Yet More Restoration of Faded Slides

Geoff Daniell
gjd@lionhouse.plus.com

June 7, 2019

1 Background

For several 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. Later I produced stand-alone code in python that was independent of the `gimp` environment. This used the PIL image processing library in python. I have recently discovered that the code called `pyrestore2.py` no longer works correctly and this is due to changes in the PIL library, in particular the colour quantising procedure `Image.quantize()`. The similiar program `pyrestore3.py` uses my own colour quantisation algorithm and is unaffected.

It was also possible, with experience, to recognise pictures that had been through the restoration process, which proves that the result must be different to the original and there is room for a better method.

The previous document 'Restoration of Faded Slides' discusses the process of colour photography and possible models for the deterioration. The conclusion there was that it should be possible to reverse the deterioration using just the `gimp` `'levels'` command if the parameters are suitably chosen and ways of determining these values were suggested. I feel that this discussion is still valid but more robust ways of fixing the parameters are needed. These are the subject of this document.

2 Colour Quantisation

Colour quantisation was originally conceived as an method of image compression in which the three bytes used for colour information was replaced by one, although this is no longer particularly important because of increases in disk space and machine speed. Its relevance to my work follows from a consideration of the processes by which a photograph could deteriorate which are discussed in detail in a previous document (Restore1.pdf). A change in the coloured dyes in the emulsion affects *all* the colours in an image in ways that can be calculated and the range of colours present can be used to deduce that the photograph has *probably* deteriorated. If almost all the colours are blueish it is likely that this results from a chemical change in the emulsion. It is always possible that the original scene had an unusual range of colours and there is no way to prove this. On the other hand even a few pixels *not* showing a blue tinge proves that the colours are genuine and not the result of deterioration. The important point is that it is the *range of colours* present in the image that is important and not how many pixels share a particular colour. This is the reason why my previous algorithms used colour quantisation as a first step.

After investigating several published methods for colour quantisation in python it has become clear that it was a mistake to rely on this step. There are several algorithms for determining the 'best' set of N colours to represent an image but 'best' is usually not defined beforehand and the output from the code is simply result of applying a particular method. It is known that in some cases the results depend on the initial guess for the best colours. The list of colours is often in arbitrary order and even when sorted the 'best' set of colours can vary a lot when produced by different algorithms.

3 A Fresh Start

Most image processing packages include procedures for image enhancement, for example by stretching the contrast range or adjusting the maximum and minimum values in each colour channel. These effectively define a 'target image' which is assumed to be better than the original. All my earlier restoration attempts (Restore1.py, Restore[2,3].py and the related `gimp` plugins) use the optimum set of colours to determine how to move the image towards a 'target image' defined in some way.

In earlier work plots of the red, green and blue intensities in the optimum colours against index number were used. It was essentially an empirical discovery that these plots could be used to estimate the deterioration that had occured in each colour. In my first method the low and high intensites only were used to estimate the restoration parameters. In the second algorithm the whole curve was used and this is obviously better than relying on just two points on the curve. As a result of the fresh start proposed here an attempt was made to give a theoretical justification for the use of such plots. By analysing simple models and from numerical experiments I have concluded that, in spite of the fact that it works well, there is very little theoretical support for this approach. In particular it does not treat the three colour channels equally because they are not equally represented in the 'lightness' of the colour. However we will see below that some ideas used in the former approaches are useful and form the basis of the latest method.

The conclusions from the above discussion of colour quantisation are:

- The range of colours present in an image is still the only evidence for deterioration and must be used in restoration.

- The colour quantisation methods are unreliable and an alternative is needed.
- Because the results of quantisation cannot be used a new definition is required for the 'target image' which is hopefully closer to the original before deterioration.

3.1 A replacement for quantisation

One reason for using the PIL library (or the `gimp` indexed mode) is that the calculation is fast because the library is written in C code and not interpreted python. This need not be a problem as the Restore3 plugin and the pyrestore3.py program use my own quantisation code in python. However we now wish to steer clear of colour quantisation. The first idea to gain speed is to do the analysis on a small copy of the image and this has been used in all the previous approaches.

The simplest solution is to merely list all the separate colours present in the small image but this can be a lot of colours and subsequent calculations will be slow. It is however the *range of colours* that is important and not fine gradations of colour. We can therefore approximate each colour before making the list and this is easily done by using only the more significant bits of the eight bits of the intensty in each colour channel. Using the four most significant bits results in 4096 different colours and tests show that this list can be handled at an acceptable speed. Using five bit rather than four will produce 32768 colours and the subsequent processing will take eight times as long. It is worth stressing that speed is not important in processing old photographs; if ten seconds processing produces a better result than one second then let it take ten seconds, the calculation will only be done once.

