Note that we have left off the column and row labels for these GLCMs. We will continue to do this.

If you are sceptical, perform the eastern and western counts separately, and verify that the GLCM produced that way is the same as this matrix.

Expressing the GLCM as a probability:

Is it more likely to find a horizontal combination of, say, 2,2 in the original image, or is 2,3 more likely? Looking at the horizontal GLCM shows that the combination 2,2 occurs 6 times out of the 24 horizontal combinations of pixels in the image (12 eastern + 12 western). In other words, 6 is the entry in the horizontal GLCM in the third column (reference pixel value 2) and third row (neighbour pixel value 2). The simplest definition of the probability of a given outcome is

"the number of times this outcome occurs, divided by the total number of possible outcomes."

Using this definition, we can calculate: the combination (2,2) occurs in 6 cells out of 24, for a probability of $6/24 = 1/4$ or 0.250. The probability of (2,3) is $1/24$ or .042.

Here is the equation to transform the GLCM into a close approximation of a probability table: It is only an approximation because a true probability would require continuous values, and we are dealing here with discrete frequency values, which can only be integers. So, this process is called **normalizing** the matrix. Normalization involves dividing by the sum of values. It differs from probability only in a formal sense.

$$P_{i,j} = \frac{V_{i,j}}{\sum_{i,j=0}^{N-1} V_{i,j}}$$

Normalization equation:

where:

- i is the row number and j is the column number.

Note: i and j keep track of cells by their horizontal and vertical coordinates. The range of summation, $(i,j=0)$ to $(N-1)$ means simply that each cell in the GLCM should be considered. It is shorthand for a double summation, once from $i=0$ to $N-1$ and once from $j=0$ to $N-1$. Usually, a count starts with "1" so summation from 1 to N would be expected. However, by numbering the upper left corner cell as $i=0$ and $j=0$, rather than as $i=1$ and $j=1$, the i value remains the same as the actual grey level of the reference cell, and the j value remains the same as the grey level of the neighbour cell. This is not important for many equations, but it comes in very handy when expressing mean, variance and correlation in terms of the GLCM (explained below).

- V is the value in the cell i,j of the image window
- P_{ij} is the probability value recorded for the cell i,j . **Remember this notation, it will occur often later on!!**
- N is the number of rows or columns

Applying this equation to the symmetrical GLCM above yields:

Normalized (horizontal) GLCM

.166 (4/24)	.083 (2/24)	.042 (1/24)	0 (0/24)
.083	.166	0	0
.042	0	.250	.042
0	0	.042	.083

Summary of steps in creating a symmetrical normalized GLCM:

1. **Create a framework matrix taking into account the bit depth**
2. **Decide on the spatial relation between the reference and neighbour pixel**
3. **Count the occurrences and fill in the framework matrix**
4. **Add the matrix to its transpose to make it symmetrical**
5. **Normalize the matrix to conceptually turn it into probabilities.**

Some things to notice about the normalized symmetrical GLCM (called simply the GLCM from here on)

- **The diagonal elements all represent pixel pairs with no grey level difference (0-0, 1-1, 2-2, 3-3 etc.).** If there are high probabilities in these elements, then the image does not show much contrast: most pixels are identical to their neighbours.

Framework GLCM showing the values of reference and neighbour pixels
Violet shown cells where the two values are the same

neighbour pixel value ->	0	1	2	3
ref pixel value: ↓				
0	0,0	0,1	0,2	0,3
1	1,0	1,1	1,2	1,3
2	2,0	2,1	2,2	2,3
3	3,0	3,1	3,2	3,3

When counts on the diagonal are summed, the result is the probability of *any* pixel's being the same grey level as its neighbour.

- Look at lines parallel to the diagonal. Cells one cell away from the diagonal represent pixel pairs with a difference of only one grey level (0-1, 1-2, 2-3 etc.). Similarly, values in cells two away from the diagonal show how many pixels have 2 grey level differences, and so forth. **The farther away from the diagonal, the greater the difference between pixel grey levels.**

Framework GLCM showing the values of reference and neighbour pixels

Violet: no difference; Green, difference of 1; Yellow: difference of 2; Black: difference of 3

neighbour pixel value ->	0	1	2	3

ref pixel value: ↓				
0	0,0	0,1	0,2	0,3
1	1,0	1,1	1,2	1,3
2	2,0	2,1	2,2	2,3
3	3,0	3,1	3,2	3,3

Sum up the counts of entries in these parallel diagonals and the result is the probability of any pixel's being 1 or 2 or 3 etc. different from its neighbour.

1. **Exercise:** use the test image and a south spatial relationship (reference pixel and the neighbour below it) to test understanding. Fill in the blanks.

Test image

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Framework matrix:

Count (south) matrix

+

Transpose (north) matrix

=

Symmetrical (vertical) matrix

--	--	--	--

divided by the sum of the elements =

Normalized symmetrical vertical GLCM			

Answer: [click here](#)

Sometimes texture measures use a **GLDV** (grey-level difference vector) instead of a GLCM. The GLDV is the sum of the diagonals of the GLCM.

Example: using the horizontal GLCM, the GLDV is either the middle or, when normalized, the right hand row:

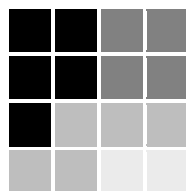
Horizontal GLCM:

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

difference of:	occurs this number of times	normalized, equals
0	16	.666
1	6	.250
2	2	.083
3	0	0

In words and visual concepts, this means that we have an image that is dominated by pixels that are similar or identical to their neighbours. There are only a few abrupt changes in tone from one to the other. Here is the original image again: see if you can relate the statement above to the spatial arrangement of grey levels in the image. After all, if descriptive statistics are going to be useful they have to describe something we can also visually grasp.

The next step will be, therefore, to reduce the table above to a single number that will summarize and quantify the qualitative perception of “smoothness” in this image. We will find that there are several ways to do this, resulting in several different descriptive statistics.



2. Exercise: Calculate the **GLDV** (not the GLCM) for the **vertical** relationship. This will require you to go through all the steps shown for exercise 1, this time using

the vertical relationship: if you need to refer back to that, it might help. [Click here to link to the answer for exercise 1.](#) It is only in the final step that you sum up the diagonals to create the GLDV.

A difference of	occurs this number of times	normalized

Answer: [click here](#)

THE TEXTURE MEASURE CALCULATIONS

Texture measures are the various single values used to summarize the normalized symmetrical GLCM in helpful ways. These are the things most often referenced in the literature: “Contrast”; “Dissimilarity”; “Entropy” etc. There are several of them because each summarizes in a different way, allowing choice to fit the problem at hand. This section shows how these are calculated.

Most texture measure calculations are **weighted averages** of the normalized GLCM cell contents.

A weighted average multiplies each value to be used by a factor (a weight) before summing and dividing by the number of values. The weight is intended to express the relative importance of the value.

Example: the most common weighted average that students encounter is the term grade. Exams usually have a higher weight than quizzes. The weights are the % of course grade assigned to each mark.

Creating a texture image

WHY create a texture image? So far we have addressed the question of the GLCM using all the pixels in a small test image. This is to make the hand counts and calculations easy to follow. But the true usefulness of descriptive statistics comes out when dealing with a large image, and using a computer to generate the numbers.

In remote sensing and allied fields such as medical imaging, we usually don't want a single measure for a whole image. Instead, we want to see how the pixel-to-pixel relationships might be different in different parts of the image. For example, we might visually describe a deciduous forest as “bumpy”, and grassland as “smooth”, and rock as “jagged”. Suppose we designed a texture measure to translate each of these words into a number. Let's say we come up with a statistic that is high over grassland areas, low over rocks, and intermediate over forests. We could then use texture as input into an automated classification algorithm. To do this, though, we have to limit the texture measure calculation to a GLCM derived from a small areas on the image. We then look at a different small area and record its texture measure, and so on to cover the whole image. This way, the measure will be different in different places and tell us quantitatively how the pixel relationships differ in different places. This is the reason for calculating the “texture image”.

FAQ: Why would you NOT create a texture image? Answer: Some software that is oriented towards statistical procedures does not calculate the image automatically. It is quite possible to input matrices either prepared by hand, or chosen from separated parts

of an image, and compare the output texture measures. A texture image is not always necessary, just make sure you know what output you need and want for your research or work situation.

How the texture image is calculated:

Defining the neighbourhood area

First, we need to define the small area to use for filling in the GLCM and doing the texture measure calculation. In a general sense, we can call this the “**neighbourhood**” for the calculation. Usually we use the term “**window**” which is a particular kind of neighbourhood. It is **square**, and it has an **odd number of pixels** on a side.

How the window size (in pixels) is chosen is a whole other topic. See that section in this tutorial!

Writing the output:

The result of a texture calculation is a single number representing the entire **window**. This number is put in the place of the centre pixel of the window.

Building up the whole texture image:

Once the first calculation is completed and the result written into that pixel in a new raster layer, the window is moved one pixel to the right and the process is repeated of calculating a new GLCM and a new texture measure. This continues across the first row, then the window is moved down one row and the process is repeated. This continues until the entire image has been covered. In this way an entire new image layer (raster) is built up of texture values. [More information](#)

Summary of steps in the image

1. Decide on window size
2. Place window in first position over top left of the image.
3. For the pixels within this window in this position, count the pixels, transform and normalize the GLCM
4. Calculate the texture measure of your choice (see below)
5. Move the window over one pixel, and repeat steps 3 and 4.
6. Continue covering all possible window positions until the texture image is complete.

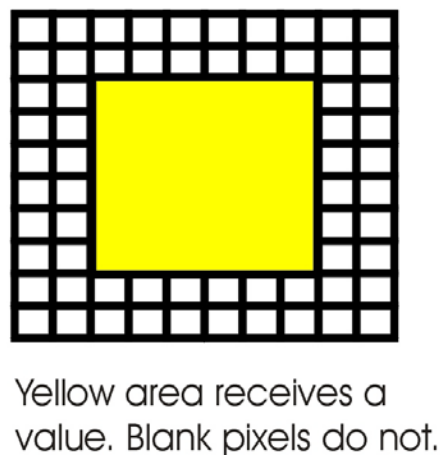
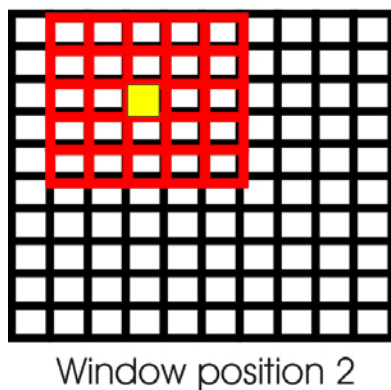
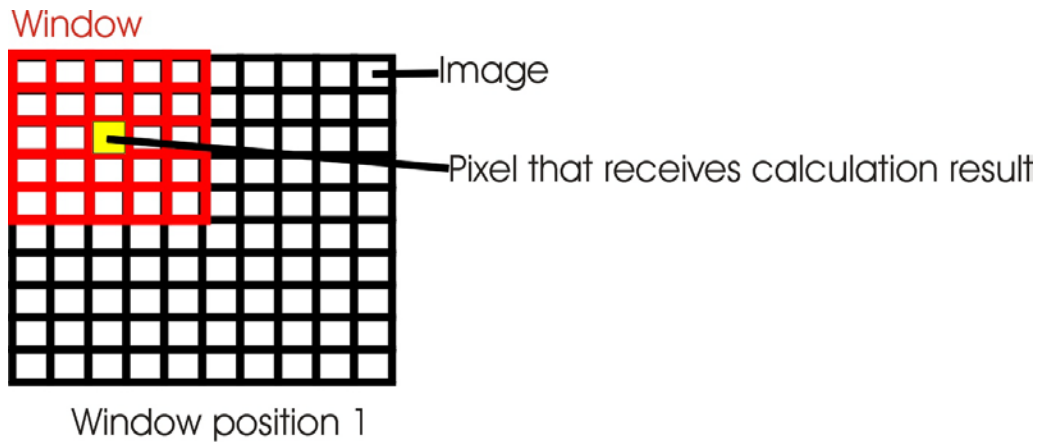
FAQ: How long does this take? There is not too much problem with today's computers, however it still is a computationally intensive procedure. Do not be alarmed if creating a texture image takes a few minute. Depending on your computer and software, it might be faster or slower. This computational intensity is a reason why many years passed between Haralick's 1973 mathematical consideration of texture and its common employment in automated image analysis.

