Colour Matching Between Stereo Pairs of Images

Stephen Willey*†, Phil Willis*, Jeff Clifford†, Ted Waine†

*University of Bath, †Double Negative

Abstract. This paper outlines the process of colour matching stereo pairs of images using a disparity map as input along with the original images. We describe the functionality of a plugin developed for Nuke, which we call Eyematch, that performs this automatic colour matching under conditions set by the user. The user is presented with various parameters to fine tune the matching process, but no prior knowledge of any of the underlying techniques is necessary. Results are produced quickly, allowing a trial-and-error based approach to fine tuning these parameters, and results are sufficiently accurate to be used in the post-production pipeline.

1 Introduction

3D movies have become much more prevalent in the last few years, almost becoming the standard format for any major release, with five out of the top ten grossing movies of 2012[1], and eight of the top ten for 2013[2], being released in the format. There has, however, been a backlash against the format which is due, in no small part, to the many examples of poorly executed, or seemingly unnecessary, 2D to 3D conversions that offer an uncomfortable viewing experience that can easily ruin an otherwise perfectly good movie.

The beam splitter rig is the most commonly used technique in production for capturing native 3D footage. This involves having one camera facing forward and the other facing perpendicular to it with a semi-reflective plane between them which will allow half of the light to reach one camera and half to reach the other. There are some major problems associated with this approach to stereo image capture, as can be seen in the results (section 5). Firstly, the alignment of the images can be incorrect and inconsistent. This is due to the fact that the beamsplitter rig is rather large and any movement of it will result in each camera moving independently ever so slightly, which is enough to cause big problems in the post-production pipeline. Secondly, the use of two separate cameras and the introduction of a semi-reflective screen causes some noticeable colour differences between the images.

These colour differences are caused by multiple factors, the three main ones being polarisation, hardware, and angle. The semi-reflective screen introduced

[1] Box Office Mojo 2012 Worldwide Grosses: `http://www.boxofficemojo.com/yearly/chart/?view2=worldwide&yr=2012`

[2] Box Office Mojo 2013 Worldwide Grosses: `http://www.boxofficemojo.com/yearly/chart/?view2=worldwide&yr=2013`

by the beamsplitter rig naturally causes some polarisation errors due to the fact that it selectively allows some light to pass through while reflecting the rest. If there is an area of a scene that is reflecting light at a particular polarisation, rather than a broad range of polarisations as is typical, it is possible for it to appear as a highlight in just one of the images. Another problem inherent in using two cameras is that, no matter how much care is taken to configure them exactly the same, there is always the possibility of a slight difference in hardware translating to some slight difference in the image recorded. Finally, the fact that each camera is recording the scene from a slightly different angle means that there is the chance of colour differences that would be perfectly natural in the real world, but would cause problems in a stereo projection (e.g. iridescence).

In this paper we discuss the use of a plugin, Eyematch, which we have created for The Foundry's digital compositing software Nuke [3], that attempts to correct the colour differences between stereo pairs of images. EyeMatch is implemented as a Nuke node in order to fit seamlessly into the post-production pipeline, since many artists are already familiar with Nuke and use it on a daily basis. This also has allowed us to take advantage of many of Nuke's existing features.

2 Related Work

The problem presented could be solved in many different ways. The most basic solution would be to take the colour palette of one image and apply it to the other through, for example, histogram reshaping [5]. Another option would be to first segment the image, then perform a more specialised colour match between each element as defined by the chosen segmentation algorithm. Alternatively, the desired result could be achieved on a per pixel basis through techniques such as optical flow [1]. The main factors to be considered in choosing an appropriate solution are speed, accuracy and scalability.

There are many methods for adjusting the whole colour palette of an image using another image as a reference [3][4] but it is clear in our case that this will not be enough, and a more sophisticated approach is necessary to make more localised adjustments rather than just making a global change.

HaCohen et al. [6] compute dense correspondences in order to calculate the necessary colour change to bring about some regularity within a collection of images. The actual changes made are applied globally to each image, however, in order to avoid many of the complications brought about by making local alterations. Due to the nature of the issues we face with the stereo problem, it will be necessary to make many local changes and a global solution will not be sufficient.

