An Investigation into Graph Cut Parameter Optimisation for Image-Fusion Applications

Xiao Bao Clark, Jackson G. Finlay, Andrew J. Wilson, Keith L. J. Milburn, Minh Hoang Nguyen, Christof Lutteroth, Burkhard C. Wünsche

Department of Computer Science
University of Auckland, Auckland, New Zealand
{xcla001,jfin052,awil308,kmil102,hngu039}@aucklanduni.ac.nz
{lutteroth,burkhard}@cs.auckland.ac.nz

ABSTRACT

The graph cut technique has been employed successfully in a large number of computer graphics and computer vision related problems. The algorithm has yielded particularly impressive results in the field of image fusion, e.g., texture tiling, image stitching, and image and video editing. An analysis of the literature shows that authors use different variations of the algorithm, such as different cost functions and parameters. However, there are no detailed investigations on how these parameters influence results and what parameters are most suitable for what type of application. In this paper we analyse the use of graph cut algorithms in different image fusion applications. We list and classify relevant parameters, suggest new cost functions for seam optimisation, and analyse the effect of parameter choices on different application scenarios. Based on the results we develop guidelines assisting users to employ the graph cut technique effectively in different image fusion applications.

Categories and Subject Descriptors

I.3.7 [**Three-Dimensional Graphics and Realism**]: Color, shading, shadowing, and texture; I.3.3 [**Picture/Image Generation**]: Display algorithms; I.2.10 [**Vision and Scene Understanding**]: Texture; G.2.2 [**Graph Theory**]: Graph algorithms

Keywords

graph cut algorithm, image editing, texture synthesis, image stitching, image processing

1. INTRODUCTION

Graph cut techniques have become increasingly popular in the computer vision and computer graphics domain as an efficient means for solving optimisation problems [2]. This is achieved by representing the set of possible solutions as a directed graph with a single sink and source, where edge weights encode the cost of a solution. The graph cut algorithm enables the computation of a maximum-flow solution, which defines a minimum-cut (minimum cost) solution.

A particular successful application field for this algorithm has been *image fusion*, i.e., the seamless combination of (parts of) multiple images into a single image. Applications range from texture synthesis and transfer [7, 1], to image and video editing [7, 12], and image-based modelling [16, 17].

In this paper we categorise parameters for the graph cut algorithm, integrate appearance space attributes into the graph cut framework, and analyse parameters' effectiveness for different image-fusion applications. Section 2 will explain the problem domain in more detail. Section 3 reviews previous work in this field with an emphasis on parameter choices for image-fusion applications. Section 4 explains our experimental set-up with results summarised in section 5. We conclude this paper and give an outlook on future work in section 6.

2. THE GRAPH CUT ALGORITHM

In this paper we consider the most basic case of an image fusion problem, i.e., to have two overlapping images A and B, and to find the cut within the overlap region, which creates the best transition between these images [14]. The overlap region is represented as directed graph, where each node represents a pixel position p in the overlap region, which is denoted $A(p)$ and $B(p)$ for the two images A and B, respectively. Nodes are connected by edges representing connectivity between pixels. Usually 4-connectivity is assumed, i.e., each node (pixel) is connected to four neighbouring nodes (neighbouring pixels in the overlap region). Each edge is given a cost encoding the pixel differences between the two source images at that position. In the simplest case the cost w corresponds to the colour difference between the images

A and B at the neighbouring pixels p and q, i.e.,

$$w = w(p, q, A, B) = ||A(p) - B(p)|| + ||A(q) - B(q)|| \quad (1)$$

where $||.||$ is the L_2 norm.

The resulting graph is converted into a flow network by introducing a *source node* and *sink node*, which correspond to the parts of the images A and B, respectively, which lie outside the overlap region. The source and sink node are connected to the boundary pixels of the overlap region using edges with infinite cost. Finding the optimal cut through the overlap region is equivalent to finding the *maximum flow* in this network subject to not exceeding the edge constraints given by the cost function. The *minimum cut* can then be obtained from the edges with full capacity [14, 13].

