ScrapCuts: Fabric Scrap Recomposition with Differentiable Cutline Optimization

Yinfei Qian
The University of Tokyo
Tokyo, Japan
yf-qian@g.ecc.u-tokyo.ac.jp

Quentin Becker
The University of Tokyo
Tokyo, Japan
quentin.becker@gmail.com

I-Chao Shen
The University of Tokyo
Tokyo, Japan
jdilyshen@gmail.com

Masahiro Kono
The University of Tokyo
Tokyo, Japan
marckono2825@g.ecc.u-tokyo.ac.jp

Hennes Rave
University of Münster
Münster, Germany
hennes.rave@uni-muenster.de

Maria Larsson
The University of Tokyo
Tokyo, Japan
ma.ka.larsson@gmail.com

Takeo Igarashi
The University of Tokyo
Tokyo, Japan
takeo@acm.org

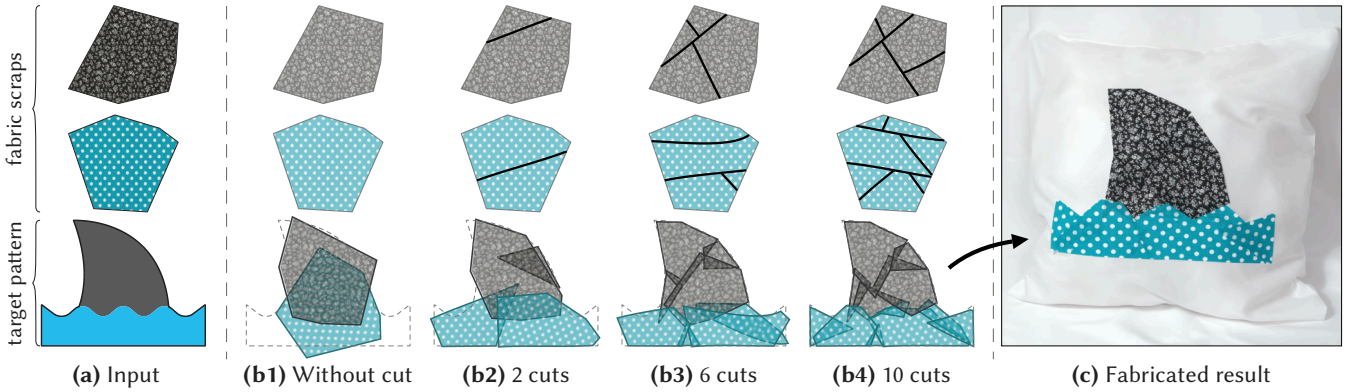


Figure 1: We present a method that recomposes scanned fabric scraps to match a given target pattern using a manageable number of cuts. Starting from the input scraps and target (a), it first optimizes the rigid poses of the uncut scraps (b1), then jointly optimizes cutline geometry and rigid piece poses, progressively improving the match with 2, 6, and 10 cuts (b2)–(b4). We glue the resulting 10-cut scrap composition to a cushion (c). Shark target artwork © Darius Dan.

Abstract

To reuse a fixed set of fabric scraps that might otherwise be discarded, we optimize cutting and piece placement to approximate a target design. Previous systems typically optimize scrap arrangements with fixed piece shapes, although cutting can improve their fit to the target. We present ScrapCuts, a gradient-based formulation that parameterizes cutlines on scraps, then jointly optimizes their geometry with piece poses so that the resulting piece shapes adapt to the target. We progressively introduce candidate cutlines while minimizing shape and color differences to the target, then reduce unnecessary cuts using a cut-sparsity energy. We demonstrate diverse recompositions, show in an ablation that optimizing cutline geometry reduces shape and color differences in every comparison, compare with a simulated-annealing baseline, and fabricate examples using generated cut plans and an assembly guide.

CCS Concepts

• **Computing methodologies** → **Shape modeling**; *Rendering*.

Keywords

Computational fabrication, textile reuse, differentiable rendering

ACM Reference Format:
Yinfei Qian, Quentin Becker, I-Chao Shen, Masahiro Kono, Hennes Rave, Maria Larsson, and Takeo Igarashi. 2026. ScrapCuts: Fabric Scrap Recomposition with Differentiable Cutline Optimization. In *Proceedings of the 11th ACM Symposium on Computational Fabrication (SCF '26), December 06–08, 2026, Tokyo, Japan.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3828611.3845647

This work is licensed under a Creative Commons Attribution 4.0 International License.

SCF '26, Tokyo, Japan

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2821-1/2026/12
https://doi.org/10.1145/3828611.3845647

1 Introduction

Fabric scraps are an inevitable byproduct of textile making. Reusing them can keep usable material from being discarded. Established approaches to reusing textile scraps include patchwork quilts, boro repairs, and textile collages. However, scraps vary widely in size, shape, and color, making it difficult to form an intended composition from a finite collection.

Previous research has developed systems that support scrap-based designs ranging from quilt-block patterns to freeform layouts. These systems assign available fabrics to predefined quilt-block faces, organize rectangular scraps into sequential strip constructions, or rigidly arrange irregular scraps into freeform layouts [Kono et al. 2025; Leake and Daly 2024; Mei et al. 2025]. However, the piece shapes come from the available scraps or a prescribed construction and may not suit a particular target.

Cutting makes piece shape itself a design decision. PatchProv provides an interface in which users manually specify cuts and arrange the resulting fabric pieces [Leake et al. 2021b]. Shinjo et al. match each fixed region in the target design to a visually similar area in the available fabrics [Shinjo et al. 2024]. Neither workflow automatically optimizes cuts and placements together. Cut geometry and placement are tightly coupled because changing a cutline reshapes the resulting pieces, which changes how they can be arranged to match the target.

To address this challenge, we present ScrapCuts, which parameterizes cutlines on the scraps, then jointly optimizes their geometry with the poses of the resulting pieces. Our method takes a fixed set of scraps and a colored target pattern as input. We focus on cases where individual scraps can cover substantial regions of the target but may not fit it intact. A manageable number of cuts can create more suitable pieces without requiring the many cutting and assembly operations of a fine-grained mosaic. We begin by optimizing the rigid poses of the uncut input scraps, then introduce candidate cutlines over successive optimization stages. Although the resulting pieces are placed rigidly, their shapes can still adapt to the target during optimization through changes in cutline geometry. Throughout this process, we minimize shape and color differences between the rendered layout and the target.

For the above optimization, we use a fixed-grid raster representation for each input source scrap. We represent each cutline as a cubic curve parameterized by orientation, offset, and bend. Each candidate cutline defines a soft split of its current piece. These representations allow piece shapes to change continuously without rebuilding polygon topology, enabling differentiable cutline optimization. Later cutlines apply to pieces produced by earlier cuts. Candidate cutlines can become inactive during optimization, and a final cleanup uses a cut-sparsity energy to reduce unnecessary active cuts while preserving visual quality. An accompanying fabrication workflow provides cut plans and an assembly guide.

We evaluate our method by demonstrating optimized recompositions across diverse target patterns. Across 48 paired comparisons on 2 targets, jointly optimizing cutline geometry and piece poses lowers final-output shape-fit error and RGB MSE in every pair. We also compare against Joint-SA, a simulated-annealing (SA) [Kirkpatrick et al. 1983] baseline adapted from ScrapReCover [Kono et al. 2025]. Our method achieves substantially lower median shape-fit error on 3 of 4 targets, although its results for the SCF logo vary more across starts. We fabricate representative recompositions using our fabrication workflow, demonstrating that the optimized designs can be physically realized.

Our main contributions are:

- A novel joint cutting and layout formulation for recomposing a fixed set of fabric scraps, treating piece shapes and their arrangement as coupled design variables for matching a target's shape and color.
- A differentiable cut representation combining parameterized cutlines with complementary soft splits, together with a hierarchical algorithm that progressively introduces cuts and performs joint optimization through differentiable rendering.

2 Related Work

Computational textile design and reuse. Computational tools support a range of patchwork design workflows. Prior systems let users paint original patterns [Igarashi and Mitani 2015], create structured patterns for foundation paper piecing [Leake et al. 2022, 2021a], or interactively cut and arrange photographed fabric pieces on a virtual design wall [Leake et al. 2021b]. Scrap-specific systems obtain piece shapes in different ways. In ScrapMap, Leake and Daly assign available fabrics to predefined quilt-block faces, then lay those faces out for cutting [Leake and Daly 2024]. Mei et al.'s PatchUp extracts maximum inscribed rectangles from irregular scraps, then proposes local trims within restricted families of sequential strip constructions [Mei et al. 2025]. ScrapReCover searches over the selection and rigid placement of user-segmented irregular polygons whose contours remain fixed within each automatic run, leveraging SA-based discrete optimization [Kono et al. 2025].

Other textile-reuse workflows build on target parts or reusable modules. Shinjo et al. [2024] independently associate parts of a processed target image with regions of scanned leftover fabric, while Qi et al. [2025] place target garment panels on source garments and allow controlled deformation to reuse existing seams and hems. Modular garment systems support repeated reconfiguration. Refashion [Lin et al. 2025] decomposes garment patterns into reusable fabric modules while minimizing the number needed for assembly. The texTile system [Del Valle et al. 2025] fits garment patterns to crochet tiles and body measurements, and supports reconfiguration through reversible connectors and plans that preserve existing joins.

Across these systems, piece boundaries are set manually, constrained by particular module or construction families, derived from target parts, or fixed before automatic placement. Our formulation instead jointly optimizes source cutline geometry and the rigid placement of all resulting pieces.

Cutting, decomposition, and material layout. Cutting and layout methods differ in whether part shapes are fixed, designed alongside the layout, or created through decomposition. Classical irregular nesting optimizes non-overlapping placements of fixed-shape items within stock material [Bennell and Oliveira 2008]. Related material-aware interfaces capture irregular stock or track available sheets while users arrange or design the parts to be cut [Saakes et al. 2013; Sethapakdi et al. 2021]. Other work links part design to material layout. Koo et al. couple parametric part dimensions with

sheet-material layouts in zero-waste furniture design [Koo et al. 2017], while Zhang et al. link edits across neighboring garment panels to preserve zero-waste layout constraints [Zhang et al. 2024]. Duncan et al. co-design a small common set of pieces and two non-overlapping rigid arrangements that approximate untextured input shapes [Duncan et al. 2017]. Their formulation does not model a fixed set of source pieces. Qi et al.'s Voronoi Scissors [Qi et al. 2026] computes an approximate decomposition of two input shapes into a common set of non-overlapping pieces by coupling continuous Voronoi-based partitioning with rigid piece matching. It also supports material reuse by holding one input shape fixed while approximating the other. Limper et al. alternate strategic axis-aligned cuts with repacking to remove voids from atlases or fabrication panels [Limper et al. 2018]. Iseringhausen et al.'s Computational Parquetry matches single-use regions of scanned wood veneer to target-image patches and supports fabricable boundary refinement [Iseringhausen et al. 2020]. It constructs a non-overlapping partition of the target and therefore does not model layer order. Its grid-based or post-matching boundary refinements do not treat source cutline geometry as a variable jointly optimized with the rigid target-space poses of all resulting pieces.

Related coupling also appears in 3D fabrication. Chen et al. jointly decompose and pack an input object within a 3D printer volume using cut-and-pack moves [Chen et al. 2015], while Baas et al. discover panels to cut from existing 3D surfaces and rigidly register them to a target surface [Baas et al. 2025]. Both determine part shapes together with their placements, but their objectives differ. Chen et al. optimize packing within a printer volume, whereas Baas et al. seek a non-overlapping approximation of a 3D target surface. Our formulation instead jointly optimizes cuts and rigid placements of planar scraps with respect to target shape and color. These placements may overlap, with visibility resolved by the optimized layer order.

Differentiable shape and layout optimization. Differentiable rasterizers expose image-space gradients to planar geometry, including simplex meshes [Jiang et al. 2019] and filled and stroked vector paths [Li et al. 2020]. Reddy et al. use differentiable compositing to recover the type, pose, and layer order of input pattern elements [Reddy et al. 2020], while Hong et al. provide differentiable transforms, color, opacity, and layered composition for large collections of masked raster elements [Hong et al. 2026]. Together, these methods establish that image-space objectives can drive the optimization of shape, pose, appearance, and visibility parameters.

Computational collage addresses related tasks. Huang et al. generate Arcimboldo-like collages by segmenting a target image, retrieving and affine-matching Internet cutouts, and inferring their layer order [Huang et al. 2011]. Mirowski et al. optimize affine transforms, colors, and soft order weights for textured patches [Mirowski et al. 2022]. Shao et al. optimize translation, rotation, and scale of fixed foreground cutouts under text guidance [Shao et al. 2026], while Wang and Lu optimize transforms of input shapes through differentiable containment and non-overlap objectives [Wang and Lu 2025]. Neural Collage Transfer also couples crop selection and placement: each learned action chooses a quadrilateral crop from a digital material image together with its target placement [Lee et al. 2023]. At each step, however, its MDP supplies or selects a new material image instead of modeling the unused complement as persistent source state.

