Subject: Re: Does IDL has histogram matching function?
Posted by Paul Sorenson on Thu, 12 Dec 2002 22:11:40 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> Oddly enough, I was just thinking about histogram matching
> because I was re-reading that section of the book in
> Digital Image Processing by Gonzales and Woods. (Have I
> mentioned what a great book this is!?) I think I finally
> understand how to do this.

There are some things in that Gonzalez and Woods book that really hurt my brain. On page 96, they say that G(z) = T(r). Get out of town! :-) How can this be? They don't *look* the same in the plots shown on page 98 (Figure 3.19). G(z) is the desired cumulative distribution and T(r) is the cumulative distribution of the input image. All of this appears in their discussion of Histogram Matching.

-Paul Sorenson

"Paul Sorenson" <aardvark62@msn.com> wrote in message
news:8270ac8d.0212091412.51094acd@posting.google.com...
> tianyf_cn@yahoo.com.cn (TIAN Yunfeng) wrote in message
news:<42e9d2cb.0211230034.560a064e@posting.google.com>...
>> Maybe I want to process float type images. Or the output data values
>> are in a narrow range. Does anyone have some ideas?
>>
>> Thanks.
>>
>> Yours,
>> Tian.
>>
> Tain,
>
> Are you wanting to specify your desired curve algebraically? As is
> done with QSIMP for example? If not, I think you will have to group
> your data into bins as is done with HIST_EQUAL, FCN. The algorithm
> that I posted Friday for the FCN keyword, or David's algorithm, might
> be a start. As they stand, these algorithms limit you to 256 bins and
> BYTSCLed results. But they probably could be translated to higher
> resolution by substituting your own algebra where they call BYTSCL, or
> simply scaling your result to fit OMIN and OMAX.
>
> -Paul Sorenson
>> David Fanning <david@dfanning.com> wrote in message
news:<MPG.1835a3e2693e7288989a0b@news.frii.com>...
>>> David Fanning (david@dfanning.com) writes:

```
>>>
>>>>  I expect it might take a day or so to write the code.
>>>>  Do you have any money? :-)
>>>
>>>  Ah, forget the money. This turned out to be too easy. :-)
>>>
>>>  Here is a routine, named HISTOMATCH, that takes an image
>>>  and a histogram that you would like to perform histogram
>>>  matching to.
>>>
>>>  ;*******************************************************
>>>  FUNCTION HistoMatch, image, histogram_to_match
>>>
>>>  ; Perform histogram matching according to the method of
>>>  ; Gonzales and Woods in Digital Image Processing, pp 94-102
>>>
>>>  ; image - The input image.
>>>  ; histogram_to_match - The histogram used for histogram matching.
>>>
>>>     ; Calculate the histogram of the input image.
>>>
>>>  h = Histogram(Byte(image), Binsize=1, Min=0, Max=255)
>>>  totalPixels = Float(N_Elements(image))
>>>
>>>     ; Find a mapping from the input pixels to s.
>>>
>>>  s = FltArr(256)
>>>  FOR k=0,255 DO BEGIN
>>>    s[k] = Total(h(0:k) / totalPixels)
>>>  ENDFOR
>>>
>>>     ; Find a mapping from input histogram to v.
>>>
>>>  v = FltArr(256)
>>>  FOR q=0,255 DO BEGIN
>>>    v[q] = Total(histogram_to_match(0:q) / totalPixels)
>>>  ENDFOR
>>>
>>>     ; Find z from v and s.
>>>
>>>  z = BytArr(256)
>>>  FOR j=0,255 DO BEGIN
>>>     I = Where(v LT s[j], count)
>>>     IF count GT 0 THEN z[j] = (Reverse(I))[0] ELSE z[j]=0
>>>  ENDFOR
>>>
>>>     ; Create the matched image.
>>>
```

```
>>>  matchedImage = z[Byte(image)]
>>>  RETURN, matchedImage
>>>  END
>>>  ;*****************************************************
>>>
>>>  I'm certain JD or someone will point out to me how to
>>>  use another Histogram to eliminate the Where function,
>>>  but, hey, this is for free. I'm trying to make a living
>>>  here. :-(
>>>
>>>  Does it work!? I think so. I'm not sure.
>>>
>>>  Try this. Let's see if we can match am image to the
>>>  histogram formed by calculating the histogram of
>>>  the histogram equalized image. (The result should
>>>  be the same as the histogram equalized image, more
>>>  or less.)
>>>
>>>  ;*****************************************************
>>>  PRO TestIt
>>>  filename = Filepath('ctscan.dat', Subdir=['examples', 'data'])
>>>  OpenR, lun, filename, /Get_Lun
>>>  image = BytArr(256, 256)
>>>  ReadU, lun, image
>>>  Free_Lun, lun
>>>
>>>  Window, XSize=3*256, YSize=256
>>>  TV, image, 0
>>>  TV, Hist_Equal(image), 1
>>>  TV, HistoMatch(image, Histogram(Hist_Equal(image), Min=0, Max=255)), 2
>>>  END
>>>  ;*****************************************************
>>>
>>>    IDL> TestIt
>>>
>>>  Wow! And this was on the *first* try. *That* doesn't happen too
>>>  often. :-)
>>>
>>>  Try this:
>>>
>>>    a = LonGen(255)
>>>    b = a#b
>>>    b = BytScl(b)
>>>    Window, 1
>>>    Plot, Histogram(b, Min=0, Max=255)
>>>    Window, 2, XSize=256, YSize=256)
>>>    TV, HistoMatch(image, Histogram(b, Min=0, Max=255))
>>>
```

```
>>>  Still looks good, I think.
>>>
>>>  OK, I'm waiting for feedback. :-)
>>>
>>>  Cheers,
>>>
>>>  David
```