Subject: Re: Conversion floating point to byte or integer
Posted by Jean H. on Wed, 10 Oct 2007 23:44:58 GMT
View Forum Message <> Reply to Message

go cats wrote:
> Hello Gurus,
>
> I've been trying to write a code to correct imagery data. Original
> imagery data format is byte format.
> Steps for the processing includes reading the original data,
> subtracting dark current, and multiplying calibration coefficients and
> saving the results. The two values; dark current and calibration
> coefficients are given by arrays with floationg point format.
> The code seemes to work without any problem. But in the resulting
> image saved in binary format , numbers higher than 255 store only
> remants of what is subtracted from 256. The reason I want to convert
> to byte is to save some disk space. In the image saved as floating
> point format pixel values look ok.
> Could you give some advice what part of program I have to look at? and
> what causes this problem?
>
> Thank you in advance,
> Kim

So what do you want to do with values greater than 255?

You could do
1) compute your new image as a float,
2) scale everything down so ALL of your values are between 0 and 255 and
save the image
or 2) brightPixels = where(image gt 255)
image[brightPixels] = 255 and save the image

Jean

Subject: Re: Conversion floating point to byte or integer
Posted by beardown911 on Thu, 11 Oct 2007 04:20:23 GMT
View Forum Message <> Reply to Message

On Oct 10, 6:44 pm, "Jean H." <jghas...@DELTHIS.ucalgary.ANDTHIS.ca>
wrote:
> go cats wrote:
>> Hello Gurus,
>
>> I've been trying to write a code to correct imagery data. Original
>> imagery data format is byte format.
>> Steps for the processing includes reading the original data,

>> subtracting dark current, and multiplying calibration coefficients and
>> saving the results. The two values; dark current and calibration
>> coefficients are given by arrays with floationg point format.
>> The code seemes to work without any problem. But in the resulting
>> image saved in binary format , numbers higher than 255 store only
>> remants of what is subtracted from 256. The reason I want to convert
>> to byte is to save some disk space. In the image saved as floating
>> point format pixel values look ok.
>> Could you give some advice what part of program I have to look at? and
>> what causes this problem?
>
>> Thank you in advance,
>> Kim
>
> So what do you want to do with values greater than 255?
>
> You could do
> 1) compute your new image as a float,
> 2) scale everything down so ALL of your values are between 0 and 255 and
> save the image
> or 2) brightPixels = where(image gt 255)
> image[brightPixels] = 255 and save the image
>
> Jean- Hide quoted text -
>
> - Show quoted text -

Jean,

Thank you for your prompt reply and 2) is what I wanted.
Could I ask why this happen? Am I asking too basic question about
programming?
I casted output format to bytarr(cols, rows) to scale everything down
0-255, but result was same.
I've read variables part in Liam's book, and tried the least
significant 8 bit extraction.
That doesn't seem to be the answer.

Well, thank you again.
Kim

Subject: Re: Conversion floating point to byte or integer
Posted by Jean H. on Thu, 11 Oct 2007 06:08:12 GMT
View Forum Message <> Reply to Message

>> So what do you want to do with values greater than 255?
>>

>> You could do
>> 1) compute your new image as a float,
>> 2) scale everything down so ALL of your values are between 0 and 255 and
>> save the image
>> or 2) brightPixels = where(image gt 255)
>> image[brightPixels] = 255 and save the image
>
> Jean,
>
> Thank you for your prompt reply and 2) is what I wanted.
> Could I ask why this happen? Am I asking too basic question about
> programming?

So you need to scale it, not to typecast it.
In IDL, if you typecast to byte, it does "wrap around" values, so that
255 is followed by 0...
IDL> print, byte(256)
0
IDL> print, byte(256+257)
1

if you scale it using BytScl(), it takes the smallest value and assign
it to 0, the max value to 255, and the result is a byte array
IDL> print, bytscl([250,251])
0 255

Jean

> I casted output format to bytarr(cols, rows) to scale everything down
> 0-255, but result was same.
> I've read variables part in Liam's book, and tried the least
> significant 8 bit extraction.
> That doesn't seem to be the answer.
>
> Well, thank you again.
> Kim
>

Subject: Re: Conversion floating point to byte or integer
Posted by Maarten[1] on Thu, 11 Oct 2007 08:05:06 GMT
View Forum Message <> Reply to Message

On Oct 10, 11:32 pm, go cats <beardown...@gmail.com> wrote:
> Hello Gurus,
>
> I've been trying to write a code to correct imagery data. Original
> imagery data format is byte format.

> Steps for the processing includes reading the original data,
> subtracting dark current, and multiplying calibration coefficients and
> saving the results. The two values; dark current and calibration
> coefficients are given by arrays with floationg point format.
> The code seemes to work without any problem. But in the resulting
> image saved in binary format , numbers higher than 255 store only
> remants of what is subtracted from 256. The reason I want to convert
> to byte is to save some disk space. In the image saved as floating
> point format pixel values look ok.
> Could you give some advice what part of program I have to look at? and
> what causes this problem?

