Subject: Re: Array resize with arbitrary arithmetic
Posted by Dick Jackson on Wed, 14 Mar 2007 03:26:45 GMT
View Forum Message <> Reply to Message

Hi Radha,

"David Fanning" <news@dfanning.com> wrote in message
news:MPG.205f7ed4a7dbb8da989ed1@news.frii.com...
> David Fanning writes:
>
>>> image=read_bmp('9 March 2007\775u.bmp') ; 640x240 array
>>> temp = Rebin(image, 2, 320, 240)
>>> temp = Total(Temporary(temp), 1)
> [...]
> I *think* the answer to the original question, which
> looks to me like we want to multiply the first pixel
> by 256 and add the second, adjacent pixel to it (does
> that seem weird to you!?) is this:
>
> image=read_bmp('9 March 2007\775u.bmp') ; 640x240 array
> temp = Rebin(image, 2, 320, 240) ; Adjacent pixels in cols
> temp[0,*,*] = temp[0,*,*] * 256 ; Multiply 1st col by 256.
> temp = Total(temp,1) ; Add columns together.

That's not bad, but if I'm right, this arithmetic isn't so arbitrary! We're just turning each pair of bytes into an unsigned short integer, so this one line should do the whole thing:

```
comp_image = UInt(image, 0, 320, 240)
```

... except if you're on a "little-endian" machine, you'll need to swap the bytes. This statement will work to fix it if needed:

```
Swap_Endian_InPlace, comp_image, /Swap_If_Little_Endian
```

Now, I don't have your data, but this test should prove the point. I tested it on Intel (little-endian) and works fine, can someone double-check it on big-endian hardware, please?

```
IDL> image=bindgen(6,2)*21B
IDL> print,image
  0  21  42  63  84 105
126 147 168 189 210 231
IDL> comp_image = UInt(image, 0, 3, 2)
IDL> Swap_Endian_InPlace, comp_image, /Swap_If_Little_Endian
IDL> print,comp_image
     21   10815   21609
  32403   43197   53991
```

If you really need the result as Long, then *after* all this, do

```
comp_image = Long(comp_image)
```

Hope this helps!

--
Cheers,
-Dick

--
Dick Jackson Software Consulting http://www.d-jackson.com
Victoria, BC, Canada +1-250-220-6117 dick@d-jackson.com
