ZWO ASI294MC Pro Versus Mallincam SkyRaider DS10C-TEC Comparison
Part 3 - Signal to Noise Ratio & TTOI
by Jim Thompson, P.Eng
Test Report - December 13th, 2018

Introduction:
This report is a continuation of the Mallincam versus ZWO IMX294CJK CMOS camera comparison testing. The final performance parameters to compare between these two cameras is their sensitivity and noisiness. These performance parameters have the most direct influence on the quality of images that the camera is able to produce. The most common way to express camera performance in terms of these parameters is using a Signal-to-Noise Ratio (SNR). Consider the typical AP image below in Figure 1. The image is made up of signal from the deepsky object of interest, overlaid with random noise. The ratio of the magnitude of the deepsky object's signal to the magnitude of the noise is the image's SNR.

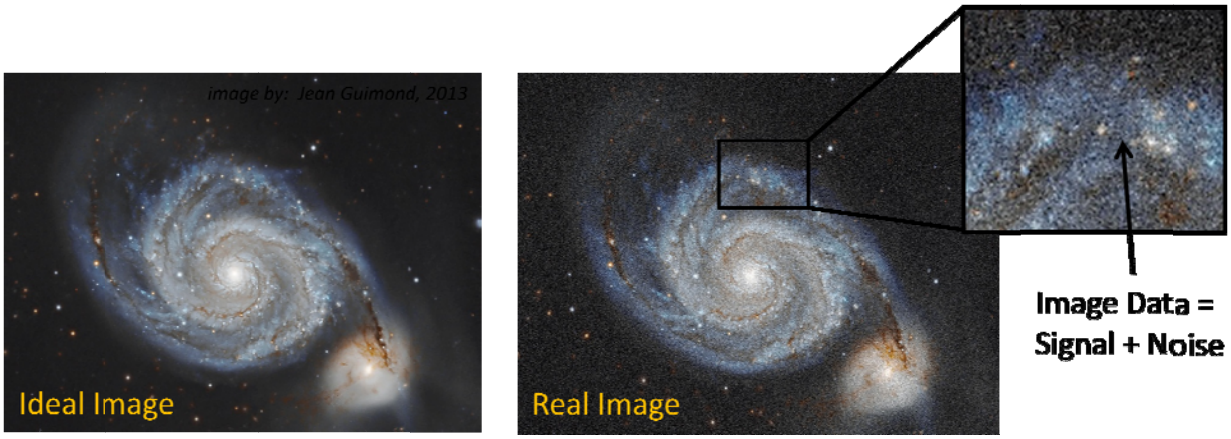

Figure 1 Ideal vs Real AP Image (original image by Jean Guimond, 2013)

SNR in an image is relatively easy to measure. It can be calculated simply by taking the mean image luminance and dividing it by the RMS noise, which is equivalent to the signal's standard deviation. Figure 2 below illustrates the process for an image I captured of an LED. The data in the plot corresponds to the pixel luminance values along the path of the red line shown on the left. Note the random appearance of the pixel luminance values; noise resulting from various sources including: shot noise, thermal sensor (dark current) noise, and read noise. In the case of this example image, I calculated the mean and standard deviation of the signal by importing the image data into a spreadsheet. I will explain later how I did this calculation for my camera comparison.

The SNR is a good measure of a camera's ability to deliver a quality image, however the SNR produced by a camera changes as a result of a number of user controlled factors:

- Gain setting: Gain is essentially the application of amplification to the signal in order to make it larger. Increasing gain will increase the magnitude of your signal, but it will also increase the magnitude of the noise as well. Depending on how gain is applied within the camera (for example where along the signal

processing path), the net result of using more gain can be an increase in SNR, a decrease in SNR, or no net change.

- Exposure time: Increasing exposure time increases signal magnitude, but also increases shot noise. There is always a net benefit to SNR of using a longer exposure that is on the order of sqrt(exposure time). For example multiplying your exposure time by a factor of 4 theoretically should increase your SNR by a factor of 2.
- Frame stacking: Stacking frames has a similar affect on SNR as exposure time, the impact being proportional to sqrt(# frames). Thus stacking 4 frames will increase your SNR over that of a single frame by a factor of 2, 100 frames by a factor of 10, etc. For the full benefit of stacking to be realized, the noise experienced in your image must be entirely random and time varying (ie. no fixed pattern noise, no warm pixels, etc.).
- Binning: Binning your sensor has a beneficial impact on SNR. The improvement is proportional to the number of bins. For example binning 2x2 will increase your SNR by a factor of 2, bin 3x3 by a factor of 3, etc.
- Cooling: Since dark current noise is directly affected by the temperature of the sensor, application of cooling reduces noise and thus increases SNR.