3.2 A New 'Target Image'

In the resoration process we use the mathematical inverse of the presumed degradation process to move the image closer to a target image. We previously defined an 'ideal image' using the distribution of best fitting colours but for the reasons explained we now reject this definition. Let us list the features of what might ordinarily be called a 'good' photograph:

1. It should contain pixels with a wide range of colours. (Note that the *number* of pixels of any colour is not important.)
2. There should be a wide range of intensities from dark to light.
3. Things that should be white are indeed white; this is normally achieved automatically in digital cameras by the 'white balance' setting.
4. The colours should be reasonably saturated and not look 'washed out'.

Note that this excludes, sunsets, underwater shots, pictures exclusively of foliage and 'arty' shots. No automatic process is going to make a good job of restoring faded versions of these.

In my earlier restoration algorithms the distribution of best-fitting colours was used to achieve the first two of these properties. The third was implemented as a second pass through the restoration code and the saturation was optionally increased using operations in the PIL library. As an aside, this last step was a problem in the android version of the code because the PIL library for android was not available. As a result the calculation was very slow.

The 'white balance' step in earlier code is described in detail in the document Restore2.pdf and was done using the (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. As this space proved useful for the colour balance phase of the earlier restoration is seems a good idea to use it for the definition of the target image and to combine it with the concept of the 'ideal image'.

Figure 1 shows a faded slide, it is the same as the one used previously.

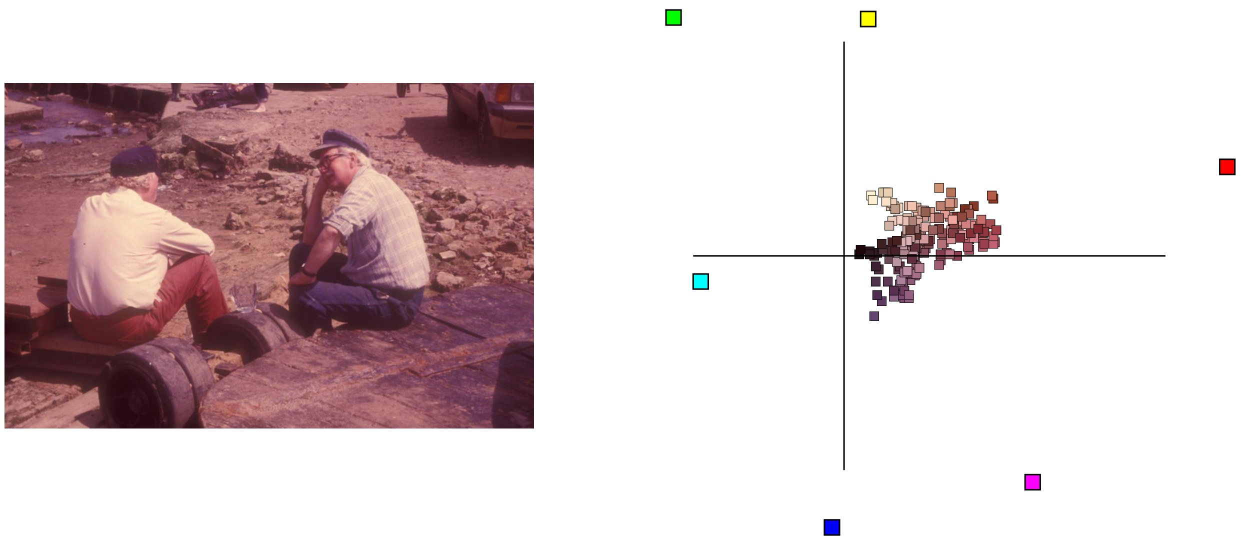

Figure 1: left: An old slide, right: The colours in the image

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. Later we will examine the distributions along this third axis. 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.

The 'ideal image' was previously defined using the plots of the three (R, G, B) colour intensities in the quantised set of colours against index number. Since the quantised colours are sorted according to their 'lightness' (either built into the software or by explicitly sorting the result) we could have looked at plots of the red, green and blue intensities against lightness instead of against index number. The curves are very similar except at the extreme low and high values of lightness. Now we can escape from the limitations of colour quantisation and simply plot the red, green and blue values against some measure of the 'lightness'. This leads naturally 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^*.

