Restoration of Faded Slides – version 6

Geoff Daniell
gjd@lionhouse.plus.com

May 27, 2026

1 Background

For many years I have worked on algorithms for the automatic improvement of colour slides or prints that have deteriorated with age. Originally the code was implemented as a plug-in for the `gimp` image processing package but more recently python code has been developed to run as a separate program. The significant feature is that the restoration is completely automatic and requires no decisions by the user as to the best result.

In recent years Artificial Intelligence software has become available which attempts the same task. Comparison shows that my algorithm produces very acceptable results for the same faded images, but the AI results look slightly more pleasing. The new code described here is an attempt to meet the challenge of AI. The result is a comparatively small python program which runs in a few seconds on a local machine and is independent of the the internet.

It is assumed that the faded slide or print has been scanned and we start with a file containing the red green and blue intensities for each pixel recorded by the scanner.

2 Overview

The general strategy used here for restoring an image involves two components: a set of possible changes to the image, for example increase the intensity in the red channel by x% and a measure of how much better the image looks after the change. A more complicated change might be to decrease the intensities of the *dark* green colours. The program makes this set of adjustments until the image looks as good as possible according to some criterion. We therefore need a suitable set of changes and a measure of how good the resulting image is.

Previous documents discuss the process of colour photography and possible models for deterioration. The most plausible processes are that the quantities of the coloured dyes in the image have changed. They have probably decreased but in some colour films precursor chemicals of the dyes remain in

the emulsion and the quantity of dye could increase with age. An approach to restoration could then involve computing the quantity of each coloured dye in the emulsion and changing it. A mathematical analysis shows that this is equivalent to scaling the intensities of the pixels in the red, green and blue channels, and adjusting their distribution; in photography this adjustment is known as altering the 'gamma'. It is difficult to think of other simple colour adjustments that might improve the image and the ones just described are the ones used in the current program.

A bigger challenge is the invent a number that represents the 'goodness' of the resulting image. I do not know how AI restoration programs are trained; presumably they need a large corpus of 'good' images. There are AI programs that make a reasonable job of coloring monochrome images, (although I do have an example with bright green sunflowers!). These programs must rely on the image *content* to decide the colours. In contrast my method relies only on the *colours* in the faded image and the content is irrelevant. Underwater scenes and sunsets are bound to fail with my approach but AI techniques ought to do better; we can however work only with the colours.

Earlier versions of this software used various ideas for deciding what is the 'best' image; some of these worked well and are retained in the current version. While it is difficult to define a 'good' picture we can easily recognise a bad one, if one colour channel is missing, for example. I am putting forward the idea that the 'best' image attainable using only colour information should be as 'average' as possible. One argument in support is that there are more images approximately 'average' than 'extreme' and so the one we pick is more likely to be a 'good' one. There is nothing fundamental about this rule; we have to make a choice and this seems to work quite well.

3 Colours in the Image

In thinking about faded coloured images and their restoration the useful information is in the *range* of colours. The number of pixels in each colour is almost irrelevant. If the image has an overall blue cast but there are a few white pixels then the blue colour cannot be due to changes in the coloured dyes which would affect all pixels.

Several algorithms for colour quantisation exist in various image processing packages. It seems better not to rely on these as the details are often obscure and the output might change in new versions of the code. To make a list of the different colours present in an image we examine all the pixels and note the most significant four bits of the intensity in each of the red, green and blue channels. This leads to a maximum of 4096 (R, G, B) triplets of colour. When a new triplet of found the colour is added to the list. The code also counts the number of the pixels with that colour in case it is useful later.

Although we have argued that the number of pixels of a particular colour is irrelevant this is mot quite true. A few very dark or bright pixels may be significant in calculating the scaling of the overall lightness (see below).

Experiments show that rejecting colours with few pixels leads to better results. The rule adopted usually rejects about 20% of the colours.