When dealing with colour adjustments over video sequences, rather than still images, it is important to apply some temporal smoothing, as highlighted by Bonneel et al. [7]. In this paper we see that sudden changes of scene content,

[3] The Foundry: Nuke 8.0. (2013) Software available at *www.thefoundry.co.uk/products/nuke-product-family/*

such as a new character walking into frame, can have a big effect on the colour adjustment algorithm and need to be accounted for.

Reinhard et al. [8] introduce the idea of converting between colour spaces in order to aid the transfer process. Due to observing that all three channels often require modification in RGB space, it is suggested that using one with less of a correlation between channels would be advantageous.

3 Motivation

3D films are popular among audiences because of the immersive experience they offer. They are popular with studios and content creators because they can command a higher premium for viewing. Due to this popularity, it is important to do it right. We aim to minimise the amount of work the viewer has to do to reconcile a stereo image by ensuring an accurate colour match between views. In this way, we hope to create a more pleasant experience for moviegoers.

If the colours are not quite matched up, or the alignment isn't quite right, then the viewer is going to have to work harder to make sense of the images in front of them and that can lead to fatigue and headaches [2]. There are many other factors that can lead to these problems, such as the projectors in a particular cinema not being set up correctly, leading to problems such as ghosting, but these issues are outside of our control, and so we focus on those that can be fixed during the post-production process.

4 Method

Having investigated the existing methods employed to alter colours within images, it became apparent that a hybrid of several suggestions would be necessary. A global colour palette shift [5] would not be appropriate for this problem due to the fact that large occluded areas are likely to exist between images and this will skew the palette of each image. Therefore, our primary focus has been on creating a colour matcher which will work on local areas in order to match colours in a more accurate and more reliable way.

Our method relies on an accurate disparity map to work properly, and for this we have some in-house tools based on the Lucas-Kanade method of optical flow. There are other methods, including commercial solutions such as Nuke's Ocula toolset, for disparity map generation but the important thing is that a reasonably accurate and reliable disparity map is available for the EyeMatch algorithm.

Due to the fact that Nuke is used heavily in the pipeline at Double Negative, EyeMatch was created as a node that makes use of many existing features in the core Nuke toolset. For example, the two images (or sequences) are taken, along with the disparity map data, as a single input and immediately processed using several Nuke plugins, such as "Blur", to reduce the effects of noise, and "iDistort", which uses the disparity map to shift the pixels in one view to the

position of their corresponding pixels in the other, before being fed into the core matching algorithm.

4.1 Algorithm breakdown

This algorithm takes as its input the left eye view, the right eye view, and the disparity map describing the relationship between the pixels in each. Figures 1 and 2 show the x and y disparity information for our example scene, while figures 3 and 4 show the input images themselves. The first step is to shift the pixels of one image by the values in the disparity map, so that both views are aligned, then break the images up according to the grid dimensions provided by the user.

Outliers are dealt with by the following process: The mean and variance are calculated for each colour channel and any values lying more than one standard deviation from the mean are changed to the mean value, this practically eliminates noise and produces a smoother result overall. For each colour channel of each pixel, values are then arranged in order from lowest to highest. Using the disparity information, the corresponding pixels are then found in the second view and the same process of ordering and removing outliers is completed.

Rather than comparing each pixel to its exact counterpart according to the disparity information, pixels are matched based on their position in their respective ordered lists. Again, this creates a smoother and more robust result since each pixel difference is minimised and it is less likely that there will be any extreme adjustment calculations. Once each colour value for each pixel in the grid square to be adjusted are divided by their counterpart in the grid square to be matched, the resulting values can then be reordered to their initial positions to give the necessary colour change for each individual pixel.

Once this is completed for every section of the image, we are presented with a colour difference map describing the exact per-pixel changes required to match the colours of one view to the other. This colour difference map can be use to match the colours by multiplying each pixel with its corresponding pixel in the view to be adjusted.

Due to the fact that the image is gridded before processing, at this stage there will often be some very distinct lines at the borders of these grid squares. The simple solution to this is to blur across these boundaries in order to create a smoother result. From an artist's point of view, it is better to have a smooth result overall even if this comes at the expense of accuracy to some parts of the image. This is the final stage for a single pair of images, but for a sequence we must finish with some temporal smoothing to avoid flickering on playback. This is accomplished with a standard temporal median filter provided by Nuke which produces the results we are after.

