EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer
(Guest Editors)

Volume 37 (2018), Number 2

Flexible Live-Wire: Image Segmentation with Floating Anchors

B. Summa[1], N. Faraj[1], C. Licorish[1], and V. Pascucci[2]

[1]Tulane University, United States
[2]SCI Institute, University of Utah, United States

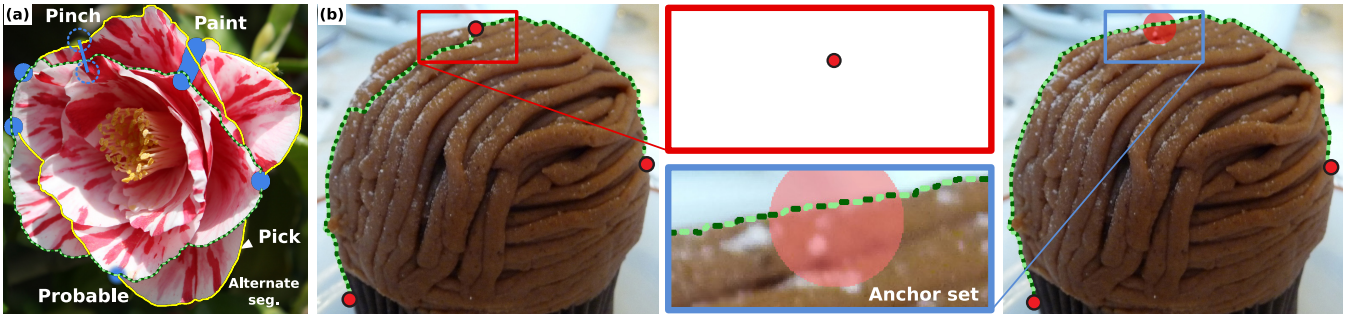

Figure 1: *Our novel generalization via floating anchors enables the flexible design of input and interactions for Live-Wire image segmentation. (a) The flower can be segmented with region painting, line drawing using a pinch interaction, a probability field about a user's click, or by picking from a set of alternate segmentations. (b) Traditional Live-Wire allows for only pixel-level anchor nodes which may be too precise for a user. A slight miss (left, top) in setting an anchor node causes an erroneous segmentation (red). Our approach allows for user controlled precision through anchor sets. A paint anchor at the same location (right, bottom) produces the correct segmentation (blue). Both anchors are valid inputs in our generalization.*

Abstract

We introduce Flexible Live-Wire, a generalization of the Live-Wire interactive segmentation technique with floating anchors. In our approach, the user input for Live-Wire is no longer limited to the setting of pixel-level anchor nodes, but can use more general anchor sets. These sets can be of any dimension, size, or connectedness. The generality of the approach allows the design of a number of user interactions while providing the same functionality as the traditional Live-Wire. In particular, we experiment with this new flexibility by designing four novel Live-Wire interactions based on specific primitives: paint, pinch, probable, and pick anchors. These interactions are only a subset of the possibilities enabled by our generalization. Moreover, we discuss the computational aspects of this approach and provide practical solutions to alleviate any additional overhead. Finally, we illustrate our approach and new interactions through several example segmentations.

CCS Concepts

•***Computing methodologies*** → ***Image segmentation;*** •***Human-centered computing*** → *Interaction techniques;*

1. Introduction

In this work, we introduce Flexible Live-Wire, a generalization of the Live-Wire interactive segmentation technique with floating anchors. This approach replaces the standard anchor node with a more general anchor set, thereby allowing novel primitives and interactions for Live-Wire input. Segmenting an image to extract objects or features is a fundamental operation that can be seen throughout the arts and sciences and found in many applications and workflows. Given the variability in image data and often the subjectiveness of what constitutes a good segmentation, interactive segmentation algorithms are particularly useful by allowing users to guide a (semi-)automatic segmentation. Popular approaches [BVZ01, Gra06] produce fast, automatic segmentations based on user input (painting, approximate boundaries, etc.). This input is often the frontend to an automatic segmentation and is therefore somewhat disconnected from the final segmentation produced. While this can be acceptable for many situations, users often want more direct control of the construction and therefore use techniques like Snakes [KWT88, BM92] or Live-Wire [MB95, MB98] to semi-automatically build a segmentation. Live-Wire can be found in a wide variety of applications with ver-

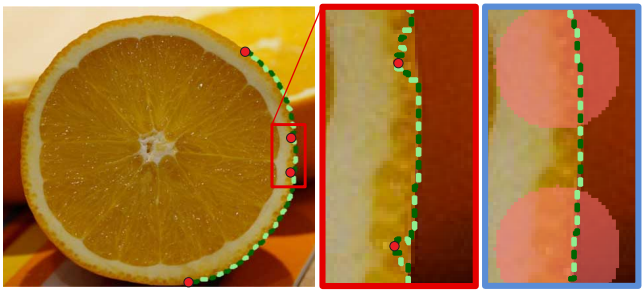

Figure 2: *Coarse, imprecise user interactions can lead to erroneous segmentations with traditional anchor nodes and is often only apparent at the finest levels of resolution. Our flexible approach alleviates the disconnect between a user's input precision and the input precision required by the technique.*

sions of the algorithm included in many image processing software including the magnetic lasso in Adobe's Photoshop [Ado17] and the intelligent scissor tool in GIMP [GIM17].

In Live-Wire, users semi-automatically guide a segmentation by the addition and manipulation of *anchors* (also called seeds [MB95]). For disambiguation, we will refer to these inputs as *anchor nodes* in this text. These anchor nodes, whether being directly set by a user or automatically set by a system, are required to be at pixel resolution for the optimization. This leads to the restrictive user interaction of precise single-point selections. There is often a disconnect between the resolution a user can quickly and reliably provide via interactions and the resolution required by anchor nodes, with the former being typically much coarser than the latter. This is especially problematic for touch devices due to the imprecision of touch input and occlusion of the anchor nodes by a user's finger. Sometimes a slight miss in anchor node placement can generate a large deviation in the segmentation (Figure 1b). In addition, some poor segmentations due to imprecise anchor node placement are not apparent unless inspected at higher resolutions (Figure 2). Often these problems are alleviated through a user segmenting an image at a high zoom level, where their input precision is close to precision required by anchor nodes. This zoom comes with tedious scrolling and loss of context of the full segmentation. In other words, a user must adjust to their input strategy to conform to the required precision of the technique.