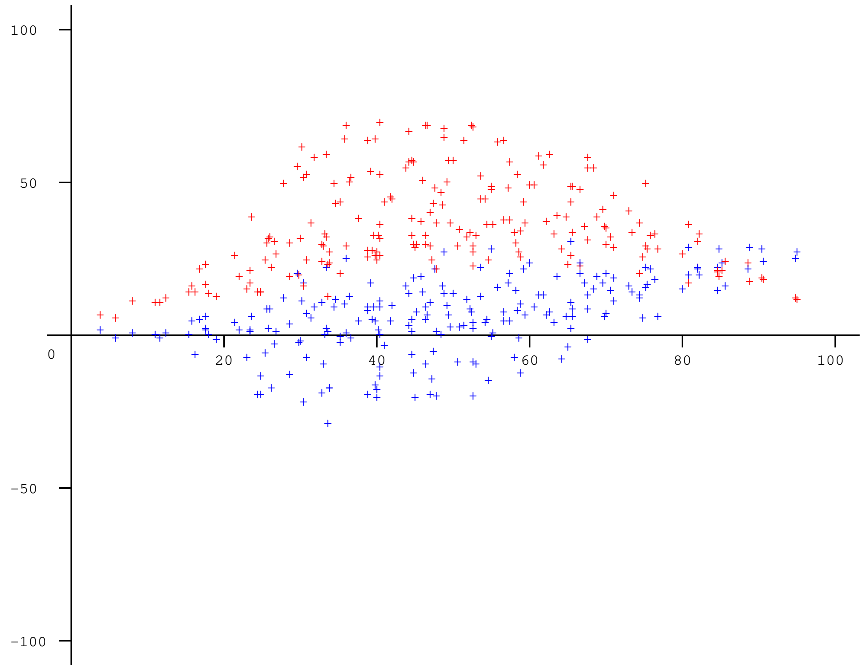

Figure 2: The colours U^* (red) and V^* (blue) plotted against L^*

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^*. The values of V^* also show some systematic variations. In the previous work the restoration was designed to move the image as close as possible to an 'ideal image' which was defined by the plots of red, green and blue intensities against approximate lightness. The ‘ideal image’ had the same curves for the three colours and a prescribed functional form. Now we can apply the same idea to U^* and V^*. The ‘ideal’ variation of U^* and V^* with L^* is there should be no systematic variation: the ‘ideal’ image has U^* and V^* scattered about zero at each value of L^*.

The observed variation of U^* with L^* has a large positive offset and a roughly parabolic form; the variation of V^* is more complicated and contains a linear component as well. Because of the approximate metric property of LUV space it is meangingful to define an average lightness denoted by $\overline{L^*}$, taken over all the colours occuring in the image. We choose the restoration parameters to remove the constant, linear and parabolic variations in the plots.

3.3 The New Algorithm

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 familliar 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 quandratic 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 plots of U^* and V^* after restoration, but before any further processing are shown here:

The symmetric distribution of U^* and V^* about zero, at all values of lightness L^* is clear.

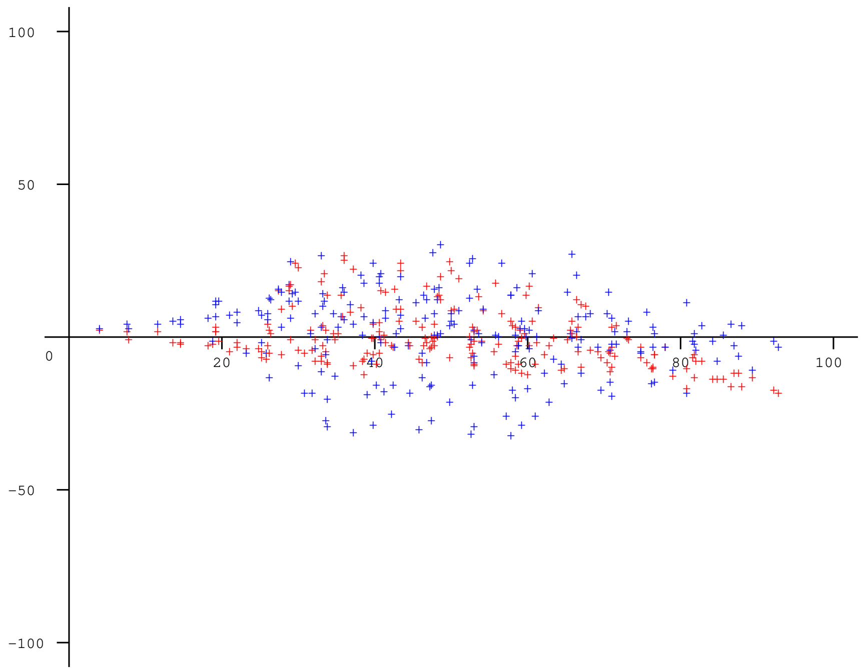

Figure 3: The colours U^* (red) and V^* (blue) plotted against L^*

4 White Balance, Scaling the Lightness and Saturation

The main restoration is done by determining the six parameters λ and σ for each colour channel. The computations are done on the small version of the image for speed. The list of colours in this restored image is then re-calculated since the restoration will not have been perfect. Previous algorithms followed the main restoration by further adjustments; in this section we discuss what additional adjustments might be appropriate in the new approach. The calculations are done using the new list of colours.