A more general formulation of this problem uses the theory of Markov Random Fields and allows more complex cost functions (*energy models*) incorporating additional image-based constraints [2]. The max-flow/min-cut problem is NP-hard [7], but can be efficiently approximated using the α-expansion algorithm [3]. The algorithm can be further expanded to take into account more than two overlapping source images. In this case the seams of old cuts become new nodes in the graph [7].

In our investigation we consider only two source images and only cost functions expressing image differences across seams.

3. LITERATURE REVIEW

Efros and Freeman use seam optimisation for tiling textures [5]. The authors use the same general problem formulation as presented above and introduce the cost function in equation 1 to capture image colour differences along seams. However, the authors use dynamic programming to find a minimum cut.

Long and Mould [9] report that Efros and Freeman's approach can produce highly visible discontinuities when the path takes a shortcut through cost peaks (a short path with high edge costs can be cheaper than a long path with low edge costs). The authors propose instead a non-scalar distance metric which only takes into account the maximum cost of an edge, rather than the cumulative cost of all edges in a graph. Zou et al. subsequently proposed a graph cut technique using this metric [18], which requires a special solution procedure to find the minimum-cut. The authors report that the new cost function works best for textures with low frequency and highly localised features, e.g., ocean waves and woodgrain.

Kwatra et al. [7] report that seams are more noticeable in low-frequency regions, and a visually more pleasing cut is computed by increasing the cost of an edge with a decreasing image gradient. The authors modify the cost function w from equation 1 to

$$w_\nabla = \frac{w(p, q, A, B)}{||G_A^{pq}(p)|| + ||G_A^{pq}(q)|| + ||G_B^{pq}(p)|| + ||G_B^{pq}(q)||} \quad (2)$$

where $G_A^{pq}(p)$ is the image gradient in the direction of the edge pq at pixel p.

Agarwala et al. [1] present a technique for interactive digital photomontage, enabling the combination of interesting features from multiple images into a single photo. The authors use different cost functions for capturing seam discontinuities:

$$w_m(p, q, A, B) = \begin{cases} X & \text{for "colours"} \\ Y & \text{for "gradients"} \\ X + Y & \text{for "colours \& gradients"} \\ X/Z & \text{for "colours \& edges"} \end{cases} \quad (3)$$

where

$$\begin{aligned} X &= w(p, q, A, B) && (4) \\ Y &= ||\nabla A(p) - \nabla B(p)|| + ||\nabla A(q) - \nabla B(q)|| && (5) \\ Z &= E_A(p, q) + E_B(p, q) && (6) \end{aligned}$$

and $\nabla A(p)$ is a 6-component colour gradient (RGB) of image A at pixel p (derivatives in x and y-direction for all colour components), and $E_A(p, q)$ is the scalar edge potential between two neighbouring pixels p and q in image A, computed using a Sobel filter [1]. The authors mention that X/Z is only a semi-metric and hence the guaranteed performance of the α-expansion algorithm [3] is lost, but no problems were observed.

Ramanarayanan and Bala incorporate neighbourhood information into the graph cut cost function and demonstrate that this can be used for a constrained texture synthesis [11].

Most image-fusion applications seem to use the RGB colour space for calculating colour distances. However, the choice of colour space can make a significant difference as illustrated by Kulkarni and Nicolls, who experimented with seven different colour models for performing image segmentation using graph cuts [6]. The authors report that *Luv* values work better in areas of high brightness and MR8 filtering improves segmentation in textured regions.

