Multi-Layer Stencil Creation from Images

Arjun Jain[a,b], Chao Chen[c], Thorsten Thormählen[a,d], Dimitris Metaxas[c], Hans-Peter Seidel[a]

[a] *MPI Informatik, Germany*
[b] *New York University, United States*
[c] *Rutgers University, United States*
[d] *University of Marburg, Germany*

Abstract

A stencil is a thin sheet of material, such as paper, plastic, or metal, with certain patterns cut from it. Applying a pigment through the cut-out holes produces a design on an underlying surface. Using multiple overlapping stencil layers, artists can create intricate, yet reproducible imagery on a variety of surfaces. Traditionally, artists have to design not only the final appearance, but also each individual stencil layer. A stencil layer needs to be connected, geometrically simple, and physically stable. Taking all these constraints into account during the design process is difficult and unintuitive even for skilled artists.

In this paper, we propose a system which separates the artistic design stage from the complex and tedious task of stencil creation. For a given user design, our algorithm automatically generates a set of stencil layers satisfying all required properties. The task is formulated as a constrained energy optimization problem and solved efficiently. Experiments, including a user study, are carried out to examine the complete algorithm as well as each individual step.

Keywords: Computer-aided art, Stencil graffiti, Automatic stencil creation, Markov random field optimization

1. Introduction

Computer-aided art, which employs algorithms to assist artists in various creation tasks, has received much attention in recent years. There are, for example, computational systems for creating shadow art [1], illuminating physical models [2], creating 3D animation [3], creating digital micrography [4], revealing the sketching sequence of a line drawing [5], generating paper architectures [6], designing origami figures [7], generating paper foldings [8, 9], or creating stylizations and abstractions of photographs [10]. An important common principle in computer-aided art is to separate tedious and difficult implementation details from the artistic design stage, so that artists can focus on expressing their ideas and leave the remaining to computer algorithms.

We believe stencil creation is an important task where such a principle could be applied. A stencil is a thin sheet of material with certain areas cut out. By applying pigment through the cut-out holes, one can easily create detailed paint-work on any surface, e.g., on a wall or a canvas. For more sophisticated designs, one applies multiple layers of stencils in a certain ordering. Using different colors for different layers, one can accomplish a piecewise constant design, which could be an approximation of some image. Figure 1 demonstrates the process of creating art with multi-layer stencils.

Email address: ajain@mpi-inf.mpg.de (Arjun Jain)

Figure 1: Creating art using multi-layer stencils. From left to right: designing and cutting stencils; creating a wall painting with stencils; final appearance (all three images courtesy of Orticanoodles).

The history of stencil art is almost as long as the human civilization. Early versions include 22,000-years-old anthropomorphic cave paintings, and inner wall decorations of Egyptian pyramids. In recent years, stencil art has started to be part of the mainstream art scene. Contemporary stencil artists such as Banksy, Blek le Rat, Jerome Mesnager, Nemo, Hugo Kaagman or Rene Gagnon have earned worldwide recognition for their works, which can be found in famous art galleries and auctions. The stenciling technique is also very important in our daily lives. Thanks to the high reproducibility, stencil art has become the *de rigueur* medium for promotional campaigns from night clubs, record labels, websites, and even multinational companies. The stenciling technique is used on clothing and accessories by well-known fashion houses such as *Louis Vuitton* and *Luella Bartley*. Stencil based designs can also be found on a vast variety of objects such as buildings, airplanes, cars, t-shirts, glass, ceramics, coffee, cakes, and even hair.

Unfortunately, the design of stencils is a complex and tedious process. After designing the final appearance, the artist has to create a set of stencil layers that could achieve it. A stencil should be a single connected component in order to be

Figure 2: Examples of results produced by users of our system.

reusable. Disconnected components, called *islands*, can either be removed or be connected via thin connections, called *bridges*. These operations would however affect the final appearance. Furthermore, the artist has to decide the ordering in which individual layers are applied. This decision is closely coupled with the design of each layer; layers that are applied early can satisfy the connectivity constraint more easily by exploiting regions that will be painted over later. Beside being faithful to the design, a stencil should have a simple geometry, *i.e*. a short boundary. No matter, whether the stencils are cut manually or by laser-cutting machines, the cutting cost is directly proportional to the boundary length. In case of thin materials, stencils should also be physically stable enough to be used in practice.

Overall, the multi-layer stencil generation is a problem with a very high dimensional solution space. There are exponentially many possible stencils one could choose from for each layer, and the number of possible stencil orderings is factorial in the number of layers. Therefore, we decided to isolate it from the artistic design stage, and solve it computationally.

In this paper we present a practical system that automates multi-layer stencil generation, while still allowing the users to achieve their individual artistic goals. Figure 2 shows several designs created with our system. Our main contribution is an algorithm which automatically generates an ordered set of stencil layers satisfying aforementioned requirements (geometric simplicity, physical stability, and connectedness), such that the final appearance resembles a given image as much as possible. Basing on this algorithm, we build a practical system which allows users to tune parameters (such as the color set, smoothness weight, etc.) and to edit stencils using brush strokes. Once the design is finished, our system will generate stencils automatically.

Related Work. Conventionally, to create stencils, one first creates a piecewise constant approximation of the input image. Afterwards, stencil layers are created through a *partition-and-fix* principle: take the complement of each color as a stencil layer, and manually fix its topology. Image editing software, such as Adobe Photoshop, can be used for such a task.

Bronson *et al*. [11] present a system which automatically generates a single-layer stencil for a given image. The system generates a black-and-white binary approximation of the given image, and then use white regions as stencil islands. Bridges connecting islands are built to ensure the stencil is connected. Among bridges between all pairs of islands, the set forming a minimum spanning tree is chosen. Meng *et al*. [12] focus on human portrait images. The stencil is generated using templates for human facial features, such as eye, mouth and nose. Igarashi and Igarashi [13] introduce an interactive system that allows non-professional users to design their own stencil plates from scratch.

Contributions. All existing stencil creation systems share the following shortcomings: (1) they only generate a single layer stencil; (2) they only fix topology by building bridges. In contrast, our system creates multi-layer stencils, allowing users to produce much more sophisticated designs. Furthermore, our algorithm considers both building bridges and removing islands, depending on what is better for the final result.

We solve the problem using a random field energy formation, in which we explicitly formulate desired properties of the output (appearance, simplicity, *etc*.) into summands of the final energy. This enables users to adjust their preference by tuning weights of these summands. The random field energy model has been broadly used in computer vision for image segmentation. It is known to be able to generate high quality segmentations for a broad range of input images. Furthermore, such a formulation allows us to use efficient algorithms like multi-

label graphcut.

Please note that energy-based formulation has been used in compute-aided art problems such as generating binary image abstraction [14], pixel-art-style image abstraction [15] and creating 3D models folded from planar sheets [8]. But these works do not usually consider the connectivity constraint.