Pre-processing
It turns out that the iterative search for the restoration parameters sometimes fails to converge. This happens for slides that have gone very dark and the problem can usually be overcome by preliminary adjustment of the brightness and contrast. It seems worthwhile including a test for this and an automatic scaling which will work in most cases. The degree of scaling does affect the result, but not a lot.

White Balance
In the earlier algorithms working on the (R, G, B) values the simple restorati-

ion produced good results but an slight overall colour cast frequently remained which was removed by a second stage which improved the 'white balance' in a manner used in digital cameras. This used the U^* and V^* colour coordinates. Now that the main restoration is done in this colour space a separate correction to the white balnce is unecessary. It turns out that the definition of the 'white point' in the colour calculations is *very* critical and the values chosen have been determined empirically by looking at a selection of results.

Lightness

Item 2 in the above list of 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. Early experiments showed that some such operation was desirable and the Restore2.py and Restore3.py code included a rather arbitrary rule for deciding the maximum overall intensity in the image. In the new method a much simpler rule is possible: the lightness value L^* is scaled so that the minimum and maximum values are moved closer to 0 and 100. Since, as was explained above, the main restoration fixes the mean lightness to be 50 it is a good idea to leave this value unchanged so the light and dark parts of the lightness range are adjusted separately with linear scalings. Because the lightness is so obviously a visible property of the image the maximum and minimum values are computed using the actual pixel values and not the separate colours.

Saturation

Item 4 in the list of desirable properties is to require reasonably saturated colours. Several definitions of saturation are in use in colour science; the one used previously in this work is defined by the HSV colour space where the letters stand for 'Hue', Saturation' and 'Value'. Let h, m, l denote the highest, middle and lowest values in the triplet of numbers r, g, b describing the colour of a pixel. The saturation S is now defined as $(h - l)/h$. This is very simple but can result in rather unnatural colours.

A more refined definition can be constructed using L^*, U^*, V^*. Since $U^* = 0$, $V^* = 0$ corresponds to white or grey the saturation can be defined as how far a point is from here relative to its lightness. The definition used is $S = (U^{*2} + V^{*2})^{1/2}/L^*$. The average of the values of S taken over the whole (small) image is computed and U^* and V^* are scaled so as to raise this value to a threshold value if it is lower.

Putting it all together

The steps in the code are as follows:

1. There is a check on maximum intensity and an overall scaling to correct overly dark images.
2. The main restoration parameters are found using the list of colours in a small copy of the faded image.

3. This small faded image is restored using these parameters and a new list of colours produced.
4. The maximum and minimum lightness and the average saturation are computed from the pixels of the restored small image..
5. The constants for the linear scalings of L^* are computed.
6. If the saturation is low then a scale factor to be applied to all values of U^* and V^* is computed.
7. The original full size image is restored using the restoration parameters.
8. 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. The following figure shows the final image and the spread of colours in the U^*, V^* plane. Note that it is now central and enlarged compared with the original

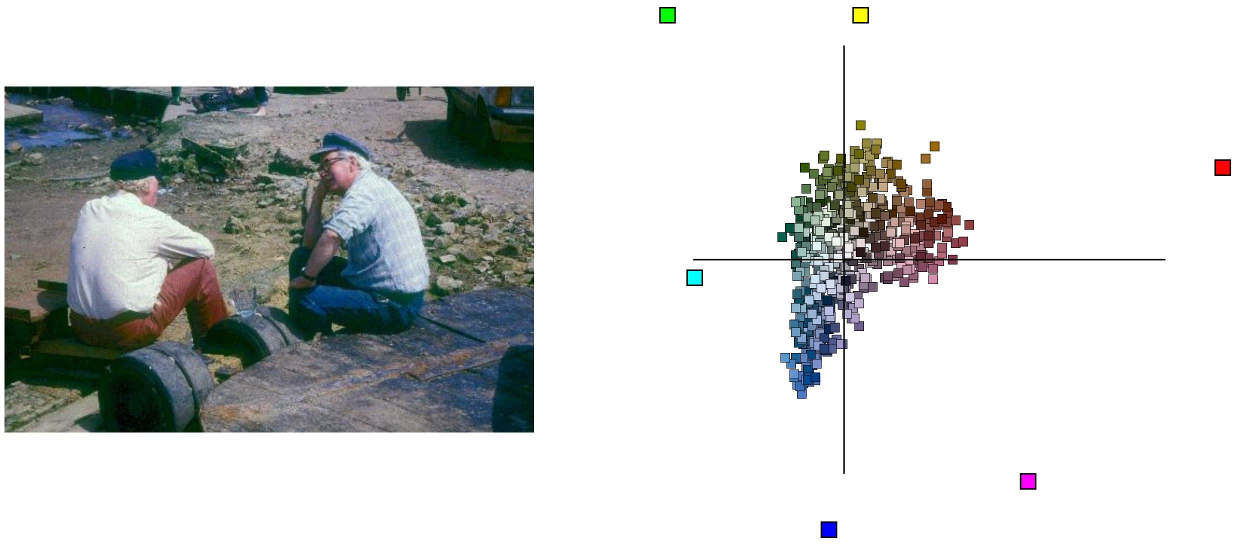

Figure 4: left: The restored image, right: The better spread of colours

5 Technical details

The code is written in python 2.7 but a python 3 version should be easy to produce. 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. Nevertheless I have compared the new algorithm with the output of `pyrestore3.py` for the same set of test images and a selection is shown below.

