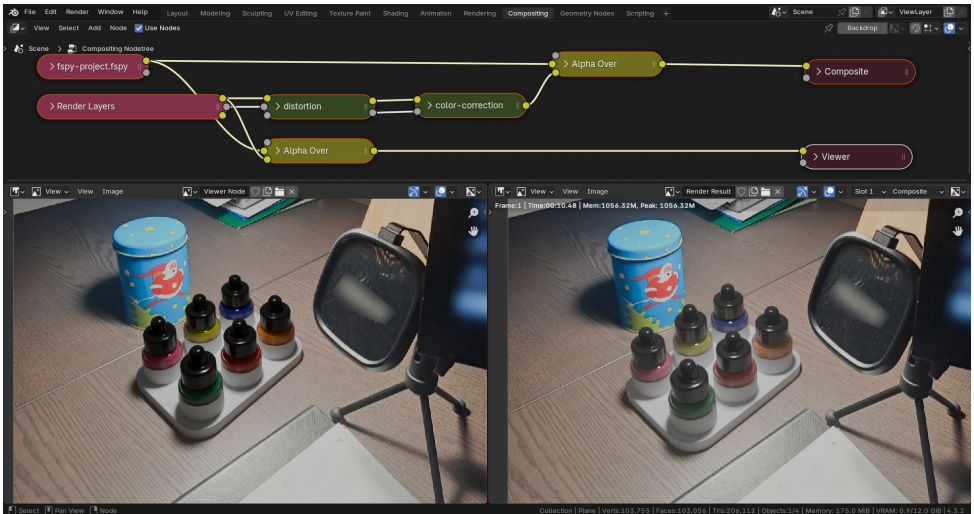


Photo Matching in Blender, Part 3: Compositing | Blender 4.x Tutorial
www.francescomilanese.com --- 2025 --- [GUMROAD \(PDF\)](#)
video version: <https://youtu.be/93OD0361-s4> (Scheduled for March 17, 2025)

This PDF is available for free; feel free to share it!

[Subscribe to my YT channel](#) & turn on notifications to get updates on new videos!



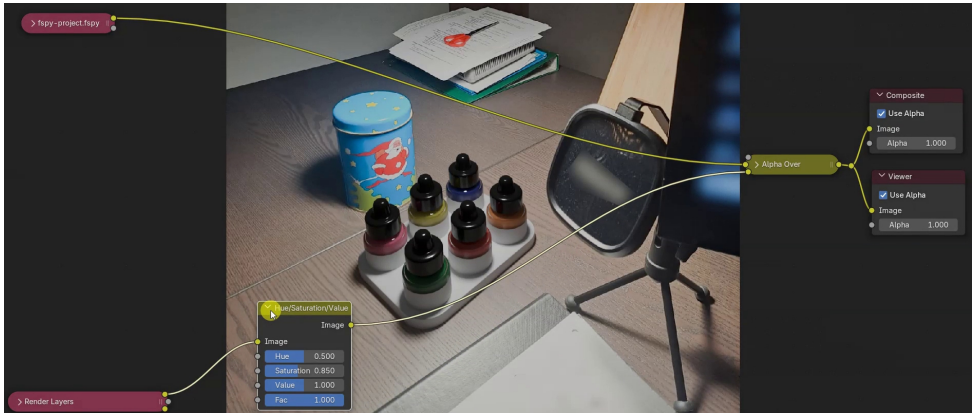
Hello everyone! This is the third and final tutorial of a mini-series focused on a small photo compositing project in Blender 4.3.

In the previous two tutorials, we covered how to recreate the framing of the original photograph using the fSpy add-on and how to insert the 3D model with shadows in Compositing using the Shadow Catcher tool.

In this tutorial, we'll see how to modify the appearance of the 3D model in Compositing with color correction, how to implement Lens Distortion, and how to retrieve the Shadow Catcher Pass information to process it separately.

