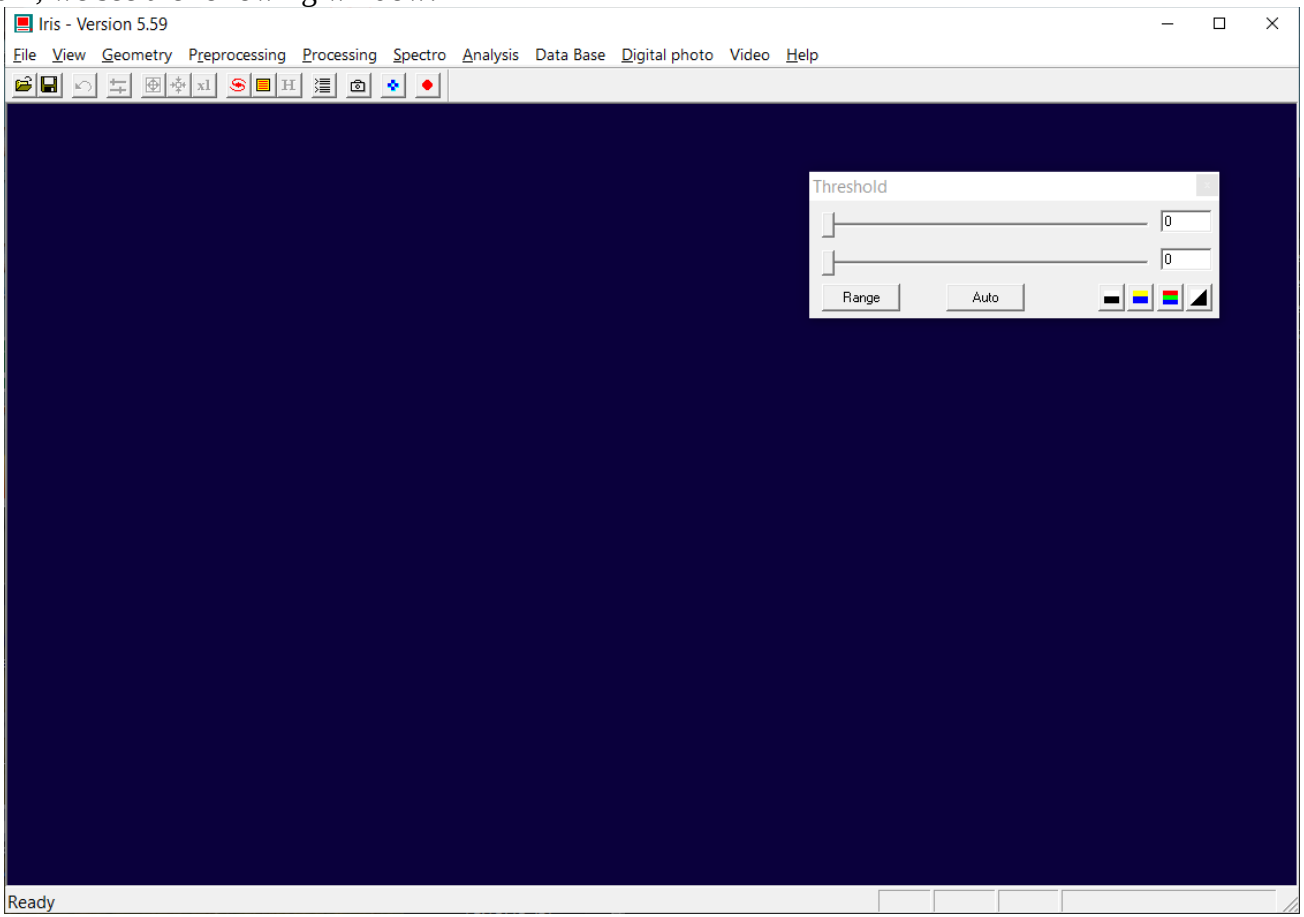
LRGB processing in Iris..

I decided to write this document after many times, after processing other people's astrophotos, I received the questions “And what software did you use?” And “How did you do it there?”. First of all, I want to say that perhaps this guide will still be supplemented, changed and systematized, since its appearance so far is very spontaneous and right now it’s rather a culinary recipe than the basics.

By default, I assume that the reader knows how to use his gear and how to make usable shots, what is the calibration process, what is the subtraction of the bias current, dark current and division by a flat field. And what’s the purpose of doing all these things.

So — here is a small guide to using Iris from me, known at the astronomy.ru as 4D, and as Anton Chechkin offline.

After starting the program, we see the following window:

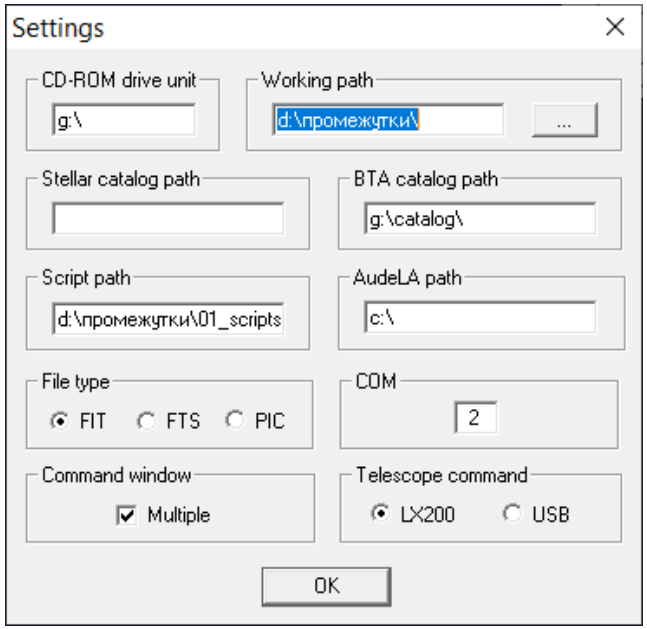


The first thing we need to do is set the working folder where Iris will save its files and the type of these same files.

Let's go to the menu

File → Settings…

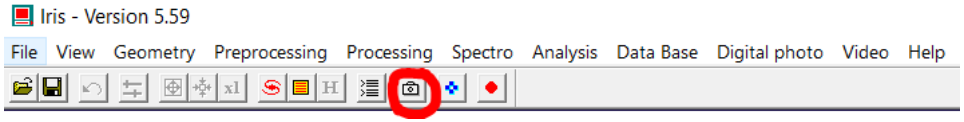
For me it looks like this:



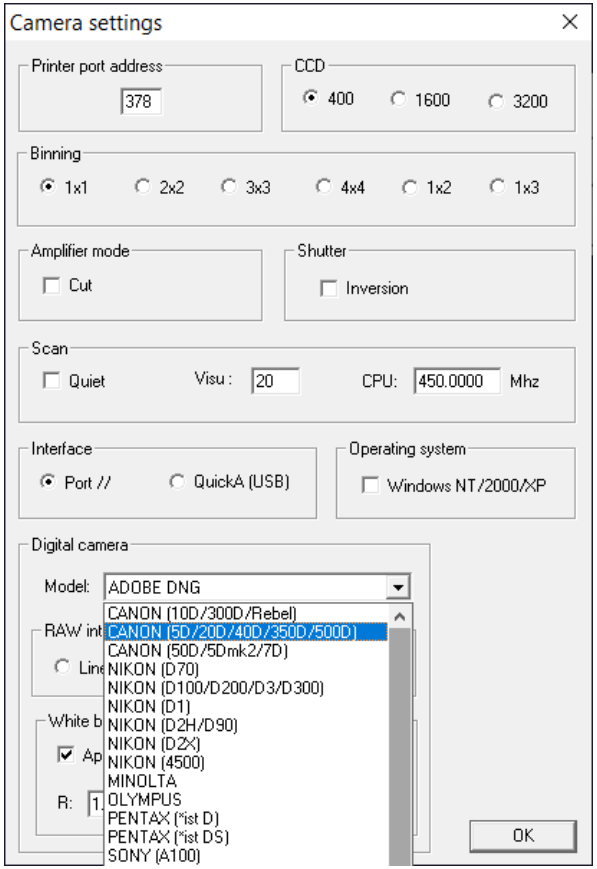
Up to recent (2008-2009 years 😆) times Iris didn't support tri-color fits images. Despite the fact that the "fit" —is an astronomy golden standard. However, this annoying oversight has been corrected for quite a long time, so we put "fit", and enjoy. The highlighted field shows the working folder. Also, since Iris strongly encourages us to work from the command line, it allows you to execute scripts: files written by hand, as a sequence of commands, and also executed sequentially. As you might guess, the field called "Script path" is responsible for this. If you are a little familiar with programming and you need to process, for example, a series of frames in some non-standard mode, scripts here can help you a lot. It doesn't matter HOW you create them — they are just a sequence of commands written into a text file with a ".pgm" extension.

But more about that some other time, if I have an excess of free time, and this part is at least useful to someone. 🧐

First things first, we need to learn how to upload a file. If we shot on a SLR camera in RAW format (we are professional astrophotographers, we shoot everything important only in RAW aren't we?!), then we need to click on the camera icon:



And in the menu that opens, you need to select your (or at least the most similar) camera model.

