Grid Seams: A fast superpixel algorithm for real-time applications

Parthipan Siva
Aimetis Corp.
Waterloo, Canada
parthipan.siva@aimetis.com

Alexander Wong
Systems Design Engineering
University of Waterloo
Waterloo, Canada
a28wong@uwaterloo.ca

***Abstract*—Superpixels are a compact and simple representation of images that has been used for many computer vision applications such as object localization, segmentation and depth estimation. While useful as compact representations of images, the time complexity of superpixel algorithms has prevented their use in real-time applications like video processing. Fast superpixel algorithms have been proposed recently but they lack regular structure or required accuracy for representing image structure. We present Grid Seams, a novel seam carving approach to superpixel generation that preserves image structure information while enforcing a global spatial constraint in the form of a grid structure cost. Using a standard dataset, we show that our approach is faster than existing approaches and can achieve accuracies close to a state-of-the-art superpixel generation algorithms.**

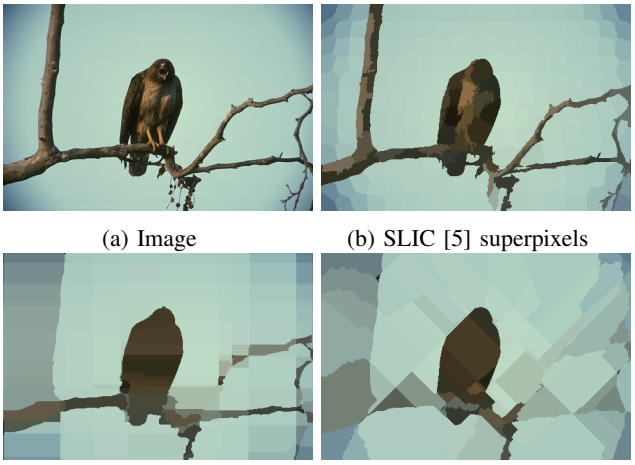
***Keywords*-Superpixels; Seam Carving; Real-time Applications**

I. Introduction

Superpixel algorithms group adjacent pixels into perceptually meaningful regions without the rectangular grid structure associated with pixels. As a result, superpixels provide a convenient and compact representation of images which still preserves the original image structure while using fewer regions than the original number of pixels (Fig. 1). Superpixels have become a key preprocessing step for many computer vision algorithms like object localization [1], depth estimation [2] and segmentation [3]. However, only limited studies [4] have looked into the use of superpixel algorithms for real-time computer vision applications such as background subtractions. This is due to the computational complexity of superpixel algorithms.

Recent studies [7], [8] have looked at computationally efficient means of computing superpixels. Pathfinder [7] and incremental superpixel [8] are proposed as fast algorithms but they don't have high accuracy in preserving image structure nor do they have desirable properties such as ensuring that all superpixels are of similar size or controlling the number of superpixels returned per image. Other methods, like Superpixel Lattice [6], have the desirable control over the size and number of superpixels, but lack the speed necessary for real-time applications.

SLIC [5], a state-of-the-art and popular superpixel algorithm, is a simple method with several advantages: similar sized superpixels, good control over the number of superpixels returned per image, and high accuracy in preserving the original image structure. SLIC achieves these desirable properties of superpixels by combining spatial and colour information. However, the iterative nature of SLIC results in slow performance for real-time applications.



(a) Image (b) SLIC [5] superpixels

(c) Superpixel Lattice [6] (d) Pathfinder [7]

Figure 1: Example of superpixels using the SLIC superpixels [5], and Superpixel Lattice [6] and Pathfinder [7]. All methods have about 250 superpixels but only SLIC and Superpixel Lattice produce 250 similarly sized superpixels.

Inspired by SLIC's use of spatial and image structure information, we present a seam carving based superpixel algorithm which confines the superpixels using a global spatial constraint in the form of a regular grid structure. We call our algorithm Grid Seams and we show that it is a computationally fast algorithm with all the desirable properties of superpixels: similarly sized superpixels per image and good control over the number of superpixels per image. Furthermore, the accuracy of Grid Seams in preserving the original image structure is very close to state-of-the-art methods like SLIC superpixels while being computationally faster.

II. Related Works

Many superpixel algorithms have been proposed [5], [7], [9], [6], [10] in the literature. A recent paper [5] evaluated many of these methods and found that most take on the order of ones, tens, or even hundreds of seconds to process a 320 by 240 pixel image. The processing time requirements makes most of these methods unusable for real-time applications. However, several methods [7], [6], [5] have been proposed which can be run under a second for a 320 by 240 images. We focus on these methods: SLIC Superpixel [5], Superpixel Lattice [6], and Pathfinder [7].

