NAME

imadjust - adjusts pixel values in an image

SYNOPSIS

imadjust [*options*] *infilename* [*outfilename*]

DESCRIPTION

imadjust adjusts pixel values in images in *infilename* and saves the adjusted images to *outfilename*. If no *outfilename* is given, imadjust saves the adjusted images back into *infilename*.

Adjustments can manipulate pixel color or alpha channel values in a wide variety of ways, depending upon the imadjust options that are chosen.

OPTIONS

imadjust has numerous options in the following categories:

File Selection	What input and output files to use
Format Selection	What image file format to use
Format Control	What variant of a file format to generate
Standard	Standard generic options on all SDSC tools
Pixel Selection	How to select which pixels to adjust
Adjustment Operation	How to adjust the pixels
Change Selection	What aspect of pixels to adjust

File Selection, *Format Selection*, *Format Control*, and *Standard* options are common to all SDSC image tools and are discussed in depth in the man page for imconv(1IM).

All options can be abbreviated to the first few unique characters.

In a typical operation, a user may select all pixels with a red component value greater than 100 and add 20 to their red component. This would have the affect of brightening the reds of an image.

As another example, a user may select all pixels with saturation values (in the HSI color space) greater than 0.8 (1.0 is maximum) and clamp them to 0.8. This would, for instance, prevent highly saturated images from bleeding when they are recorded to video.

Pixel Selection

During adjustment, imadjust scans the input image looking for pixels that meet a user-defined selection criterion. All pixels that match that criterion will be adjusted. All pixels that do not match the selection criterion are copied to the output image untouched.

Statements like the following are typical selection criteria:

- All pixels with a red component less than 100.
- All pixels with a saturation component greater than 0.8.
- All pixels with an alpha channel value of 255.
- All pixels with an intensity component between 0.0 and 0.2.

In all cases, the selection criterion gives a pixel component (such as red, green, hue, saturation, color index, alpha, etc.) and what value or range of values it must have in order to pass the test and be adjusted. The following options define the selection criteria:

Option	Select pixels based on
-red *range*	red component
-green *range*	green component
-blue *range*	blue component
-mono *range*	monochrome component
-index *range*	color index component
-hue *range*	hue component
-saturation *range*	saturation component
-intensity *range*	intensity component
-alpha *range*	alpha component

imadjust allows at most one selection criterion. To adjust based upon multiple selection criteria, invoke imadjust multiple times on the same image, once for each criteria.

The -red, -green, and -blue options operate on pixels in the RGB color space.

The -mono option operates on pixels which store monochrome values per pixel. Input images that are not monochrome images will be rejected with an error message from imadjust.

The -index option operates on pixels which store monochrome, grayscale, or color index values per pixel. Input images that are not monochrome, grayscale, or color index images will be rejected with an error message from imadjust.

The -hue, -saturation, and -intensity options operate on pixels in the HSI color space. To check the selected criteria imadjust converts each pixel from a monochrome, grayscale, color index, or RGB value into the HSI color space. The image's pixel value is not altered by this testing. (See the imintro(3IM) man page for a description of RGB and HSI color spaces.)

The -alpha option operates on the pixel's alpha (coverage mask) component. Alpha values range from 0 (transpareant) to 255 (opaque). If this option is used with input images without alpha components, alpha components will be created for each pixel and initialized to 255 (opaque).

Each of the selection options accept an argument giving a single value, or a range of values that the pixel must have in order to match the selection criteria and be adjusted. Value *range* syntax takes any one of the following forms:

range	Select on pixels with values of
n	*n* only.
n-	*n* through the maximum legal value for the component.
n-*m*	*n* through *m*.

There is no space between *n*, the dash, and *m*.

The following would be typical selection criteria:

```
-red 0-99
-saturation 0.8-1.0
-alpha 255
-intensity 0.0-0.2
```

Adjustment Operation

Pixels that match the selection criteria will be adjusted depending upon which adjustment operation is chosen. The following operations are available:

Option	Action
-add	Add a value to each pixel that matches
-subtract	Subtract a value from each pixel that matches
-multiply	Multiply each matching pixel by a value
-divide	Divide each matching pixel by a value
-set	Set each matching pixel to a new value

imadjust allows at most one adjustment operation at a time. To perform multiple operations, invoke imadjust multiple times, once for each operation.

Change Selection

Pixels that match the selection criterion will be adjusted using the adjustment operation. Change options select what aspect of the pixel will be adjusted. For instance, the following statements are typical examples of change options:

- Set red to 200.
- Multiply intensity by 0.5.
- Set alpha to 255.
- Add 4 to the color index.
- Set saturation values to the range 0.0 to 0.8.