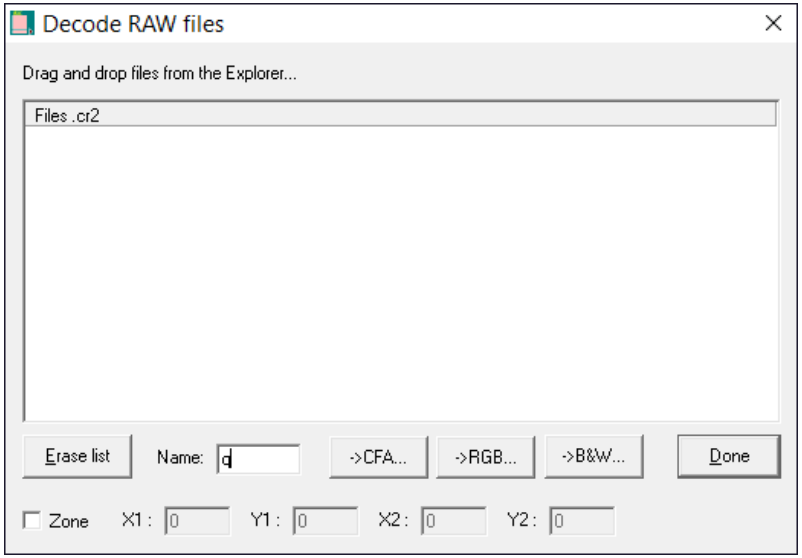


All other parameters such as binning, RGB balance values and other things in this window only affect somehow if you control the camera directly from the Iris, which I personally have never done. Therefore, it is unlikely that you need to change anything here.

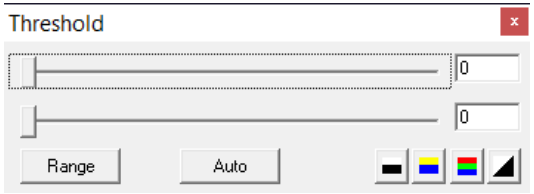
If we continue to discuss the very beginning of work, namely, loading RAW files into Iris, then we exit this window and go to the menu:

Digital Photo → Decode RAW files…

Which causes the program window to be minimized and opens the download window (which may be under other, already open Windows windows, yes, there exists such a bug):

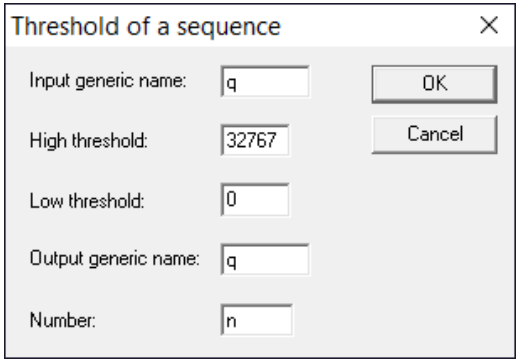


Here we can drag and drop the files that we want to convert using the Drag and Drop method, set a prefix for them in the “Name” field, click on -> CFA (because calibration should be done BEFORE debayerization) and get a series of images like: q1, q2, ... qn with the extension “.fit”, where n is the number of frames we loaded. After that, click “Done”, and return to the program, which itself will set the levels in the Threshold window according to the values that are present in the file:



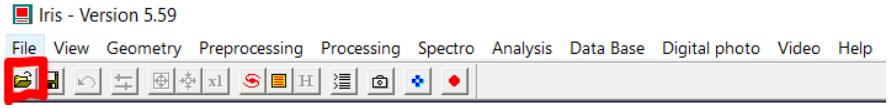
Sidenote: If you previously touched these sliders yourself, then all subsequent downloaded files will open at the position of the sliders at which they were loaded, and not at the values set automatically. This can of course be easily changed later. For example, from the menu:

View → Threshold of a sequence



The name of the output series does not have to be the same as the input. It depends on whether you need files with originally set levels. Yes, I hope that you already know without me that the values in the Threshold box do not affect the source file in any way, but only its display on the screen. Everything that is below the lower level is shown in black, above the top — white, and what is in the interval is accordingly considered according to the rule of proportions: the closer to the maximum value, the brighter. This is done due to the fact that the range of pixel brightness from the matrix is usually too large, and the correct setting of the white balance, photometry and other astronomical operations work only on data whose brightness has not been changed in a non-linear manner.

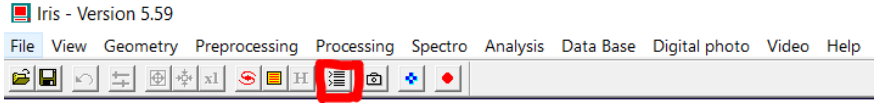
However, I digress. After we loaded the images, they were saved somewhere on the disk. And the program displays only the last one of the series. How to load for example the very first one? There are at least three ways:



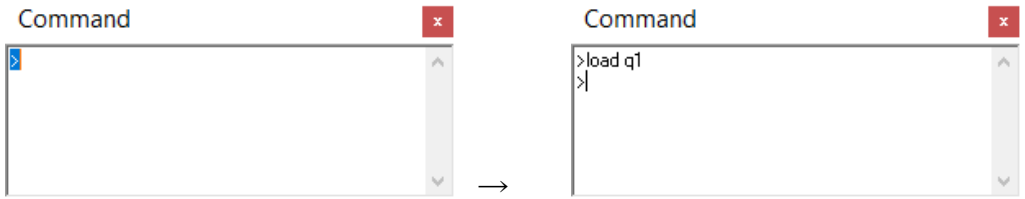
From the menu:

File → Load

And the third way, in my opinion the most natural for Iris, and the most unusual for me, as a windows user. Although for so many years I have already become quite accustomed to the Iris interface. Open that damned command line already!

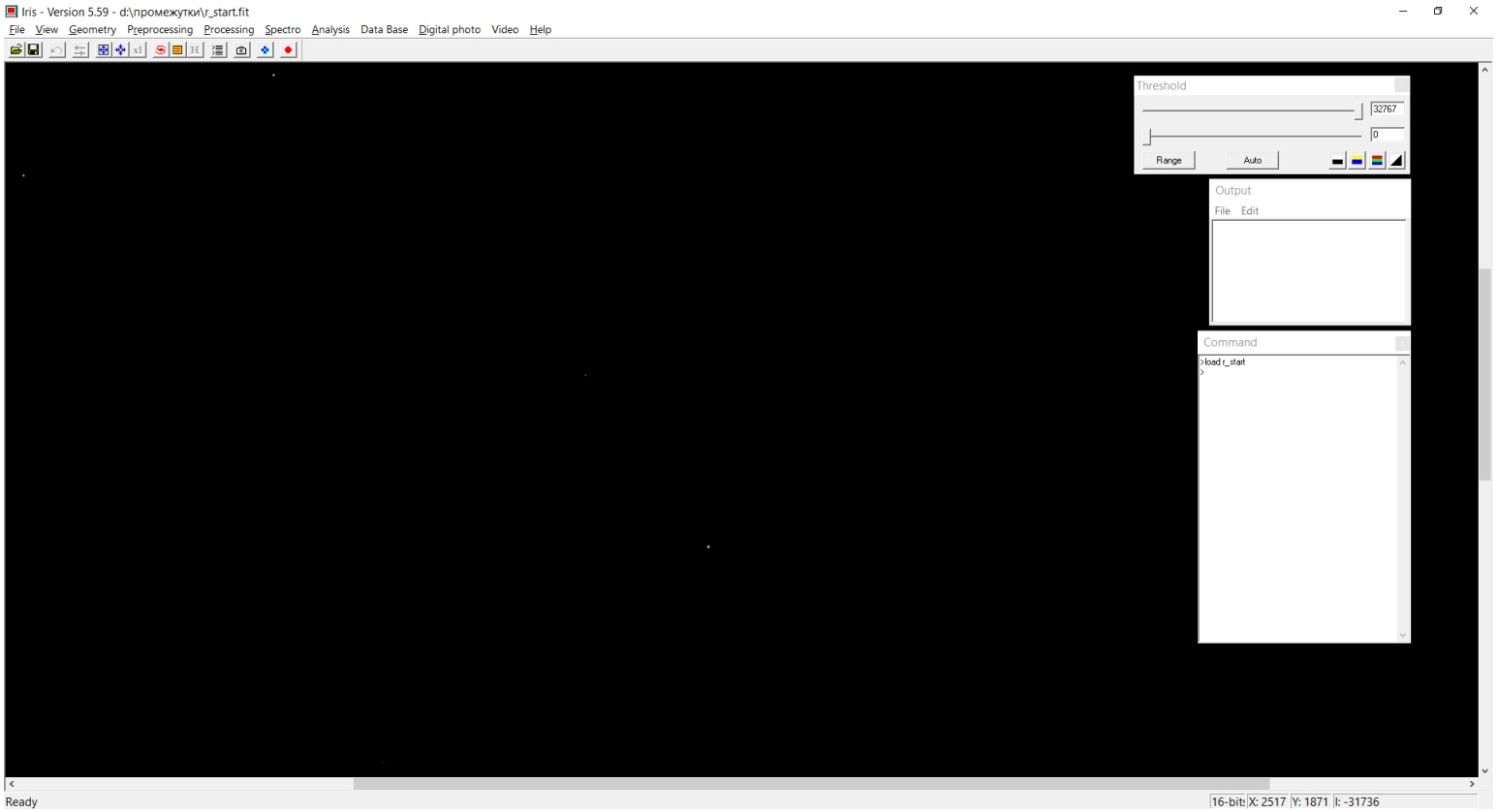


And to see a window shown below, which I will further call the terminal by analogy with operating systems:



Where on the right I entered the first command from the keyboard. I think that no explanation here is required. 😊

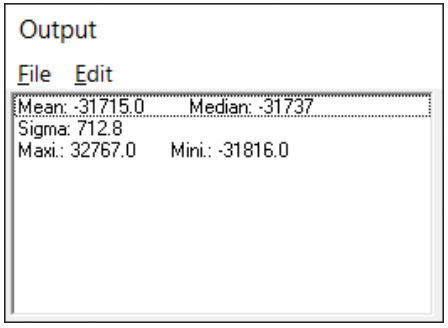
Omitting for now the calibration and alignment processes, I want to focus on the assembly of LRGB images, where the frame obtained without color filters is used as the brightness channel. Suppose that we are given 4 preprocessed shots made by monochrome camera, which have already been calibrated in advance and aligned with respect to each other. Let's call them R_start, G_start, B_start, L_start and put them in the program folder that we specified earlier. Let's load, for example, the frame corresponding to the red channel and see the "black square":