Other things to think about

Edge of image problems Each cell in a window must sit over an image pixel containing a GL value. This means that the centre pixel of the window cannot be an edge pixel of the image. If a window has dimension $N \times N$, a strip $(N-1)/2$ pixels wide around the image will remain unoccupied.

FAQ: So what does a computer program do with edges? Answer: The usual way of handling this is to fill in these edge pixels with the nearest texture calculation, thus having a repeated uniform border. Remote sensing images have thousands of pixels and rows. Texture uses windows that are usually no larger than about 51×51 pixels. Therefore the edges occupy a very small part of the resulting texture image, and can be disregarded. This edge problem only is of concern when the size of the window is similar to the size of the whole image. This can occur in texture applications to things like industrial quality control. In these cases, individual solutions must be sought to fit the problem at hand.

Example: For a 5×5 window, the outer 2 rows and columns of the image receive the texture values calculated in row 3 (top edge), column 3 (left edge), row $L-2$ (bottom edge) and column $P-2$ (right edge) where P, L are the dimensions in pixels (columns) and lines (rows) of the original image. For the illustrated image below, $L=P=10$, so values are calculated through rows 3 through 8 and columns 3 through 8.



Groups of texture measures

This tutorial groups the texture measures according to the **purpose of the weights** in the equations.

*Another common way to classify textures is according to their **degree**, meaning the highest exponent used. Most measures - and all used here - are first or second degree. This tutorial does not make degree the main classification criterion for entirely practical purposes. The intent here is to relate the texture measures to their ability to approximate a visual distinction between textures.*

*Remember that all GLCM measures are “second **order**” meaning that they calculate the measure from the GLCM (pairs of pixels). “First order” measures would calculate something (for example standard deviation) from the original pixel values themselves within a window, not from the GLCM values. It is easy to get these confused when reading the literature, and indeed it*

is not too clear in some software help files.

Example: If a squared term is used in the equation, the measure is second degree. If a cubed term is used, it is third degree. This is similar to non-GLCM descriptive statistics, where mean is first order, variance is second order, and so on.

THE GROUPS USED HERE

1. Contrast group

Measures related to *contrast* use weights related to the *distance from the GLCM diagonal*.

Principle: To emphasize a large amount of contrast, create weights so that the calculation results in a larger figure when there is great contrast between adjacent pixels. Values on the GLCM diagonal show no contrast, and contrast increases away from the diagonal. So, create a weight that increases as distance from the diagonal increases.

1. A. Contrast (this is also called "sum of squares variance" and occasionally "inertia"):

$$\sum_{i,j=0}^{N-1} P_{i,j} (i-j)^2$$

Contrast equation

Explanation:

When i and j are equal, the cell is on the diagonal and (i-j)=0. These values represent pixels entirely similar to their neighbour, so they are given a weight of 0 (no contrast).

If i and j differ by 1, there is a small contrast, and the weight is 1.

If i and j differ by 2, contrast is increasing and the weight is 4.

The weights continue to increase exponentially as (i-j) increases.

Calculation example: for the horizontal GLCM

Contrast weights:				X	horizontal GLCM				=	Multiplication result			
0	1	4	9		0.166	0.083	0.042	0		0	0.083	.168	0
1	0	1	4		0.083	0.166	0	0		0.083	0	0	0
4	1	0	1		0.042	0	.249	0.042		.168	0	0	.042
9	4	1	0		0	0	0.042	0.083		0	0	.042	0

Sum of all elements in the multiplication result table = **0.586**. Below is the actual computation as it would be entered in a spreadsheet or carried out by hand.

$$\begin{aligned}
 &.166*(0-0)^2 + .083*(0-1)^2 + .042*(0-2)^2 + 0*(0-3)^2 + \\
 &.083*(1-0)^2 + .166*(1-1)^2 + 0*(1-2)^2 + 0*(1-3)^2 + \\
 &.042*(2-0)^2 + 0*(2-1)^2 + .250*(2-2)^2 + .042*(2-3)^2 + \\
 &0*(3-0)^2 + 0*(3-1)^2 + .042*(3-2)^2 + .083*(3-3)^2 \\
 &= .166(0) + .083(1) + .042(4) + .083(1) + .166(0) + .042(4) + .25(0) + \\
 &.042(1) + .042(1) + .083(0) \\
 &= .083 + .168 + .083 + .168 + .042 + .042 \\
 &= \mathbf{.586}
 \end{aligned}$$

Important practical matter: This example shows right away that Contrast can be <1. Therefore the computer output **must be recorded in an image channel (raster) equipped to handle real numbers, usually 32-bit Real (32R)**. If put into an 8-bit or 16 bit integer channel, the value would be recorded as 0.

***FAQ: How does this relate to the bit depth?** Answer: This has nothing to do with the reduction of bit depth described above, which relates to the original bit depth of the received imagery. 32R is fine for the purpose here. If for some reason an integer channel is desired for the texture measure output (for example to meet software-specific input requirements for multi-band classification), automatic scaling algorithms are available in most software. Be careful what you do so as not to lose information!*

3. Self test: What is the degree of this measure? What does a Contrast of 0 mean? [Answer](#)

4. Exercise: Calculate the Contrast for the vertical GLCM and compare it with the Contrast for the horizontal GLCM [Answer](#)

1. B. Dissimilarity (Contrast Group)

Instead of weights increasing *exponentially* (0, 1, 4, 9, etc.) as one moves away from the diagonal as Contrast did, the dissimilarity weights increase *linearly* (0,1,2,3 etc.).

$$\sum_{i,j=0}^{N-1} P_{i,j} |i - j|$$

Dissimilarity equation

This is a first degree measure. Why?

Dissimilarity weights:

0	1	2	3
1	0	1	2
2	1	0	1
3	2	1	0

Exercise 5: Try out the Dissimilarity calculation for the horizontal image and compare the value with horizontal Contrast. Also, compare horizontal with vertical Dissimilarity. Is the pattern the same as for horizontal vs vertical Contrast? [Answer](#)

1. C. Homogeneity (Inverse Difference Moment) (Contrast group)

Dissimilarity and Contrast result in *larger* numbers for more windows showing more contrast. If weights *decrease* away from the diagonal, the calculated texture measure will be larger for windows with little contrast. Homogeneity weights values by the *inverse* of the Contrast weight, with weights decreasing exponentially away from the diagonal:

$$\sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1 + (i-j)^2}$$

Homogeneity equation

Homogeneity weights:

1	½	¼	1/9
½	1	½	¼
¼	½	1	½
1/9	¼	½	1

6. Exercise: Calculate the Homogeneity value for the horizontal GLCM and compare it with the Dissimilarity value. [Answer](#)

7. Self test: The weight used for the Contrast texture measure is $(i-j)^2$. The weight for Homogeneity is $1/[1+(i-j)^2]$.

- What would happen if the Contrast weight were $[1+(i-j)^2]$? This might seem logical when we describe the Homogeneity weights as being the inverse of the Contrast weights.
- Would this be a bad thing?
- What degree is this measure?

[Answer](#)

8. Exercise: Do-it-yourself Similarity texture:

Homogeneity is the most commonly used measure that increases with less contrast in the window. However, it would be easy to use the above model to construct a first degree "similarity" measure. Write the equation and perform the calculation for the horizontal GLCM.

Why do you think "similarity" is not a commonly programmed measure (there is no single good answer to this, by the way! It's just to help you think it through.).

[Answer](#)

2. Measures related to *orderliness*




Orderliness means: how regular (“orderly”) the pixel value *differences* are within the window.

Example: the two images below have the same degree of horizontal contrast (every pixel is one less than its eastern neighbour). But the degree of order is quite different.

Orderly image					Disorderly image			
1	2	3	4		3	4	2	3
1	2	3	4		1	2	3	4
1	2	3	4		2	3	4	5
1	2	3	4		4	5	6	7

In the more orderly image on the left, each pair of values occurs many times: 2 is next to 1 four times, 3 is next to 2 four times, etc. For the less orderly image, combinations occur less often: 2 is next to 1 only once, 3 next to 2 three times, and so on. *Using 3-bit data on the disorderly image does not affect the texture measure: it simply makes the visual difference more apparent.*

For a visual appreciation of this, here are the grey levels shown for each of these images:

Orderly image					Disorderly image			
								
								
								
								

Principle behind orderliness measures:

Orderliness measures, like contrast measures, use a weighted average of the GLCM values. The weight is constructed related to *how many times a given pair occurs*, so

A weight that *increases* with commonness will yield a texture measure that *increases* with orderliness.

A weight that *decreases* with commonness yields a texture measure that increases with *disorder*

Weights:

Since the P_{ij} values in the GLCM are already a measure of commonness of occurrence, it makes sense to use them in some form as weights for themselves. Thus the weights are derived from each GLCM itself and cannot be shown in a table.

2. A. Angular Second Moment (ASM), Energy and MAX

ASM and Energy use each P_{ij} as a weight for itself. High values of ASM or Energy occur when the window is very orderly.

$$\sum_{i,j=0}^{N-1} P_{i,j}^2$$

ASM equation

The square root of the ASM is sometimes used as a texture measure, and is called **Energy**.

$$Energy = \sqrt{ASM}$$

Energy equation

The terms ASM and Energy

The name for ASM comes from Physics, and reflects the similar form of Physics equations used to calculate the angular second moment, a measure of rotational acceleration. If you are interested, look up "moment of inertia" in a Physics textbook. The meaning of "energy" is explained [below](#).

9. **Exercise:** Perform the ASM calculation for the horizontal GLCM. [Answer](#)

10. **Self test:** The maximum value of 1 for either ASM or Energy occurs when all pixels in the window are identical. This is as orderly as an image can get – but doesn't tell us much! Such an image would look more like a fog that we can't see anything through. Quickly draw the GLCM for this situation and perform the ASM calculation. [Answer](#)

Maximum Probability (MAX)

A simple statistic is simply a statement of the largest P_{ij} value found within the GLCM. High MAX values occur if one combination of pixels dominates the pixel pairs in the window. MAX is not commonly implemented in image processing software

2. B. Entropy

Since $\ln(0)$ is undefined, assume that $0 * \ln(0) = 0$:

$$\sum_{i,j=0}^{N-1} P_{i,j} \left(-\ln P_{i,j} \right)$$

Entropy equation

Entropy is usually classified as a first degree measure, but should properly be a "zerth" degree!

