Color matching for stereoscopic cinema

Marcelo Bertalmío
Universitat Pompeu Fabra
Barcelona, Spain
marcelo.bertalmio@upf.edu

Stacey Levine
Duquesne University
Pittsburgh, PA, USA
sel@mathcs.duq.edu

ABSTRACT

In a stereo camera set-up for 3D cinema, much care is taken to ensure that the cameras are both the same model and use identical parameter values. But an imperfect adjustment, the use of beam splitters, as well as unavoidable differences in the characteristics of the cameras and lens systems, often cause visible differences in color among the two views. Many of these differences appear to be local and therefore cannot be fully removed by common color matching approaches, which are typically global methods. We propose a simple yet effective method for local color matching, that operates in three steps. First, one view is chosen as the target and morphed so that it becomes locally registered with the source view, the one whose colors will be modified. Next, color matching is performed on the source view by aligning each local histogram with the corresponding local histogram in the warped target view. Finally, Poisson editing [9] is applied on the source regions which have no correspondence in the target. For several professional, high quality examples of 3D sequences, we show that our algorithm outperforms four different conventional color matching approaches.

Categories and Subject Descriptors

I.4 [**IMAGE PROCESSING AND COMPUTER VISION**]: Enhancement, Applications

Keywords

Color matching, Stereoscopic cinema, Image warping, Local color transfer, Color propagation

1. INTRODUCTION

In professional 3D movie production it is crucial that the twin cameras used in a stereo set-up perform in exactly the same way. First, they have to be the same model, produced in the same year (even preferably with consecutive serial numbers), running the same software, with all presets reset to factory defaults. Second, all camera parameters must be matched: white balance, sensitivity, shutter angle, exposure, focus distance, focal length, frame rate, and gain [8]. Optics also must be paired, but due to manufacturing processes it is impossible for lens makers to produce two identical lenses. Therefore, discrepancies must be solved with production and post-production tools [8]. The twin cameras are mounted on a rig, and there are two possible configurations: "side-by-side" with a parallel rig, and "mirror" with a beam-splitter rig. The latter is very popular since close-up shots require bringing the camera interocular down to a few millimeters, and the large cameras and optics of professional productions make parallel layouts inadequate [8]. In the beam-splitter rig the cameras are placed at 90 degrees and a semi-transparent mirror (the beam splitter) reflects the scene into one camera while allowing the other camera to see through the mirror. Apart from physical considerations (the mirror is fragile, dust prone, reduces light by one f-stop and requires the image shot off the mirror to be flipped) the beam splitter produces color differences among the views, which again can only be dealt with in post-production [12]. Due to these artifacts from the mirror (typically in the right eye when shooting with a beam-splitter rig) the left eye image is usually considered the master image, and is generally used when creating 2D from 3D material [12].

From these considerations we can see that color differences among views are usual and often unavoidable, even in high-end productions: a threshold value of 3% is commonly used in the film industry as the maximum acceptable deviation in color. These problems are typically fixed in post-production by color-matching one view to the other, the master view, which is taken as reference.

2. RELATED WORK

In a recent work, Reinhard [13] surveys methods for color transfer among images, mentioning essentially four types of approaches: warping clusters or probability distributions in a three-dimensional color space, as in Pitié et al. [10]; matching means and standard deviations in an appropriate, de-correlated color space, as in Reinhard et al. [14]; aligning and scaling principal axes of the color distributions, as Kotera does in [7]; and straightforward histogram matching, performed independently on each color channel. In the experiments section we compare our proposed method with the four techniques just mentioned. These are all *global* methods, i.e. they modify the value of each pixel ignoring its location and hence also ignoring the values of the neighboring pixels. Recent methods use pixel correspondences to identify similar regions between the image pair and then refine the color matching transform which still is global, as in Kagarlitsky et al. [6]. Other local methods are for instance Tai et al. [16], where the images are segmented into regions in which color distributions are represented as Gaussian mixtures and then matched, and Huang et al. [5] where colors are transferred among corresponding landmarks and then extended to the rest of the image. While these methods provide very good results in a variety of situations, they are also much more involved than

global approaches and rely on solutions to challenging tasks such as image segmentation.