The numbers in the list of colours represent the intensities recorded by a film scanner. The whole subject of 'Colour Science' is very confusing because these intensities are not what are recorded by the human eye and interpreted by the human brain. Several alternative 'colour spaces' have been invented to better describe how we interpret colour; for example instead of (red, green, blue) a colour can be represented as (hue, saturation, value). Some of these alternatives attempt to get closer the how we *perceive* different colours. Of particular relevance here is (L^*, U^*, V^*) colour space which is a set of three numbers derived from the (R, G, B) values by non-linear transformations. L^* is designed so that it represents 'lightness' and U^* and V^* contain colour information. The important point is that the (L^*, U^*, V^*) colour space has an approximate metric, that is the space can be divided into cells of equal size which correspond to differences of lightness and colour that are just distinguishable to the human eye. L^* is arranged to lie between 0 and 100 and U^* and V^* are usually between -100 and +100. It is important to stress that there is nothing special about (L^*, U^*, V^*) and there are more recent alternatives that are more useful in particular circumstances. The whole subject is very complicated!

Figure 1 shows a faded slide.

Figure 1: left: An old slide, right: The colours in the image

On the right the approximately 200 separate (R, G, B) colours in the image have been converted to (L^*, U^*, V^*) values and their U^* and V^* values are plotted as small squares which for convenience are coloured with the actual colours. The L^* coordinate is perpendicular to the page but some information about the 'lightness' is deducible from the colours. It turns out that the distributions along this third axis are interesting. The positions in (U^*, V^*) space of the primary colours are also shown.

It is clear that red colours predominate and all have $U^* > 0$. On the other hand the values of V^* are both positive and negative. It is also clear that the darkest and lightest colours are near the origin.

This leads to the idea of plotting the values of U^* and V^* against L^* for all the colours in the image. Figure 2 shows the approximately 200 separate colours with the values of U^* (in red) and V^* (in blue) plotted against L^*.

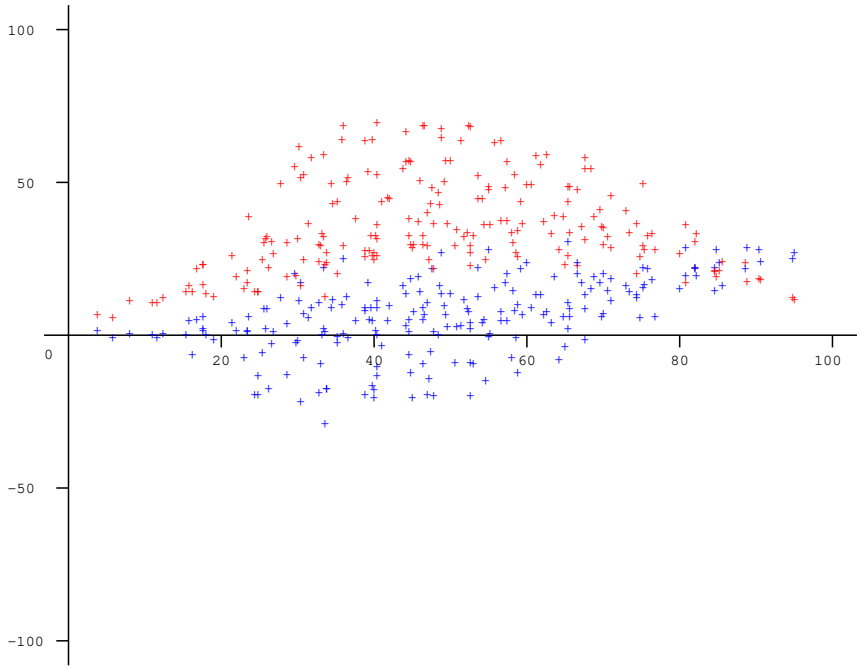

Figure 2: The colours U^* (red) and V^* (blue) plotted against L^*

Again the fact that all colours have $U^* > 0$ is clear but the darkest and lightest colours (at the ends of the L^* axis) have smaller values of U^*; so the observed variation of U^* has a large positive offset and a roughly parabolic form. V^* also shows some systematic variations but these are more complicated and contain a linear component as well. This means that the image is not an 'average' one which we suggest could be the 'best'. For an 'average' image the variation of U^* andV^* with L^* should exhibit no systematic variation and U^* and V^* should be scattered symmetrically about zero at each value of L^*. The mathematical appendix describes how we can alter the image and construct a number that describes this departure from average.

