# MITRE Image Quality Measure Computer Program© (IQM - v7.3)

**********

Executables are provided for Apple Macintosh and Microsoft Windows Operating Systems:

- Mac OSX: Double-Click executable and Command-Line executable (Unix terminal mode) for Mac/Intel and Mac/PPC computers
- Windows 95/98/2000/NT/XP/?Vista: Double-Click executable and Command-Line executable (command prompt or DOS window mode)
- Source code was compiled using Absoft ProFortran-95, copyrighted by Absoft Corporation.

**********

Conversion of “IQ”, the output quality value, to “NIIRS”, has not been certified or endorsed by any United States government agency.

**********

Website: http://www.mitre.org/tech/mtf

Point of contact: MITRE MTF and IQS Assessment
The MITRE Corporation
202 Burlington Road - M/S K228
Bedford, Massachusetts 01730
USA

phone: (+1) 781-271-3365
email: mtf@mitre.org

## 1. BREIF HISTORY OF IQM

The foundation for IQM can be traced to the body of work initiated in the 1960s and 1970s on the measurement of power spectra of hardcopy imagery by coherent optical means, for applications such as SAR signal processing, cloud screening, terrain-type classification, and automatic pattern recognition, as well as image quality assessment. IQM derives most directly from a specific paper in that body of work:

> "Scene Power Spectra: the Moment as an Image Quality Merit Factor", November 1976, N.B.Nill, Applied Optics, vol.15, pp. 2846-2854

To fulfill a need on a government project, MITRE initiated an in-house R&D effort in 1989 to develop an all-digital approach to the previous optical approach for obtaining image quality from digital image power spectra. Use of the inherent flexibility of a digital computer and software programming enabled significant extensions of the basic concept to include, for example, a filter for the human visual system, a noise filter, and a directional scale factor. MITRE delivered the first complete IQM software program to the US Air Force in January 1990. A paper documenting MITRE's IQM was published in 1992:

> "Objective Image Quality Measure Derived from Digital Image Power Spectra", N.B.Nill and B.H.Bouzas, April 1992, Optical Engineering, vol.31, pp. 813-825

Under the direction of Norman B. Nill, the initial IQM versions were programmed by Brian H. Bouzas (1989-1991), who also assisted in experimentation and reducing the theory to practice. Additional coding was performed (1991) by Barry R. Paine. The program's original concept/design, and all MITRE enhancements, testing, experimentation, and coding since 1992 have been performed by N.B. Nill.

The only IQM code segment not developed by MITRE is the FFT routine. Three different open source FFT routines have been incorporated in different versions of IQM; each required some revision as well as additional coding to reformat the scrambled FFT output into a usable form (see “References” section for information on current FFT).

A MITRE copyright on the IQM software program was formally applied for and granted by the Library of Congress Copyright Office in 1992.

## 2. PROGRAM OVERVIEW

IQM computes image quality based on the two-dimensional (2-D), spatial frequency power spectrum of a digital, or digitized, image. The power spectrum is the square of the magnitude of the Fourier transform of the image. The power spectrum contains information on the sharpness, contrast, and detail rendition of the image and these are components of 'image quality'. The power spectrum is normalized by the image brightness and weighted by a visual response function; if sufficient noise is detected a modified Wiener noise filter is also applied to the spectrum. Depending on the user-selected sensor type, the spectrum may also be normalized by the image size (in pixels) and/or weighted by a direction-dependent scale factor. The fundamental output Image Quality factor, "IQ" , is the sum of the weighted, 2-D power spectrum values. The visual weight aligns IQ with what a human perceives as 'quality'. The noise filter compensates for the power spectrum's sensitivity to noise. The brightness and image size normalizations allow intercomparisons between images with different average gray levels and different image sizes, respectively. [One optional exception to image size normalization applies to digital camera sensor #5, when user selects "sensor array size" to have an effect on computed quality.] The scale factor converts the power spectrum's inherent scale-independence to scale-dependence, a prerequisite for conversion to a National Imagery Interpretability Rating Scale (NIIRS) value. NIIRS is a visual quality rating scale used on aerial images, wherein object-image scale has a significant impact on quality. The conversion from IQ to NIIRS uses the slope and intercept from a previously established IQ - NIIRS regression equation.

Three Input Files are needed to run IQM:

- image file
- preferences data file
- auxiliary data file

The preferences data file contains over 30 parameters that users can tune for their own image types. The program has a default preferences data file that the user may select from the program's initial display; this file, called "DefaultPrefs", is created and recreated each time IQM is run.

The auxiliary data file (hereinafter called "AuxDataFile") contains information about the image and sensor/image acquisition geometry. It can either be constructed beforehand, in order to run a series of images in batch mode, or it can be constructed for each image during IQM runtime, from user key-in data.

There are several Output Data File options (default names):

Concise file, 1 printline per image: *IQM_OUTPUT*
Concise file: *IQM_OUTPUT*, plus separate data dump file: *iqmDataDump*
Concise file: *IQM_OUTPUT*, plus separate paired-image output file: *IQratio_Output*
Verbose file, ~3 pages per image: *IQM_OUTPUT*

The Verbose output includes more parameter values, and the one-dimensionalized power spectrum as a function of spatial frequency (power summed around concentric circles, or rings) and the power spectrum as a function of angle (power summed in angular wedges); these spectra are both tabulated and plotted.

The data dump file contains the concise output, preferences file, and AuxDataFile, all in comma-separated numeric form, ready for direct input to a spreadsheet or plot program.

The only (optional) image output is the input image with a boundary line drawn around the IQM-selected peak IQ area; this output image is in PGM format, regardless of the input image format. PrefsFile setups for activating this output image option are:

| Sensor | peakavg* | SubImageInfo* | minsize*, maxsquare* |
|---|---|---|---|
| 1 - 5 | P, PA, or A | NW or YW | minsize = maxsquare |
| 6 | P, PA, or A | NW or YW | any values (not used) |

* parameter in PrefsFile
In AuxDataFile, No subimages are specified for given image.

## 3. IMAGE FORMATS

IQM reads uncompressed grayscale images or uncompressed RGB color images stored as 1 byte per pixel (8bpp) or 2 bytes per pixel (16bpp) per color layer. The only image format that IQM reads, interprets, and understands is PGM format; images in any other format are read as 'raw'.
IQM cannot read:

-- compressed images
-- pixel values stored as floats(real), complex numbers, or signed integers
-- images where non-image bytes are interspersed between image rows (e.g., striped TIFF)

The following goes into more detail on some common image formats and how IQM handles them.

### 3.1 PGM Format

IQM reads the PGM format that is described at:
http://badc.nerc.ac.uk/data/claus/pgm_format.html
This is the so-called "P5" PGM format applicable to 8bpp grayscale images.
For recognition by IQM, the image filename Must end with: **.**pgm
The image width, height, and header size are read from the PGM file and 8bpp is assumed. Values in the AuxDataFile for these 4 parameters are ignored when **.**pgm is detected, so the beginning of a data line in AuxDataFile could be (width height header bpp order): 0 0 0 0 L
When using IQM to create the AuxDataFile, a bit order 'L' will automatically be placed in the file for a PGM image; you will need to edit the AuxDataFile offline if the true bit order is 'B' (see section 3.5 for discussion on bit order).

### 3.2 TIFF Format

A TIFF image can be used, but it will be read as raw; the user must supply the header bytes, pixel width and pixel height of the TIFF image. When read as 'raw', the image can have non-image bytes before the first image pixel bytes (i.e., header bytes) and/or non-image trailing bytes after the last image pixel byte,

but there cannot be any non-image bytes implanted between image rows, i.e., the TIFF image cannot be striped.

Specific to IQM, “header bytes” refers to the sum total of ALL bytes in the image file that come before the first image pixel byte (sometimes called “offset” bytes). The TIFF standard (v6.0) definition of "header size" is not the same as used in IQM. TIFF defines a constant header size of 8 bytes, but the standard allows additional bytes of information to come before the first image pixel byte, such as for image directories. All of these non-image bytes must be included in the count of "header bytes" in IQM. Since IQM does not read support data in TIFF files, it needs to be told how far into the file to go to find and start reading the first image pixel byte; it discards all bytes before the first image pixel byte. Since all extra bytes are discarded, any information in those bytes, e.g., grayscale gamma or color balance, is Not used in IQM; the image is read as 'raw'.

A number of image viewer applications will give the total header size of a TIFF image that is needed for IQM input, e.g., the following freeware applications:

*NIHimage* (Mac classic), *ScionImage* (Windows):
http://rsb.info.nih.gov/nih-image
http://www.scioncorp.com
Use the import menu item to display the TIFF image, then go back to import menu item to see “offset” size.

*ImageJ* (Mac OSX & Windows):
http://rsb.info.nih.gov/ij/
Before opening the TIFF image, enable debug mode via edit/options/miscellaneous, open image, then see “offset “ in log window. Also can verify that “rowsperstrip” = image height, if not equal, the TIFF image is striped and may need to be destriped prior to IQM processing.