Our work is most closely related to two local color transfer approaches. HaCohen et al. [4] compute a dense correspondence map, then perform local histogram matching and finally fill-in missing regions with Poisson editing [9]. Bertalmío and Levine [2] perform fusion of exposure bracketing pairs transfering colors from a (warped) long-exposure blurry image to a short-exposure dark and noisy image, and for this purpose they propose an energy functional with a term that measures local histogram differences among an image pair: when minimized, this term ensures local histogram matching. These are recent works that provide excellent results in a variety of situations. They both rely on local histogram matching for the color transfer procedure, though, and we have found that local histogram shifting is better suited to our problem: we want a fast, one-shot method, and one-shot histogram matching may produce artifacts, as pointed out in [2]. These artifacts could be reduced by blending the results from overlapping neighborhoods, and they don't appear with the iterative procedure of [2], but this comes at the prize of a higher computational cost. Also, the correspondence map used by HaCohen et al. [4] has some limitations, including the difficulty of finding reliable correspondences in very large smooth regions, such as clear sky. This would imply that these very large regions should be color-corrected with Poisson blending, further increasing the computational cost of the procedure. For our problem, on the other hand, given that all image pairs are registered and have a small disparity, it is very easy to compute a dense and reliable matching map using a very fast stereo matching algorithm such as [3].

In this article we propose a very simple yet very effective method for local color matching, consisting of three steps, each of them inspired by successful related works. Firstly, the master view is morphed so that it becomes locally registered with the source view, as Bertalmío and Levine very recently proposed in [2]. Next, color matching is performed on the source view by aligning each local histogram with the corresponding one in the warped target view; we will see that this histogram alignment is simply the mean shifting of the image values proposed by Reinhard et al. [14], although in a different color space. Finally, the Poisson editing technique of Pérez et al. [9] is applied on the source regions which have no correspondence in the target, so that the corrected colors are propagated inside these regions, as done by HaCohen et al. [4]. Our contribution resides then in the combination of these existing methodologies so as to suit our particular problem, the simplicity of the overall approach (which has only one, very intuitive, parameter to set), its low computational cost, and the effectiveness of its results. This is demonstrated with examples from several professional, high quality 3D sequences that show our algorithm outperforms four different popular color matching approaches.

3. THE ALGORITHM

We will work in RGB and deal with each channel separately and in the same way. Because of this, in what follows we make a slight abuse of notation so that when we discuss images S and T we are referring to only one of their color channels. Likewise, histograms will always be a 1D histogram of a single color channel.

Our problem is to modify the source image S so that its colors locally match those of the target view T. We can do this in the following way: for each pixel $x \in S$,

- find its corresponding pixel $y \in T$: x and y are correspondents when they are the projections on S and T of the same 3D scene point;
- consider an $n \times n$ window $W_x \subset S$ centered at x, and find its corresponding region $W_y \subset T$;
- shift the histogram h_x of W_x so that it is aligned with the histogram h_y of W_y: this will give us a new value v' for $S(x)$;
- replace $S(x)$ with v'.
- in case we cannot find a correspondent y for x, often the case when x lies near a boundary of S which is not seen from the other view T, then $S(x)$ is not modified for now;
- after all possible pixels have been modified, their colors are propagated into the remaining, untouched pixels.

Each of these steps will now be described in detail, starting with a modification of the target image which will allow us to increase the computational efficiency of our method without changing its behavior.

3.1 Morphing the target image

Since each W_x is square, we can compute the histograms h_x of S very efficiently: we can have stored in memory the current local histogram h_x so that when we move from x to its neighbor x', one pixel to the right, to compute $h_{x'}$ we only need to update h_x by adding n new values (a new column on the right) and removing n old values (corresponding to the left column). That is, computing the histogram of an $n \times n$ region in S requires only $2n$ look-up operations, not n^2. But notice that while W_x is always a square window by construction, its corresponding region W_y in general *will not* be square, and this poses practical problems for the efficiency of the computation of the histograms of T, which now go from linear to polynomial complexity.

This problem can easily be solved by making the following observation: with the map of all correspondences $x \leftrightarrow y$ between S and T we can morph T so that it's locally registered with S, i.e. deform T so that each pixel $y \in T$ moves to the position x but keeping the value $T(y)$ intact. This gives us a morphed target image T_m and for each pixel $x \in S$ its corresponding pixel $y \in T_m$ will be at the same location: $y = x$. Therefore, W_y will also be square and now we can compute efficiently the local histograms both at S and T_m.