As with many of my other video tutorials, the PDF version of this tutorial is available for free download via the link in the description. This is also a great time to subscribe to the channel and turn on notifications so you won't miss the next video tutorials!

Let's start with the rendered image and the Compositing setup just as we left it at the end of the previous tutorial, with a Hue Saturation Value node slightly desaturating the 3D model's rendering.



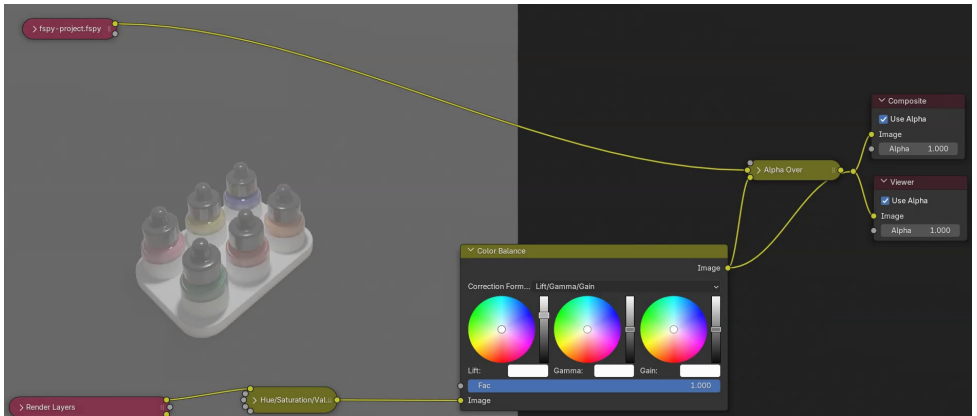
On the rendering of the model alone, we can apply further adjustments and corrections, for example using a Color Balance node. In this node, we can modify the intensity and hue of the shadows, midtones, and highlights of the image using the Lift, Gamma, and Gain controls, respectively.

In this project, we can observe that the dark areas of the original image are not black. This is because the original light source is very close and very intense. For this reason, one adjustment we can make involves the Lift channel, which controls the shadows, where we can slightly increase the intensity.

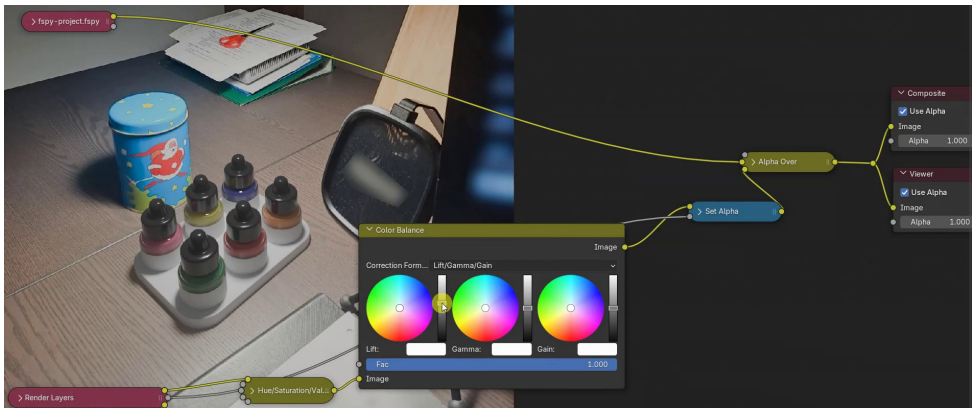
We immediately notice an issue, which becomes more evident if we connect Color Balance directly to the Viewer and significantly increase the Lift value. It's not just the rendering that becomes lighter in the dark areas, but also all the other parts that should remain transparent.

Therefore, we need to assign transparency to the image we're processing. Fortunately, this can be easily done by using a Set Alpha node between Color Balance and Alpha Over, using the Alpha output from Render Layers for transparency.