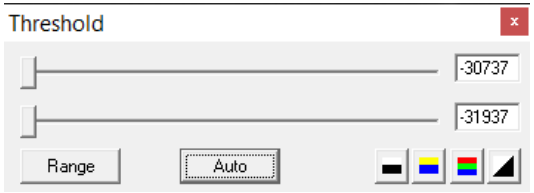
If we write the command in the terminal

```
>stat
```

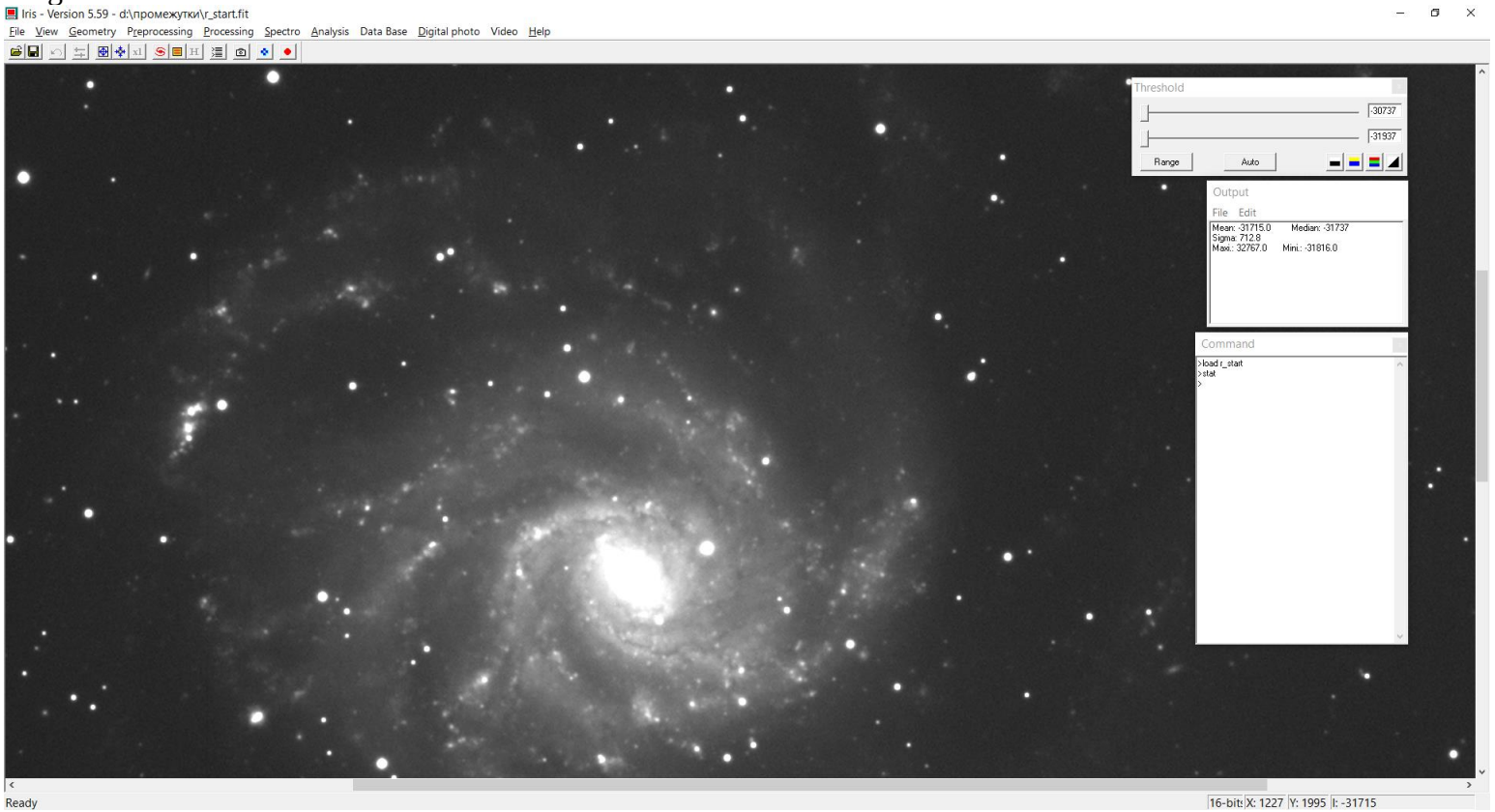
Then we get a window with the stats:



You can try bring the picture to a watchable view by clicking on the "auto" button in the Threshold box or by typing values manually:



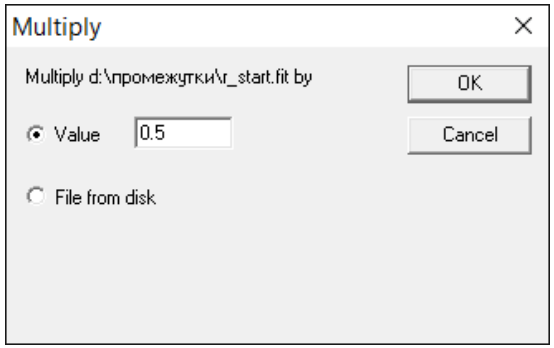
And get something like this:

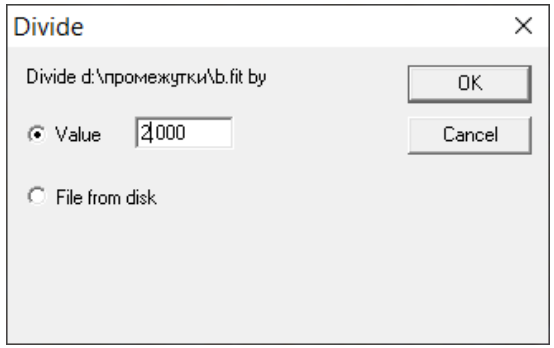


You can see that the pixels have values practically from -32768 to +32767, and most of them lie in the negative area. Although Iris is able to work with negative values, non-linear brightness conversions on them give incorrect results. Therefore, for further work, we should translate everything into a positive semiaxis. There is, however, a small caveat, the working range of Iris is strictly limited, firstly, to whole numbers, so no “54.89’ and other fractional values. And secondly, its range is also limited by the very -32768 - +32767. In order to fit all the pixel values, you obviously need to divide them in half, which will give us the values -16384 - + 16384. And then add the value +16384 to all pixels. So, we get values from 0 to 32767.

You can multiply values by 0.5 again in as many as three ways. The first two use the GUI, and the last one uses the terminal:

Processing → Multiply… ИЛИ Processing → Divide…

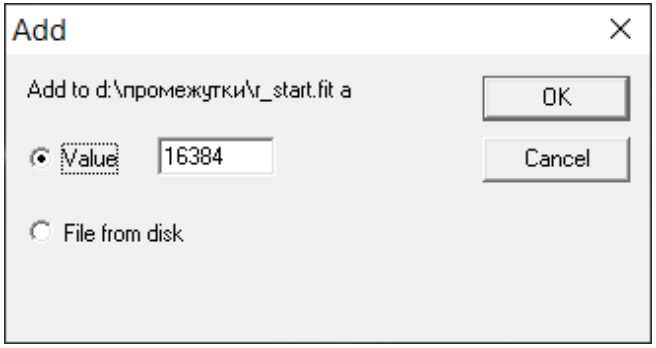


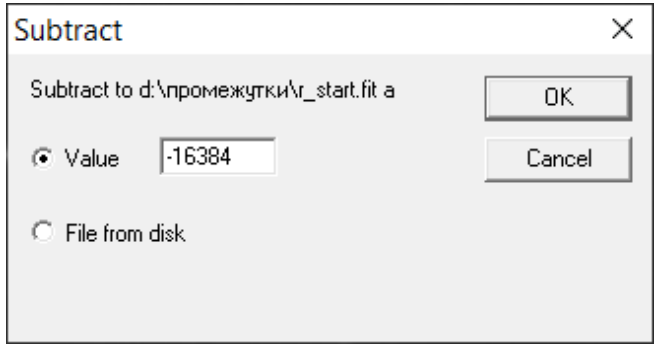


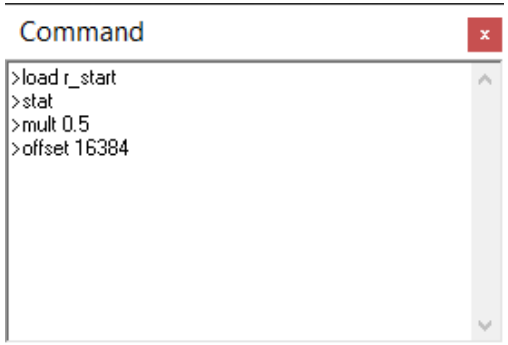
I hope no explanation is needed.

Next, we need, as we agreed, to add a constant to all values. Again, three ways:

Processing → Add⋯ or Processing → Subtract⋯





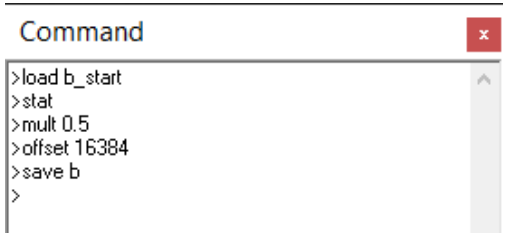


Then we save the result by entering the command in the terminal:

>save r

Although we are free to choose any name, if only without special characters and spaces.

Further, since we have the same type of files, we can also process them. At the same time, it is not necessary to enter a new command in the terminal each time, you can also change existing ones, put the cursor on the desired line and press Enter. This is how my terminal looked like after processing the green and blue channels:



After we have prepared the BW images, it's time to make color-version from them and set the correct white balance.