2. Background

Random Field Image Segmentation. Since being introduced to computer vision, Markov random field (MRF) has become a popular tool for image segmentation [16, 17]. Given an image, one constructs an 8- or 4-connected grid graph whose vertices are all pixels and edges are all pairs of neighboring pixels. The segmentation problem is formulated as finding a discrete *labeling* of vertices $z : \mathcal{V} \to \mathcal{L}$ which minimizes a given random field energy

$$\mathrm{E}_{\mathrm{RF}}(z) = \sum_{i \in \mathcal{V}} \sum_{\ell \in \mathcal{L}} \phi_{i,\ell} [\![z_i = \ell]\!] + \lambda \sum_{(i,j) \in \mathcal{E}} \psi_{i,j} [\![z_i \neq z_j]\!], \qquad (1)$$

in which the *Iverson bracket* $[\![\cdot]\!]$ has value one if the predicate inside it is true, and zero otherwise. A *unary potential* $\phi_{i,\ell}$ is the cost of assigning a pixel i a label ℓ. Its value is based on the probability of the color of pixel i being a sample from the color distribution of label ℓ. A *pairwise potential* $\psi_{i,j}$ is the cost of assigning two neighboring pixels i and j different labels. Pairwise potentials play the role of a regularizer, so that the segmentations have shorter boundaries that are attracted to sharp color changes (edges) in the image.

For binary segmentations, the set of labels $\mathcal{L} = \{\mathrm{fg}, \mathrm{bg}\}$ consists of *foreground* and *background*. In this case, under certain assumptions, the problem can be transformed into a max-flow/min-cut problem, and solved in polynomial time [18, 19]. For the multi-label ($L = |\mathcal{L}| > 2$) case, Boykov *et al*. [20] prove that the problem is NP-hard and use the alpha-expansion technique to solve the problem efficiently when the energy satisfies certain conditions. In this paper, we will employ this algorithm in the `Multi-Label-Segmentation` subroutine. Please refer to [21, 17] for more details.

Binary Segmentation with Topology Constraints. Connectivity, as a natural global property, has been of interest in image segmentation. One may want to find a binary labeling that minimizes the energy in Eq. (1) under the constraint that the foreground ($z^{-1}(\mathrm{fg}) = \{i \in \mathcal{V} \mid z_i = \mathrm{fg}\}$) is connected. This problem, however, has been shown to be NP-hard [22]. Several approximation methods have been proposed [23, 22]. Nowozin and Lampert [24] formulate the problem as a linear programming problem with exponentially many linear inequality constraints. The algorithm solves the problem exactly, but does not scale well with the size of the image.

Using ideas from computational topology [25, 26], Chen *et al*. [27] propose a novel algorithm which is efficient even for natural images. Intuitively, the algorithm computes a binary segmentation using the graphcut algorithm [19], and then fixes the topology. Each island of the foreground is either removed completely, or merged to other islands via an optimal bridge. The choice of removing or merging depends on which operation is less expensive. The final segmentation is guaranteed to be connected after all islands are fixed. The proposed algorithm is called `TopoCut` and will be later employed as a subroutine in this paper.

For each island, *i.e*. a component C, the expense of removing it completely is the maximum of $\phi_{i,\mathrm{bg}} - \phi_{i,\mathrm{fg}}$ for all $i \in C$. The alternative option is merging C to another component by building a bridge connecting them. The expense of this bridge is the maximum of $\phi_{i,\mathrm{fg}} - \phi_{i,\mathrm{bg}}$ for all pixel i in the bridge. Out of exponentially many candidate bridges, the algorithm efficiently finds the one with the minimal expense. In the end, the algorithm compares the expenses of removing and merging, and chooses the less expensive one. The operation can be achieved by changing unaries accordingly and rerun the graphcut segmentation algorithm; to remove an island (resp. build a bridge), let all pixels within the island (resp. bridge) have $+\infty$ (resp. $-\infty$) foreground unaries.

3. Multi-layer Stencil Generation

Given a set of colors $\mathcal{L} = \{1, \ldots, L\}$, our algorithm computes stencils for each color, as well as an ordering in which these stencils are applied to the background (a wall, a canvas, or some other surface). The problem is formulated as an energy minimization problem. We emphasize three properties of the generated stencils.

- The resulting painting is faithful to the original input image;
- Stencils are geometrically simple and physically stable;
- Each stencil is topologically connected.

We denote the multi-layer stencils by $y : \mathcal{V} \times \mathcal{L} \to \{0, 1\}$. Pixel i belongs to the stencil (or respectively the paint area) for color ℓ if and only if $y_{i,\ell} = 0$ (or respectively $y_{i,\ell} = 1$). We use π, a permutation of the sequence $(1, \ldots, L)$, to specify an ordering of the colors in which the corresponding stencils are applied. We will refer to π_1 as the *bottom* stencil that is applied first and π_L as the *top* stencil that is applied last. We formulate the energy of a multi-layer stencil configuration to be

$$E(y, \pi) = \mathrm{E}_{\mathrm{appearance}}(y, \pi) + \mathrm{E}_{\mathrm{stencil}}(y, \pi) \quad . \qquad (2)$$

The first summand $\mathrm{E}_{\mathrm{appearance}}$ enforces the faithfulness of the resulting stencil painting to the original input image. A natural choice would be the multi-label random field energy, $\mathrm{E}_{\mathrm{appearance}}(y, \pi) = \mathrm{E}_{\mathrm{RF}}(z^\pi)$ (Eq. (1)), where z_i^π is the last color in the ordering with which pixel i is painted, in other words, the topmost drawn color of pixel i. The second summand, $\mathrm{E}_{\mathrm{stencil}}$, measures the simplicity of all the stencils, i.e., the total length of the boundaries of the stencils. Furthermore, we have the hard constraint that all generated stencil layers have to be connected. Taking all these requirements into account, we solve the following optimization problem.

Main Problem. *Compute* (y, π) *minimizing*

$$E(y,\pi) = \sum_{i\in\mathcal{V}} \sum_{\ell\in\mathcal{L}} \phi_{i,\ell}[\![z_i^\pi = \ell]\!] + \beta \sum_{(i,j)\in\mathcal{E}} \psi_{i,j}[\![z_i^\pi \neq z_j^\pi]\!] + \alpha \sum_{(i,j)\in\mathcal{E}} \sum_{\ell\in\mathcal{L}} [\![y_{i,\ell} \neq y_{j,\ell}]\!] \quad (3)$$

such that $\forall\ell$, $y_{*,\ell}^{-1}(0) = \{i \in \mathcal{V} \mid y_{i,\ell} = 0\}$ *is connected.*

The parameter α tunes the bias towards the simplicity of the stencils. Increasing α would uniformly simplify the resulting image as the boundaries of all stencils are shortened. Increasing the parameter β would also simplify stencils. But edges of the final appearance tend to stay with high contrast edges of the original image, and thus details of the image are better preserved.

