Light Field Montages

Vera M. Dadok*
University of California, Berkeley

Figure 1: *A montage of two light fields creates a novel scene. The left two images are representative images from two source light fields, and the righ-most image is representative of the final light field montage in which two plants from the first plant box have been removed.*

Abstract

This paper presents an algorithm for creating a light field montage from two source light fields with user interaction. The ability to generate new scenes from static light fields could be useful for both personal and commercial uses including scenarios such as creating an ideal scene containing multiple people or removing unwanted objects. This work also provides a starting point for a variety of possible extensions that may build a valuable toolbox for those interested in manipulating light fields.

Keywords: light field, graph cut

1 Introduction

Light fields as presented in this paper have been in use academically since their development in 1996 [Levoy and Hanrahan 1996]. They provide an interesting area of research due to their rich information content, containing a 4-dimensional sample of the plenoptic function which allows many interesting manipulations beyond those possible with single images. Due to their relative scarcity, few tools exist to manipulate light fields in comparison to the number of tools available to manipulate images.

However, recently interest in light fields has been growing as researchers and enthusiasts anticipate a new wider availability of light fields. In 2005, researchers at Stanford developed a way to sample light fields with a single hand-held plenoptic camera shot [Ng et al. 2005]. A camera based on this idea has been developed commercially and may be available by next year for purchase, promising the existence of many light fields that will need light field-specific tools to achieve their full potential.

Like pictures, light fields often hold memories or ideas in addition to pixels, and due to the ephemeral nature of the world, light fields will fail to capture the ideal scene desired by the photographer. This desire to improve upon existing scenes is the motivation for developing an algorithm to create a montage from multiple light fields.

Consider the top two images in Figure 2, one contains a woman looking away from the camera, and the other lacks the amusing symmetry between the poses of the other two people. To achieve an ideal picture a montage was created, which is displayed in the third image. This particular situation has rarely occurred for light fields due to the slow speed of capture and portability issues when using traditional measurement devices. However, with a single-shot hand-held camera, imperfect images will become commonplace.

Light fields are challenging for users to manipulate using current commercial programs designed for photographs due to their large sizes and formatting. In this paper we investigate manipulations to light fields formatted as a large array of images taken at different coordinates. Regardless of the storage format, looking at light fields a single image at a time would be too time consuming for the average user, not to mention extremely tedious due to the redundant nature of images extracted from a light field.

Additionally, it is challenging for a user to keep track of relevant geometries and work through a large array of images while keeping track of how the final light field will look when the images are composed. For these reasons, it is essential that interactive tools for manipulating light fields be developed.

Although automatic tools are needed, their development is not trivial. Due to the large file sizes and high dimensionality of light fields, thoughtful decisions must be made about how to process automatic tasks. Additionally, due to the nonintuitive nature of light fields for many users, the user interface and user interactions must be developed carefully to allow the user to control automatic tools without confusion or frustration.

This work presents a method of interactively combining two light fields taken from the same viewpoint to create a novel light field scene. Currently, the method is only applied to two light fields, but is extendable to multiple inputs via an interactive loop.

2 Related Work

The original name of light field originated in the early 1900s but its modern meaning was created when Marc Levoy and Pat Hanrahan wrote the paper "Light Field Rendering" [Levoy and Hanrahan 1996]. Since then, many techniques have been developed to manipulate the lighting, geometry, and other properties of light fields.

*e-mail: vdadok@berkeley.edu

Figure 2: *Motivation*

Of particular relevance are papers that deal with compositing and light field warping. These papers include [Chen et al.], which shows several manipulation techniques including compositing and warping. A work by some of the same authors, [Chen et al. 2005] demonstrates light field warping and deformation under user interaction. In both of these papers the focus is more on operations for animation, on real-time interactions for speed, on on making a framework of binary operations on light fields that can manipulate light fields in more complex ways. In contrast, this work is more application based, focused on combing elements from two similar scenes in a simple user environment. Also, this approach is aimed at more complex geometries that aren't as clean or well known as animated geometries and manipulating full scenes rather than scene elements.

Seam carving has been tested on light fields in [Birklbauer and Bimber 2011], interestingly, the light fields are treated as a depth stack. Finally, [Cossairt et al. 2008] has a more mechanical approach to combining a real and synthetic light fields using a constructed stage.

