Automatic PhotoMontage - Automated composites of group images based on SLIC superpixels and facial expression recognition

Rahul Ghangas
rahul.ghangas@anu.edu.au
Australian National University
Canberra, ACT

Figure 1: A panorama composite

ABSTRACT

With the advent of smartphones, access to a camera has become second nature to us. Taking photographs of important event, or even day to day life is common. However, when taking group photographs, it is often that not everybody is showing their best expression. We often take multiple photographs to curb this issue and still end up multiple images that are not perfect. This paper explores the idea that a single perfect group photograph can be stitched together and whether it is possible to automate this process. With exceptional compute power right in our hands in form of our smartphones, seamless image compositing, which has been a hard problem to date can be offloaded to the device. We show that such automation is possible and present a framework for the automatic computation of photomontages.

KEYWORDS

photmontage, image stitching, image composites, facial expression recognition

1 INTRODUCTION

Multiple image compositing is a problem that has been studied extensively in the field of Image processing. Smith and Blinn (1996) introduced the process of seamless manual extraction called image matting, which was complemented by image insertion without visible artifacts, called image compositing, Porter and Duff (1984), and Blinn (1994). To composite a new (or foreground) image on top of an old (background) image, the over operator was first proposed by Porter and Duff and then studied extensively by Blinn. These advances were quite significant, but image compositing was still constrained by the fact that all regions need to be manually defined. Further advancement came with automatic and semi-automatic segmentation algorithms that provided good seams without significant human effort. Some popular image segmentation techniques were:

- Active contours (Isard and Blake 1998)
- Texture and intervening contour-based normalized cuts (Malik, Belongie, Leung, et al. 2001)
- Mean shift (Comaniciu and Meer 2002)
- Graph-based merging (Felzenszwalb and Huttenlocher 2004)
- Graph Cuts and Efficient N-D Image Segmentation (Boykov and Funka-Lea 2004)
- Binary MRF solved using graph cuts (Boykov and Funka-Lea 2006)
- Level sets (Cremers, Rousson, and Deriche 2007)

Of these, the most important are energy-based segmentation methods. These methods establish a target (energy) function that reaches a minimum when the ideal segmentation is achieved. Live wire, active contours, level sets, and a graph cut are all grouped into this category. Seeds must be identified by the user for live wire, and these seeds must be close to the boundary of the desired object. Then the location of the curve is optimized by minimizing the manufactured energy function. An initial curve is needed for

active sets of paths and levels. When a curve's evolution results in the form of a predefined rule, the extraction of a reasonable curve is attempted that tries to minimize the energy function. However, the active circuit and the set level use only information about the boundary and are very sensitive to the initial curve. In addition, no guarantee can be given that they will receive a globally optimal result because they tend to converge at local minima. For the segmentation of a graph section, the energy function is built on the basis of regional and boundary information, and it can achieve an optimal global result. Graph segmentation was first proposed by Boykov and Jolly in 2001. This resulted in the evolution of various graph-based methods, and these approaches are widely used in the segmentation of medical images, video, and natural images.

Probably the most notable work in image compositing is "Interactive Digital PhotoMontage" (Agarwala et al. 2004), which uses user context-based graph cut optimization and gradient-domain fusion to combine a set of images into a single composite picture. The power of the proposed framework lay in its generality, its applicability to a wide variety of applications such as selective composites, relighting, extending the depth of field, panoramic stitching, clean-plate production, time-lapse mosaics, etc.

But with the recent advent of neural network architectures, especially various deep learning ones, and their ability to solve complex problems in a black box format with higher precision than traditional methods, we must rethink the problem of creating digital photo montage(s) and whether we can automate this process. This could lead to a mobile application deployable on various photographic equipment such as smartphones and consumer drones, allowing users to create a desirable composite from multiple pictures of a large group of people. The goal of this paper is to extend the original research with a process that creates automatic digital photomontage and the proposal of a new, faster approach to finding masks with good seams compared to iterative graph cut on the entire image.