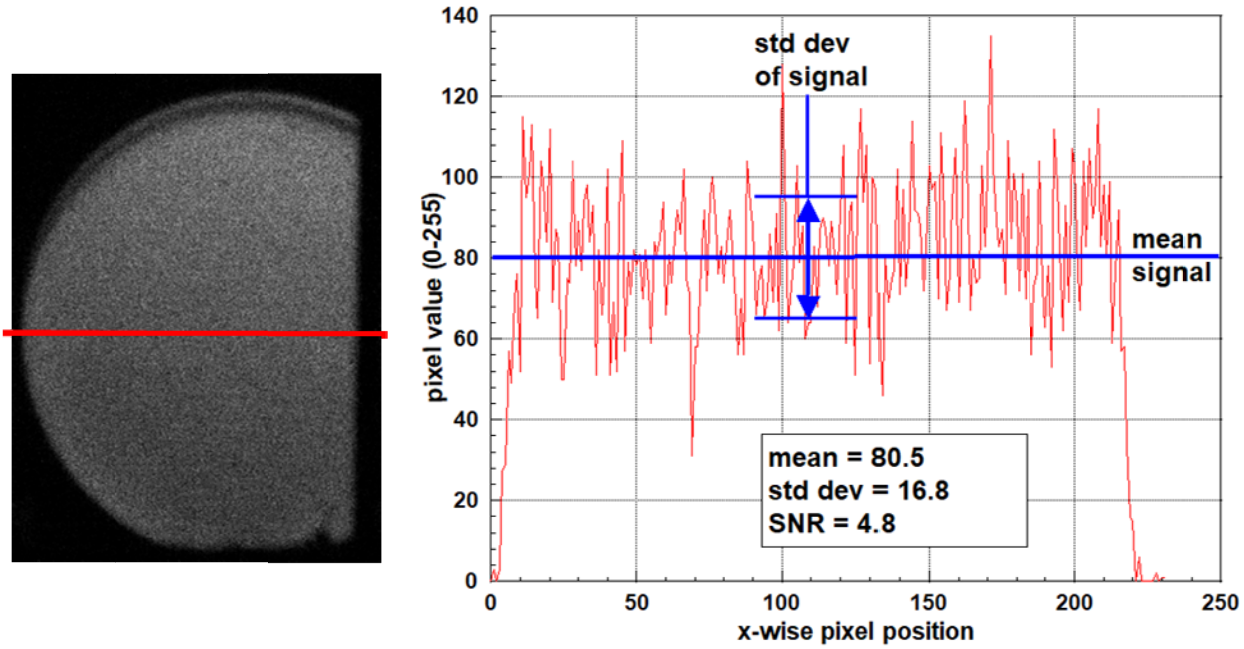

Figure 2 Example Calculation of Image SNR

From the perspective of a person using a camera for EAA, perhaps a more relevant performance parameter to use for comparing cameras is how long it takes to generate an image with an acceptable SNR level for observing. I have created such a parameter called the Time To Observable Image (TTOI), defined as the total amount of time required to achieve an image with SNR = 20. The target SNR value is somewhat arbitrary, but for the purposes of my camera

comparison testing a value of 20 does a pretty good job of tying my bench testing to what can be expected under real observing conditions, as will be explained more below.

Objectives:
In this third part of my comparison testing I will measure the SNR for each camera under a range of settings. User parameters varied during my testing include: exposure time, gain setting, and cooling level. From the measured data I also determine each camera's TTOI.

Methodology:
The main component of this test is the measurement of SNR. I achieved this by first constructing a representative indoor target, in my case a 4700K colour temperature halogen bulb powered by a regulated 5VDC power supply. The light is contained within an enclosed box with 8"x10" piece of opal glass at one end to act as a diffuser. On the other side of the opal glass is a ~1" x 0.5" rectangular orifice that forms the target at which I aimed the camera.

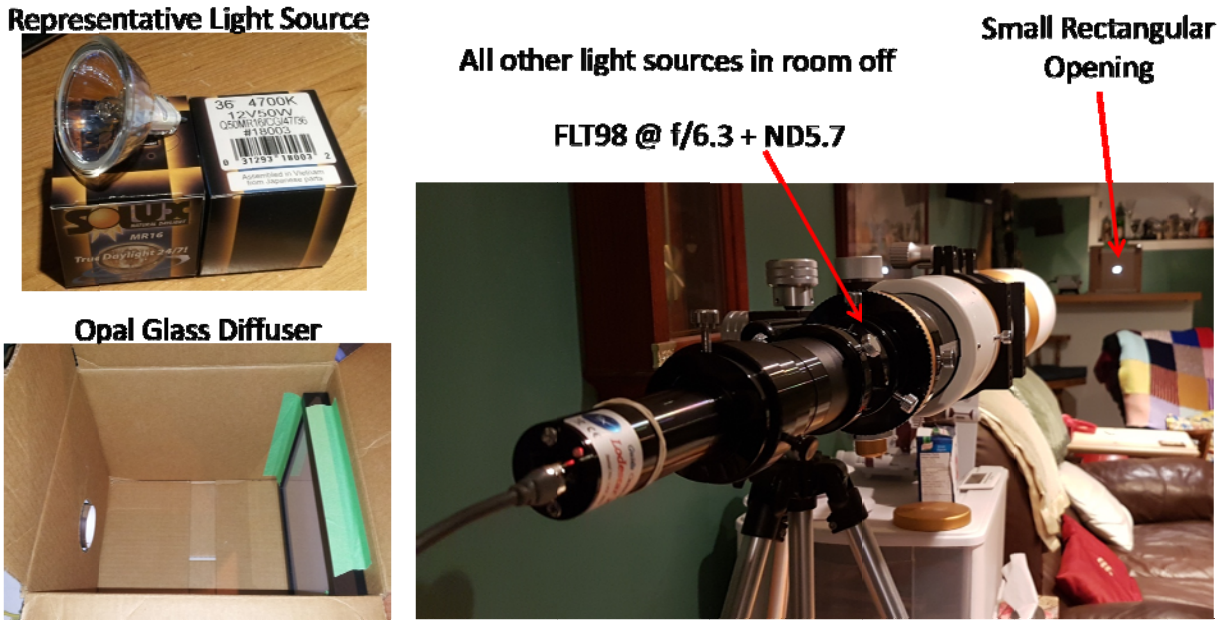

Figure 3 Reference Light Source for SNR Measurements

All of my measurements were made using a William Optics FLT98 APO refractor positioned across a darkened room, 20' away from the target. The telescope was operated at its native f/6.3, but I added a stack of neutral density filters (total ND5.7, 0.0002%) to reduce the brightness of my light source down to something more in line with a real deepsky target. For each camera I made my measurements with the focuser set at 10mm outside of focus to further ensure the uniformity of the target as observed by the camera. With this setup I measured single frames from each camera over a range of settings:

- Gain: On the ASI294 I used four gain settings: 0, 220, 430, and 570 (max). On the DS10C-TEC I used 1 (min) and 160 (max), for both the LCG and HCG settings. HCG engages automatically on the ASI294 for gain settings above 120.
- Exposure time: Varied from 50ms up to the max allowable by the camera software: 16 minutes on the DS10C-TEC, and 2000s (33 minutes) on the ASI294. Exposures varied depending on the gain used, ie. longer exposure times when less gain used.
- Cooling: Varied between TEC off and maximum TEC. TEC off temperatures on the DS10C-TEC were all around +25°C, and for the ASI294 were around +16°C. Maximum TEC temperatures for the DS10C-TEC were around -20°C, and for the ASI294 were -16°C.

