
Subject: Re: joining images into a colored one

Posted by [Guilherme Gualda](#) on Mon, 19 Jun 2017 03:50:30 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Friday, June 16, 2017 at 7:18:08 AM UTC-3, Helder wrote:

> On Friday, June 16, 2017 at 11:59:15 AM UTC+2, Nikola Vitas wrote:

>> On Thursday, June 15, 2017 at 12:18:49 PM UTC+1, Helder wrote:

>>> Hi,

>>> I couldn't find anything on this in previous posts, but I think (hope) that there's an easy solution to it.

>>> I want to create a colored image out of a subset of images that are "labelled regions", that is:

>>> image[0] has patches with values 1l, zero otherwise.

>>> image[1] has patches with values 2l, zero otherwise.

>>> ...and so on

>>>

>>> Typically I'm dealing with n images, where $n < 10$. Fortunately, all images have the same dimensions.

>>> I want to join the images so that the color of each [i,j] pixel is dependent on which and how many images (from the subset) had this value different from zero.

>>> Example:

>>> [i0,j0] is only different from zero in image[0], so it will be, e.g., red.

>>> [i1,j1] is only different from zero in image[1], so it will be, e.g., blue.

>>> [i1,j1] is different from zero only in image[0] and image[1], so it will be, red+blue=magenta.

>>>

>>> I know how to handle the pixels, any clue how to programmatically handle the colors?

>>>

>>> [I was thinking of using something like number conversion from basis-to-basis. That is if I have n images, I will assign to each pixel the following value:

>>> $[i,j] = \text{image_0}[i,j] * n^0 + \text{image_1}[i,j] * n^1 + \text{image_2}[i,j] * n^2 + \dots$ and then use a long number for indexing colors (24-bit)... but I think/hope that there is a nicer/cleaner way of doing this]

>>>

>>> Thanks,

>>> Helder

>>

>> If I understand your problem, the first part is how to combine your images into one where the value of each pixel shows how many of the images have non-zero value at that location.

Something like

>>

>> Images:

>>

>> 1 0 0 0 2 0 0 0 3 1 1 1

>> 0 1 0 0 2 0 0 3 0 - combined -> 0 3 0

>> 0 1 0 0 0 2 3 0 0 1 1 1

>>

>> If that's what you want to do then the quickest way is to sum up the binary maps (assuming that images stored in one 3D variable `images = FLOAT(nx, ny, nimages)`)

>>

>> `combined = INTARR(nx, ny)`

```

>> FOR i = 0, nimages-1 DO combined += images[*, *, i] EQ i
>>
>> (or i+1 or whatever else you have as indicator).
>>
>> Once you create the combined image, you want to display it with only nimages colors. For that
you need to create a new color scale. Every color scale has 3 arrays (representing R, G and B
channels) with 256 values each. For you only the first nimages values of each array will matter.
The details will depend on which routine you use for displaying the data.
>>
>> Regarding the choice of colors, check:
>> http://colorbrewer2.org/
>> http://www.idlcoyote.com/ng\_tips/brewer.html
>
> Hi,
> sorry for not being able to explain better...
> Your suggested solution will not distinguish, for example, the following case:
>
> image Nr -->   1   2   3   4
> image-----> 001  020  030  004
> combine-----> 022
>
> I want to be able to distinguish between "1+4" and "2+3"
>
> I'm using Superchromix's solution for the moment. After adapting and fighting a bit, I got that to
work.
>
> Thanks anyway!
> Helder

```

Hi Helder,

If you need to distinguish between 1+4 and 2+3, you could use the sum of the powers of 2. In that case:

```

1+4 --> 2^1 + 2^4 = 18
2+3 --> 2^2 + 2^3 = 12

```

The only problem is that you end up with lots of gaps in your sequence of numbers (values are bound by $2^{(n+1)!}$), and you will probably want to renumber the summed image to remove the gaps.

You can remove the gaps using HISTOGRAM. You can do it the elegant way using reverse indices (see http://www.idlcoyote.com/tips/histogram_tutorial.html), or you can use a for loop (oh no!!) through the histogram values and replace the values in your summed image.

For the coloring, why don't you use a color table to display your summer image, rather than try to come up with RGB colors directly? If you only have a small number of values (say, order 20?), you may want spread the values by some common factor before applying the color table. If you

need the color image in the end, you can always use TVRD to retrieve it.

Maybe this helps? I can be more specific if needed. But you posed the problem conceptually, so I am guessing you can handle the actual coding...

Best,

Guil