A better approach would remove the restriction on anchor node precision enforced by the optimization. A more flexible approach would allow a user to dictate their desired input precision and the optimization should conform to that precision. In this work we have designed a more general version of Live-Wire called *Flexible Live-Wire*. Through the use of our *floating anchors*, user input is no longer limited to pixel-level anchor nodes, but will be a more general set of nodes, which we call an *anchor set*. The generalization provides the same functionality as the traditional approach but also allows for a larger set of user interaction primitives with varying precision. We experiment with this new flexibility to design new Live-Wire primitives for different inputs and precision where an anchor set defines lines or areas in an image (*paint*, *pinch*, and *probable* anchors). See Figure 1a. Furthermore, our anchor sets are ideally situated to complement previous work [STP17] on extracting alternate segmentations. This extraction provides a novel *pick* anchor where a user selects from several segmentation choices. These proposed interactions are only a subset of the many possible novel inputs enabled by our generalization.

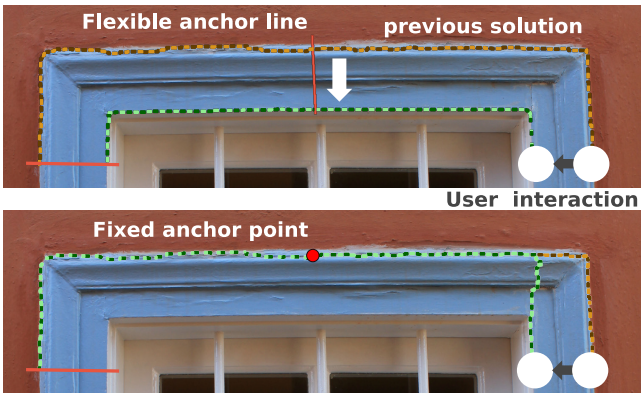

Figure 3: *The imprecision of a user's interaction should be maintained. (top) Imprecise anchors allow the segmentation optimization to be flexible and adjust to new input. (bottom) Approaches that take imprecise input but keep the underlying algorithm the same fix locations and can lead to a poor segmentation.*

In particular, the major contributions of this work are:

- A generalization of Live-Wire using novel floating anchors that relaxes the traditional, pixel-precise anchor nodes and enables anchor sets that can span arbitrary regions with no restrictions on size or even connectedness;
- Examples of new Live-Wire interactions tailored to specific use cases for our anchor sets including: paint, pinch, pick, and probable anchors;
- A detailed analysis of the efficiency of the method including practical solutions that maintain interactivity despite the added capabilities; and
- Results illustrating interesting use cases for our example anchor set primitives.

2. Related Work

In this section, we detail popular interactive image segmentation approaches and their requirements for user input. First, region growing approaches such as ones based on watersheds [BM92, CBNC09], random walks of a graph [Gra06], or minimum geodesic distances [BS09] require a set of initial seed locations as the basis for their optimizations. There are no constraints on the size, shape, or connectedness of these seeds. To this, the most common interaction for a user is to paint or scribble the seed locations [Gra06, BS09, YCZL10]. This free-form input is both intuitive to users and allows manual or automatic [WAC07] variance in precision based on their task by, for example, adjusting the size of the brush in a painting interaction. Next, interactive techniques based on a minimum cut of a graph constructed from an image [BVZ01, BK04], graph cuts, also requires a similar set of seed locations to initialize their optimization. Also, similarly to region-based approaches there are no constraints on this input and user can interact with the technique with free-form painting and scribbles [BJ01, RKB04, LSS09]. In fact, there has been work to relax

the amount of painting required for these approaches and, for example, have a user only supply a simple bounding box around an object [RKB04]. Finally, snakes [KWT88] segmentations build a solution from an approximate segmentation boundary provided by a user which is then optimized to the final segmentation. Precise or imprecise input is naturally handled with this approach. These techniques, through their lack of restrictive input lead to the design of flexible segmentation environments and easily transfer to use cases where the precision of the input is potentially low.

Live-Wire [MB95, MB98] is a fast and intuitive segmentation algorithm that can be found in a wide variety of medical [FU97, HYML05], scientific [MJS*04, HBG*11, Ali14], and artistic [MB95, MB98, STP12] applications. Compared to the competing segmentation algorithms, Live-Wire's required input is restrictive. As we will describe in Section 3, the core of the technique requires pixel-level seed locations as the basis for its segmentation. Techniques have tried to improve a user's experience by decreasing the complexity of the underlying optimizations [FUS*98, FUM00] or through automatic anchor node setting [MB95, MB98] when a portion of a segmentation has stabilized. Despite this work, the core input requirement for the technique has not changed. For example, Adobe's Photoshop [Ado17] constructs a Live-Wire-like segmentation by automatically setting anchor nodes in a fixed neighborhood about a user's cursor. While such an approach allows for a level of flexibility in input, due to the core algorithm's requirement, a user must set at at least one precise anchor node to start and often many additional anchor nodes in the cases where the automatic routine does not provide the desired segmentation. In addition, these automatically set anchor nodes restrict a segmentation to be precise even if the input wasn't. As illustrated in Figure 3, this can cause a segmentation to be pigeonholed in a sub-optimal configuration.