Figure 3 below shows the typical appearance of one of the captured frames. All the frames were post processed using the following steps:

1. Crop out the uniform part of the target and a uniform part of the background from each image and save as separate BMP files;
2. Run each cropped image, signal and background, through a custom program that reads all the pixel intensity data and calculates the mean and standard deviation;
3. Import the mean and standard deviation data into a spreadsheet for further analysis, the main calculations being SNR and TTOI.

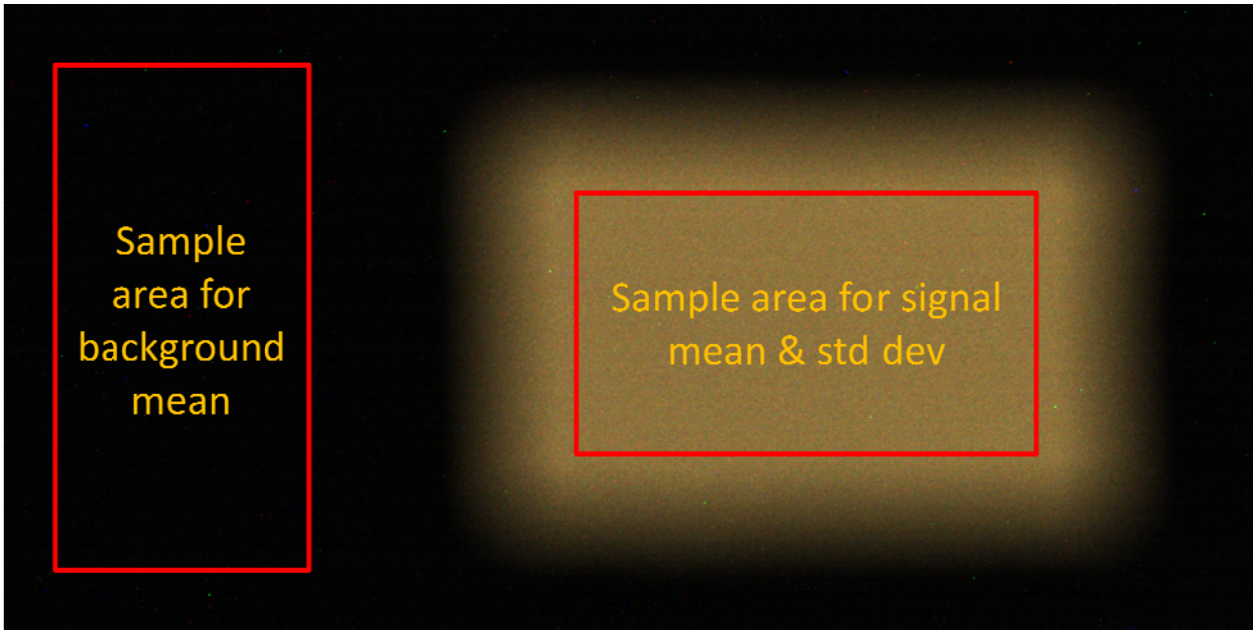

Figure 4 Sample Image Data From Testing

The source code for the program I wrote to extract mean and stddev is attached to the end of this test report. It is a very simple FORTAN code, crude but it does the job. I chose to use the 2-sigma clipped mean for my SNR calculations in order to minimize the impact of hot and warm

pixels on the calculation of the signal mean which would have artificially increased the SNR values. SNR was calculated using the following equation:

$$SNR_{signal} = (MEAN_{signal}\text{-}MEAN_{background})/STDDEV_{signal}$$

TTOI is calculated using the equation below:

$$TTOI = (SNR_{target}/SNR_{signal})^2 \times T_{exposure}$$

, where SNR_{target} has been set to 20 to give TTOI values that align well with my experience in practice using the ASI294 and other cameras for EAA.

During my data collection process, both cameras were used in 14/16 bit mode. However I saved images as viewed on the screen (ie. as they would be observed by the user doing EAA) into 8-bit per channel image files. I did this simply because it is much easier to work with 8-bit per channel images. I was willing to accept the reduction in accuracy, which really only affects data points at the lowest of exposure times, when my mean signal value was small.

Results:
Figures 5 and 6 below are plots of the measured SNR for the ASI294MC Pro and DS10C-TEC respectively. A log-log scale was required to capture the very large range of SNR and exposure time values encountered during the testing. There are a number of traits that seem to be common between the two cameras:

- SNR increases with exposure time, a behaviour that was expected and is consistent with theory. Note that for all the plotted data, the right-most data point on each curve was the longest exposure time achievable without saturating the image, or without running up against the max exposure limit set by the camera software.
- Increasing gain resulted in an increase in SNR for the same exposure time, but also a reduction if the max achievable SNR in a single frame due to the image saturating sooner. This ties to the fact that as camera gain is increased, the full well depth of the sensor reduces. Thus the largest SNR achievable with a single frame occurs when gain is set to minimum (ie. well depth is maximum).
- SNR increases with the application of sensor cooling. The extent to which the cooling improves the SNR varies depending on the exposure time, and varies differently between cameras as will be discussed more below.

For the ASI294 camera it appears that cooling does almost nothing to improve SNR for exposure times less than 100sec. This is an interesting observation since it implies that for EAA one does not need to use the TEC version of this camera. The TEC would only appear to be useful in AP applications where exposure times are in the multiple minute range. Even then, the improvement in SNR resulting from cooling is on the order of 2x at most.