SLIC superpixel [5] is one of the state-of-the-art superpixel methods. It boasts some very useful properties like similar sized superpixels and high accuracy in preserving image structure. SLIC superpixel clusters images into segments using both colour and location of pixels. Initializing the clusters based on a regular grid results in similar size superpixels with similar colours. While this approach is very simple in nature it has proven highly effective in preserving image structure. However, the iterative nature of the algorithm results in higher computational costs.

Superpixel Lattice [6] doesn't take a clustering approach like SLIC. Instead, it iteratively subdivides the image using vertical and horizontal seams that follow image edges in order to preserve image structures. Optimal seams are found using dynamic programming. To ensure that parallel seams don't cross each other, the cost function is updated after finding each seam, placing a very high cost along previously found seams. Dynamic programming is fast at finding optimal seams but the need to update the cost function after each seam results in high computation costs. Furthermore, while Superpixel Lattice preserves a lattice structure, it doesn't necessarily adhere to a regular rectangular grid.

Finally, Pathfinder [7] also subdivides images using vertical and horizontal seams that follow image edges. Again, dynamic programming is used to find optimal seams. However, Pathfinder doesn't add any cost to ensure that parallel seams remain parallel. This increases speed because there is no need to recompute the cost function, but results in an inability to control the number or size of superpixels in the image. Furthermore, allowing seams to deviate uncontrollably results in strange diagonal edged artefacts in the superpixel structure.

III. Method

Let $\mathbf{I}$ be an $n \times m$ image. The underlying challenge of superpixels is to subdivide $\mathbf{I}$ into a set of spatially-connected regions. The pixels within each region are perceptually similar, resulting in fewer regions than the original nm pixels while preserving the original image structure. One strategy for solving the superpixel problem is to subdivide the image using a set of vertical and horizontal seams, as previously employed by [7] and [6].

For simplicity, we will describe dividing the image by a set of vertical seams. An identical formulation will be used for the horizontal seams. We define a vertical seam [11] in this image as:

$$\mathbf{s}^x = \{s^x(i)\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \; s.t. \forall i, |x(i) - x(i-1)| \leq 1 \quad (1)$$

where x is a mapping $x : [1, \ldots, n] \rightarrow [1, \ldots, m]$. By this definition, a vertical seam is an 8-connected, one pixel thick path from the top of the image to bottom of the image such that each row contains only one pixel. Now we must place M vertical seams

$$\mathcal{S}^x = \{\mathbf{s}_1^x, \ldots, \mathbf{s}_k^x \ldots, \mathbf{s}_M^x\} \quad (2)$$

in the image somewhere such that the image is split into $M + 1$ strips. If we place N horizontal seams in a similar way, we will end up with approximately $(M + 1)(N + 1)$ superpixels.

The simplest method is to place the seams S pixels apart (i.e., $M = \lfloor m/S \rfloor$, where m is the image width). In this way we can easily control the number and size of the superpixels. The result is simply a regular grid which is equivalent to down sampling the original image by a scale factor of $1/S$. The problem with this approach is that the structures in the image are not preserved. To preserve image structure, we must place seams such that they follow the structure in the image but strictly following image structure results in a lack of control over superpixel size.

To allow superpixels to conform to a grid structure and at the same time preserve image structure, the set of vertical seams $\mathcal{S}^x$ should minimize a cost function composed of both an image structure cost $f()$ and a grid structure cost $g_x()$.

$$\arg\min_{\mathcal{S}^x} C(\mathcal{S}^x) = \sum_{k=1}^{M} \left(\sum_{i=1}^{n} \left[f(s_k^x(i)) + w g_x(s_k^x(i), kS) \right] \right) \quad (3)$$

where w is a weighting between the image structure and grid structure terms, and S is the spacing of the grid structure. Given an image of width m and grid spacing of S, $M = \lfloor m/S \rfloor$.

One of the key difference between our approach and existing seam based algorithms like Pathfinder [7] and Superpixel Lattice [6] is the use of grid structure cost and the fact that we penalize every single point in the seam that deviates from the grid structure. Pathfinder has no strict grid structure cost other than enforcing all seed points (single point on each vertical seams) to have a minimum gap between then. The lack of grid structure cost causes the seams to deviate greatly from a straight line, resulting in diagonal artefacts in the superpixel structure as seen in Fig. 1d.