(a) Digital photo montage using user context - Composite selection

(b) Digital photo montage using user context - Shadow removal

Figure 2: Digital photo montage as implemented by Agarwala et. al. (2004)

2 BACKGROUND

The following section provides a background overview of the algorithms used in our application.

2.1 SLIC superpixel segmentation

The SLIC algorithm generates regions of pixels, termed superpixels, by clustering pixels based on the affinity of position and color in the image. This is done in a 5-dimensional space [labxy], where the first three values comprise the color vector of the pixel in the latter values that capture the pixel position. Normalization of spatial distances is required to use Euclidean distance in this type of space. This is because the maximum possible distance between the two colors in this space is limited. However, the spatial distance is only limited by image size, which means, in order to cluster pixels in this space, a new distance measure is introduced that takes into account the number of pixels in the superpixels.

The algorithm begins by sampling N homogeneous clusters and moves them to seed locations corresponding to the minimum gradient position in a defined neighborhood grid, which is done to avoid choosing a noisy pixel by placing them at the edge.

Algorithm 1 SLIC super pixel segmentation

1: Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S
2: Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
3: **while** $E \leq$ threshold **do**
4: **for** each cluster center C_k **do**
5: Assign the best matching pixels from a $2S \times 2S$ square neighborhood around the cluster center according to the distance measure (Eq. 1).
6: Compute new cluster centers and residual error $E\{L1$ distance between previous centers and recomputed centers
7: Enforce connectivity.

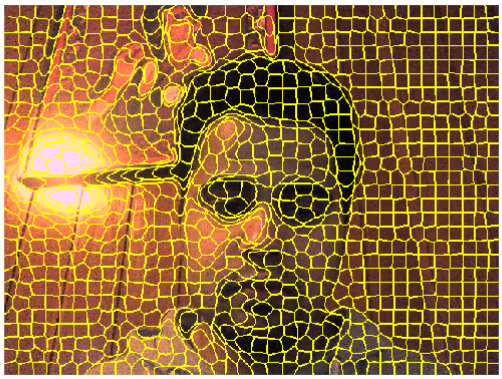


Figure 3: Superpixel segmentation

2.2 Graph Cuts

For an undirected graph, denoted as ***G*** = <***V, E***>, ***V*** is a sequence of vertices and E is an edge of the graph connecting all pairs of neighboring vertices. The vertex ***V*** consists of two kinds of nodes. The first type of vertex are neighbour nodes that correspond to pixels, and the other type are called terminal nodes, which consist of ***s*** (source) and ***t*** (sink). This variety of graph is also called a ***s-t*** graph, where node ***s*** usually represent an object, and ***t*** node denotes a background. There are also two kinds of edges in this type of graph. The edges of the first type are called ***n-links***, which connect neighboring pixels in the image (here we accept a 4-connected system in a 2D image). And the second type of edge is called ***t-links*** that connects the terminal nodes to adjacent nodes. Each edge is initialized with a non-negative weight, denoted as w_e, called cost. A cut is defined as a subset of the edges of E that can be denoted as C. Cut Cost $\sum$ C is the sum of the weights on the edges of C, which is expressed as follows.

2.2.1 Energy approximation using Graph cuts. Energy-based methods try to simulate some global image properties that cannot be captured, for example, by local correlation methods. The issue, though, is that interesting energy(ies) are difficult to minimize. For example, it has been proven that minimizing the Potts energy on a multi-label grid is an NP-hard problem. Therefore, in order to find an approximate solution, Boykov et al. proposed two algorithms called alpha-beta swap and alpha-extension. Unlike standard moves (one pixel changes its mark at a time), these movements allow a large number of pixels to simultaneously change their labels. This makes the set of labels within one movement locally optimal f exponentially large. In addition, the use of these displacement solutions does not change significantly by changing the initial labels. We will only be discussing alpha expansion since it pertains to this paper.