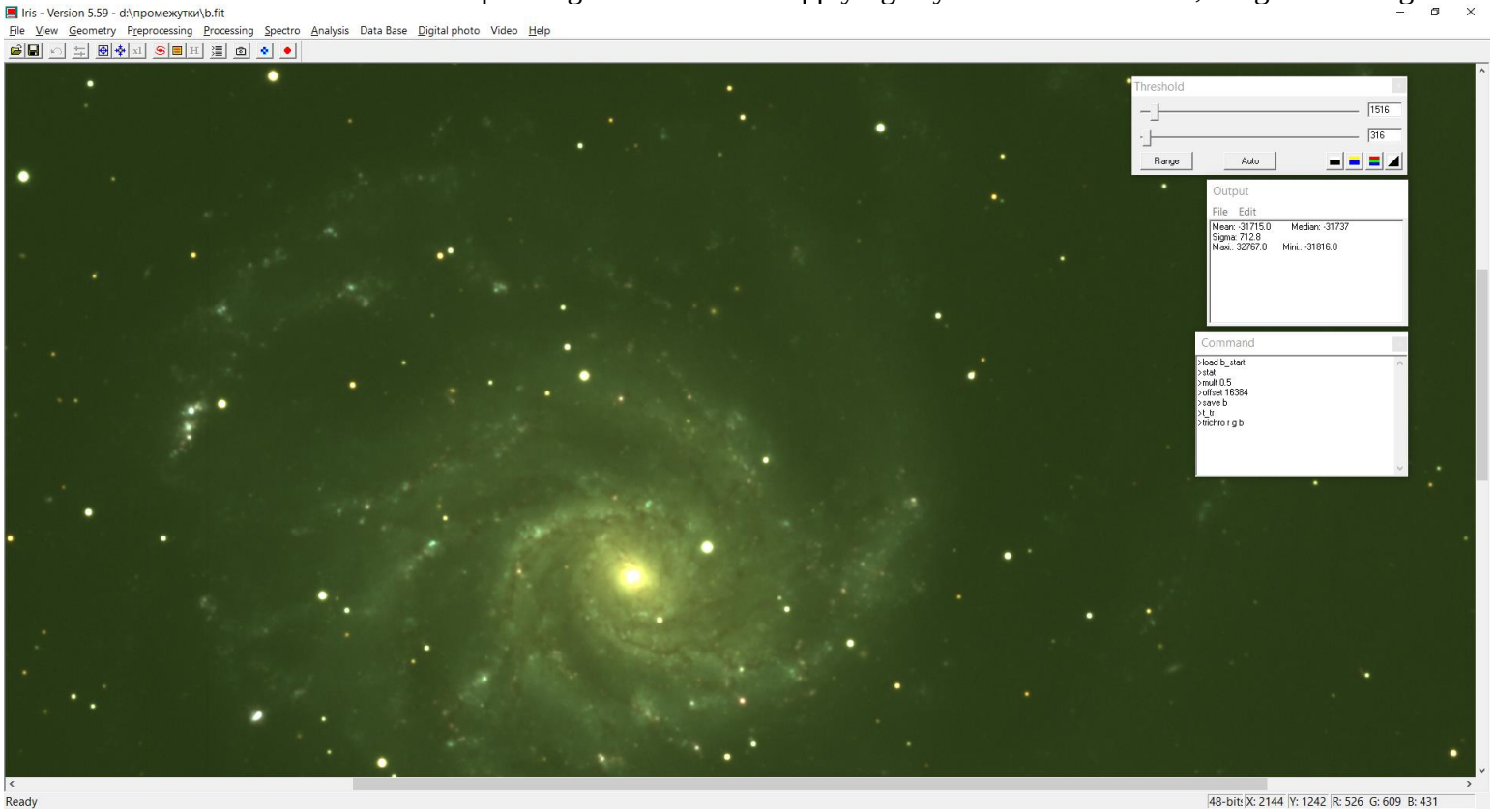
There are two commands for assembling a color image from channels. If the RGB channel names are r, g and b respectively, then you can write a short command:

>t_tr

If not then you can use the alternative:

>trichro [red] [green] [blue]

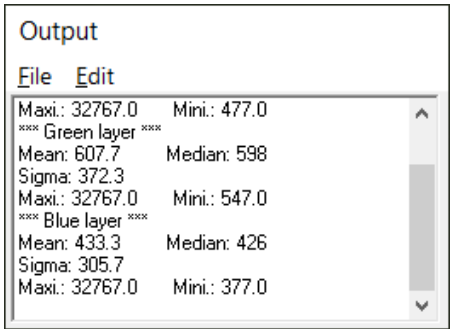
Where in square brackets are the names of the corresponding channels. After applying any of these commands, we get this image:



Where in the levels window the Auto button was previously pressed. Already it’s more interesting, but the white balance is obviously wrong. Using a familiar command:

>stat

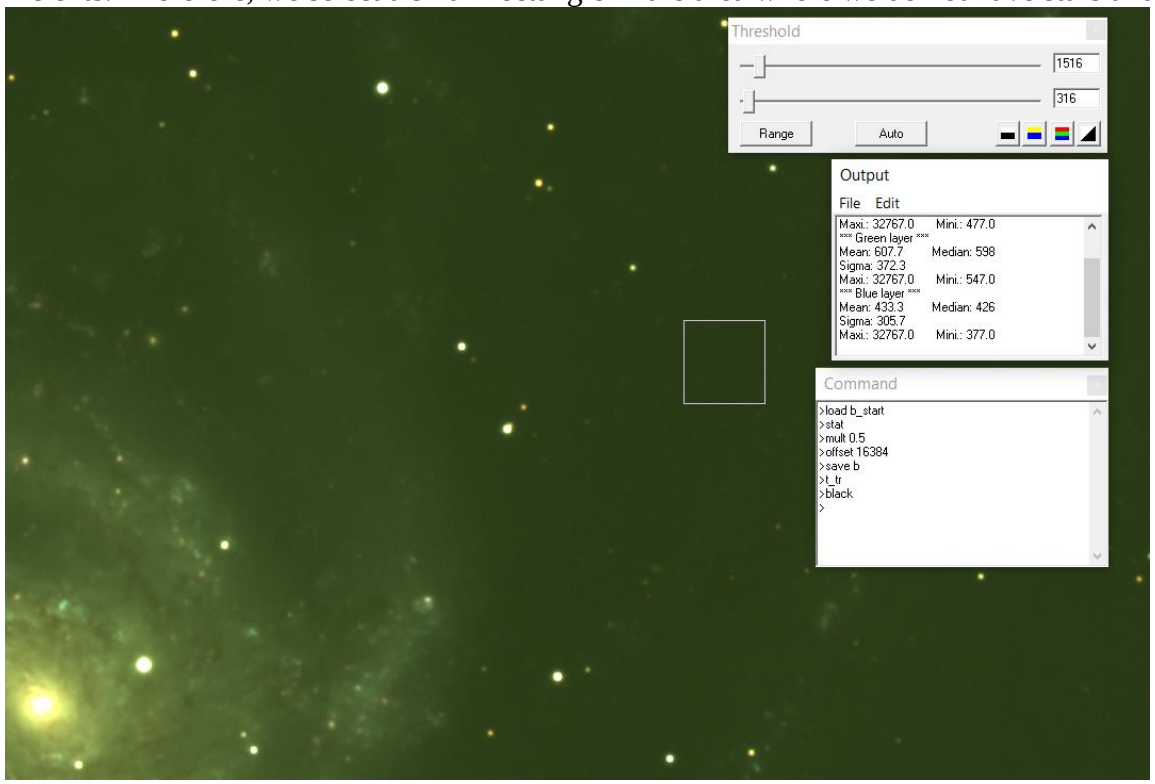
We can see:



The white balance adjustment algorithm is as follows:

1) We must bring the background to a neutral gray, or better — black..
2) After neutralizing the background, it is necessary to stretch (multiply) each channel by the appropriate factor so that the picture acquires decent colors.