There has also been work on integrating both minimum cut and Live-Wire approaches into a single segmentation routine [SdMF14] or in unifying minimum cut and Live-Wire segmentations [SGSP15]. In this way, these works add a level of flexibility in input by adding a painting interaction, although both maintain the requirement for precise input when a user manipulates the Live-Wire components. In addition, there has been work that has combined a graph cuts segmentation with an approximate boundary with anchors [LSTS04]. In this case, there is no Live-Wire optimization and the boundary is used to drive a graph cuts segmentation. Finally, there has been work in uniting active contouring and Live-Wire [LMT06], using Live-Wire anchors to provide hard and soft constraints for a snakes optimization. Hard constraints, again, require pixel-level precision, while soft constraints introduce a level of imprecision. Soft constraints only *suggest* and do not enforce locations for the segmentation in this approach. Our work gives users more control in that a segmentation is required to travel through the imprecise anchors. All assume and are limited to the standard interactions for region (painting for a cut) and boundary (nodes for Live-Wire) delineation. In this work, we provide a new generalization of Live-Wire that enables unrestricted, free-form, and flexible interactions through a generalization. Any set of nodes (including, but not limited to, painting) can denote a Live-Wire anchor. G-Wire [Kan05] is a generalization that adds snake-like contouring to Live-Wire by adding a boundary energy

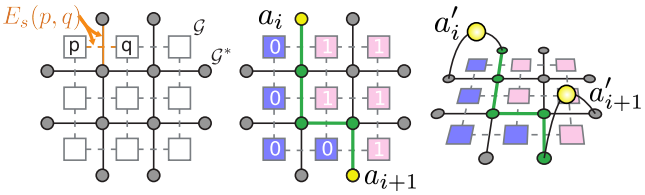

Figure 4: *(Left) The pixel graph $\mathcal{G}$, its dual $\mathcal{G}^*$, and energy E_s. (Middle) On the dual, the min path (green) provides the minimal segmentation (blue, pink) given two anchor nodes (yellow). (Right) Our floating anchors independent of graph nodes but connected to sets of nodes.*

to the standard optimization on a multi-dimensional graph, but the core Live-Wire formulation remains the same.

3. Live-Wire Generalization

Live-Wire. Live-Wire segmentations can be considered a binary labeling of a planar graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ formed from an image, where the nodes, $\mathcal{V}$, are image pixels and the edges, $\mathcal{E}$, encode all pairs of 4-neighbor pixels, (p,q). This labeling, L, is computed by the minimization of an objective function, $E(L) = \sum_{(p,q)\in\mathcal{E}} E_s(p, l_p, q, l_q)$, where E_s is the cost incurred from neighbors p and q having labels l_p and l_q respectively with $l \in \{0,1\}$. If $l_p == l_q$, E_s is typically 0. Otherwise ensures (non-)smoothness of the boundary between labels. For example, a simple E_s for object segmentation maximizes the gradient across the boundary with $E_s(p, l_p, q, l_q) = 1/(\|\mathcal{I}_p - \mathcal{I}_q\|)$, where $\mathcal{I}_p$ and $\mathcal{I}_p$ are the pixel values at locations p and q respectively when $l_p \neq l_q$.

Rather than operate on $\mathcal{G}$ directly, Live-Wire segmentations are computed on a dual graph $\mathcal{G}^*$. Each edge of $\mathcal{G}^*$ is weighted with the E_s cost of its corresponding orthogonal edge in $\mathcal{G}$. See Figure 4 (left). On this dual graph, a minimum cost path between a pair of nodes produces a minimal partial segmentation between each node in terms of E_s. Therefore Live-Wire requires a set of *anchor nodes* as input, which are provided by user interaction. Given a set of anchor nodes, $\{a_i, i \leq k\}$, Live-Wire segmentations are formed from a set of $k-1$ minimum paths between pairs of consecutive anchor nodes, $\{\mathcal{P}_i, i \leq k-1\}$ where $\mathcal{P}_i$ is the minimum path between a_i and a_{i+1}. See Figure 4 (middle). The segmentation is finalized when $a_1 == a_k$ with $\cup_{\forall i \leq k-1} \mathcal{P}_i$ producing the boundary of the minimal segmentation of $\mathcal{G}$ in terms of E_s under the condition that the boundary must pass through the anchor nodes.

The most common interaction for Live-Wire is a user adding the anchor nodes sequentially to build a segmentation. To this, a minimum path tree is computed for each anchor node, $\mathcal{T}_i$ for anchor node a_i. Each tree can be computed in linear time in the number of pixels [MB95, MB98] and encodes the minimum paths and their costs between a_i and all other nodes in our graph. This includes its sequential anchor node neighbors a_{i-1} and a_{i+1}, $\mathcal{P}_{i-1}$ and $\mathcal{P}_i$ respectively. Therefore the paths of a Live-Wire segmentation can be extracted from these trees. In fact, given a tree $\mathcal{T}_i$, a user can add an additional anchor node a_{i+1} and query $\mathcal{P}_i$ as a simple walk of the tree. After a user is satisfied with the location of a_{i+1}, $\mathcal{T}_{i+1}$ is computed in anticipation of an additional anchor node. This pre-computation and instant path lookup yields a highly

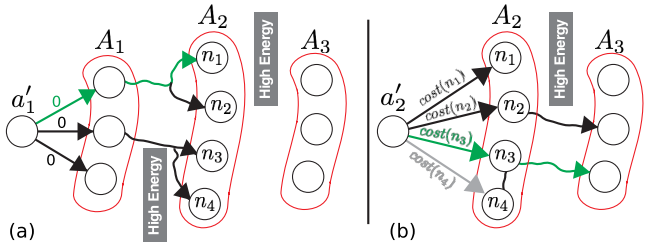

Figure 5: *The order and dependencies in our new generalization. (a) to compute the minimum segmentation (green) between A_1 and A_2 we compute the minimum path tree for a'_1, $\mathcal{T}'_1$. (b) To correctly compute the minimum segmentation to A_3 the edges from a'_2 to A_2 are weighted with the path costs from $\mathcal{T}'_1$. Note that this flexibility allows the segmentation to avoid areas of high energy and not be forced into a poor segmentation..*

interactive technique. Finally, maintaining these trees can allow a user to edit sequentially older anchor nodes [STP12] by noting that the paths on either side of an anchor node, $\mathcal{P}_{i-1}$ and $\mathcal{P}_i$ for a_i, can also be provided via the trees of an anchor node's neighbors, a_{i-1} and a_{i+1} respectively. As you can see, the requirements of the algorithm assume an input that is at the pixel-level, a precision that is prohibitive and possibly undesirable for users. A better approach is to relax this requirement and allow a varying level of precision.