Superpixel Lattice [6] does maintain a regular lattice but the lattice cost is defined differently. Our approach penalizes every point on the seam that deviates from a fixed regular

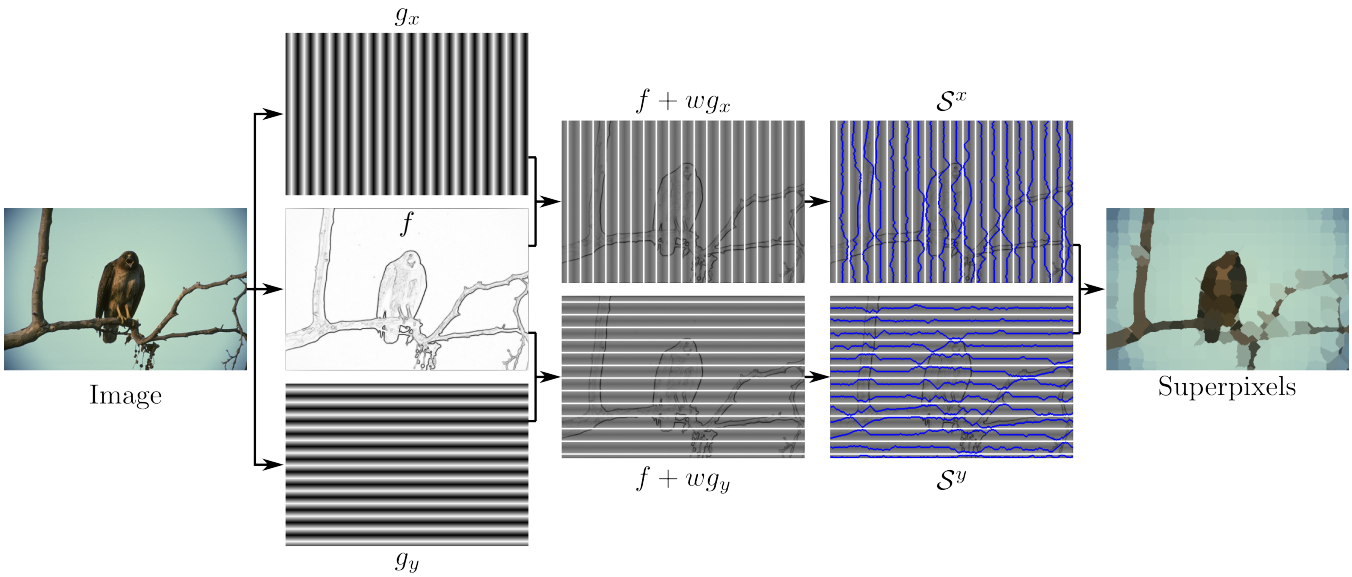


Figure 2: In our Grid Seams approach, the image structure cost f and the grid structure cost g_x or g_y are combined with a weighting parameter to produce the seam cost. All vertical/horizontal seams are found concurrently using dynamic programming. Combining the vertical and horizontal seams results in grid structured superpixels. In this example, a grid structure cost with $S = 25$ is used.

grid structure. As a result, we favour all points on the seam being as close as possible to a regular rectangular grid structure. In contrast, Superpixel Lattice penalizes seams curving (i.e. lower cost to stay in a straight line) but the location of the seam is simply bounded within a vertical strip. As a result, Superpixel Lattice doesn't guarantee a regular grid structure with similar sized cells; it can only guarantee a lattice structure where cell sizes can vary greatly. Furthermore, unlike Superpixel Lattice method, we have a direct control over how closely the superpixel structure resembles a regular grid through parameter w in (3).

A. *Image Structure Cost* $f()$

The image structure cost must follow some form of structure in the image. Structure can be anything from color to saliency to edges. For the purpose of this paper we will be using edges, which has been used in Pathfinder [7] and Superpixel Lattice [6].

$$f() = 1 - e() \tag{4}$$

where $e()$ is the normalized absolute magnitude of the edge derivative obtained using the Sobel detector [12]. We illustrate the Image Structure Cost in Fig. 2.

B. *Grid Structure Cost* $g_x()$

The grid structure cost penalizes each point $(x(i), i)$ in the k^{th} seam $\mathbf{s}_k^x$ based on how much it deviates from the grid location kS. In order to maintain a grid structure the grid structure cost must also ensure that parallel seams don't cross each other. Therefore we define grid structure cost as:

$$g_x(s_k^x(i), kS) = \begin{cases} \frac{\min |x(i) - kS|}{S/2} & |S/2 - |x(i) - kS|| \geq 1 \\ \infty & \text{else} \end{cases} \tag{5}$$

where S is the regular grid spacing and ∞ is a fixed large cost value. The use of ∞ cost occurs at every S pixels along the image starting at column $S/2$ and it ensures that no vertical seams cross each other. We illustrate the grid structure cost over the image in Fig. 2.

Our grid structure cost function ensures that seam k doesn't cross any other seams without needing to know the locations of the other seams. This allows us to obtain a solution for all the vertical seams simultaneously. In contrast, Superpixel Lattice is an iterative approach. At each iteration a single seam is found and the location of the seam is used to update the cost function to ensure consecutive seams don't cross any seams that have already been found. This results in much slower performance for Superpixel Lattice.