### 3.3 BMP Format

BMP can also be read-in as raw, but note that BMP stores the image 'upside down', the first row of pixels in the file is really the bottom row of the image. If you simply run it through IQM (with correct header size) as is, the program will assume the image is right side up. To avoid this, either flip the image before saving as BMP, or convert a non-flipped BMP image to TIFF, PGM, or raw, before input to IQM. Freeware that extracts BMP image properties:

*ImageInfo* (java class)
http://schmidt.devlib.org/image-info/

### 3.4 Color Image

RGB full color images can be read-in, but not CMYK, YCbCr, or other non-RGB color coding formats.
(1) Full color RGB images can be read-in if the pixel data is contiguous, such as TIFF "chunky" format, i.e., the 3 color layers are intermingled: R1,G1,B1,R2,G2,B2,…Rn,Gn,Bn (where 1,2,..n refer to first pixel, second pixel, ... nth pixel).

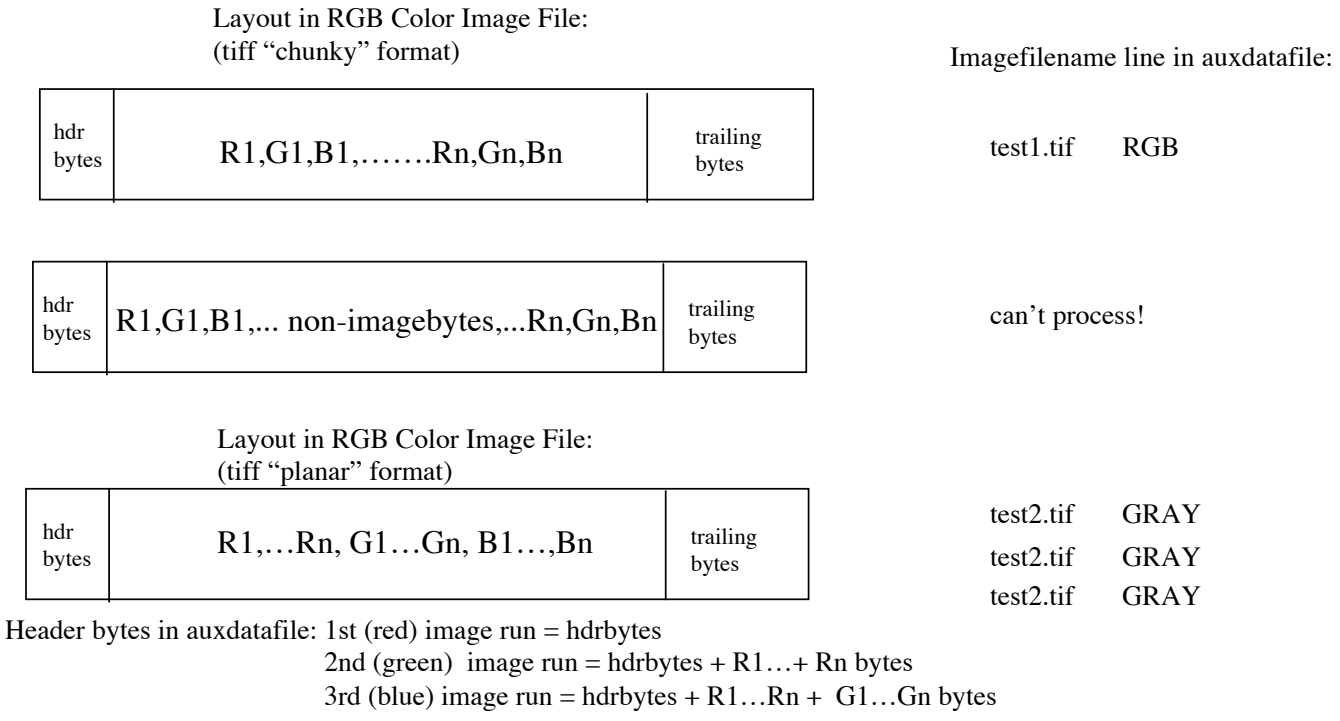
"RGB" must appear in AuxDataFile after the imagefilename (on same line).
With this option all 3 color layers are processed, one after another, and the single output quality value is the weighted sum of the quality values of the 3 layers; the DefaultPrefs file has default weights for the red, green, and blue layers.

(2) Images with sequential color layers, such as TIFF "planar" format, can be read-in. In this sequential pixel format the color layers follow each other: R1,R2,R3,…Rn, G1,G2,G3…Gn, B1,B2,B3,...Bn. In this case the entire color image is input to IQM but each color layer is run as a separate gray image, with separate output, so "GRAY" must appear in AuxDataFile after the imagefilename (on same line); e.g., for an image with 3 sequential color layers, the AuxDataFile would have 3 identical image entries, except for a change in header bytes:
if headerbytes for first layer = H
then headerbytes for second layer = H + imagewidth x height
then headerbytes for third layer = H + 2 x imagewidth x height
This case is also applicable to multispectral, false color images.

One approach for color quality assessment applicable to this case is to select the 'most important' color layer (often the green layer) and just process this one layer; or else, use the color-weighted quality values of all 3 color layers.

Note that color fidelity or color 'truthfulness' of a color image is not measured by IQM. IQM measures quality factors such as spatial detail, sharpness, and contrast, not color fidelity. The following graphic illustrates some of the readable/unreadable color image formats in IQM.

**Some RGB Color Image Formats:**



### 3.5 Bit Order

For each input image, IQM must be told what is here called the image's "bit order", i.e., whether it is Big-endian or Little-endian[1] (B or L), which is an entry in the AuxDataFile.

For 16bpp images it is a little more complicated because there are four possible bit orders: -B, B, -L, or L. IQM menu option 3: "determine image format" is an aid in identifying the correct 16bpp bit order for a given image (also usable for 8bpp image).

The selected bit order can be verified during IQM processing, as follows:
For each image processed, the gray level values of the first 2 pixels in the upper left corner, top row of image, and the first 2 pixels in the lower left corner, bottom row of image, are displayed during runtime, and the gray level value of the first pixel in upper left corner, top row is also printed to the IQM output file. **These are the pixel gray levels as read by IQM, and they must be correct!**

- If all 4 values are wrong, try the other bit orders in AuxDataFile; if still wrong, the number of header bytes may be wrong.

- If the 2 pixel values in top row are correct but the 2 pixels values in bottom row are not, then some possible reasons are: input image width is wrong, or there are extra, non-image bytes imbedded between rows of pixels, e.g., striped TIFF format (in this case you will need to reformat into plain TIFF).

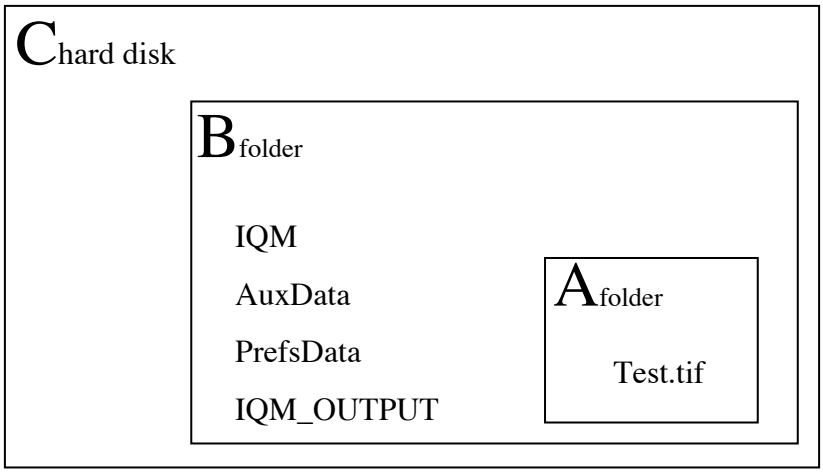
The correct, true gray levels of the above pixels are obtained by displaying the image at full resolution i.e., one digital image pixel maps to one display pixel, and placing the display cursor over the pertinent pixels (the image viewer must have single pixel gray level readout). IQM expects pure white to be gray level 255 for an 8bpp image, so a light area in the displayed image should have a gray level > 128 when placing the cursor over it.

### 3.6 File Pathnames

If the IQM executable, all images, AuxDataFile, and PrefsFile are all in the same folder (or "directory"), then file pathnames are not needed; all output files will also be sent to this folder.
If the files are in different folders, then the appropriate pathnames must appear in the AuxDataFile.
For input to IQM, pathnames and filenames cannot contain empty spaces.

Consider the following example:

[1] By most definitions endianess pertains to Byte order, not bit order, and would therefore not have meaning for 1Byte pixels. Here, however, we use a generalized definition as a convenience to denote the black/white polarity of 1 Byte pixels, as well as its more formal meaning with respect to 2 Byte pixels (hope the Lilliputians don't mind).




