Subject: Re: distribution of colors for an image
Posted by JD Smith on Wed, 27 Oct 2004 17:27:17 GMT
View Forum Message <> Reply to Message

On Wed, 27 Oct 2004 15:57:00 +0200, Reimar Bauer wrote:

> David Fanning wrote:
>> Reimar Bauer writes:
>>
>>
>>> fine, I have seen a lot of instruction on your marvellous web page.
>>>
>>> But I don't understand the result I got. Lets show an example.
>>>
>>> a=dist(20)
>>> h=histogram(a)
>>> print,max(a),max(h)
>>>   14.1421   56
>>>
>>>
>>> u=uniq(a,sort(a))
>>> help,u,h
>>> U   LONG   = Array[61]
>>> H   LONG   = Array[15]
>>>
>>> Why could be h higher as a?
>>> Why doesn't I got a vector length of 61 as uniq tells?
>>
>>
>> You asked about color distribution in an image. A histogram
>> will tell you (with a byte scaled image, of course) how many
>> pixels in the image have a particular color. It will even
>> tell you which pixels those are, but that is another story,
>> best explained with JD's Histogram Tutorial.
>>
>> In your case H is fifteen elements long, because your data
>> had values between 0 and 15, and you used a bin size of 1,
>> by default. The *numbers* returned from histogram, told you
>> the pixel distribution of those 15 "colors". In one bin, for
>> example, you had 56 pixels values that fell into that bin.
>>
>> You had 61 unique numbers in your data, but all 61 of them fell
>> into one of the 15 bins you set up.
>>
>> To see your color distribution, you want to plot the histogram
>> of your data:
>>
>>   data = dist(200)

>>   Plot, Histogram(data), XStyle=1, $
>>   XTitle='Color Distribution', YTitle='Number of Pixels'
>>
>> Does that help?
>>
>
> Yes, this is very good explained.
>
> Now it is clear and I know why I was so irritated of the result I got.
>
>
> I have used a circular clipping of an image and have missed that's
> histogram uses always rectangular input. The background color which
> clips the data to invisible is count highest. If I don't use max_value I
> see nothing on the plot.

Another way to recover all 61 of your input values, illustrating how bin size has everything to do with the number of non-zero histogram elements:

```
IDL> a=dist(20)
IDL> h=histogram(a,BINSIZE=.05)
IDL> wh=where(h gt 0,cnt)
IDL> print,cnt
         61
```

But beware of using HISTOGRAM on floating point data in cases where you care whether a given value falls in one bin or another (see the "razor's edge" article http://www.dfanning.com/math_tips/razoredge.html).

JD