As might be expected the results are *different* but in general 'good' and since we do not have the original this is all we can ask. Almost none of the results with the new algorithm are worse than with the older ones. My impression is that in general the colours are better with the new method. In particular the chocolate browns, greenish skies and violet shadows which were charcteristic of the earler algorithms seem absent. Since these colours are mixtures of red and green, green and blue, and blue and red I had attributed their prevelence to be in some way a consequence of processing using the red, green and blue channels. The new algorithm using LUV space ought to be better and this seems to be true.

In some cases the results from `pyrestore3.py` are more 'colourful' than from the new code. I attribute this to the better definition of saturation that is now used; the more 'colourful' images may be more attractive but I suspect they are less authentic.

Figure 5 shows an example used previously to illustrate a poor result using `pyrestrore3.py`. See the document **Restore2.pdf** for a discussion of why this image presents problems. The result now better but still not as good as can be achieved by hand processing. The distributions of colours before and after in U-V space are shown below the images.

The examples below these compare the original and the restorations using `pyrestore3.py` and the new `pyrestore4.py`. They have been chosen to emphasise ways in which the new algorithm works better than the old.

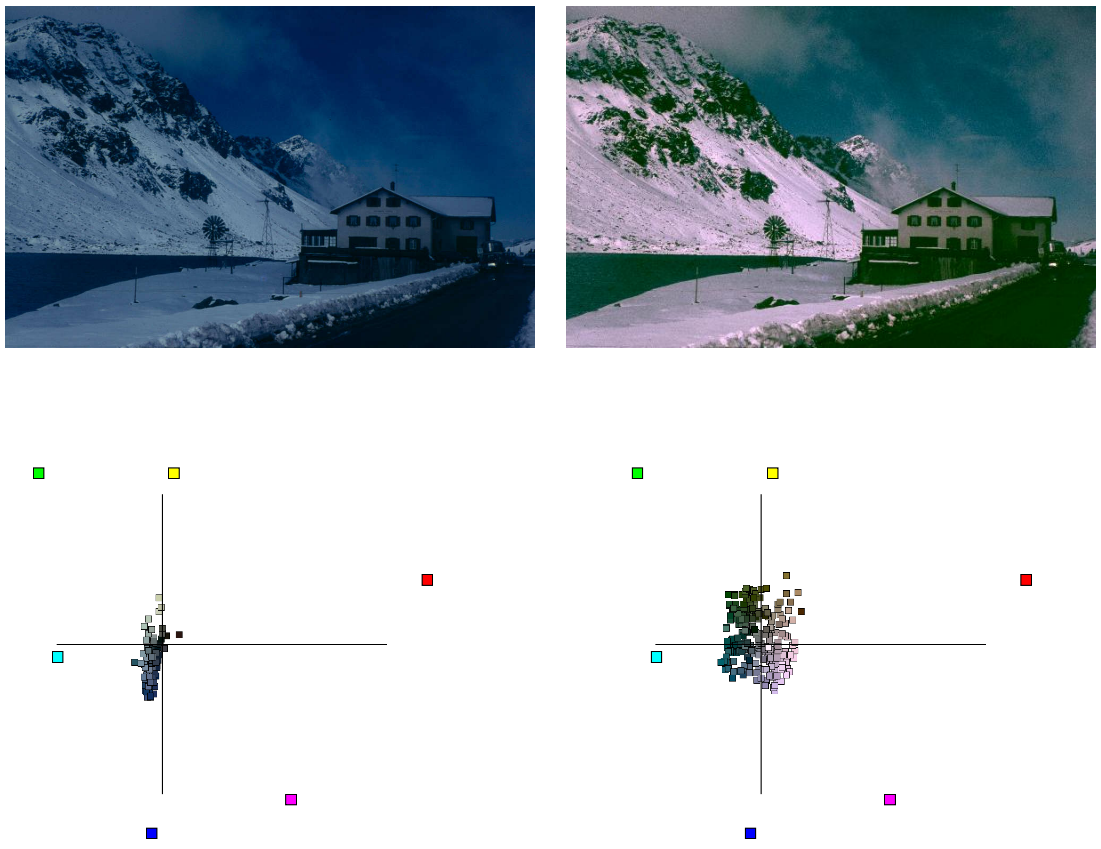

Figure 5: left: Original, right: Restored, Below the corresponding colour distrubutions