Valid pathnames would be:

| | |
|---|---|
| /B/A/Test.tif | (Mac OSX run) |
| C:\B\A\Test.tif | (Windows run) |
| \B\A\Test.tif | (Windows run) |

## 4. PROCESSING FLOW

### 4.1 Normal Case

The input image can be a square or rectangle, but the Fast Fourier Transform (FFT) algorithm used in the program is currently constrained to operate only on square images, of any width (not necessarily power-of-2 width). The program applies the following logic when presented with an image for processing:

IF entire image is square,

THEN: entire image is processed, although it may be processed in a sequence of square subimages if: maxsquare < entire image width, where "maxsquare" is user defined in the preferences data file.

IF entire image is rectangular,

THEN: square, subimages are formed for individual processing,

Either:

from user input of specific sizes and locations of subimages to process

Or

the subimages are automatically generated by the program. In this case, all square, independent-location subimages, from the maximum possible size down to "minsize " size are generated; this may or may not cover the entire image; the program prints to display the percentage of the entire image area that it will be processing. Also, should note that the automatic subimage location is based on the geometry of the image, not on the structure content or busyness of various scene parts. If the images you are using are known to contain large areas of no scene structure, which you may not want to measure, then it would be better to 'hand pick' the subimage sizes and locations by previewing the images.

Notes:

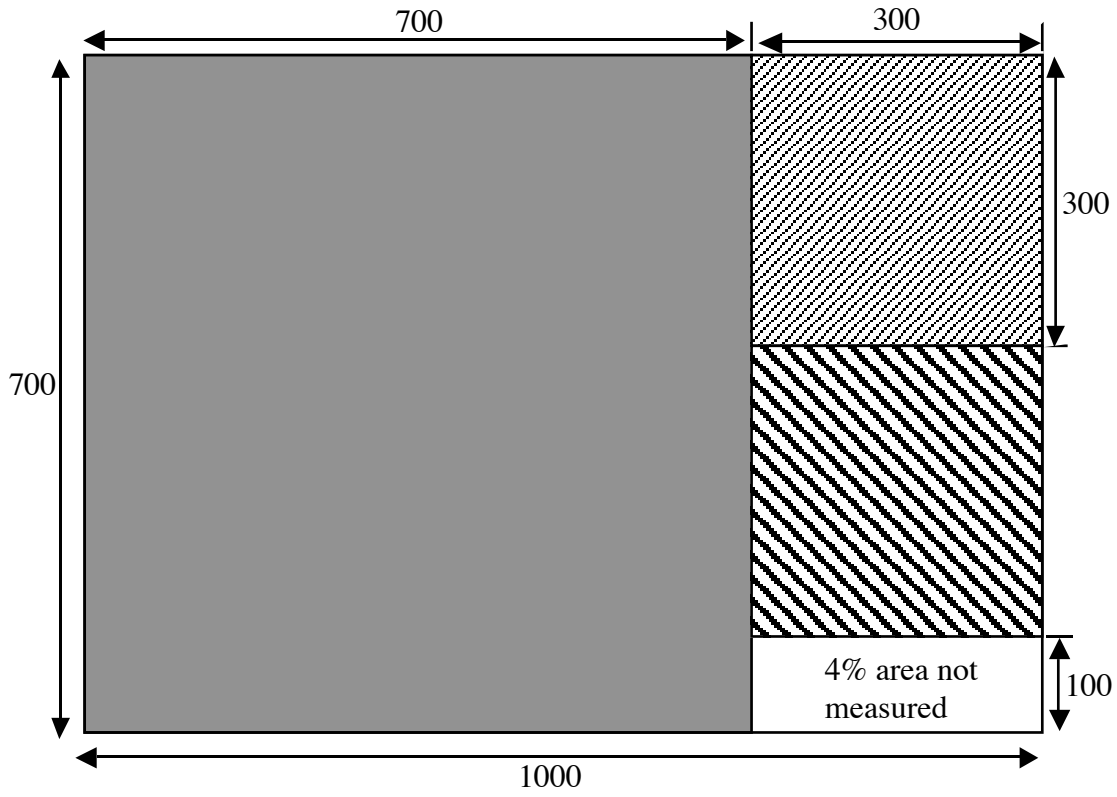
- Coordinate reference origin for subimages:
  if upper left corner of a subimage coincides with upper left corner of entire image, the subimage coordinate would be 0,0 not 1,1 (in pixel units), and col=0, row=0 would be printed in *IQM_OUTPUT* file for that subimage.

- If multiple subimages are processed, the quality values for each subimage can be output along with the weighted average quality; in preferences data file, set: "SubImageInfo = Y".
  The weighting is performed by subimage width, not subimage area.

Example (see following graphic): 1000 x 700 pixel image is input to IQM and IQM is asked to automatically identify the subimages for processing;
in preferences data file: minsize=256, maxsquare=2048, peakavg=A,
then the program finds three independent subimages for processing: 700x700, 300x300, 300x300. The 4% of the area in the white rectangle is not measured. IQ is computed for each of the 3 subimages and a size-weighted average IQ is then computed, i.e.,

avg IQ = SUM{(subimagewidth(i) x IQ(i)} / SUM{subimagewidth(i)} ,SUM over all i (i=1,2,3 in this case)

.



As another example, suppose you have a 2000x2000 pixel image and would like the quality in each 250x250 pixel segment (64 total segments); in preferences file set:

minsize=250
maxsquare=250
peakavg=PA
subimageinfo=Y

In AuxDataFile, set image width & height to 2000 and no subimages; then the result will be computation and printout of 64 separate IQ values, together with the average of 64 IQs and the peak IQ.

A similar IQ gridding can be applied to image pairs, see next section 4.2.

## 4.2 Special Case - Paired Image Assessment

### Compute Quality Difference Between Image Pairs

IQ of image#1 is compared to IQ of image#2; IQ of image#3 is compared to IQ of image#4, etc.

This option assumes the two images in any given pair are the same scene, at same scale, but they differ, e.g., sharpness, contrast, or image compression processing differences.

Turn-on option by setting highhaze parameter in Preferences file to:
highhaze = –99.
also, minsize & maxsquare must have same value, which should be less than half of the image width. This will automatically grid the image into equal size subimages and IQ is then computed in each subimage.

There are 3 IQ comparison options, or measurement strategies, the user can choose from via runtime instructions. In following, "IQratio" is IQ of first (A) image / IQ of second (B) image of a given image pair.

1) Paired Subimage: outputs IQratio that is furthest from 1.0, percentage-wise, for corresponding same subimage location in A and B

2) Unpaired Subimage: outputs maximum IQ of image A / maximum IQ of image B, regardless of location of maximum IQ in either image.

3) PeakPaired Subimage: output maximum IQ of image A / IQ of image B at same location as in A. if IQratio <1, image B is better than image A, reverse is true if IQratio >1

IQratio > 1.0 implies image A is higher quality than image B
IQratio < 1.0 implies image B is higher quality than image A

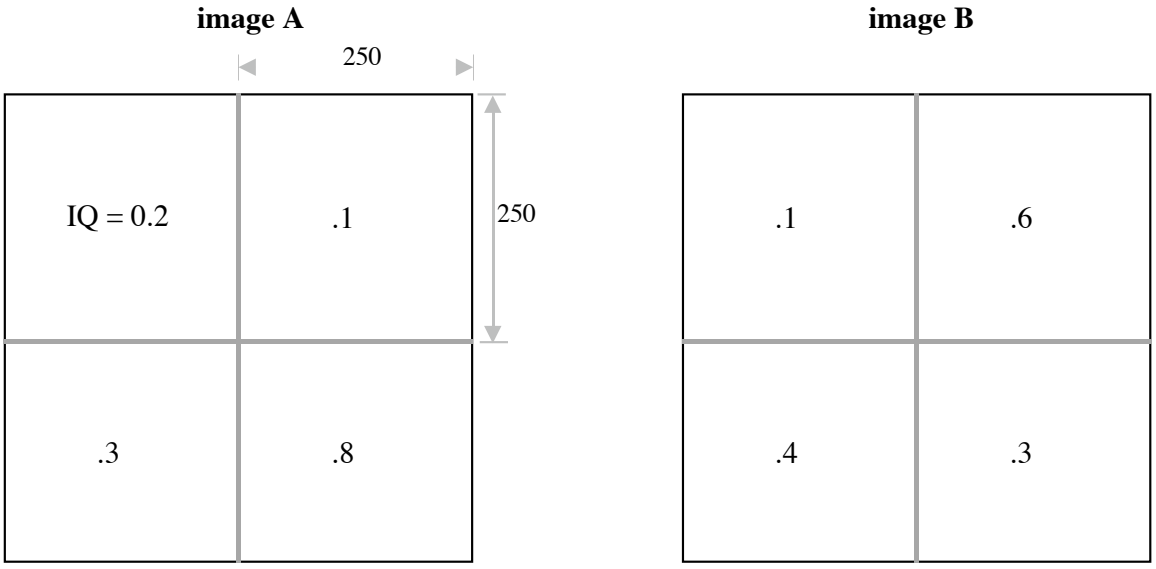
This option is setup for sensor #4, which does not compute NIIRS. However, the NIIRS Difference between the two images of a pair is computed, from:

delta NIIRS = M x $Log_{10}$ (IQratio)
(M is slope of NIIRS & log(IQ) relation defined in prefs file)