2.2.2 α-Expansion Algorithm. The expansion algorithm is very similar to the α-β Swap algorithm with one difference, in each iteration (for each label), we try to find a label that is one α-Expansion from the current label. Some pixels whose label is different from α (the label by which we iterate) can change their labels to α, but pixels whose label is α do not change their label. Thus, α-Expansion can be considered as a special case of α-β swap, in which β is allowed to collect all labels except α and only from movement β to α is valid. The expansion algorithm pseudo-code is shown in Algorithm 2.

Algorithm 2 α-Expansion Algorithm

```
1:  procedure MyProcedure
2:      f ← Arbitrary Labelling
3:      success ← 0
4:      while success != 0 do
5:          for each label α ∈ L do
6:              f_cut ← argmin(E(f')) ∀ f' in one α-expansion of f
7:              if E(f_cut) < E(f) then
8:                  f ← f_cut
9:                  success ← 1
10:     return f
```

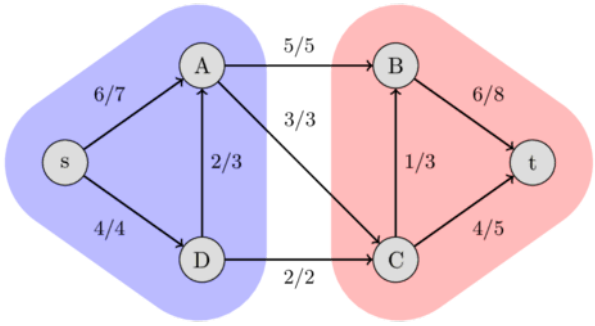


Figure 4: Graph cut using max flow-min cut

2.3 Gradient Domain fusion

The purpose of this Gradient-domain fusion is to create a source image and a target image in a gradient region. The reason it (Gradient-domain fusion- Poisson blending) provides a more realistic composition than naively gluing two images painted with the same color is that the human visual system is more sensitive to contrast than to intensity values.

Gradient Domain Fusion was first introduced in compression with a large dynamic range [Fattal et al. 2002] and image compositions [Perez et al. 2003]. This powerful idea gives a different view of the images. Instead of taking into account and changing the image intensity, we could manipulate the gradient field. This perspective offers us two advantages [Bhat et al. 2010]:

- Great for human perception. The difference between the intensities (i.e., the gradient) is much more noticeable for our human vision system.

- High level of control over images: a local change in the gradient field can affect the overall image.

Gradient area methods have been applied to many problems in both computer vision and computer graphics. Gradient Brush introduced a user-friendly interface for drawing in a gradient area with real-time feedback on large images [McCann and Pollard. 2008], while the Gradient Shop [McCann and Pollard. 2008] provided a gradient area optimization infrastructure for determining perceived motivated images and video filters.

The gradient domain framework could be viewed as a energy function optimization problem, where the result image is generated by minimizing the following function:

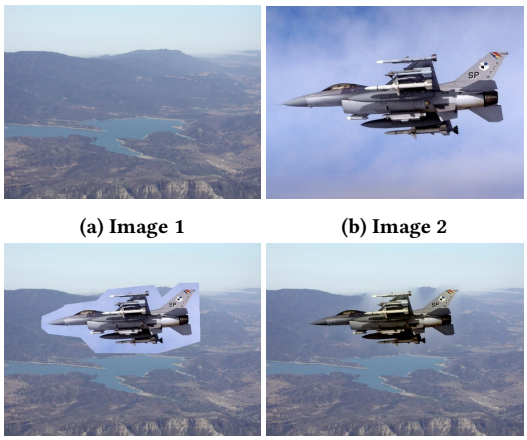
$$E(f) = \sum_{p \in f} \lambda_d E_d(p) + E_g(p) \quad (1)$$

where p is a pixel in image f, E_d is intensity cost function, and E_g is gradient cost function. The energy terms E_d and E_g are quadratic functions defined as follows:

$$E_d(p) = w^d(p)[f(p) - d(p)]^2 \quad (2)$$

$$E_g(p) = w^x(p)\left[f_x(p) - g^x(p)\right]^2 + w^y(p)\left[f_y(p) - g^y(p)\right]^2 \quad (3)$$

The energy terms E_d and E_g are the squared errors between the desired values (the guidance intensity image d and guidance gradient field g), and the values of the final image.