The idea of morphing one image before comparing histograms was very recently proposed by Bertalmío and Levine [2] in the continuous setting: they have an energy functional with a term that measures the difference between a long-exposure blurry image and a short-exposure dark and noisy image, and they need to introduce "motion compensation" into this term in order to correctly match the colors of the functional minimizer to the colors of the blurry image.

As for the morphing itself, we can use any technique that produces a dense map of correspondences between the two views. Given that they are stereo views there is abundant literature on the subject, see for instance the comprehensive evaluation of dense-stereo algorithms by Scharstein and Szeliski [15]. The final result of our method does not seem to be influenced by the particular choice of morphing technique: for all the examples in this paper we have

used the stereo matching algorithm of Cox et al. [3], but using a commercial package [1] we obtain virtually indistinguishable results despite the fact that the warped images are different, see fig. 1.

3.2 Aligning the histograms

In order to adapt the histogram h_x to the histogram h_y we have several options: straightforward histogram matching, dynamic time warping as Porikli does in [11], shifting so that some peak of h_x matches a corresponding peak of h_y, etc. We have tried the above solutions and finally decided upon the following one, for simplicity of computation and quality of results: treat the histograms as distributions of mass and shift h_x so that its barycenter matches the barycenter of h_y. The barycenter is a more robust measure than peak location, its computational cost is much smaller than that of dynamic time warping, and the subsequent shifting does not produce artifacts unlike what happens with histogram matching.

But we can also make the following observations:

- each histogram value $m_i = h_x(r_i)$ represents the number m_i of pixels in the window W_x that have the value r_i;
- the barycenter G of h_x is computed as:

  $G = \frac{1}{M}\sum_i m_i r_i$, where $M = \sum_i m_i$ is the total mass.

Therefore, G is just the average of all the window pixel values $S(q)$, $\forall q \in W_x$. Consequently, shifting h_x by matching barycenters simply amounts to matching the local means of image values in S and T_m. This is precisely the original approach for color transfer of Reinhard et al. [14], although they worked globally and in a different color space. Reinhard et al. used the variances as well to scale the values, but we have found that this is not necessary for our problem, where local color differences among stereo views of the same scene seem to be adequately modeled just by histogram shifts. Notice also that, although there might be some unmatched pixels in each window W_x, this isn't a problem since they can simply be ignored when matching barycenters.

As we mentioned in the previous section, a local histogram matching approach like the one used by HaCohen et al. [4] and Bertalmío and Levine [2] is problematic for a one-shot method like ours because it may produce visual artifacts. Fig. 2, left image, shows such problems on the flag, near the propeller on the right, on the central cartoon depicting a bomb, etc. In the same figure we can see that, while the result of Bertalmío and Levine [2] does not present those problems, this comes at the price of a significantly increased computational cost (a running time two orders of magnitude higher than the running time of our algorithm), given by the iterative procedure.

3.3 Propagating colors to unmatched pixels

The above steps can only be applied to pixels x of S for which we have found a corresponding pixel y in T. But in stereo views there are always pixels that cannot be matched, typically due to occlusions or due to their proximity to an image border which makes them visible in only one view. We consider that a pixel x is unmatched when the sum of absolute differences (SAD) between the $11x11$ blocks of S and T_m centered at x is larger than 10%. Fig. 1 shows in black in images (e) and (f) the set of unmatched pixels for two different morphing procedures.

Let Ω be the set of all pixels in S that cannot be matched, and $\partial\Omega$ the boundary of Ω. We have to color-match the pixels in Ω as well, otherwise our result would show visible color jumps at $\partial\Omega$. What we need then is to propagate the corrected colors of $\partial\Omega$ inside Ω, but maintaining all the image details and shapes of Ω intact. This problem was posed and successfully solved in the Poisson editing work of Pérez et al. [9], by finding a solution for the Poisson equation in Ω with boundary conditions at $\partial\Omega$. The last step of our algorithm then is to apply Poisson editing to Ω so that finally all values of S are corrected. This process is usually very fast given that the area of Ω normally amounts to a small percentage of the total area of S.