In addition to standard *IQM_OUTPUT* file, a new output file is created: IQratio_Results

Following example illustrates the 3 IQ ratio options (numbers in grid represent computed IQ value of 250x250 subimage).

in preferences file: highhaze = –99.
minsize = 250
maxsquare = 250
in AuxDataFile: sensor #4




Paired Subimage IQ ratios: .2/.1 = 2 (.1/.6 = 0.167) .3/.4 = 0.75 .8/.3 = 2.67

UnPaired Subimage IQ ratio: .8/.6 = 1.33

PeakPairedSubimage IQ ratio: .8/.3 = 2.67

## 5. COMPUTER ITEMS

### 5.1 Memory Requirements

*Approximate* memory needed to run IQM, in MegaBytes = ( $3WH + 12S^2$ ) / $1x10^6$

W = width of entire input image
H = height of entire input image
S = width of largest square image processed by FFT routine as a single image,

S equals the smaller of: W, H, maxsquare, or subimage width (if defined by user)
Maxsquare is the width of the maximum size square subimage that will be processed by the FFT routine as a single image, as defined in the preferences data file.
Example: W=2048, H=2500, maxsquare=4096, no subimages defined, then:
3*2048*2500+12*2048*2048 = 63 MB of memory required

### 5.2 Compute Time

Example results for an IMAC 2Ghz Intel core duo machine:

| image size = FFT size: | IQM process time: | Comment: |
|---|---|---|
| 1024 x 1024 pixels | 1.2 seconds | 1024 is power-of-2 size |
| 1021 x 1021 | 14.5 seconds | 1021 is prime number |

As seen above, the FFT routine runs considerably slower when presented with a prime number image width; a msg is displayed if image width is a prime number > 701.

### 5.3 Runtime Problems

• IQM 'hangs' the computer: good possibility that there isn't enough memory available to IQM

• "Fatal unknown problem with first image"
If the first image cannot be read the program stops; an image could be unreadable because the program cannot find the image; actual image filesize is smaller than indicated by combination of input width, height & header values; or other problem. If the first image is OK but subsequent images cannot be read, the program continues, skipping the unreadable images.

• If IQM output lists multiple problem codes for a given image, possibility that AuxDataFile isn't correct, image is wrong size, wrong subimage locations, wrong header size, wrong polarity, etc.

• IQM won't read an AuxDataFile or PrefsFile, even though you may have successfully used it in the past, symptom: program starts and ends without processing any images. This could happen if the file was created with an older, OS9 version of IQM and is now being used with the Mac OSX compiled IQM version. You need to change the file's end-of-line character to the format that is compatible with the compiled IQM you are running: Mac OSX is line feed, Mac OS9 is carriage return, and DOS/Windows is carriage return followed by line feed. BBedit is one of many applications that will make these end-of-line changes, just open file and resave with selected format.

• Separation of AuxDataFile values by tab versus spaces: single tab or spaces separating imagefilename from imagetype (Gray, RGB) is OK, but data in datalines must be separated by spaces

## 6. PREFERENCES DATA FILE

Some of the parameter values in the default preferences data file are tuned to 8 bpp aerial images acquired from visible or near-visible spectrum imaging sensors (film camera, EO/IR digital sensors). Different parameter values may be required for SAR images (e.g., change in noise thresholds), or for various types of terrestrial images.

### 6.1 Power Normalization

The power spectrum, from which the IQ value is computed, is normalized by either DC, TC, or AC, as defined in the PrefsFile parameter "psntype":

psntype = DC
Normalization by zero frequency power, resulting in image contrast having a major impact on computed IQ (higher contrast, higher IQ); this is the normalization usually used.

psntype = TC
Normalization by total power (AC+DC), resulting in image contrast having some impact on computed IQ, but not quite as much of an impact as for DC normalization. [In most cases, relative IQ results between images in a set will be the same as for DC normalization, except for magnitude change.]

psntype = AC
Normalization by all non-zero power, resulting in image contrast having only a minor impact on computed IQ. AC normalization is useful for images exhibiting noticeable haze; it allows computation of the NIIRS value that would result if the haze were to be removed from the image. This is useful when assessing sensor performance alone, since it removes the effect of quality loss due to atmospheric haze, which is not part of the sensor. It can also be used to ascertain the expected quality resulting from posterior haze removal processing. It can also be useful in better defining relative quality between multiple images of the same scene when, for example, there is a large change in contrast between the sample images, but, for whatever reason, contrast changes are not pertinent to the definition of quality.

If IQM is run with psntype = DC, the output IQ value can be directly converted to what it would have been with TC or AC normalization for the same image, without the need to rerun IQM, i.e.,

$$IQ_{ac} = IQ_{dc} / contrast^2$$
$$IQ_{tc} = IQ_{dc} / (1+contrast^2)$$

where contrast is the contrast value computed by IQM, which is proportional to the standard deviation of image gray levels / mean gray level.

The above conversions are accurate for single images; when subimages are processed and only the average values are selected to be output (subimageInfo = N), there would be some loss of accuracy because the average contrast would have to be used. In this case, better to set: subimageInfo = Y. Also, note that if $IQ_{dc}$ is run when peakavg = P in preferences data file, then conversion to $IQ_{ac}$ or $IQ_{tc}$ via the above formulas, does Not necessarily correspond to the peak $IQ_{ac}$ or peak $IQ_{tc}$ subimage. Note that with sensor #6 processing, the psntype value set in the preferences file is overridden.

### 6.2 Noise Filter

The noiseflag value is either 0 or 1; but in either case the noise filter is only applied to the image power spectrum if sufficient noise is detected in the given image, i.e., noise filter is only applied if:
noiseratio < detectlevel
where,
noiseratio is computed from image power spectrum before noise filter application
detectlevel is a noise threshold value set in preferences file
noiseflag = 0 The program's internal modified Wiener noise filter is used.
noiseflag = 1 The user defines his/her own noise filter via 6 parameters in preferences data file. These parameters are defined in Opt.Eng. paper, 4/92, Table 1, pg.820. The user can substantially modify the filter via these 6 parameters, but note that the program always uses the calculated noise variance in the filter, which assumes a 'white noise' power spectrum.
Starting in IQM v5.6, the only purpose of the "highnoise" parameter is to indicate the presence of very severe noise (problem code=5 in output file). With highnoise < detectlevel in preferences file, the noise filter is applied if the image spectrum noise ratio < detectlevel (it is Not applied again, if noiseratio < highnoise).

### 6.3 Smear Detection

A test for the presence of one-dimensional image smear is performed; if detected, smear direction and smear magnitude are computed and are reported out (in verbose output mode). The test value is the ratio of the highest power wedge to the lowest power wedge, where lowest power wedge is at/near a right angle to the highest power wedge. This 'wedgeratio' is compared to the threshold value, 'highsmear', to determine if smear is present. In the current IQM version, a first order estimate of smear magnitude is made by finding the spatial frequency location of the first null in the power spectrum.