The multi-layer stencil energy E is obviously an extension of the multi-label random field energy. But the two energies have an even closer relationship. If we drop the connectedness constraint, and assume that each pixel is painted with one and only one color, then the energy $E(y, \pi)$ is equivalent to the multi-label random field energy (Eq. (1)) of the top color z^π with unary potential ϕ and pairwise potential $\widetilde{\psi} = \beta\psi + 2\alpha$ (with $\lambda = 1$ in Eq. (1)), formally

$$E(y,\pi) = \widetilde{\mathrm{E}}_{\mathrm{RF}}(z^\pi) = \sum_{i\in V} \sum_{\ell\in\mathcal{L}} \phi_{i,\ell}[\![z_i^\pi = \ell]\!] + \sum_{(i,j)\in\mathcal{E}} (\beta\psi_{i,j} + 2\alpha)[\![z_i^\pi \neq z_j^\pi]\!] \quad (4)$$

If we take a multi-label segmentation of the image, and construct the stencil of each label by strictly taking the complement of this label's region in the segmentation, $\widetilde{\mathrm{E}}_{\mathrm{RF}}$ of this segmentation and E of this stencil set are equivalent. Note that in such a condition, the ordering does not affect the energy any more. Formally, we have

Lemma 1. *The energy* $E(y, \pi) = \widetilde{\mathrm{E}}_{\mathrm{RF}}(z^\pi)$ *if for any i, there is one and only one color* ℓ *such that* $y_{i,\ell} = 1$.

The proof exploits that neighboring color regions share exactly one border pixel and thus the last summand of Eq. (3) collapses to $\alpha \sum_{(i,j)} 2[\![z_i^\pi \neq z_j^\pi]\!]$. The full proof can be found in the appendix. This lemma gives a theoretical justification to initialize the optimization of our problem using the result of a multi-label segmentation with the energy according to Eq. (4) at certain stages of our algorithm, as will be explained in details later.

Next, we present components of our system. A block digram is given in Fig. 3 where dashed blocks indicate optional steps.

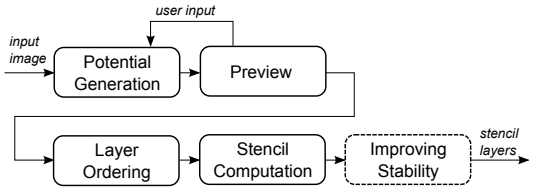

Figure 3: Block diagram of our system

3.1. Potential Generation and Preview

In the preparation stage, we generate potentials of the multi-layer stencil energy, (ϕ, ψ), based on user input. A user chooses an input image and specifies parameters including the number of colors L and the weights α and β. To decide colors, we apply k-means clustering of all input pixels' colors. The means of the L clusters are chosen as the L colors, $(\mathbf{c}_1, \ldots, \mathbf{c}_L)$. The system then computes the potentials accordingly (details will be given later).

We provide a user interface where a user previews the final appearance and changes the design. A preview is a mutli-label segmentation minimizing the random field energy $\widetilde{\mathrm{E}}_{\mathrm{RF}}$ in Eq. (4), which can be computed very efficiently using a standard multi-label segmentation algorithm. It is a good approximation of the final appearance because of the equivalence between $\widetilde{\mathrm{E}}_{\mathrm{RF}}$ and E (cp. Lemma 1). If a user does not like the generated preview, all mentioned parameters can be modified. Furthermore, we provide an interactive editing interface such as selecting individual colors, and using brush strokes to modify stencils (similar to GrabCut [16]). Once the user is satisfied, the potentials are passed to the automatic stencil computation pipeline. Our user interface is implemented as a dynamic webpage using AJAX technologies. It is available at http://www.stencilcreator.org.

We conclude this subsection with details on how to generate the potentials. For layer ℓ and pixel i, the unary potential $\phi_{i,\ell}$ is computed as the negative-log-likelihood of the probability of i's color $\mathbf{c}_i$ evaluated in a single Gaussian distribution $\mathcal{N}(\mathbf{c}_\ell, \sigma^2)$, formally, $\phi_{i,\ell} = -\log(\mathcal{P}_{\mathbf{c}_\ell,\sigma^2}(\mathbf{c}_i)) = \frac{1}{2\sigma^2}\|\mathbf{c}_\ell - \mathbf{c}_i\|^2$. For convenience, we assume the same σ for all ℓ, and thus drop it in the energy model[1]. As for pairwise appearance potentials, we use the contrast sensitive measure $\psi_{i,j} = \exp(-\|\mathbf{c}_i - \mathbf{c}_j\|^2/\mathrm{const})$ introduced by [28]. Such pairwise potentials enforce not only a shorter boundary of color regions, but also that the region's boundary is attracted to high contrast edges of the image. Every stencil layer includes a frame encapsulating the original image domain. We extend the domain with an outer border, in which the potentials are set to $\phi_{i,\ell} = \infty$. Thus, every pixel in the extended area is assigned the stencil label, $y_{i,\ell} = 0$, forming the frame. The topology constraint ensures the rest of the stencil is always connected to the frame.

3.2. Automatic Stencil Computation

Our main algorithm computes stencils and an ordering minimizing the energy in Eq. (3). A naive algorithm would be to enumerate all possible orderings and to compute the optimal stencils for each ordering. But this is too expensive. Therefore we propose a heuristic algorithm which approximates the optimal ordering, and then computes the optimal stencil with regard to the computed ordering.

[1] We do not use a Gaussian mixture model with multiple components [16], because we have the constraint that each label ℓ should be approximated by a single color in the final appearance. A further extension to an EM type algorithm of estimating the color center and the variance is possible and left for future work.

For ease of exposition, we first explain for a given ordering π how to compute stencils minimizing the energy $E(y, \pi)$ under the connectedness constraint. Afterwards, we explain how to approximately compute the optimal ordering. Lastly, we provide details of how to improve the physical stability of stencils.

Computing Stencils for a Given Ordering. To compute stencils, a straightforward solution would be generating a multi-cut segmentation of the image and apply the conventional partition-and-fix principle. Our novel idea, however, is built on the following observation. Inside the painted area of top stencil layers, bottom layers can be of arbitrary shape without affecting the final appearance. In other words, bottom layers can explore such appearance-penalty-free area in order to achieve geometric simplicity and topological correctness. We propose an algorithm that exploits this strategy and achieve a much smaller cut length, as we will show in Section 4.

Our algorithm computes stencil layers from top (π_L) to bottom (π_1). Notice that unary potentials for a pixel i would only contribute to the energy once as we proceed. As soon as we find the first layer at which pixel i is painted, its unaries, $\phi_{i,*}$, become unrelated, as the corresponding predicate $z_i^\pi = \ell$ are false for all the remaining undecided labels ℓ. Therefore it is safe to suppress all these terms, *i.e.* set to zero, and proceed. A similar argument can be applied to the pairwise appearance terms, which only depend on z_i^π, but not to the pairwise stencil term.

As we proceed from top layer to bottom, after computing each layer π_s, we suppress the unary potentials ϕ and pairwise ψ at areas painted by label π_s (illustrated in Fig. 4). Then π_s is removed from the set of colors $\mathcal{L}$ for the subsequent computations. The updated potentials are used to compute stencils for the remaining layers $\{\pi_1, \ldots, \pi_{s-1}\}$. For the special case of the bottom layer, we let all pixels be painted, so that every pixel has at least one painted color. The corresponding pseudo-code is given in Procedure 1.