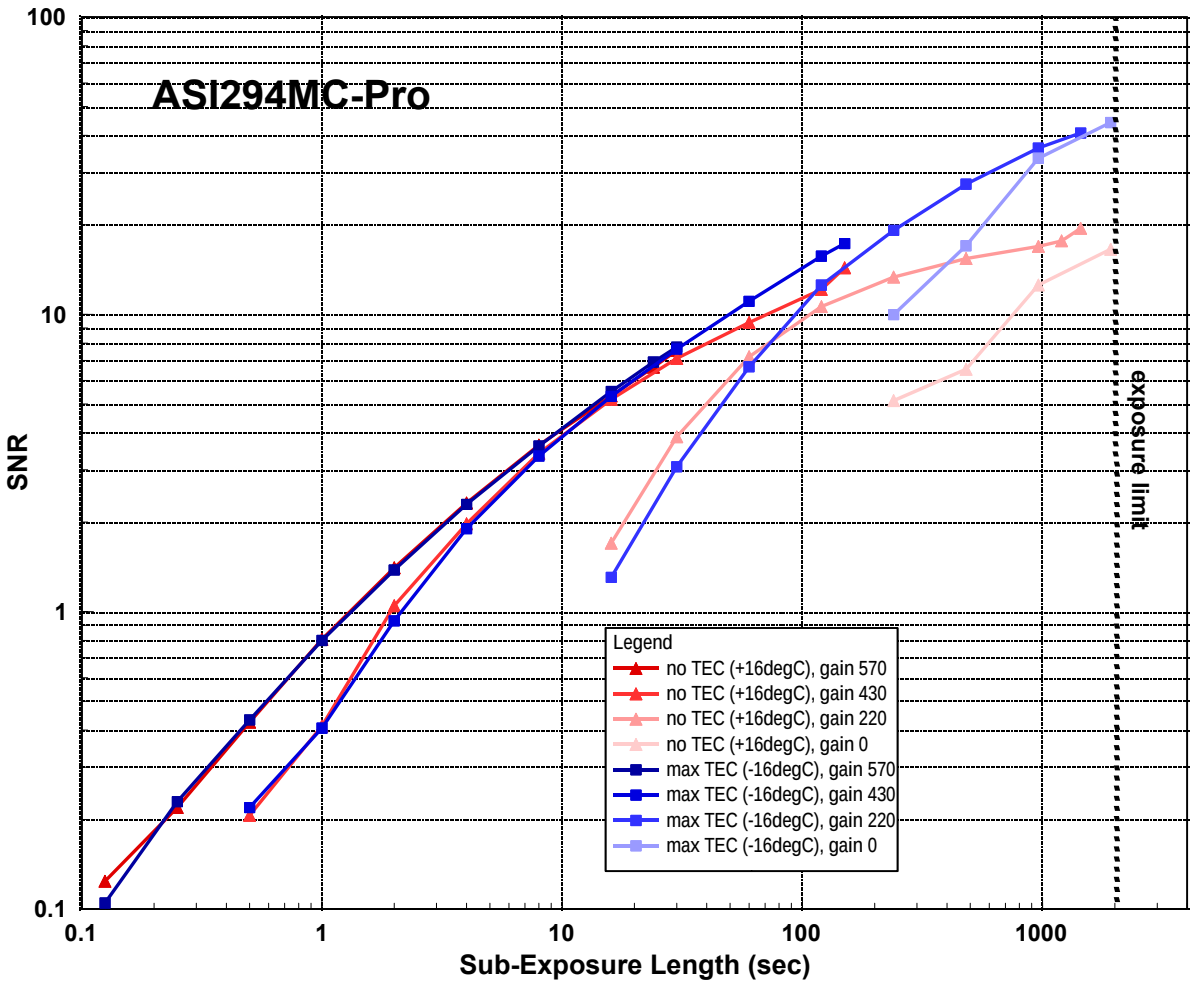

Figure 5 Measured SNR for ASI294MC Pro

For the DS10C-TEC, cooling has a much larger positive influence on SNR. When un-cooled the DS10C-TEC produces SNR values significantly lower than the ASI294. But with cooling engaged the DS10C-TEC produces SNR values that are comparable to that of the ASI294. Comparing SNR values more closely it appears that the cooled DS10C-TEC has SNR values superior to the ASI294 for short exposure times, exposures less than 100sec. For exposures greater than 100sec the ASI294 produces better SNR values. This observation implies that these two cameras are optimized for different applications. Although they are both capable of doing EAA and AP, the DS10C-TEC is optimized for EAA, and the ASI294 is optimized for AP. Whether or not this optimization was a conscious decision made by the two camera manufacturers, I don't know for sure, but it is a reasonable assumption.