### 6.4 Preferences File Listing

In the following preferences data file listing, 'frequency' refers to the spatial frequency axis of the image power spectrum, in units of cycles per pixel width, which ranges from 0.0 to 0.707. The width of a 'ring' is 1/64 cycles per pixel width. Parameter values given in the following are for the program's DefaultPrefs file:

```
#       Preferences file for IQM   v7.3

#       To create your own prefs file:
#       Edit IQM DeFaultPrefs file, rename & save.
#       DeFaultPrefs file is created after first 2 steps in program:
#            start IQM
#            select option 1
#            Return key
#            Return key
#            Quit program

#   comment line must have pound sign (#) in column one.
```

```
#   data line can have trailing comments, no # needed.
#   blank lines are OK
#   all frequencies in cycles per pixelwidth (x-axis of power spectrum)

iqmname = IQM_OUTPUT          Output filename
iqmnametype = 1          = 1 to append to existing output file,
#                        = 0 to write over existing output file
psntype = DC             power spectrum normalization options:
#                        DC: IQ value highly dependent on image contrast (normal case)
#                        TC: IQ value dependent on image contrast
#                        AC: IQ value not dependent on image contrast
spot = 0.6               spot & viewdist are used to fix spatial frequency location of
viewdist = 351.3288      peak of Human Visual System (HVS) response curve on power spectrum
#                        292.774*spot/viewdist = peak location in cycles per pixelwidth
minsize = 256            min width subimage program will find; but smaller square images can be input
maxsquare = 8192         max width image processed whole (single FFT); but larger images OK
peakavg = P              P = peakIQ, A = avgIQ, PA = peak&avgIQ (computed if subimages exist)
SubImageInfo = N         N = avg of subimages info to outfile; Y = info for each subimage to outfile;
#                        NW or YW = write entire image to file with boundary line around peakIQ subimage
beginfreq = 0.10         lowest frequency used for wedge power
wedgewidth = 4.0         angular width of one wedge;   <=180 degrees
slopecut = 999.          image blur if computed avg slope (0.05-0.25 frequency) > slopecut,
#                        then IQ sum switches (if NIIRS sensor) to: adjfreq to freqmax,
#                        slopecut=999. disables test, real working value is about 12.8
adjfreq = 0.10           see slopecut
freqmin = 0.01           0.010 is normal min frequency for IQ power summation
freqmax = 0.707107       0.707107 is normal max frequency for IQ power summation
highblur = 80.           severe blur if highblur < computed lowfreqslope
highsmear = 40.          smear if highsmear < computed wedgeratio (dir.& mag. computed)
highhaze = 0.02          heavy haze if contrast < highhaze
#                        highhaze = -99. switches on paired image processing option
sighaze = 0.07           highhaze < contrast < sighaze implies some haze present
ccdband = 3.             pixel banding if power@.5cy/pixwdth > ccdband*power@0.45cy/pixwdth,
#                        then NoiseRatio & NoiseVar computed from frequency < Nyquist

dcM = 1.6092             NIIRS = dcM * Log10(IQ) + dcB,  DC normalization
dcB = 8.6849
acM = 2.2923             NIIRS = acM * Log10(IQ) + acB,  AC normalization
acB = 8.0
tcM = 1.650              NIIRS = tcM * Log10(IQ) + tcB,  TC normalization
tcB = 8.8745
highdcM = 1.6092         NIIRS = highdcM * log10(IQ) + highdcB, blur&DC normalization
highdcB = 8.6849
highacM = 2.2923         NIIRS = highacM * Log10(IQ) + highacB, blur&AC normalization
highacB = 8.0
hightcM = 0.             NIIRS = hightcM * Log10(IQ) + hightcB, blur&TC normalization
hightcB = 0.

RedW = .21               for color image: R,G,B weights for color IQ,niirs,contrast
GreenW = .72             the 3 weights must sum to 1.000
BlueW = .07
```

| | |
|---|---|
| noiseflag = 0 | = 0 for IQM Wiener noise filter |
| # | = 1 for user-supplied noise filter (6 parms: sigmaG2,...Kappa2) |
| detectlevel = 5. | some noise if noiseratio < detectlevel (noise filter applied) |
| highnoise = 1.1 | severe noise if noiseratio < highnoise (noise filter applied) |
| # | always set: highnoise < detectlevel |
| sigmaG2 = .078 | definition of 6 noise filter parms (sigmaG2,...Kappa2) is in |
| rapw = .9268 | see Opt.Eng.J.:"Objective Image Quality.." 4/92,Table1,pg.820 |
| sigmaS2 = 6400. | |
| DenomExp = 1.5 | exponent in denominator of eq. 10, which = 1.5 in paper |
| Kappa1 = 19.2 | |
| Kappa2 = 1.5 | |

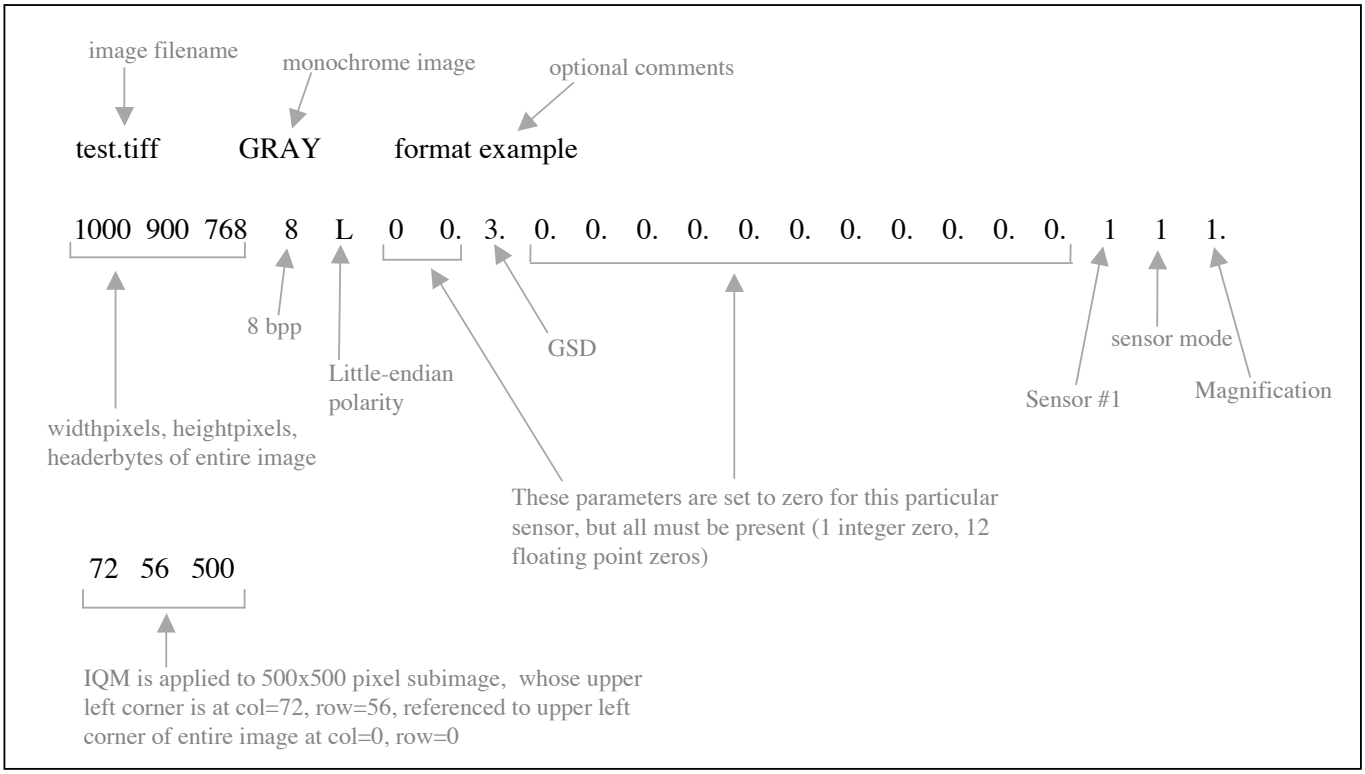
## 7. AUXILIARY DATA FILE

Basic sensor parameters are set in the AuxDataFile. It would be very tedious to have to manually key-in all the parameter data for each and every image in a large set of images. This isn't necessary, the AuxDataFile can be independently setup for Batch Processing of a large number of images in a single IQM run, where all the images can have different parameter values. For example, suppose you have 1,000 same-size images and all use sensor #4. Run IQM on first image with manual key-in of data to create AuxDataFile format. Then use a text editor to duplicate the aux data for the other 999 images (just change image filenames). Result is a single AuxDataFile that runs all 1,000 images in a single IQM run. Aux data can also be different for each image in single AuxDataFile (different image sizes, sensors, etc.), but all images in a single IQM run must use the same Preferences data file.

For any one sensor, not all of the parameters appearing in the auxdata file will necessarily be used in processing; these unused parameters are set to zero by the program, but they MUST ALL appear in the AuxDataFile!

The following diagram shows the basic setup of the standard AuxDataFile for an image; in this example it is desired to process a 500 x 500 pixel subimage rather than the whole image:

```
test.tiff     GRAY     example format
1000  900  768   8   L   0   0.   3.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1   1   1.
72   56   500
```




**Components of Standard AuxDataFile**

Magnification:
A magnification value of 1.0 should normally be used.
Changing magnification to <1 or >1 alerts IQM that the digital image being processed has been previously scaled up or down; for example:
mag = 2.0 if original image is digitally stored in 2X magnified form.
mag = 0.25 if original image is digitally stored in 1/4X minified form.
mag = 0.50 if every 2 sensor pixels on a 1-D sensor line have been aggregated together and output from sensor as 1 pixel (assumes pixelwidth parameter value is set to width of 1 original pixelwidth).