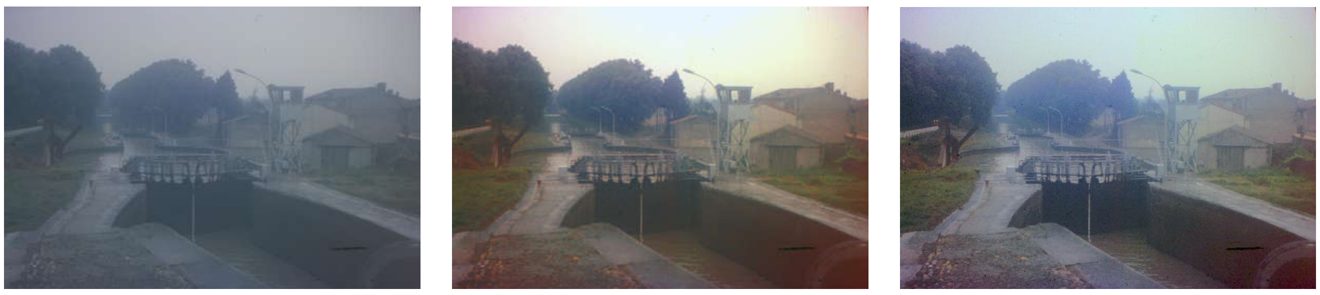

Figure 6: left: Original, centre: Previous restoration, right: New restoration

The first example, Figure 6, is a very badly faded slide taken on cheap colour film and both attempts at restoration are improvements. The result using `pyrestore3.py` has warmer colours but I suspect the new version is more authentic, (it was poring with rain at the time!). Note the purple tinge to the sky in the earlier version.

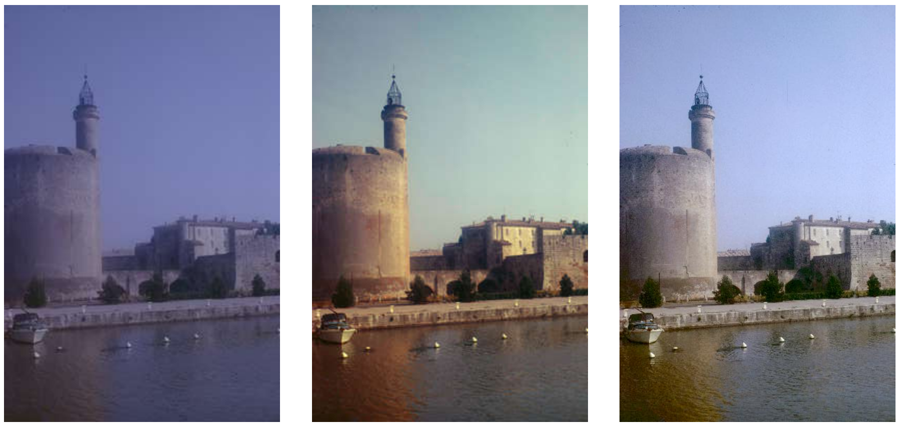

Figure 7: left: Original, centre: Previous restoration, right: New restoration

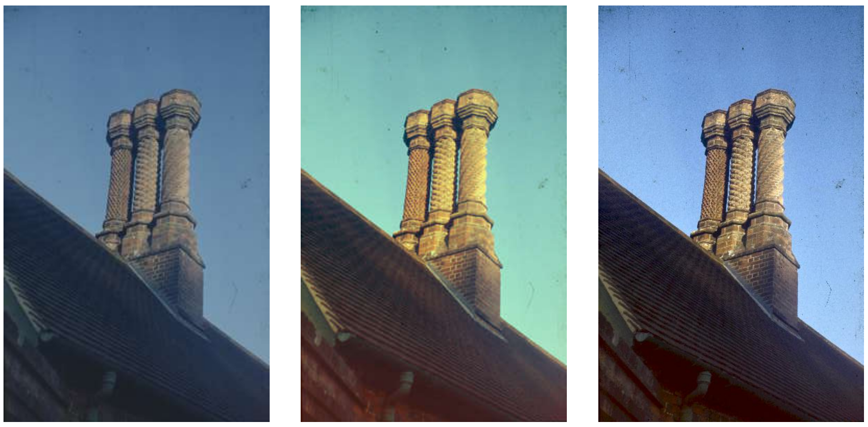

Figure 8: left: Original, centre: Previous restoration, right: New restoration

Figures 7 and 8 are examples of the tendency of the earlier method to produce chocolate brown colours and greenish skies and although the new version in Figure 7 is not very good the problem is much reduced. Figure 9 is an even more dramatic example.