Generalization. Let us begin our generalization by removing the condition that roots of our tree computations and a user's input are equivalent nodes in $\mathcal{G}^*$, which to this point has been an intrinsic assumption of the technique. With this disconnection, our new *floating* anchors $\{a'_i, i \leq k\}$ that drive the segmentation computation are now independent of the nodes in our graph. We attach each with zero cost edges to a set of nodes in $\mathcal{G}^*$, $\{A_i, i \leq k\}$. Each set, A_i, is called an *anchor set*. It is easy to see that the standard Live-Wire anchor nodes are just a special case of this new formulation with $|A_i| == 1, \forall i$. In this new augmented graph, a minimum path computed between pairs of floating anchors, $\mathcal{P}'_i$ for a'_i and a'_{i+1}, will produce a minimum cost partial segmentation under the condition that the segmentation must pass through the corresponding anchor sets, A_i and A_{i+1}. See Figure 4 (right). Note that in this formulation A_i does not necessarily need to be equal to A_k, just that $A_1 \cap A_k \neq \emptyset$. While the floating anchors provide the algorithmic backend, practically a user is only required to add and remove nodes from the anchor sets. Therefore, with this new formulation Live-Wire anchors are relaxed to be sets of nodes, where user input is generalized to the addition and manipulation of these anchor sets. As described in Section 4, this enables the design of novel Live-Wire input.

This simple, yet powerful, concept requires some algorithmic care to realize. With these new floating anchors, the computation of the minimum path trees, $\mathcal{T}'_i$, needs to be adjusted to guarantee that $\cup_{\forall i \leq k-1} \mathcal{P}'_i$ is the contiguous minimum path that passes through every anchor set. To achieve this guarantee, we can use the edges of our floating anchors, which need not be zero. Consider the simple case of three anchor sets in Figure 5. The tree computed for A_1, $\mathcal{T}'_1$, encodes the minimum paths and costs from A_1 to all nodes in our graph but, in particular, to the set of nodes that correspond to our second anchor, A_2. Now, in order to guarantee that the segmentation is the minimum that passes through A_2 as it continues on to

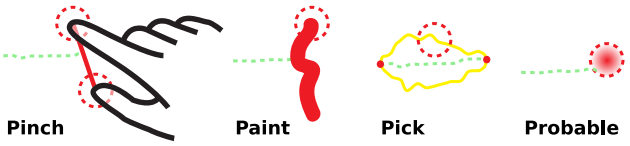

Figure 6: *This work allows the design of novel Live-Wire primitives with new interactions.*

A_3, the costs of those minimum paths need to be included in the calculation of $\mathcal{T}'_2$ for anchor A_2. Also, note that the addition of A_3 has no effect on the minimum paths from A_1 to A_2. Therefore accounting for the previous path costs can be simply accomplished by weighting the edges from a'_2 to the nodes of A_2 with those costs. Intuitively, this can be considered collapsing the previous minimum paths. Doing this for all anchor sets computes the minimal path that must pass through each set. This approach mirrors sequential anchor addition and therefore has the same complexity as the traditional approach. Note, that if relying on the traditional anchor nodes of Live-Wire, we would have needed to pick a single anchor node given, for instance, the imprecise input of A_2. A logical choice would have been n_1 in Figure 5a, which is the minimal path at that point in the computation. This restriction would have pigeonholed the computation into an area of high energy when A_3 was added. With our approach, the optimization can adapt to new input to find the optimal segmentation.

Some further adjustment is needed to allow for the editing of older anchor sets. The tree calculation described above accounts for the paths and costs in the sequentially increasing, $\mathcal{F}orward$, direction of anchor sets. If a user wishes to do more than add anchor sets to the end of the segmentation, the equivalent decreasing, $\mathcal{B}ackward$, direction needs to be also computed. This leads to two tree calculations per anchor set, $\mathcal{T}_i^{\mathcal{F}}$ and $\mathcal{T}_i^{\mathcal{B}}$ for anchor A_i. With these trees, the minimum paths associated with anchor set A_i, $\mathcal{P}_{i-1}$ and $\mathcal{P}_i$, can be found via the trees $\mathcal{T}_{i-1}^{\mathcal{F}}$ and $\mathcal{T}_{i+1}^{\mathcal{B}}$ respectively. Note that segmentation cost for nodes in A_i is the sum of the costs from both trees. This property allows a user to edit any anchor set and receive the segmentation instantaneously. Note the requirement of directed trees causes a cascade of tree computation. For an edited anchor set A_i, trees $\mathcal{T}_j^{\mathcal{F}}, \forall j \geq i$ and $\mathcal{T}_k^{\mathcal{B}}, \forall k \leq i$ will need to be recomputed. Finally, the $\cup_{\forall i \leq k-1} \mathcal{P}'_i$ computed above is the minimum path that must pass through each anchor set, but is not the minimum closed path for a final segmentation when $A_1 \cap A_k \neq \emptyset$. To properly compute the minimum segmentation, one would have to run the entire process for every node in A_1, finding the minimum path to the equivalent node in A_k. The complexity added by the directed trees and final closed path search are discussed in Section 5 with practical solutions to reduce the overhead. Pseudocode for both the fast minimum segmentation lookup and tree calculations after interaction are provided as supplemental.

4. Example Anchors

With our new approach, adding or editing Live-Wire anchors is now simply the process of adding and removing nodes from anchor sets. This generalization opens a new range of user interactions in addition to the standard anchor node. In Figure 6 and below, we will describe some novel anchors and the corresponding interactions now

possible. Note that given the generality of our approach there is no such thing as sidedness or orientation of all of the interactions below, a consideration that needed to be addressed in other interactive segmentation approaches [WAC07]. Additionally, all of the interactions below are well-suited for touch devices, where a user's input precision is typically low.