## 8. SENSOR PARAMETERS

Parameters for an imaging sensor on an aircraft or satellite are used to establish the directional scale in cycles per ground foot, which is used internally to rescale the cycles-per-unit-distance axis (x axis) of the power spectrum. This allows conversion from 'raw' IQ value, which is scale-independent, to NIIRS value, which is scale-dependent. Sensor types # 1, 2, 3 are generalized; greater accuracy in the computation of directional scale, and thus greater accuracy in computing NIIRS, can result by using a specific sensor's geometric image acquisition model and parameters; IQM includes 'placeholders' for sensor-specific models. Since NIIRS has no meaning for most terrestrial images (pictorial, medical, etc.), image rescaling is not performed and NIIRS is not computed for sensor types 4, 5, or 6.

### 8.1 General Aerial/Space Digital Sensor - SENSOR #1

Use for vertical sensor, also oblique sensor if little information is known.
Vertical sensor: optical axis is the aircraft/satellite-to-ground plumb line.
Oblique sensor: optical axis points to imaged ground object and this 'line' is not the aircraft/satellite-to-ground plumb line.
Any one of 3 parameter sets supply the information needed by the program

| | |
|---|---|
| GSD | sensor mode = 1 |
| or | |
| Range & Focal Length | sensor mode = 2 |
| or | |
| Range & Instantaneous Field of View | sensor mode = 3 |

GSD = ground sample distance (feet) = width of 1 sensor detector element projected to ground(use average of in-track GSD & cross-track GSD if both are known)
Range = sensor altitude above ground for vertical view (feet)
Range = slant range to ground object for oblique view (feet)
= straight line distance from sensor to target = altitude / cosine(lookangle)
Instantaneous angular field of view (milliradians) = viewing angle subtended by a single sensor detector element, where milliradians = 17.453293 x degrees.

SAR sensor: select sensor type #1 and use IPR value when it asks for GSD, where IPR = SAR resolution cell diameter, projected to ground.
References to 'sensor detector element width' assume that active detector elements are contiguous with no 'dead space' between them; otherwise, use detector element 'pitch' instead of 'width'.

### 8.2 General Aerial/Space Oblique Digital Sensor - SENSOR #2

Oblique sensor: optical axis points to imaged ground object and this 'line' is not the aircraft/satellite-to-ground plumb line. There are 2 options:

(1) In-track GSD and cross-track GSD are known; sensor mode = 1
No other information is needed but some simplifying assumptions are made, such as sensor is side-looking and field angle is 0.
If independently creating AuxDataFile for this case, i.e., without using IQM program, then
In AuxDataFile, in-track GSD occupies "GSD" parameter and cross-track GSD occupies "EP" parameter.

(2) Seven sensor parameters must be known; sensor mode = 2:
Three distances:
Sensor altitude (feet), sensor focal length (inches), detector pixel width (microns)
Two angles:
g = angle formed between image array x axis (=line formed by array length for 1-D detector array) and aircraft crosstrack line, both projected to ground and referenced to quadrant where 'g' is between 0 and 90 degrees.

e = field angle of image center, along principle line[2]
– if image center is on horizon side of optical axis
+ if image center is on nadir side of optical axis

Two more angles (either one of following two sets):

Azimuth = angle between aircraft crosstrack line and ground projection of sensor look direction

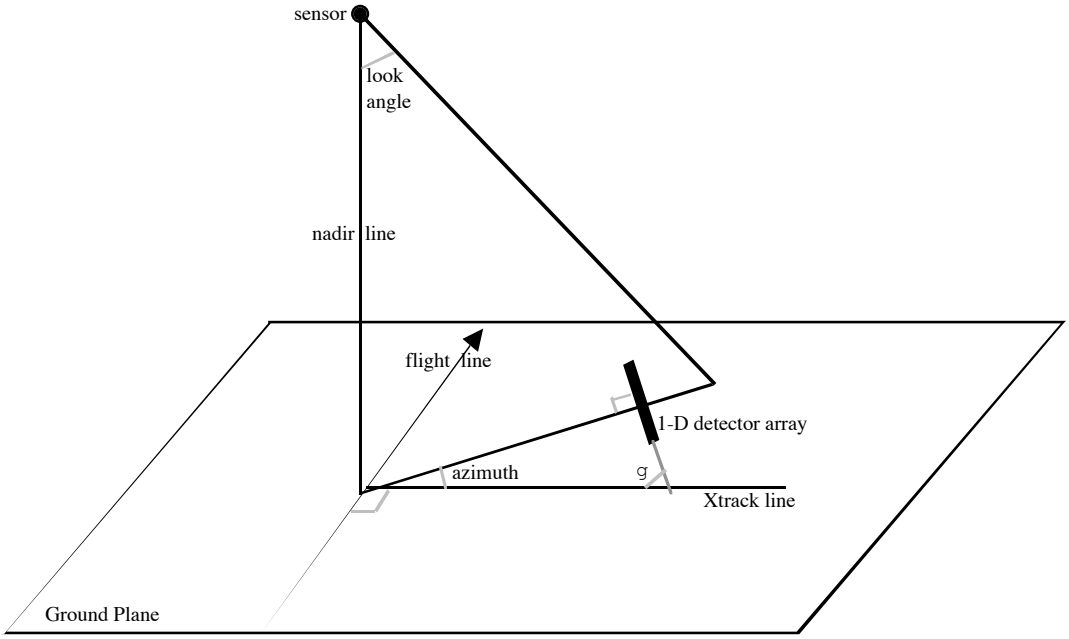
Look Angle = angle between nadir plumb line & sensor optical axis
= $90^{o}$ – depression angle

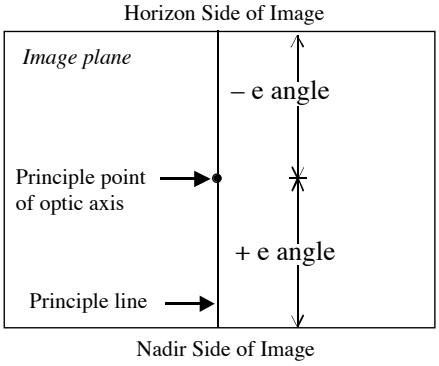
or:

Forward tilt = angle between nadir plumb line & look angle component in flight direction

Side tilt = angle between nadir plumb line & look angle component in crosstrack direction

(resultant angle of forwardtilt and sidetilt is look angle)




[2] Principle line (see diagram): the line in the image plane which goes through the point of intersection between the optical axis and the image plane, and is oriented in (colinear with) the imaging direction.

### 8.3 Aerial/Space Film Camera - SENSOR #3

This sensor option handles digitized film imagery if the following 3 parameters are known:
Scanner/Digitizer Resolution (ppi):

ppi = pixels per inch resolution of film scanner = 1 / (scan spot diameter or width)

Photo Scale:

=camera altitude / {focal length x cosine(lookangle)} (flat earth model)

Ground Resolved Distance (GRD):

= width on ground, in feet, of narrowest bar+space that is resolved on the film image (typically measured via visual resolution of film image of bar target laid out on ground).
If GRD is not known, but film resolution and basic lens parameters for the film camera are known, then one can apply the following rough conversion:

$GRD \approx 0.00328084 \times photoscale \times [(1/filmreso) + (wave \times aperture / 0.9)]$

filmreso = film resolution in cy/mm
wave = peak sensitivity illumination wavelength in millimeters
aperture = F-number of lens during image acquisition

The combination of the ppi, photo scale, and GRD are used by the program to compute an equivalent GSD, at which point processing is the same as for sensor #1 with GSD input (this computed GSD is printed out in 'Verbose' output mode). In AuxDataFile, GRD will be found to occupy "FL" parameter, photo scale occupies "ALT" parameter, scanner ppi occupies "PIXL" parameter.

### 8.4 General Sensor - SENSOR #4

This is the most general sensor case; the computed quality is neither a function of sensor array size nor a function of object-image scale; NIIRS is not computed.

### 8.5 Ground-Based Digital Camera - SENSOR #5

This sensor case is useful for quantifying the relative image quality between different digital cameras (NIIRS is not computed); the user has several options:

• If image quality is a function of number of pixels in digital camera's sensor array (user option), then in AuxDataFile, sensor width in pixels occupies "AZ" parameter, sensor height in pixels occupies "LOOK" parameter.

• If image quality is a function of scene object - image scale you will need to identify an object of interest that occurs in all cameras' images and know either:

effective width of sensor pixel projected to the scene object (sensor mode = 1)
or
object distance, lens focal length, and actual sensor pixel width (sensor mode = 2).

• If image quality is not a function of number of pixels in digital camera's sensor array, sensor mode = 3.

Some items to keep in mind when using IQM to compare digital cameras:
• It is best to acquire the same scene with all of the cameras under test; use optical lens zoom (not digital zoom) or change object distance to fill the camera's field of view with the selected constant scene. Comparing quality of different scenes taken with different cameras is a very difficult task for both humans and the computer!
• For IQM use, "number of sensor pixels" is the manufacturer quoted number of pixels at 'highest quality' level, but without selecting any 'superresolution' mode.
• Need to be careful in selecting digital camera controls, to maintain comparability:
- "quality level" (generally want highest quality that uses full sensor without superresolution)
- "image compression" (generally want lossless compression or no compression)
- use tripod to minimize camera shake during exposure
- maintain sharp focus on object plane of primary interest
- adjust exposure, highlight, etc. controls to achieve 'best that camera can offer' (e.g., avoid washed out highlights in image).