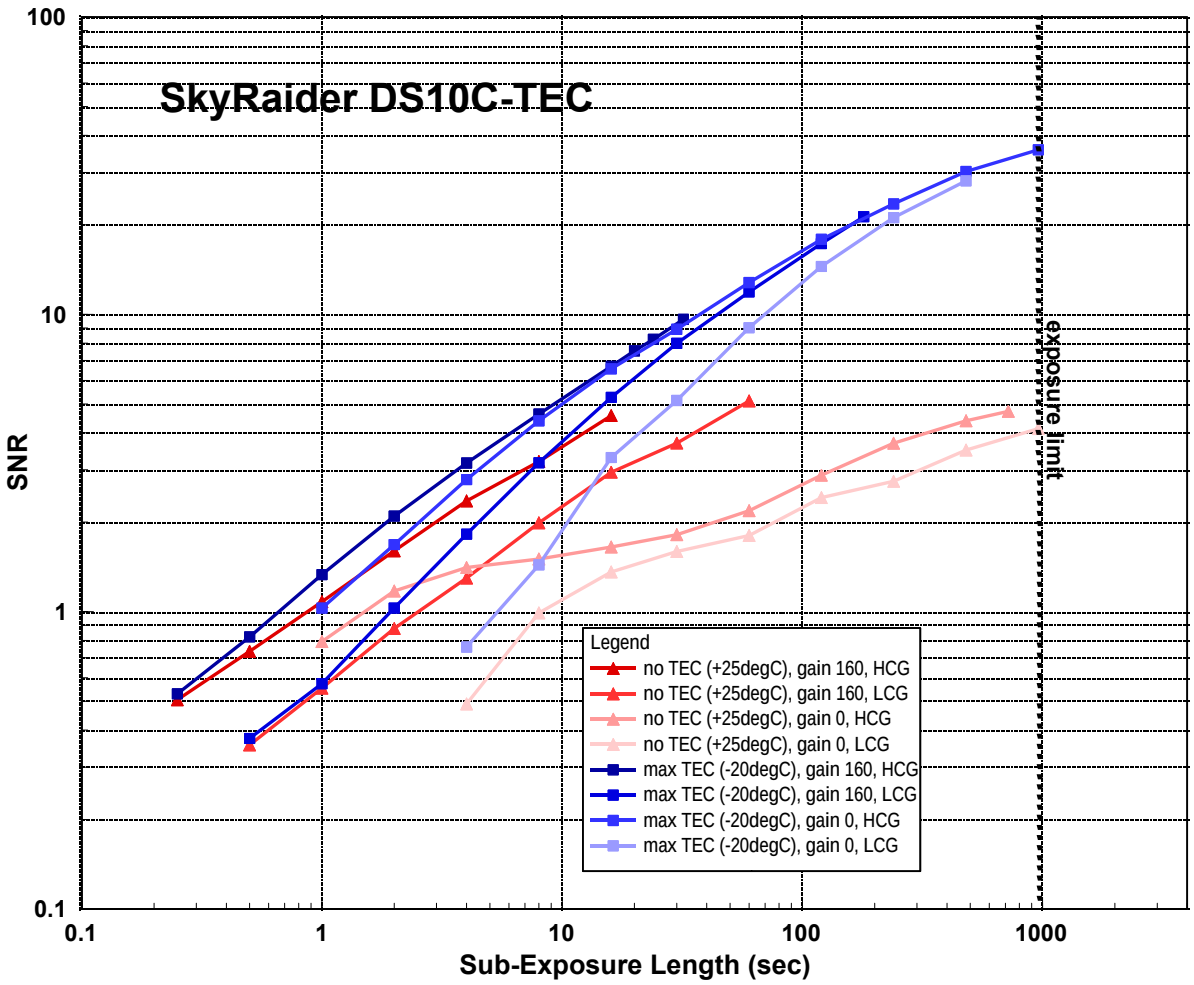

Figure 6 Measured SNR for DS10C-TEC

The TTOI plots for the two cameras are shown below in Figures 7 and 8. Some common observations that can be made from these plots are as follows:

- At each gain setting, there is very clearly a single exposure time that gives the minimum TTOI.
- If one looks at all the gain setting curves together the TTOI plots trace out an overall minimum TTOI curve for each camera. This family curve defines what gain and exposure combination will give the best overall TTOI for the camera. For the cooled ASI294 the combination is roughly Gain 430 + Exp 100sec, and for the cooled DS10C-TEC the combination is roughly Gain 0 (HCG) + Exp 30sec. The optimum values are different when cooling is off, in general pushing gain higher and exposure time lower.
- Cooling has a very clear positive impact on TTOI, much more obvious than the impact on SNR.

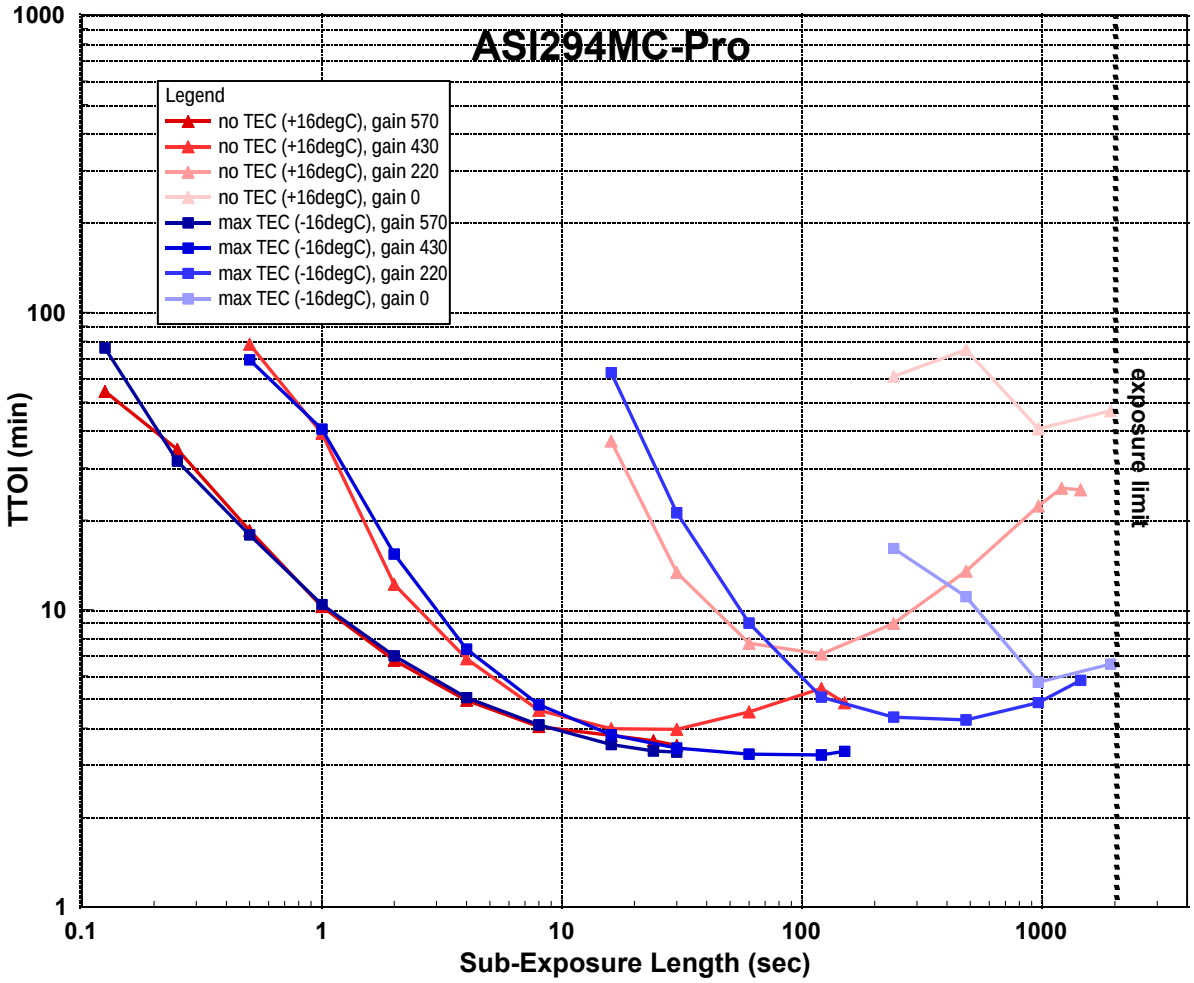

Figure 7 Measured TTOI(min) for ASI294MC Pro

With the ASI294 camera, cooling has little to no impact on TTOI for exposure times below 30sec. This is consistent with what was observed in the SNR data. Cooling does improve the overall best TTOI achievable with the camera, which is around 3.3 minutes compared with a no-TEC value of around 4 minutes. Note that the curves for the 0 gain case are not as smooth as those for the other gain settings because the absolute signal magnitude for those test points were very low. Better quality data for the 0 gain case would have been achieved if I had performed all my data gathering and analysis in 16-bit per channel instead of 8-bit per channel. Nonetheless, the end conclusions of my testing would have been the same if I had used 16-bit per channel.