Logarithms: a brief refresher tutorial (ln or log)

- **ln** is the "natural logarithm" and uses a base close to 2.718; **log**, with a base of 10, is more familiar. The properties of the two are the same.
 - **example:** $\log(10) = 1$. Read this as "the log of 10 equals 1." This means $10^1 = 10$; $\log(100) = 2$. This means $10^2 = 100$.
 - Generalizing, if $\log(x) = n$, then $10^n = x$.
 - Similarly, if $\ln(x) = n$, then $2.718^n = x$
- **$\ln(1) = 0$.**
- **ln of values between 0 and 1 are negative.**
 - ln becomes increasingly negative as the reference value decreases further below 0
 - **example**

$\ln(0.99)$	-0.010
$\ln(0.10)$	-2.302
$\ln(0.001)$	-6.908

log or ln of 0 and of negative numbers are not defined.

Technically, the limit of $\ln(x)$ as x approaches 0 is negative infinity. But a calculator will balk at encountering a calculation involving $\ln(0)$. This is why it is necessary to state the assumption that $0 \cdot \ln(0) = 0$ in the Entropy equation. *If manually calculating \ln of a pixel value in a software's image arithmetic (or similar) function, make sure that this is the default (see help file for your software) or enter it as appropriate. The default might also yield a "no data" value on input pixels with a value of 0. See if this will cause you problems – 0 pixels are often edges outside the image area anyway.*

It is easy to confuse original 0 values on the image (border areas, for example, or codes) with 0 values in the GLCM (a reference and neighbour pixel combination simply has a count of 0). Thus for the GLCM, an entry of 0 is quite possible, even common. The Entropy equation refers to the GLCM entries, not the 0 pixel values. The GLCM cannot have negative entries, so this additional specification of what to do with $\ln(0)$ is quite adequate.

What using the \ln means for the GLCM Entropy equation:

- P_{ij} is always between 0 and 1, because it is a probability and a probability can never be higher than 1 nor less than 0.
- Therefore **$\ln(P_{ij})$ will always be 0 or negative.**
- The smaller the value of P_{ij} (i.e. the less common is the occurrence of that pixel combination), the larger is the *absolute value* of $\ln(P_{ij})$
- the (-1) multiplier in the entropy equation makes each term positive.
- Therefore, the smaller the P_{ij} value, the greater the weight, and the greater the value of $-[P_{ij} \cdot \ln(P_{ij})]$.
- This can be confusing, as can anything where negatives are used, but even if you choose not to work this out in detail for yourself, be assured that the overall effect is to make larger Entropy values indicate greater disorder, and appear brighter on the entropy texture image!

FAQ: I got an Entropy output of 3.89, and I don't think that is possible using this equation. Is there an error in my software? Answer: Where the equation by its nature limits the possible texture measure values, sometimes a software output can be startling, as it falls outside of that range. This is most often simply due to a final output auto-scaling that is included in the algorithm, generally in the interests of visualization. Often, there is no indication of scaling in software help files or other documentation. For many years, there were so few software packages that calculated this. It never seems to have occurred to developers that someone might want to compare values of this data-dependent descriptive statistic across platforms. Given this, avoid problems by only compare actual texture measure values calculated using the same software and inputs. The texture image is normally used to compare texture differences in different parts of the image anyway. However when adapting this procedure to disciplines outside of image processing, it can become hairy!

The term "entropy"

Entropy is a notoriously difficult term to understand; the concept comes from thermodynamics. It refers to the quantity of energy that is permanently lost to heat ("chaos") every time a reaction or a physical transformation occurs. Entropy cannot be recovered to do useful work. Because of this, the term is used in non technical speech to mean irremediable chaos or disorder.

Also, as with ASM, the equation used to calculate physical entropy is very similar to the one used for the texture measure called Entropy.

Energy is, in this context, the opposite of entropy. Energy can be used to do useful work. In that sense it represents orderliness. This is why "Energy" is used for the texture that increases with increasing order in the image.

The entropy equation can also be used in Information Science to indicate the amount of information in a system. I am not well enough informed to say much more than that. However, again qualitatively, remember that the perfectly orderly image- with all pixel values identical, no matter what their grey level value is – resembles a dense fog, visually. Only one piece of information is there – there is no variability. Visually, it is variability that gives us access to the image information that we might want. So greater and more complex variability is in fact likely to yield more useful information – and a higher entropy value indicates more complex variability. Depending on our research question, this might be information or it might be noise: more work is necessary to find out which! If you want to isolate more complex areas from less complex areas, entropy is right down your alley.

This seems to run counter to the physics-based idea that "entropy is energy you can't employ for useful work", but in this case that is not quite the way we would

want to express it. Don't worry, the equation will give the information the image analyst needs, whether the term makes intuitive sense or not!

11. Exercise: Calculate the Entropy value of the horizontal GLCM. [Answer](#)

3. Descriptive Statistics of the GLCM texture measures

The third group of texture measures uses equations similar to those for common descriptive statistics, such as mean or standard deviation (or variance). However, all are calculated using the entries in the GLCM, not the original pixel values. Details below. The important thing is to be clear when using software that you are calculating the GLCM, not the first-order statistic. Many programs contain algorithms to calculate the first-order statistic as well. Also, be clear when writing a report that your terminology is always clear so that the reader will understand that you are using the GLCM statistic.

3. A. GLCM Mean

How GLCM mean differs from the mean of the pixel values in the window:

The GLCM Mean is not simply the average of all the original pixel values in the image window. It is expressed *in terms of the GLCM*. The pixel value is weighted not by its frequency of occurrence by itself (as in a "regular" or familiar mean equation) but by its frequency of its occurrence *in combination with* a certain neighbour pixel value.

GLCM Mean Equation

$$\mu_i = \sum_{i,j=0}^{N-1} i(P_{i,j}) \quad \mu_j = \sum_{i,j=0}^{N-1} j(P_{i,j})$$

The left hand equation calculates the mean based on the reference pixels, μ_i . It is also possible to calculate the mean using the neighbour pixels, μ_j , as in the right hand equation. **For the symmetrical GLCM**, where each pixel in the window is counted once as a reference and once as a neighbour, **the two values are identical**. The two equations are included for mathematical and theoretical clarity, just in case a symmetrical GLCM is not being used.

Calculation and notation details:

- The summation is from 0 to (N-1), not from 1 to N. Since the first cell in the upper left of the GLCM is numbered (0,0), not (1,1), the i value (0) of this cell is the same as the value of the reference pixel (0). Similarly, the *second* cell down from the top has an i value of 1, and a reference pixel value of 1. If this is not clear, go back and look at the framework GLCM.
- The P_{ij} value is the probability value from the GLCM, i.e. how many times that reference value occurs *in a specific combination with a neighbour pixel*. It is not a measure of how many times the reference pixel occurs, period, which would be the "regular" first-order mean for the original window. First-order mean is a tool for smoothing an image to remove random noise, and also of systematically degrading the spatial resolution.
- Multiplying i by P_{ij} effectively divides the entry i by the sum of entries in the GLCM, which is the number of combinations in the original window. This is the same as is done when calculating a mean in the "usual" way. If this is not clear, review how P_{ij} is calculated in the first place.
- The GLCM Mean for the horizontal GLCM is different from that for the vertical GLCM because the combinations of pixels are different in the two cases. If you are using a "non-directional" texture measurement, then the software will simply calculate both and average them together. The horizontal or vertical are referred to here to make the procedures more understandable.

12. Exercise: Calculate the GLCM Mean for the horizontal test image then for the vertical test image. [Answer](#)

3. B. Variance (Standard Deviation) (Descriptive statistics group)

Variance equation

$$\sigma_i^2 = \sum_{i,j=0}^{N-1} P_{i,j} (i - \mu_i)^2 \quad \sigma_j^2 = \sum_{i,j=0}^{N-1} P_{i,j} (j - \mu_j)^2$$

Standard deviation equation

$$\sigma_i = \sqrt{\sigma_i^2} \quad \sigma_j = \sqrt{\sigma_j^2}$$

Calculation details:

Variance in texture performs the same task as does the common descriptive statistic called variance. It relies on the mean, and the dispersion around the

mean, of cell values within the GLCM. However, GLCM variance uses the GLCM, therefore it deals specifically with the *combinations* of reference and neighbour pixel, so it is not the same as the simple variance of grey levels in the original image.

Variance calculated using i or j gives the same result, since the GLCM is symmetrical.

There is no particular advantage to using Standard Deviation over Variance, other than a different range of values.

Properties of Variance

Variance is a measure of the dispersion of the values around the mean. It is similar to contrast or dissimilarity. It answers the question "What is the dispersion of the difference between the reference and the neighbour pixels in this window?"

It is not intuitively visually obvious what this might mean. Neither is it obvious from looking at an image how GLCM Variance will behave. It seems to have quite different values for different textures, however, and so can fulfill the function of telling classes with different textures apart. There is no way of knowing in advance whether the classes of interest in a particular case will be distinguished from one another by GLCM Variance, so trial and error may be in order. For a somewhat more in-depth discussion of the role of GLCM Variance (and the other Descriptive Statistical textures in general) see M. Hall-Beyer 2017, [Practical guidelines for choosing GLCM textures to use in landscape classification tasks over a range of moderate spatial scales](#), IJRS vol. 38.

13. Exercise: Calculate the Variance texture for both the horizontal and vertical GLCM of the test image to see if they are the same.

What is the variance calculated using the original image values rather than using the GLCM?

[Answer](#)

3. C. Correlation (Descriptive statistics group)

The Correlation texture measures the linear dependency of grey levels on those of neighbouring pixels.

$$\sum_{i,j=0}^{N-1} P_{i,j} \left[\frac{(i - \mu_i)(j - \mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right]$$

Correlation equation

For the symmetrical matrix, the denominator above reduces to the variance (sigma squared).

Understanding how to use this equation depends on understanding the [Mean](#) equation and [Variance](#) equation expressed in terms of the GLCM.

What does correlation mean?

Correlation between pixels means that there is a predictable and linear relationship between the two neighbouring pixels within the window, expressed by the regression equation.

Example: Suppose there is a very high correlation between the reference and neighbour pixel, expressed by $n=2r+2$, where n is the value of the neighbour and r of the reference. Therefore, if $r=1$, n is very likely to equal 4; if $r=4$, $n=10$, etc.

*A high Correlation texture means high predictability of pixel relationships. This gives rise to some interesting **properties of Correlation**:*

- *Single patches of a particular ground cover usually have a higher correlation value within them than between adjacent objects.*
- *Pixels are usually more highly correlated with pixels nearby than with more distant pixels (Spatial Autocorrelation).*
 - *So, smaller window sizes will usually have a higher Correlation value than larger windows.*
- *If Correlation is calculated for successively larger window sizes, the size at which the Correlation value declines may be taken as a rough indication of the size of definable objects within an image. This works only if all objects in the image are about the same*

size. Often different classes (for example ground covers) on an image will have different patch (object) sizes – for example grasslands and lakes.