C. *Minimizing* $C(\mathcal{S}^x)$

Minimizing (3) is complex combinatorial optimization problem. However, note that grid structure cost (5) for all points on seam $\mathbf{s}_k^x$ is ∞ at $x(i) = kS \pm S/2$. As a result, all points on seam $\mathbf{s}_k^x$ are bound between $kS \pm S/2$ (i.e. $kS - S/2 < x(i) < kS + S/2$) and no other seam can have a point between $kS \pm S/2$ because that would result in the seam crossing the ∞ cost. As a result, we can independently find the optimal seam $\mathbf{s}_k^x$ for all k using the

dynamic programming optimization technique proposed in [11]. We give a brief overview here.

First, the cumulative minimum energy E for all possible connected seams is computed as:

$$E(i,j) = D(i,j) + min(E(i-1,j-1), E(i-1,j), E(i-1,j+1)) \quad (6)$$

where $D(i,j) = f(i,j) + wg_x(i,j)$ (Fig. 2). Computation of E starts at row $i = 2$ and for the first row $E(i = 1, j) = D(i = 1, j)$. Once E is computed, we need to find the minimum cost seams $\mathbf{s}_k^x$ where $k = 1, \ldots, M$. The minimum values between $kS \pm S/2$ on the last row of E will indicate the end location of seam $\mathbf{s}_k^x$:

$$s_k^x(i = n) = \arg\min_t E(n,t) \quad (7)$$

$$\text{where } t = kS - S/2, \ldots, kS + S/2 \quad (8)$$

Backtracking on E starting at these minimum locations will give all the vertical grid seams $\mathcal{S}^x$.

D. Horizontal Seams

Horizontal seams are treated identically to vertical seams and obtained in the same manner. For implementation efficiency, horizontal seams can be treated as vertical seams using $\mathbf{I}^T$, the transpose of the original image.

IV. Results

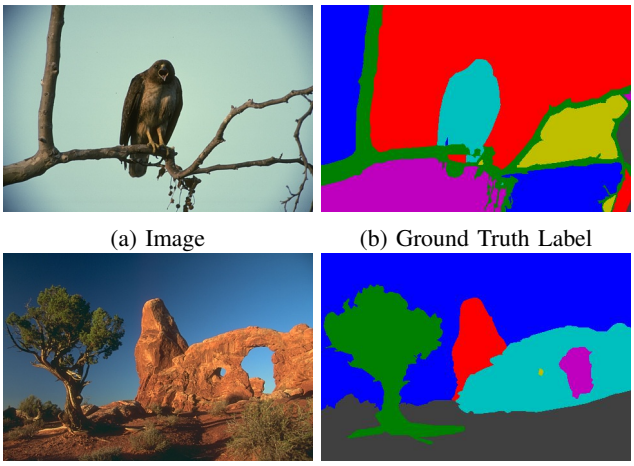
To evaluate the effectiveness of the proposed Grid Seams superpixel algorithm against existing state-of-the-art superpixel methods in a quantitative manner, we evaluate superpixel approaches' effectiveness as a precursor to segmentation using the Berkeley Segmentation Dataset (BSD) [13]. The first 50 images of the BSD test set were used for evaluation purposes, the same test set that was used in [7]. The test set consists of colour images of size 480 by 320 pixels. For each of the 50 images, there is an accompanying set of manual segmentations that were annotated by multiple users. Some example images from the BSD are shown in Fig. 3.

A. Quantitative Evaluation Metric

For quantitative evaluation of the tested superpixel methods, we use the mean accuracy measure as outlined by Moore et al [6]. All pixels in each superpixel are assigned to the ground truth segment to which the superpixel has the greatest overlap. Accuracy is computed as the percentage of correctly labelled pixels in the image:

$$Accuracy = \frac{TP}{mn} \quad (9)$$

where TP are the true positives or correctly labelled pixels in the image and mn is the image size in pixels.



(a) Image (b) Ground Truth Label

(c) Image (d) Ground Truth Label

Figure 3: Example images and ground truth from BSD [13].

This accuracy measure can be interpreted as the performance of an ideal classifier. Example error maps, indicating white for correctly labelled pixels (TP) and black for incorrectly labelled pixels, are presented in Fig. 4. It is important to note that as the number of superpixels in the image increases, the mean accuracy will tend to 100%. The accuracy is averaged over all manual annotations per image (because the BSD has multiple annotations for each image) and all 50 images from the BSD to get the mean accuracy.