On the DS10C-TEC camera, the improvement in TTOI realized by using cooling is quite large. Without cooling the DS10C-TEC delivers TTOI values significantly longer than the ASI294, but with cooling on the DS10C-TEC delivers TTOI values that are shorter than the ASI294. Using the optimum gain-exposure combination a minimum TTOI value of 2.3 minutes can be achieved, roughly 70% the best TTOI of the ASI294. The TTOI plots for the DS10C-TEC also rather flat on the bottom, allowing for the use of a wide range of gain-exposure combinations and still

achieve nearly optimum TTOI values. For the ASI294 the range of gain-exposure combinations giving near-optimum TTOI values is more narrow.

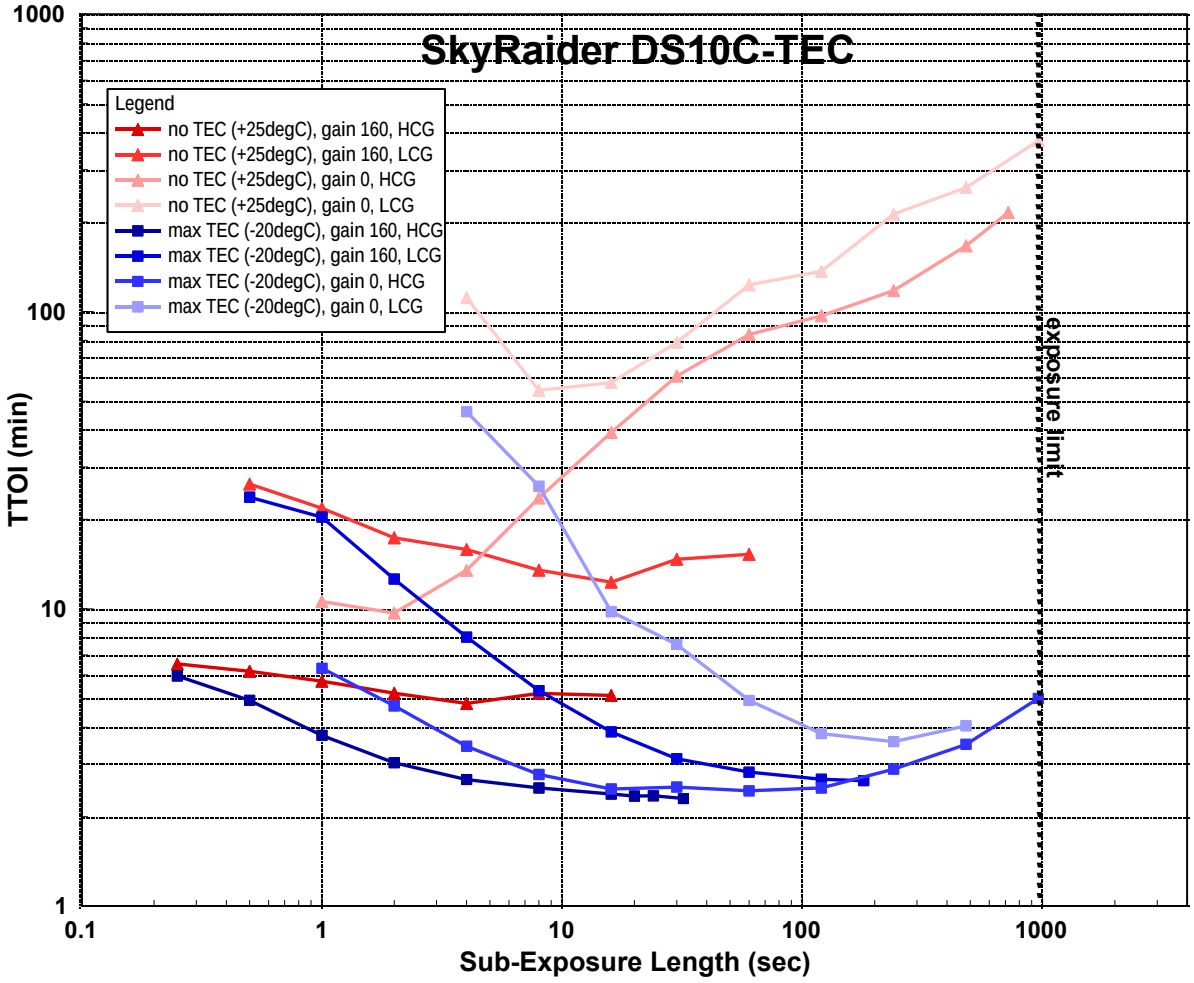

Figure 8 Measured TTOI(min) for DS10C-TEC

Conclusions:

In summary, the following conclusions have been drawn from my SNR and TTOI measurements:

1. The belief that having the same CMOS sensor in these two brands of camera means that they must have the same performance is not borne out by my test data. It would seem that two cameras using the same sensor CAN have quite different performance. The design and build of the remainder of the camera components, outside of the CMOS package, have as much to do with overall performance as the sensor itself.
2. With cooling off, the ASI294 has superior imaging performance compared with the DS10C-TEC. I believe that this is a result of a combination of higher sensor temperatures and higher gain used in the DS10C-TEC camera. On a side note, from a

previous batch of testing it appears that the non-TEC version of the DS10C has un-cooled performance that is much more similar to the ASI294 (see Figure 9 below).