4. EXPERIMENTAL SETUP

4.1 Graph Cut Algorithm Parameters

Our analysis of the literature suggests that the choice of cost function, distance metric, neighbourhood information, and colour space can make a significant difference on the location of the optimal cut. We categorise these parameters as follows:

Cost functions: The cost function always has the form $w(p, q, A, B)$, but can differ in the pixel information and distance metrics employed. We investigate the following parameters:

- Distance metrics: L_2 norm, L_1 norm (Manhattan distance), L_∞ (supremum norm).
- Non-linear distance measures: In order to penalise high edge costs relative to long cut paths, we replace the L_2 norm $||.||_2$ with $||.||_2^n$, where $n = 2, \ldots, 9$. We denote the resulting semi-norms with $L^2, \ldots, L^9$.
- Pixel measures: We use as pixel measures colour distance (equation 1), gradient weighted colour distance (equation 2), gradient (equation 3, case 2), and colour distance + gradient (equation 3, case 3). In addition we propose a modification of equation 2, which uses for each component of the colour difference the gradient of that colour component only as weighting factor. The resulting 3-vector for each term in the expression below is transformed into a scalar by computing its norm

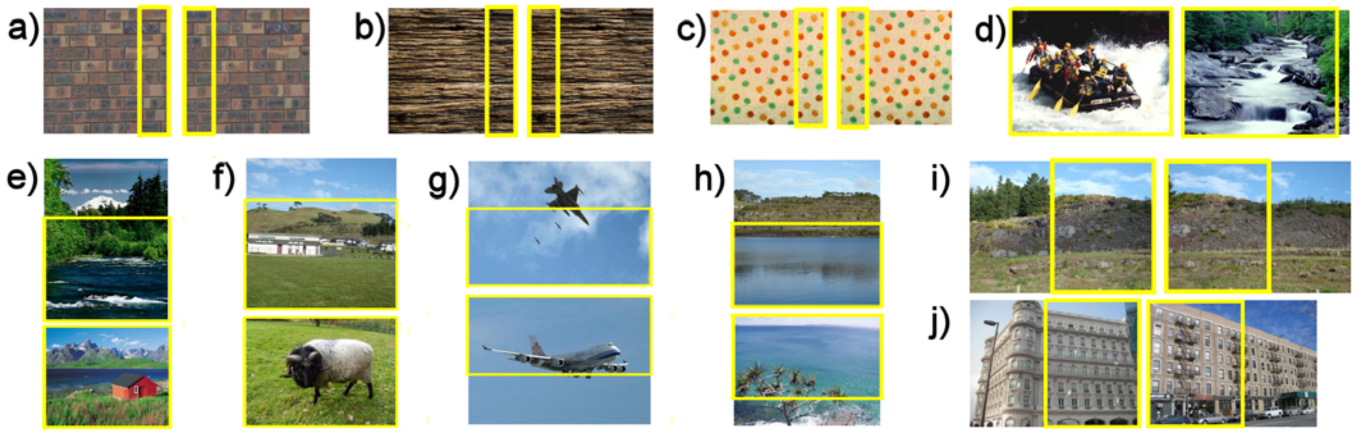

Figure 1: Examples of image pairs used in our experiments: image quilting examples (a-c), image editing examples (d-h,j) and an image stitching example for creating a panorama (i). The overlapping regions for the graph cut algorithm are indicated in yellow.

(default is L_2):

$$w_{\nabla 2} = \left\| \frac{|A_{C_i}(p) - B_{C_i}(p)|}{|G^{pq}_{A_{C_i}}(p)| + |G^{pq}_{B_{C_i}}(p)|} \right\| + \left\| \frac{|A_{C_i}(q) - B_{C_i}(q)|}{|G^{pq}_{A_{C_i}}(q)| + |G^{pq}_{B_{C_i}}(q)|} \right\| \quad (7)$$

where (C_1, C_2, C_3) are the three components of the utilised colour space (e.g., RGB).

Image Representation: Most articles on image-fusion techniques do not mention the colour space employed, but several authors report using the RGB space. As indicated in the literature review the choice of colour space can significantly affect results. We hence investigate several colour spaces which have been successfully employed in related graphics/vision applications [4]. In order to incorporate a wider measure for the visual appearance of a texture we also investigate appearance space attributes [8, 10]:

- Colour Spaces: RGB, HSV, $CIELuv$, $CIELab$, Perception-based Colour Space [4].
- Appearance spaces: Signed feature distance [10] (to obtain consistent spatial distribution of features) and edge information [Canny edge detector] (to achieve alignment of features).