Compared with these works, we optimize where to cut a fixed set of source scraps and how to place all resulting pieces.

Table 1: Core terminology and notation.

Term or notation	Meaning
Source scrap s	Flat, fixed-scale fabric input
Piece i	Region obtained after zero or more cuts
Cutline	Oriented curve in source space
Occupancy field	Fixed-grid raster representation of a 2D shape
(Ω_s, O_s, I_s)	Source space, occupancy field, and RGB image
(Ω_t, O_t, I_t)	Target space, occupancy field, and RGB image
$O_i,\ \alpha_i$	Piece occupancy field and target-space opacity
$g_i,\ z_i$	Rigid target-space pose and layer variable
E_{fit}	Combined shape and color fitting energy
$E_{\text{shape}},\ E_{\text{color}}$	Shape-fitting term and RGB MSE

3 Method Overview

We formulate the coupled cut-and-place problem as a joint continuous optimization over cutline geometry, piece poses, and layer variables. We solve it through a coarse-to-fine sequence of optimization stages in which candidate cutlines are progressively introduced. A workflow diagram of our method is shown in Figure 2, and Table 1 summarizes our core terminology and notation.

Source scraps and target pattern. Given a fixed set of flat, fixed-scale source scraps and one target pattern, our goal is to determine how to cut the source scraps and place the resulting pieces so that their assembly visually resembles the target pattern. We retain all resulting pieces in the composition and allow them to overlap. We use a fixed-grid raster representation for source scraps and the target pattern. Throughout, subscripts s and t denote a source scrap and the target, respectively. Each source scrap $s \in \{1, \dots, S\}$ is represented in its fixed source space $\Omega_s \subset \mathbb{R}^2$ by a 2D occupancy field $O_s : \Omega_s \to [0, 1]$ and an RGB image $I_s : \Omega_s \to [0, 1]^3$. Similarly for the target pattern with $\Omega_t \subset \mathbb{R}^2$, $O_t : \Omega_t \to \{0, 1\}$, and $I_t : \Omega_t \to [0, 1]^3$.

Scraps are cut and combined into a *scrap composition.* A scrap composition is defined by the shape of cutlines, the positioning of the resulting cut scraps, and their layer ordering (controlling visibility in overlapping regions).

Fitting. The scrap-fitting problem treats the number of cuts as a discrete optimization variable. Exploring all possibilities is computationally prohibitive, so we instead solve a continuous relaxation. In practice, we allow cut lines to be deactivated by moving them away from the source scraps, and we smoothly regularize their number to limit fabrication complexity. We then update the scrap composition variables by continuous optimization so that the composited pattern best matches the target. Our objective E favors assemblies that cover the target shape, avoid spilling outside it, and minimize color error. To explore this discrete, non-convex search space and avoid poor local minima, we optimize multiple cut-insertion trajectories in parallel.

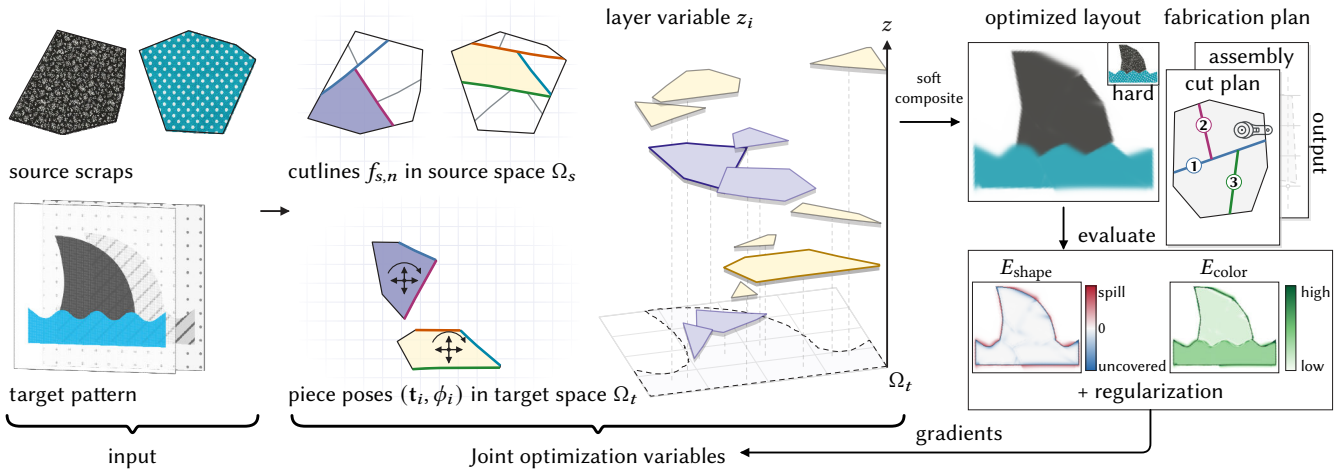


Figure 2: Workflow of our method. Given a fixed set of source scraps and a target pattern, we jointly optimize source-space cutlines, target-space piece poses, and per-piece layer variables to match the scrap composition to the target pattern. E_{shape} penalizes uncovered target area and spill, while E_{color} compares this composition with the target image. The resulting gradients update all variables. The optimization result then defines paper-based source cut plans and an assembly guide for fabrication.

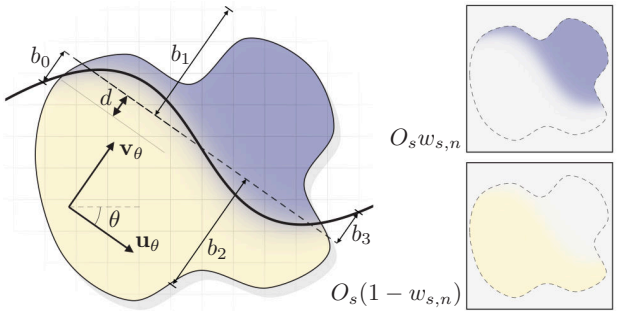


Figure 3: An oriented offset–bend cubic cutline and its complementary soft split of a current piece occupancy field.

Fabrication. The method outputs actual-size source cut plans and an assembly guide. The cut plans show how each piece is cut from its source scrap, while labeled paper carriers preserve the piece's identity and orientation after cutting. The assembly guide uses matching edge guides to show where and how to place each piece in the final bottom-to-top layer order. We describe this fabrication workflow in detail in Section 6.

4 Scrap Composition

The process of cutting and arranging the scraps yields an instance of a scrap composition. We detail our *differentiable* composition process, enabling downstream tasks such as pattern fitting.

4.1 Cutline Representation

We define a cut as an operation that divides a fabric scrap into at least two pieces. A cutline specifies where this division occurs. We first describe how candidate cutlines are parameterized, then how their effect on the fabric is represented during optimization.

Parameterization. As shown in Figure 3, we represent candidate cutline n of source scrap s in source space Ω_s as the rotated cubic polynomial function

$$f_{s,n} : [0,1] \to \mathbb{R}^2 \qquad t \mapsto R_{\theta_{s,n}} \cdot \begin{bmatrix} h_s(2t-1) \\ d_{s,n} + C_{s,n}(t) \end{bmatrix}, \tag{1}$$

where $R_{\theta_{s,n}} := [\mathbf{u}_{\theta_{s,n}}, \mathbf{v}_{\theta_{s,n}}] \in \mathrm{SO}(2)$, is the in-plane rotation matrix of angle $\theta_{s,n} \in [0, 2\pi)$. The *scrap length* h_s is such that the box $[-h_s, h_s]^2$ contains the centered scrap, and $d_{s,n} \in [-h_s, h_s]$ offsets the curve along $\mathbf{v}_{\theta_{s,n}}$. Additionally, $C_{s,n}$ is a cubic Bernstein polynomial with coefficients $\mathbf{b}_{s,n} \in \mathbb{R}^4$ such that $\sum_k b_{s,n,k} = 0$ to avoid parameterization redundancy with $d_{s,n}$. We pick $b_{s,n,k} \in [-1.2h_s, 1.2h_s]$ in our experiments, which we find strikes the right balance in terms of cut complexity and expressivity. If all *bend values* are simultaneously zero *i.e.*, $\mathbf{b}_{s,n} = 0$, the cutline becomes straight.

Soft splitting. During optimization, we represent the two sides of each candidate cutline with a smooth side-assignment field $w_{s,n} : \Omega_s \to [0,1]$ and its complement. Let O_s denote the occupancy field of the current piece to which candidate cutline n is applied. For $\mathbf{x} \in \Omega_s$, we first define

$$w_{s,n}(\mathbf{x}) = \sigma\left(\frac{v_{\theta_{s,n}}(\mathbf{x}) - \left[d_{s,n} + C_{s,n}(t_{s,n}(\mathbf{x}))\right]}{\tau_c}\right), \tag{2}$$

where σ is the sigmoid function, τ_c controls cut softness, and the projected coordinates are obtained as

$$\begin{bmatrix} u_{\theta_{s,n}}(\mathbf{x}) \\ v_{\theta_{s,n}}(\mathbf{x}) \end{bmatrix} := R_{\theta_{s,n}}^{\top} \cdot \mathbf{x}, \quad t_{s,n}(\mathbf{x}) = \frac{1}{2}\left(\frac{u_{\theta_{s,n}}(\mathbf{x})}{h_s} + 1\right).$$

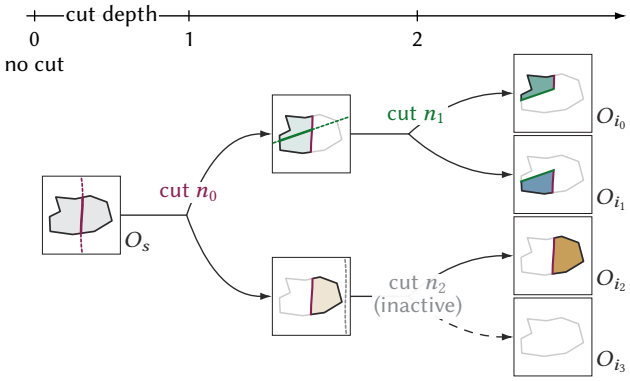


Figure 4: Recursive cut construction through depth $D = 2$. Cut n_0 produces two depth-1 pieces, and cuts n_1 and n_2 produce the depth-2 piece fields. Cut n_2 is inactive, leaving O_{i_3} nearly empty.

The two resulting piece occupancy fields are obtained from the soft intersection of the current piece occupancy field with the side-assignment field and its complement

$$O_{s,n}^{(0)}(\mathbf{x}) = O_s(\mathbf{x})w_{s,n}(\mathbf{x}), \quad O_{s,n}^{(1)}(\mathbf{x}) = O_s(\mathbf{x})(1 - w_{s,n}(\mathbf{x})). \tag{3}$$

The two resulting fields sum pointwise to O_s and vary smoothly with all cutline parameters. All the cutline variables are gathered into $\mathbf{p}_{s,n} := [\theta_{s,n}, d_{s,n}, \mathbf{b}_{s,n}^\top]^\top \in \mathbb{R}^6$. Further details are given in Appendix A.

4.2 Cutlines Hierarchy

Each scrap s is cut recursively into candidate pieces. We represent the process using a binary tree where nodes are cut scraps and leaves are the pieces available for composition, see Figure 4. The resulting piece occupancy fields are denoted by O_i.

We include candidate cutlines up to a maximum depth D, bounding the number of cuts for source scrap to $2^D - 1$. To characterize the candidate cutlines in this hierarchy, we introduce the notions of *inactive* and *proper* cuts. Cuts are classified based on the values of the *cutline occupancy function*.

Inactive cuts. An inactive cut is a candidate cut whose cutline lies outside its current piece. It leaves the piece effectively uncut. Therefore, one of the two smooth children occupancy fields from Equation 3 is nearly unchanged and the other nearly empty. We retain the nearly empty child as a candidate piece, preserving a signal that can move the cutline back into the piece if needed. Inactive cuts remain in the hierarchy during optimization and are omitted only from the final cut plan.

Proper cuts. A candidate cut does not necessarily realize a single boundary-to-boundary cut of its current piece. We call an active cut *proper* when its cutline splits a connected, hole-free piece into two physical pieces (Figure 6).

A cutline may instead leave and later re-enter the piece, especially when the piece is non-convex or the cutline is strongly bent, producing two or more disjoint intersection intervals. We call such an active cut *improper*. One of the two children nodes contains two or more disconnected physical patches which our representation treats as a single entity positioned in the plane. Those grouped components cannot be placed independently (Figure 5). We challenge this representation against other options in Section 8. The regularizer in Section 5 discourages rather than rules out improper cuts. This allows transient improper states during optimization, e.g., as a cutline moves out of a non-convex piece and becomes inactive.

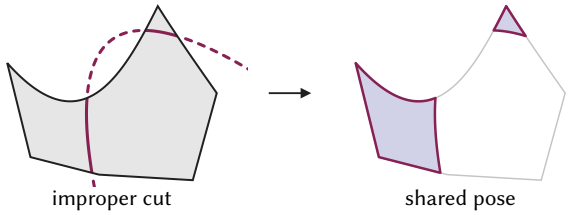


Figure 5: Disconnected patches in one child node share a pose and cannot be placed independently. The pale outline shows the original piece.