3. With cooling on, the DS10C-TEC delivers imaging performance that is very similar to the ASI294. The DS10C-TEC performs better at exposure times below 100sec, and the ASI294 performs better at exposure times above 100sec.
4. I have used both cameras in practice for EAA (the DS10C non-TEC as well), and find them both to be very capable cameras. They have similarly low ampglow, and produce relatively low amounts of noise, making them both easy and enjoyable to use.

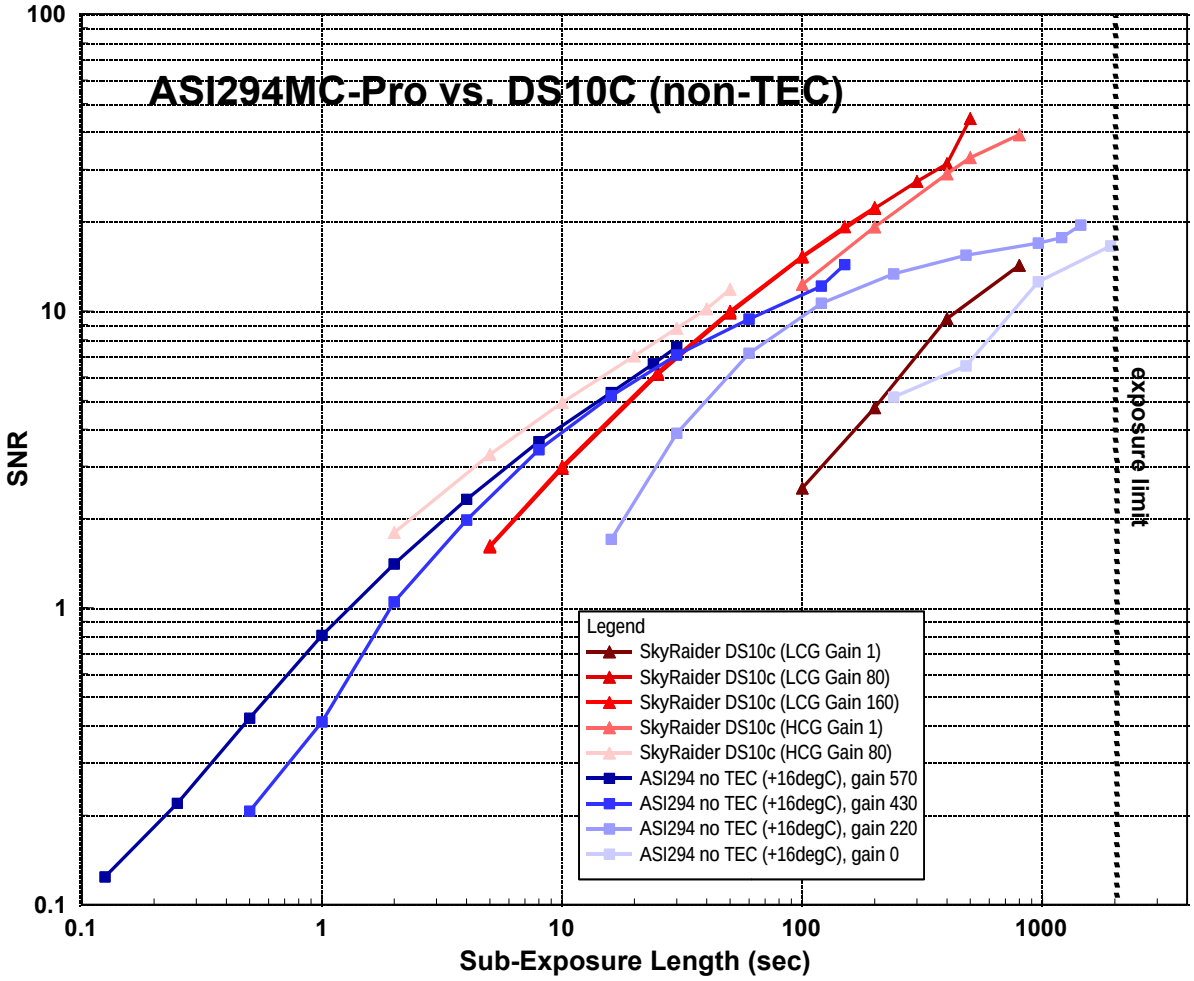

Figure 9 Measured SNR: ASI294MC-Pro vs DS10C (non-TEC)

The measurement of SNR and TTOI was the last major test activity I had planned. There are two smaller items that I still want to explore however: how the SNR of actual stacked frames compares to theory, and how resistant to sensor window fogging the two cameras are. Stay tuned!