- People familiar with semivariograms will see a resemblance to the information provided by them and by GLCM Correlation.
- Correlation is quite a different calculation from the other texture measures described above. As a result, it is **independent** of them (gives different information) and can often be used profitably in combination with another texture measure
- See more discussion in Hall-Beyer 2017 [Practical guidelines for choosing GLCM textures to use in landscape classification tasks over a range of moderate spatial scales](#). Correlation behaves in an unpredictable way in much the same way that GLCM Variance does.

14. Exercise: Calculate the Correlation measure for the horizontal test image. Do the [GLCM Mean](#) and [Variance](#) exercises first and use their results. Click on the name of each to link to its answer.

Hint: Correlation must always be between -1 and +1 before scaling. It will be a decimal value. Any result outside of these bounds shows an error. It is easy to make calculation errors. If you are familiar with setting up formulae on a spreadsheet, this would be a good way to do the calculation. On a spreadsheet, you can at least be certain that there have been no arithmetic errors, even though it is still easy to make an error in entering the formula.

[Answer](#)

Some very important practical considerations

Texture is a descriptive statistic, without hard values that can be transferred from one situation to another. In other words, there is no way of saying “forests always have Contrast values between .5 and .7”. It is primarily useful in comparing one part of an image to another part. If more than one image is to be included (perhaps by mosaicking), all the usual radiometric corrections should be carried out before mosaicking, and the texture run afterwards. All the usual caveats of multiple image analysis must be taken into consideration: the images analysed must be equivalent radiometrically, in regards to sun angle, and phenologically with regards to cyclically variable ground phenomena. Check out the change analysis literature for details.

It is also important to realize that the analyst must make quite a few decisions before starting the texture analysis, if it is to be most efficient at answering the question at hand and avoid misinterpretation. Some of these, and pointers for best practice, are listed below.

Textures may be calculated using only one channel of data at a time.

- Many different texture measures can be calculated for each channel
- Channel information can be consolidated using an index (for example NDVI), principal components (PCA) or other method before running a texture measure.
- The same texture measures can be calculated for more than one channel
- Any three texture measures can be viewed simultaneously using rgb projection; numerical analysis generally does not limit one to using only three channels
- It is possible to take the principal components of many texture images, but the result is very difficult to interpret. See Hall-Beyer 2017.

Texture images are image channels (rasters) with a value for each pixel.

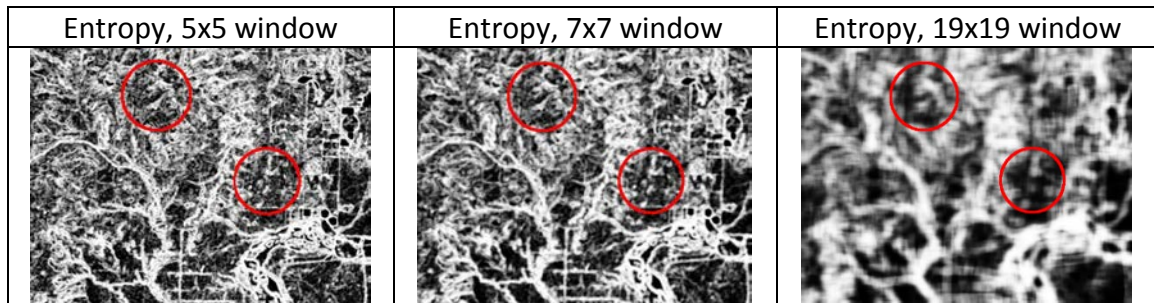
- Texture images can be used alone or with other data to define signatures for supervised or unsupervised classification.
- Texture images (channels) can be included in classifications.
- Texture output should be put into a 32R channel, at least at first. It may then be scaled into an 8-bit channel.
- The actual numerical value of a texture measure is unimportant for most purposes. More important is whether it is relatively high or relatively low compared to the other areas of the image, and its relationship to the same texture measure values elsewhere on the image.

Textures are calculated within a *window*, a small region of the image.

The test image used in this tutorial considers the entire image to be the area contributing to the texture. For larger images, a window is chosen to define this area. This window is, in practice, square and with odd numbered side lengths. In theory, a window can be any dimension, but again, practical calculation problems occur for even sizes and non square shapes of windows.

- The relative size of the window and the objects in the image will determine the usefulness of the texture measure for classification.
- It is expected that different objects will have different characteristic texture measures. To capture this, the window must be smaller than the object, but big enough to cover the characteristic variability of the object
 - **Example:** The texture of a forest is determined by light and shadow among tree crowns. A window covering one tree will not measure the texture of the forest. A window covering the entire forest and the fields next to it will not measure the texture of the forest either.
- Windows used for texture, like those used with filters, always have an "edge effect" where the window overlaps the border between distinct objects on the image.

To see the edge effect of window size, compare these images. They are all entropy calculations, and use the same parameters, except to vary window size.



The “edges” appear white or pale. This is because a window partly over one natural texture and partly over another – along the edge between them – will very likely be less ordered than a window entirely inside a patch of similar texture. Less ordered windows write a higher entropy result to the texture image, and when viewed on a screen higher values are lighter in tone.

Click to return to the section on information about [window edges](#).

Co-occurrence of two pixels: some basic ideas

Spatial relation: The relation between reference and neighbour may be in any one of 8 directions (N, S, E, W, or the four diagonals). Only half of these are actually used, since W is the opposite of E and there are simpler ways to account for W than counting it separately. "Spatially invariant" relations may be chosen by counting in four directions (N, NE, E, SE) and then summing the counts.

Why only two pixels? It is in theory possible to choose three or more pixels in a given relation (for example, a reference pixel, its neighbour to the right and also its neighbour to the NE). However, this becomes extremely unwieldy for calculations and is not an operational procedure. Besides, experience has shown that the simpler texture measures are generally more useful than are more complex ones in terms of improving classification accuracy. A calculation involving three pixels would be "third order" and so forth.

There is a very large number of possible texture measure calculations within software GLCM algorithms: how to choose

Note for the person not working with remotely sensed images: This section applies only to large images where texture is run using one of the standard image processing software algorithms. It is possible to calculate texture for any given input matrix of numbers using statistical software. It is possible to do the same using a spreadsheet for one's own purposes. However these methods will input one matrix and give one number as output for each measure. They do not "move" the window, and do not build up a texture image that can be used in conjunction with spectral channels. If your purposes are strictly to understand the calculations, or for some reason only need one measure for one matrix, then this section is not needed. If you only have access to such one-input software, you will need to make your own app to ingest the series of matrices and output the series of results in the correct spatial configuration.

Texture measurement requires the choice of

- **window size,**
- **direction of offset,**
- **offset distance,**
- **which channel to run,**
- **which measure to use.**

***FAQ: Why is this a problem?** Answer: It would be conceptually reasonable to have, for example, 7 window sizes, 5 directions, 3 distances, 6 channels and 10 measures, for a total of 6,300 measures for a single TM image! This is clearly impractical – and even then would not represent all possibilities. Besides, many of these channels would be highly correlated with one another, thus biasing and procedure they were used with. Thus we are left with either taking defaults, random choice, or some kind of heuristics to decide which to use. Hall-Beyer (2017) explores a limited but representative number of these choices among which measures for differing window sizes and channels, and demonstrates a heuristic that is derived from understanding of the GLCM procedure. To see how this is done, refer to that article.*

However, there are some rules of thumb, which can at least be an improvement over pure random or default choice.

- Visual examination of the image channels can help eliminate some of them. Can you see different textures better in some channels than in others? For example, for vegetation, the nir and red channels would be most useful, or some combination of them such as a vegetation index. This is because there is a functional relationship between vegetation identification and the red and nir channels. Principal components can also reduce channel numbers. If this rule is used in the example above, it will reduce 6 Landsat

channels or 4 r,g,b,nir image channels, to one. That would reduce the possibilities in the example above to 1050.

- Visual examination will also show any directionality that is likely to be important on the image. If there is none, "spatially invariant" (average of all directions) is the best choice. By eliminating 4 directions, possibilities are reduced to 210.
- A distance between pixels is almost always 1 pixel. By eliminating 2 distances, we are down to 70.
- Many of the texture measures are correlated with one another. There are really only at most 4 or 5 truly independent textures (see [correlation](#)). By reducing to 4 measures we are down to 28 possibilities. How to do this? Calculate and look at correlation statistics, choosing one of a highly correlated group, or calculate and perform PCA on them and use only the first 3 or 4 components, or check out [Hall-Beyer \(2017\)](#) for a rule of thumb and its justification. It might also be useful to choose one from each group (contrast, order and descriptive statistics) mentioned above, or either the edge or patch interior texture groups in [Hall-Beyer \(2017\)](#).
- With this number, techniques of feature selection can be used (comparison of mean and minimum Bhattacharyya Distance, for example).

Example: [Clausi \(2002\)](#) worked on classification using texture of SAR (radar) sea ice imagery. He analyzed correlation among textures to determine the best subset of texture measures. He found that Contrast, Correlation and Entropy used together outperformed any one of them alone, and also outperformed using these and a number of others all together. If only one can be used, he recommends choosing among Contrast, Dissimilarity, Inverse Difference Moment Normalized (what we have called Homogeneity) or Inverse Difference normalized (what we have called Similarity). [Clausi](#) also summarized the texture measures found to be useful in a number of SAR sea-ice studies. He found Entropy always used, and Contrast and Correlation a close second ([Clausi 2002](#) Table 2, p. 47).

[Hall-Beyer \(2017\)](#) systematically examined associations of different texture measures over three Landsat images and three window sizes, to find commonalities that might serve as rules of thumb in selecting texture measures for classification problems. Texture measures were separated into groups more suited to finding edges and those more suited to distinguishing the textures on the inside of patches of a given ground cover. The distilled recommendations found can be summarized as "choose Mean and, where a class patch is likely to contain edge-like features within it, Con. Cor is an alternative for Mean in these situations, Dis may similarly be used in place of Con. For more detailed texture study, add Ent."

[Click here](#) if you want to go to the texture calculations section.

Correlation of texture measurements with one another

Because of the way the texture equations are constructed, many of them are correlated with one another.

Example: Contrast uses a weight of $(i-j)^2$ and Dissimilarity uses a weight of $(i-j)$. Otherwise there is no difference between them. The range of values will be different, but the two measures contain essentially the same information. Since correlation coefficients are a measure of linear correlation, and these two texture measures will be related exponentially, r^2 will not be 1.0. Nonetheless it is very high.