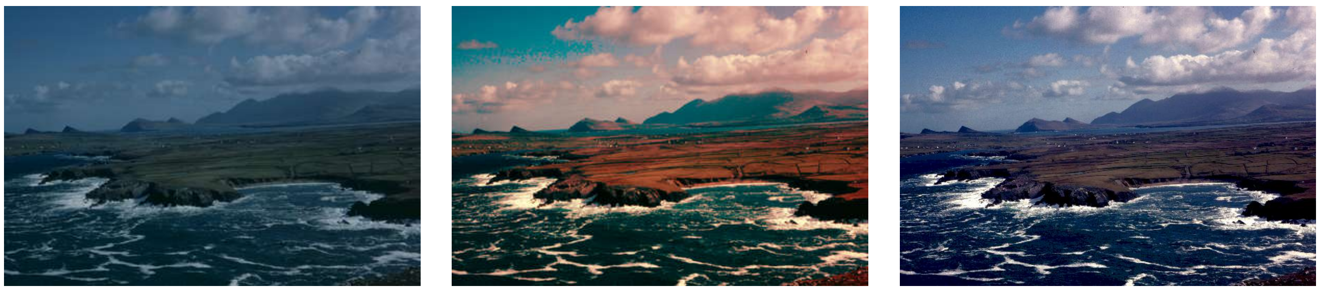

Figure 9: left: Original, centre: Previous restoration, right: New restoration

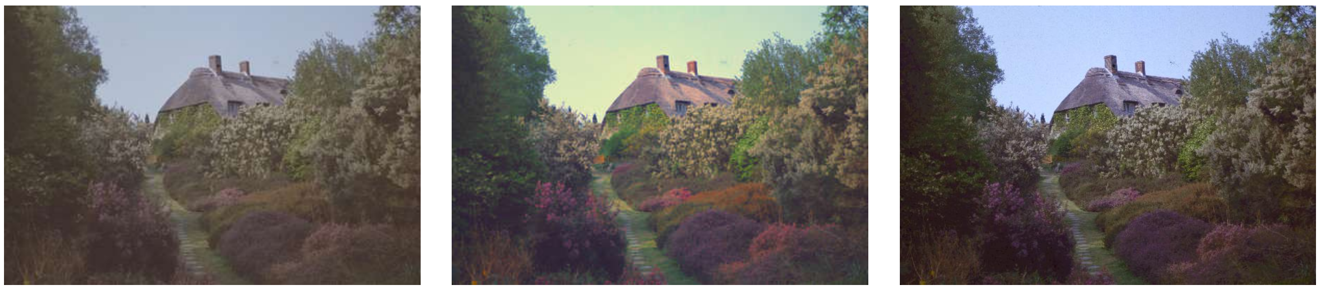

Figure 10: left: Original, centre: Previous restoration, right: New restoration

Figure 10 shows a case where the result using the original method is more 'colourful' and many people would prefer it. However I suspect this is the result of using the simple definition of saturation and not the better one using LUV colour space. Most software packages for image processing will include an option for increasing saturation but these will probably use the simple definition described in section 4 since this is much easier to implement. As we are exposed to more images that have been digitally processed, on TV for example, we will begin to expect bright colours even if they are less natural.

7 Examples

In this section, Figures 11 to 14, I show a few more examples; I have, of course, chosen cases where the the method works fairly well.