Why bother with dark-current correction and calibration, if you are
destructing the data afterwards by scaling to byte again? I'd suggest
to save as floating point in a format that supports internal
compression (hdf4/hdf5), and play with the compression settings. You'l
still end up with the minimal data set on disk, but you won't
introduce artefacts.

Maarten

Subject: Re: Conversion floating point to byte or integer
Posted by beardown911 on Thu, 11 Oct 2007 13:20:01 GMT
View Forum Message <> Reply to Message

On Oct 11, 1:08 am, "Jean H." <jghas...@DELTHIS.ucalgary.ANDTHIS.ca>
wrote:
>>> So what do you want to do with values greater than 255?
>
>>> You could do
>>> 1) compute your new image as a float,
>>> 2) scale everything down so ALL of your values are between 0 and 255 and
>>> save the image
>>> or 2) brightPixels = where(image gt 255)
>>> image[brightPixels] = 255 and save the image
>
>> Jean,
>
>> Thank you for your prompt reply and 2) is what I wanted.
>> Could I ask why this happen? Am I asking too basic question about
>> programming?
>
> So you need to scale it, not to typecast it.
> In IDL, if you typecast to byte, it does "wrap around" values, so that
> 255 is followed by 0...
> IDL> print, byte(256)
> 0

> IDL> print, byte(256+257)
> 1
>
> if you scale it using BytScl(), it takes the smallest value and assign
> it to 0, the max value to 255, and the result is a byte array
> IDL> print, bytscl([250,251])
> 0 255
>
> Jean
>
>
>
>> I casted output format to bytarr(cols, rows) to scale everything down
>> 0-255, but result was same.
>> I've read variables part in Liam's book, and tried the least
>> significant 8 bit extraction.
>> That doesn't seem to be the answer.
>
>> Well, thank you again.
>> Kim- Hide quoted text -
>
> - Show quoted text -

Jean,

Thanks a lot.

Kim

Subject: Re: Conversion floating point to byte or integer
Posted by beardown911 on Thu, 11 Oct 2007 13:22:07 GMT
View Forum Message <> Reply to Message

On Oct 11, 3:05 am, Maarten <maarten.sn...@knmi.nl> wrote:
> On Oct 10, 11:32 pm, go cats <beardown...@gmail.com> wrote:
>
>> Hello Gurus,
>
>> I've been trying to write a code to correct imagery data. Original
>> imagery data format is byte format.
>> Steps for the processing includes reading the original data,
>> subtracting dark current, and multiplying calibration coefficients and
>> saving the results. The two values; dark current and calibration
>> coefficients are given by arrays with floationg point format.
>> The code seemes to work without any problem. But in the resulting
>> image saved in binary format , numbers higher than 255 store only
>> remants of what is subtracted from 256. The reason I want to convert

>> to byte is to save some disk space. In the image saved as floating
>> point format pixel values look ok.
>> Could you give some advice what part of program I have to look at? and
>> what causes this problem?
>
> Why bother with dark-current correction and calibration, if you are
> destructing the data afterwards by scaling to byte again? I'd suggest
> to save as floating point in a format that supports internal
> compression (hdf4/hdf5), and play with the compression settings. You'l
> still end up with the minimal data set on disk, but you won't
> introduce artefacts.
>
> Maarten

Maarten,

Thank you for the idea. I will try it.

Kim

Subject: Re: Conversion floating point to byte or integer
Posted by M. Katz on Thu, 11 Oct 2007 21:04:11 GMT
View Forum Message <> Reply to Message

You have to guard against byte values going below zero.
Here is what I would recommend.
1) Immediately convert the raw data into float, before any processing.
2) Subtract the background array but "crop" any negative values at
zero.
  image1 = (raw_data - background) > 0.
3) Then apply your calibration
  image2 = image1 * calibration
  If instead you have to divide by the calibration array, watch out
for zeros.
  image2 = image1 / (calibration > 1.) might be a format you should
consider.
  (The 1. here is just an example. It depends on your actual data.)

Of course be careful about spikes or zeros in the calibration
file or background file. Those can cause problems.

MKatz

Subject: Re: Conversion floating point to byte or integer
Posted by beardown911 on Fri, 12 Oct 2007 01:44:27 GMT

On Oct 11, 4:04 pm, "M. Katz" <MKatz...@yahoo.com> wrote:
> You have to guard against byte values going below zero.
> Here is what I would recommend.
> 1) Immediately convert the raw data into float, before any processing.
> 2) Subtract the background array but "crop" any negative values at
> zero.
> image1 = (raw_data - background) > 0.
> 3) Then apply your calibration
> image2 = image1 * calibration
> If instead you have to divide by the calibration array, watch out
> for zeros.
> image2 = image1 / (calibration > 1.) might be a format you should
> consider.
> (The 1. here is just an example. It depends on your actual data.)
>
> Of course be careful about spikes or zeros in the calibration
> file or background file. Those can cause problems.
>
> MKatz

Thanks a lot.

Kim