Cutline occupancy function. We distinguish cut types using the cutline occupancy function $a_{s,n} : [-h_s, h_s] \to [0, 1]$, which aggregates the occupancy around each point along the curve. Observing the field $w_{s,n}(1 - w_{s,n})$ forms a narrow band around the cutline, we define

$$a_{s,n}(u) \propto \int_{\mathcal{L}_{s,n,u}} w_{s,n}(\mathbf{x})(1 - w_{s,n}(\mathbf{x}))O_s(\mathbf{x})\mathrm{d}\mathbf{x} \tag{4}$$

where $\mathcal{L}_{s,n,u}$ is the line oriented along $\mathbf{v}_{\theta_{s,n}}$ passing by the cutline point $f_{s,n}(t_{s,n}(u))$. We normalize each value by the weighted line integral $\int_{\mathcal{L}_{s,n,u}} w_{s,n}(\mathbf{x})(1 - w_{s,n}(\mathbf{x}))\mathrm{d}\mathbf{x}$. The band's effective width is controlled by τ_c. We typically set $\tau_c = 0.03$ in normalized source coordinates, corresponding to an effective band width of approximately $0.13h_s$ at half maximum. More practical implementation details can be found in Appendix B.

As illustrated in Figure 6, the above-threshold intervals of $a_{s,n}$ approximate the cutline–piece intersections: a proper cut gives one interval, an improper cut gives multiple separated intervals, and an inactive cut gives no sustained response.

4.3 Compositing

Compositing a set of cut scraps consists in (i) positioning the pieces on the canvas, and (ii) choosing their layer ordering. The shape U and appearance $\hat{I}$ of the resulting scraps composition are defined and reused for target pattern fitting in Section 5.

Target-space placement. Each piece i is positioned following $g_i = (\mathbf{t}_i, \phi_i) \in \mathrm{SE}(2)$, a translation and rotation. Each piece rotates around a fixed source-space pivot $\boldsymbol{\rho}_i$, set to its centroid when the piece is created and kept constant.

The shape of the union can be readily obtained in the form of a smooth occupancy function $U : \Omega_t \to [0, 1]$. Let $\mathcal{W}$ denote bilinear inverse warping [Jaderberg et al. 2015] from the piece's source space to Ω_t. The target-space opacity is

$$\alpha_i(\mathbf{y}) = \mathcal{W}(O_i; g_i, \boldsymbol{\rho}_i)(\mathbf{y}), \qquad \mathbf{y} \in \Omega_t. \tag{5}$$

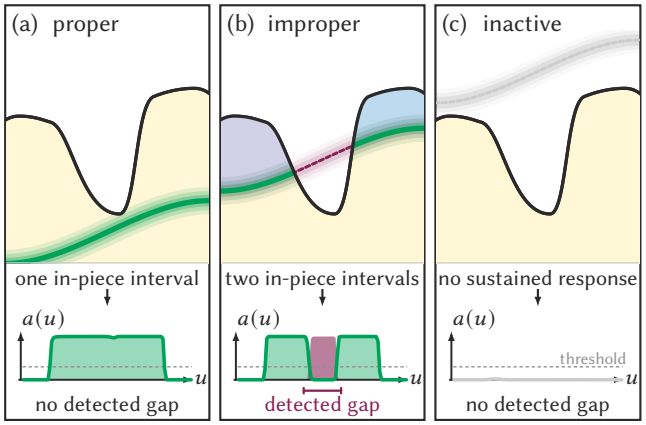


Figure 6: Cutline–piece intersection patterns. The function $a(u)$ measures the current piece's occupancy within a soft band around the cutline. Portions above the dashed threshold define the intersection intervals. (a) A proper cut has one contiguous in-piece interval. (b) An improper cut has separated in-piece intervals with a gap. (c) An inactive cut has no sustained in-piece response.

We form an order-independent soft union of the target-space opacities [Liu et al. 2019]:

$$U(\mathbf{y}) = 1 - \prod_i (1 - \alpha_i(\mathbf{y})). \tag{6}$$

Because α_i depends on both g_i and O_i, the union smoothly varies with both piece poses and cutline parameters.

Layer order and color compositing. Where opaque pieces overlap, their layer order determines which source color is visible. Each piece receives a scalar variable z_i, with larger values corresponding to higher layers as shown in Figure 2. Final rendering sorts these values to obtain a discrete layer order. Inspired by differentiable layer-order compositing [Reddy et al. 2020], we use smooth pairwise comparisons: For $i \neq j$, $P_{ij} = \sigma((z_j - z_i)/\tau_z)$ estimates whether piece j lies in front of piece i. The temperature $\tau_z > 0$ controls the softness of this comparison. These comparisons give piece i the visibility weight

$$\tilde{V}_i(\mathbf{y}) = \alpha_i(\mathbf{y}) \prod_{j \neq i} \left(1 - P_{ij}\alpha_j(\mathbf{y})\right). \tag{7}$$

We define the normalized piece weights $V_i(\mathbf{y}) \propto \tilde{V}_i(\mathbf{y})$ and normalized background weights $V_{\text{bg}}(\mathbf{y}) \propto 1 - U(\mathbf{y})$ so they sum to 1 at each $\mathbf{y} \in \Omega_t$. Let $\mathbf{c}_i \in [0,1]^3$ be the mean RGB color of piece i's source scrap and let $\mathbf{c}_{\text{bg}} = (1,1,1)$ be the background color. The composited scrap appearance is

$$\hat{I}(\mathbf{y}) = V_{\text{bg}}(\mathbf{y})\mathbf{c}_{\text{bg}} + \sum_i V_i(\mathbf{y})\mathbf{c}_i. \tag{8}$$

Compared to the opaque alternative, this soft formulation provides gradients to the lower layers when z_i values are close. As an undesired byproduct, it may also blend colors that no single ordering of opaque pieces can reproduce. We extend this gap during optimization by annealing τ_z and penalizing near-ties between overlapping pieces. For final rendering and fabrication, we sort the optimized z_i values and use alpha compositing with the original source RGB images in the resulting layer order.

We gather the compositing variables into $\mathbf{p}_i = [g_i, z_i] \in \mathbb{R}^4$. The *scrap composition variables* $\mathbf{p}$ are therefore the concatenation of all cutline variables $\mathbf{p}_{s,n}$ and all compositing variables $\mathbf{p}_i$.

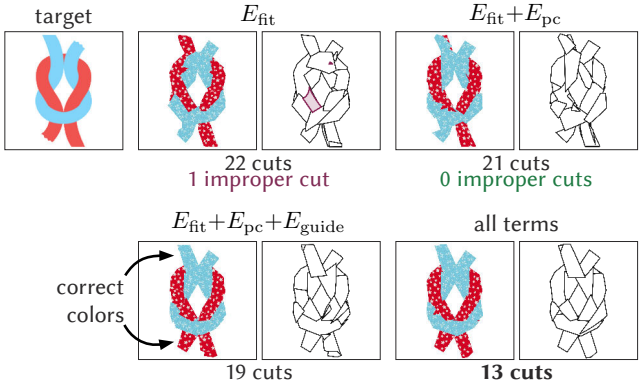


Figure 7: Scrap fitting terms ablation. Given the same target pattern, our scrap fitting algorithm progressively recovers a matching scraps composition with sparse and proper cuts as regularization terms are added to the objective. Knot target artwork © Google (Noto Emoji).

5 Scraps Fitting

Scraps fitting seeks the most faithful composition of a given set of cut source scraps to a target pattern. Following our representation, we allow a maximum of $2^D - 1$ cuts per scrap and mollify the cutting process. The fitting problem results in a continuous optimization problem in the scrap composition variables $\mathbf{p}$,

$$\min_{\mathbf{p}} E_{\text{fit}} + R^{(k)}, \tag{9}$$

where $R^{(k)}$ denotes regularization terms we used to enhance the fitting results. As detailed in Section 5.2, our optimization algorithm is multi-staged and progressively adds cutlines. Within each stage, we denote the optimization step by k.

Fitting is further split into shape and color matching as

$$E_{\text{fit}} = E_{\text{shape}} + \lambda_{\text{color}} E_{\text{color}}. \tag{10}$$

Let $|O_t| = \sum_{\mathbf{y} \in \Omega_t} O_t(\mathbf{y})$ denote the target area. We measure shape fitting with a normalized mask error and color with RGB MSE over the full target space:

$$E_{\text{shape}} = \frac{\|U - O_t\|_1}{|O_t|}, \qquad E_{\text{color}} = \frac{\|\hat{I} - I_t\|_2^2}{3|O_t|}. \tag{11}$$

Both errors are evaluated over the full target space, with E_{color} summing over all three RGB channels. Because O_t is binary, E_{shape} penalizes uncovered target regions and spill equally. The RGB MSE over the full target space also penalizes colored spill. Normalizing by target area makes both terms comparable across target sizes.

5.1 Regularization

Exclusively optimizing the target-matching terms E_{fit} produces scrap compositions with improper cuts that are prone to bad local minima (Figure 7). We therefore introduce three regularization terms we weight appropriately at optimization step k

$$R^{(k)} = \lambda^{(k)}_{\text{guide}} E_{\text{guide}} + \lambda^{(k)}_{\text{pc}} E_{\text{pc}} + \lambda^{(k)}_{\text{sparse}} E_{\text{sparse}}. \tag{12}$$

The guidance and cut-regularization terms guide pieces toward compatible target regions, discourage improper cuts, and favor fewer active cuts when target matching is preserved.

Color guidance E_{guide}. After a cut, a piece may be far from a target region of similar color. Local RGB MSE does not reliably point toward that region, and occlusion can weaken this signal further because the MSE is evaluated after layer compositing.

E_{guide} provides a long-range signal instead, using low-resolution entropic transport [Cuturi 2013] computed before compositing and independent of layer order. Every piece is rendered separately and downsampled to its own 16×16 grid, which together form the source. The target is downsampled to a single grid of the same size. One transport plan is solved jointly over all pieces, softly assigning source mass to target cells under a cost that combines spatial distance and color difference. This way, pieces are drawn toward regions of similar color. Each target cell has a fixed capacity, which prevents pieces from collapsing into a single best-matching region.

As the scraps may cover more area than the target, we use partial transport in which an auxiliary cell absorbs the surplus source mass, so both marginals are normalized and pieces are not forced onto the target. The weight $\lambda^{(k)}_{\text{guide}}$ starts high in each core stage and decays to zero as the RGB MSE takes over.

Proper-cut regularization E_{pc}. We introduce a proper-cut regularizer to discourage leave-and-re-enter improper cuts as mentioned before. We build it directly from the soft split defined above based on two observations. First, along each candidate cutline n, we measure how much of its soft band lies inside the current piece. A gap between two in-piece intervals produces a positive, differentiable measure e_n. The measure vanishes for a proper or inactive cut (Figure 6).

Second, we delay this penalty when such a gap may be transient. On a non-convex piece, a cutline moving out of the piece may briefly leave and re-enter it before becoming inactive. Penalizing this intermediate state can prevent the cut from becoming inactive. We therefore keep a candidate-specific weight $\gamma^{(k)}_n \in [0, 1]$ at zero for a grace period and increase it only if the gap persists:

$$E^{(k)}_{\text{pc}} = \text{mean}_n \left[\gamma^{(k)}_n e_n \right]. \tag{13}$$

The grace period lets a cut pass through a transient improper state and become inactive. A persistent leave-and-re-enter cut receives an increasing penalty. The gap measure and schedule are detailed in Appendix B.

Cut sparsity E_{sparse}. Although candidate cutlines can naturally become inactive during optimization, we introduce a cleanup stage with E_{sparse} after the maximum-depth stage to further reduce unnecessary cuts for fabrication. This final cleanup encourages as many cutlines as possible to become inactive while preserving the visual quality of the recomposition.

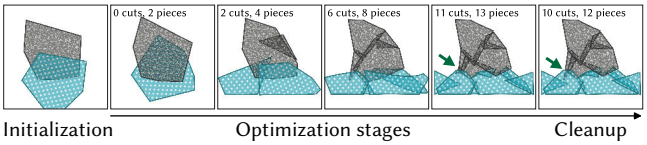


Figure 8: Multi-stage optimization of the shark example. The panels show the initialization, the final states of core stages 0–3, and the final state after cleanup. The green arrows mark the cut removed by E_{sparse} during cleanup.

For candidate cutline n of source scrap s, let $q_{s,n} \in [0, 1]$ be the fraction of its current piece assigned to one side. We define the smaller-side fraction and the basic cut-sparsity energy as

$$m_{s,n} = \min(q_{s,n}, 1 - q_{s,n}), \qquad E_{\text{sparse}} = \sum_{s,n} \frac{m_{s,n}}{m_{s,n} + \tau_{\text{sparse}}}. \tag{14}$$

Here, $\tau_{\text{sparse}} > 0$ sets the scale of the soft count. Each summand is near one when both sides contain substantial material and approaches zero as one side becomes empty. Minimizing the sum therefore favors fewer active cuts.