B. Algorithm Setup

The proposed Grid Seams algorithm is compared to three different superpixels methods: SLIC superpixels [5], Pathfinder [7], and Superpixel Lattice [6]. The tested superpixels methods are configured as follows:

Grid Seams – There are only two parameters to our proposed Grid Seams algorithm: 1) the spatial weight w in (3), and 2) superpixel size S in (3). To evaluate the performance of Grid Seams as a function of superpixel size, we fix the spatial weight $w = 0.01$ and the superpixel size is varied within the range of $(5 - 25)$. However, in section IV-F we take a brief look at the qualitative effect of varying the spatial weight w.

SLIC Superpixels [5] – We use the C++ implementation provided by the authors[1] with the default parameters and the RGB colour space option. In addition to the baseline SLIC superpixels, without any modifications based on the original code (which we will refer to as SLIC), we also evaluated against a modified SLIC superpixels (which we will refer to as SLICfast). SLICfast is designed for computational efficiency via one minor modification. For SLICfast the iterative k-means process is only run for 1 iteration instead of

[1] http://ivrg.epfl.ch/research/superpixels

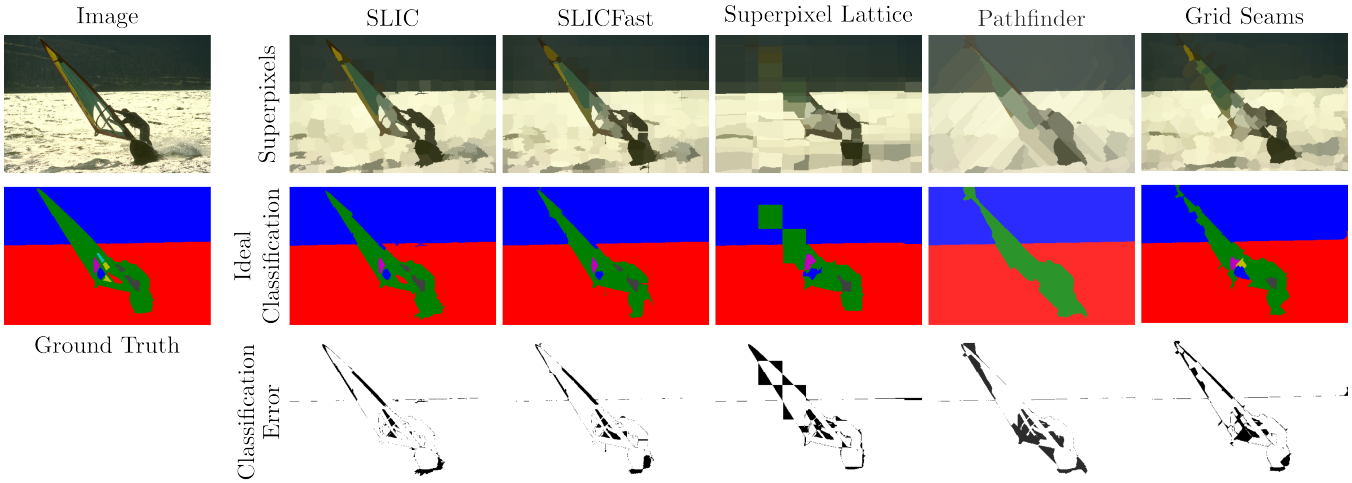


Figure 4: Outline of the evaluation process. All pixels in each superpixel are assigned to the ground truth segment whith which the superpixel has the greatest overlap. This can be interpreted as the performance of an ideal classifier and we measure the accuracy as the percentage of correctly classified image pixels. Error in the ideal classification, as outlined in black on the classification error images, is the result of the superpixel algorithm's failure to capture the image structure. All methods were run to produce about 250 superpixels for this image. As can be observed, the Grid Seams method results in lower error than Superpixel Lattice and Pathfinder, and only slightly higher error than SLIC.

the 10 used in the original SLIC code. There is a significant increase in computational cost for running 10 iterations in exchange for a small increase in accuracy.

Pathfinder [7] – We use the Java implementation provided by the authors[2]. All default parameters present in their code were used.

Superpixel Lattice [6] – We use the precompiled C++ code with the matlab interface provided by the authors[3]. All default parameters present in their matlab interface function were used.

For consistency in the performance analysis, all methods that require the computation of an edge map (Grid Seams, Pathfinder and Superpixel Lattice) make use of the Sobel edge detector [12]. Sobel edge detector was chosen because it is a standard, simple and fast edge detection method, and helps to better focus the evaluation on the effectiveness of the actual superpixel methods themselves.

For consistency in the timing analysis, all methods that require the computation of a grayscale image and/or edge image (Grid Seams, Pathfinder, Superpixel Lattice) will have the time needed to compute the grayscale image and/or edge image included in their timing analysis.

Finally, again for consistency in the timing analysis, all timing results reported here include the time needed to compute a superpixel label image. The rationale behind this is that all methods have different post-processing steps to obtain a label image: SLIC has a connectivity post processing step and Grid Seams, Pathfinder, and Superpixel Lattice require finding the intersections of horizontal and vertical seams.