There is a significant amount of research related to creating montages and mosaics from images but this work was primarily inspired by one paper in particular, [Agarwala et al. 2004]. Closely related work includes matting and compositing, and seam carving. The scope of this work does not include a thorough review of these topics.

3 Methods

Following the philosophy of [Agarwala et al. 2004], which presents an interactive environment for image montages, montages of light fields are considered from the perspective of graph cuts. Consider a montage created from two light fields: in this formulation, all elements (pixels for the format used in this work) of the light field montage are derived from either the first or second source light field. The question then is simply how to choose which pixel comes from which source light field?

Note that this current methodology is limited somewhat by its statement: we are making several assumptions in this formulation of the problem. First of all, we are assuming that the two source light fields are comparable in both size and geometry, or can be rescaled in some way to have matching elements (pixels). This could be a limitation when solving more general montage problems, but this exploration is motivated by the idea of composing a novel scene from a series of captured light fields taken from a single scene undergoing changes between each capture.

The format of the light fields used in this work is a set of images of a scene taken at different (u, v) coordinates. Essentially, this is a 2-d array of 2-d images, and one of the most straightforward ways to store light fields. Any pixel in a light field can be identified with a 4-vector (u, v, x, y) identifying an image at (u, v) coordinates and the (x, y) position within the image. New views can be interpolated or calculated from this light field, but for now we consider the storage data format, which is the foundation of any transformation. For more information on the formatting of the light fields and to learn how these were generated, please see Stanford's New Light Field Archive at `http://lightfield.stanford.edu/`. The light fields used in this work are from this archive.

From two source arrays of images, a montage with the same dimensions is created using a graph-cut algorithm which labels each pixel in the montage array of images as coming from a specific source. Consider pixel p in the final montage, its value, $F(p)$, will be derived from either source 1, S_1 or source 2, S_2 at pixel p. Thus, the algorithm must discover a labeling L for every pixel in the final light field montage F, identifying each pixel as originating from S_1 or S_2. With this labeling, it is simple to construct the final montage F.

3.1 User interaction

Any choice of labeling would create a viewable light field, but our goal is to allow the user to specify properties of the resulting light field F by identifying which sections of the input sources must be present in the final montage. Then the algorithm creates a light field that looks good to the human eye under these requirements. The user interface now provides two windows, each containing four images from the (u, v) corners of the light field, one set of corners from each source. On these 8 images, the user selects pixels which are required to be present in the final light field. Essentially, the user selects parts of each source light field to 'keep'.

Then these source-fixed pixels defined at the corner are linearly interpolated for every other image located at a (u, v) between the four (u, v)-corners, defining for these inner images a set of pixels with a fixed final labeling.

The user can also select what cost function to use in the graph cut optimization by selecting in the code what smoothness function to use. Currently, the code uses the seam objectives described as in [Agarwala et al. 2004], however for most of the images present in

this paper, the *Color* smoothness function has been used for simplicity.

3.2 Graph Cut

Graph cut chooses a labeling that minimizes a cost function comprised of a data term and a smoothness term. The data term contains the fixed-pixel-source information defined through user interaction with the light field images. It is augmented in this implementation with a inverse square law around the selected pixels to create an area of pixel preference rather than points. This data term is defined for every pixel. Different data terms and objectives can be developed in the future as more light fields to manipulate become available.

The smoothness term is a function defined for the edges between adjacent pixels in the light field. Its cost represents the cost of breaking a connection between two pixels; that is the cost of labeling two adjacent pixels with two different labels rather than the same. The smoothness functions mentioned above are different ways to define this cost depending on the goal of the montage. For more details on the effects of these different cost functions, see [Agarwala et al. 2004]. Equation 1 is the *Color* smoothness cost $C_c(p, q)$ between two pixels p and q.

$$C_c(p, q) = ||S_1(p) - S_2(p)|| + ||S_1(q) - S_2(q)|| \qquad (1)$$

3.3 Implementation

In this interactive platform, the user may choose between 3 different graph cut algorithms: a 2-, 3-, or 4-dimensional graph-cut. In this choice there is a tradeoff between computation time and the presence of artifacts in the resulting light field. The general algorithm for implementation is as follows:

1. Input: select 2 source lightfields
2. User interaction phase: select pixels from each source
3. Graph-cut optimization: generate L via n-d graph-cut
4. Construction: Build montage from labeling L

The graph cut algorithm used is a MATLAB wrapper described in [Bagon 2006], which uses tools and algorithms from [Boykov and Kolmogorov 2004], [Boykov et al. 2001], and [Kolmogorov and Zabih 2004]. It is available for download at `http://www.wisdom.weizmann.ac.il/~bagon/matlab.html`.