During cleanup, we continue to optimize all variables and gradually increase the sparsity weight. For color-fitting runs, we cap its contribution relative to the weighted RGB MSE so that cut sparsity does not dominate fitting. The exact computation of $q_{s,n}$ and the cleanup schedule are given in Appendix D.

5.2 Optimization Algorithm

Our optimization follows the recursive cut construction through several parallel trajectories. Starting from the uncut scraps, these trajectories progressively introduce candidate cutlines up to depth D, followed by a final cut-cleanup stage.

Progressive cut introduction. We optimize in stages as new candidate cutlines are introduced. We begin with the uncut scraps and proceed to the maximum cut depth D. Each stage sets an upper bound on the number of active cuts. Some candidate cutlines may remain inactive (Section 4.2). At each stage, we jointly optimize all current cutline parameters, piece poses, and layer variables by minimizing Equation 9. Figure 8 shows the initialization, the final states of core stages 0–3, and the final state after cleanup.

Between successive stages, we add one cutline to each current piece. The new stage inherits the final state of the previous stage. Each new cutline is initialized to produce a roughly balanced, compact split. The piece poses and layer variables of the two resulting pieces are initialized from those of the piece being divided. We give them slightly different initial poses and then jointly optimize all existing and newly introduced variables. Within each stage, color guidance fades as the RGB MSE takes over. We lower τ_z over the stage and activate layer-variable separation late, allowing the layer variables to adapt while the pieces move and stabilizing the final layer order. Proper-cut weights are scheduled independently for each candidate, so transient gaps do not immediately constrain the cutline while persistent gaps are progressively penalized. The exact schedules are reported in Appendix B and Appendix C.

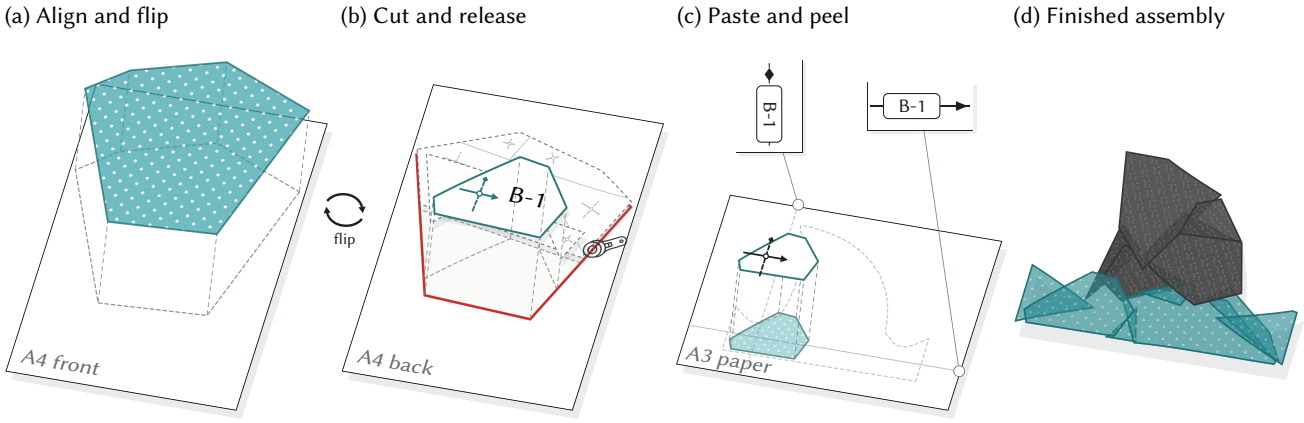


Figure 9: Fabrication workflow for a 12-piece shark result. (a) The fabric is placed front-side down on the mirrored side of an A4 template and flipped. (b) On the reverse side, the black internal cuts are made in numbered order, and the red source outline is cut last, releasing each piece with its labeled paper carrier. The diagram highlights the three cuts that form B-1. The light-gray lines show the remaining cut plan. (c) The solid x-axis and dashed y-axis on B-1's carrier are aligned with the matching guide lines on the A3 template. The insets show the rightward $+x$ triangle and upward $+y$ diamond. The exposed fabric back is adhered to the growing assembly, and the carrier is peeled away. (d) Repeating this operation in the final bottom-to-top layer order completes the assembly. This figure is schematic; fabrication uses the full-scale source cut plans and assembly guide exported by our system.

Multi-start exploration. The energy is nonconvex even at depth zero. We therefore optimize several trajectories in parallel. Each trajectory initializes the uncut scraps near color-compatible target regions, with independent position perturbations, rotations, and layer variables, and retains its own optimized state as the cut depth increases. At each new depth, each trajectory receives its own initialization of the newly introduced cutlines and small perturbations to the poses and layer variables of the resulting pieces. Together, these inherited states and per-trajectory initializations let the trajectories explore different basins as candidate cutlines are introduced. We carry all trajectories through the maximum-depth stage. In practice, we use 8–32 starts.

5.3 Implementation Details

Table 2 summarizes our main hyperparameters. Detailed regularization schedules are given in Appendices B–D. We solve Equation 9 with Adam [Kingma and Ba 2015] ($\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$) using PyTorch automatic differentiation [Paszke et al. 2019]. In standard core stages, initial learning rates are 0.01 for piece poses and cutline parameters and 0.005 for layer variables, smoothly halved over each stage's final 20%. Cleanup inherits the Adam state. Unless otherwise noted, we use a 192×192 raster target space and run 1,000 optimization steps per core stage and 1,000 cleanup steps. A small feasibility term encourages each nonempty piece to intersect the target space. We run all experiments on an NVIDIA GeForce RTX 4080 GPU. Across the five standard examples in Figure 11 (excluding Flag), the optimization batch that produced the retained result, including the subsequent 1,000-step cleanup, took approximately 3.3 minutes per example on average. We additionally flag results that contain improper cuts or pieces lying entirely outside the target space. Note that, such results can still match the target well and remain fabricable. None of the results shown in Figure 11 contains either case.

Table 2: Main hyperparameters and typical settings.

Parameter	Meaning and setting
D	Maximum cut depth (3 or 4 in the gallery)
τ_c	Cut softness (typically 0.03)
τ_z	Layer-order temperature (0.9 to 0.005, geometrically annealed per core stage)
λ_{color}	RGB MSE weight (1 in standard color-fitting core stages)
$\lambda^{(k)}_{\text{guide}}$	Color guidance (3 to 0 per core stage; off in cleanup)
λ_{pc}	Proper-cut weight (0.02)
λ_{zsep}	Layer-separation weight (20 in color fitting)
$\bar{\lambda}_{\text{sparse}}$	Basic soft-count weight (nominal max. 0.01)
τ_{sparse}	Basic soft-count scale (0.02)

6 Fabrication Workflow

Fabrication of our results can be guided in many ways, including projected instructions or specialized hardware. Here, we demonstrate a convenient paper-based workflow designed for standard A4 and A3 printing (Figure 9). In our examples, the longest dimension of every input scrap is below 20 cm, so its outline fits on an A4 sheet. We use an A3 sheet for the complete assembly layout. Larger scraps or targets simply require larger printed sheets.

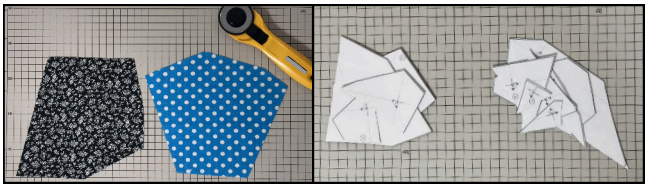
We generate one two-sided cut template per input scrap and one assembly template for the target. Printed at full scale, these templates directly guide cutting and assembly. Each cut-plan side also lists the final layer order from bottom to top.

This workflow addresses two practical problems. After cutting, loose pieces can lose their identity and orientation. During assembly, earlier pieces can hide the guides needed to place later ones. We

keep each piece attached to a labeled part of its A4 template as a temporary paper carrier. We extend the A3 alignment guides to the page edges so they remain usable as the assembly grows.

Cutting with paper carriers. Each A4 cut template has a mirrored alignment side and a cut-plan side. The maker places the fabric's front side against the source outline on the alignment side, secures it with temporary fabric adhesive, and flips the paper-and-fabric sandwich over its long edge. The mirrored outline makes the cutlines on the reverse side match the fabric in its intended orientation.

The black cutlines are numbered so that each cut is completed before either resulting region is divided again. The surrounding paper keeps the fabric in place during these internal cuts. The red source outline is cut last to release the individual paper carriers. Each carrier remains attached to the front of its matching fabric piece, with the piece ID and a two-axis alignment mark facing upward. The piece can therefore be moved to the assembly template without being flipped or losing its identity. Figure 10 shows the input scraps and the resulting pieces with their paper carriers attached before placement.



(a) (b)

Figure 10: Fabrication materials for the shark example. (a) Input scraps before cutting. (b) Pieces after cutting, with their paper carriers attached, ready for assembly.

Placing pieces with edge guides. Each carrier has a solid x-axis and a dashed y-axis centered at a point inside the piece. We orient this mark so that the two axes become horizontal and vertical at the piece's intended placement. The A3 template contains matching guide lines with the same piece ID, extended to the page edges. A triangle marks rightward $+x$, and a diamond marks upward $+y$. Matching the center and directions of the carrier mark to these guide lines fixes the piece's 2D position and orientation.

We place the pieces in the final bottom-to-top layer order. Earlier pieces can cover the central parts of guide lines needed later, so we choose the carrier marks and guide lines such that at least one page-edge segment of each axis remains visible when that piece is placed. The maker extends the two visible segments with taut threads or straightedges held above the growing stack, then slides and rotates the carrier until its mark matches both guides. Once aligned, the exposed fabric back is adhered to the growing assembly with temporary fabric adhesive. The guides are removed, and the paper carrier is peeled upward, revealing the fabric front without flipping the piece. Repeating these steps yields the optimized assembly. For the shark example, temporary bonds at overlaps held the composition together as we lifted it from the assembly sheet, before attaching it to a cushion with permanent fabric glue. Fabrication took approximately 45–60 minutes, including cutting and assembly. The completed shark assembly is shown in Figure 1(c).

7 Results

We evaluate our method on six recomposition tasks with different target geometries, color layouts, and source scraps. We first present a gallery of the optimized layouts and their cut plans. We then isolate the effect of cutline optimization through a paired ablation with frozen cuts. Finally, we compare our method with Joint-SA, which modifies ScrapReCover [Kono et al. 2025] for joint cut-and-place optimization, across successive cut depths.

Recomposition gallery. Figure 11 presents six recompositions from sets of two or three irregular source scraps. The target shapes include curved silhouettes, straight-edged patterns, and multiple disconnected components. Their color layouts range from broad regions to narrow bands. We chose target sizes through exploratory runs, favoring larger targets to limit excess material when visual quality and cut count were comparable. The configured target-to-source area ratios are 0.80 for Shark, Moon/cloud, and Knot, 0.70 for Island and SCF logo, and 0.65 for Flag. These ratios remain fixed within each optimization run. Gallery cleanup settings were adjusted to balance fit and cut count. Our shape-only comparisons (Appendices E–F) use fixed cleanup settings across targets.

Across these examples, our method adapts the source scraps to match the main target shapes and color regions. The source cut plans and piece maps trace each placed piece back to its input scrap. These results also confirm that candidate cutlines become inactive during optimization. The examples use cut depths D = 3 or D = 4 and contain 10–21 pieces. Although optimization introduces 14–45 candidate cutlines, the inactive-cut property of our method allows the final layouts to use only 8–18 cuts. Hence, fewer cutting operations are required during fabrication.

Ablation study. An ablation study was conducted to evaluate the effectiveness of optimizing cutlines jointly with piece placement. As shown in Figure 12, we compare joint optimization with a frozen-cut condition on two targets of different complexity of target patterns. At each refinement stage, both variants initialize new cutlines using the same roughly balanced, compact split procedure (Section 5.2), with small bend perturbations. The frozen-cut condition keeps each cutline's orientation, offset, and bend fixed after initialization while continuing to optimize piece poses and layer variables. Joint optimization achieves lower shape-fit error and RGB MSE in every paired run, improving the final recompositions. The difference is larger for the SCF logo, whose geometry is more complex.

We also compared straight and cubic cutlines on four shape-fitting tasks at maximum depth $D = 3$, with 24 paired starts per task and final cleanup in both conditions. The median paired reduction in shape-fit error with cubic cutlines ranged from 0.92 to 3.59 percentage points across tasks. Cubic runs also produced more geometric-check failures, reported alongside all outcomes in Appendix F.

We further compared cubic cutlines at maximum depths $D = 2, 3, 4$, with 24 paired starts per task and final cleanup at each depth limit. Increasing depth from 3 to 4 yielded smaller median paired fitting gains than from 2 to 3 on all four tasks, while taking approximately 2.4–2.6 times the batch runtime and increasing median active-cut counts. Appendix F reports this tradeoff between fitting quality, active cuts, and runtime.