C. Quantitative Evaluation

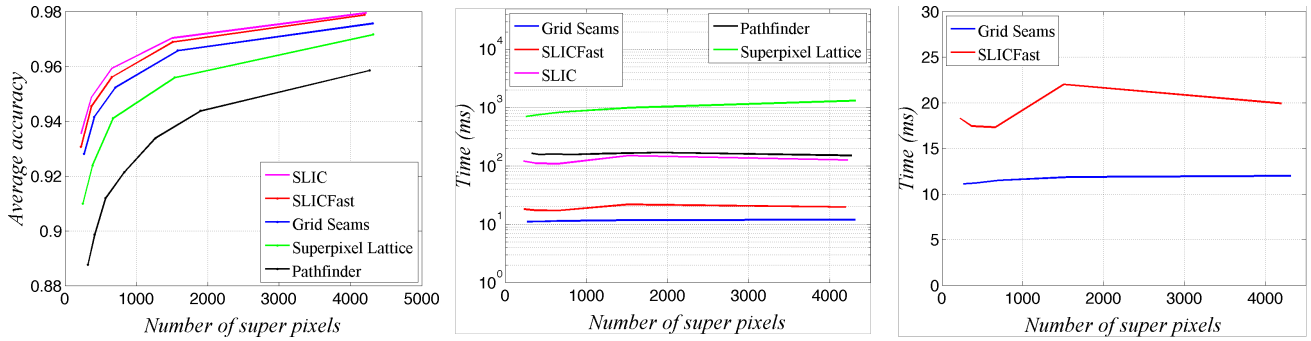
We plot the mean accuracy as a function of the average number of superpixels in the image in Fig. 5a. Note that SLIC, SLICfast, Grid Seams, and Superpixel Lattice have a parameter that has good control over the number of superpixel per image, as seen by consistent testing points on the plot. Pathfinder, on the other hand, has no direct control of the number of superpixels returned. As a result, we change Pathfinder's gap parameter till we get the desired ranged in the number of superpixels.

It can be observed that SLIC has the best accuracy but SLICfast has just a slight drop in mean accuracy even though only one iteration of clustering was applied. Closely following SLIC is the proposed Grid Seams method. It is important to note that, though Grid Seams has slightly lower accuracy than SLICfast, it is about 1.7x faster than SLICfast (timing analysis is discussed later in section IV-E).

It can also be observed that the Superpixel Lattice approach achieves lower accuracy than Grid Seams. However, it should be noted that the performance of Superpixel Lattice as reported in [6] is higher due to the use of a more advanced edge detection technique [14]. For consistency we use the same, fast edge detection technique for all methods. The use of different edge detection methods will have an impact on the performance of all methods using edges (Grid Seams

[2]We thank the authors for providing us the code.

[3]http://web4.cs.ucl.ac.uk/research/vis/pvl/index.php?option=com_content&view=article&id=76:superpixel-lattices-code&catid=49:downloads&Itemid=62



(a) Accuracy vs Superpixel (b) Timing vs Superpixel (c) Timing vs Superpixel

Figure 5: Mean accuracy and average timing result for running various superpixel algorithms on 50 images from the BSD [13]. BSD dataset contains colour images of size 480 by 320 pixels. Grid Seams is faster than all tested methods while achieving fairly high accuracy in preserving image structure when compared to state-of-the-art methods.

and Pathfinder) not just Superpixel Lattice, and would be an interesting point of investigation in future work.

D. Qualitative Evaluation

We show some examples of the superpixel results achieved using the tested methods in Fig. 6, where each superpixel is filled in by the average RGB value from all pixels that belong to the superpixel. We magnify portions of these examples to better illustrate the various behaviours of the different superpixel approaches.

From Fig. 6 we can see that both SLIC and SLICfast are better able to preserve the details of the image structure than Pathfinder and the proposed Grid Seams method. While visually appealing, this behaviour might not be desirable in all applications of superpixels. For example, multiscale approaches [15] that utilize superpixels as a means for content simplification might require details of image structure to appear only at certain scales.

Methods like Pathfinder can't control superpixel sizes or number of superpixels per image, resulting in a large variance in superpixel sizes. Furthermore, it can be observed that Pathfinder introduces diagonal lines in the resulting superpixel images due to strictly following edge information as seen in Fig. 6. Grid Seams, even though it follows a similar strategy as Pathfinder, is able to easily overcome these drawbacks due to the grid structure cost.

Superpixel Lattice [6], also preserves a lattice structure like our method but as seen in Fig 6 their results are much more pixelated in appearance and also has higher variation in superpixel sizes when compared to Grid Seams. Grid Seams has better behaviour due to the penalization of every point in seams for deviating from a global grid structure.