The method of Bertalmío and Levine [2] does not employ Poisson editing, and as a result it is somewhat sensitive to incorrect warping. The way to overcome this problem is to use a larger size for the local neighborhoods, but this might reduce the accuracy of the color transfer. See fig. 3: the left image has visible problems near the antennae on the background, and while the middle image doesn't show artifacts, its contrast and color saturation are a bit reduced, particularly on and around the cockpit.

4. EXAMPLES AND COMPARISONS

Figure 4 shows example results obtained with our algorithm on stereo pairs from several professional sequences. Color differences are more noticeable in some examples than others, but all of them have large regions where the color mismatch goes well above 3%, which as we mentioned is a threshold value commonly used in film post-production as the maximum acceptable deviation in color. Notice for instance the colors on the hippo's cheek, the sky in the train image, the background trees and the foreground pavement in the car image, or the paint and cabin reflections in the plane image. Furthermore, several color mismatches appear to be local, as in the hippo and the plane images. With our method we are able to reduce all color differences and make most of them unnoticeable. The only parameter for our algorithm is the size n of the side of the windows W_x. In all cases we have used the same value of $n = 41$ (corresponding to a window half-width of 20) because it appears as a good compromise between computational efficiency and good sampling of the local histogram. The image dimensions are 960×540 and the processing time is approximately 0.14 seconds per frame (i.e. 7FPS) on a system with the following specifications: Intel Core i7 - 3960X - 3.3 GHZ - 6 cores Sandy Bridge with 8 GB of RAM.

Figure 5 compares our method with four of the techniques mentioned in the introduction: Reinhard et al. [14], Pitié et al. [10], Kotera [7] and global histogram matching (recall artifacts caused by local histogram matching were demonstrated in figure 2). For each method we show details of the results obtained on two different pictures from fig. 4. As we said the literature on the subject is very extensive, but these four works appear to be representative of the sort of approaches that are prevalent in color transfer. We can see that our method compares favorably in all four cases, but we must point out that our algorithm is specifically adapted to the stereo color matching problem, while the other techniques are useful in many other situations (such as when the images are completely different, a case that cannot be handled with our approach).

Finally, figure 6 shows the results for twelve consecutives frames of a sequence. We can appreciate that our method produces temporally coherent results, which is of paramount importance given that our application is for cinema. This lack of any visible tempo-

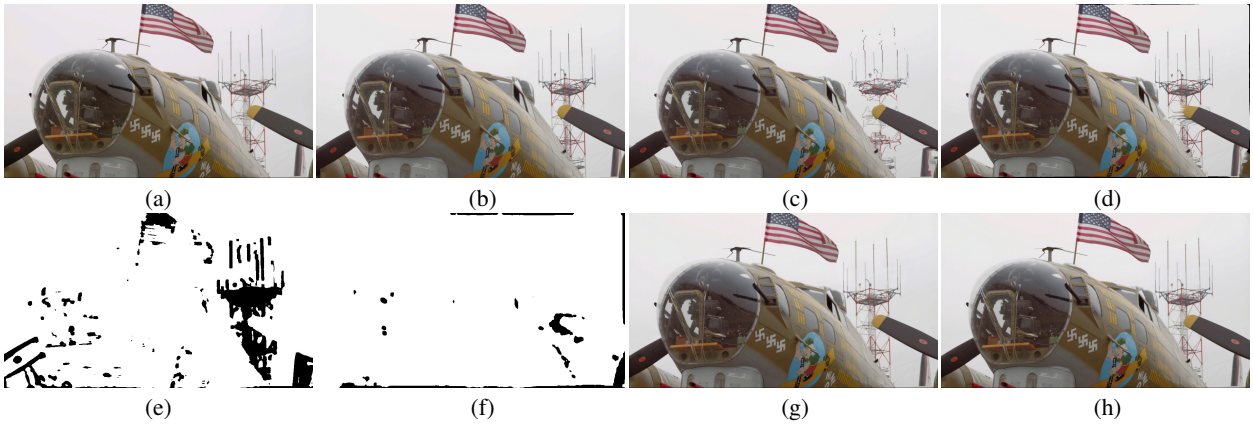

Figure 1: (a) Original left view, to be color-corrected. (b) Original right view, considered as reference. (c) Right view warped with [3]. (d) Right view warped with [1]. (e) Unmatched pixels in (c). (f) Unmatched pixels in (d). (g) Final result of proposed method, using (c). (h) Final result of proposed method, using (d). Original images provided by Mammoth HD Footage library (www.mammothhd.com).