Pinch. Anchor sets can be used to represent 1D geometry. This anchor allows users to denote lines that are rasterized into an anchor set. Intuitively, these lines are *(approximately)* orthogonal to the segmentation. For this work we chose a line with 1-pixel thickness, but any thickness schemes are possible with the approach. This anchor has also a direct analog to the pinch used in multitouch interactions. Given two touches, we can form a line between each finger. In this domain, the segmentation is constrained to pass between a user's fingers. In other words, one finger is set to be inside and the other outside the segmentation. Again, given the generality of our anchor sets no specific finger orientation is required.

Paint. Each anchor set can also be one or more 2D areas. Similar to other interactive segmentation algorithms, these nodes can be denoted by a paint interaction with a user-defined brush size. Note that these areas need not be a single connected component.

Pick. One can view the varying precision of our anchor sets as adding an element of uncertainty to where a segmentation must go. An interesting extension of our formulation is to extract and present to a user sets of alternate segmentations from which they can pick. Recent work has shown how to extract these alternatives from a Live-Wire segmentation's minimum path trees [STP17]. In this approach, all pairs of consecutive anchors, a_i and a_{i+1}, and their trees, $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$, can define a field over all pixels, p, that is the minimum of the cost for a path from a_i to a_{i+1} through p. Taking the minimum cost for p over all consecutive anchor pairs forms a *min-path stability* field, which encodes both the minimum cost of perturbing a segmentation through a node but also the path associated with the perturbation. Therefore, iterating though the nodes of this field from low to high cost, provides a space to search and extract alternative segmentations. In addition, these alternative segmentations can be enforced to have uniqueness with a user parameter that denotes the percent of allowable overlap between potential alternate segmentations, which is applied greedily during extraction. Given their larger span anchor sets seem to be an ideal match for this extraction compared to the standard Live-Wire anchor nodes. Given an alternate segmentation between two anchor sets, the full alternate segmentation can be extracted by walking the trees of the remaining of the anchors. With this extraction, a user can pick from the set of alternate segmentations and this pick can be enforced by the insertion of a anchor node between the corresponding anchor sets that produced the alternate.

Probable. Finally, if we again consider the anchor sets as uncertain, we need not consider each point in the set to have uniform probability. Specifically, a probability density function can be applied to the nodes in the set. These probabilities can then applied to the floating anchor edge costs as discussed in Section 3. In this work, we've tested sets with normal (gaussian) distribution with a user-defined controlled standard deviation about a user's click/touch.

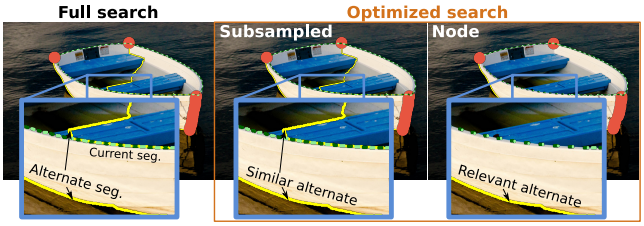

Figure 7: *The difference between searching for alternate paths in the full image, subsampling the search space every 4 nodes, and only searching the anchor set nodes. Coarse subsampling still extracts the proper alternate paths, while only using the anchor sets still finds the major paths. Image courtesy of Paul VanDerWerf.*

5. Practical Computational Aspects

Generalizations often come at the cost of increased complexity, as is the case for some aspects of our approach. The first complexity consideration is in the search for the minimum closed path in finalizing the segmentation. This involves the full recomputation of all anchor trees for all nodes in anchor set A_1, leading to a $O(k|A_1|N)$ procedure using a linear tree calculation of Mortensen et al. [MB95, MB98] where k is the number of anchor sets and N is the number of pixels. An initial approach to minimize this overhead is to reorder the anchors such that $|A_1|$ is minimized. Note that if $|A_1| == 1$, no search is necessary since it is guaranteed to be closed. While this may reduce some complexity, the search can still be a overly expensive for an interactive technique. To address this overhead, we leverage the fact that for a vast majority of cases, a user is satisfied with the current segmentation before the final closing anchor is set. In fact, searching for the final closed path at this point may have the undesirable property that the minimal closed path can change a portion or all of this segmentation. This final "pop" of the segmentation may be undesirable and frustrating for users. Rather than the expensive optimization that can result in a disconcerting shift for users, we modify this final computation in the following way. Keeping the currently computed segmentation as fixed, we connect node at the end of our path in A_k with the minimum cost path to the node at the start of our path in A_1. With this new close routine, the complexity of our technique in a typical segmentation scenario, where a user consecutively adds anchors, has the similar complexity of the standard Live-wire algorithm $O(kN)$.

The second complexity consideration is in the $\mathcal{F}orward$ and $\mathcal{B}ackward$ tree computation necessary to correctly edit older anchors [STP12] or to extract alternate segmentations for the *pick* interaction. While each direction is inherently independent and parallel, updating the trees properly would require a $O(kN)$ complexity update per anchor edit, a bottleneck to interactivity as k gets large. To this, we will leverage the fact that a user is unlikely to want to edit anchors that are very old in the interactive segmentation. Therefore we limit the tree updates and alternate segmentation extraction to be a fixed window (2) about the last set anchor. For both considerations above, since our approach is just a modification of the standard Live-Wire algorithm and tree calculations, techniques like Live-Lane [FUS*98] that limit the computation domain (N) as a function of cursor location, speed, and acceleration, could still be used as speedup strategies.