4.2 User defined parameters

The user interface shown in figure 5 shows the parameters that can be specified by the user in order to adapt the algorithm for different situations, the features

Fig. 1. Disparity map showing x-disparity values from left to right view

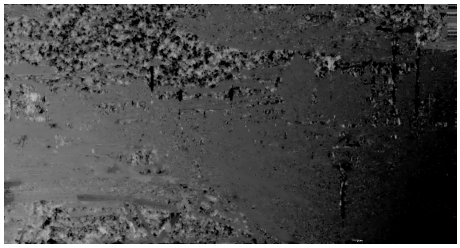

Fig. 2. Disparity map showing y-disparity values from left to right view

Fig. 3. Left view input image from a beamsplitter rig

Fig. 4. Right view input image from a beamsplitter rig

of which are described below. Once a user changes one of the parameters, it is applied immediately. Depending on the resolution of the images, the updated results will be displayed in the Nuke viewer in a matter of seconds.

Grid size. The grid size can be specified, and does not need to be defined as square since height and width can be adjusted independently. This is important since there is always a tradeoff between desired smoothness and required accuracy. A smaller grid size will result in a greater accuracy (as long as a good quality disparity map is provided as input) but may produce a less robust result as a smaller grid square will mean less pixels being looked at when deciding what constitutes an outlier. This means there is more chance of some noise sneaking through. Conversely, a larger grid size will result in a smoother result but could result in some errors around edges, especially those with large occlusions.

Blur. The user is also presented with two blur options. The first of these is for blurring the input disparity map, which can help when the disparity map is of a lower quality. This will have a negative impact on precision but is very helpful for smoothing out noise. Secondly, the user may specify the blur of the output. Some blurring is always necessary in order to lessen the obvious boundaries between grid squares, but sometimes more will be needed or less would be desired in order to get the necessary result. The user is also given the choice of a median

filter, which is appropriate for most tasks, or a slightly more sophisticated, but much slower, bilateral filter.

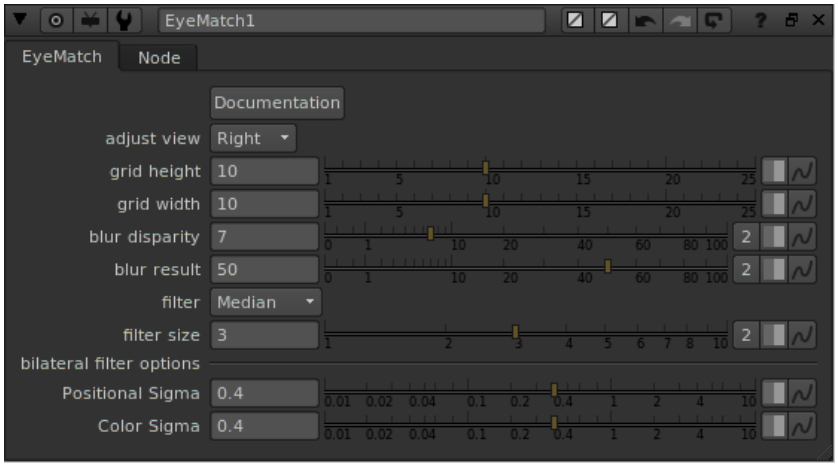

Fig. 5. The user interface showing default values as requested by users during development. Smaller values result in more locally specific changes, but require a more accurate disparity map; larger values will create a smoother result, at the expense of accuracy.

5 Results

We have produced some encouraging results with this algorithm and even made extensive use of it in the post production pipeline. There are some areas for improvement, but also some areas where the output produced is better than the leading commercial alternatives. The simplicity of this algorithm is its greatest strength and allows it to work robustly for large areas of a sequence, making it useful in spite of the errors which can be found in other areas of the output.

In order to better show the problem at hand, and to demonstrate the efficacy of our plugin, we have created checkerboard images with alternating squares from the left and right view. Figure 6 highlights the general problem well, showing not only the existence of colour differences between views, but also its non-uniform nature - some areas are already well matched. A closer look at some areas of the image (figures 7 and 8) further shows the extent that colour differences can reach.