Graph Representation: Many of the reviewed articles optimise parameters related to the graph representation, e.g., the size of overlapping regions and placement of images with respect to each other. All reviewed articles seem to assume 4-connectivity, but a graph for 8-connectivity could also be constructed. An important and interesting problem is the incorporation of old seams when performing a sequence of image fusion operations. We do not consider these issues in our experiments.

4.2 Test Cases

In order to determine the effect of different parameters we selected the following application scenarios commonly encountered in research and practical applications:

1. **Image stitching:** combining overlapping photographs of a panorama. We perform a manual placement and do not consider issues such as distortion correction.
2. **Texture quilting:** Similar to [5] we use two overlapping copies of an exemplar texture.
3. **Image editing:** Similar to [7] we want to insert one image into another unrelated image and create a realistic scene.

In order to investigate the effect of different parameter choices we use images with the following characteristics:

1. Images with self-similarity suitable for tiling (e.g., from Efros and Freeman's paper [5, 9])
2. Images with smooth low frequency variations (e.g., landscape photos) [7]
3. Images with high frequency variations (e.g., sand, marble, rough water surface) [7]
4. Images with noise (e.g., old grainy photos or noise added with Photoshop) [14]
5. Images with low frequency and highly localised features (e.g., ocean waves and woodgrain) [18]
6. Images with regular spatially distributed features (e.g., random dots with similar gaps, brick pattern, stones/fruits with similar size) [10]

We created 30 pairs of images covering the application scenarios and types of images listed above. A subset of our test cases is displayed in figure 1.

4.3 Experimental Setup

We implemented our framework for testing graph cut algorithms using C/C++. We used OpenCV for image processing and I/O operations. The minimum cut was determined using the Planar Graph Cut library [13]. For image quilting and image stitching examples we overlapped image pairs by $25\% - 50\%$, whereas for image editing examples we used a larger overlap. The overlap for the image pairs in figure 1 is visualised using yellow boxes. The minimum cuts were computed fully automatically (i.e., without manually inserting constrained pixels as in [7]).

