The algorithm goes like this:

* **Step 1:** Input a source and a target image. The source image contains the color space that we want our target image to mimic.
* **Step 2:** Convert both the source and the target image to the L\*a\*b\* color space. The L\*a\*b\* color space models perceptual uniformity, where a small change in an amount of color value should also produce a relatively equal change in color importance. The L\*a\*b\* color space does a substantially better job mimicking how humans interpret color than the standard RGB color space, and as we’ll see, works very well for color transfer.
* **Step 3:** Split the channels for both the source and target.
* **Step 4:**Compute the mean and standard deviation of each of the L\*a\*b\* channels for the source and target images.
* **Step 5:** Subtract the mean of the L\*a\*b\* channels of the target image from target channels.
* **Step 6:** Scale the target channels by the ratio of the standard deviation of the target divided by the standard deviation of the source, multiplied by the target channels.
* **Step 7:** Add in the means of the L\*a\*b\* channels for the source.
* **Step 8:** Clip any values that fall outside the range *[0, 255]*. (**Note:** This step is not part of the original paper. I have added it due to how OpenCV handles color space conversions).
* **Step 9:** Merge the channels back together.
* **Step 10:** Convert back to the RGB color space from the L\*a\*b\* space.

While the Reinhard et al. algorithm is extremely fast, there is one downside — it relies on global color statistics, and thus large regions with similar pixel intensities values can dramatically influence the mean (and thus the overall color transfer).

To remedy this problem, we can look at two solutions:

**Option 1:** Compute the mean and standard deviation of the source image in a smaller region of interest (ROI) that we would like to mimic the color of, rather than using the entire image. Taking this approach will make our mean and standard deviation better represent the color space we want to use.

**Option 2:** The second approach is to apply k-means to both of the images. We can cluster on the pixel intensities of each image in the L\*a\*b\* color space and then determine the centroids between the two images that are most similar using the Euclidean distance. Then, compute our statistics within each of these regions only. Again, this will give our mean and standard deviation a more “local” effect and will help mitigate the overrepresentation problem of global statistics. Of course, the downside is that this approach is substantially slower since we have now added in an expensive clustering step.