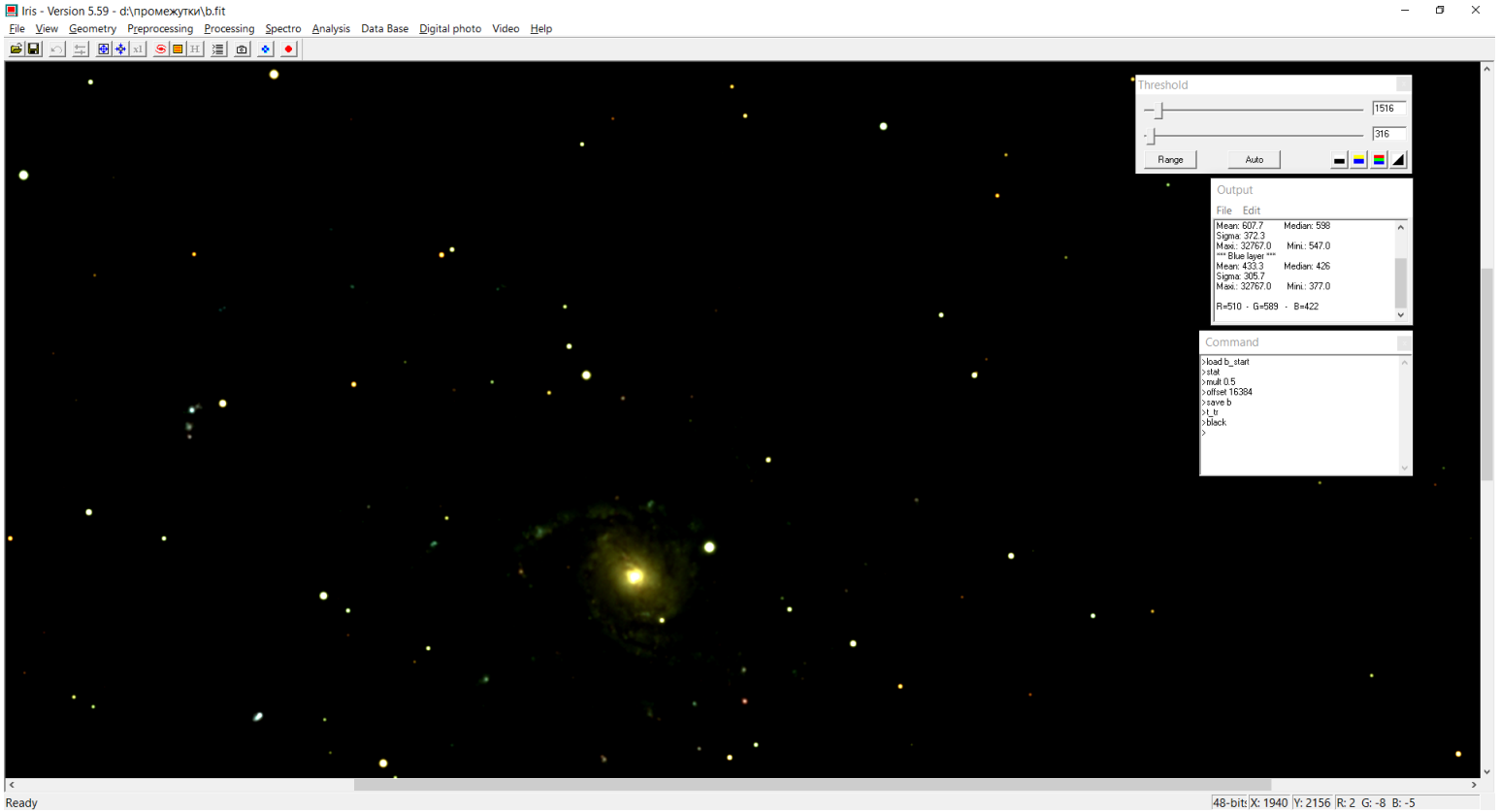
At the same time, it is clear that if the background level is not initially close to zero, then multiplying it by a coefficient will shift it, which complicates the process of selecting the correct coefficients. Therefore, we select a small rectangle in the area where we do not have stars and a useful signal:



And in terminal apply the following command:

```
>black
```

Obtainig:



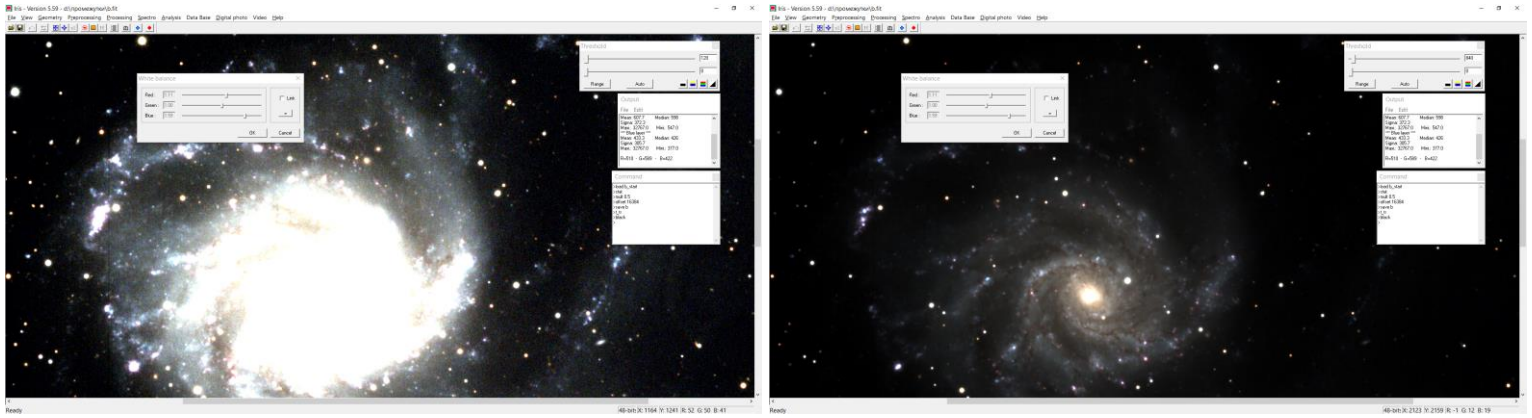
There is a small observation here. Personally, I manage to find the most correct balance by stretching the levels to the extreme. That is, we leave the black slider where it should be — at zero, and raise the upper levels until the structure of the galaxy appears. After that we go to the menu:

View → White balance adjustment

And move the sliders till the desired result. Of course we can ser these values from terminal:

>rgbbalance [coeff1] [coeff2] [coeff3]

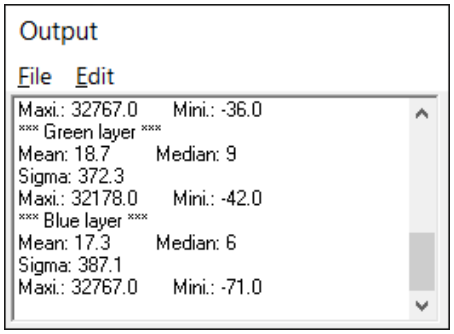
But this is only suitable if we know the coefficients in advance, because one could go nuts through such trial and error, especially considering that there is no cancel button in the Iris, and therefore it is better to write all intermediate calculations to a new file.



Having picked up the appropriate coefficients and played with the levels to check that the rest of the frame also looks natural, you can press OK. After that, we again apply our useful command:

>stat

And we see in the window the minimum values in the frame:



They are negative. Not good. Apply for example:

>offset 73

It should also be noted what our maximum values are. If the frame originally had an overexposed star, then the maximum values in all channels MUST match. Otherwise, later, when overlaying the color, we will get an incorrect result. If some (usually green) channel is “undercooked”, then ALL channels should be multiplied by the value $\frac{I_{max}}{I_{max}-I_{min}}$, estimating this value on the calculator (or in the mind😳) and entering into the terminal:

>mult [coeff]

After all the manipulations, we save the result, for example, as:

>save color

Now it's time to deal with the L-channel. If necessary, we bring it to the range 0-32767, as described above, and then — in one of the most natural ways — take the logarithm of the pixel values. In this case, the new values will be calculated according to the formula:

$$y = k\frac{\lg x}{\lg x_{\max}}$$

Where x — is the original pixel value, $x_{\max}$ — pixel in the image with maximum brightness, and k — is the constant by which the result will be multiplied. As usual, this command can be entered from the terminal:

>log 32767

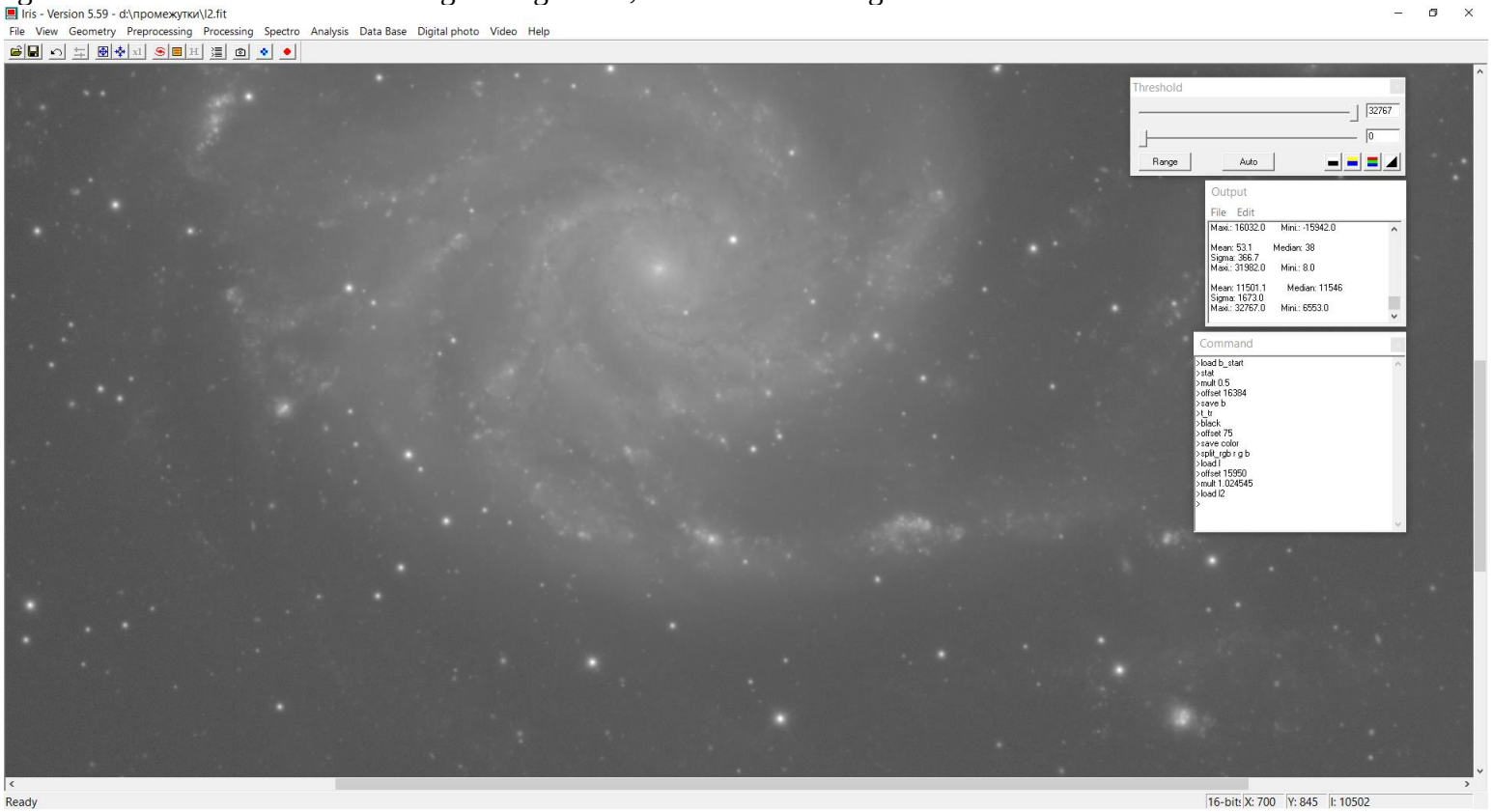
Or get the same result from the menu:

View → Logarithm

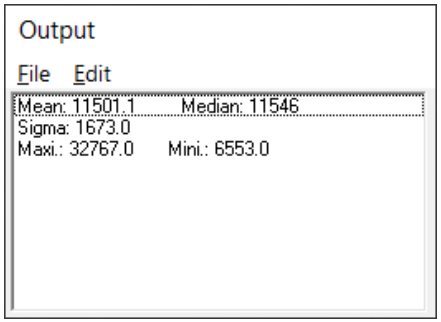
In this case the value of $k = 32767$ will be assumed by default.