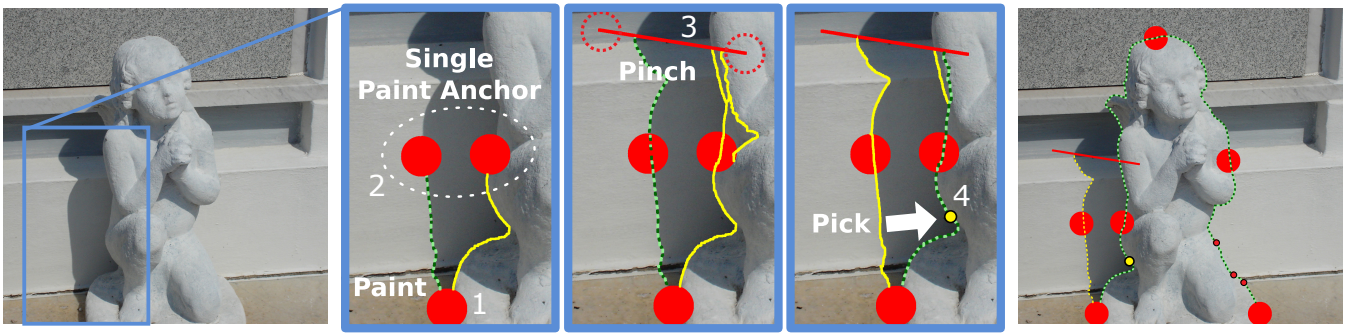

Figure 8: *The progression of a user segmentation of the statue with paint and pinch anchors. The second paint anchor has two connected components. A pick anchor is used to ensure the segmentation follows the statue rather than the shadow.*

The final complexity consideration is in the extraction of the alternate segmentations. The previous work [STP17] has shown how to extract these segmentations with $O(|\mathcal{P}^a|N)$ complexity where $\mathcal{P}^a$ is the set of alternate segmentations. In the cases $|\mathcal{P}^a|$ is not small, this extraction can become a bottleneck for interactivity. Rather than limit the number of paths extracted and thereby limit a user's choices, we instead target schemes to limit the search space, N. First, we have found that it is often unnecessarily to sample the entire image in searching for the alternate segmentations. Consider an alternate segmentation between a pair of anchors. If any of the points on that path are sampled, then the path is found. Therefore we have found practically, one can subsample (4 pixels) the search and still find the proper alternate segmentations. Finally, a scheme used in our examples is to limit the search space to only the nodes of our anchors themselves ($N = \sum_{\forall i} |A_i|$), achieving a very small N with fast extraction. While not guaranteed to find all, as our examples show it still finds the significant alternate segmentations. See Figure 7. Finally, for simplicity in picking and to keep the computation as parallel as possible, we only extract the alternate segmentation geometry between the two corresponding anchor sets. This independent calculation can cause a discontinuity in the alternates in the interior of anchors, although as the example in Figure 10 shows this discontinuity has no significant effect on the usability.

6. Results

In this section and in the companion video, we provide some interesting use cases of the new interactions enabled by our *Flexible Live-Wire* formulation. Demoed segmentations were captured on a 2.8 GHz Intel Core i7 laptop. Segmentations were computed using the simple E_s given in Section 3 to maximize the gradient and alternate path segmentations are provided using the anchor node constrained search.

Figure 9: *A user segments the boat. Using a pick anchor the user can choose to segment the interior or the entire boat.*

In Figure 1a, we show an example of a user segmenting a flower by mixing all of our new interactions. The user can add paint, pinch, or probable anchors (blue) or pick from alternate segmentations (yellow) to either include or exclude petals. Figures 1b and 2 are two examples of a user's precision not matching the pixel-level precision of Live-Wire with varying results. In Figure 1b, a user misses the boundary of the cake by only a few pixels. As you can see, due to the large gradients on the interior of the cake, this has a drastic effect on the segmentation. In contrast, a paint anchor at the same location allows this imprecision and produces the correct segmentation. Figure 2 also has imprecise interactions that at a coarse level seem to segment the orange properly. Upon inspection, you can see that the imprecisely set anchors cause portions of the orange to be missed. Again, a paint anchor at the same location resolves this issue. In Figure 3, we show why flexibility and precision of user input are useful to maintain. This is an example of the case that is illustrated in Figure 5. In the example, given two pinch constraints (red) on the window, systems using a variation of standard Live-Wire algorithm [MB95, MB98] will *cool* (fix) a portion of the segmentation to handle imprecise user input. In this example, the segmentation is fixed to the location of highest gradient which is the exterior of the window. Given an interaction for a new anchor, the fixed segmentation does not have enough flexibility to produce the right segmentation under the new constraint.

Figure 8 illustrates the flexibility of mixing the different anchors.

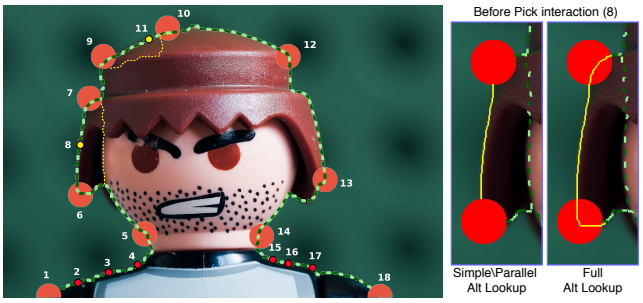

Figure 10: *A user mixes paint, node, and pick anchors to segment the toy. Anchors are numbered in order of addition. Image courtesy of Daniel Novta.*

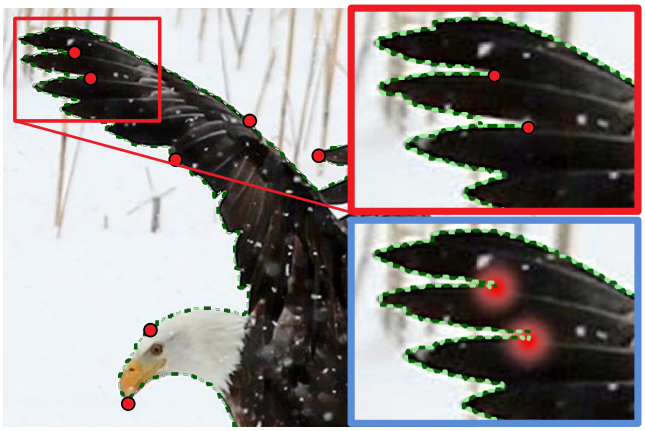

Figure 11: *Probable anchors combine the precision of anchor nodes with the flexibility of paint anchors. Here anchor nodes alone do not segment the interior of the feathers well. Probable anchors allow the optimization to achieve a better configuration. Image courtesy of Parasaran Raman.*