Each of these examples selects an operation, like "set", "multiply", "add", etc., and a pixel component to operate upon, like "red", "intensity", "alpha", "color index", "saturation", etc. The operation is selected using the operation options discussed in the previous section (-add, -subtract, etc). The component to operate upon is determined by the choice of any of the following options:

Option	Which pixel component to change
-red *range*	red component
-green *range*	green component
-blue *range*	blue component
-mono *range*	monochrome component
-index *range*	color index component
-hue *range*	hue component
-saturation *range*	saturation component
-intensity *range*	intensity component
-alpha *range*	alpha compoent

The option format is the same as that used for the selection criteria. Command-line order determines whether a -red option, for instance, is a selection criteria, or a change option. Selection criteria options precede the operation argument, while change options follow it.

The -red, -green, and -blue change options cause the red, green, or blue component of the pixel to be adjusted. Input images must be RGB images in order to adjust their red, green, or blue components.

The -mono change option causes the monochrome value of a pixel to be adjusted. Input images must be monochrome images in order to adjust their monochrome values.

The -index change option adjusts the color index of a pixel. Input images must be monochrome, grayscale, or color index images to allow their color index values to be adjusted.

The -hue, -saturation, and -intensity change options cause the hue, saturation, or intensity aspects of a pixel to be adjusted. Input images must be RGB images to allow their red, green, or blue components to be adjusted. RGB pixel values are converted into the HSI color space; the hue, saturation, or intensity component is adjusted; and the result is converted back into the RGB color space.

The -alpha change option adjusts the alpha component of a pixel. If this option is used with input images without alpha components, the alpha components will be created for each pixel and initialized to 255 (opaque), prior to adjusting.

Each of the change options accept an argument giving a single value, or a range of values to be used by the adjustment operation. Value *range* syntax takes any one of the following forms:

range	Operate using values of
n	*n* only.
n-	*n* through the maximum legal value for the component.
n-m	*n* through *m*.

There is no space between *n*, the dash, and *m*.

The meaning of the change option's value or *range* of values depends upon the adjustment operation:

-add
: The value is added to each adjusted pixel.

-subtract
: The value is subtracted from each adjusted pixel.

-multiply
: The value is a multiplication factor by which the adjusted pixel's component should be multiplied.

-divide
: The value is a division factor by which the adjusted pixel's component should be divided.

-set
: The value is a new value for the adjusted pixel's component. This is a set operation.

In each case, if the resulting pixel value is greater than the maximum legal value for the pixel component (such as greater than 255 for red), or less than the minumum legal value for the component (such as -42 for red), the value is clamped to the maximum or minimum before it is written to the output image.

Selection criteria and change options may each specify a single value or a range of values. The following are typical command lines:

```
imadjust   -intensity 0.0       -set        -alpha 0              infile.ras outfile.pix
imadjust   -red 0-100           -add        -red 5                infile.ras outfile.pix
imadjust   -index 10            -subtract   -alpha 5-40           infile.ras outfile.pix
imadjust   -saturation 0.0-1.0  -set        -saturation 0.0-0.8   infile.ras outfile.pix
```

The selection criterion and the change option may be a single value or a value range. This leaves four possible combinations:

1. Both are single values.
   All pixels with the single value selected by the selection criterion will be modified using the single value in the change option. For instance,

   ```
   -intensity 0.0 -set -alpha 0
   ```

   changes all pixels with intensities of 0.0 to have alpha values of 0.

2. The selection criterion has a range but the change option is a single value." All pixels within the range of selection criteria values will be modified using the single value in the change option. For instance,

   ```
   -red 0-100 -add -red 5
   ```

   adds 5 to all pixels with red values between 0 and 100.

3. The selection criterion has a single value but the change option has a range." To have a change option of range, but no selection range doesn't make sense. This variation will be treated as if the change range is only the single starting value.

4. Both have a value range.
   Pixels matching the first value of the selection criteria value range will be adjusted using the first value of the change option. Likewise, pixels matching the second value of the selection criteria value range will be adjusted by the second value of the change option. Pixels with values in between will be adjusted using change values in between. For instance,

   ```
   -saturation 0.0-1.0 -set -saturation 0.0-0.8
   ```

   sets all pixels with saturation values between 0.0 and 1.0 to the new range 0.0 to 0.8. Original pixels with values of 1.0 will be mapped to values of 0.8, and so on.

Interesting settings may be done by using value ranges where the first value in the range is greater than the second value. For instance,

```
-intensity 0.0-1.0 -set -intensity 1.0-0.0
```

will set intensities from 0.0 to 1.0 to intensities from 1.0 to 0.0. In other words, the intensity values will be inverted, creating a pseudo-negative effect.

NOTES

Error messages are reported to stderr.

Different pixel components have different types and different minimum and maximum values. This affects how value ranges given for, say, -red versus -intensity, will be interpreted.