The restoration proceeds as follows: from the (R, G, B) values in the original image we compute numbers representing the quantities of the coloured dyes in the emulsion. We try increasing or decreasing these and compute the resulting (R, G, B) values of the changed image and convert these to (L^*, U^*, V^*). The variations of U^* and V^* with L^* are examined and our quality measure of the image is calculated. The process continues until the quality

ceases to improve.

Figure 3 shows the result and the better spread of colours in (U^*, V^*) space.

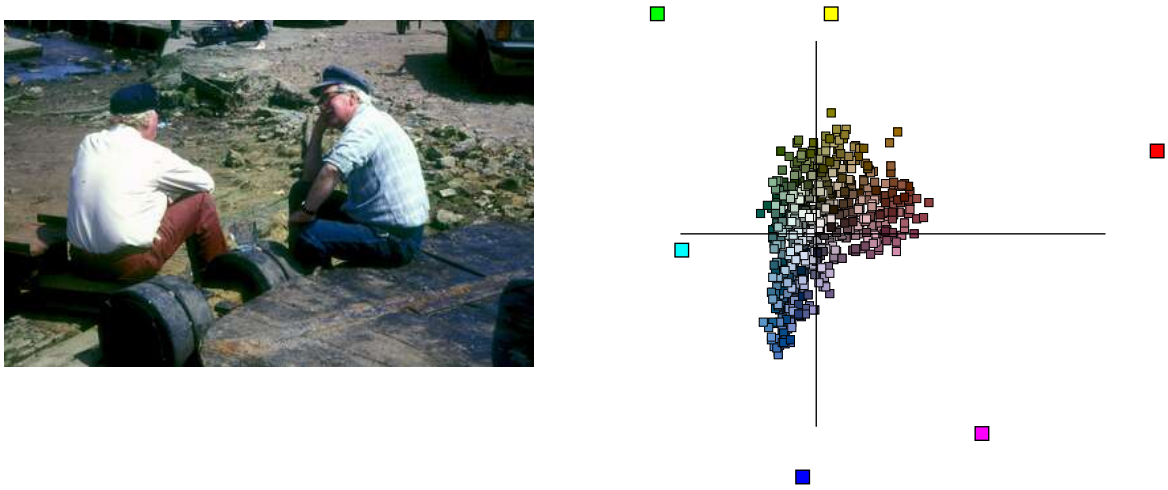

Figure 3: left: The restored image, right: The better spread of colours

Figure 4 shows the plots of U^* and V^* against L^* and the symmetric distribution of U^* and V^* about zero, at all values of lightness L^* is clear.