(a) Image 1 (b) Image 2

(c) Naive Blend (d) Poisson Blend

Figure 5: Two images, their juxtaposition, and a composite image using Gradient domain fusion (Poisson Blend)

3 FRAMEWORKS USED

3.1 Face detection and recognition

We use face detection to define regions of interest in an image. These regions are treated as a potential context for the graph cut algorithm. A Histogram of Oriented Gradients (HOG) descriptor is used for CPU only devices, and a Convolutional Neural Network (CNN) is used for devices with a GPU present.

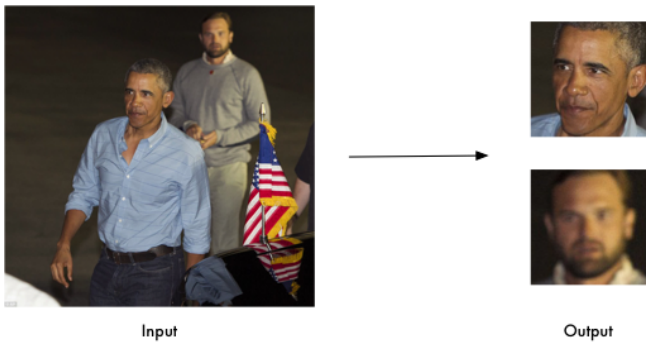


Figure 6: Face detection

Although face detection helps us identify regions of interest (humans) in all images, the final image would only contain only one image of every unique individual. For this, we use face recognition across all images to keep track of all unique individuals so that the application can pick one image of the person in the final stitched image.

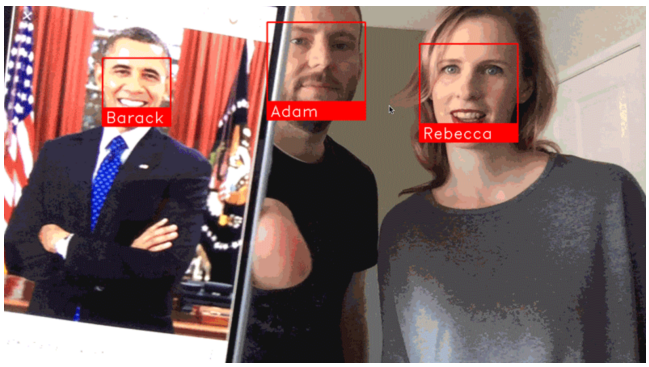


Figure 7: Face recognition

For both face detection and recognition, we use an implementation in Python based on " dlib C++ Library", a robust machine learning toolkit with various state of the art implementations to choose from.

3.2 HaarDeXpression: Deep Convolutional Neural Network for Expression Recognition

Although face detection and recognition provide with potential context, it will segment the same human in all the provided images. Ideally, a user would select which segment to consider based on the pose or facial expression, but since we want to automate this process, we will consider an existing deep learning architecture to define a metric for choosing between different segmentations.

HaarDexpression is based on GoogleNet architecture that depends on Convolution Neural Networks. It first came to light during the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2014). The challenge documented the accuracy of multiple classification approaches submitted by various research groups. The images were partitioned into 1000 different labels organized by the WordNet hierarchy. In the challenge "object detection with additional training data" although GoogleNet performed well, it was far from being used in a reactive environment. These results, however, demonstrated that this kind of architecture could be utilized for increased accuracy. HaarDexpression, however, performs much better in terms of expression detection and achieves 86% accuracy on the same dataset.