E. Timing Analysis

Timing results were produced on a 2.53 GHz Intel Core 2 Duo machine. We plot the average time to process the images[4] in the BSD dataset for different numbers of superpixels in Fig. 5b.

Superpixel Lattice's computation time increases as the number of superpixels per image increases. This is due to the iterative nature of the algorithm and the need to recompute cumulative cost after each seam because the location of one seam adds a cost to the following seams. Pathfinder, unlike Superpixel Lattice, does not need to recompute the cumulative cost after each seam. While being faster, the drawback of the Pathfinder approach is that there is no guarantee that the superpixels will follow a lattice.

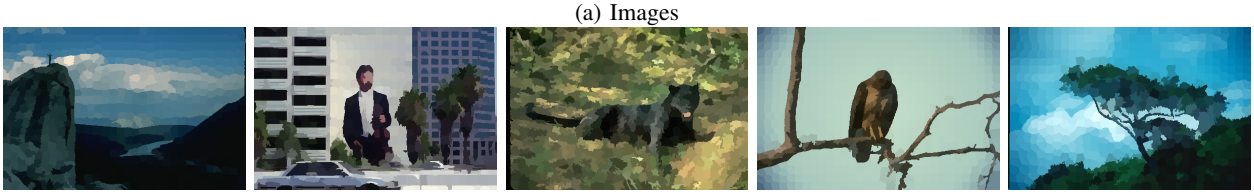
Grid Seams has a grid structure cost for each seam that is independent of other seams and thus doesn't require iterative computation of cumulative cost like Superpixel Lattice. However Grid Seams can still find similar sized superpixels just like Superpixel Lattice method. As a result, Grid Seams is a much faster algorithm than Superpixel Lattice or Pathfinder.

However, it should be noted that there is an inherent bias in the timing analysis because Pathfinder code is in Java and Superpixel Lattice, while coded in C++, is called through a matlab interface. Both of these algorithms should perform faster if implemented in C++ like Grid Seams and SLIC superpixel. However, both have lower accuracy performance than Grid Seams (Fig. 5a) and Superpixel Lattice requires iterative update of cumulative cost which Grid Seams does not.

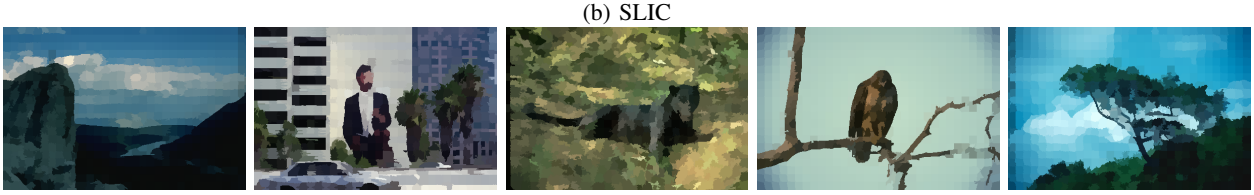
In comparison to SLICfast, Grid Seams has on average 1.7x speed improvement for a small drop in accuracy. While the 1.7x improvement, which translates to about 8ms, might seem like a small improvement, for real-time applications it will still be a significant improvement. Consider a real-time system processing video at 15 frames per second (fps) at 480 by 320 resolution. Each frame must then be processed in 67 ms. SLIC, which requires about 19 ms/frame, constitutes

[4]Recall all images are 480 x 320 in size

(a) Images

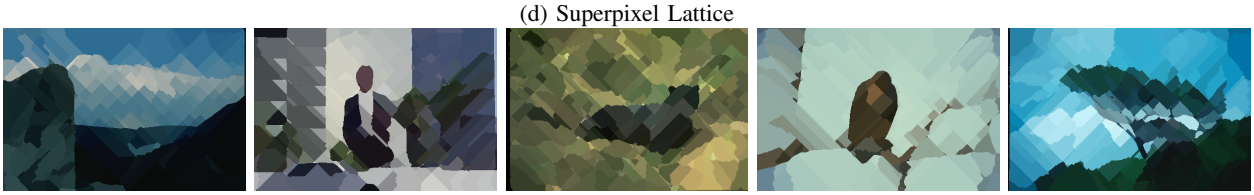


(b) SLIC



(c) SLICfast

(d) Superpixel Lattice



(e) Pathfinder

(f) Grid Seams

Figure 6: Examples of superpixel algorithms where each image is divided into approximately 700 superpixels. Pathfinder, which doesn't have any spatial constraints, has seams that strictly follow image edges resulting in diagonal artefacts. Superpixel Lattice has a loose spatial constraint, but still has some pixelation artefacts. Grid Seams and SLIC are better at preserving image structure.