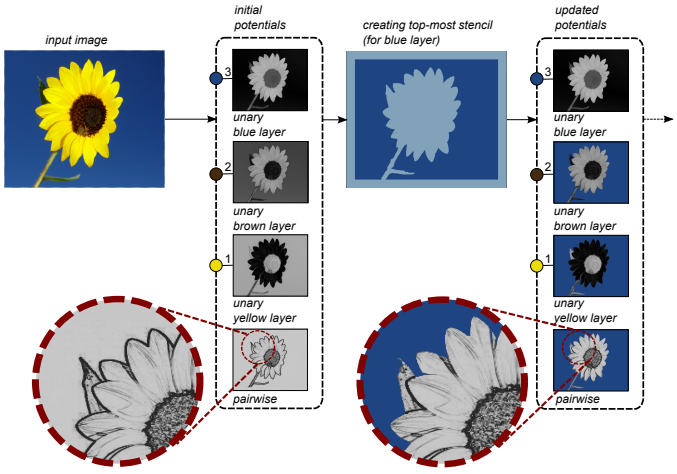

Figure 4: Illustration of the potential suppression strategy: *(left to right):* the input image; initial potentials; the top-most stencil is computed (stencil is light blue, paint area is blue); at the painted area, unaries of the remaining two colors (brown and yellow) and the pairwise potentials are suppressed.

Procedure 1 `Compute-All-Stencils` : computing all stencils for a given ordering

Input: $\mathcal{L}, \pi, \phi, \psi$
Output: y
  for $s = L$ to 2 **do**
    y_{*,π_s} = `Compute-One-Stencil`($\mathcal{L}, \pi, s, \phi, \psi$)
    for all i with $y_{i,\pi_s} = 1$ **do**
      $\phi_{i,*} = 0$
    for all $(i, j) \in \mathcal{E}$ with $y_{i,\pi_s} = 1$ or $y_{j,\pi_s} = 1$ **do**
      $\psi_{i,j} = 0$
    $\mathcal{L} = \mathcal{L} \setminus \{\pi_s\}$
  $y_{*,\pi_1} = 1$

This suppression strategy is well justified. We can prove that given the layers that have been computed $(\pi_s, \ldots, \pi_L)$, the optimizer of energy E is the same as the optimizer of the suppressed energy. In other words, if we have computed the layers $\pi_s, \ldots, \pi_L$ of the optimal solution of E (Eq. (3)), the remaining layers could be computed by optimizing the suppressed energy.

Lemma 2. *Given the computed stencils for the top layers, $y'_{\pi_s}, \ldots, y'_{\pi_L}$, let $\widetilde{E}$ be the energy using the suppressed potentials, and parameterized by the set of undecided stencils $y_{*,\pi_1}, \ldots y_{*,\pi_{s-1}}$. Assuming the ordering π is given, we have*

$$\underset{y_{\pi_1}, y_{\pi_2}, \ldots, y_{\pi_{s-1}}}{\operatorname{argmin}} \widetilde{E}((y_{\pi_1}, \ldots, y_{\pi_{s-1}}), \pi) = \underset{y: y_{\pi_s} = y'_{\pi_s}, \ldots, y_{\pi_L} = y'_{\pi_L}}{\operatorname{argmin}} E(y, \pi) \quad (5)$$

In fact, we can prove an even stronger result: the two energies are equivalent for any solution (the proof is given in the appendix).

Computing One Stencil. Next we explain how to compute a particular stencil y_{*,π_s}. Unfortunately, solving such problem exactly is computationally infeasible: for the special case of $\alpha = 0$, the problem becomes an image segmentation task with connectedness constraint, which is proved to be NP-hard [22]. Therefore, we do not pursue an exact solution. Instead, we use a generalization of the TopoCut approach in a multi-label setting.

We first apply a multi-label segmentation using the suppressed potentials. The label set consists of the remaining labels $\{\pi_1, \ldots \pi_s\}$. The area given label π_s is the paint area and the complement, namely the union of the regions given any other label, is the stencil. Lemma 1 gives us confidence that multi-label segmentation would give us a reasonable stencil for the current top layer π_s.

In the case when the stencil is not connected, we fix the topology by adapting the TopoCut algorithm. Recall TopoCut fixes the connectivity of the foreground of a binary segmentation by removing islands and building bridges. TopoCut changes the foreground unaries accordingly and then segments the image again. We reduce our problem into a binary segmentation problem in which π_s is the background and the union $\{\pi_1, \ldots, \pi_{s-1}\}$ is the foreground. For each pixel i, we use the unary of label π_s, ϕ_{i,π_s}, as the background unary, and use the

Procedure 2 `Compute-One-Stencil` : computing stencil y_{*,π_s} using ϕ, ψ on label set $\mathcal{L}$

Input: $\mathcal{L}$, π, s, ϕ, ψ
Output: y_{*,π_s}

```
z = Multi-Label-Segmentation(φ, βψ + 2α, L)
y_{*,π_s} = [[z_i = π_s]]
while y^{-1}_{*,π_s}(0) not connected do
    φ̄_{*,fg} = φ_{*,π_s}
    φ̄_{*,bg} = min_{ℓ∈L\{π_s}} φ_{*,ℓ}
    φ_{*,π_s} = TopoCut(φ̄_{*,fg}, φ̄_{*,bg}, βψ + 2α)
    z = Multi-Label-Segmentation(φ, βψ + 2α, L)
    y_{*,π_s} = [[z_* = π_s]]
```

minimum of all other unaries, $\min_{\ell \in \mathcal{L}\setminus\{\pi_s\}} \phi_{i,\ell}$, as the foreground unary. These background and foreground potentials give a confidence of whether a pixel i should be assigned to π_s or not. TopoCut guarantees the connectivity of the foreground/stencil. The corresponding pseudo-code is given in Procedure 2).

Layer Ordering. Finding a good layer ordering is very important. In the synthetic example in Fig. 5, each color segmentation for the white, yellow, and green layer would result in a disconnected stencil with a large round island in the middle. To fulfill the connectedness constraint for each layer, the island either needs to be connected to the outer border via a bridge or needs to be removed. As shown in the bottom row of Fig. 5, creating bridges or removing islands does not necessarily result in visual artifacts if the layer ordering is chosen such that layers drawn later cover the affected region. In this example each islands in the white, green, and yellow layer could be removed without affecting the result. The input image is perfectly reproduced. In contrast, for a non-optimal layer ordering (see Fig. 5, top row) the layers drawn later can not cover the removal of islands in the white, green, and yellow layer and bridges must be created as a result. The yellow color (layer 2) is more similar to the desired green (layer 3) and white (layer 4) than red (layer 1). Therefore, the optimization tries to create yellow bridges instead of red ones. As a result, the painted area of layer 2 comprises the areas of the bridges of layer 3 and 4 while trying to maintain a minimal stencil boundary. In Sec. 4, we show more evidence on real images that the layer ordering is essential for a good reproduction of the input.