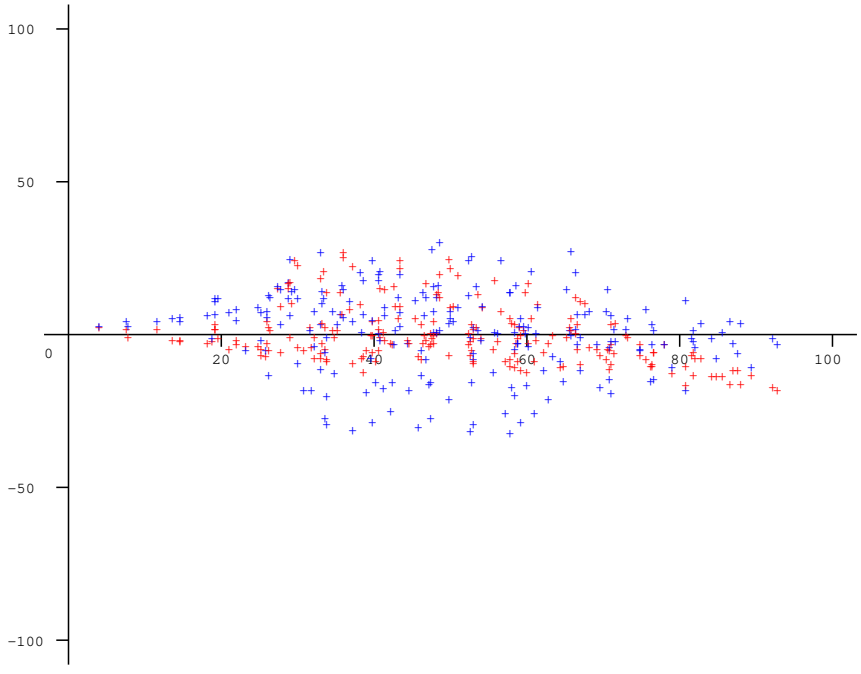

Figure 4: The colours U^* (red) and V^* (blue) plotted against L^*

4 Pre-processing, Lightness, White Balance and Saturation

Pre-processing
It turns out that the iterative search for the restoration parameters sometimes fails to converge. This can happen for slides that have gone very dark. To improve the stability of the iteration it is carried out in two stages. The first checks the spread of intensities in each colour channel separately. According to our theory of 'the average' the spread of colours (not pixel numbers) in each of the red, green and blue channels ought to be similar and should extend from near zero to near 255. The parameters needed for this are computed and remembered. The image is restored using these values and the new colours calculated. The proper restoration described above is then carried out and the parameters for this are combined with the values noted for the first stage. This achieves the same result as a single stage search but is less likely to get lost.

Lightness
Among the desirable properties of a good image is a wide range of intensities; in fact most image processing software includes a procedure for stretching the range automatically. Earlier code included a rather arbitrary rule for deciding the maximum overall intensity in the image. The new version uses a better method that is likely to work for a wide range of images.

It can be seen from figure 4 above that there are few very dark or bright colours and the majority of L^* values cluster about the middle. It is straightforward to change the range of lightness using a look-up table to arrange that there are at least a few very dark colours and a few very bright ones. It is also desirable that the spread of lightness value is reasonably symmetric about a value of about half the maximum. The details of how this is achieved are set out in the appendix.

Saturation
The restoration as described so far produces a good colour balance even from images that have a strong overall colour cast. I am reluctant to abandon this, but one of the side effects is that the colours tend to be rather unsaturated. This is because U^* and V^* values are pushed to be closer to zero to make the distributions symmetric. The appearance of the restored images is usually improved by an increase in saturation.

Several different definitions of saturation are in use and some are chosen for their computation convenience rather than their effect. In terms of (L^*, U^*, V^*) a good measure is

$$S = \frac{\sqrt{U^{*2} + V^{*2}}}{L^*}$$

The top is the radial distance from the point $U^* = 0$, $V^* = 0$ which corresponds to grey or white. Plots of this against L^* similar to figure 4 show

that S usually decreases significantly as L^* gets large. The correction chosen here multiples U^* and V^* by $(\alpha L^* + \beta)$ in which the constants α and β are chosen so that the saturation is roughly independent of brightness and the average value over all colours is fixed.

White Balance
A complication is that although the initial restoration leads to good white pixels, if there is a slight deviation from white this is amplified by the boost in saturation and spoils the result. To prevent this a 'white balance' step is performed. Digital cameras normally include such a step automatically. It turns out that the definition of the 'white point' in the colour calculations is *very* critical and the default values chosen have been determined empirically by looking at a selection of results. An very small correction is applied for each individual image which is calculated by making the 'white point' the average colour of a selection of pixels chosen to be bright and near white. Note that in this case the number of pixels is taken into account in addition to the colour.

Putting it all together
The steps in the code are as follows:

1. The main restoration parameters are found using the list of colours in a small copy of the faded image.
2. This small faded image is restored using these parameters and a new list of colours produced.
3. A look-up table for the scaling of L^* is computed.
4. A scale factor to be applied to all values of U^* and V^* is computed.
5. The original full size image is restored using the restoration parameters.
6. The full sized restored image is 'enhanced' by scaling L^*, U^* and V^*.