## 9. VERBOSE OUTPUT MODE

Selected at runtime, will printout 1 page of tabulated data and 2 pages of plots for each image.

wedgeratio = ratio of highest energy wedge to minimum energy wedge, the latter selected from the 3 wedges at & around 90 degrees away from highest energy wedge

MidFreqSlope = average slope magnitude of 1-D log(power) vs frequency curve over 0.05 to 0.25 cy/pixel width frequency range; tied to slopecut in prefs file.

Noise Ratio = ratio between low and high frequency band power values (from 1-D power spectrum curve); if ratio is less than threshold ("detectlevel" in prefs file) then IQM assumes noise is present and applies noise filter.

Noise Var = computed noise variance if noise ratio < detectlevel
= 0 if noise ratio > detectlevel (i.e., noise variance not used in this case)

Noise Flag = see preferences file

Contrast = square root of [ac power / dc power] , which is proportional to the standard deviation of image gray levels / mean gray level (contrast can be > 1.0).

LowFreq Slope = average slope of 1-D log(power) vs frequency curve over 0.01 to 0.05 cy/pixel width frequency range

Smear: if image smear is detected, then its estimated magnitude and direction are printed out.

**The image power spectrum is output in two forms:**

1) average power in defined frequency bands (‘rings’) versus spatial frequency in cycles per pixel width
2) average power in defined angular ‘wedges’ versus wedge angle

The above ring & wedge spectra are output in both an unfiltered and filtered form:

- Unfiltered Ring & Wedge Spectra - image power spectrum normalized by image brightness and image size, where the first ring power ("DC"), is equal to the imagewidth x imageheight, in pixels.

- Filtered Ring & Wedge Spectra - image power spectrum normalized by image brightness and image size, and weighted by eye response filter, noise filter (if applied), and scale.

- The Filtered Ring & Wedge Spectra are also plotted as simple row/column plots; for high precision ‘publication quality’ plots, it will be necessary to read the tabulated spectra data into a separate plotting program.

It is possible to see the effect of an individual filter on the image power spectrum by comparing the filtered to unfiltered spectra and 'nulling-out' the other filters, i.e.

to see effect of noise filter alone:
use sensor #4 (which sets scale to constant 1.0)
in preferences file set: viewdist = 0.1, spot = 10000.
then printout is: filtered spectrum = unfiltered spectrum x noise filter x 0.040

to see effect of visual system filter alone:
use sensor #4 and make sure noise filter is not invoked (set detectlevel=0.0001 in preferences file)
then printout is: filtered spectrum = unfiltered spectrum x eye response filter

to see effect of scale alone:
in preferences file set viewdist = 0.1, spot = 10000., detectlevel = 0.0001
then printout is: filtered spectrum = unfiltered spectrum x scale x 0.040
For sensor #2, scale varies with radial direction in power spectrum;
for sensors #1, 3, 4, 5 scale is a direction-independent constant.

## 10. REFERENCES

Technical basis of IQM:

"Objective Image Quality Measure Derived from Digital Image Power Spectra", N.B.Nill and B.H.Bouzas, April 1992, *Optical Engineering*, vol.31, pp. 813-825.
Available on-line at: http://www.mitre.org/tech/mtf

Derivation of Human Visual System Response Function incorporated into IQM :

"A Visual Model Weighted Cosine Transform for Image Compression and Quality Assessment", N.B.Nill, June 1985, *IEEE Transactions on Communications*, COM-33, pp.551-557.

Scene contrast in terms of scene power spectra:

"Contrast Effect on Imagery Power Spectra", N.B.Nill, 1 July 1979, *Applied Optics*, vol.18, pp. 2147-2151; and Errata: *Applied Optics*, 15 December 1980, vol.19, pg. 4135.

Review article, references on the statistics and scale invariance of scenes:

"The Statistics of Natural Images", D.L.Ruderman, 1994, *Network: Computations in Neural Systems*, vol.5, pp.517-548.
Similar references at: http://www.nslij-genetics.org/wli/1fnoise/

Early work – optical approach to image quality from scene power spectra:

"Scene Power Spectra: the Moment as an Image Quality Merit Factor", N.B.Nill, November 1976, *Applied Optics*, vol.15, pp. 2846-2854.

Fast Fourier Transform (FFT) used in IQM v6.0 and later:

"A New Principle for Fast Fourier Transformation", C.M. Rader and N.M. Brenner, June 1976, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, pp. 264-266.
The Fortran source code written by N.M. Brenner is available from a number of websites, e.g., do a Google search on: "numbers rich in factors of two"

## 11. WHAT'S NEW

**Version 7.3** April 2007
• Added print-out of average gray level corresponding to "IQ" window.
• Simplified read-in of 16bpp image.
• Deleted ancillary check of image file size.
• Various internal code cleaned up.

**Version 7.2** September 2006
• Fingerprint processing has been deleted as a dedicated option (sensor #6). That processing, which was in v7.0 & v7.1, was a beta version of what is now a totally separate, more advanced (non-beta) program.
• Bug fixed in entry of aux data for general oblique sensor #2
• Various internal code cleaned up.

**Version 7.1** February 2006
• Several enhancements made for processing sensor #6 (fingerprint assessment):
refined logic for weeding-out dense windows; added test for weeding-out low structure windows; refined fingerprint edge finder algorithm; introduced additional logic to effectively speed-up FFT compute time, and corrected scaling factor when ppi is not equal to 500 [$IQ_{v7.0}$ / (ppi/500) = $IQ_{v7.1}$].
• Fixed a problem with possible wrong polarity of input & output pgm images.
• Changed filename for output pgm image option, to avoid inadvertently modifying original pathname; from: peak_IQxxxx to: xxxx_IQ.pgm (xxxx is original input filename).
• AuxDataFile: added capability to read separation of imagefilename from imagetype via tab as well as via spaces.
• Corrected some minor printout bugs; deleted some redundant code; updated this Guide document.

**Version 7.0** November 2005
• Added option for fingerprint quality assessment (sensor #6); the image must be 8bpp grayscale and it cannot contain more than one fingerprint. Basic algorithm finds fingerprint in image, applies traveling IQ measurement window within fingerprint area, avoids extremely dense areas, makes adjustments for window contrast, all resulting in output of peak IQ.
• Added Unix-style command line version for Mac OSX terminal mode and Windows/Cygwin.
• Modified writeimage option so that if the PGM image file already exists it is not recreated (original file is not overwritten), with msg to display. [Writeimage option was added in v6.8, invoked in preferences file with "W", i.e., SubImageInfo=YW or NW.] Also, if image is very dark a white boundary line is now used to draw boundary around peak IQ subimage, rather than black line.
• Changed parameter value in DefaultPrefs file changed from peakavg=A to peakavg=P (when subimages exist, peak IQ subimage is output, rather than average IQ across all subimages).
• Increased maximum number of subimages (per image) from 5,000 to 10,000.
• Cleaned up some internal logic flow; modified some display msgs; added some verification checks for preferences file parameters; revised the IQM_Guide document.

**Version 6.8** April 2005
• Option added to duplicate the input image, with boundary line around subimage that program selects as having peak IQ; in prefs file see *subimageinfo* and set *minsize* equal to *maxsquare*.

• Added Menu option “11” for data dump; dumps the concise output data + auxdata + prefsfile to separate *iqmDataDump* file with all comma-separated data values; ready for direct input to spreadsheet or plot program.
• Clarified some output messages, combined Notice with Help display, cleaned-up some internal code.

**Version 6.6** January 2004
• Added check for correct end-of-line format in auxdatafile.
• Deleted rounded number writes when creating auxdatafile in OSX version; exact number input is now written to file.
• Added calculation of in-track GSD and cross-track GSD for oblique sensor.
• General code clean-up.
• Revised this Guide document.

**Version 6.5.2** March 2003
• Fixed bug in reading file pathnames in OSX version.
• If > 50 unprocessable images, program quits (previous limit was 100); content of output list revised.
• If excess bytes in image file (non-image, non-header bytes), runtime displays quantity.
• Fixed infinite loop bug when AuxDataFile is corrupted.
• Max number of subimages per image and max number of image pairs processed, both increased to 5000.
• Fixed empty file creation/overwrite bug affecting Windows version.
• If PGM file can’t be read, the problem encountered is printed to runtime display.
• Revised this Guide document.