Jim Thompson

Appendix: Image Analysis Source Code

```
!*****************************************************************************
!
!  PROGRAM: BMPSTATS
!
!  PURPOSE:  This program reads in a list of Windows 24bit .BMP files, and
!          writes out the histogram statistics for the image, including:
!          mean, standard deviation, and sigma clipped mean.  Developed
!          for use with Jim T's camera SNR measurement method.  Based off
!          BMPHISTO code.
!
!  DATE:  06-Dec-18
!
!   rev 0, 06-Dec-18: initial release
!
!*****************************************************************************

          program BMPSTATS

          use DFLIB

          implicit none
          integer I,J,K,MAXWIDTH,MAXHEIGHT,MAXFILES,irec,iwidth,iheight,ipad
          integer numfiles,numsig
          parameter(MAXWIDTH=1000,MAXHEIGHT=1000,MAXFILES=100)   ! limit max image size & number of images
          integer image(MAXHEIGHT,MAXWIDTH,3)        ! matrix to put image RGB values into
          character*24 buffer,outfile,infile(MAXFILES)
          integer*1 header(54),ch
          integer hiwi(8),numpixels
          real pad,mean(3),sigmean(3),stddev(3),gmean,gsigmean,gstddev,lrgb(3)

! read in script file that gives list of .bmp filenames
          call getarg(1,buffer)
          open(unit=1,file=buffer)
          read(1,*)numfiles ! number of .bmp files to process
          read(1,*)numsig    ! number * stddev to clip
          read(1,*)lrgb(1),lrgb(2),lrgb(3)
          read(1,*)outfile ! name of csv file to write results to
          open(unit=2,file=outfile)

! print header to output file
          write(2,*)'gmean,gsigmean,gstddev'

! ********************  Per File Loop *************************
! for each .bmp file listed in the script file, read file name and then
! read in the image data, then calculate the stats and write to output
          do 10 I=1,numfiles
```

```
                read(1,*)infile(I)
                print*,infile(I)
                open(unit=3,file=infile(I),form='binary',access='direct',recl=1)

! read .bmp file header
                do 2 irec=1,54
                        read(3,rec=irec) header(irec)
2               end do

! check header info to make sure .bmp is format needed
                if((header(11).ne.54).or.(header(29).ne.24).or.(header(31).ne.0)) then
                        print*,'sorry, can not handle this file'
                end if

! get image height and width
        do 5 irec=19,26
          if(header(irec).lt.0) then            ! header records are 1-byte integer -128 to 127, so change to 0 to 255
                                hiwi(irec-18)=header(irec)+256
                        else
                             hiwi(irec-18)=header(irec)
                        endif
5       end do
                iheight=hiwi(5)+256*(hiwi(6)+256*(hiwi(7)+256*hiwi(8)))
                iwidth=hiwi(1)+256*(hiwi(2)+256*(hiwi(3)+256*hiwi(4)))
                numpixels=iheight*iwidth
                print*,'height=',iheight,'width=',iwidth,'pixels',numpixels

! now read in image data
        pad=(1.0-((iwidth*0.75)-floor(iwidth*0.75)))*4
        print*,pad
                ipad=pad
                if(ipad.eq.4) then
                   ipad=0
                endif
                print*,'padding=',ipad
                irec=54
                image=0    ! reset image(j,k,l)
                do 20 J=1,iheight
                        do 30 K=1,iwidth
                                irec=irec+1
                                read(3,rec=irec) ch       ! read B
                                if(ch.lt.0) then          ! ch is 1-byte integer -128 to 127, so change to 0 to 255
                                        image(J,K,3)=ch+256
                                elseif(ch.gt.0) then
                                        image(J,K,3)=ch
                                endif
                                irec=irec+1
                                read(3,rec=irec) ch       ! read G
```

```
                        if(ch.lt.0) then            ! ch is 1-byte integer -128 to 127, so change to 0 to 255
                            image(J,K,2)=ch+256
                        elseif(ch.gt.0) then
                            image(J,K,2)=ch
                        endif
                        irec=irec+1
                        read(3,rec=irec) ch       ! read R
                        if(ch.lt.0) then            ! ch is 1-byte integer -128 to 127, so change to 0 to 255
                            image(J,K,1)=ch+256
                        elseif(ch.gt.0) then
                            image(J,K,1)=ch
                        endif
30                  end do
                    irec=irec+ipad
20              end do
                close(unit=3)

! find the image mean value for each colour channel
                mean=0    ! reset mean(L)
                do 40 J=1,iheight
                    do 50 K=1,iwidth
                        mean(1)=mean(1)+image(J,K,1)
                        mean(2)=mean(2)+image(J,K,2)
                        mean(3)=mean(3)+image(J,K,3)
50                  end do
40              end do
                mean(1)=mean(1)/numpixels
                mean(2)=mean(2)/numpixels
                mean(3)=mean(3)/numpixels

! find the image standard deviation for each colour channel
                stddev=0        ! reset stddev(L)
                do 60 J=1,iheight
                    do 70 K=1,iwidth
                        stddev(1)=stddev(1)+(image(J,K,1)-mean(1))**2
                        stddev(2)=stddev(2)+(image(J,K,2)-mean(2))**2
                        stddev(3)=stddev(3)+(image(J,K,3)-mean(3))**2
70                  end do
60              end do
                stddev(1)=(stddev(1)/numpixels)**0.5
                stddev(2)=(stddev(2)/numpixels)**0.5
                stddev(3)=(stddev(3)/numpixels)**0.5

! find the image sigma clipped mean value for each colour channel.  This is a method of ignoring
! hot pixels in the image that might affect the calculation of the mean.
                sigmean=0    ! reset sigmean(L)
                do 80 J=1,iheight
                    do 90 K=1,iwidth
```

```
                        if(image(J,K,1).ge.(mean(1)+numsig*stddev(1))) then
                            sigmean(1)=sigmean(1)+mean(1)
                        elseif(image(J,K,1).lt.(mean(1)+numsig*stddev(1))) then
                            sigmean(1)=sigmean(1)+image(J,K,1)
                        end if
                        if(image(J,K,2).ge.(mean(2)+numsig*stddev(2))) then
                            sigmean(2)=sigmean(2)+mean(2)
                        elseif(image(J,K,2).lt.(mean(2)+numsig*stddev(2))) then
                            sigmean(2)=sigmean(2)+image(J,K,2)
                        end if
                        if(image(J,K,3).ge.(mean(3)+numsig*stddev(3))) then
                            sigmean(3)=sigmean(3)+mean(3)
                        elseif(image(J,K,3).lt.(mean(3)+numsig*stddev(3))) then
                            sigmean(3)=sigmean(3)+image(J,K,3)
                        end if
90                  end do
80              end do
                sigmean(1)=sigmean(1)/numpixels
                sigmean(2)=sigmean(2)/numpixels
                sigmean(3)=sigmean(3)/numpixels

! combine RGB values into a single intensity value
                gmean=lrgb(1)*mean(1)+lrgb(2)*mean(2)+lrgb(3)*mean(3)
                gsigmean=lrgb(1)*sigmean(1)+lrgb(2)*sigmean(2)+lrgb(3)*sigmean(3)
                gstddev=lrgb(1)*stddev(1)+lrgb(2)*stddev(2)+lrgb(3)*stddev(3)

! print the stats to the outfile
                print*, mean(1),mean(2),mean(3)
                print*, stddev(1),stddev(2),stddev(3)
                print*, sigmean(1),sigmean(2),sigmean(3)
                print*, gmean,gsigmean,gstddev
                write(2,100)gmean,',',gsigmean,',',gstddev
100             format(F9.5,A1,F9.5,A1,F9.5)

10      end do
        close(unit=1)
        close(unit=2)
        print *, 'All done list in ',buffer

        end program BMPSTATS
```