Photo Matching in Blender, Part 3: Compositing | Blender 4.x Tutorial
www.francescomilanesi.com --- 2025 --- [GUMROAD \(PDF\)](#)
video version: <https://youtu.be/93OD0361-s4> (Scheduled for March 17, 2025)



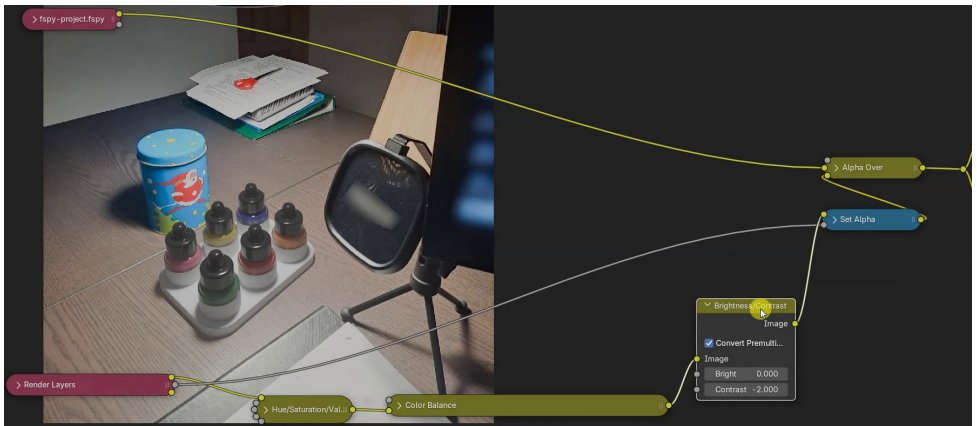
More generally, this Set Alpha node should be placed between the last rendering adjustment node and the Alpha Over node in the final Compositing.



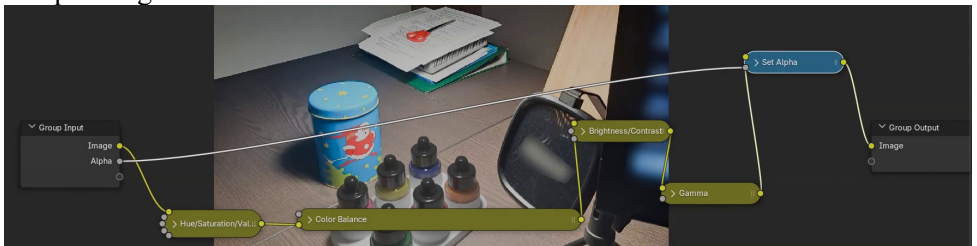
We can then modify the Color Balance and add other adjustment nodes, such as Brightness Contrast or Gamma, to better integrate the rendering with the original photograph. For example, I'm trying to reduce the contrast in the rendering.

If you want to make the rendering lighter or darker, I recommend using a Gamma node instead of Brightness or RGB Curves. The Gamma node applies a brightness correction that avoids pushing the lightest and darkest areas to pure white or pure black. Pure black and pure white correspond to values of 0 and 1, which can cause issues with further operations involving color addition or multiplication.

Photo Matching in Blender, Part 3: Compositing | Blender 4.x Tutorial
www.francescomilanese.com --- 2025 --- [GUMROAD \(PDF\)](#)
video version: <https://youtu.be/93OD0361-s4> (Scheduled for March 17, 2025)



To separately evaluate the effects of these adjustments on the original rendering, I'm grouping the nodes from Hue to Set Alpha into a sub-group by selecting them and pressing the shortcut CTRL G.