Figure 3 shows a visual representation of some of the internal terms of a 2-d graph-cut for an image from a light field with jellybeans. The 'Hsmooth' and 'Vsmooth' terms are smoothness terms based on the *Color* smoothness function. These terms are zero everywhere except for in the vicinity of certain jellybeans because the images are the same in all other areas (see 'Image 1 - Image 2' plot). The log of the data term is shown to show the spread of the data term from the user-selected 4 pixels. This term obeys an inverse distance squared law and decays rapidly. The 'Labels' image shows the final labeling chosen, which achieved the user-goal of setting a green jellybean next to a maroon jellybean in the final montage. This example is somewhat contrived but demonstrates well the operations of the graph cut algorithm.

3.4 Viewer Software

To view the light field montages, one needs viewer software. For the light fields formatted in the supplementary material, one must use a modified version of the Stanford light field viewer available at `http://lightfield.stanford.edu/aperture.html`. This software also uses a version of FZip (`https://github.com/claus/fzip/`). Unfortunately, the original viewer does not read all .zip files, and thus changes were necessary. The modified version works with a combination of zipped and unzipped files, and is currently set up in a website without any downloading necessary.

4 Results

Light fields are difficult to present in a two-dimensional format, so a few results are currently posted at `http://me.berkeley.edu/~vdadok/LightFieldProject/` for closer inspection. The source light fields used in to produce these results were obtained from Stanford's light field repository, located at `http://lightfield.stanford.edu/`, and thus the credit for all original light fields goes to Stanford Computer Graphics Laboratory.

The algorithms developed in this paper work well in some cases and poorly in others. Often more careful user interactions result in a large change in results which implies that the user interface may need to be extended with additional tools or displays. Figures 7 and 8 show several sample images from a light field in which 2-d, 3-d, and 4-d graph cut were used to optimize the cost function. Each set of four images displayed corresponds to one of the graph cut algorithms and the four images within the set are from adjacent u and v coordinates in the light field. Thus these images are neighbors in the (u, v) plane and ideally switching between them will appear to the human eye as a small change in perspective. By looking at the portions highlighted in Figure 8 and comparing that portion to same region in the neighboring 3 images, one can see artifacts that appear when using the 2-d and 3-d graph-cut algorithms.

In the image set created with 2-d graph-cut, the consistency from image to image is only enforced by the interpolated data term, which provides a global smoothness to the result. However, at a local level, artifacts begin to appear. Note that between the top left and bottom left images, a half-leaf clearly disappears, only to reappear again in the bottom right corner. This kind of artifact is expected for a 2-d graph cut since there is no cost on such single-image artifacts.

In the middle set of images created with the 3-d graph-cut algorithm, the additional dimension in which the graph-cut operated was in the v-dimension, which is the horizontal direction in these images. Note that unlike the previous case, the discontinuity from left to right is minimal. However, from top to bottom there are some artifacts clearly visible in the discontinuities between the leaves.

Finally the set of images created with 4-d graph-cut optimization has a much smoother look and fewer artifacts in this portion of the light field.

Although the 4-d graph-cut algorithm is superior in this example, the difference in computation time is extreme and the amount of memory required is large enough to make larger light fields impossible to run on a typical laptop using this implementation of the graph-cut algorithm. Also, the 4-d graph cut can create different types of artifacts, which are demonstrated in Figure 4. Here, it becomes obvious that the smoothness term across neighbors is too strong compared to the smoothness term within images, as you can see that a certain pattern of pixels persists along the border in spite of the changing content at each view. This could be fixed by changing the weights of the terms between (u, v)-neighbors as compared with the internal image neighbors, but adds another parameter to the algorithm.

Figure 4: *Some artifacts in 4d graph cut*

Finally in many cases, 2-d graph-cut algorithms can work quite well given a reasonable user interaction module. For instance, the jellybean light field shown in Figure 5 and available to view online, was constructed using a 2-d graph cut. There are artifacts under closer inspection (see Figure 6), but these are not immediately obvious, and may be fixed mostly with a different edge cost function (the color cost function was applied in this case). Additionally, online at `http://me.berkeley.edu/~vdadok/LightFieldProject/lightfieldViewer2.html`, a larger plant montage was made with more careful user interaction, and the results are quite reasonable from just the smoothness of the data term.