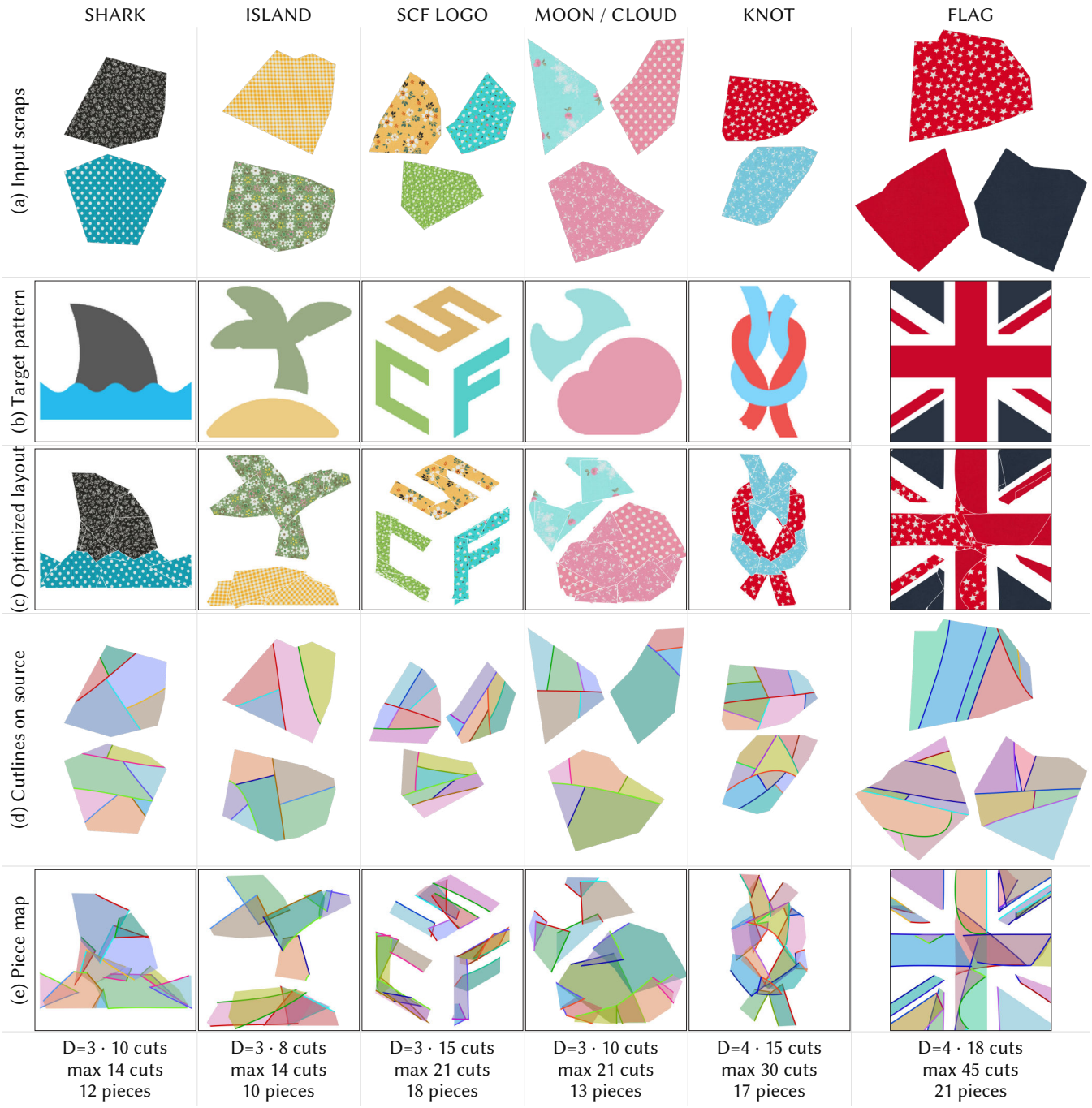


Figure 11: Result gallery of recomposition. Each column shows (a) the input scraps, (b) the target pattern, (c) the optimized layout, (d) cutlines on the source scraps, and (e) the final piece map. For each case, the input scraps and the optimized layout are shown at the same physical scale. Piece colors link source pieces in (d) to their placed counterparts in (e). Colored outlines in (e) mark cut edges and match the cutline colors in (d). Boundary portions without colored outlines come from the original scrap boundaries. Except for Flag, we manually chose each displayed result from 16–32 starts to balance recomposition quality and cut count, rather than automatically selecting the lowest-E_{fit} run. The Flag example is a special case discussed in Section 8. Island and Moon/cloud target artwork © wishforge.games. Flag target by NikNaks, Stefan-Xp, Yaddah, and AnonMoos (public domain). SCF monogram inspired by Asao Tokolo's SCF 2026 logo.

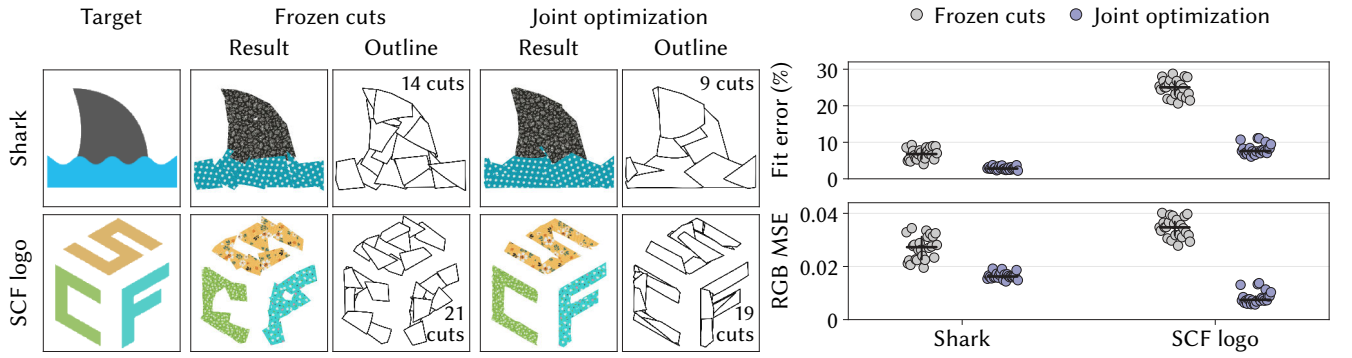


Figure 12: Cutline-optimization ablation. In the "Frozen cuts" variant, we fix the cutline orientation, offset, and bend after initialization, and only optimize piece poses and layer variables. To ensure a fair comparison, we disable the cleanup stage in both variants because frozen cutlines cannot respond to its sparsity objective. The qualitative panels show one paired restart per target: the frozen and joint variants begin from the same initialization. "Result" panels show the final textured recompositions, while adjacent "Outline" panels show the corresponding piece boundaries without textures or piece colors. Labels in the outline panels report final active-cut counts. The plots show shape-fit error (top) and RGB MSE (bottom) evaluated on the final hard layouts for 24 paired optimization restarts per target. Points show individual runs; bars show medians and interquartile ranges. Joint optimization improves both metrics in all 48 pairs.

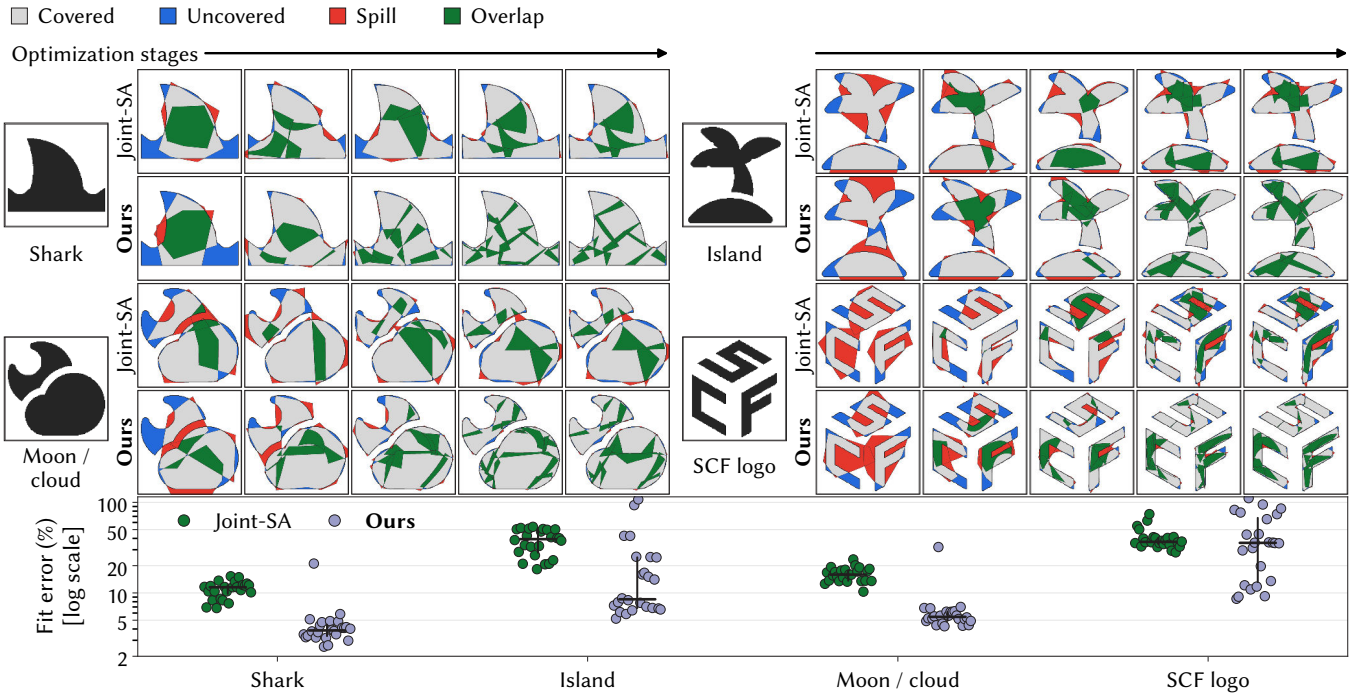


Figure 13: Comparison between Joint-SA and our method on four target patterns. Joint-SA is a variant of ScrapReCover [Kono et al. 2025] adapted for joint cut-and-place optimization. In each case block, the target pattern is shown on the left, with results from Joint-SA and our method shown on the right. In each row, we show the optimization progress from refinement depth $D = 0$ to $D = 3$ and cleanup. At each stage, including cleanup, the columns show our method's final state and Joint-SA's best-so-far state. Gray indicates target regions covered by one piece, blue uncovered target, red spill, and green overlap within the target. Neither method penalizes overlap. It is shown only for visualization. Bottom: shape-fit error over all 24 runs per target. The error is the uncovered target area plus spill of the placed-piece union, normalized by target area. Pieces are not clipped to the target space for this evaluation. Dots show individual runs, vertical black bars span the interquartile range, and horizontal black ticks mark medians.

Comparison with Joint-SA. To our knowledge, no existing method addresses the same problem setting as ours. The closest prior work is ScrapReCover [Kono et al. 2025], which arranges leftover fabric scraps to cover a target pattern and permits overlap. Its placement search uses simulated annealing (SA) [Kirkpatrick et al. 1983]. Because scrap contours remain fixed, the original method has no cutting moves. To compare the shared joint cut-and-place problem, we construct Joint-SA by modifying ScrapReCover's method to include cutting proposals for our representation. These proposals update cutline orientation, offset, and bend or change a cutline between active and inactive states. A coupled proposal can also relocate newly active pieces. Joint-SA is detailed in Appendix E.

The two methods use the same source scraps, source scale, cut-depth schedule, shape-fit term, and paired depth-zero initial states. We omit ScrapReCover's overlap penalty in Joint-SA to match the shape-fitting objective used in this comparison. Across the 24 runs per target shown in Figure 13, our method achieves substantially lower median shape-fit error on 3 of 4 targets: shark, island, and moon/cloud. For the SCF logo, the median errors are comparable (35.71% for our method versus 36.62% for Joint-SA), but our results vary more across starts. We discuss the strengths and limitations of these optimization approaches in Section 8. In additional timing runs for this shape-only comparison, a batch of eight starts took 1.38–2.08 minutes for our method and 14.45–21.09 minutes for Joint-SA across the four targets. Detailed statistics and timing details are reported in Appendix E.

8 Discussion, Limitations, and Future Work

Scrap use and waste. Our method focuses on recomposing a target from a fixed set of source scraps. It does not select a subset of the supplied scraps or remove an otherwise nonempty piece during optimization. With fixed source and target areas and full target coverage, the extra overlapping area plus the placed-piece union area outside the target equals the total source area minus the target area. The gallery's overlap and spill thus reflect the chosen target sizes as well as the optimization. We prioritize matching the target appearance and do not explicitly penalize overlap. An overlap penalty alone can redistribute this excess toward spill without reducing the total material committed to the composition.