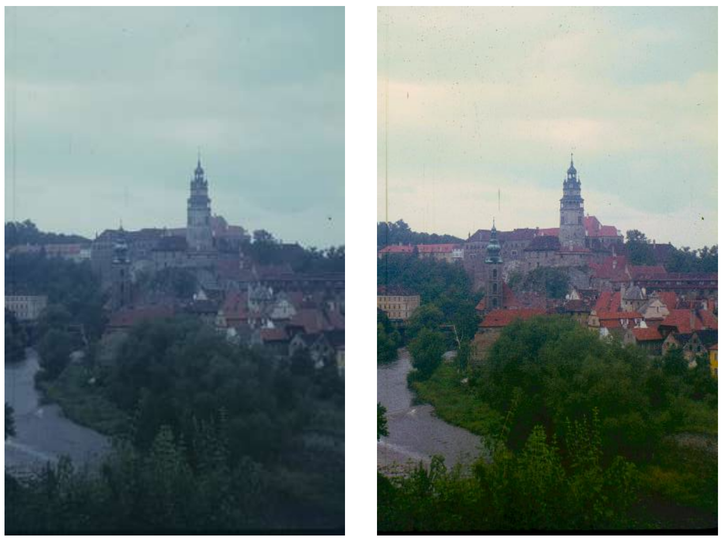

Figure 11: left: Original, right: Restored

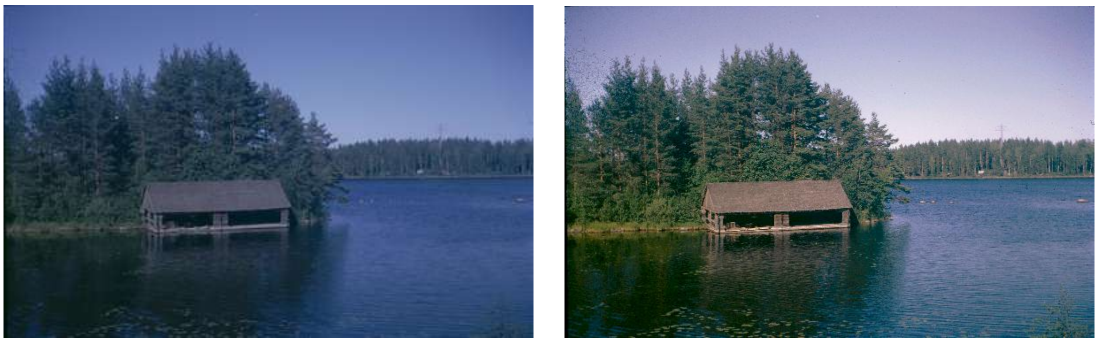

Figure 12: left: Original, right: Restored

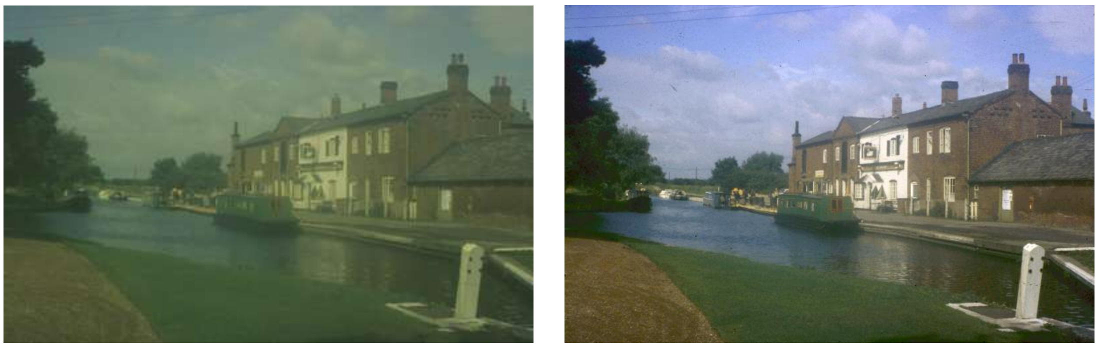

Figure 13: left: Original, right: Restored

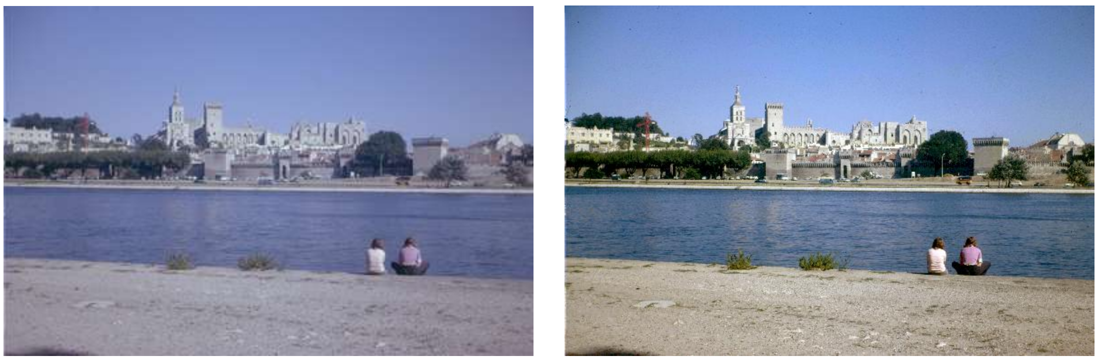

Figure 14: left: Original, right: Restored

8 Conclusions

Previous documents have described computer code for restoring colour photographs that have deteriorated with age. `Restore1.py` was a plug-in for the `gimp` image processing package. An improved algorithm was used in the plug-ins `Restore2.py` and `Restore3.py`. Since the operations used in the `gimp` package were also available in the PIL library stand-alone python programs `pyrestor2.py` and `pyrestore3.py` were inroduced. The first of these no longer works in many cases and is abandoned.

The latest code `pyrestore4.py` used a new approach which treats colour information in a way that is better match with human vision rather than the convenience of computer processing. The results appear to be significantly better but one should always remember that the original is not available for comparison. In addition to the tests included here there is the argument that the calculations in colour space have a much more solid theoretical basis.

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 enirely automatic and there are no options for the user.