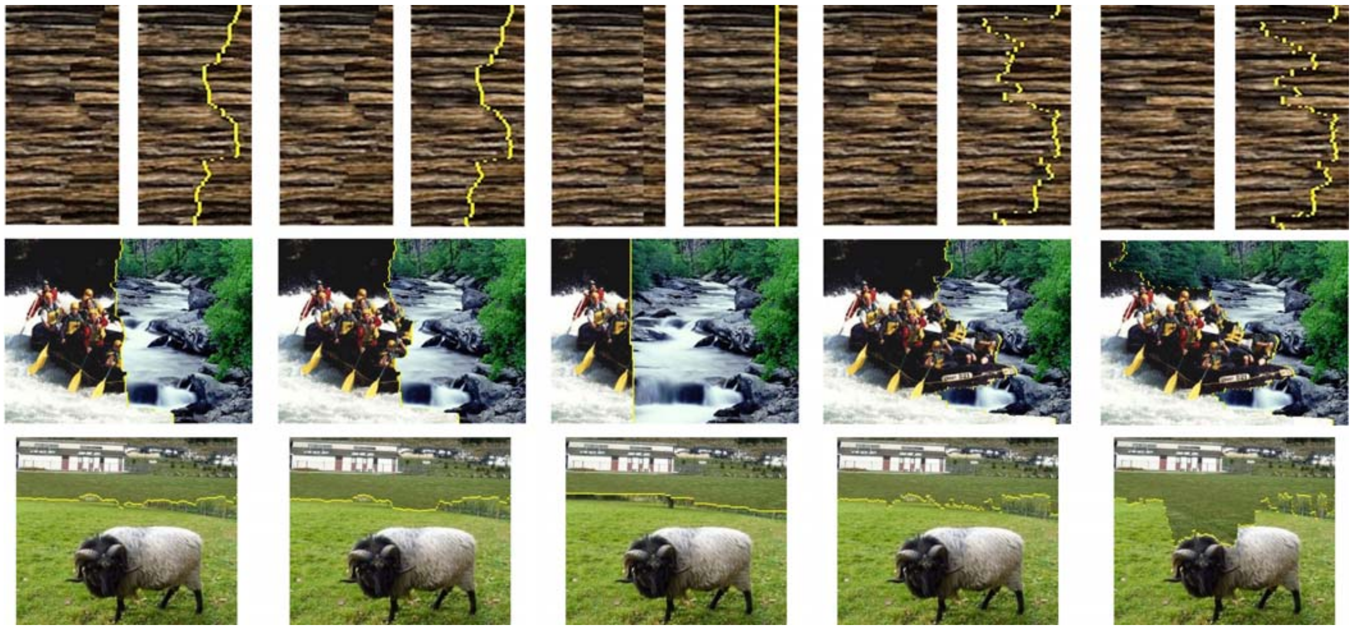

Figure 2: Examples of the results obtained using different norms. From left to right: L_2, L_1, L_∞, L^2 and L^3.

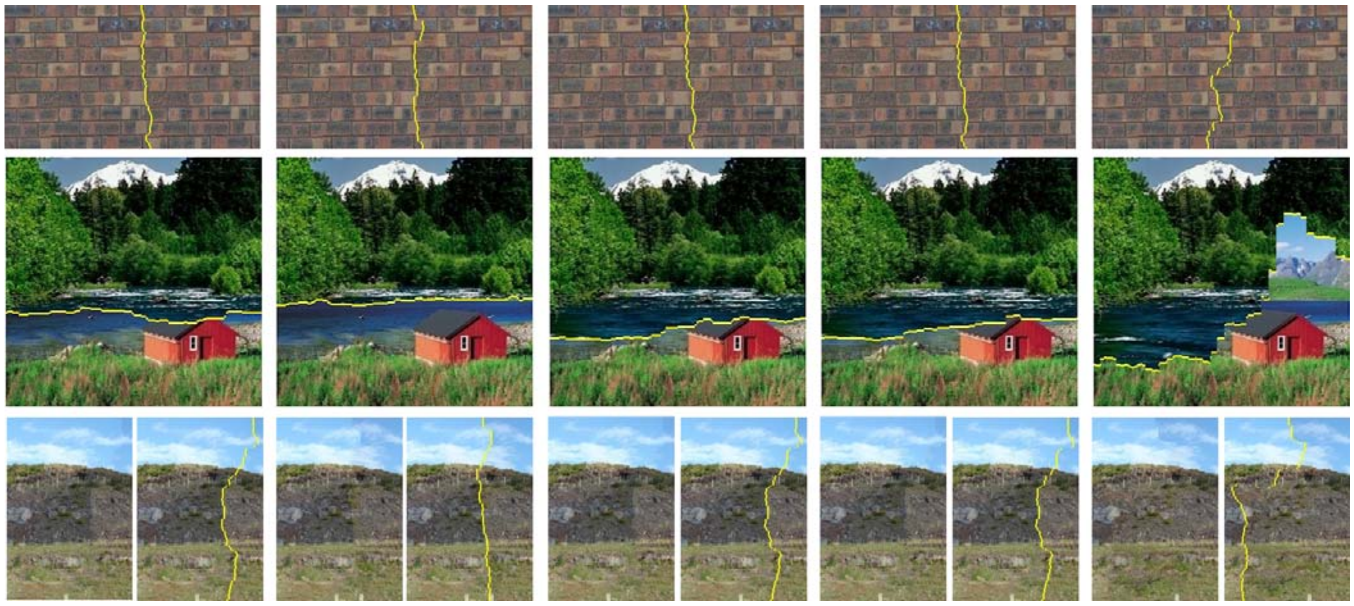

Figure 3: Examples of the results obtained using different colour spaces. From left to right: RGB, HSV, CIELab, CIELuv, Perceptual Colour Space [4].