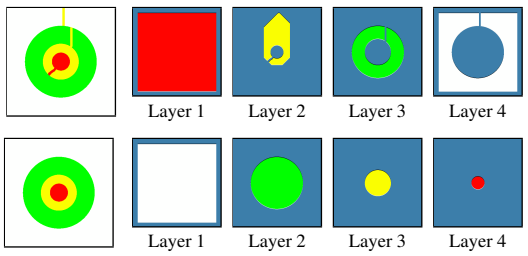

Figure 5: Without a good ordering *(top row)* many bridges need to be created; with a good ordering *(bottom row)* the input image is perfectly reproduced.

To decide the optimal layer ordering π, a naive method is to enumerate all possible orderings ($L!$ many). For each ordering, compute stencils using `Compute-All-Stencils` and evaluate the energy (Eq. (3)). However, for a large number of layers this is impractical as `Compute-All-Stencils`, the most expensive operation, is executed $L!$ times.

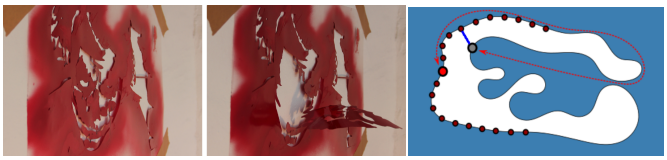

Figure 6: Left and middle: stencils with and without stability bridges when they are attached to the wall (corresponding to the "joker" painting shown in Fig. 2). Right: illustration of how the stability bridge is built. A stability bridge (blue) is built to connect p (grey) and q, which is chosen from some boundary candidates (red). The boundary point with the maximal geodesic distance from p is highlighted and its geodesic path from p is drawn.

We propose an efficient heuristic algorithm to decide ordering before actually calling `Compute-All-Stencils`. The idea is to pre-compute all stencils using the partition-and-fix strategy. The algorithm then enumerates all possible orderings. For each ordering, evaluate the energy E using these precomputed stencils. Procedure 3 shows the corresponding pseudo-code. The intuition behind this approach is that TopoCut would fix the topology while sacrificing the appearance. We find the ordering with the least amount of appearance damage due to TopoCut. Note that computing each stencil using the partition-and-fix strategy is equivalent to calling `Compute-One-Stencil` with the corresponding stencil as the top-most layer.

Our algorithm calls `Compute-One-Stencil` L times for precomputation, and $L-1$ times inside `Compute-All-Stencils`. In contrast, the naive method calls `Compute-One-Stencil` for $(L-1) \times L!$ times. We run experiments on 31 images, using 3 colors. Our method is almost as good as the naive method; the former is only 0.27% worse than the latter in terms of achieved energy.

Improving Stability. Ensuring connectedness does not necessarily guarantee that the stencil is fully usable. A good stencil must be stable and no part of the stencil should droop when the stencil is lifted from the ground or attached to a wall. This is especially important for thin stencil materials such as paper, plastic, *etc*. We provide the option to automatically find unstable parts of the stencil using a physical simulation and then improve the stability of these identified weak areas by building additional support bridges. Fig. 6 demonstrates stability improvement in a real world example.

Procedure 3 `Create-Stencils` : compute ordering and stencils

Input: $\mathcal{L}$, ϕ, ψ
Output: y, π

```
for ℓ = 1 to L do
    δ = (1, ..., ℓ − 1, ℓ + 1, ..., L, ℓ)
    z_{*,ℓ} = Compute-One-Stencil( L, δ, L, φ, ψ )
π = argmin_τ E(z, τ)
y = Compute-All-Stencils( L, π, φ, ψ )
```

Figure 7: Results (from left to right): predicted result on the wall; the generated stencil layers (light blue is the stencil material, all other colors should be cut away); and the input image; (input images courtesy of flickr users Fred baby, Alexandre Moreau, and Nickwheeleroz)

We first locate unstable stencil points for each generated stencil layer. A physical simulation is executed using a mass-spring system placed in a downward gravity field. To this end, we instantiate a unit mass-particle at each stencil points, *i.e.* pixels i's with $y_{i,\ell} = 0$. We then create 1-neighbor and 2-neighbor spring connections to the mass-particle if both involved pixels are a part of the stencil. Locations of the stencil's outer border are fixated. Once the mass-spring simulation has reached an equilibrium, we perform non-maximum suppression [29] with a window size of $0.1 \times \max\{h, w\}$ (with image height h and width w), in order to find particles with large displacements. All stencil points with displacements greater than the threshold $0.25 \times \max\{h, w\}$ are considered unstable and will be stabilized.

For each identified unstable point p, our algorithm finds another stencil point q and builds a stabilizing bridge connecting them. But not all boundary points are equally well suited. It is important to ensure that q is not within the unstable area so that the bridge actually improves the stability. We only choose q from boundary points which have large enough geodesic distances from p within the stencil. Particularly, we measure the maximal geodesic distance of all boundary points from p, and only consider points with geodesic distance at least a half of the maximum. Among these candidates, we choose the closest one to p as q, and the line segment (p, q) as the bridge. See Fig. 6 (right) for an illustration. In addition, we weight pixels by their unaries and use the shortest path between p and q in the weighted graph as the bridge. This makes the bridge more likely to go through regions where the image is less likely to be the paint color (more likely to be the stencil).

Figure 8: Real world stencil designs: mural of Jimi Hendrix (4.5ft wide), graffiti of Audrey Hepburn on canvas, Garfield decoration on a cake.

4. Results

In general, our system generates high quality approximations of input images (see Fig. 7 for examples). To validate the practicality of our system, we also create real world examples (photographs are shown in Figs. 2 and 8). All employed stencils were printed on polyester film and then manually cut. The producing process is demonstrated in the supplemental video.

In order to justify different components of our contribution, we carry out qualitative and quantitative evaluations.

Figure 9: Images used for qualitative comparisons (courtesy of flickr users Sergiu Bacioiu, Vramak and ArloMagicMan).