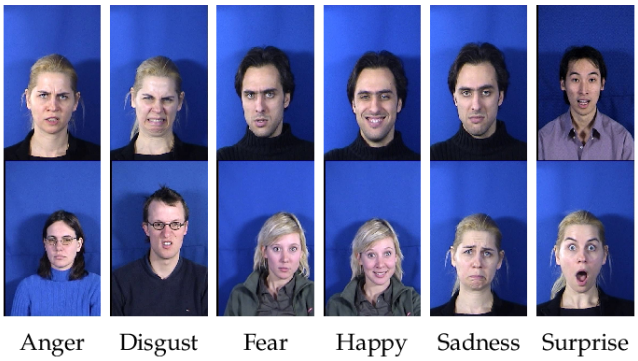


Figure 8: HaarDexPression

4 AUTOMATIC PHOTO MONTAGE

Our approach is primarily based on four aspects:

- Facial detection/recognition to mimic user context but with larger explicitly defined regions from each image.
- SLIC superpixels to find better seams in each individual image and then using a small scale local graph to find the best mask.
- Use facial recognition and facial feature classification to define metrics for simulating desirable regions, hence eliminating the need for user context.
- Develop a user interface that allows the user to made minute modifications on the automated composite in order to remove unwanted artifacts and create the perfect picture

The graphical user interface (GUI) allows users to provide contextual information using colored strokes before or after calculating a mask using *autoMontage*. The GUI is generated using Python bindings to Tkinter, and camera images are captured in on a separate thread. It is based on an event-driven design model with multiple listeners for various events like image capture, non-uniform state rollback, user input, and various buttons to perform specific operations.

While performing automatic photo montage, all faces in each image are identified and a table of unique faces is created. All identified faces in each image are iteratively added to the table while cross-matching against faces already present in the table. Matched faces are appended to the corresponding list to each unique person, while unmatched faces are registered as a new unique person. This process, when completed, leads to a map where all captured faces are stored with an identifier attached to each list of faces that correspond to a single person.

For each person, only one segmentation is required in the final image, so we need to define a metric of which face to select. For this, we use a neural net to detect facial expressions in each face corresponding to a unique person and select the face with the best feature (expression). In the case of multiple faces with the best feature present, the face with the highest probability corresponding to the desired feature is selected. The metric of best features is defined as follows, with a higher number being attached to features that are more desirable.

- 1 : Disgust
- 2 : Fear
- 3 : Sadness
- 4 : Anger
- 5 : Neutral
- 6 : Surprised
- 7 : Happy

Finally, a bounding box for each selected face is, and a corresponding mask matrix is created. This mask matrix is used to pick superpixel segments from each image, and these segments are used to define graph nodes as a cluster of pixels rather than a pixel for each node for the graph cut algorithm. This leads to a smaller graph that can be iterated over in 2 dimensions faster. Then, a new mask is calculated using graph cut, and the final image composite is made by applying gradient domain fusion on the corresponding images of the mask.

However, unwanted artifacts are often left over in the final composite. These artifacts can be in the form of multiple faces of the same person, jagged seams, etc. Multiple faces can often result from the fact that the person moved places between images, or the face got occluded in one or more pictures, leading to faces belonging to the same person not being matched. In this case, we provide the user the ability to remove small portions of the current mask or even add context from any image. This can be done using the erase pointer and dragging across the overlaid mask of any given image.

5 RESULTS

Firstly we demonstrate how graph cut can be used with user-supplied context to form composite images. We use the set of images in Figure 9 and the set of images in Figure 10 for this experiment.

Using user context, graph cut can be used for selecting different body parts from different images. For the images in Figure 9, Figure 11 shows the user-supplied context, and Figure 12 shows the composite result.

Secondly, we demonstrate choosing different people from different images from a group photo. For the images in Figure 10, Figure 13 shows the user-supplied context, and Figure 14 shows the composite result.

Figure 9: Image set 1

Figure 10: Image set 2

Figure 11: User supplied context for choosing body parts

Finally, we show the results from automatic photo montage. Figure 15 shows the automatically selected context, and Figure 16 shows the merged image. As one can see, the merged image has left-over artifacts. So, we add additional user context, as shown in Figure 17, to remove artifacts and get a final merged image as illustrated in Figure 18.