5 Discussion

Applications for this method are similar to those of other montage algorithms. Creating novel or physically impossible but breathtaking scenes out of an array of light fields is one of the goals of this line of work.

In conclusion, both the 2-d and 4-d graph cut optimization algorithms are very promising methods for creating light field montages. The tradeoff between speed, user interaction, and artifacts will evolve with improvements in the user interface and graph-cut algorithm. For the average user, a faster algorithm such as the 2-d algorithm with an iterative user interface that allows tweaks may be the most viable for the short-term. Finally, the 3-d algorithm could be useful in certain cases in which the artifacts occur only along one dimension, but this is a rare case, and thus 2-d and 4-d graph cut algorithms are likely the best for future directions.

6 Future Work

There is a tremendous amount of future work possible in this area. Within the scope of this particular approach, significantly more work could be done on improving the user interactions. This includes tasks such as creating a wider array of tools that are intuitive for the user, running user studies on the most effective way to edit light fields, speeding up the algorithm to make the software run interactively and iteratively, allowing multiple light fields in a montage rather than just two, and adding additional montage modes such as those presented in [Agarwala et al. 2004].

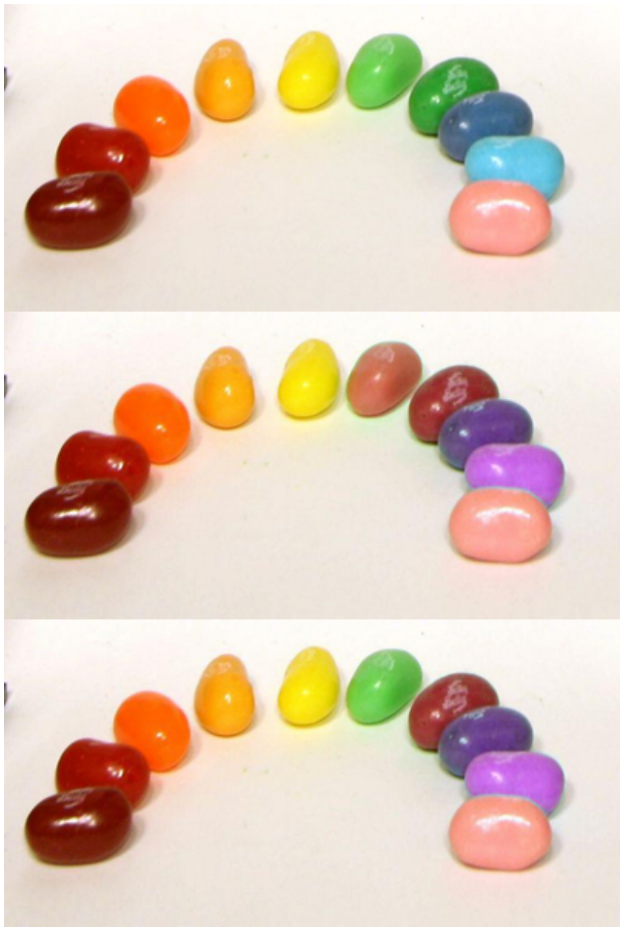

Figure 5: *An example image from a light field montage in which 2-d graph cut worked well*

In addition, the hidden tasks used during user interaction must be improved upon. For instance, developing automatic reality checks for the user selections by using the depth map or other geometry information could be invaluable. Also, the current method of interpolating between corners to find the source-fixed pixels is reasonable as a first attempt, however, knowledge of the geometry and depth and limits on the user's choice of points (for instance asking if the points are on the same object) could be used to improve the interpolation of interior points. As-is there is certainly room for spectacular failure if an unknowing user chose points fixed to one source across an occlusion containing points fixed to the other source.

Also, with this method there is room to develop new montage tools that are specific to light fields. For instance, cost functions for smoothness terms could include a function based on the depth map of pixels or on other geometries extracted from the scene. Also, in the graph cut algorithm, we currently use pixels at the same coordinates (x, y) in adjacent (u, v) as neighbors in the smoothness function, but there may be a better way to calculate neighbors based on the depth map or geometry rather than pixel distance. In addition to this, there may be a transformation from this pixel-based format other elements that would lend themselves even more naturally to graph-cut.