This last step is slow but can be justified by the good result.

5 Technical details

The code is written in python 3.14. Portions of the PIL library are used but only for standard operations. Because the final restoration of the image is computationally intensive the `numpy` module is used. This calculates with whole arrays of data without the overheads of interpreting python code. The program prints information about the restoration process and detects cases where it appears to have failed.

6 Testing

I explained in the document **Restore2.pdf** that I do not think a comparison with previous algorithms is very helpful; it is always possible to find an image that is restored better by algorithm A than by algorithm B. Also the criterion for approval is whether we like the final result; we do not know if it is like the original before fading.

The first test is an artificially changed digital image. The green channel has been scaled and then the image restored as if it were degraded by age. In fact the reverse of this scaling is precisely one of the changes used in the restoration, so it is not surprising that the result is good.

Figure 5: left: Original. centre: Degraded. right: Restored

Figures 6 and 7 show typical old Kodachrome which has turned blue. Figure 7 was included in a previous document as a example of a poor restoration because the snow had a pink tinge. The latest code is much better.

Figure 6: left: Original, right: Restored

Figure 8 is an unusual case where the slide has become green. The restored picture is a bit dark but has reasonable colour balance.

In contrast to Kodachrome, which turns blue, Agfa slides, which use a different process turn magenta with age, often quite rapidly. The discussion in the text above uses one example, Figure 9 is another.

Figure 7: left: Original, right: Restored

Figure 8: left: Original, right: Restored

Figure 9: left: Original, right: Restored

The next two examples show the results from badly degraded images taken using cheap colour film. The first was taken in pouring rain. it contains almost no information and could not be expected to be improved much. The second was taken in better weather using the same cheap film.

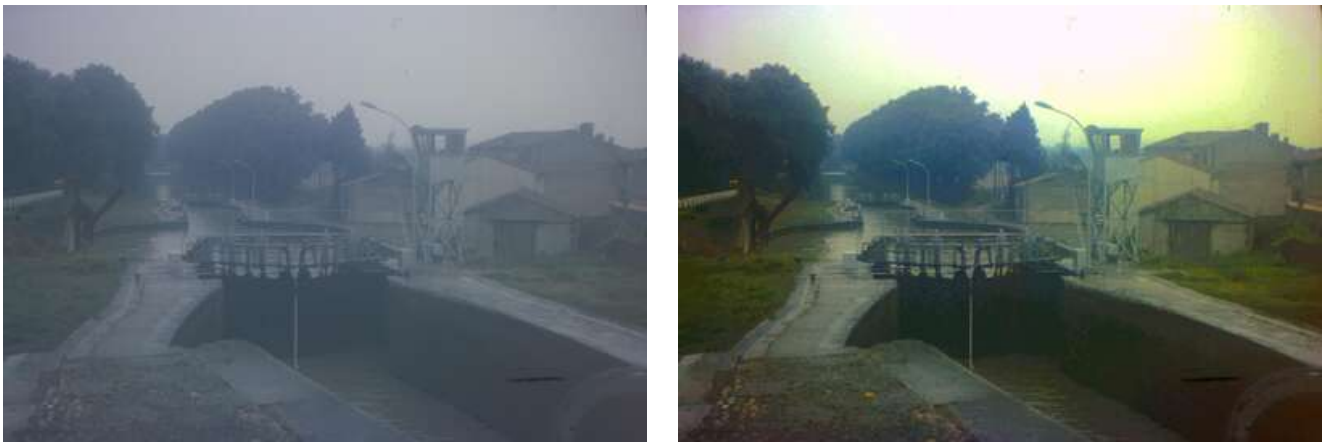

Figure 10: left: Original, right: Restored

Figure 11: left: Original, right: Restored

At one time I listened to warnings about poor colour film processing by film manufacturers and chemist shops and I used a specialist film processing company. The resulting slides have not lasted well and although the coloured dyes seem satisfactory the pictures turned dark grey. I suspect this is silver. Colour photographic processes all start with a silver halide emulsion; this is processed to produce coloured dyes and the silver compounds subsequently removed. I suspect the washing part of the processing was inadequate and the remaining silver compounds turned to metallic silver with age. The following three examples show that the restoration process deals well with these cases.