5. RESULTS

We evaluated different parameter combinations for the graph cut algorithm using 30 pairs of test images representing different application scenarios and image properties as described in section 4.2. We used different combinations of the parameters listed in section 4.1 resulting in 465 images in total. The complete set of images can be found at `www.cs.auckland.ac.nz/~burkhard/Research/GraphCut`.

The following subsections discuss key findings and give representative examples.

5.1 Effect of Norm

Figure 2 shows the results of using the cost function from equation 1, an RGB colour space, and different norms for calculating the colour distance. The L_2 norm is most commonly used in the literature. Our results indicate that the L_1 norm is similarly good, but is much faster to evaluate and hence of interest in applications where speed is important. The L_∞ norm is not useful since it only considers the largest colour component. One problem of using the L_2 norm is that it discourages long cutting paths, since the sum of many small errors is often larger than one big error. For images with high frequency information and large colour distances between image features we obtained best

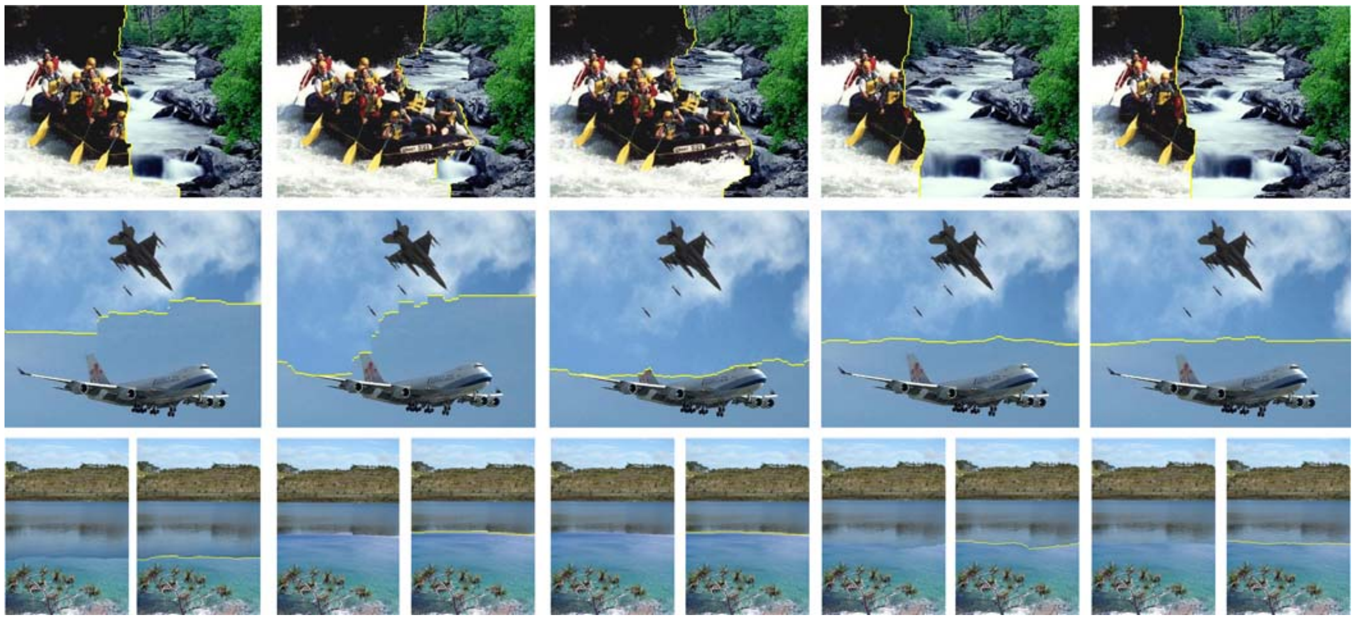

Figure 4: Examples of the results obtained using different cost functions. From left to right: colour distance (equation 1), gradient weighted colour distance (equation 2), componentwise gradient weighted colour distance (equation 7), gradient (equation 3, case 2), and colour distance + gradient (equation 3, case 3).

results by using the L^2 and L^3 distance measures, which effectively polynomially scale edge weights in the graph. This results in a more complex path, which follows feature boundaries and is often visually more pleasing. This is most evident in the "woodgrain" example (top row of figure 2), which has no visible seam when using L^2 and L^3. The "river and raft" example (middle row of figure 2) demonstrates that the use of this exponential distance measure creates a path which closely follows feature boundaries (raft), even if they are complex. Interestingly, the exponential distance measures perform badly for small scale stochastic textures such as grass (bottom row of figure 2).