However, our formulation does not constrain physical thickness or account for fraying of uncovered cut edges. Excessive local layering may complicate sewing or gluing. A textile without a backing would require durable connections between pieces, with sufficient overlap for permanent fabric glue or seam allowances for sewing. Our optimization does not currently enforce these joining requirements. Our fabrication workflow shares procedural similarities with traditional foundation paper piecing. Future work could incorporate fabrication requirements into the optimization to support sewn assembly, drawing on the geometric and sewing-order constraints formalized by Leake et al. [2021a]. We also do not model recovering excess material from overlapping or protruding regions and returning it to an inventory for reuse.

In our method, a placed piece may extend beyond the target space. Its off-panel portions are clipped from the render and excluded from the rasterized fitting loss used during optimization. We include the Flag result in Figure 11 as an extreme example of this spill

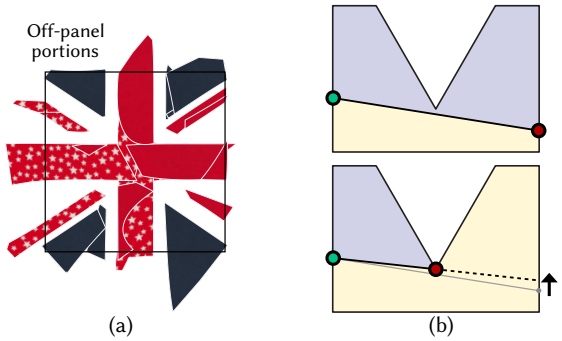


Figure 14: (a) Off-panel material in Flag. (b) A small rotation in a boundary-pivot alternative makes the endpoint and piece shapes jump.

(Figure 14a). Large protrusions may require trimming, whereas slight spill may be acceptable for an approximate match to the target outline. Expanding the target space around an unchanged target pattern would expose more protruding material to the fitting penalty.

We measure extra overlapping material as the sum of piece areas minus their union area, without clipping to the target space. As percentages of the total source fabric area, the six examples in Figure 11 have 20.17% (Shark), 29.99% (Island), 31.46% (SCF logo), 19.62% (Moon/cloud), 20.62% (Knot), and 15.02% (Flag). Off-panel material, summed across pieces, accounts for 20.55% in Flag and less than 0.02% in each remaining example, using the same denominator. Overlap can also occur off-panel, so these percentages are not additive. No piece in these six examples is fully occluded, though the least-visible piece in SCF logo exposes only 1.64% of its area. Complete occlusion remains possible under our formulation. Fully occluded pieces remain in the optimization and assembly guide despite contributing no visible area to the final composition. For the shape-only comparison in Figure 13, Appendix E reports the least- and most-overlapping outcomes across our 24 starts per target, alongside their off-panel material and shape-fit errors. Lower overlap alone does not imply better material reuse.

Jointly optimizing overlap, material selection, and recovery is an important direction for improving material reuse. The workflow could select scraps from a larger inventory, omit fully occluded pieces from assembly, and return unused pieces for reuse. If spill could be trimmed and returned to the inventory for reuse, an overlap penalty could encourage excess material to move from overlapping layers into recoverable spill while maintaining the target appearance and any overlap needed for joining. Recovering this spill would reduce the material retained in each composition and preserve more for subsequent reuse.

Choice of optimization method. Continuous cutline offsets, orientations, and bends, together with piece poses, motivate differentiable rendering. Its strength is joint refinement when pieces occupy suitable target regions but their orientations and cut boundaries need adjustment. Backpropagation through splitting and compositing translates image-space errors into directions for these coupled variables [Baydin et al. 2018], accounting for how one cut affects

several placed descendants. Informative gradients can reduce trial and error in fine adjustment. However, backward computation and storage add cost, and guidance remains local in our non-convex objective.

Joint-SA explores alternative arrangements using the same continuous parameters. Relocating pieces between target components or exchanging placements may cross regions with weak gradients or temporarily increase error. Relocation and swap proposals directly test alternatives, while uphill acceptance permits temporarily worse intermediate states [Kirkpatrick et al. 1983]. These mechanisms support exploration when local refinement stalls. Without derivatives, even fine adjustment can require many evaluations. Both refinement and rearrangement depend on proposal design, move scales, cooling, and budget.

Both methods support additional objectives. Differentiable terms provide gradients through the existing computation, while SA also accommodates non-differentiable terms. Joint-SA shares our soft representation's approximation to hard geometry. Figure 13 and the timings in Section 7 show lower median errors on three tasks and shorter runs for our method under the reported budgets, but greater variation across SCF logo starts. Different regularization schedules (Appendix E) prevent isolating differentiability, and timings do not measure convergence. Future work could explore combining stochastic rearrangements with gradient-based refinement.

Scalability. Our experiments use two or three source scraps. At cut depth D, S scraps have $S2^D$ candidate pieces, including nearly empty children of inactive cuts. At fixed depth and resolution, more scraps therefore add placement variables and increase raster computation and memory use, even if many cuts are inactive in the final plan. The SCF logo results in Figure 13 show sensitivity to initialization already at this small input size. We have not systematically evaluated the effect of larger input sets on fitting quality or convergence.

Source appearance. Even when source and target have the same number of colors, different palettes can leave color errors because cutting and placement cannot recolor the fabric. Our current optimization represents each source scrap by its mean RGB color and therefore assumes that color is approximately uniform within the scrap. The targets considered here use piecewise-uniform color layouts and do not require fine-scale color alignment within a piece. Some scraps may also contain regions with substantially different colors. For gradients or non-repeating motifs, our mean-color objective does not optimize the alignment of these features, so a good mean-color match can still leave visible pattern mismatches. Future work could carry per-pixel source RGB values into the color objective, allowing the method to exploit such scraps. Doing so, however, may introduce competing local alignments and make the optimization more prone to poor local minima.

Choice of cutline parameterization. A boundary-anchored parameterization would more directly represent a physical cut between two boundary points. During early experiments for this work, we considered a boundary-pivot representation in which one endpoint lies on the boundary of the current piece and the other is the first subsequent boundary intersection along the cutline. For a non-convex boundary, however, a small change in the pivot or cutline geometry can make this first intersection switch abruptly between boundary segments. The endpoint and resulting cut can then jump, producing discontinuous piece shapes and renders that can destabilize gradient-based optimization (Figure 14b). We instead define each cutline directly in source space by its orientation, offset, and bend, without selecting an endpoint from boundary intersections. The cutline changes smoothly with these parameters, while the fixed-grid split determines how it divides the current piece. This choice trades direct boundary anchoring for a smoother optimization path. However, as discussed in Section 4.2, it does not itself guarantee that the cutline intersects the current piece in a single continuous boundary-to-boundary segment, as required for a proper cut. This motivates the proper-cut regularizer and an independent check for improper cuts in the final cut geometry.

When a cut is improper, one possible extension would treat disconnected components on the same side of the cutline as separate pieces, each with its own pose. This would avoid forcing these components to share one pose, but it would not eliminate discontinuities. If the cut becomes proper as the cutline moves, components already placed at different target-space locations merge into one source-space piece and must abruptly share a pose. The rendered layout would therefore jump.

This component-merging event and the boundary-intersection switch in the boundary-pivot alternative expose the same broader difficulty: when a cutline acts on a non-convex piece, changes in boundary intersections and the resulting piece topology are difficult to represent without jumps. A restrictive workaround would be to decompose each input scrap into convex pieces before optimization and permit only straight cuts thereafter, thereby preserving convexity and avoiding leave-and-re-enter intersections. Rather than impose these restrictions, future work could seek a cutline representation that retains a smooth optimization path and direct boundary anchoring while ideally guaranteeing proper cuts by construction.

All cutlines of a source share a fixed domain, including those applied to small descendant pieces. With bounded bend values, this can limit the relative bending of the short segments used by those pieces, yielding near-straight local edges even when the full cutline remains curved. Future work could explore cutline domains centered on each current piece, with domain sizes and bend bounds scaled to its size. The ablation in Section 7 shows that retaining curved cutlines can nevertheless improve fitting, with more geometric-check failures.

9 Conclusion

We presented ScrapCuts, which treats cutting and placement as a single coupled problem in fabric scrap recomposition: rather than asking only how a fixed set of scraps should be arranged, we also ask how they should be cut. By representing scraps as occupancy fields and cutlines as smooth curves defined by orientation, offset, and bend, we make piece shape a continuous variable. This lets us optimize cut geometry, piece poses, and layer order jointly under one rendered-assembly objective. Because each cut is a manual fabrication step, we reduce their number in two ways: candidate cutlines may become inactive during optimization, and a final cut-sparsity cleanup removes further cuts that do not improve visual quality. Across our results, only 8–18 of the 14–45 candidate cutlines

remain active in the final layouts. Joint optimization improved shape-fit error and RGB MSE over frozen cuts in all 48 paired runs and achieved substantially lower median fit error than a simulated-annealing baseline on three out of four target patterns, though with greater variance across restarts. Fabricated examples confirm the layouts can be physically realized from the generated cut plans and assembly guide.

Acknowledgments

We thank Mikhail Bessmeltsev for his valuable suggestions on this project. We are grateful to the anonymous reviewers for their constructive feedback. This work was supported in part by the Japan Science and Technology Agency (JST) SPRING grant number JPMJSP2108, JST ASPIRE grant number JPMJAP2401, and the Japan Society for the Promotion of Science (JSPS) Grant-in-Aid grant number JP23K16921.

References

Berend Baas, David Bommes, and Adrien Bousseau. 2025. Shape Approximation by Surface Reuse. *Computer Graphics Forum* 44, 5, Article e70204 (Aug. 2025), 11 pages. doi:10.1111/cgf.70204

Atılım Güneş Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2018. Automatic Differentiation in Machine Learning: a Survey. *Journal of Machine Learning Research* 18, 153 (2018), 1–43. https://www.jmlr.org/papers/v18/17-468.html

Julia A. Bennell and Jose F. Oliveira. 2008. The Geometry of Nesting Problems: A Tutorial. *European Journal of Operational Research* 184, 2 (Jan. 2008), 397–415. doi:10.1016/j.ejor.2006.11.038

Xuelin Chen, Hao Zhang, Jinjie Lin, Ruizhen Hu, Lin Lu, Qixing Huang, Bedrich Benes, Daniel Cohen-Or, and Baoquan Chen. 2015. Dapper: Decompose-and-Pack for 3D Printing. *ACM Transactions on Graphics* 34, 6, Article 213 (Nov. 2015), 12 pages. doi:10.1145/2816795.2818087

Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems*, Vol. 26. Curran Associates, Inc., Red Hook, NY, USA, 2292–2300. https://proceedings.neurips.cc/paper_files/paper/2013/hash/af21d0c97db2e27e13572cbf59eb343d-Abstract.html

Ashley Del Valle, Jennifer Jacobs, and Emilie Yu. 2025. texTile: Making and Re-making Crochet Granny Square Garments Through Computational Design and 3D-printed Connectors. In *Proceedings of the 2025 ACM Designing Interactive Systems Conference (DIS '25)*. Association for Computing Machinery, New York, NY, USA, 2445–2464. doi:10.1145/3715336.3735819

Noah Duncan, Lap-Fai Yu, Sai-Kit Yeung, and Demetri Terzopoulos. 2017. Approximate Dissections. *ACM Transactions on Graphics* 36, 6, Article 182 (Nov. 2017), 13 pages. doi:10.1145/3130800.3130831

Seongmin Hong, Junghun James Kim, Daehyeop Kim, Insoo Chung, and Se Young Chun. 2026. DiffBMP: Differentiable Rendering with Bitmap Primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Denver, CO, USA, 26741–26750. https://openaccess.thecvf.com/content/CVPR2026/html/Hong_DiffBMP_Differentiable_Rendering_with_Bitmap_Primitives_CVPR_2026_paper.html

Hua Huang, Lei Zhang, and Hong-Chao Zhang. 2011. Arcimboldo-like Collage Using Internet Images. *ACM Transactions on Graphics* 30, 6, Article 155 (Dec. 2011), 8 pages. doi:10.1145/2070781.2024189

Yuki Igarashi and Jun Mitani. 2015. Patchy: An Interactive Patchwork Design System. In *ACM SIGGRAPH 2015 Posters (SIGGRAPH '15)*. Association for Computing Machinery, New York, NY, USA, Article 10, 1 pages. doi:10.1145/2787626.2792601

Julian Iseringhausen, Michael Weinmann, Weizhen Huang, and Matthias B. Hullin. 2020. Computational Parquetry: Fabricated Style Transfer with Wood Pixels. *ACM Transactions on Graphics* 39, 2, Article 12 (Feb. 2020), 14 pages. doi:10.1145/3378541

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. 2015. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, Vol. 28. Curran Associates, Inc., Red Hook, NY, USA, 2017–2025. https://proceedings.neurips.cc/paper/2015/hash/33ceb07bf4eeb3da587e268d663aba1a-Abstract.html