Figure 12: left: Original, right: Restored

Figure 13: left: Original, right: Restored

Figure 14: left: Original, right: Restored

Some images have quite good colour balance but the colours are just faded. The following are a couple of examples.

Figure 15: left: Original, right: Restored

Figure 16: left: Original, right: Restored

7 Conclusions

Previous documents have described computer code for restoring colour photographs that have deteriorated with age. This is the latest and is probably the best. It does however have to compete with AI models which can use predictions based on the image *content* while the code described here can

only use *colour* information in the faded image. I have not done a systematic comparison as the AI code will be rapidly evolving but a few trials show that while my code produces acceptable results those from the AI code are somewhat more appealing.

This needs to be balanced by the facts that my code runs locally, is fast and allows batch processing.

The program processes all the `.jpg` or `.JPG` files in a given directory and puts the restored versions in a subdirectory called `restored` (which must exist). The process is now entirely automatic and there are no options for the user.

Appendix

The colour balance My original discussion of the process of deterioration showed that if C is the intensity in one of the (R, G, B) channels in the faded image then the corresponding value in the original can be calculated as

$$C' = 255\,\sigma\left(\frac{C}{255}\right)^{\lambda}$$

where σ and λ are parameters describing the deterioration of each colour channel. These six numbers are found by a systematic search to minimise some property of the restored image. In the current method we choose them to get the plots of U^* and V^* against L^* as flat as possible

The calculation is described below; readers who are familiar with the theory of orthogonal functions will appreciate why the calculation is done in this way. Some properties of Legendre polynomials are assumed without explanation.

We start with the function $U(L)$ where L covers the range 0 to 100 and this function is assumed to include all the fine detail visible in the plots of U against L. In the current software the average L is forced to equal 50 and the symbol L_0 is used for this number.

If we define $\widetilde{U}(x) = U((x+1)L_0)$ then x lies in the range $[-1, 1]$. $\widetilde{U}(x)$ can now be expanded as a series of Legendre polynomials:

$$\widetilde{U}(x) = \sum_n u_n\,P_n(x)$$

and the coefficients u_n are given by

$$u_n = \frac{2n+1}{2}\int_{-1}^{+1}\widetilde{U}(x)P_n(x)\,\mathrm{d}x$$

We can now construct a measure of the average deviation of U from zero as

$$\int_{-1}^{+1}\left[\widetilde{U}(x)\right]^2\,\mathrm{d}x$$

and substituting the series of Legendre polynomials this can be shown to equal

$$\int_{-1}^{+1}[\sum_n u_n\,P_n(x)]^2\,\mathrm{d}x = \sum_n \frac{2}{2n+1}u_n^2$$

In the plots of U against L in figure 2 we have considered only constant, linear and quadratic variations, so we can truncate the series at $n = 2$. Including only terms up to quadratic the quantity to minimise is

$$2u_0^2 + \frac{2}{3}u_1^2 + \frac{2}{5}u_2^2$$

although this formula can obviously be generalised.

The value of the above discussion is that it gives the relative importance of the constant, linear and quadratic variations in U. We now need to consider how to calculate the coefficients u_n. The formula above for u_n can be written

$$u_n = \frac{2n+1}{2}\int_{-1}^{+1} U((x{+}1)L_0)\,P_n(x)\mathrm{d}x = \frac{2n+1}{2L_0}\int_0^{2L_0} U(L)\,P_n((L{-}L_0)/L_o)\mathrm{d}L$$

The integrals can now be approximated by sums over all the individual values of L_i and U_i that occur.

$$u_n \simeq \left(\frac{2n+1}{2L_0}\right)\left(\frac{2L_0}{N}\right)\sum_i U_i\,P_n((L_i - L_0)/L_0)$$

N is the number of pairs of values L_i and U_i.