5.2 Effect of Colour Space

Figure 3 shows the results of using the cost function in equation 1, the L_2 norm, and different colour spaces. We found that the RGB, HSV, CIELuv and CIELab colour spaces result in slightly different cuts, but with few, if any, visible differences in terms of seam quality. In general the RGB colour space is sufficient. The HSV colour space can be advantageous when dealing with scenes with large hue variations. For example, in the "mountain and hut" scene (middle row of figure 3) the HSV space produces the only cut lying entirely within the water region (blue hue), whereas other colour spaces produce cuts across the roof or the vegetation.

The Perceptual Colour Space [4] demonstrates a very uneven performance, but works well in image stitching examples. For example, in the bottom row of figure 3 this colour space is the only one, which creates a cut avoiding regions with illumination differences. The result is a virtually invisible seam across the rock face.

5.3 Effect of Cost Function

Figure 4 shows the effect of using different cost functions. In all cases we used the RGB colour space and the L_2 norm. We found that Kwatra et al.'s gradient weighted colour distance (equation 2) performed almost always best. Our modification of applying the gradient to each channel separately (equation 7) is a possible alternative. For example, in the middle row of figure 4 this creates a cut closely following the feature boundary of the plane, which reduces the length of the visible seam within the air layer. However, since the vertical stabiliser at the tail of the plane has a similar colour to the surrounding air, the cut goes across it. The gradient-based cost functions produce good results if overlapping image regions contain similar materials, e.g., the water region in the bottom row of figure 4.

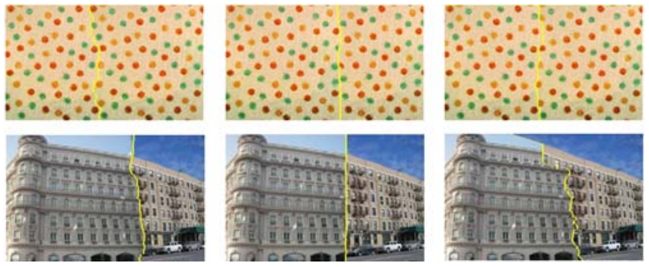

Figure 5: Examples of the results obtained using different appearance spaces. From left to right: standard approach, edge information, signed feature distance.

5.4 Effect of Appearance Space

Figure 5 shows the results of using different appearance spaces, i.e., rather than applying the cost function to the image we compute an appearance space (edge information, signed feature distance [10]) and apply the cost function and graph cut algorithm to these derived images. The results are disappointing. The edge information yields occasionally interesting and plausible images, but results are inconsistent.

The signed feature distance does not maintain spatial relationships of features as expected, and performs always worse then the standard approach (equation 1, RGB colour space, and L_2 norm).

6. CONCLUSION AND FUTURE WORK

In general Kwatra et al.'s cost function (equation 2) in combination with the RGB colour space and the L_2 norm works well for most applications. Our new componentwise gradient weighted colour distance is a good alternative and works well for images where different colours have a strong semantic meaning (different materials). For image stitching the non-linear cost function L^3 is a good alternative, since it avoids cuts across features such as clouds. This often results in a smoother transition between images. For image quilting non-linear cost functions become even more important, especially for images with high frequency information and strong textural features such as the "woodgrain" and "brick".

Changing the colour space has in general little influence on the results. The perceptual colour space [4] proved useful for combining overlapping images of a panorama with slight illumination differences. However, the colour space failed several times badly in image editing examples.