**Version 6.5** February 2003
• Program now reads the “P5” PGM format applicable to 8bpp grayscale images (width, height, header automatically read from image file); image filename must end in: .pgm
• Computation of total power was extended out to corners of square array in FFT domain; most often this has a negligible effect on IQ and contrast values, as compared to previous versions of IQM (i.e., less than 1% change); although if image has a lot of noise, change can be larger. This power summation extension was done for theoretical reasons and slightly increases accuracy of IQ and contrast values.
• Added a third normalization option: power spectrum normalization by total power; this required adding 4 parameters to preferences file.
• Runtime information displays: fixed various minor bugs and revised displays; added some sanity checks on prefs file values read-in. All problem messages that are displayed during runtime now start with **** (for easier compilation of problems). If subimages are processed, now only the information (IQ, NIIRS) on the first subimage is displayed.
• Added numerous checks and workarounds for potential occurrence of fatal "division by zero" and NAN conditions (these are usually a result of incorrect values in AuxDataFile). Added placeholder code for MTF division.
• The sensor mode parameter in Auxdatafile is now used, for sensors 1,2, 5; but setup in previous IQM versions, which did not use sensor mode values, is 'grandfathered' into current version.
• Revised on-line Help information; revised this IQM_Guide.
• Added check for image widths that are prime numbers, because FFT routine runs much slower on primes.
• If user selects DefaultPrefs file and that file already exists, the existing file will be replaced with newly generated file (previous versions didn't replace existing file).

• For "image pairs" processing option, deleted constraint that only allowed IQratio-to-deltaNiirs conversion with slope corresponding to DC normalization (now can be DC, AC, or total power slope); also increased number of image pairs that can be processed, from 500 pairs to 1000 pairs.
• added Mac native OSX executable.

**Version 6.3** March 2002
• Added plots of ring spectrum and wedge spectrum to verbose output; these are simple row/column Log(power) plots to show trends in spectra. For high accuracy, 'pub' quality plot user will need to transfer tabulated output data into a separate plot program.
• Added option for image pairs assessment. Use when quality difference between different versions of the same image is wanted, e.g. sharp & blurry images of same scene.
See description in section: "Special Case - Paired Image Assessment".
• Added freqmax parameter to prefs data file; user can now redefine frequency range used to compute IQ The default values are: freqmin=0.01, freqmax=0.707107 cy/pixelwidth !Note! the default IQ-to-NIIRS conversion parameters, M&B, are Only valid for this default frequency range.
• Added some sanity checks for input data values; activated magnification parameter for sensor #4; page break instructions to printer; revisions to Help file and this Guide document.
• Internal: cleaned up comments, added history, added tltmti file read.

**Version 6.2** November 2001
• Verbose mode output now prints two spectra: one is image spectrum before any filtering, the other is image spectrum after application of eye response filter, noise filter (if applicable), and scale.
• Simplified routine for determining image format (menu item 3) and deleted the now unneeded "startbit" parameter. [The new AuxDataFile therefore excludes this parameter, but program will run correctly with either this new file format, or with the previous versions' AuxDataFile format.]
• Corrected bug in reading-in 16 bpp images. All images with > 8 bpp are now read-in as 16 bpp; e.g., a 12 bpp image with true gray range of 0 to 4095 is read-in and 'spread-out' to 16 bpp with gray range of 0 to 65535, with the appropriate gray level gaps. [Previous versions did not correctly read-in 16 bpp images if gray levels were > 32768.]
• During runtime, 4 pixel values are now displayed for each image, as a check that image was read-in correctly: toprow @ columns 1, 2 and bottomrow @ columns 1, 2. [If toprow values are correct but bottomrow values are not, then image may be in striped TIFF format, which IQM cannot handle.]
• More informative error messages.

**Version 6.0** April 2001
• A different FFT algorithm has been incorporated in code (see References), which now allows FFT processing of ANY size square image; all IQM versions before version 6.0 were restricted to FFT processing of square, power-of-2 size images only.
• When an image is segmented into subimages, either by the AuxDataFile instructions or automatically by program, program will now printout the peak subimage IQ and/or the average IQ across all subimages, according to user instructions in preferences data file.
• Boundary conditions in the Fourier power spectrum domain were refined, which results in a slightly more correct IQ value; e.g., same image run with v6.0 versus v5.6 will show, on average, less than 0.1% change in IQ value.
• Dealt with division-by-zero (" NaN") condition; Minor changes to display/instructions; Some further streamlining of code.

**Version 5.6** September 2000

• Fixed several bugs in handling color images. A color image can now be processed with any input sensor type (previous version didn't work correctly for sensors 1-3). Color processing instructions have been clarified in the on-line 'Help' and in this Guide.

• Increased the threshold value in DefaultPrefs file for detection of image blur, such that blur will essentially not be detected (slopecut changed from 12.8 to 999.). By itself, detection of image blur is sometimes useful, but the program logic that kicks-in when blur is detected changes the IQ summation range in the power spectrum, which changes the IQ magnitude and plays havoc with trying to determine the average IQ quality across several subimages, some of which may or may not have 'blur'. [If comparing answers between v5.6 and previous version, and "code 1" appeared on printout for a particular image in that previous version run , then there Will be a difference in IQ and NIIRS values between v5.6 and previous code version (assuming both run with DefaultPrefs file for the given code version.]

• For subimage lines in the AuxDataFile, deleted need for 4th parameter for sensors 1-5, which is always zero for these sensors.

• Users can now form their own Noise Filter via a set of 6 new parameters in preferences data file (although noise is still assumed to be 'white').

• Fixed bug in calculation of noise filter strength when AC power normalization is selected in preferences data file and noiseflag = 0 (previous produced much too strong a noise filter under these conditions).

• Fixed bug that occasionally caused a small difference in computed output values, between PC and Mac runs (switched more code to double precision).

• Revisions and clarifications made to this IQM_Guide, to on-line Help, and to run-time instructions.

**Version 5.5.1** June 2000

• Graylevel, as read by program, of pixel in upper left corner of image added to printout; this is good check on correctness of the input AuxDataFile values: "endianess" and "start bit".

• When image blur is detected, power spectrum summation for computing IQ shifts to higher frequency range; this range shift is now Only applied to sensors that compute NIIRS, i.e., not applied to General Sensor 4 or Digital Camera 5. [Range shift induces magnitude change in IQ which is (only) compensated for when converting IQ to NIIRS.]

• Fixed bug on number of images that can be processed; now set to 1 million images.

• Info msg added on how to 'Batch' process images.

**Version 5.5** April 2000

• Previous "Terrestrial Sensor" replaced by two sensor options: "General Sensor" (sensor 4) and "Ground-Based Digital Camera" (sensor 5). Digital camera sensor has sub-options for user to define which parameters affect image quality. General sensor is same as previous terrestrial sensor.

• Changes to DefaultPrefs file:

-"minsize" changed from 128 to 256 (i.e., smallest subimage is 256 x 256 pixels).

-Added option to print or not print the subimage information, is set to No: SubImageInfo = N

-Added relative weights for red, green, blue color layers of color image processed.; set equal to: red = 0.21, green = 0.72, blue = 0.07 (these weights must sum to 1.000).

• COLOR images can now be processed, such as TIFF color format red/green/blue color images. Each color layer is treated by the program as a unique image and IQ, contrast, etc. are computed for that image. After the 3 color layers have been processed, the results for each layer are weighted, according to the red, green, blue weights given in the preferences data file, to compute the final, overall color image quality.

Primarily for sensors 4 & 5, but can be applied to any sensor, just need to put RGB after imagefilename in AuxDataFile.
• Added parameter to AuxDataFile: GRAY or RGB must appear after the imagefilename, on same line. GRAY is for any gray tone image; RGB signifies a color image input.
• Concise output mode no longer prints smear magnitude & direction; this info is now in Verbose output mode.
• HELP information display was updated.
• A number of internal changes were made to clean-up, streamline the Fortran-90 code.

**Version 5.2** April 1999
• Fixed bug in calculation of wedge angle power: in previous versions, the wedge angles were inadvertently flipped about the horizontal axis, e.g., power reported out for –24 degree wedge should have been power in +24 degree wedge, similarly for all other +– wedge angle pairs.
• Added user option for defining wedge angular width in preferences data file, and changed default values for 'highsmear' & 'wedgewidth'
• Added computation of image smear direction & magnitude, if smear is detected ('highsmear' is the breakpoint between smear vs. no smear, it is compared to computed 'wedgeratio' value).
• Breakpoint between use of spherical earth model and flat earth model, which is used to determine ground plane - image plane scale, was changed from 5000 feet to 100 feet, i.e., flat earth model now used only when sensor altitude is less than 100 feet.
• Added more error checks and cleaned-up user interface.
• Added listing to verbose output: min,mid,max angles for each wedge angle

**Version 5.1** March 1999
• added new sensor option: oblique sensor with intrack & crosstrack GSD known
• more double precision computations used (scaling & iqmhvs modules)
• corrected error in azimuth entry for oblique sensor
• revised some user input displays

**Version 5.0** January 1999
• Baseline, first version posted to World Wide Web.