In this example a user initializes a segmentation of the statue with a coarse paint, followed by two pinch anchors. Under these constraints, the segmentation follows the statue's shadow. A user can correct this, if desired, by picking the alternate segmentation that follows the statue. Similarly, in Figure 9 a user segments the boat with paint anchors. The first two are placed at the stern with the third at the bow. Given this configuration, the alternate paths allow a user to quickly switch between either segmenting either the full boat or just the interior (or possibly producing a segmentation for both). Figure 10 illustrates a more sophisticated example. In this figure, a user adds a mixture of quick paint anchors (1,5,6,7,9,10-14,18), anchor nodes for areas that require more precision like where the toy's black shoulders are close to the the background color (2-4,15-17), and pick anchors to correct a segmentation that, while following the maximum gradient, is an undesirable segmentation (8,11). In Figure 11 we illustrate the probable anchor. In our testing, we found that this anchor is a nice intermediary between a node and paint anchor. This anchor provides a level of precision equivalent to a anchor node but allows the optimization some flexibility to escape from a bad local minima in the optimization. In this figure, two anchor nodes at the interior tips of the eagle's feathers give a poor segmentation. A probable anchor at the same location allows the segmentation to "pop" into a better configuration.

Table 1 documents the running times for our schemes for all of the examples in this paper. Included are the maximum time after an anchor is set for both the forward only scenario (equivalent to standard Live-Wire) and with a windowed scheme to allow for the editing/picking of the last two anchors. While there is an overhead in the windowed approach equivalent to the increased tree computation, as the companion video shows this is an acceptable delay as a user moves between constraints. In addition, we provide the average time for alternative segmentation extraction for searching the full image (subsampled every 4 nodes) and the anchor set constrained search. While the subsampled search time is acceptable for situations where a user can wait for a full search, the anchor set provides an alternate path extraction that is well suited to providing results as constraints are added. Interactive feedback of the segmentation is instantaneous under anchor addition or editing.

			Max $\mathcal{T}$		Avg. Alt	
Fig.	N	$\|A\|$	Fwd	Win.	Sub.	Anchor
Flower 1a	1200x1220	7	0.3	0.89	4.7	0.4
Cupcake 1b	1728x1296	5	0.5	1.4	2.6	0.1
Window 3	1500x1812	3	0.8	2.1	5.6	0.2
Angel 8	2304x1728	12	1.0	2.7	6.1	0.1
Boat 9	1024x764	4	0.2	0.5	0.4	0.03
Toy 10	1024x680	18	0.1	0.4	0.05	0.01

Table 1: *Runtimes (seconds) for paper examples.*

The software was evaluated via user feedback after a series of segmentation tasks with the standard Live-Wire approach and with our flexible anchors (16 users total: 7 on a touch device and 9 with a keyboard and mouse). We collected the following statistics: 81% of the users found the new anchors easier to use; 87% felt that they reached their desired segmentation faster with the flexible anchors, which is validated by recorded timings for the touch users with an average of 40% faster segmentation; and 93% found the flexibility of having different constraints useful. When asked to rank the different anchors, 93% of the users showed preference for one of our flexible anchors (e.g., all the touch users chose the paint anchor). 93% would use the new anchors if present in image processing software. The methods and results are provided as supplemental material.

7. Discussion

Given that this work only adjusts the graph with floating anchors and the order/dependency of computations, the core Live-Wire computation remains unchanged and therefore this approach can potentially leverage the large amount of work on improving Live-Wire segmentation computation [FUS*98, FUM00], energies [MB95, MB98, Kan05], or extending Live-Wire to 3D [Gra10].

This work opens new avenues of research for Live-Wire segmentations. For instance, the primitives and interactions provided in this work are only a small subset of the types input now possible with this approach given its generality. An interesting area of research is if this approach allows novel integration of node-based, *data* energy, E_d, common in other graph segmentation approaches [BVZ01, BK04]. For instance, an area anchors that are bisected by a partial Live-Wire segmentation could be used to build a color-profile for the object and background and used to drive further segmentations. In addition, hierarchical, coarse-to-fine segmentations used for performance and/or scalability are common in other interactive segmentation approaches [LSGX05, XAB07]. With traditional Live-Wire, it was unclear how best to resample a coarse anchor node to finer resolutions where the represent 2D areas. Our approach naturally fits into this type of scheme. Finally, and most interestingly, this work with its anchor sets and alternate path extraction opens interesting applications in segmentation-like, but non-traditional segmentation applications. For example, in Figure 12 (left) 4 pinch anchors are used to extract the rings of the tree. Figure 12 (right) uses 2 paint anchors: one in the center and one as four disconnected components at the exits. In this configuration,

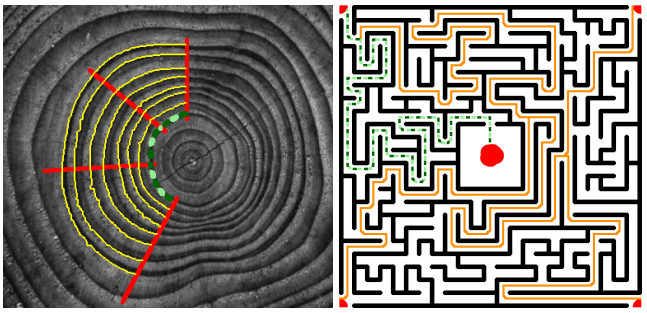

Figure 12: *Examples of non-traditional segmentations enabled by our approach. (left) Four 1D anchors along with alternate segmentations are used to segment the rings of the tree. (right) Two 2D anchors, one at the center of the labyrinth and one as a four-disconnected anchor at multiple exits, are used to find the minimum path to all exits. Labyrinth courtesy of amazeaweek.net.*

our approach can extract both the fastest path out of the labyrinth, but also all paths to all exits.

Acknowledgements

This work was supported in part by NSF:CRII 1657020, NSF:QuBBD 1664848, NSF:CGV 1314896, NSF:IIP 1602127, NSF:ACI 1649923, DOE/SciDAC DESC0007446, CCMSC DE-NA0002375, and DE-SC0010498. This work was also supported by the DOE Office of Science, ASCR under the guidance of Dr. Lucy Nowell and Richard Carlson.