Additionally, since part of the motivation for this research is the advent of public availability of light field cameras, this

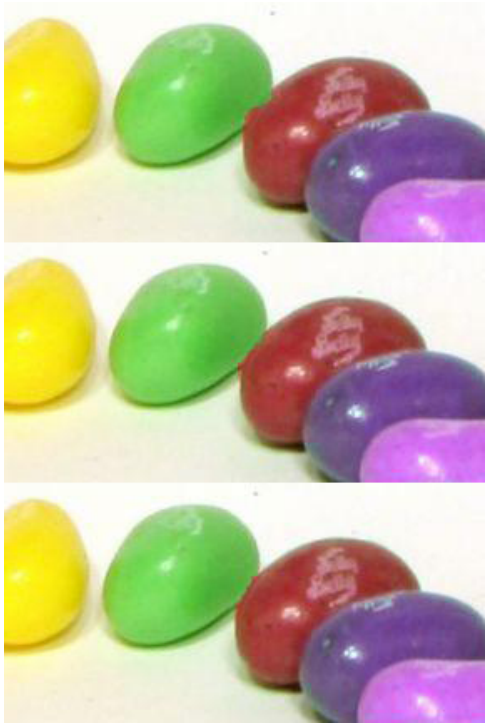

Figure 6: *Some artifacts in 2d graph cut*

type of tool should be developed for the format used by hand-held light field cameras. From the Lytro website's posted light fields, located at `lytro.com`, one can see that the viewing format is a depth stack (a fact verified by tools developed in `https://github.com/nrpatel/lfptools` and `http://eclecti.cc/computervision`). From the documentation online, it seems that the light fields may originally be a different format that is changed to a depth stack as they are uploaded to Lytro's home page, thus the format is unknown. Algorithms to handle various file types and format styles should be investigated for more wide applicability. Interestingly enough, it has been suggested in [**?**] that for certain types of light fields, a depth stack is a more concise representation which can be converted to and from a full 4-d light field. Thus further work should investigate operating a montage algorithm on a depth stack as an internal step or as the entire algorithm for some light fields.

Finally, there are a number of other small details that should be investigated in the future as this algorithm evolves–is there a simple way to develop a Poisson step to use gradient methods to make graph edges less noticeable? How can we leverage the depth map and any geometry knowledge to improve this method? How difficult would it be to rescale and align completely different light field scenes to montages? It will be exciting to see what tools are developed for light fields in the coming years.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. In *ACM Transactions on Graphics (TOG)*, vol. 23, ACM, 294–302.

BAGON, S., 2006. Matlab wrapper for graph cut, December.

BIRKLBAUER, C., AND BIMBER, O. 2011. Light-field retargeting with focal stack seam carving. In *ACM SIGGRAPH 2011 Posters*, ACM, 37.

BOYKOV, Y., AND KOLMOGOROV, V. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on Pattern Analysis and Machine Intelligence 26*, 9 (September), 1124–1137.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Efficient approximate energy minimization via graph cuts. *IEEE transactions on Pattern Analysis and Machine Intelligence 20*, 12 (November), 1222–1239.

CHEN, B., HORN, D., ZIEGLER, G., AND LENSCH, H. Interactive light field editing and compositing.

CHEN, B., OFEK, E., SHUM, H., AND LEVOY, M. 2005. Interactive deformation of light fields. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, ACM, 139–146.

COSSAIRT, O., NAYAR, S., AND RAMAMOORTHI, R. 2008. Light field transfer: global illumination between real and synthetic objects. In *ACM Transactions on Graphics (TOG)*, vol. 27, ACM, 57.

KOLMOGOROV, V., AND ZABIH, R. 2004. What energy functions can be minimized via graph cuts? *IEEE transactions on Pattern Analysis and Machine Intelligence 26*, 2 (February), 147–159.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, 31–42.

NG, R., LEVOY, M., BRÉDIF, M., DUVAL, G., HOROWITZ, M., AND HANRAHAN, P. 2005. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR 2*.