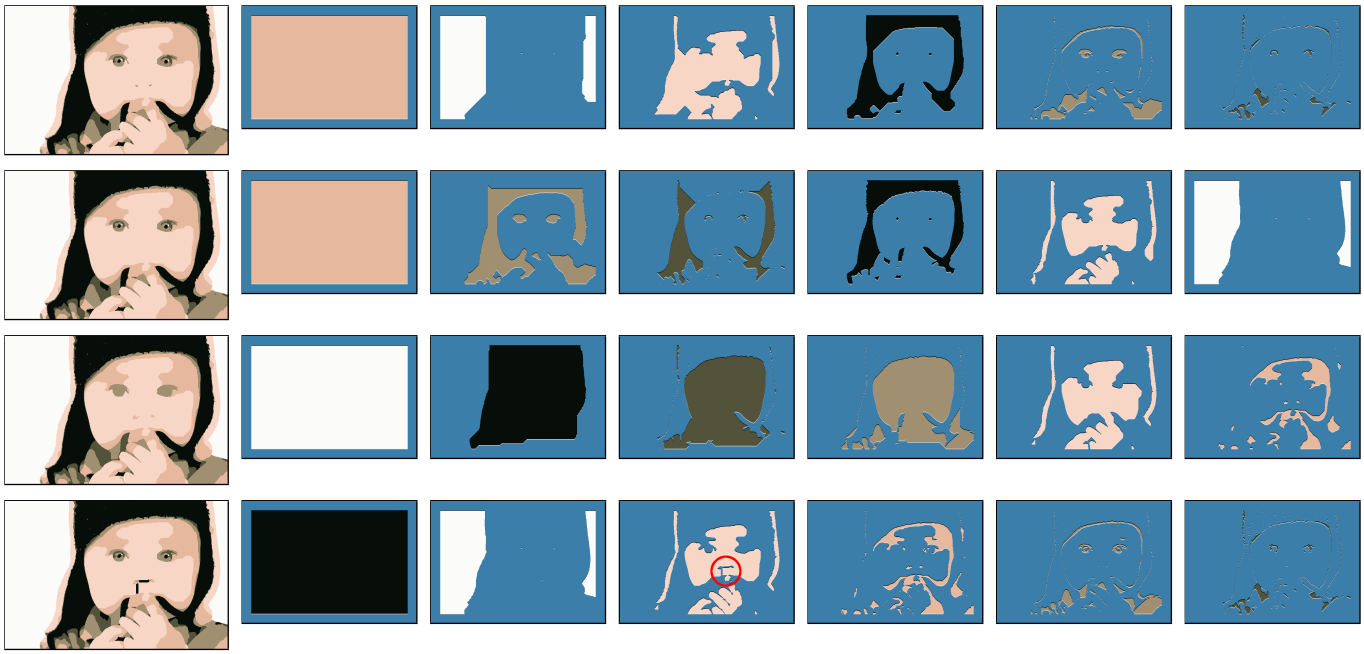

Figure 10: Qualitative comparisons. *First row:* our default result; *Second and third row:* results using random orderings; *Fourth row:* bridge-only baseline result. The bridges which are responsible for the unnatural black strokes in the bridge-only baseline results are highlighted by a red circle. The input image is in Fig. 9 (left).

4.1. Qualitative Comparison

We visually compare our *default method* with different downgraded versions. Note that the default method does not include stability enforcement. Images used in this section can be found in Fig. 9.

Necessity of layer ordering. We first compare our default method with the same approach without the optimal ordering. Fig. 10 (first row) shows the result of our default method, which uses the optimal ordering. Second and third rows are results with two random orderings, in which the topology constraint leads to unnecessary TopoCut operations, and thus loss of important semantic features, such as the nose and the pupils of the baby.

Necessity of TopoCut. Fig. 10 (fourth row) shows the result when we replace the stencil generation step with a multi-layer generalization of the method by Bronson et al. [11]. This baseline method uses the multi-label segmentation result to directly generate stencils and then fixes their topology. For each color, the stencil is computed by taking the complement region. Contrary to TopoCut, the topology is fixed by only connecting islands with bridges. Even though these bridges are optimized with regard to unaries so that they go through regions which are less likely to be the paint, unnecessary bridges would still introduce unnatural strokes in the appearance.

Necessity of potential suppression. One of our contributions is the potential suppression strategy in multi-layer stencil computation (Procedure `Compute-All-Stencils`). To illustrate the importance of this strategy, we compare our default method with a baseline method using the partition-and-fix strategy. Recall that we use the partition-and-fix strategy to generate pre-computed stencils for layer ordering computation. This baseline method is implemented by replacing the last line in `Create-Stencils` by simply stacking precomputed stencils $z_{*,\ell}$ according to the computed ordering π. In experiments, our default method and the baseline method have similar final appearance. However, the proposed potential suppression approach achieves a much smaller cut length for stencils. As an example, please compare the third stencil layer in Fig. 11.

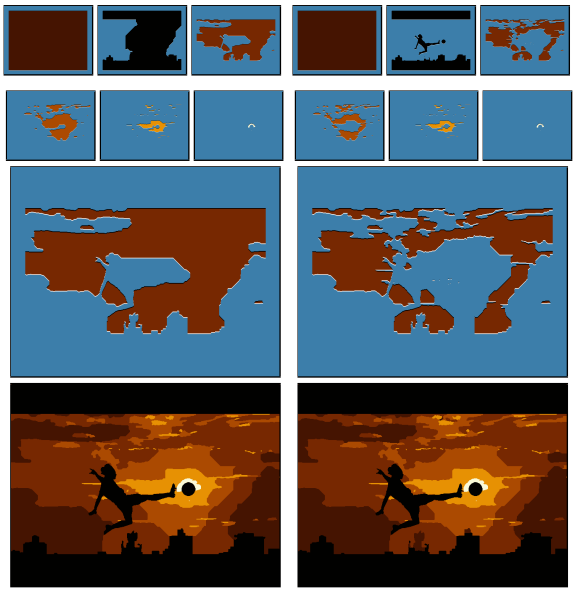

Figure 11: Comparison of results with (left) and without (right) the suppression strategy. *Top:* all stencil layers; *Middle*: magnification of the third layer; *Bottom*: final appearance. The input image is in Fig. 9 (middle).

4.2. Quantitative Comparison

We compare our default method (*M0*) with the three downgraded versions mentioned in the previous section: *M1:* our approach without layer ordering optimization (using random ordering instead), *M2:* bridge-only baseline, *M3:* our approach without suppression strategy (using partition-and-fix instead). In addition, we compare with *M4:* stencil creation using the conventional method. Given the preview result as a starting point, a human subject employs a professional image editing software (Adobe Photoshop) to manually create stencils and decide the ordering. Please refer to the supplemental material for the detailed instructions that were given to the subjects.

We compare the methods on 75 data items. Each data item includes one input image and one parameter setting (L, α and β). Results are compared in terms of similarity to the input segmentation (the appearance energy $E_{\text{appearance}}$), simplicity of the stencils (the cut length of stencils) and our multi-layer stencil energy (Eq. (2)). The average scores are given in the table below. Since for different data, these measures are not comparable, we normalize them before taking average over all data items. For each data item and each measure, we normalize by dividing by the corresponding measure of the preview (multi-label segmentation result). The conventional method (M4) is time-consuming. Therefore, it was not executed for all 75 input images. In total, we asked 13 human subjects to perform this task, each on a different input image.

	M0	M1	M2	M3	M4
$E_{\text{appearance}}$	1.012	1.025	1.043	1.013	1.045
Cut Length	0.797	0.795	0.885	0.883	0.929
Total Energy	0.984	0.997	1.023	0.996	1.033
Avg. Time (Sec.)	61	22	32	40	588

As shown in the table above, in terms of the total multi-layer stencil energy, our method is better than all others. The difference is statistically significant; applying a t-tests to compare our method and each downgraded method (M1 to M4), we observe that the p-value is always below 0.003. We also observe that using random ordering (M1) generates a slightly shorter cut length than our default method. However, this is often due to dropping some big islands, leading to worse appearance, and thus higher total energy. Using partition-and-fix strategy (M3) has similar appearance as our default method. But the cut length is generally much longer, penalizing the total energy more.

Due to the normalization, the quantitative difference between different methods could appear small. However, the actual difference is huge. For example, below we show the mean cut length in number of pixels per layer. (The cut length of each data is divided by the number of layers.) It can be oberved that M0 and M1 are approximately 300 pixels shorter than M2 and M3. They are 950 pixels shorter than M4. Since the size of our images are approximately 500×600 pixels, these differences are very noticeable.