References

[Ado17] ADOBE: Adobe Photoshop, 2017. 2, 3

[Ali14] ALI A.: Live-wire segmentation in agriculture. In *Computing, Management and Telecommunications (ComManTel), 2014 International Conference on* (2014), IEEE, pp. 107–110. 3

[BJ01] BOYKOV Y. Y., JOLLY M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (2001), vol. 1, IEEE, pp. 105–112. 2

[BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE TPAMI 26*, 9 (2004), 1124–1137. 2, 7

[BM92] BEUCHER S., MEYER F.: The morphological approach to segmentation: the watershed transformation. *Optical Engineering-New York-Marcel Dekker Incorporated- 34* (1992), 433–433. 1, 2

[BS09] BAI X., SAPIRO G.: Geodesic matting: A framework for fast interactive image and video segmentation and matting. *International journal of computer vision 82*, 2 (2009), 113–132. 2

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE TPAMI 23*, 11 (2001), 1222–1239. 1, 2, 7

[CBNC09] COUSTY J., BERTRAND G., NAJMAN L., COUPRIE M.: Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE TPAMI 31*, 8 (2009), 1362–1374. 2

[FU97] FALCAO A. X., UDUPA J. K.: Segmentation of 3d objects using live wire. In *Medical Imaging* (1997), International Society for Optics and Photonics, pp. 228–235. 3

[FUM00] FALCÃO A. X., UDUPA J. K., MIYAZAWA F. K.: An ultra-fast user-steered image segmentation paradigm: live wire on the fly. *IEEE transactions on medical imaging 19*, 1 (2000), 55–62. 3, 7

[FUS*98] FALCÃO A. X., UDUPA J. K., SAMARASEKERA S., SHARMA S., HIRSCH B. E., LOTUFO R. D. A.: User-steered image segmentation paradigms: Live wire and live lane. *Graphical models and image processing 60*, 4 (1998), 233–260. 3, 5, 7

[GIM17] GIMP: GNU Image Manipulation Program, 2017. 2

[Gra06] GRADY L.: Random walks for image segmentation. *IEEE TPAMI 28*, 11 (2006), 1768–1783. 1, 2

[Gra10] GRADY L.: Minimal surfaces extend shortest path segmentation methods to 3d. *IEEE TPAMI 32*, 2 (2010), 321–334. 7

[HBG*11] HÖLLT T., BEYER J., GSCHWANTNER F., MUIGG P., DOLEISCH H., HEINEMANN G., HADWIGER M.: Interactive seismic interpretation with piecewise global energy minimization. In *Pacific Visualization Symposium, 2011 IEEE* (2011), IEEE, pp. 59–66. 3

[HYML05] HAMARNEH G., YANG J., MCINTOSH C., LANGILLE M.: 3d live-wire-based semi-automatic segmentation of medical images. In *Medical imaging* (2005), ISOP, pp. 1597–1603. 3

[Kan05] KANG H. W.: G-wire: A livewire segmentation algorithm based on a generalized graph formulation. *Pattern Recognition Letters 26*, 13 (2005), 2042–2051. 3, 7

[KWT88] KASS M., WITKIN A., TERZOPOULOS D.: Snakes: Active contour models. *IJCV 1*, 4 (1988), 321–331. 1, 3

[LMT06] LIANG J., MCINERNEY T., TERZOPOULOS D.: United snakes. *Medical image analysis 10*, 2 (2006), 215–233. 3

[LSGX05] LOMBAERT H., SUN Y., GRADY L., XU C.: A multilevel banded graph cuts method for fast image segmentation. In *ICCV 2005* (2005), vol. 1, IEEE, pp. 259–265. 7

[LSS09] LIU J., SUN J., SHUM H.-Y.: Paint selection. 69. 2

[LSTS04] LI Y., SUN J., TANG C.-K., SHUM H.-Y.: Lazy snapping. *TOG 23*, 3 (2004), 303–308. 3

[MB95] MORTENSEN E. N., BARRETT W. A.: Intelligent scissors for image composition. In *Proceedings of SIGGRAPH '95* (1995), ACM, pp. 191–198. 1, 2, 3, 5, 6, 7

[MB98] MORTENSEN E. N., BARRETT W. A.: Interactive segmentation with intelligent scissors. *Graphical models and image processing 60*, 5 (1998), 349–384. 1, 3, 5, 6, 7

[MJS*04] MEIJERING E., JACOB M., SARRIA J.-C., STEINER P., HIRLING H., UNSER M.: Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. *Cytometry Part A 58*, 2 (2004), 167–176. 3

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: Grabcut: Interactive foreground extraction using iterated graph cuts. *TOG 23*, 3 (2004), 309–314. 2, 3

[SdMF14] SPINA T. V., DE MIRANDA P. A., FALCÃO A. X.: Hybrid approaches for interactive image segmentation using the live markers paradigm. *IEEE ToIP 23*, 12 (2014), 5756–5769. 3

[SGSP15] SUMMA B., GOOCH A. A., SCORZELLI G., PASCUCCI V.: Paint and Click: Unified Interactions for Image Boundaries. *Computer Graphics Forum 34*, 2 (Apr. 2015), 385–393. 3

[STP12] SUMMA B., TIERNY J., PASCUCCI V.: Panorama weaving: fast and flexible seam processing. *TOG 31*, 4 (2012), 83:1–83:11. 3, 4, 5

[STP17] SUMMA B., TIERNY J., PASCUCCI V.: Visualizing the uncertainty of graph-based 2d segmentation with min-path stability. *Computer Graphics Forum 36*, 3 (2017), 133–143. 2, 5, 6

[WAC07] WANG J., AGRAWALA M., COHEN M. F.: Soft scissors: an interactive tool for realtime high quality matting. 9. 2, 5

[XAB07] XU N., AHUJA N., BANSAL R.: Object segmentation using graph cuts based active contours. *Computer Vision and Image Understanding 107*, 3 (2007), 210–224. 7

[YCZL10] YANG W., CAI J., ZHENG J., LUO J.: User-friendly interactive image segmentation through unified combinatorial user inputs. *IEEE Transactions on Image Processing 19*, 9 (2010), 2470–2479. 2