Value	Type	Minimum	Maximum
red	Integer	0	255
green	Integer	0	255
blue	Integer	0	255
monochrome	Integer	0	1
index	Integer	0	255
hue	Float	0.0	360.0
saturation	Float	0.0	1.0
intensity	Float	0.0	1.0
alpha	Integer	0	255

For those comfortable viewing pseudo-code, the following code illustrates the adjustment algorithm used by imadjust. *criteriaRange* is the value range given with the selection criterion (such as -red 5-10). *changeRange* is the value range given with the change selection (such as -alpha 100-200). In the algorithm, both ranges have been normalized to values between 0.0 and 1.0. Finally, pixel.*criteriaComponent* is the pixel color component being tested in the selection criterion (such as -red), while pixel.*changeComponent* is the pixel color component being changed (such as -intensity).

```
factor = (changeRange.second - changeRange.first) /
      (criteriaRange.second - criteriaRange.first)
for each pixel in the input image,
      if pixel.criteriaComponent is within criteriaRange,
            value = (pixel.criteriaComponent - criteriaRange.first) *
                  factor + changeRange.first
            switch on the operation,
                  add:       pixel.changeComponent += value
                  subtract:  pixel.changeComponent -= value
                  multiply:  pixel.changeComponent *= value
                  divide:    pixel.changeComponent /= value
                  set:   pixel.changeComponent = value
            if pixel.changeComponent > maximum,
                  pixel.changeComponent = maximum
            if pixel.changeComponent < minimum,
                  pixel.changeComponent = minimum
```

EXAMPLES

Example #1

Vivid, high-saturation images look beautiful on computer graphics screens. Unfortunately the color gamut supported by such screens is wider than that available on video devices, such as home VHS and S-VHS decks. High-saturation images recorded to video produce undesirable color effects. To prevent this, imadjust may be used to uniformly desaturate an image by setting saturation values normally in the range 0.0 to 1.0 to the less-saturated range 0.0 to 0.8.

```
imadjust vivid.rgb -saturation 0.0-1.0 -set -saturation 0.0-0.8 video.rgb
```

In this example, -saturation 0.0-1.0 is the selection criterion. All pixels with saturation values between 0.0 and 1.0 will be selected. Since this is the full range possible for saturation, this effectively selects all pixels.

This example uses the -set operation to cause pixel values to be set, rather than added to, subtracted from, and so on.

Finally, -saturation 0.0-0.8 indicates that the saturation component of the pixel's color will be changed to the new range 0.0 to 0.8.

Example #2

In order to save image compute time, animators often compute a static background once, without any moving objects in front of it. The background objects are then removed from the scene, speeding up scene computation, and the animation is computed with the moving objects moving about in front of a black background. To finish the animation, the individual black background frames need to be pasted atop the static background, one frame at a time. This operation is known as compositing and can be done with imcomp(1IM).

When compositing, the compositing program needs to know what pixels of the "front" image are transparent and let pixels in the "back" image show through. This is indicated by a coverage mask known as an "alpha" plane. Alpha values of 0 are considered transparent, while values of 255 are opaque.

To composite the moving object frames atop the static background image, the black pixels of the moving object frames need to be marked as transparent: their alpha components need to be set to 0. This can be done using imadjust by selecting all pixels with intensities of 0 (black) and setting their alpha values to 0 (transparent).

```
imadjust moving.rgb -intensity 0.0 -set -alpha 0 moving_alpha.rgb
```

In this example, -intensity 0.0 selects all black pixels and -set requests that pixel values be set to new values. -alpha 0 indicates that the alpha component of these pixels should be set to 0.

The resulting file moving_alpha.rgb is ready for use in a digital compositor, such as imcomp(1IM).

Example #3

An image may be brightened by increasing the intensity value for all pixels.

```
imadjust dim.hdf -red 0-255 -add -intensity 0.2 bright.rla
```

-red 0-255 selects all pixels with red values between 0 and 255. Since this is the full range of the red component, this effectively selects all pixels in the image. We could have used -green 0-255 or -blue 0-255 or -hue 0.0-360.0 or any of several other combinations. All of these would select all pixels.

-add -intensity 0.2 adds 0.2 to the intensity component, in the HSI color space, of all pixels. Remember that intensity is measured from 0.0 to 1.0, not 0 to 255.

Example #4

A pseudo-negative of an image can be created by inverting its intensity values. Dark values will become light, while light values will become dark. The hue and saturation will remain the same.

```
imadjust normal.iff -intensity 0.0-1.0 -set -intensity 1.0-0.0 invert.ras
```

Example #5

Interesting color effects can be created by tweaking individual color components of image pixels. For instance:

```
imadjust normal.pix -hue 0.0-360.0 -add -hue 20.0 weird.pix
imadjust normal.pix -red 0-100 -multiply -blue 0.5 weird.pix
imadjust normal.pix -saturation 0.25-0.75 -set -hue 360.0-0.0 weird.pix
imadjust normal.pix -blue 0-100 -subtract -intensity 0.5-0.7 weird.pix
```

SEE ALSO

imcomp (1IM), imdissolve (1IM), imfill (1IM), ImVfbAdjust (3IM)

For information on SDSC's image library, see imintro(3IM).

AUTHOR

Chris Groening and Dave Nadeau
San Diego Supercomputer Center

See the individual file format man pages for the authors of the underlying format read and write code. The names of these man pages begin with the letters "im", followed by the format name. For example, the name of the TIFF man page is imtiff. To display it, enter man imtiff.

CONTACT

SDSC consultants, (619)534-5100, consult@y1.sdsc.edu