Advanced PixInsight PixelMath Operations

David Ault

Why use PixelMath

- PixelMath is a very powerful tool that gives you access to all sorts of features that otherwise would require javascripts, plug-in development (PCL) or standalone programs
- I use it regularly for:
  - Blending images with various functions (averaging, max, min, etc.)
  - Hot pixel removal
  - Altering or creating masks
  - Testing calibration data
  - Linear gradient based clipping or merging
  - Noise reduction
  - Drawing lines, circles or other geometry on an image
  - Removing unwanted artifacts (star halos, etc.)

Syntax

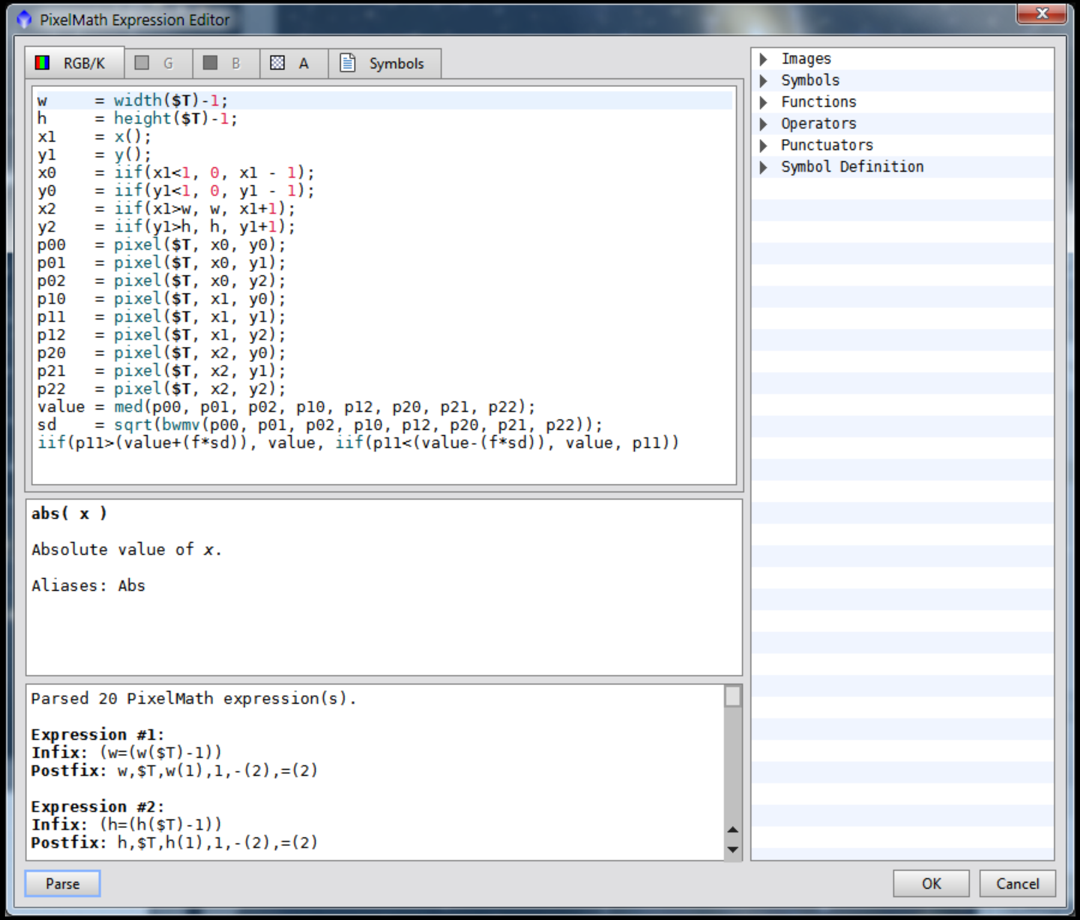
- Symbols
  - These are the equivalent of variables in programming languages
  - For every assignment made in the expression space there must be a corresponding symbol
  - Symbols can also be assigned values but only constants. This is the equivalent of initializing variables.
  - There is a built in symbol, **$T**, which is used to reference the active or new instance view (T stands for target)
  - Symbols need to be separated by commas
- Expressions
  - This is where all the math is done
  - All sorts of functions are available from, log to sine to image specific functions like biweight midvariance
  - The PixelMath engine can handle parenthetical equations
  - Symbols are available for most functions: addition, subtraction, multiplication, powers.
  - Expressions are separated by semi-colons with the result from the last expression returned as the pixel value

How it works

- PixelMath runs in a big loop over each pixel in the active view
- For example, the expression **0.5 * $T** will multiply every pixel in the target view by 0.5 reducing the brightness by half
- There are functions available for determining where you are in the image loop like **x()** & **y()** which can be used for targeting location based variations
- There are also functions that are non loop based like **mean()**, **median()**, **bwmv()**. These operate on an entire image and return a single value result. Some of these functions, like **mean()**, can also operate on a list of values.