Figure 7: *These are several adjacent images from a montaged image using different graph cut dimensions. Each set of four are taken from adjacent (u,v) pairs. The top set of images was created using a 2d graph cut algorithm, the second set of images was created using a 3d graph cut algorithm which linked adjacent (u,v) images in the v dimension (here the horizontal direction), and the bottom set of four images was created using a 4d graph cut algorithm*

Figure 8: *These are several adjacent images from a montaged image using different graph cut dimensions. Each set of four are taken from adjacent (u,v) pairs. The top set of images was created using a 2d graph cut algorithm, the second set of images was created using a 3d graph cut algorithm which linked adjacent (u,v) images in the v dimension (here the horizontal direction), and the bottom set of four images was created using a 4d graph cut algorithm. Note the highlighted regions*

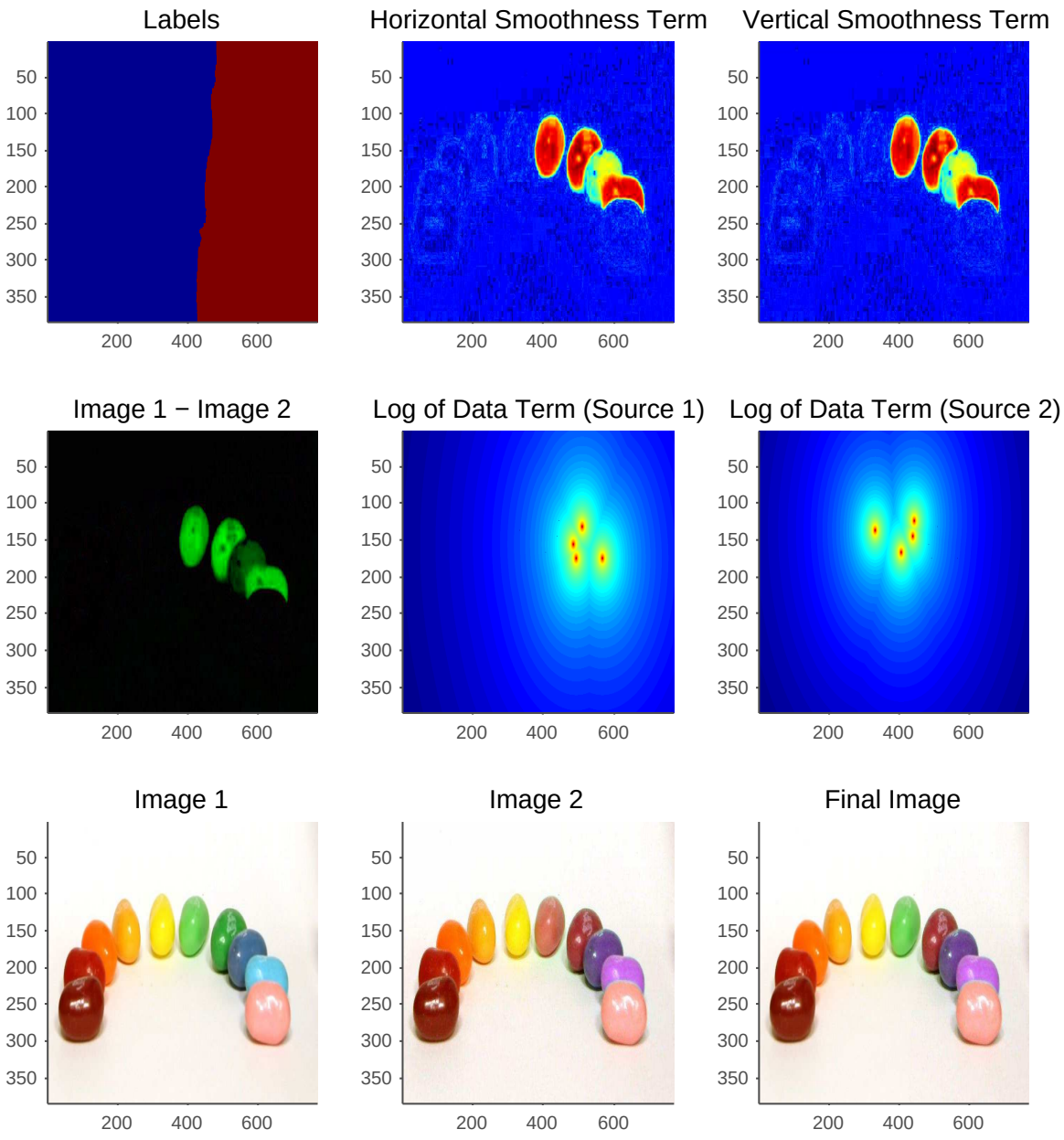

Figure 3: *Terms in a 2-d graph cut algorithm*