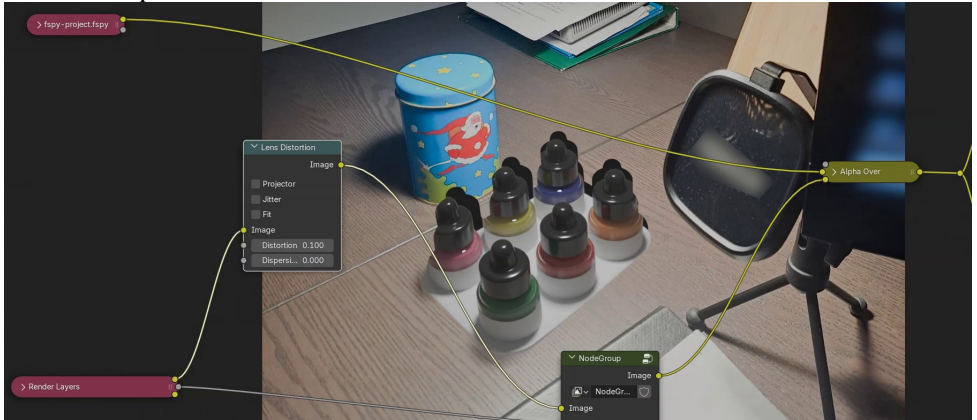
We can access this subgroup by selecting it and pressing the **TAB** key. The same key can be used to exit the subgroup and return to the original node setup.

This approach allows us to modify the color correction nodes more easily within the group while also making it possible to disable the entire group by pressing the **M** key, which serves as the shortcut for both **Mute** and **Unmute**, allowing us to evaluate the overall effect of the color correction on the original rendering. Here, I'm only showing a few basic examples, but adjustments to color, brightness, and contrast depend heavily on the specific project you're working on.

A separate discussion is needed for the **Lens Distortion** operation, which allows us to distort the rendering to better integrate it with the photograph, where some lens-induced distortion is typically present.

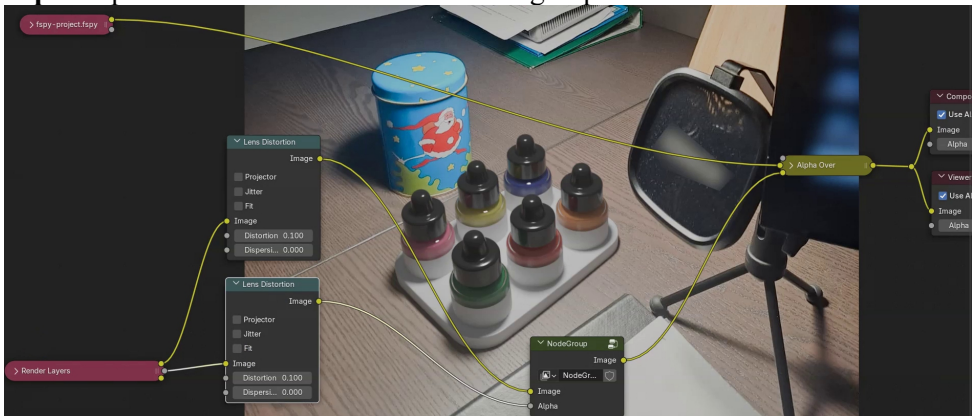
Photo Matching in Blender, Part 3: Compositing | Blender 4.x Tutorial
www.francescomilanese.com --- 2025 --- [GUMROAD \(PDF\)](#)
video version: <https://youtu.be/93OD0361-s4> (Scheduled for March 17, 2025)

My recommendation is to place this node between the original rendering and other operations, such as the color correction subgroup, because the distortion will introduce an issue with transparency. This is noticeable even when using a low **Distortion** parameter value.