The Expression Editor

- You can do a lot of math with the base PixelMath form but if you want to do more complex functions spanning multiple equations the Expression Editor is very useful
- You get quick access to Images, Symbols, Function, etc. which can be added to your expression by double clicking on them
- You can also check the syntax of your expressions without running it on the entire image
- There is also syntax highlighting making it easier to read the expressions



Examples

- Image Blending
  - Weighted linear blend (also called alpha blend or weighted averaging)
  - Photoshop equivalents
  - Star mask combination
  - Synthetic channel generation
- Rendering
  - Inserting lines and circles
  - Cross-sections
- Hot pixel removal / noise reduction
- Star Halo Removal
- Manual calibration evaluation

Reference

Image Blending

- Alpha Blend
  - RGB/K: a*Image1 + (1-a)*Image2
  - or
  - RGB/K: 0.4*R + 0.3*G + 0.3*B
  - Symbols: a=0.5
- Synthetic Green
  - G: iif(B>0.5, 1-(1-R*(1-(B-0.5))), R*(B+0.5))
  - G: tg = 0.1*R + 0.9*B;
    a*tg + (1-a)*min(tg, (R+B)/2)
  - Symbols: tg, a=0.5
- Star Mask Combination
  - RGB/K: max(star_mask, star_mask1, star_mask2)

Photoshop blending modes

The a and a-1 portions of the equations are the alpha blend. This equates to the opacity slider in Photoshop except it has a range of 0 to 1 instead of 0 to 100.

- Normal
  a*top + (1-a)*bot
- Multiply
  a*top*bot + (1-a)*bot
- Screen
  a*(1-(1-top)*(1-bot)) + (1-a)*bot
- Overlay
  a*iif(bot<0.5, 2*bot*top, 1-2*(1-top)*(1-bot)) + (1-a)*bot
- Darken
  a*min(top, bot) + (1-a)*bot
- Lighten
  a*max(top, bot) + (1-a)*bot
- Addition
  a*(top + bot) + (1-a)*bot
- Subtraction
  a*(top - bot) + (1-a)*bot
- Division
  a*(top / bot) + (1-a)*bot

More Photoshop blending modes

- Linear Burn

  a*(top+bot-1) + (1-a)*bot
- Color Burn

  a*(1-(1-top)/bot) + (1-a)*bot
- Color Dodge

  a*(top/(1-bot)) + (1-a)*bot
- Soft Light

  a*iif(bot>0.5, 1-(1-top)*(1-(bot-0.5)), top*(bot+0.5)) + (1-a)*bot
- Hard Light

  a*iif(bot>0.5, 1-((1-top)*(1-2*(bot-0.5))), 2*top*bot) + (1-a)*bot
- Exclusion

  a*(0.5-2*(top-0.5)*(bot-0.5)) + (1-a)*bot

Rendering

- Simple Circle
  - RGB/K: `r = sqrt((x()-cx)^2 + (y()-cy)^2); iif(abs(tr-r)<0.5, 1, $T)`
  - Symbols: `cx=500, cy=500, tr=400, r`
- Horizontal Line
  - RGB/K: `iif(x()==xloc, 1, $T)`
  - Symbols: `xloc=685`
- Aliased Circle
  - RGB/K: `r = rdist(cx, cy); a = abs(tr-r)/(w/2); iif(a<1, a*$T+(1-a), $T)`
  - Symbols: `cx=1700, cy=1200, tr=300, w=5, r, a`
- Aliased Line
  - RGB/K `r = d2line(x1, y1, x2, y2); a = (r/(w/2))^0.5; iif(a<1, a*$T+(1-a), $T)`
  - Symbols: `x1=332, y1=788, x2=1472, y2=1112, tr=300, w=5, r, a`