Most of the texture measures within a given group are strongly correlated. GLCM Variance, being a measure of variability, is commonly closely correlated to the measures within the contrast group. The following values all assume the same window size. The correlation coefficient will vary somewhat from image to image, but the general trend is clear. In the image [shown below](#),

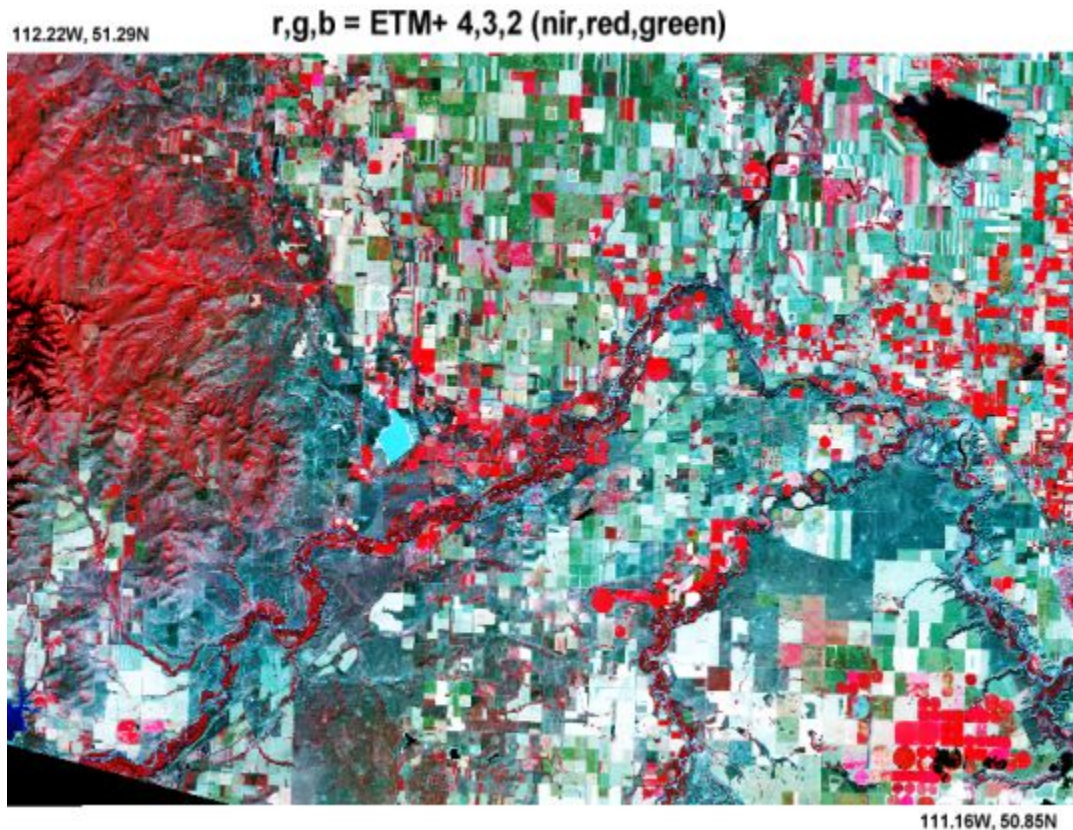
- Homogeneity is correlated with Contrast, $r^2 = -0.80$
- Homogeneity is correlated with Dissimilarity, $r^2 = -0.95$
- GLCM Variance is correlated with Contrast, $r^2 = 0.89$
- GLCM Variance is correlated with Dissimilarity, $r^2 = 0.91$
- GLCM Variance is correlated with Homogeneity, $r^2 = -0.83$
- Entropy is correlated with ASM, $r^2 = -0.87$

GLCM Mean and Correlation are more independent. For the [image below](#),

- GLCM Mean shows $r^2 < 0.1$ with any of the other texture measures demonstrated in this tutorial.
- Correlation shows $r^2 < 0.5$ with any other measure.

Practically, then, for classification purposes choose one of the contrast measures, one of the orderliness measures, and two or three at most of the descriptive statistics measures. Which one will depend on the textures of the classes desired. [Consult](#) a good image processing textbook for more information on feature selection.

Examples of various texture calculations, on an image.

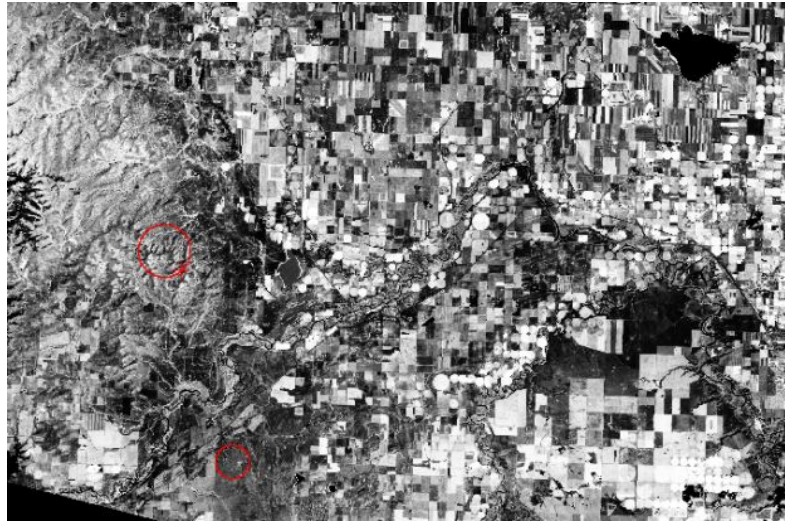


Description of the image

The image used is excerpted from Path 41, Row 25 of Landsat 7 ETM+, dated 4 September 1999, shown with north at the top. This is an area in the Rocky Mountain Foothills near Waterton Lakes National Park, Alberta. The western edge of the image contains steep slopes and deep valleys. To the east is both grassland and annual crops, mostly grains. The eastern area is dissected by numerous small streams or "coulees." This image provides a variety of textures and edges. Standard false-colour representation is used to highlight differences within vegetated areas and between soil and vegetated areas, as these are the main ground cover distinctions of general interest.

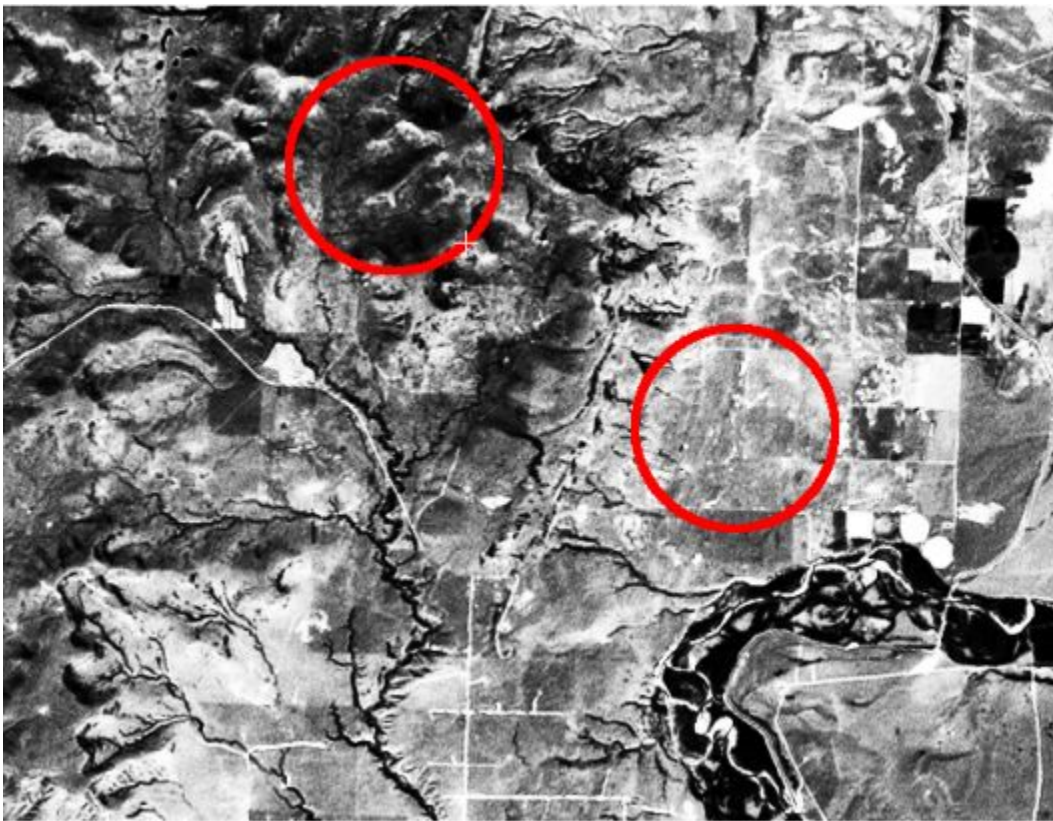
Two areas (subregions) have been circled in all images to make comparisons easy.

Entire image in red band (ETM+ band 3) alone. This band will be used for all following



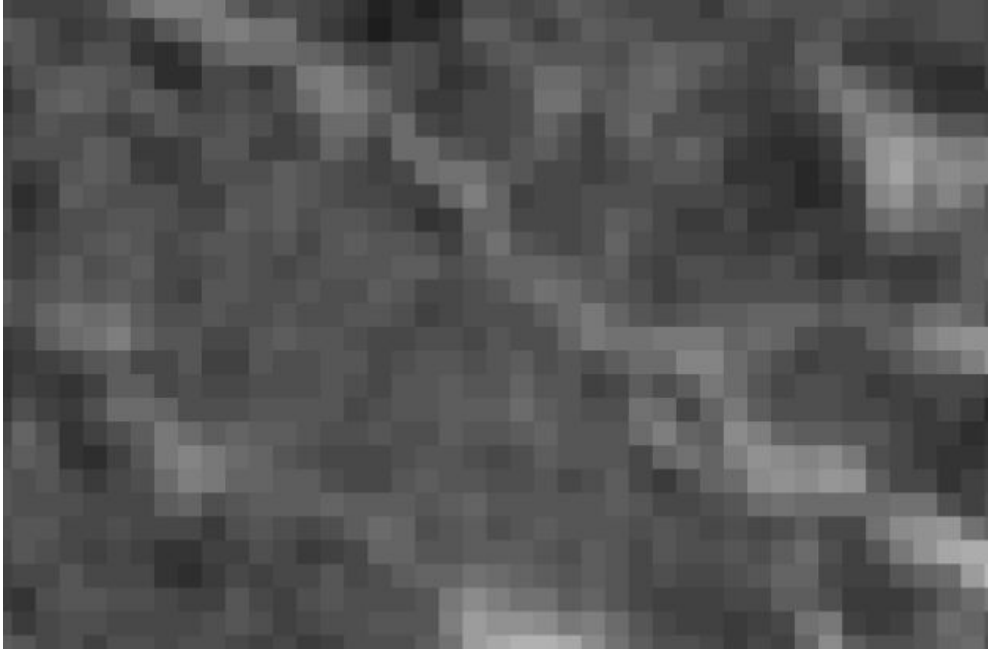
texture images.

