Subject: Unexpected rebin behavior
Posted by mchinand on Wed, 03 Sep 2003 16:20:10 GMT
View Forum Message <> Reply to Message

Performing rebin on 2-d (or any higher dimension) integer arrays sometimes give results I didn't expect. Here's a trivial example array:

```
IDL> help, b
B               INT       = Array[2, 2]

IDL> print, b
       8       9
      10      13
```

Minifying this to a 1x1 array with rebin gives:

```
IDL> print, rebin(b,1,1)
       9
```

But,

```
IDL> print, fix(total(b)/4)
      10
```

which is what I would have expected as the result from rebin. I think what rebin is doing is first performing a rebin in one direction and then performing it in the other direction on the new, intermediate rebinned array. So the above is equivalent to:

```
IDL> print, rebin(rebin(b,1,2),1,1)
       9
```

Switching the order of the rebinning yields a different result:

```
IDL> print, rebin(rebin(b,2,1),1,1)
      10
```

Or equivalently:

```
IDL> print, rebin(transpose(b),1,1)
      10
```

I was using rebin to reduce the size of an image and expected it to do neighborhood averaging for the new array. The integer truncation after rebinning along each direction can lead to different results than averaging the neighborhood in a single step. I guess this isn't really a bug (the manual only describes how rebin works in one dimension), but was

unexpected, at least for me. I simple work-around is to cast the array to float and then back to integer (or just keep as float) after the rebin:

```
IDL> print, fix(rebin(float(b),1,1))
      10
```

--Mike Chinander

--
Michael Chinander
m-chinander@uchicago.edu
Department of Radiology
University of Chicago