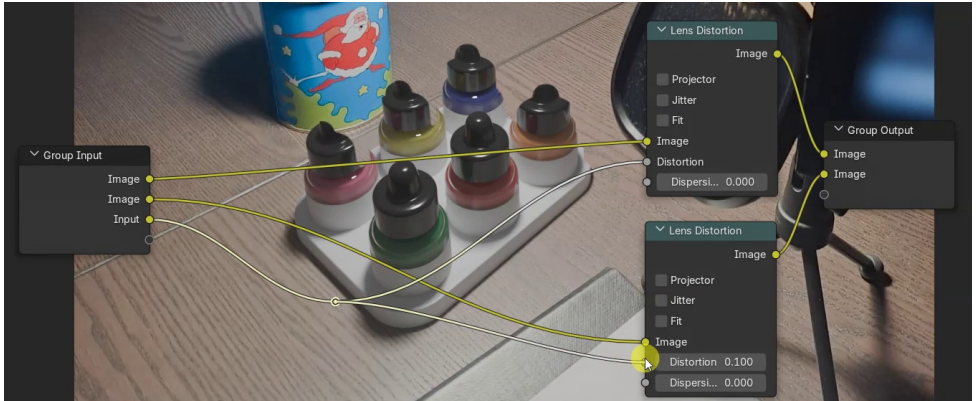


The problem arises because **Lens Distortion** modifies the appearance of the rendering but not its transparency, resulting in black areas where the image has been separated from its shadow. Similarly, there will be semi-transparent areas where the image overlaps regions that were originally transparent in the rendering.

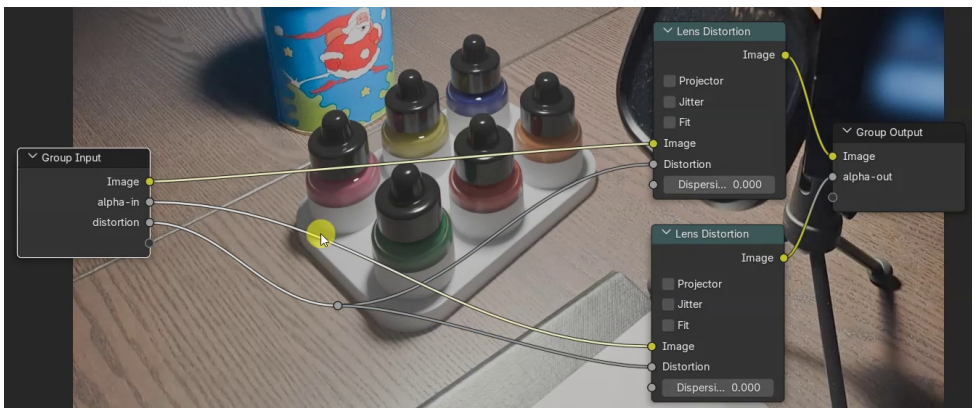
To solve this issue, I apply the same distortion operation to the **Alpha** channel of the rendering using a second **Lens Distortion** node. Then, I send this result to the **Alpha** input of the color correction node subgroup.



To modify the **Distortion** values of both nodes simultaneously, I can group these two nodes into a new subgroup and connect their **Distortion** inputs to a single input of the subgroup, using a **Reroute** node for this operation. This way, we can adjust both values with a single parameter, preventing the object from losing its shadow.



In the **Group** tab within a subgroup, we can also modify the names and colors of the input and output ports. This is particularly useful for the **Distortion** input of the new subgroup, as we can change the input port colors and, more importantly, switch the port type for the **Distortion** value from color to numeric.

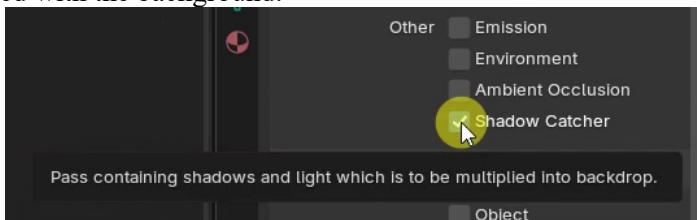


The names of node groups can be changed in the **Node Properties** tab of the group, with the group closed and selected. These small adjustments make it easier

to work with node groups, especially in complex node setups or those rich in parameters.

The final topic of this tutorial covers how to extract only the shadows produced by the **Shadow Catcher** so that they can be processed separately. The first step, of course, is to enable the **Shadow Catcher** option in the **Light Passes** section of the **View Layer** tab.