Chiyu Jiang, Dana Lansigan, Philip Marcus, and Matthias Nießner. 2019. DDSL: Deep Differentiable Simplex Layer for Learning Geometric Signals. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Seoul, Republic of Korea, 8768–8777. doi:10.1109/ICCV.2019.00886

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations (ICLR)*. San Diego, CA, USA. doi:10.48550/arXiv.1412.6980

Scott Kirkpatrick, C. Daniel Gelatt, Jr., and Mario P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220, 4598 (1983), 671–680. doi:10.1126/science.220.4598.671

Masahiro Kono, Maria Larsson, I-Chao Shen, and Takeo Igarashi. 2025. ScrapReCover: An Interactive Optimization System for Freeform Patchwork Layouts. In *Proceedings of the ACM Symposium on Computational Fabrication (SCF '25)*. Association for Computing Machinery, New York, NY, USA, Article 20, 15 pages. doi:10.1145/3745778.3766653

Bongjin Koo, Jean Hergel, Sylvain Lefebvre, and Niloy J. Mitra. 2017. Towards Zero-Waste Furniture Design. *IEEE Transactions on Visualization and Computer Graphics* 23, 12 (Dec. 2017), 2627–2640. doi:10.1109/TVCG.2016.2633519

Mackenzie Leake, Gilbert Bernstein, and Maneesh Agrawala. 2022. Sketch-Based Design of Foundation Paper Pieceable Quilts. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 21, 11 pages. doi:10.1145/3526113.3545643

Mackenzie Leake, Gilbert Bernstein, Abe Davis, and Maneesh Agrawala. 2021a. A Mathematical Foundation for Foundation Paper Pieceable Quilts. *ACM Transactions on Graphics* 40, 4, Article 65 (Aug. 2021), 14 pages. doi:10.1145/3450626.3459853

Mackenzie Leake and Ross Daly. 2024. ScrapMap: Interactive Color Layout for Scrap Quilting. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 61, 17 pages. doi:10.1145/3654777.3676404

Mackenzie Leake, Frances Lai, Tovi Grossman, Daniel Wigdor, and Ben Lafreniere. 2021b. PatchProv: Supporting Improvisational Design Practices for Modern Quilting. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 500, 17 pages. doi:10.1145/3411764.3445601

Ganghun Lee, Minji Kim, Yunsu Lee, Minsu Lee, and Byoung-Tak Zhang. 2023. Neural Collage Transfer: Artistic Reconstruction via Material Manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Paris, France, 2394–2405. doi:10.1109/ICCV51070.2023.00227

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. 2020. Differentiable Vector Graphics Rasterization for Editing and Learning. *ACM Transactions on Graphics* 39, 6, Article 193 (Dec. 2020), 15 pages. doi:10.1145/3414685.3417871

Max Limper, Nicholas Vining, and Alla Sheffer. 2018. Box Cutter: Atlas Refinement for Efficient Packing via Void Elimination. *ACM Transactions on Graphics* 37, 4, Article 153 (Aug. 2018), 13 pages. doi:10.1145/3197517.3201328

Rebecca Lin, Michal Lukáč, and Mackenzie Leake. 2025. Refashion: Reconfigurable Garments via Modular Design. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25)*. Association for Computing Machinery, New York, NY, USA, 1–18. doi:10.1145/3746059.3747632

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Seoul, Republic of Korea, 7708–7717. doi:10.1109/ICCV.2019.00780

Yuxuan Mei, Lirong Que, Mackenzie Leake, and Adriana Schulz. 2025. PatchUp: Interactive Patchwork Design for Scrap Fabric Upcycling. In *Proceedings of the ACM Symposium on Computational Fabrication (SCF '25)*. Association for Computing Machinery, New York, NY, USA, Article 22, 17 pages. doi:10.1145/3745778.3766667

Piotr Mirowski, Dylan Banarse, Mateusz Malinowski, Simon Osindero, and Chrisantha Fernando. 2022. CLIP-CLOP: CLIP-Guided Collage and Photomontage. In *Proceedings of the 13th International Conference on Computational Creativity*, Maria M. Hedblom, Anna Aurora Kantosalo, Roberto Confalonieri, Oliver Kutz, and Tony Veale (Eds.). Association for Computational Creativity, Bozen-Bolzano, Italy, 190–194. https://computationalcreativity.net/iccc22/papers/ICCC-2022_paper_104.pdf

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc., Red Hook, NY, USA, 8026–8037. https://proceedings.neurips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Anran Qi, Nico Pietroni, Mikhail Bessmeltsev, Nobuyuki Umetani, Adrien Bousseau, and Takeo Igarashi. 2026. Voronoi Scissors: Approximate Dissection of 2D Shapes with Continuous Optimization. In *SIGGRAPH Asia 2026 Conference Papers* (Kuala Lumpur, Malaysia, December 1–4, 2026) *(SA Conference Papers '26)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3829340.3842173

Anran Qi, Nico Pietroni, Maria Korosteleva, Olga Sorkine-Hornung, and Adrien Bousseau. 2025. Rags2Riches: Computational Garment Reuse. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 45, 11 pages. doi:10.1145/3721238.3730703

Pradyumna Reddy, Paul Guerrero, Matt Fisher, Wilmot Li, and Niloy J. Mitra. 2020. Discovering Pattern Structure Using Differentiable Compositing. *ACM Transactions on Graphics* 39, 6, Article 262 (Dec. 2020), 15 pages. doi:10.1145/3414685.3417830

Daniel Saakes, Thomas Cambazard, Jun Mitani, and Takeo Igarashi. 2013. PacCAM: Material Capture and Interactive 2D Packing for Efficient Material Usage on CNC Cutting Machines. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. Association for Computing Machinery, New York, NY, USA, 441–446. doi:10.1145/2501988.2501990

Ticha Sethapakdi, Daniel Anderson, Adrian Reginald Chua Sy, and Stefanie Mueller. 2021. Fabricaide: Fabrication-Aware Design for 2D Cutting Machines. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 664, 12 pages. doi:10.1145/3411764.3445345

Zefan Shao, Jin Zhou, Hongliang Yang, and Pengfei Xu. 2026. SCORE: Semantic Collage by Optimizing Rendered Elements. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 3 (March 2026), 2038–2046. doi:10.1609/aaai.v40i3.37185

Musashi Shinjo, Maria Larsson, and Hironori Yoshida. 2024. A Design and Fabrication Workflow for Upcycling Leftover Fabrics as Mosaic Art. In *Proceedings of the 15th International Conference on Computational Creativity*, Kazjon Grace, Maria Teresa Llano, Pedro Martins, and Maria M. Hedblom (Eds.). Association for Computational Creativity, Jönköping, Sweden, 354–358. https://computationalcreativity.net/iccc24/papers/ICCC24_paper_153.pdf

Zhenyu Wang and Min Lu. 2025. Image-Space Collage and Packing with Differentiable Rendering. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 172, 11 pages. doi:10.1145/3721238.3730690

Ruowang Zhang, Stefanie Mueller, Gilbert Louis Bernstein, Adriana Schulz, and Mackenzie Leake. 2024. WasteBanned: Supporting Zero Waste Fashion Design Through Linked Edits. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 110, 13 pages. doi:10.1145/3654777.3676395

A Detailed Cutline Parameterization and Occupancy Field Construction

For $\mathbf{x} = (x, y) \in \Omega_s$, the graph coordinates at angle θ are

$$u_\theta(\mathbf{x}) = x\cos\theta + y\sin\theta, \qquad v_\theta(\mathbf{x}) = -x\sin\theta + y\cos\theta. \tag{15}$$

Here, θ specifies the graph's u-axis direction. The orthogonal v-axis is the normal direction of the corresponding straight cut. The cutline at node n uses the paper-facing offset–bend convention $v = d_{s,n} + C_{s,n}(t)$.

The graph domain $[-h_s, h_s]$ conservatively covers the source region for every orientation. On an $H_s \times H_s$ source raster whose coordinates span $[-1, 1]^2$, the implementation uses

$$h_s = \max_{\mathbf{x}:O_s(\mathbf{x})>0.5} \|\mathbf{x}\|_2 + 2\Delta_s, \qquad \Delta_s = \frac{2}{H_s - 1}, \tag{16}$$

The two-pixel margin keeps the curve domain beyond the source region. Since $|u_\theta(\mathbf{x})| \le \|\mathbf{x}\|_2$, the same radial bound covers every orientation.

The normalized parameter $t(u) = \frac{1}{2}(u/h_s + 1)$ maps $[-h_s, h_s]$ to $[0, 1]$. The centered cubic bend polynomial is

$$C_{s,n}(t) = \sum_{k=0}^{3} B_k(t) b_{s,n,k}, \tag{17}$$

using the standard cubic Bernstein basis

$$\begin{aligned} B_0(t) &= (1-t)^3, \qquad B_1(t) = 3(1-t)^2 t, \\ B_2(t) &= 3(1-t)t^2, \quad B_3(t) = t^3. \end{aligned} \tag{18}$$

The bend coefficients are centered to separate shape variation from the common offset, enforcing $\sum_{k=0}^{3} b_{s,n,k} = 0$. We optimize $d_{s,n}$ and $b_{s,n,k}$ within the ranges stated in Section 4.1. Setting all bend coefficients to zero gives the exact straight-cut case.

For $\mathbf{x} \in \Omega_s$, let $t_{s,n}(\mathbf{x}) = \frac{1}{2}(u_{\theta_{s,n}}(\mathbf{x})/h_s + 1)$. The soft side-assignment field is

$$w_{s,n}(\mathbf{x}) = \sigma\left(\frac{v_{\theta_{s,n}}(\mathbf{x}) - \left[d_{s,n} + C_{s,n}(t_{s,n}(\mathbf{x}))\right]}{\tau_c}\right). \tag{19}$$

For a current piece represented by O_s, the soft split produces the two occupancy fields $O_s w_{s,n}$ and $O_s(1 - w_{s,n})$.

B Proper-Cut Gap Measure and Schedule

For candidate cutline n, the field $w_{s,n}(1 - w_{s,n})$ forms a soft band around the curve. We divide its u-domain into intervals and let $a_{n,j} \in [0, 1]$ denote the mean occupancy of the current piece within the band over interval j. The values $a^L_{n,j}$ and $a^R_{n,j}$ indicate whether the band passes through the piece before and after that interval. Their product with $1 - a_{n,j}$ therefore identifies a gap between separated in-piece intervals.

The response should vanish when a cut becomes inactive. Let $s_n \in [0, 1]$ be high when neither resulting occupancy field is nearly empty. When s_n is low, a re-entry response $r_n \in [0, 1]$ retains the penalty only when the cutline clearly enters the piece again. We define

$$e_n = \mathrm{mean}_j \left[(s_n + (1 - s_n)r_n)\, a^L_{n,j} \left(1 - a_{n,j}\right) a^R_{n,j}\right]. \tag{20}$$

Both s_n and r_n are computed from the occupancy fields involved in the current split and held fixed when differentiating e_n.

A discrete geometric check runs every five optimization steps and updates the multiplier $\gamma_n^{(k)}$ in Equation 13. When a cut previously observed as proper or inactive becomes improper, its multiplier remains zero for 150 steps. If the improper state persists, the multiplier rises from zero to one over 50 steps with a half-cosine ramp. A cut without a proper or inactive history starts the ramp immediately. Five consecutive gap-free checks reset its schedule. We use $\lambda_{\mathrm{pc}} = 0.02$. The check changes only $\gamma_n^{(k)}$. All gradients come from e_n.

Controlled validation. Two controlled tests probe the schedule rather than complete recompositions. Across 192 paired active–inactive–active trajectories per condition, all 192 reached the inactive endpoint with no E_{pc}, a fixed weight of 0.02 from the first step, or scheduled weighting. Under geometric and occupancy-field criteria, the unregularized and scheduled conditions then returned to the active reference in 192/192 trajectories, versus 66/192 with fixed weighting. In 96 paired persistent-improper starts optimized only with E_{pc}, the unregularized control reached 0/96 proper endpoints. Fixed and scheduled weighting reached 90/96 and 92/96, respectively, and none became inactive. The 90/92 difference does not establish an advantage of scheduling. These controlled tests neither guarantee proper endpoints nor establish fewer improper outputs in complete recompositions.

C Layer-Variable Separation

To resolve near-ties in the soft layer-order relaxation, we apply an auxiliary z-only separation penalty late in each core stage. This numerical stabilization is not one of the three task-level regularizers

collected in Equation 12. We use the overlap-weighted hinge

$$\sum_{i \neq j} \bar{o}_{ij} \left[\max\left(0, m_z - |z_i - z_j|\right)\right]^2 .$$

Here, $\bar{o}_{ij}$ is the overlap $\sum_{\mathbf{y}} \alpha_i(\mathbf{y})\alpha_j(\mathbf{y})$, normalized over all ordered pairs $i \neq j$ and detached before differentiation. Thus, this penalty affects only the layer variables, not piece poses or cutline parameters.