Rendering

- Green Tick Mark
  - R: `iif(((x()>(cx+xo)) && (x()<(cx+xo+xl)) && (y()==cy)) || ((y()>(cy+yo)) && (y()<(cy+yo+yl)) && (x()==cx)), 0, $T)`
  - G: `iif(((x()>(cx+xo)) && (x()<(cx+xo+xl)) && (y()==cy)) || ((y()>(cy+yo)) && (y()<(cy+yo+yl)) && (x()==cx)), 1, $T)`
  - B: `iif(((x()>(cx+xo)) && (x()<(cx+xo+xl)) && (y()==cy)) || ((y()>(cy+yo)) && (y()<(cy+yo+yl)) && (x()==cx)), 0, $T)`
  - Symbols: cx=345, cy=322, xo=15, yo=15, xl=30, yl=30
- Line Segment
  - RGB/K: `d = d2seg(llx, lly, urx, ury); a = 1 - d/(lw/2); iif(d<(lw/2), a + (1-a)*$T, $T)`
  - Symbols: llx=30, lly=356, urx=965, ury=179, lw=5, d, a
- Highlight Box in Yellow
  - R: `$T[0]`
  - G: `$T[1]`
  - B: `iif(x()>llx && x()<urx && y()>lly && y()<ury, 0, $T[2])`
  - Symbols: llx=32, lly=374, urx=723, ury=403

Cross Section analysis

- Cross-section variation (two pass)

RGB/K: pixel($T, x(), 0.5*h($T))

RGB/K: iif(((1-$T)*h($T))>y(), 0, $T)

or

RGB/K: d = abs(((1-CIEL($T))*h($T))-y());
iif(d>r, 0, r-d) where r=3

Hot Pixel Removal

- Symbols

```
f=9.0, w, h, x0, x1, x2, y0, y1, y2, p00, p01, p02, p10, p11, p12,
p20, p21, p22, value, sd
```

- RGB/K

```
w    = width($T)-1;
h    = height($T)-1;
x1   = x();
y1   = y();
x0   = iif(x1<1, 0, x1 - 1);
y0   = iif(y1<1, 0, y1 - 1);
x2   = iif(x1>w, w, x1+1);
y2   = iif(y1>h, h, y1+1);
p00  = pixel($T, x0, y0);
p01  = pixel($T, x0, y1);
p02  = pixel($T, x0, y2);
p10  = pixel($T, x1, y0);
```

- RGB/K continued

```
p11  = pixel($T, x1, y1);
p12  = pixel($T, x1, y2);
p20  = pixel($T, x2, y0);
p21  = pixel($T, x2, y1);
p22  = pixel($T, x2, y2);
value = med(p00, p01, p02, p10, p12, p20, p21, p22);
sd   = sqrt(bwmv(p00, p01, p02, p10, p12, p20, p21, p22));
iif(p11>(value+(f*sd)), value, iif(p11<(value-(f*sd)), value,
p11))
```

- I found biweight midvariance to be more robust for such a small set of pixels compared to standard deviation

Removing Purple Stars

- Magenta Star Reduction

  R: `$T[0]`

  G: `iif(min($T[0],$T[2])>$T[1],min($T[0],$T[2]),$T[1])`

  B: `$T[2]`

- In order to work on just stars this needs to be combined with a good star mask.

Calibration Math

- Bias and flats only, assuming flats have been calibrated w/ dark flat or bias frames
  calibrated light = (light - bias) * mean(flat) / max(0.00002, flat)
- Bias, scaled darks and flats, assuming flats have been calibrated w/ dark flats or bias frames
  calibrated light = ((light - bias) - k*(dark - bias)) * mean(flat) / max(0.00002, flat)
- Dark and flats only, assuming flats have been calibrated w/ dark flats or bias frames
  calibrated light = (light - dark) * mean(flat) / max(0.00002, flat)
- Bias and Flats Only, with uncalibrated flats
  calibrated light = (light - bias) * (mean(flat) - mean(bias)) / (max(0.00002, flat-bias)
- Bias, scaled darks and flats, with uncalibrated masters
  calibrated light = ((light - bias) - k*(dark -bias)) * (mean(flat) - mean(bias)) / (max(0.00002, flat-bias)
- In most cases bias and dark flats are interchangeable, however if your flat frames are very long and your sensor has high dark current then dark flats will work better

Resources

- http://pixinsight.com.ar/en/
- http://pixinsight.com/forum/index.php?board=11.0
- http://en.wikipedia.org/wiki/Blend_modes
- http://harrysastroshed.com/pixinsight/pixinsight%20video%20html/Pixinsighthome.html
- http://pixinsight.com/tutorials/master-frames/index.html

- Handbook of CCD Astronomy – Howell
- Lessons from the Masters – Gendler et al.