Following the use of the EyeMatch node on default settings (as shown in figure 5), the result shown in figure 9 is produced. This colour matched right view is shown in alternating grid squares with the original left view in figure 10. Another look at the close up sections shown earlier reveals the accuracy of the result (figures 11 and 12).

Fig. 6. By viewing the left and right views in alternating grid squares of the same image, it is easy to see the colour differences present throughout.

Fig. 7. A closer look at a portion of the scene with a fairly uniform colour, with alternating squares taken from the left and right view, highlights the difference between views.

Fig. 8. Another close-up view, with alternating squares taken from the left and right views, further shows the colour differences that must be corrected in another portion of the scene.

Fig. 9. Output right view produced using default values for the Eyematch plugin, as shown in figure 5.

Fig. 10. Once the images have been run through our plugin, the colours are much more closely matched. This image shows the original left view in alternating squares with the colour corrected right view.

Fig. 11. The right view matches so closely that only by looking at the lamppost on the right is it clear that this image shows alternating squares from the original left view and the colour corrected right view.

Fig. 12. The road portion of the scene also shows a very close colour match when viewed as alternating squares of the original left view and the colour corrected right view.

6 Conclusion

We have presented an approach to correct the problem of colour mismatch when recording images on separate cameras and from slightly different viewpoints. Users can adjust parameters according to their requirements for a particular scene and see the results quickly, allowing them to select the best options on a trial-and-error basis if necessary. The user does not require any knowledge of any of the underlying techniques in order to use this tool, and acceptable results can be obtained quickly, and with very little effort.

7 Limitations and future work

This technique has been used in production and proved effective for many areas of images, though it is not usable as a complete package solution. It has been reported by artists to be particularly good for matching skin tones, which is an area some other solutions seem to struggle with. The main problem with this solution, however, is that it does not currently handle occlusions in any way and so when presented with particularly noticeable areas of occlusion it will produce poor results, as can be seen in figures 13 and 14. Potential ways to handle these occlusion errors include looking to the surrounding areas to interpolate a sensible value, or to look further along the sequence to see if the occluded area becomes visible. This temporal solution raises more issues, however, such as the chance of a lighting change between frames.

The Eyematch node relies heavily on having a good disparity map as input. Therefore future work, aside from better occlusion handling, will involve improving optical flow algorithms and introducing some occlusion detection at an early stage in order to better handle them further down the pipeline.

Fig. 13. Incorrect colour matching at the extreme edge of the image, caused by a lack of corresponding information in the left view.

Fig. 14. Incorrect colour matching due to the fact that the car occludes the wall too much, leaving EyeMatch without enough information to work with.

References

1. Lucas, Bruce D and Kanade, Takeo and others: An iterative image registration technique with an application to stereo vision. IJCAI **81** (1981) 674–679
2. Shibata, Takashi and Kim, Joohwan and Hoffman, David M. and Banks, Martin S.: The zone of comfort: Predicting visual discomfort with stereo displays. Journal of vision **11** (2011) 11
3. Pitié, François and Kokaram, Anil C and Dahyot, Rozenn: Automated colour grading using colour distribution transfer. Computer Vision and Image Understanding **107** (2007) 123–137
4. Grundland, Mark and Dodgson, Neil A: Color histogram specification by histogram warping. Electronic Imaging, International Society for Optics and Photonics (2005) 610–621
5. Pouli, T. and Reinhard, E.: Progressive histogram reshaping for creative color transfer and tone reproduction. Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering (2010) 81–90
6. HaCohen, Yoav and Shechtman, Eli and Goldman, Dan B and Lischinski, Dani: Optimizing color consistency in photo collections. ACM Transactions on Graphics (TOG) **32** (2013) 38
7. Bonneel, Nicolas and Sunkavalli, Kalyan and Paris, Sylvain and Pfister, Hanspeter: Example-based video color grading. ACM Transactions on Graphics **32** (2013) 2
8. Reinhard, E. and Adhikhmin, M. and Gooch, B. and Shirley, P.: Color transfer between images. Computer Graphics and Applications, IEEE **21** (2001) 34–41

8 Acknowledgments

We would like to thank Engineering and Physical Sciences Research Council for funding this research. Many thanks to Darren Cosker, Paul Hogbin, Oliver James and Alexandros Gouvatsos for useful feedback and guidance.