Let k/K be the progress through a core stage, η_z the start fraction, and r_z the ramp fraction. With $h(a) = a^2(3 - 2a)$ and $a = \mathrm{clamp}((k/K - \eta_z)/r_z, 0, 1)$, the coefficient of this penalty is $\lambda_{\mathrm{zsep}} h(a)$. The reported color-fitting runs use $m_z = 0.1$, $\lambda_{\mathrm{zsep}} = 20$, $\eta_z = 0.6$, and $r_z = 0.25$.

D Final Cut Cleanup

The cleanup stage continues from the terminal maximum-depth state. For each candidate cut n of source scrap s, we compute the side fraction $q_{s,n}$ in Equation 14 from the occupancy field O_s of its current piece and its soft side-assignment field $w_{s,n}$:

$$q_{s,n} = \frac{\sum_{\mathbf{x}} O_s(\mathbf{x}) w_{s,n}(\mathbf{x})}{\sum_{\mathbf{x}} O_s(\mathbf{x}) + \epsilon}. \tag{21}$$

We detach O_s, which routes this term only to the current cutline geometry. Shape-fit and color gradients continue to update cutlines, piece poses, and layer variables.

The nominal sparsity weight follows a half-cosine ramp,

$$\lambda_{\mathrm{sparse}}^{(k)} = \frac{\bar{\lambda}_{\mathrm{sparse}}}{2} \left[1 - \cos\left(\pi \min\left(1, \frac{k}{r_s K}\right)\right)\right]. \tag{22}$$

For each start in color-fitting runs, we cap the effective weight so the sparsity contribution does not exceed a fraction ρ_{rgb} of the weighted direct-color objective:

$$\widehat{\lambda}_{\mathrm{sparse}}^{(k)} = \min\left(\lambda_{\mathrm{sparse}}^{(k)}, \frac{\rho_{\mathrm{rgb}} \lambda_{\mathrm{color}} E_{\mathrm{color}}}{\max(E_{\mathrm{sparse}}, \epsilon)}\right). \tag{23}$$

We use $\rho_{\mathrm{rgb}} = 1$; shape-only runs omit this cap. The cap and the occupancy fields of the current pieces are detached. Consequently, the sparsity term has no gradient path through the direct-color objective or an ancestor cut.

For Shark and our shape-only runs, we use the basic soft count with $K = 1000$ cleanup steps, $\tau_{\mathrm{sparse}} = 0.02$, $\bar{\lambda}_{\mathrm{sparse}} = 0.01$, and $r_s = 0.6$.

Other gallery results use a reweighted soft count, with per-cut weights favoring small nonzero splits. The weights are updated each iteration and held fixed during differentiation. These runs also cap sparsity by the shape-fitting loss.

Color guidance is disabled during cleanup. In color fitting, the temperature τ_z starts at one half of its core-stage initial value and is geometrically annealed to the same final temperature.

E Joint-SA Comparison

Joint-SA is our comparison method based on ScrapReCover [Kono et al. 2025]. We modify its optimization for our joint cut-and-place problem with a fixed set of source scraps. It is not the original ScrapReCover method. Joint-SA searches over the same cutline parameters and piece poses as our method.

Because the set of source scraps is fixed, Joint-SA omits ScrapReCover's add and erase moves. Table 3 summarizes the placement and cutline proposals used instead. At each simulated-annealing step [Kirkpatrick et al. 1983], Joint-SA samples one proposal and applies the Metropolis acceptance rule. The initial temperature is estimated from sampled uphill objective changes and then decreases geometrically.

Table 3: Joint-SA proposal families and their relative weights.

Proposal and weight	Effect
Translate / rotate (0.20/0.12)	Update one piece pose
Swap / relocate (0.05/0.06)	Exchange two poses or move one piece toward uncovered target
Orientation / offset / bend (0.10/0.13/0.12)	Perturb one cutline parameter group
Combined cutline update (0.12)	Update orientation, offset, and bend together
Activate / deactivate (0.10)	Change cutline activity and optionally relocate newly active pieces
Cutline reinitialization (0.06)	Resample one cutline, all deeper cutlines in its resulting pieces, and their piece poses

For this comparison, both methods use the shape-fitting term E_{shape} in Equation 11. RGB MSE, color guidance, and layer order are not used. Both methods use the same source scraps, source scale, cutline representation, cut-depth schedule, and paired depth-zero states. At later depths, the same expansion procedure and paired random streams are applied to each method's own state.

Our method schedules proper-cut regularization during each stage, whereas Joint-SA uses a fixed weight for its proper-cut regularizer. The comparison therefore evaluates the two complete optimization procedures rather than differentiability alone. Joint-SA uses 32,000 steps per stage. Preliminary sweeps tested seven budgets from 500 to 32,000 steps per stage, with eight chains for each of the four targets. We used the largest tested budget. After the maximum-depth stage, it continues from the terminal state for 6,400 cleanup steps using the same proposals. The cut-sparsity weight follows a half-cosine ramp from zero to 0.01 over the first 60% of these steps and then remains fixed. Our method uses 1,000 Adam steps per stage and 1,000 cleanup steps. Across depths $D = 0$ through $D = 3$ and cleanup, these budgets total 134,400 SA steps and 5,000 Adam steps per start, respectively.

For each target, we run 24 paired starts for each method. The reported shape-fit error is the uncovered target area plus spill of the placed-piece union, normalized by target area. Pieces are evaluated without clipping to the target space.

Table 4: Shape-fit error over 24 runs per target and method. Values are median [interquartile range], in percent.

Target	Joint-SA	Ours
Shark	11.56 [9.77, 12.18]	3.86 [3.34, 4.47]
Island	39.26 [27.79, 49.93]	8.51 [6.80, 24.73]
Moon/cloud	15.94 [13.63, 17.75]	5.44 [4.90, 6.19]
SCF logo	36.62 [34.50, 40.67]	35.71 [13.19, 67.43]

Table 5: Least- and most-overlapping ScrapCuts outcomes among all 24 starts per target in Figure 13. Overlap and off-panel material are percentages of source area. Fit error is a percentage of target area.

Target	Extreme	Overlap	Off-panel	Fit error
Shark	Least†	7.46	13.50	21.10
	Most	21.00	0.00	4.84
Island	Least	21.55	7.57	25.09
	Most	58.28	0.00	42.61
Moon/cloud	Least	19.29	0.21	6.18
	Most‡	43.04	0.00	32.12
SCF logo	Least†	6.65	36.39	85.48
	Most	52.79	0.00	36.17

† Contains an entirely off-panel piece. ‡ Contains an improper cut.

Runtime. We timed one additional batch of eight starts per target and method using the comparison settings above, on an Intel Core i9-13900KF CPU and an NVIDIA GeForce RTX 4080 GPU. Both methods used a single CPU thread and batched GPU evaluations. Joint-SA is our Python adaptation and uses the same cut representation and raster evaluation backend as our method. CUDA-synchronized wall-clock times include initialization, all depth stages, and cleanup, including Joint-SA's temperature calibration and incumbent validity checks. Shared input loading, result export, and visualization are excluded. Table 6 reports the complete optimization times under these iteration budgets.

Table 6: Wall-clock time in minutes for one batch of eight starts per target and method in the shape-only comparison.

Target	Joint-SA	Ours
Shark	14.45	1.38
Island	14.84	1.39
Moon/cloud	18.80	2.06
SCF logo	21.09	2.08

F Cutline Design Comparisons

We compare straight and cubic cutlines on the four shape-fitting tasks in Figure 13, using the same source scraps, source scale, and paired optimized depth-zero states. The straight condition fixes the bend values to zero while continuing to optimize cutline orientations, offsets, and piece poses. The cubic condition also optimizes the bend values. Both conditions follow progressive optimization through maximum depth $D = 3$, with 1,000 updates per core stage and 1,000 final cleanup updates.

Table 7 summarizes all 24 starts per target and condition, including outcomes that fail the computational checks. Shape-fit error is the uncovered target area plus spill of the placed-piece union, normalized by target area and evaluated without clipping to the target space. Active cuts count nonempty splits in the cut trees. Geometric checks use numerical area and length thresholds to test for proper cuts and connected, hole-free pieces.

The median paired reduction in shape-fit error, computed as straight minus cubic error within each pair, ranged from 0.92 to 3.59 percentage points across tasks. Cubic cutlines achieved lower error in 15–22 of the 24 pairs per task. These results support retaining curved cutlines to improve fitting under the tested settings. All 96 straight outcomes passed the geometric checks, while 8 of the 96 cubic outcomes failed. A further 9 straight and 14 cubic outcomes failed the panel-presence check because at least one piece had insufficient intersection with the panel.

Table 7: Straight versus cubic cutlines at maximum depth $D = 3$. Errors are medians [interquartile ranges] over all 24 starts per condition. Active-cut counts are medians over the same starts.

Target	Cutlines	Fit error (%)	Active cuts
Shark	Straight	4.84 [4.27, 5.14]	7
	Cubic	3.78 [3.36, 4.16]	7
Island	Straight	11.70 [9.23, 20.79]	10.5
	Cubic	9.37 [7.40, 36.38]	10
Moon/cloud	Straight	6.88 [6.32, 7.89]	8
	Cubic	5.66 [5.22, 6.59]	9
SCF logo	Straight	38.92 [22.96, 53.40]	15
	Cubic	35.95 [15.74, 42.53]	17

Maximum cut depth. We compare cubic cutlines at maximum depths $D = 2, 3, 4$ using the same 24 paired starts per target. Each setting uses 1,000 updates per core stage and a separate 1,000-update cleanup, so the total update budget increases with depth. The $D = 3$ results are those in Table 7. These depth limits allow 6, 14, and 30 candidate cuts for two source scraps, or 9, 21, and 45 for three. Optimization and cleanup determine which cuts remain active. Table 8 includes all outcomes. Timings use the hardware in Appendix E and cover all optimization stages, cleanup, and stage-checkpoint saving.

Table 8: Maximum-depth comparison with cubic cutlines. Shape-fit errors are medians [interquartile ranges] and active-cut counts are medians over all 24 starts per condition. Batch time is the median wall-clock duration of three batches of eight starts, in seconds.

Target	D	Fit error (%)	Active cuts	Batch time (s)
Shark	2	5.02 [4.75, 5.74]	4.5	41.9
	3	3.78 [3.36, 4.16]	7	83.3
	4	3.28 [3.01, 3.59]	8	196.6
Island	2	35.87 [21.86, 51.22]	6	41.2
	3	9.37 [7.40, 36.38]	10	82.9
	4	6.55 [5.89, 16.62]	12	194.9
Moon/cloud	2	9.47 [6.86, 12.96]	6	57.2
	3	5.66 [5.22, 6.59]	9	124.3
	4	5.22 [4.78, 5.48]	10	318.5
SCF logo	2	42.55 [34.19, 64.11]	8	57.2
	3	35.95 [15.74, 42.53]	17	121.9
	4	35.15 [7.97, 36.95]	19	318.3

Median paired reductions in error were 1.43–17.32 percentage points from depth 2 to 3 and 0.30–2.13 from depth 3 to 4 across tasks. Geometric checks failed for 5, 8, and 10 of the 96 outcomes at depths 2, 3, and 4, respectively. Panel-presence failures numbered 17, 14, and 7, with one depth-4 outcome failing both checks.

An additional comparison used eight paired starts per task and 5,000 updates in both conditions. Extending the final depth-2 core stage did not close the fitting gap, as depth 3 achieved lower error in all pairs. The extension also stretched the learning-rate schedule, so this controls update count without isolating the effect of candidate capacity alone.

G Input Asset Credits

All source-scrap images used in our experiments are scans captured by the authors. The patterned fabrics were purchased from seller ZHONGL on Amazon.co.jp (brand: DFsucces; ASIN: B0BHHVBPTY). Source credited as requested in the seller's reply. The underlying fabric patterns are excluded from the article's Creative Commons license. The target graphics were prepared as follows. The SVG Repo license labels below were checked on September 29, 2026.

- **Shark.** Adapted from *Animal Domestic Pet 14* by Darius Dan, via SVG Repo, listed under the CC Attribution License; black outlines removed and graphic rasterized for use as an input target.
- **Island.** Adapted from *Holidays Vacation Summer Palmtree Island* by wishforge.games, via SVG Repo, listed under the CC Attribution License; recolored and rasterized for use as an input target.
- **SCF monogram.** Drawn by the authors, inspired by the letter arrangement of Asao Tokolo's SCF 2026 logo.
- **Moon/cloud.** Adapted from *Forecast Weather Night Cloud Moon* by wishforge.games, via SVG Repo, listed under the CC Attribution License; recolored and rasterized for use as an input target.
- **Knot.** Adapted from *Knot* by Google Noto Emoji, via SVG Repo, under the Apache License 2.0; colors and shading simplified by the authors, then rasterized at the experimental target resolution.
- **Flag.** Adapted from *Flag of the United Kingdom Square*, a derivative by NikNaks from work by Stefan-Xp and Yaddah, with edits by AnonMoos (Wikimedia Commons, public domain); recolored and rasterized.