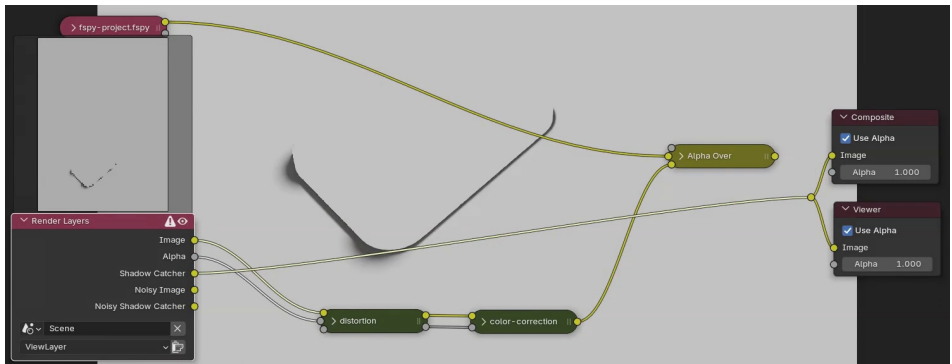
The tooltip for this option immediately tells us that this information will need to be multiplied with the background.



If we run a render now and switch to the **Compositing** workspace, we'll notice that the shadows are no longer integrated into the final rendering.



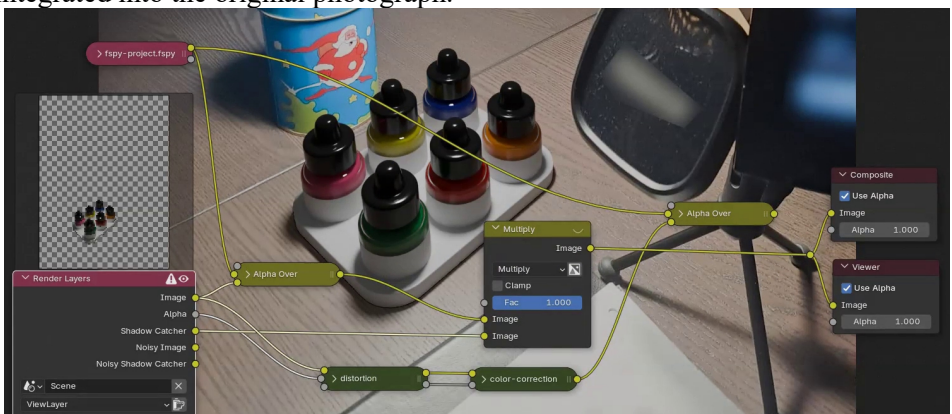
This happens because enabling that **Render Pass** separates the **Shadow Catcher** information from the rendering. We can view this information by connecting it directly to the **Viewer** node.



As mentioned in the tooltip of the **Shadow Pass** option, this mask needs to be multiplied in the Compositing to integrate these shadows into the final result. However, since we've applied several operations to the rendering, including a **Lens Distortion** operation, this separated information doesn't suit our purposes.

To recreate the basic Compositing with this separated information, we first need to connect the **Image** output from **Render Layers** to the second **Image** input of an **Alpha Over** node. Then, we'll connect the original photograph to the first **Image** input of the **Alpha Over** node.

Now, we need to multiply the output of **Alpha Over** with the information from the **Shadow Catcher** using a **Color Mix** node set to **Multiply** mode, with **Shadow Catcher** connected to the second **Image** input and the **Factor** parameter set to 1. This will give us the initial Compositing, with the object and its shadows integrated into the original photograph.



Separating the **Shadow Catcher** information in Compositing is useful when we only need the shadows, for example, because we plan to replace the model used in the rendering with something else later on.

For our purposes, the **Shadow Catcher** Render Pass should not be enabled, as we want that information already integrated into the output image from **Render Layers**.

That's it for this third and final episode of this small photo compositing project in Blender! I hope you found this mini-series helpful.
See you next time!