28% of the available time while Grid Seam, which requires about 12 ms/frame, constitutes only 18% of the available time. As a result Grid Seams is saving 10% of the available time to process a frame. If we consider a 30 fps system (standard recording frame rate for video cameras), we are looking at 20% reduction in total available time for Grid Seams computation when compared to SLICfast.

F. Spatial Weight

The spatial weight affects how closely the returned superpixels represents a rectangular grid. Higher the weight w in (3), the closer the superpixels follow a regular rectangular grid (Fig. 8) because the edge following nature of seam carving is ignored. The lower the weight w in (3) the closer the seams follow edge information. In this way the behaviour is very similar to the spatial weight used by SLIC [5].

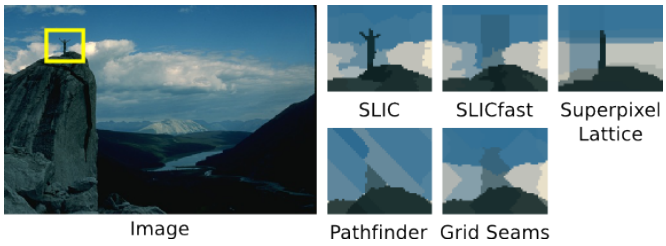
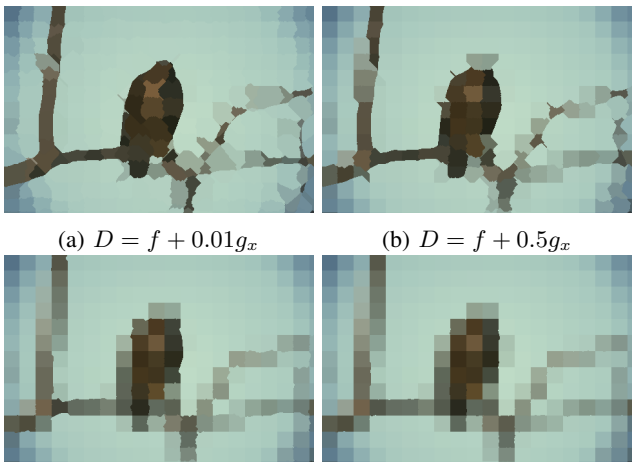


Figure 7: Zoomed-in view to illustrate different algorithms' ability to preserve image structure details. SLIC is better able to preserve details compared to most other methods.



(a) $D = f + 0.01g_x$ (b) $D = f + 0.5g_x$

(c) $D = f + 2.0g_x$ (d) $D = f + 4.0g_x$

Figure 8: Example results of Grid Seams as the grid cost weight increases. The higher the grid cost, the more closely the superpixels resemble a regular grid. The grid spacing was set at $S = 25$ giving approximately 250 superpixels.

V. Conclusion

We presented, Grid Seams, a seam carving approach to superpixel generation that is faster than existing methods while achieving similar accuracy to state-of-the-art superpixel methods. The speed and accuracy are made possible by combining image structure cues with a global grid structure cue. We hope that this fast superpixel algorithm will promote the use of superpixels for real-time applications such as background subtractions and video processing.

Acknowledgement

The authors would like to thank NSERC, Ontario Ministry of Research and Development, and Canada Research Chairs program.

References

[1] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *International Conference on Computer Vision*, Sept 2009.

[2] C. Zitnick and S. Kang, "Stereo for image-based rendering using image over-segmentation," *International Journal of Computer Vision*, vol. 75, no. 1, 2007.

[3] Z. Li, X.-M. Wu, and S.-F. Chang, "Segmentation using superpixels: A bipartite graph partitioning approach," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2012.

[4] A. Schick, M. Bauml, and R. Stiefelhagen, "Improving foreground segmentations with probabilistic superpixel markov random fields," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012.

[5] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, Nov 2012.

[6] A. P. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[7] F. Drucker and J. MacCormick, "Fast superpixels for video analysis," in *Workshop on Motion and Video Computing*, Dec 2009.

[8] J. Steiner, S. Zollmann, and G. Reitmayr, "Incremental superpixels for real-time video analysis," *Proceedings of the Computer Vision Winter Workshop*, 2011.

[9] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, Dec 2009.

[10] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, Sep. 2004.

[11] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Transactions on Graphics*, vol. 26, no. 3, Jul. 2007.

[12] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[13] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings of the International Conference on Computer Vision*, vol. 2, July 2001.

[14] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006.

[15] H. Liu, Y. Qu, Y. Wu, and H. Wang, "Class-specified segmentation with multi-scale superpixels," in *Computer Vision - ACCV 2012 Workshops*, ser. Lecture Notes in Computer Science, J.-I. Park and J. Kim, Eds., 2013, vol. 7728.