Image at maximum spatial resolution (1:1) around each subregion (red circled) area:



SERIES OF IMAGES OF COMMONLY USED GLCM TEXTURE MEASURES

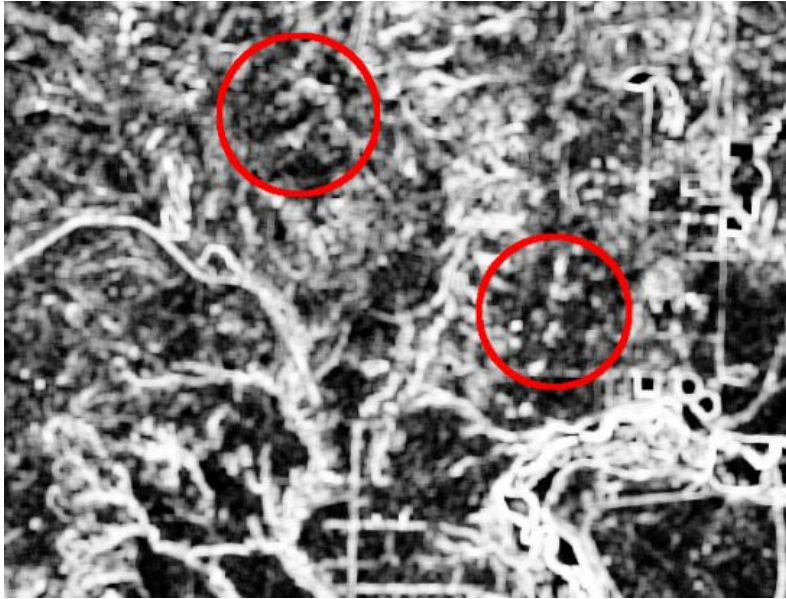
Detailed views of northern subregion (top) and southern subregion (bottom), showing pixel value relationships between adjacent pixels. Visual comparison with the idea of “contrast” in mind would be useful to understanding



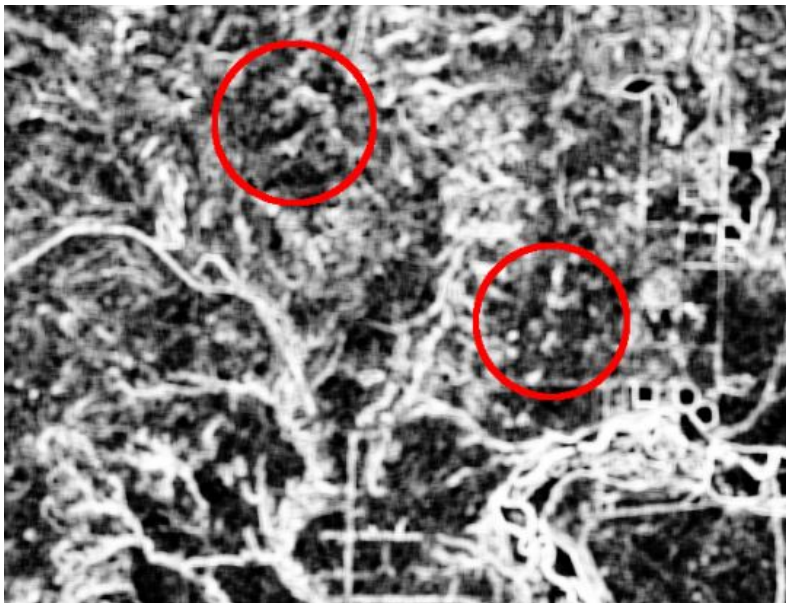
The textures below were run using a 7x7 window on the red band. All used the invariant direction, which is an average of all four spatial arrangements. Pixel offset of 1 (i.e. pixels and their adjacent neighbour constitute the pairs for the GLCM count).

CONTRAST GROUP

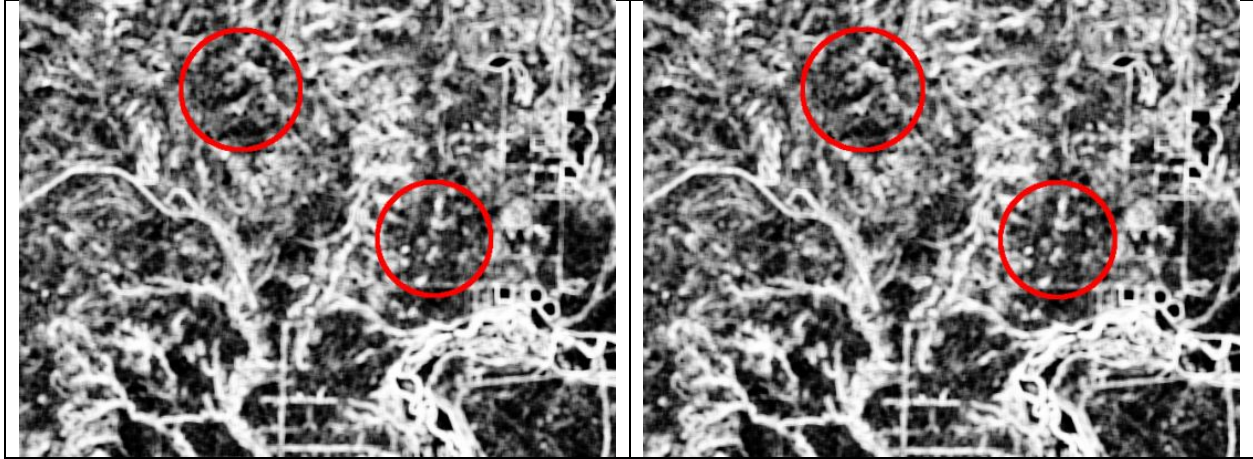
CONTRAST



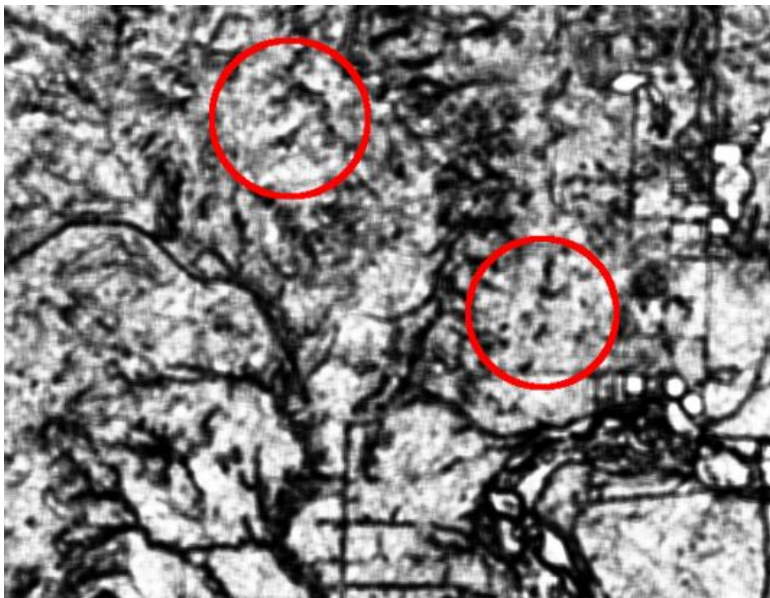
DISSIMILARITY



GLDV MEAN, which is identical in calculation to GLCM DISSIMILARITY: GLDV MEAN on left, GLCM DIS on right (and above). This shows that simply accepting all texture measures in a software program is not efficient! Make sure the ones selected are not redundant and do what you want.



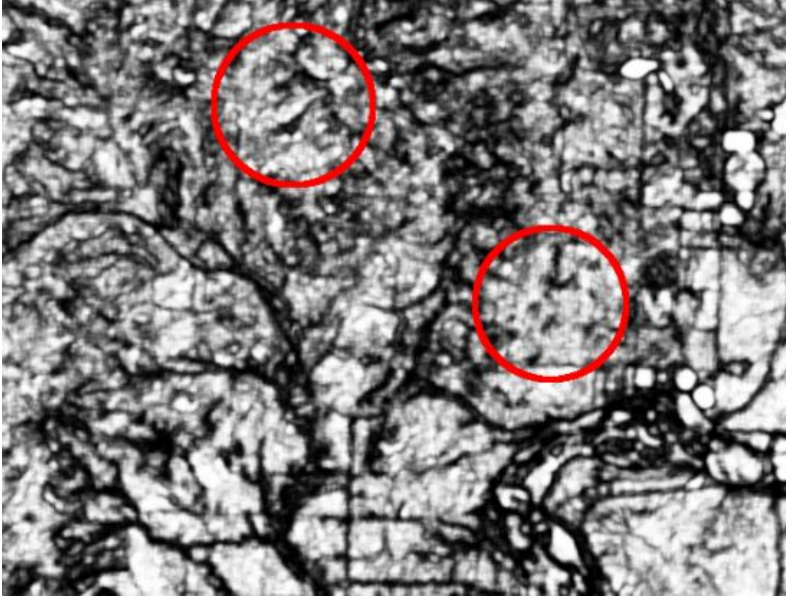
HOMOGENEITY



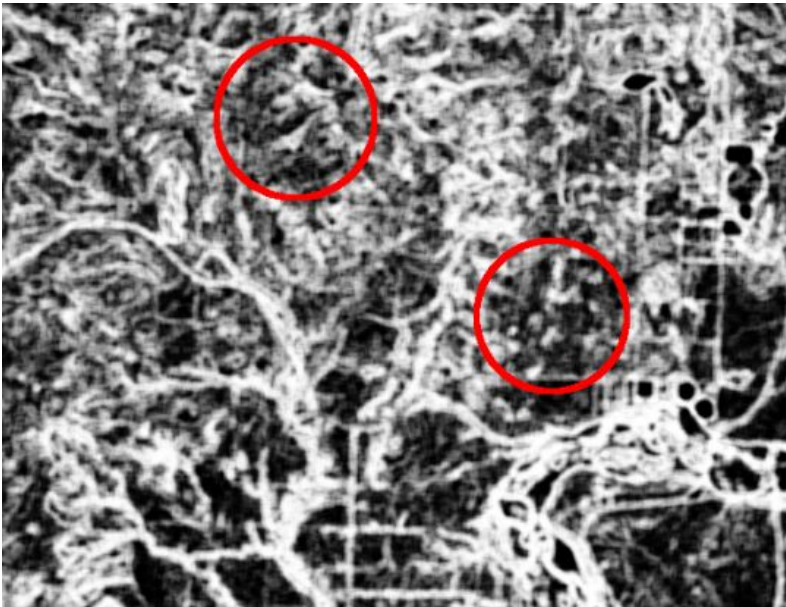
END OF CONTRAST GROUP

ORDERLINESS GROUP

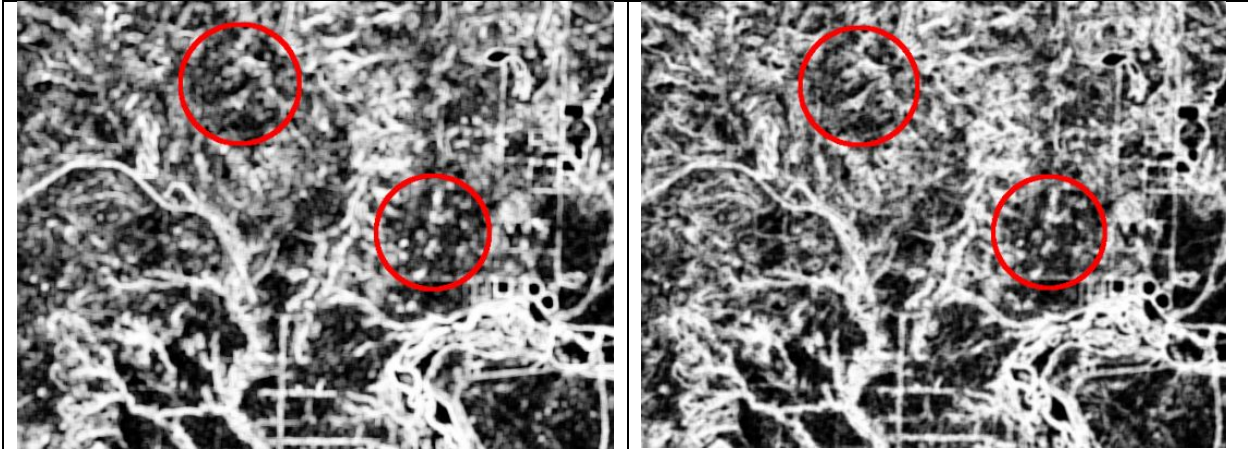
ASM:



ENTROPY



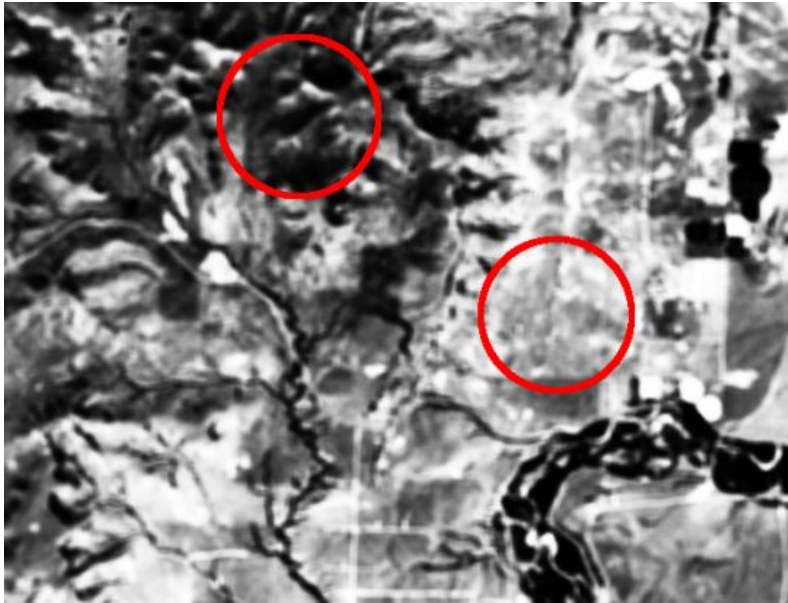
Comparison of Entropy calculated using the GLDV (left) and GLCM (right). Once again, these are identical, as with the DIS calculations above.



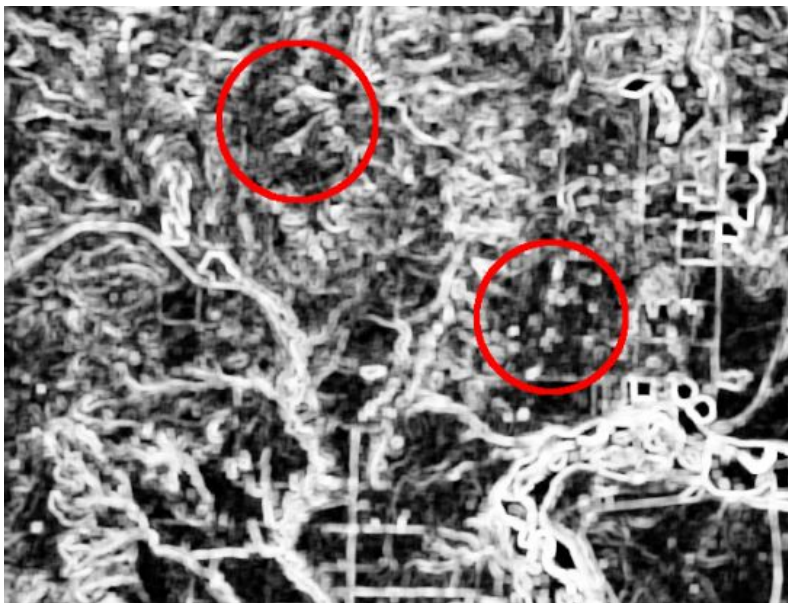
END OF ORDERLINESS GROUP

DESCRIPTIVE STATISTICS GROUP

GLCM MEAN

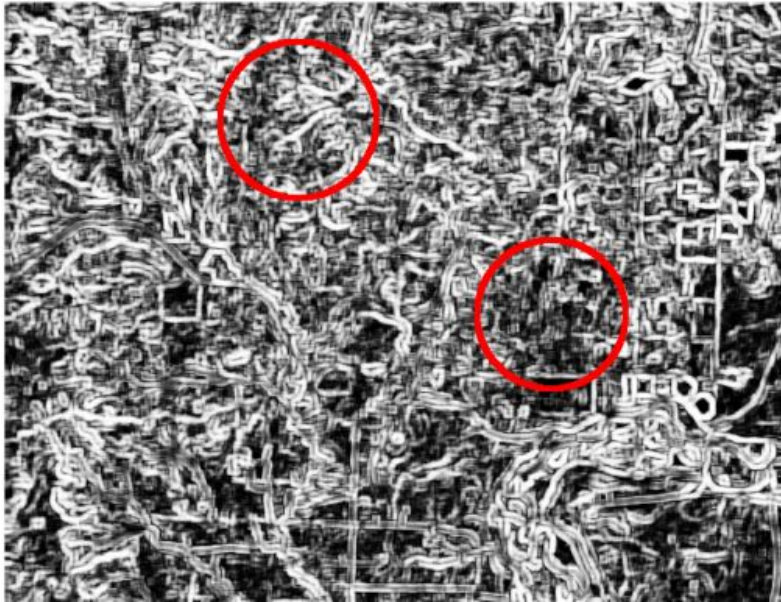


GLCM STANDARD DEVIATION (VARIANCE)



GLCM CORRELATION

Correlation, GLCM, 7x7, invariant

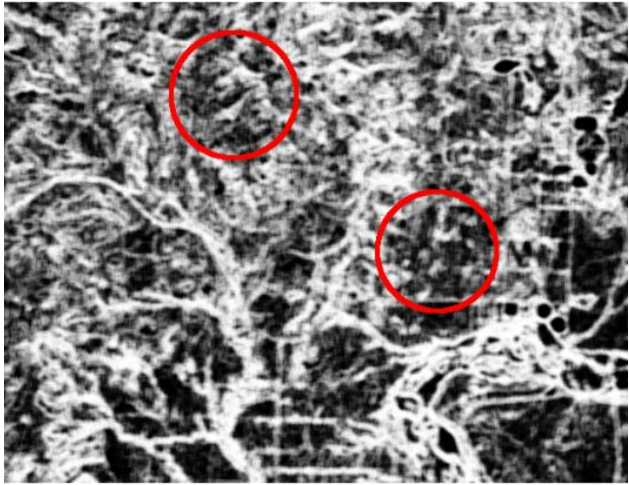


END OF DESCRIPTIVE STATISTICS GROUP

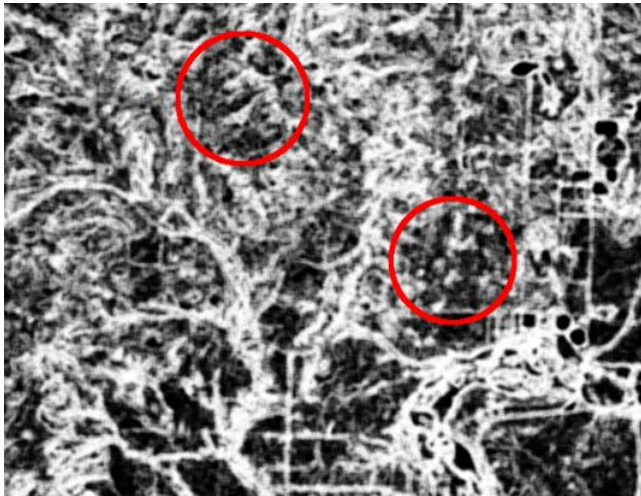
COMPARISON OF DIFFERENT SPATIAL RELATIONSHIPS

The following are all entropy calculations using a 7x7 window:

Spatial (0,1) and (0,-1): this is a "vertical" relationship, both "north" and "south". The area just north of the rightmost circle shows distinctive traces of a vertical alignment. Notice that a directional texture measure will show direction visibly where such a directionality of pixel contrast occurs on the image: it does not force a directional outcome where there is no such alignment on the original image.



Spatial (1,1) and (-1,-1): a NE-SW diagonal relationship



ANSWERS TO EXERCISES

1. Exercise: use the test image and a south spatial relationship (reference pixel and the neighbour below it) to test understanding. Fill in the blanks.

Test image grey levels:

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Framework matrix:

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

Count (south) matrix	+	Transpose (north)	=	Symmetrical (vertical)
3 0 2 0		3 0 0 0		6 0 2 0
0 2 2 0		0 2 0 0		0 4 2 0
0 0 1 2		2 2 1 0		2 2 2 2
0 0 0 0		0 0 2 0		0 0 2 0

Symmetrical (vertical) matrix divided by the sum of the elements (24) =

Normalized symmetrical vertical GLCM

.249	0	.083	0
0	.166	.083	0
.083	.083	.083	.083
0	0	.083	0

[Click here to return to exercise 1 place in main document](#)

2. Exercise:

Calculate the GLDV for the vertical relationship.

A difference of	occurs this many times	normalized (divided by sum of column 2)
0	12	.500
1	8	.333
2	4	.166
3	0	0

[Click here to return to exercise 2 place in main document](#)

3. Self test:

a. What is the degree of this measure?

Ans. **Second degree**

b. What does a Contrast of 0 mean?

Ans. In such an image there is no difference between any reference pixel and its neighbour. All non-zero values in the GLCM are along the GLCM matrix diagonal. This GLCM refers to a horizontal spatial relation. Therefore the Contrast of 0 for *this* GLCM does not mean pixels are necessarily the same as those above, below or diagonal to them. Visually, this means that the image consists of "stripes" of equal GL values running E-W. It says nothing about how wide the stripes are.

Similarly, a Contrast of 0 in a vertical spatial relationship would mean the image has stripes running N-S. Contrast of 0 in all directions would mean an entirely uniform image.

[Click here to return to self-test 3 place in main document](#)

4. Exercise: Calculate the Contrast for the vertical GLCM and compare it with the Contrast for the horizontal GLCM

Calculation: **Vertical Contrast**

Contrast weights				X	vertical GLCM				=	multiplication result			
0	1	4	9		0.249	0	0.083	0		0	0	.332	0
1	0	1	4		0	.166	.083	0		0	0	.083	0
4	1	0	1		0.083	0.083	0.083	0.083		0.332	.083	0	.083
9	4	1	0		0	0	.083	0		0	0	.083	0

Sum of all elements in the multiplication result table = **0.996**

The result is actually = 1 if rounding is not carried out at every step to the same decimal places as above. You may get a value of 1 if calculated on a spreadsheet, depending on the decimal place settings.

Here are the detailed, step by step calculations. The weight matrix is the result of the calculations inside parentheses:

$$\begin{aligned}
 &.249*(0-0)^2 + .0*(0-1)^2 + .083*(0-2)^2 + 0*(0-3)^2 + \\
 &.0*(1-0)^2 + .166*(1-1)^2 + .083*(1-2)^2 + 0*(1-3)^2 + \\
 &.083*(2-0)^2 + .083*(2-1)^2 + .083*(2-2)^2 + .083*(2-3)^2 + \\
 &0*(3-0)^2 + 0*(3-1)^2 + .083*(3-2)^2 + 0*(3-3)^2
 \end{aligned}$$

$$= .250(0) + .083(4) + .166(0) + .083(1) + .083(4) + .083(1) + .083(0) + .083(1) + .083(1)$$

$$= .332 + .083 + .332 + .083 + .083 + .083$$

$$= .996$$

Horizontal Contrast = .586 ([link to calculation](#)); vertical contrast = .996. **Contrast is higher in the N-S direction than for the E-W direction.**

[Click here to return to exercise 4 place in main document](#)

5. Exercise: Try out the Dissimilarity calculation for the *horizontal image* and compare the value with Contrast for the same matrix:

Calculation: Vertical Dissimilarity:

Dissimilarity weights				X	Vertical GLCM				=	multiplication result			
0	1	2	3		.280	0	0.083	0		0	0	.166	0
1	0	1	2		0	0.166	0.083	0		0	0	0.083	0
2	1	0	1		0.083	0.083	0.083	0.083		0.166	0.083	0	.083
3	2	1	0		0	0	0.083	0		0	0	.083	0

sum of all elements in the multiplication result matrix = **.664**

Vertical Dissimilarity is 0.664; vertical Contrast is 0.996 ([link to vertical Contrast exercise answer](#)). Contrast gives a higher number than does Dissimilarity, which is as expected since Contrast weights are larger for every pixel more than 1 off the diagonal. This indicates that the actual number resulting from the calculation is only important in relation to the same texture measure number in other windows elsewhere on the image.