Another small live hack: if you logarithming a color image, then Iris does it channel by channel. So if some channel does not have a pixel with maximum brightness, then either it needs to be added manually, or the white balance can shift significantly (For BW images, this problem obviously does not exist).

So, after loading the luminance channel and taking the logarithm, we have something like this.



Let's look at the statistics, and it is:



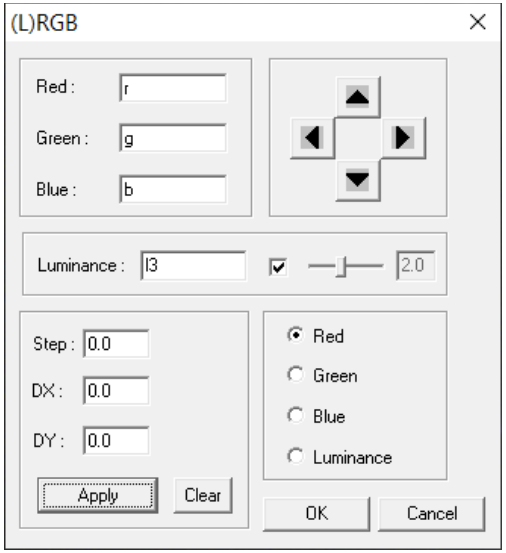
In order to use dynamic range consciously let’s apply:

>Offset -6550

>Mult 1.25

Where 1.25 is from $32767/(32767-6550)$. And we save a logarithmic picture, for example, as L3.
And now let's try to put everything together by going to the menu:

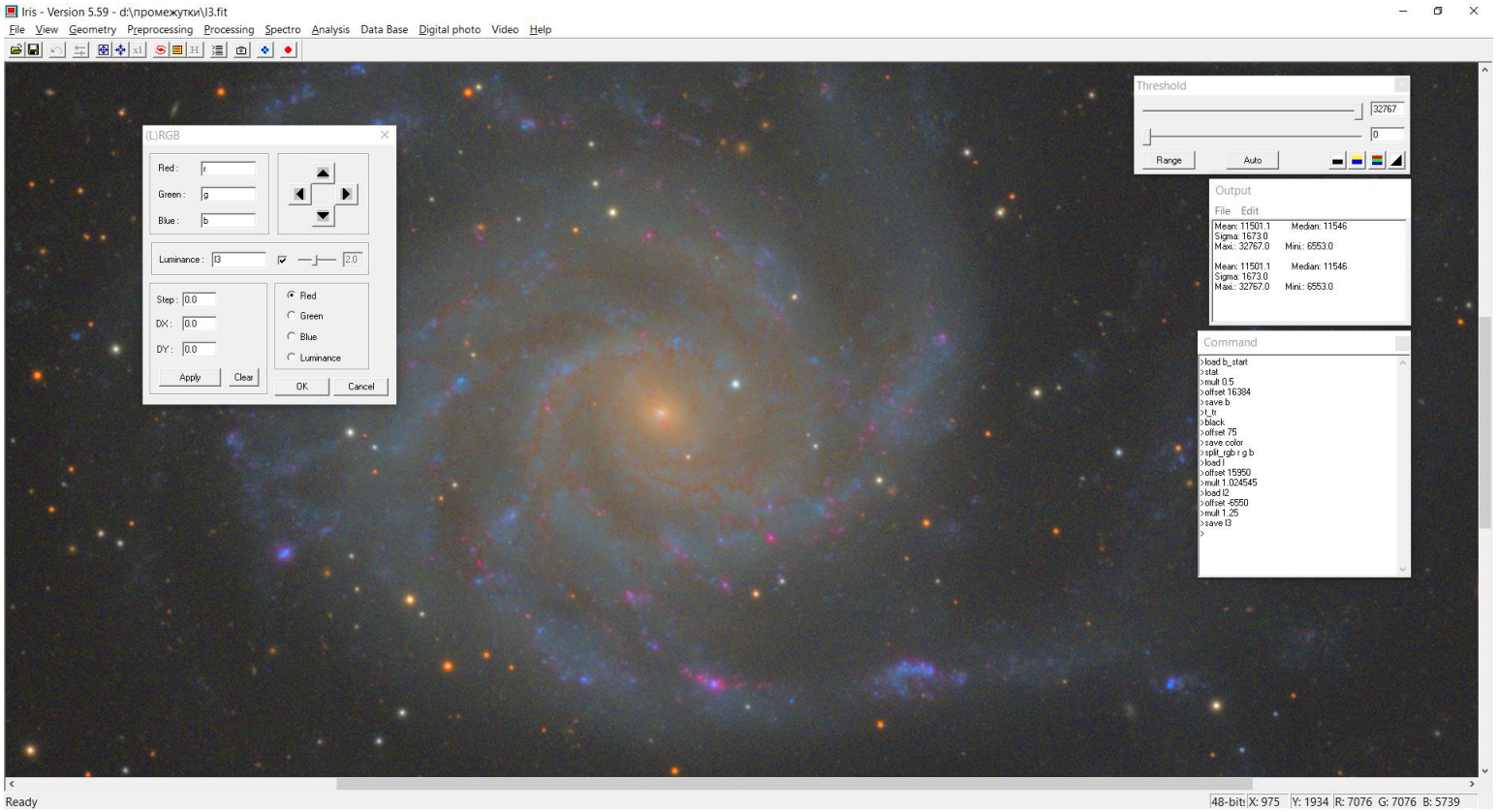
View → (L)RGB



But what a bummer, it requires to have each channel separately, and we have a color file color.fit ...
Well, not a big deal. We will load our color.fit and apply one more command:

>Split_rgb r g b

Which, as you might guess, splits a color image into 3 separate files with names that we can set ourselves. Now let's go back and fill the fields required:



By playing with the luminance slider, you can control the saturation of the color..

This standard method is bad… But I prefer the custom approach, because. there are too few variables that we can intelligently manage. Therefore, I propose to have some more fun with the command line.

We already have r g b files that we can interpret differently. Any pixel in a color image is characterized by three numbers. And from school we know that in our three-dimensional space any point can also be characterized by three numbers. So we've arrived to the idea of a color space. And as we know, coordinate systems are not only rectangular Cartesian. There are spherical, ellipsoidal, cylindrical, hyperbolic... Why don't we characterize pixels by some other values? Nobody forbids us that. There is, for example, such a system as Lab, where L is responsible for the brightness of a pixel, and the other two axes characterize its position relative to "green-magenta" and "yellow blue". But we will now look at another system, HSI, where the three values describe Hue, Saturation and Lightness. If we now take our RGB image, rewrite from it the values corresponding to the hue and saturation, and take the brightness from the logarithm frame, then we will perform the same LRGB assembly. But at the same time, for example, we can intervene in the process at intermediate stages, which gives us more freedom.

There is a special command for this conversion of RGB to HSI:

```
>RGB2HSI r g b h s i
```

Very "original". Less problems with memorizing it though. (File names still could be set by the user.)

If we now enter our files there, load the image responsible for saturation and apply the command a couple of times to it:

>gauss 0.7

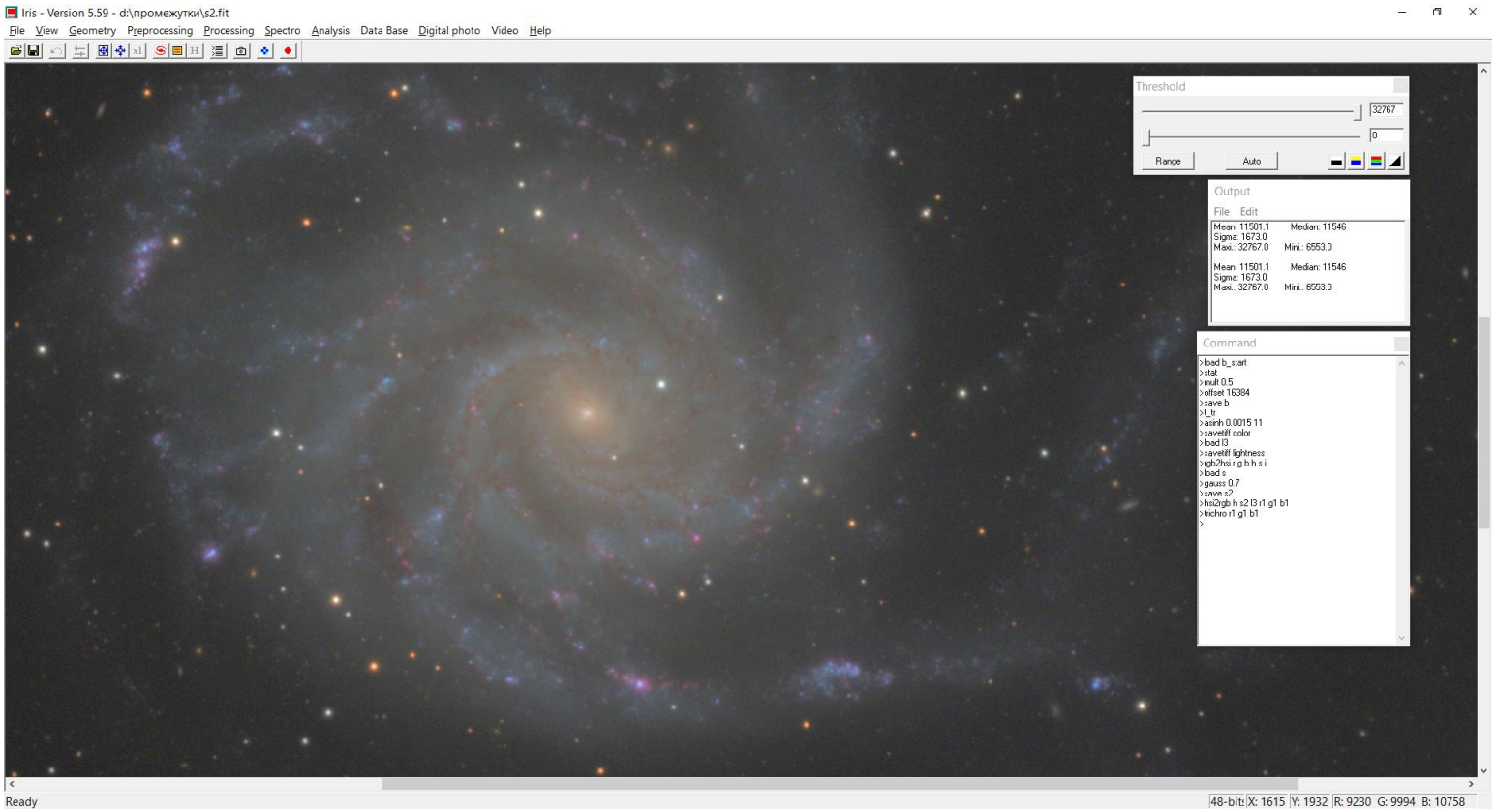
>save s2

Then we will reduce color fluctuations contribution to the image, and not distort the overall picture. The file responsible for hue (h) cannot be blurred, because otherwise, the values responsible for the positions of shades on the color wheel will change and we do not need this. After we have prepared the ”h” and “s” files in this way, we can reverse the conversion, I think you can already guess with which command:

>HSI2RGB h s2 L3 r1 g1 b1

It remains only to collect the image back:

>trichro r1 g1 b1



Now you can save it and drag it, for example, into Photoshop.

>savetiff iris_lrgb

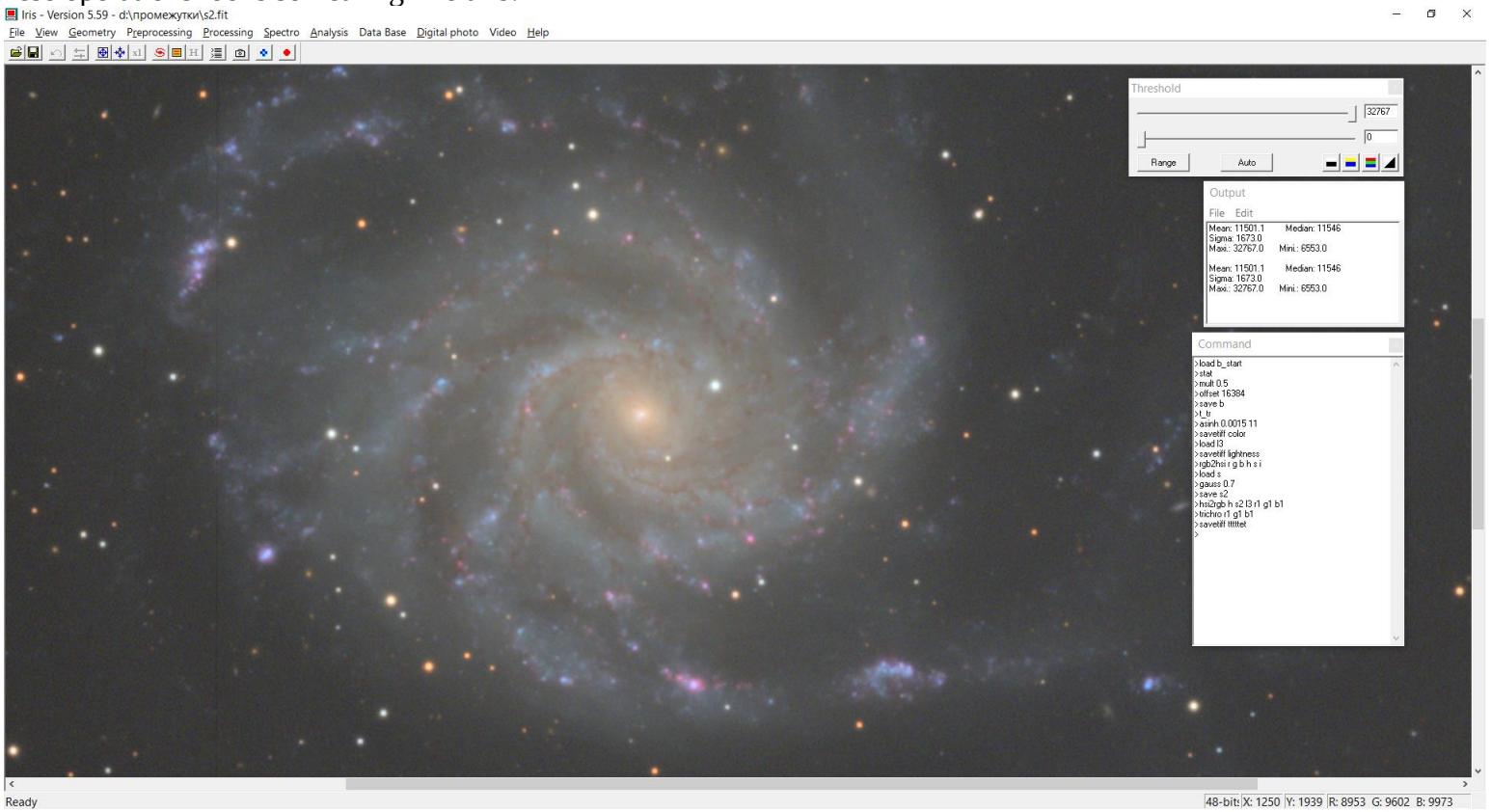
As you might guess from the name of the command, we have just saved the result to a tiff file.

It would seem that we can stop here, but I still want to show that although the Iris is a good and versatile tool, but with the aid of Photoshop we can get an even cleaner result.

Let's load our color image and apply a non-linear histogram stretch to it twice:

```
>load color
>asinh 0.0015 11
>asinh 0.0015 11
```

The result of these operations looks something like this:

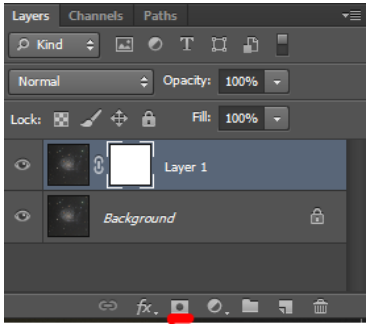


Then we save it and our transformed L-channel.

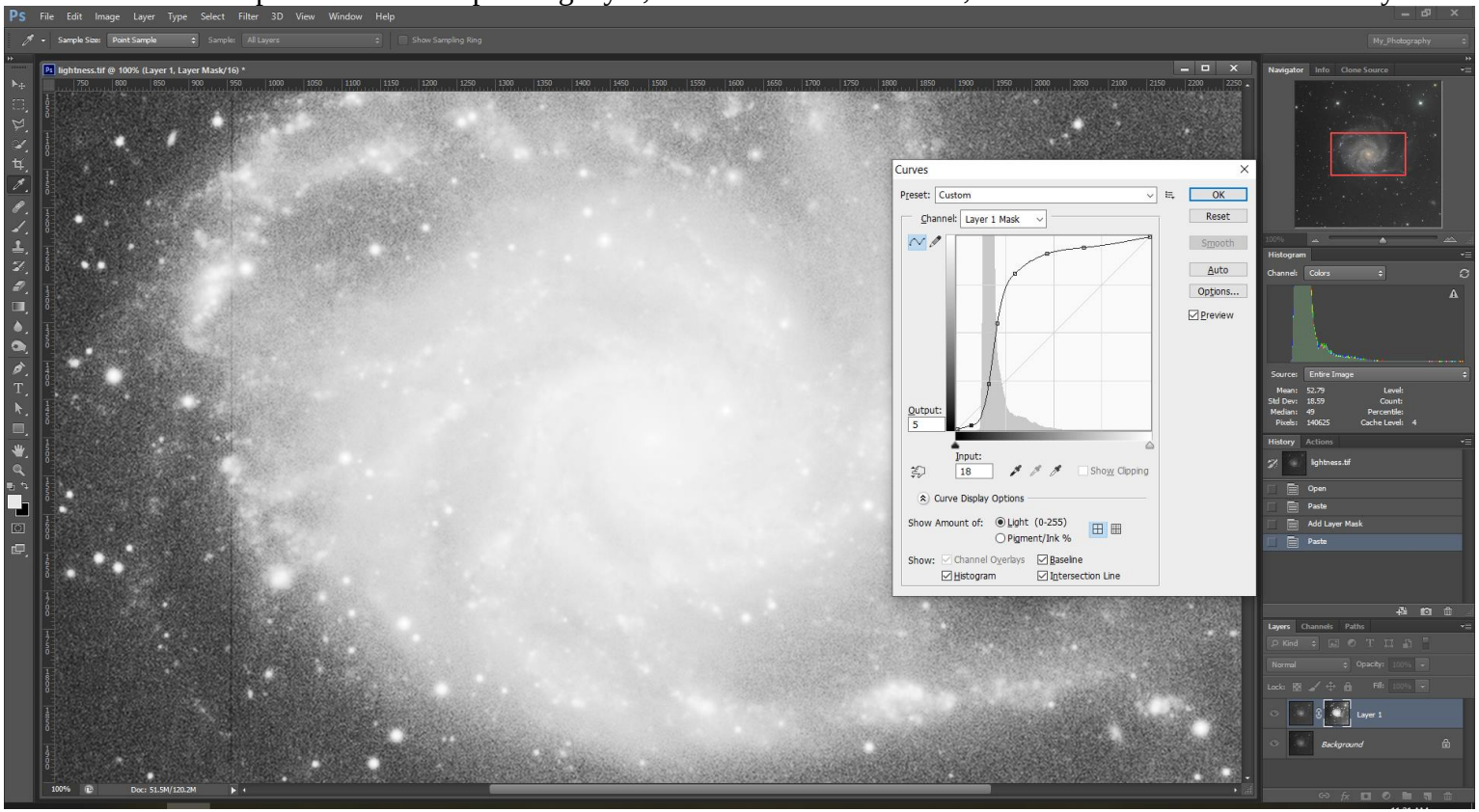
```
>savetiff color
>load L3
>savetiff L
```