Figure 12: Composite result for choosing different body parts using graph cut

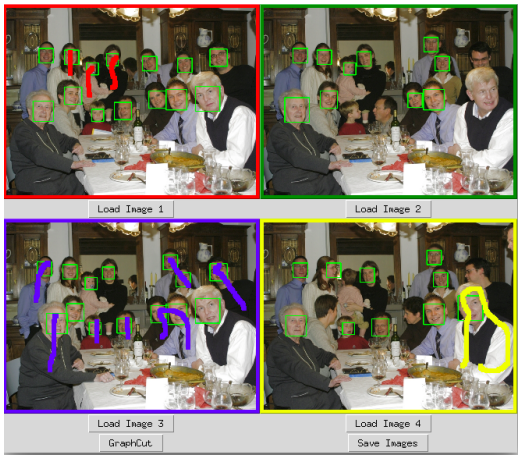


Figure 13: User supplied context for set of group photos

Figure 14: Composite result for choosing different people using graph cut

Figure 15: Automatic selection of regions of interest

Figure 16: Composite result for choosing different people using graph cut

Figure 17: Additional user context to remove unwanted artifacts

Figure 18: Final photo montage after minimal user input

6 CONCLUSION

In this paper, we have presented a new approach to creating image composites from a group of similar images. We have shown that our approach requires minimal user context to find desirable composites. Accompanying our implementation is also a graphical user interface that allows a user to make composites based entirely on user context, giving the user flexibility in choosing their approach towards finding good stitches. The accompanying code can be found at https://github.com/rahulghangas/Digital-PhotoMontage-Python

7 REFERENCES

1. Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. 2004. Interactive digital photomontage. In ACM SIGGRAPH 2004 Papers (SIGGRAPH '04). Association for Computing Machinery, New York, NY, USA, 294–302. DOI:https://doi.org/10.1145/1186562.1015718

2. Achanta, Radhakrishna Shaji, Appu Smith, Kevin Lucchi, Aurélien Fua, Pascal Susstrunk, Sabine. (2012). SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. IEEE transactions on pattern analysis and machine intelligence. 34. 10.1109/TPAMI.2012-.120.

3. Satiyan, M. Nagarajan, R.. (2010). recognition of facial expression using Haar-like feature extraction method. 1 - 4. 10.1109/ICIAS.-2010.5716228.

4. Saravanan, Sharma Shanmugasundaram, Karthikeyan Ramasamy, Sathees. (2016). FAREC — CNN based efficient face recognition technique using Dlib. 192-195. 10.1109/ICACCCT.2016.7831628.

5. F. Yi and I. Moon, "Image segmentation: A survey of graph-cut methods," 2012 International Conference on Systems and Informatics (ICSAI2012), Yantai, 2012, pp. 1936-1941, doi: 10.1109/ICSAI.2012.6223428.

6. Boykov, Yuri Funka-Lea, Gareth. (2006). Graph Cuts and Efficient ND Image Segmentation. International Journal of Computer Vision - IJCV. 70. 109-131. 10.1007/s11263-006-7934-5.

7. Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2004. Mesh editing with poisson-based gradient field manipulation. ACM Trans. Graph. 23, 3 (August 2004), 644–651. DOI:https://doi.org/10.1145/1015706.1015774

8. A. Smolic, K. Muller, K. Dix, P. Merkle, P. Kauff and T. Wiegand, "Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems," 2008 15th IEEE International Conference on Image Processing, San Diego, CA, 2008, pp. 2448-2451, doi: 10.1109/ICIP.2008.4712288.

9. P. J. Burt and R. J. Kolczynski, "Enhanced image capture through fusion," 1993 (4th) International Conference on Computer Vision, Berlin, Germany, 1993, pp. 173-182, doi: 10.1109/ICCV.1993-.378222.

10. Raanan Fattal, Dani Lischinski, and Michael Werman. 2002. Gradient-domain high dynamic range compression. ACM Trans. Graph. 21, 3 (July 2002), 249–256. DOI:https://doi.org/10.1145/566654-.566573