For image editing all cost functions produced relatively poor results. In general the use of the L_1 and L_2 norm is recommended. The exponential distance measures can work well for images with high colour and feature variations, since in this case the extra cost of a longer cutting path can be offset by smaller colour transitions across the path. However, this characteristic leads to poor cuts for stochastic textures (e.g., grass). The gradient-weighted cost functions (equation 2 and 7) produced the most consistent results for image editing applications, but still fail frequently. Kwatra et al. mention in their paper [7] that they "constrain some pixels from the output image". They report further that the title image of their paper took roughly one hour to complete. For image editing manual interaction seems to be crucial and so far no satisfactory combination of parameters exist for automatically creating consistently visually pleasing results.

Many of the successful cost functions are not norms, but semi-norms. Hence several properties of the alpha-expansion algorithm, such as the worst case upper bound, do not apply. This drawback is most crucial in 3D applications where alternative solutions such as [13] do not apply. Using different graph cut solvers, such as GridCut [15], produced occasionally slightly different solutions.

In future work we want to investigate more cost functions and especially the use of multiple appearance space attributes simultaneously. We are particular interested in using graph-cut techniques for image-based modelling applications to reconstruct and transfer textures.

7. REFERENCES

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. *ACM Trans. Graph.*, 23(3):294–302, Aug. 2004.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, Sept. 2004.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, Nov. 2001.

[4] H. Y. Chong, S. J. Gortler, and T. Zickler. A perception-based color space for illumination-invariant image processing. *ACM Trans. Graph.*, 27(3):61:1–61:7, Aug. 2008.

[5] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of SIGGRAPH '01*, pages 341–346, New York, NY, USA, 2001. ACM.

[6] M. Kulkarni and F. Nicolls. Interactive image segmentation using graph cuts. In *Proceedings of 20th Annual Symposium of the Pattern Recognition Association of South Africa (PRASA2009)*, Stellenbosch, South Africa, 2009.

[7] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, July 2003.

[8] S. Lefebvre and H. Hoppe. Appearance-space texture synthesis. *ACM Trans. Graph.*, 25(3):541–548, July 2006.

[9] J. Long and D. Mould. Improved image quilting. In *Proceedings of Graphics Interface 2007*, GI '07, pages 257–264, New York, NY, USA, 2007. ACM.

[10] F. Manke and B. C. Wünsche. Analysis of appearance space attributes for texture synthesis and morphing. In *Proceedings of the 24th International Image and Vision Computing New Zealand Conference (IVCNZ 2009)*, pages 85–90, Wellington, New Zealand, 2009.

[11] G. Ramanarayanan and K. Bala. Constrained texture synthesis via energy minimization. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):167–178, Jan. 2007.

[12] M. Rubinstein, A. Shamir, and S. Avidan. Improved seam carving for video retargeting. *ACM Trans. Graph.*, 27(3):16:1–16:9, Aug. 2008.

[13] F. Schmidt, E. Toppe, and D. Cremers. Efficient planar graph cuts with applications in computer vision. In *Proceedings of Computer Vision and Pattern Recognition (CVPR 2009.)*, pages 351–356, June 2009.

[14] S. N. Sinha. Graph cut algorithms in vision, graphics and machine learning. Technical report, UNC Chapel Hill, 2004. `http://cs.unc.edu/~ssinha/pubs/SinhaGraphCutsIP2004.pdf`, Last retrieved 4th September 2012.

[15] D. Sykora and O. Jamriska. Gridcut homepage, 2012. `http://gridcut.com`, Last retrieved 26th October 2012.

[16] T. Thormählen and H.-P. Seidel. 3d-modeling by ortho-image generation from image sequences. *ACM Trans. Graph.*, 27(3):86:1–86:5, Aug. 2008.

[17] G. Vogiatzis, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Proceedings of Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 391–398, 2005.

[18] K. Zou, Y. Li, Z. Li, R. Li, and X. Xu. Improved graph cuts for patch-based texture synthesis. In *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT 2009)*, pages 122 – 125, Aug. 2009.