Figure 2: Left: result of our method using local histogram matching. Middle: result from [2]. Right: our result.

Figure 3: Left: result from [2], neighborhood of radius 20. Middle: result from [2], neighborhood of radius 100. Right: our result.

ral artifacts is, we think, due to the fact that our method is not very dependent on a very precise stereo matching result (as shown in fig. 1), because the local histogram means are computed on neighborhoods of size $41 \times 41 = 1681$ pixels: small matching errors have therefore a negligible effect on the mean, as it is computed over many pixels.

5. CONCLUSION AND FUTURE WORK

We have proposed a simple yet effective method for color matching stereo views. The reference view is warped so that it is locally registered with the source, whose values are then modified by shifting local histograms. Pixels left untouched by this procedure are then corrected by propagating colors from the already processed pixels. The results are quite good and the method compares favorably with several widely used techniques.

A limitation of our method could arise from taking one view as reference; if there is a problem in this reference image (e.g. a highlight, blur, occlusion, etc.) it will affect the appearance of the colors in the other view. A possible solution would be for the user to mask these problematic regions so as not to perform color matching in them, and later apply Poisson editing as we already do for pixels appearing only in one view.

6. ACKNOWLEDGEMENTS

We would like to thank Pierre Jasmin of RE:Vision Effects Inc. for his unwavering support and his help running tests, and Clark Dunbar of Mammoth HD Inc. for supplying us with professional 3D footage. M. Bertalmío acknowledges support by European Research Council, Starting Grant ref. 306337, and Spanish grants

Figure 4: Left column: original left view, to be color-corrected. Right column: original right view, considered as reference. Middle column: our result. Original images provided by Mammoth HD Footage library.

AACC, ref. TIN2011-15954-E, and Plan Nacional, ref. TIN2012-38112. S. Levine is funded in part by NSF-DMS #0915219.

7. REFERENCES

[1] http://www.revisionfx.com/products/twixtor/.

[2] M. Bertalmío and S. Levine. Variational approach for the fusion of exposure bracketed pairs. *Image Processing, IEEE Transactions on*, 22(2):712–723, February 2013.

[3] I. Cox, S. Hingorani, S. Rao, and B. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, 1996.

[4] Y. HaCohen, E. Shechtman, D. Goldman, and D. Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM Transactions on Graphics (TOG)*, 30(4):70, 2011.

[5] T. Huang and H. Chen. Landmark-based sparse color representations for color transfer. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 199–204. IEEE, 2009.

[6] S. Kagarlitsky, Y. Moses, and Y. Hel-Or. Piecewise-consistent color mappings of images acquired under various conditions. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2311–2318. IEEE, 2009.

[7] H. Kotera. A scene-referred color transfer for pleasant imaging on display. In *Proceedings of IEEE ICIP 2005*, pages 5–8, 2005.

[8] B. Mendiburu. *3D movie making: stereoscopic digital cinema from script to screen*. Focal Press, 2009.

[9] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 313–318. ACM, 2003.

[10] F. Pitié, A. Kokaram, and R. Dahyot. Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding*, 107(1):123–137, 2007.

[11] F. Porikli. Inter-camera color calibration by correlation model function. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 2, pages II – 133–6 vol.3, sept. 2003.

[12] S. Reeve and J. Flock. Basic principles of stereoscopic 3D. BSKYB Whitepaper, 2012.