And now let's take our tiffs to Photoshop and apply a color according to the brightness mask on the image from the L-channel.

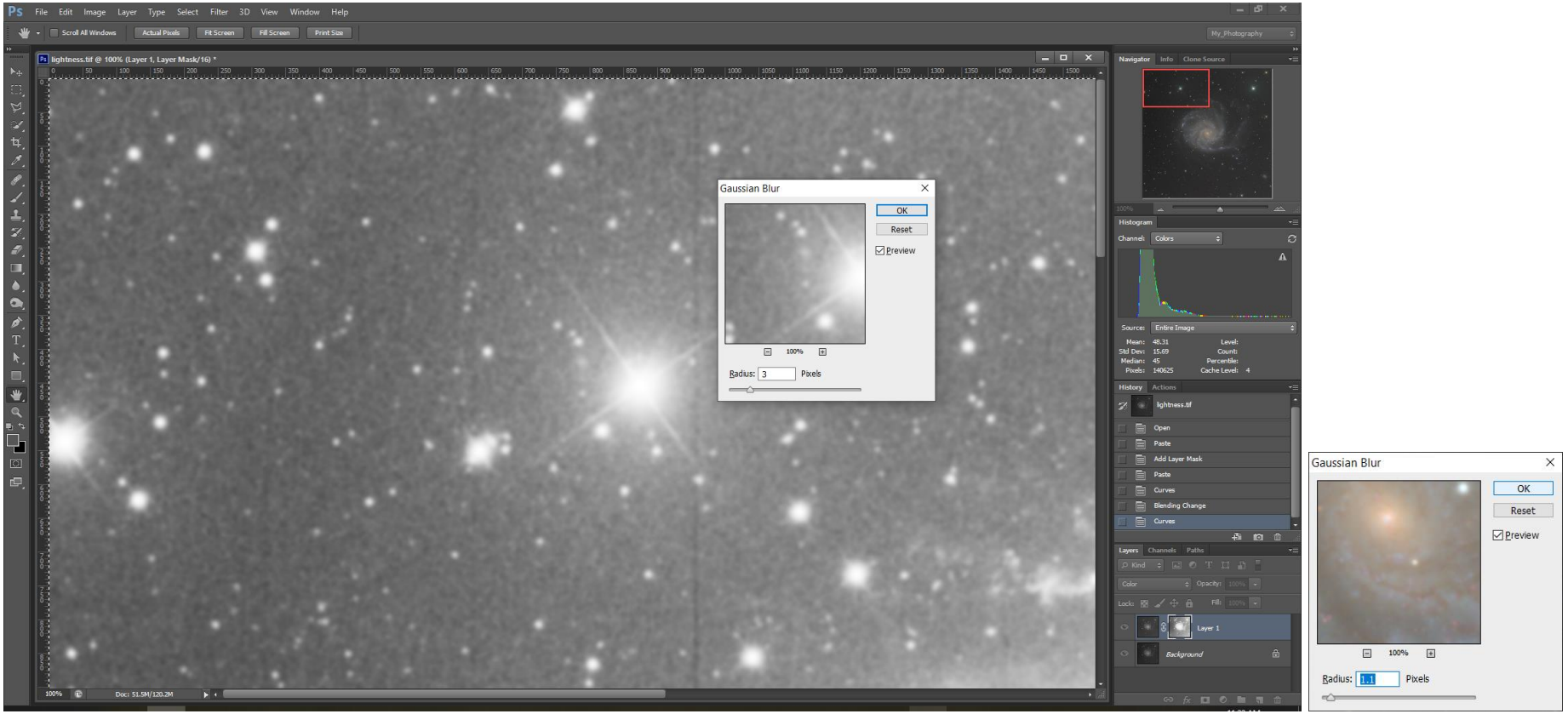
First, copy what is in the brightness layer, selecting it, and then press Ctrl + A and then Ctrl + C in sequence.
The brightness mask can be created by clicking on the icon:



And in order to enter it, you can hold down the Alt key and right-click on the white rectangle that appears. Then paste the copied image with Ctrl+V.
Now the white areas of the mask represent the corresponding layer, and the blacker the mask, the less the contribution of the layer. Let's edit it with curves:

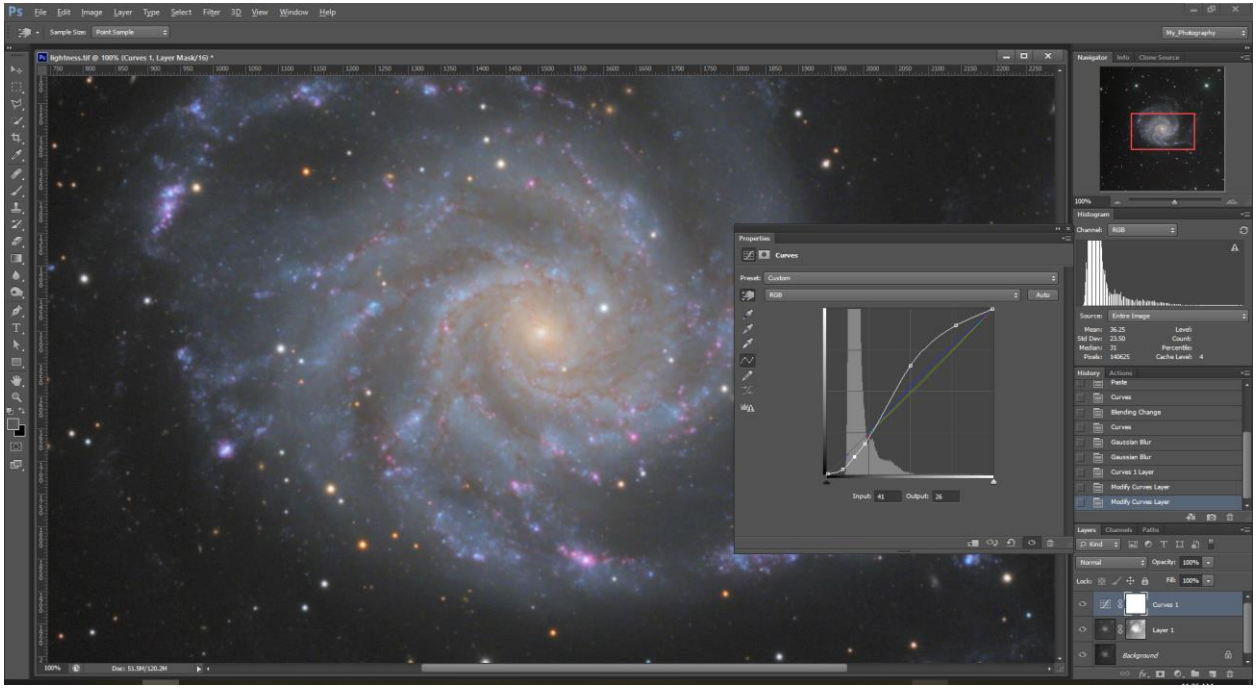


The next step is to change the blending method of the color layer to "color" This way, the bright details will be colorized, and the dark parts of the sky will remain almost neutral gray. Also, no one forbid us to blur this mask (as well as the layer containing the color itself) with a little Gaussian blur, for smoother borders and elimination of color artifacts.



Also, as long as we haven't changed the brightness and color too much, we can directly compare the results obtained with Iris alone and with Iris and Photoshop combined. I will give an enlarged area with the core of the galaxy and the periphery. It can be seen that mask addition introduces much less color noise into the dark areas of the image, so we leave the hybrid version of Iris + Photoshop for further work.

Actually, the result already looks good. You can boost the contrast a little more and correct the slight white balance shift with the help of curves. Keeping in mind that, yes, the curves can help with a LITTLE excess tint. But if the white balance was initially set grossly incorrectly, then practically no attempts to restore it will lead to anything, since the color transformations are very non-linear in nature and the balance must be set at least in the first approximation correctly even on a linear image!



For now, I propose to wrap it up, and if there are reviews, additions, wishes, then I will try to somehow expand this manual and try to systematize, although for now - I have little idea what all this will result in.