The first three Legendre polynomials are $P_0(x) = 1$, $P_1(x) = x$ and $P_2(x) = \frac{1}{2}(3x^2 - 1)$ giving the results

$$u_0 = \frac{1}{N}\sum U_i \quad u_1 = \frac{3}{N}\sum U_i L_i' \quad u_2 = \frac{5}{2N}\left[3\sum U_i L_i'^2 - \sum U_i\right]$$

where $L_i' = (L_i - L_0)/L_0$.

There is one slight complication; the minimum can be achieved by making the image black so all values of L^* are zero. This is overcome by adding $(\overline{L}{-}L_0)^2$ as another term in the quantity to be minimised. This ensures that the average lightness is fixed close to L_0. The issue of the overall lightness of the restored image is discussed below, for the moment assume $L_0 = 50$ and its purpose is to stop the silly all-black restoration.

In visual terms this means that there should not be an overall preference for any colour at any intensity from dark to light. This is a very similar objective to the white balance used before, the difference however is that all colours are involved whereas the white balance considered pixels that were almost white.

The lightness distribution This is based on the number of colours (not pixels) that have a particular value of L, or the *probability* $p(L)$ that a colour will have lightness L. For simplicity we omit the $*$ on L. Suppose we map the values of L to new values L' using a function $L' = F(L)$. Consider two intervals of L and L' and make the numbers of colours the same so that the probability distributions of L and L' are related by$p(L)\,\mathrm{d}L = p(L')\,\mathrm{d}L'$. The probabilities of colours with a lightness L in the range 0 to L and with L' between 0 and L' are

$$\int_0^L p(L)\,\mathrm{d}L \qquad \text{and} \qquad \int_0^{L'} p(L')\,\mathrm{d}L'$$

Making these equal fixes the value of L'. We do not have the exact function $p(L)$ but we can approximate it using the numbers of colours in each L bin. We can choose any preferred function for $P(L')$. The current code uses an inverted parabola

$$p(L') = (1/A)(L' - B_1)(100 + B_2 - L')$$

B_1 and B_2 are small constants and $A = 10^6/6+(10^4/2)(B2+B1)+100B1B2$ is chosen to make the total probability unity. The number of colours with lightness $< L'$ is got by integrating this distribution between limits of 0 and L'.

The equation fixing L' is then

$$\frac{1}{N}\sum_{l<L} n(l) = (-L'^3/3 + (100 + B2 - B1)L'^2/2 + B1(100 + B2)L')/A$$

where $n(l)$ is the number of colours with lightness l and N the total number. This is a cubic equation for L' and there are several ways to solve it. We use a highly robust binary chop so that the code could be modified to handle other preferred distributions. It is easily fast enough.

Saturation We use $S = \sqrt{U^{*2} + V^{*2}}/L^*$ as the definition of saturation and measurements of this for a range of restored images show that it often varies systematically with lightness. In accordance with our principle that the restored image should not be in any way special it seems best to adjust the saturation so there is no large systematic variation with L^*. Obviously the saturation of individual pixels varies a lot and all we can do is to reflect the general trend. We multiply the saturation by a factor $\alpha L^*+\beta$ and choose α and β to make the average scaled saturation close to a given constant$\bar{S}$, that is we minimise

$$\sum_{\text{colours}} [(\alpha L^* + \beta)S - \bar{S}]^2$$

with respect too α and β. This leads to two linear equations for the parameters in the factor. Currently $\bar{S} = 0.5$. Although the aim is generally to increase the saturation it may also decrease it.

White Balance This is a variant of the processing used in digital cameras. In the conversion of (R,G,B) values to L^*, U^*, V^*) a value is used for the 'white point' which fixes the colour with $U^* = V^* = 0$. The exact definition depends of the context in which colour is being discussed but that need not concern us here; a value is defined at the start of the program. This normally good enough but after the saturation enhancement a small adjustment for each individual image is warranted. A list is made of the pixels that are nearly white and have high lightness. In the current code the rules are $U^{*2} + V^{*"} < 200$ and $L^* > 80$ The average of these pixels is used to adjust the global value. Note that this is the only place where numbers of *pixels* is used rather than *colours*..