[13] E. Reinhard. Example-based image manipulation. In *6th*

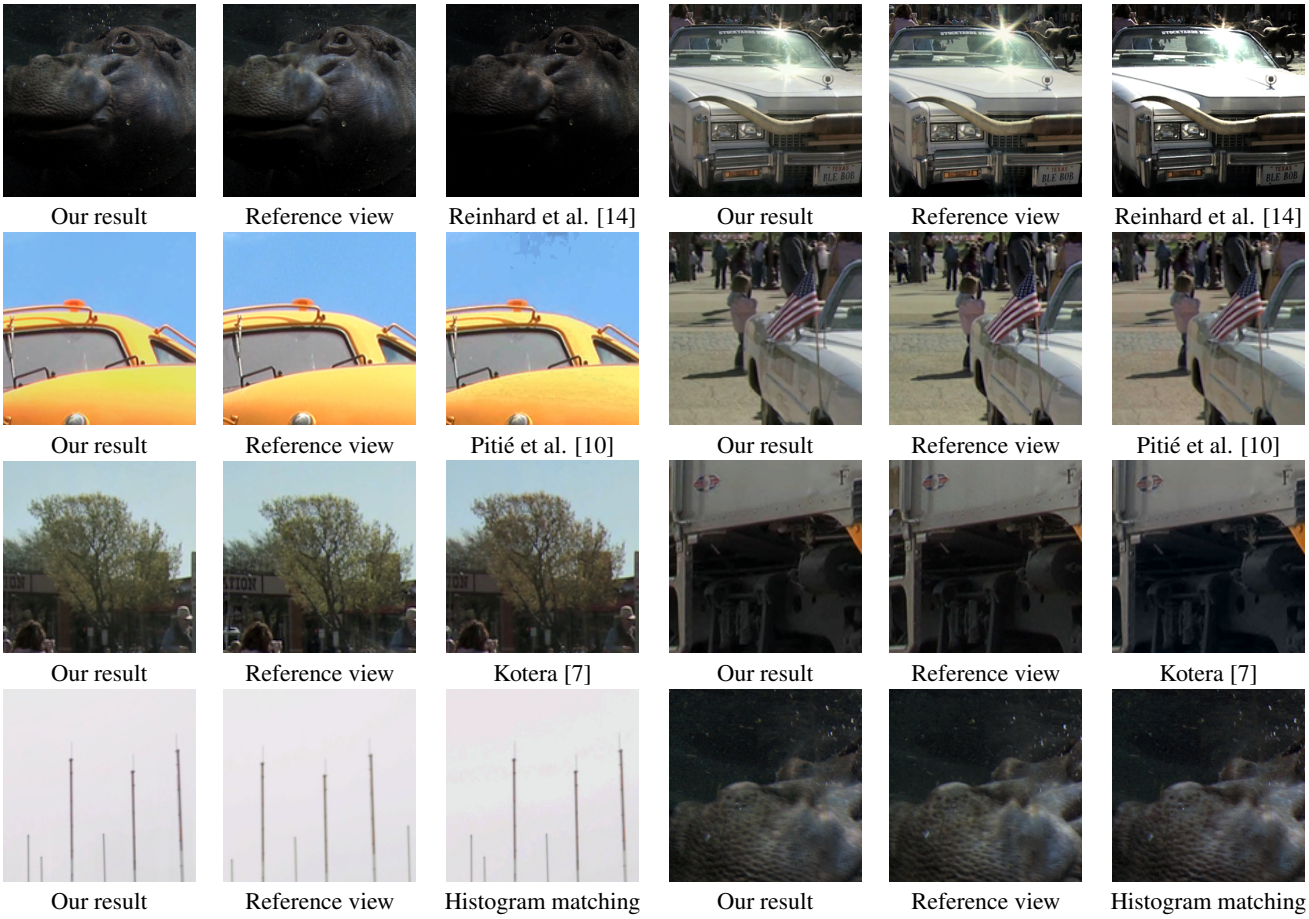

Figure 5: Details comparing the proposed method with the approaches of Reinhard et al. [14], Pitié et al. [10], Kotera [7] and histogram matching. For each method, results from two different pictures from fig. 4 are shown.

Figure 6: Result of our method on twelve consecutive frames of a sequence. Original images provided by Mammoth HD Footage library.

European Conference on Colour in Graphics, Imaging, and Vision (CGIV 2012), Amsterdam, May 2012.

[14] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *Computer Graphics and Applications, IEEE*, 21(5):34–41, 2001.

[15] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[16] Y. Tai, J. Jia, and C. Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In *Procc. of CVPR*, volume 1, pages 747–754. IEEE, 2005.