	M0	M1	M2	M3	M4
Cut Length (Pixel)	2188	2194	2504	2467	3157

We also compare the computation times of all methods. For the conventional method (M4), we measure the time a human subject needs for editing stencils using Photoshop, under the condition that the subject has read the complete instructions and is instructed to finish the task as fast as possible without sacrificing the quality. Although slower than our other automatic baselines, our default method is approx. 10 times faster than the conventional method (M4).

User study. Visual quality cannot be simply measured by comparing energy. In practice, we notice cases in which the visual quality is dramatically changed when we change some small yet semantically important regions, *e.g.* pupils, nose, *etc*. We carried out a user study to visually compare our default method with the downgraded versions. A group of 17 participants is asked to rank the results of different methods basing on visual quality. For each data item, the result with the best visual quality receives a rank score of 1 and the worst a score of 5. A lower rank is considered better. We use all 13 data items for which we have results for the conventional method (M4). Detailed user instructions and all data items can be found in the supplemental material. Below we show the average user rankings for all methods.

	M0	M1	M2	M3	M4
Avg. Ranking	2.08	4.21	3.31	2.04	3.37

Consistent with our previous finding by comparing appearance energy, our default (M0) and the partition-and-fix method (M3) received the best average rankings. This is expected because only appearance is evaluated here and the smaller cut length of M0 is neglected. Though the average for M3 is slightly lower, statistically our default method is on a par with M3 (equivalence testing with two one-sided Wilcoxon-Mann-Whitney tests [30] shows no statistical difference). Other methods (M1, M2 and M4) have worse average user ranking. Using a Wilcoxon-Mann-Whitney test, we can reject the null hypothesis that there is no difference between our default method and them (all p-values $< 10^{-21}$). Therefore, the superiority of our default method in terms of user rating is statistically significant.

4.3. Improving Stability

In this section, we evaluate the proposed method to improve the stencil's stability. In terms of speed, adding stability enforcement makes our algorithm about 3 times slower, largely due to the expensive operation of physical simulation. On the other hand, the approach improves stability drastically. See Fig. 12 for a comparison of the results with and without stability bridges. The added stability bridges improves the stability, while negatively affecting the final appearance slightly.

In Fig. 13, we plot the stencil energy E and the stability as more and more stability bridges are built. In order to measure the stability, we carry out physical simulation as explained in Sec. 3.2. The evaluation score for stability is the maximal displacement of all simulation particles. A smaller score indicates a more stable stencil. The x-axis is the maximal number of stability bridges that are allowed to be built for each stencil layer (bridges would not be built if a stencil has reached the stability

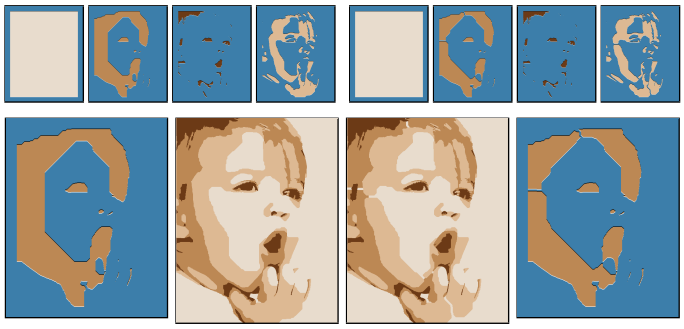

Figure 12: Comparison of the default method (left) and the result with stability enforcement (right). *Top:* all stencil layers; *Bottom*: magnification of the second layer, and the final appearance. The input image is in Fig. 9 (right).

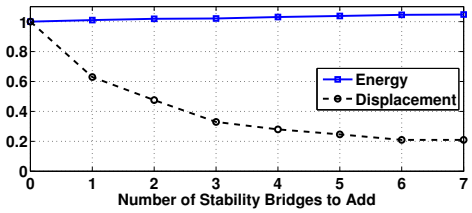

Figure 13: Plot of energy and stability over the number of stability bridges.

threshold). The y-axis is the ratio of displacement (resp. energy) of stabilized stencils over displacement (resp. energy) of the original stencils. The stability is improved by up to 70% if we build up to 3 stability bridges for each stencil layer. As a result, the stencil energy increases by 3%. The extra penalty includes a decreased final appearance as well as extra cut length due to additional bridges. In the example shown in Fig. 12 (right), three stability bridges per stencil layer were generated.

5. Conclusion and future work

In this paper, we propose a system which automatically generates stencil layers, so that users only need to focus on the designing stage in the stencil creation process. The problem is formulated as a constrained energy optimization problem and solved efficiently. We evaluated different components of our algorithm in qualitative and quantitative comparisons and showed that our default approach receives the best result.

One issue of our system is that the generated stencils may lose some important semantic features in the final appearance, such as the nose or eyes in a human portrait. At this stage, we leave it to the users to fix these areas using strokes. However, in future, we would like to train our system to keep such features intact. This could be achieved by training a classifier to decide whether to keep an island in the TopoCut procedure.

Computer-aided art creation has received large interest in the computer graphics community as well as among contemporary artist. Multi-layer stencil design is difficult for humans because of the huge solution space and the complicated factors that need to be considered. We believe our system is a good example where computational methods can assist artists without affecting their artistic intent.

In future, we plan to extend our method to other applications such as paper folding [8], paper craft [9], and popup design [31, 6]. In these applications topological connectivity, geometric simplicity, and physical stability are also important factors. Our energy based framework, which incorporates these factors seamlessly, could help human designers to transfer their ideas easily into reality.

References

[1] Mitra NJ, Pauly M. Shadow art. ACM Transactions on Graphics (TOG) 2009;28(5).

[2] Raskar R, Ziegler R, Willwacher T. Cartoon dioramas in motion. In: Non-Photorealistic Animation and Rendering. NPAR; 2002, p. 7–ff.

[3] Borosan P, Jin M, DeCarlo D, Gingold Y, Nealen A. RigMesh: Automatic rigging for part-based shape modeling and deformation. ACM Transactions on Graphics (TOG) 2012;31(6).

[4] Maharik R, Bessmeltsev M, Sheffer A, Shamir A, Carr N. Digital micrography. ACM Transactions on Graphics (TOG) 2011;30(4).

[5] Fu H, Zhou S, Liu L, Mitra N. Animated construction of line drawings. ACM Transactions on Graphics (TOG) 2011;30(6).

[6] Li XY, Shen CH, Huang SS, Ju T, Hu SM. Popup: Automatic paper architectures from 3d models. ACM Transactions on Graphics (TOG) 2010;29(4):111–9.

[7] Lang RJ. A computational algorithm for origami design. In: Computational Geometry. CG; 1996,.

[8] Kilian M, Flöry S, Chen Z, Mitra NJ, Sheffer A, Pottmann H. Curved folding. In: ACM Transactions on Graphics (TOG); vol. 27. 2008,.

[9] Mitani J, Suzuki H. Making papercraft toys from meshes using strip-based approximate unfolding. ACM Transactions on Graphics (TOG) 2004;23(3).