Also, compare horizontal with vertical Dissimilarity. Is the pattern the same as for horizontal vs vertical Contrast?

Calculation: Horizontal Dissimilarity:

Dissimilarity weights X Horizontal GLCM = multiplication result

0	1	2	3		0.166	0.083	0.042	0		0	.083	.084	0
1	0	1	2		0.083	0.166	0	0		.083	0	0	0
2	1	0	1		0.042	0	0.249	0.042		.084	0	0	.042
3	2	1	0		0	0	0.042	0.083		0	0	.042	0

sum of all elements in the multiplication result matrix = **.418**

Vertical matrix Dissimilarity is 0.664, horizontal is 0.418. The Dissimilarity is greater in a N-S direction than in an E-W direction. Both Contrast and Dissimilarity show greater contrast in N-S than in E-W. This is as expected since both measure the same thing in different ways.

[Click here to return to exercise 5 place in main document](#)

6. Exercise: Calculate the homogeneity value for the horizontal GLCM and compare it with the Dissimilarity value. Homogeneity = **.807** Dissimilarity = **.418**

Homogeneity calculation:

homogeneity weights: X horizontal GLCM = multiplication results

1	.5	.2	.1		.166	.083	.042	0		0.166	0.042	0.008	0
0	1	.5	.2		.083	.166	0	0		0.042	0.166	0	0
0	.5	1	.5		.042	0	.25	.042		0.008	0	0.250	0.021
0	.2	.5	1		0	0	.042	.083		0	0	0.021	0.083

sum of multiplication results matrix = **.807**

In non-matrix form:

$$\begin{aligned}
 &.166/1 + .083/2 + .042/5 + 0/10 + \\
 &.083/2 + .166/1 + 0/2 + 0/5 + \\
 &.042/5 + 0/2 + .250/1 + .042/2 + \\
 &0/10 + 0/5 + .042/2 + .083/1
 \end{aligned}$$

$$\begin{aligned}
 &= .166 + .042 + .0084 + 0 + \\
 &.042 + .166 + 0 + 0 + \\
 &.0084 + 0 + .250 + .021 + \\
 &0 + 0 + .021 + .083
 \end{aligned}$$

$$= \mathbf{.807}$$

[Click here to return to exercise 6 place in main document](#)

7. Self-test: The weight used in contrast is $(i-j)^2$. The weight in homogeneity is $1/[1+(i-j)^2]$.
a. What would happen if the contrast weight were $[1+(i-j)^2]$?

Ans. There would be a slightly larger value, as the diagonal pixels would be weighted as 1.

b. Would this be a bad thing?

Ans. It would still give you a weight increasing exponentially away from the diagonal. But it would count pixels with identical values as having a positive value of contrast which would seem counterproductive. The main reason the "1+" is in the Homogeneity denominator is to avoid having to divide by 0 for pixels on the diagonal (which would essentially give them infinite weight).

c. What degree is this measure?

Second degree (it contains a squared term in the denominator). Whether the square is in the numerator or the denominator, it is considered second degree.

[Click here to return to self-test 7 place in main document](#)

8. Exercise: Do-it-yourself Similarity texture:

Homogeneity is the most commonly used measure that increases with less contrast in the window. However, it would be easy to use the above model to construct a first degree "Similarity" measure.

$$\sum_{i,j=0}^{(N-1)} \frac{P_{i,j}}{1 + |i - j|}$$

Similarity Equation:

Calculation for the horizontal GLCM:

"Similarity" weights: X horizontal GLCM = multiplication results

1.000	0.500	0.333	0.250	0.166	0.083	0.042	0	.166	.042	.014	0
0.500	1.000	0.500	0.333	0.083	0.166	0	0	.042	.166	0	0
0.333	0.500	1.000	0.500	0.042		0.25	0.042	.014	0	.25	.021
0.250	0.333	0.500	1.000			0.042	0.083	0	0	.021	.083

Sum of multiplication results matrix = **0.804**

[Click here to return to exercise 8 place in main document](#)

9. Exercise: Perform the ASM calculation for the horizontal GLCM:

P_{ij}^2 :

0.027	0.007	0.002	0
0.007	0.028	0	0
0.002	0	0.0625	0.002
0	0	0.002	0.007

summed = **.145**

[Click here to return to exercise 9 place in main document](#)

10. Self test: The maximum value of 1 for either ASM or Energy occurs when all pixels in the image are identical. Quickly draw the GLCM for this situation and perform the ASM calculation. The GLCM below assumes that all pixels are identical and have a value of 2

GLCM:

0	0	0	0
0	0	0	0
0	0	24	0
0	0	0	0

Normalized, this gives:

0	0	0	0
0	0	0	0
0	0	1.0	0
0	0	0	0

ASM: $1*1 = 1$

Note that this would be the same no matter what value you put in the original image, so long as all pixels were identical.

[Click here to return to self-test 10 place in main document](#)

12. Exercise: GLCM Mean for the horizontal test image is:

$$\begin{aligned} &0*(.166 + .083 + .042 + 0) + \\ &1*(.083 + .166 + 0 + 0) + \\ &2*(.042 + 0 + .250 + .042) + \\ &3*(0 + 0 + .042 + .083) \\ &= .249 + 2(.334) + 3(.125) = .249 + .668 + .375 = \mathbf{1.292} \end{aligned}$$

GLCM Mean for the vertical test image is:

$$\begin{aligned} &0*(.250 + .083) + \\ &1*(.166 + .083) + \\ &2*(.083 + .083 + .083 + .083) + \\ &3*(.083) \\ &= \mathbf{1.162} \end{aligned}$$

This was not asked, but the mean for the *original* pixel values in the window is 1.25. This would be a first-order "texture" measure, though it is difficult to see how it could be called texture in any practical sense.

[Click here to return to exercise 12 place in main document](#)

13. Exercise:

Calculate the GLCM Variance texture for both the horizontal and vertical GLCM of the test image to see if they are the same.

GLCM Variance (horizontal) =

$$\begin{aligned} &.166(0-1.292)^2 + .083(0-1.292)^2 + .042(0-1.292)^2 + 0 + \\ &.083(1-1.292)^2 + .166((1-1.292)^2 + 0 + 0 + \\ &.042(2-1.292)^2 + 0 + .250(2-1.292)^2 + .042(2-1.292)^2 + \\ &0 + 0 + .042(3-1.292)^2 + .083(3-1.292)^2 \end{aligned}$$

$$= \mathbf{1.039067}$$

GLCM Variance (vertical) =

$$\begin{aligned} &.250(0 -1.162)^2 + 0 + .083(0 -1.162)^2 + 0 + \\ &0 + .166(1 -1.162)^2 + .083(1 -1.162)^2 + 0 + \\ &.083(2 -1.162)^2 + .083(2 -1.162)^2 + .083(2 -1.162)^2 + .083(2 -1.162)^2 \\ &0 + 0 + .083(3 -1.162)^2 + 0 \end{aligned}$$

$$= \mathbf{.969705}$$

Variance calculated on the original image values rather than on the GLCM = **1.030776**

[Click here to return to exercise 13 place in main document](#)

14. Exercise: Calculate the Correlation measure for the horizontal test image. Do the GLCM Mean and GLCM Variance exercises first and use their results.

Link here to the answers to [exercise 12](#) (GLCM Mean) and [exercise 13](#) (GLCM Variance)

Hint: Correlation must always be between -1 and +1 before scaling. Any other result shows an error. It is easy to make calculation errors. The most error-free way to do this calculation is to set up the formula on a spreadsheet. On a spreadsheet, you can at least be certain that there have been no arithmetic errors, even though it is still easy to make an error in entering the formula.

GLCM Correlation (horizontal GLCM):

$$\begin{aligned}
 &.166[(0-1.292)(0-1.292)]/[(1.039067)(1.039067)]^{.5} + \\
 &.083[(0-1.292)(1-1.292)]/[(1.039067)(1.039067)]^{.5} + \\
 &.042[(0-1.292)(2-1.292)]/[(1.039067)(1.039067)]^{.5} + \\
 &0 + \\
 &.083[(1-1.292)(0-1.292)]/[(1.039067)(1.039067)]^{.5} + \\
 &.166[(1-1.292)(1-1.292)]/[(1.039067)(1.039067)]^{.5} + \\
 &0 + \\
 &0 + \\
 &.042[(2-1.292)(0-1.292)]/[(1.039067)(1.039067)]^{.5} + \\
 &0 + \\
 &.250[(2-1.292)(2-1.292)]/[(1.039067)(1.039067)]^{.5} + \\
 &.042[(2-1.292)(3-1.292)]/[(1.039067)(1.039067)]^{.5} + \\
 &0 + \\
 &0 + \\
 &.042[(3-1.292)(2-1.292)]/[(1.039067)(1.039067)]^{.5} + \\
 &.083[(3-1.292)(3-1.292)]/[(1.039067)(1.039067)]^{.5}
 \end{aligned}$$

=

$$\begin{aligned}
 &0.2770978/1.07966 + \\
 &0.0313129/1.07966 + \\
 &-0.038419/1.07966 + \\
 &0.0313129/1.07966 + \\
 &0.0141538/1.07966 + \\
 &-0.038419/1.07966 + \\
 &0.125316/1.07966 + \\
 &0.0507891/1.07966 + \\
 &0.0507891/1.07966 + \\
 &0.2421329/1.07966
 \end{aligned}$$

= (continued on next page)

0.2566528 +
0.0290026 +
-0.035584 +
0.0290026 +
0.0131095 +
-0.035584 +
0.1160698 +
0.0470417 +
0.0470417 +
0.2242677 +

= 0.69102

For those who have referred to a previous version of this document, note that the calculations are correct, but that I had mistakenly inserted the variance value instead of the std dev value into the equation. The equation uses σ^2 , which is of course the same as the variance. I had previously squared the variance, which is of course incorrect. It has been corrected here.

*Several people asked why the equation included two separate terms for σ^2 : one in the i and one in the j direction. In this example the values are the same, so why not just collapse the equation from $(\sigma^2 * \sigma^2)^{0.5}$ to σ^2 ? The uncollapsed form is easier to follow, and it is not the case that the i and j sigmas are always the same. We have used the normalized GLCM throughout this tutorial. However for directional purposes (east vs. west, not vertical vs. horizontal) one might want to have the GLCM non-symmetrical, in which case the sigma i and sigma j would be different.*

[Click here to return to exercise 14 place in main document](#)