[10] DeCarlo D, Santella A. Stylization and abstraction of photographs. ACM Transactions on Graphics (TOG) 2002;21(3).

[11] Bronson J, Rheingans P, Olano M. Semi-automatic stencil creation through error minimization. In: Non-Photorealistic Animation and Rendering. NPAR; 2008, p. 31–7.

[12] Meng M, Zhao M, Zhu SC. Artistic paper-cut of human portraits. In: Iinternational Conference on Multimedia. MM; 2010, p. 931–4.

[13] Igarashi Y, Igarashi T. Holly: A drawing editor for stencil design. In: Non-Photorealistic Animation and Rendering. NPAR; 2009, p. 8–14.

[14] Xu J, Kaplan CS. Artistic thresholding. In: Non-Photorealistic Animation and Rendering. NPAR; 2008, p. 39–47.

[15] Gerstner T, DeCarlo D, Alexa M, Finkelstein A, Gingold Y, Nealen A. Pixelated image abstraction. In: Non-Photorealistic Animation and Rendering. NPAR; 2012, p. 29–36.

[16] Rother C, Kolmogorov V, Blake A. Grabcut: interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics (TOG) 2004;23(3):309–14.

[17] Blake A, Kohli P, Rother C. Markov Random Fields for Vision and Image Processing. MIT Press; 2011. ISBN 9780262015776.

[18] Freedman D, Drineas P. Energy minimization via graph cuts: Settling what is possible. In: Computer Vision and Pattern Recognition. CVPR; 2005, p. 939–46.

[19] Kolmogorov V, Zabih R. What energy functions can be minimized via graph cuts? Pattern Analysis and Machine Intelligence (PAMI) 2004;26(2):147–59.

[20] Boykov Y, Veksler O, Zabih R. Fast approximate energy minimization via graph cuts. Pattern Analysis and Machine Intelligence (PAMI) 2001;23(11):1222–39.

[21] Nowozin S, Lampert C. Structured learning and prediction in computer vision. Now Publishers; 2011.

[22] Vicente S, Kolmogorov V, Rother C. Graph cut based image segmentation with connectivity priors. In: Computer Vision and Pattern Recognition. CVPR; 2008, p. 1–8.

[23] Zeng Y, Samaras D, Chen W, Peng Q. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in n-d images. Computer Vision and Image Understanding (CVIU) 2008;112(1).

[24] Nowozin S, Lampert CH. Global connectivity potentials for random field models. In: Computer Vision and Pattern Recognition. CVPR; 2009, p. 818–25.

[25] Edelsbrunner H, Harer J. Computational topology: an introduction. American Mathematical Society, Providence, RI; 2010.

[26] Bendich P, Edelsbrunner H, Morozov D, Patel A. The robustness of level sets. In: European Symposium on Algorithms. 2010, p. 1–10.
[27] Chen C, Freedman D, Lampert C. Enforcing topological constraints in random field image segmentation. In: Computer Vision and Pattern Recognition. CVPR; 2011, p. 2089–96.
[28] Blake A, Rother C, Brown M, Perez P, Torr P. Interactive image segmentation using an adaptive gmmrf model. In: ECCV. 2004, p. 428–41.
[29] Neubeck A, Van Gool L. Efficient non-maximum suppression. In: International Conference on Pattern Recognition. ICPR; 2006, p. 850–5.
[30] Lehmann E. Nonparametrics: statistical methods based on ranks (POD). Prentice-Hall: 1st edition (1975). Springer (Berlin): Revised edition; 2006.
[31] Li XY, Ju T, Gu Y, Hu SM. A geometric study of v-style pop-ups: Theories and algorithms. ACM Transactions on Graphics (TOG) 2011;30(4).

Appendix

Proof of Lemma 1. It suffices to show that $f_{i,j}(y) = \sum_{\ell\in\mathcal{L}} [\![y_{i,\ell} \neq y_{j,\ell}]\!]$ is equal to two if $z_i^\pi \neq z_j^\pi$ and zero otherwise. Due to the assumption, we have $y_{i,\ell} = 1$ if $\ell = z_i^\pi$, and zero otherwise. So does $y_{j,\ell}$. If $z_i^\pi = z_j^\pi$, then $y_{i,\ell} = y_{j,\ell}$ for all ℓ, and $f_{i,j}(y) = 0$. If $z_i^\pi \neq z_j^\pi$, then $y_{i,\ell} \neq y_{j,\ell}$ if and only if $\ell = z_i^\pi$ or z_j^π. Then we have $f_{i,j}(y) = 2$. □

Proof of Lemma 2.. We separate $\mathcal{L}$ into the sets of undecided labels and decided labels, $\mathcal{L}_0 = \{\pi_1, \ldots, \pi_{s-1}\}$ and $\mathcal{L}_1 = \{\pi_s, \ldots, \pi_L\}$. The energy of Eq. (3) can be rewritten as

$$\begin{aligned} E(y,\pi) = &\sum_{\substack{i\in\mathcal{V}\\ z_i^\pi\in\mathcal{L}_0}} \sum_{\ell\in\mathcal{L}} \phi_{i,\ell}[\![z_i^\pi = \ell]\!] + \sum_{\substack{i\in\mathcal{V}\\ z_i^\pi\in\mathcal{L}_1}} \sum_{\ell\in\mathcal{L}} \phi_{i,\ell}[\![z_i^\pi = \ell]\!] \\ &+ \beta \sum_{\substack{(i,j)\in\mathcal{E}\\ z_i^\pi\in\mathcal{L}_0 \text{ and } z_j^\pi\in\mathcal{L}_0}} \psi_{i,j}[\![z_i^\pi \neq z_j^\pi]\!] + \beta \sum_{\substack{(i,j)\in\mathcal{E}\\ z_i^\pi\in\mathcal{L}_1 \text{ or } z_j^\pi\in\mathcal{L}_1}} \psi_{i,j}[\![z_i^\pi \neq z_j^\pi]\!] \\ &+ \alpha \sum_{(i,j)\in\mathcal{E}} \sum_{\ell\in\mathcal{L}_0} [\![y_{i,\ell} \neq y_{j,\ell}]\!] + \alpha \sum_{(i,j)\in\mathcal{E}} \sum_{\ell\in\mathcal{L}_1} [\![y_{i,\ell} \neq y_{j,\ell}]\!] \quad (6) \end{aligned}$$

In this equation the terms are separated into the set on which the predicate is unknown (left) and the set on which the predicate is known (right). (The predicate $z_i^\pi \neq z_j^\pi$ is known as long as either z_i^π or z_j^π is known.) All the terms on the right are constants, since we are given y_ℓ, $\ell \in \mathcal{L}_1$. The sum of all terms on the left is equal to the suppressed energy $\widetilde{E}$, namely, the energy model E with $\phi_{i,\ell}$ set to zero for all $z_i^\pi \in \mathcal{L}_1$, and $\psi_{i,j}$ set to zero for all $z_i^\pi \in \mathcal{L}_1$ or $z_j^\pi \in \mathcal{L}_1$. □