LayerD: Decomposing Raster Graphic Designs into Layers

Tomoyuki Suzuki[1] Kang-Jun Liu[2] Naoto Inoue[1] Kota Yamaguchi[1]
[1]CyberAgent [2]Tohoku University

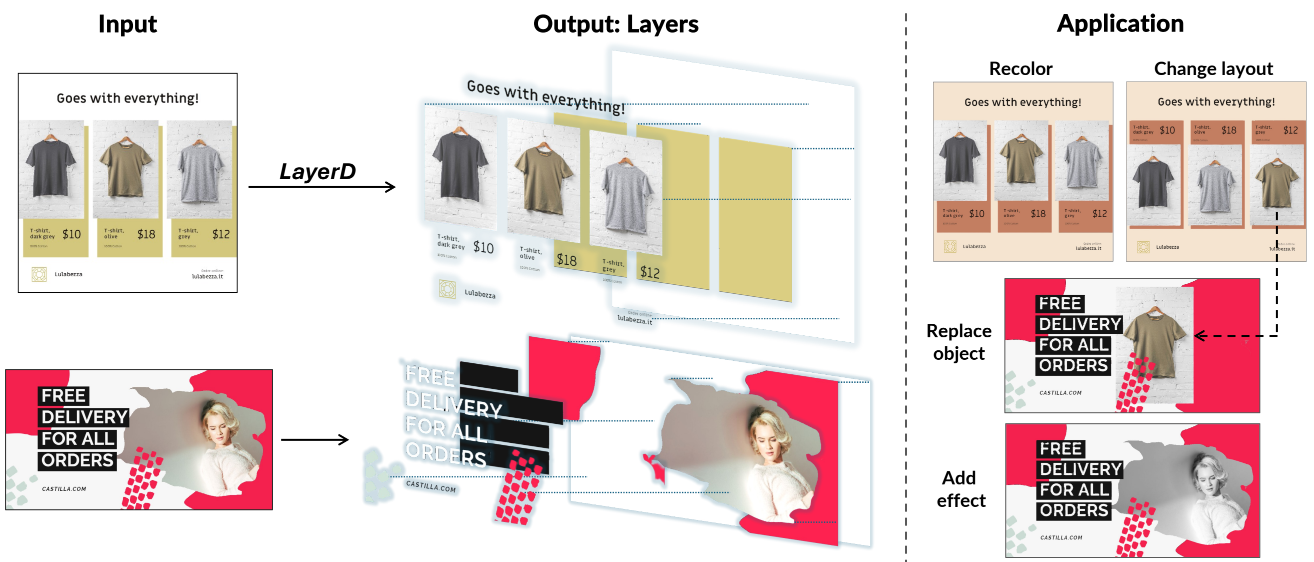


Figure 1. LayerD effectively decomposes raster graphic design images into layers, where the input design contains various elements such as typographic entities, embellishments, vector shapes, or even image materials. Once decomposed, one can apply image editing operations such as color conversion or translation at the layer level, or further apply other post-processing such as OCR to vectorize each raster layer.

Abstract

Designers craft and edit graphic designs in a layer representation, but layer-based editing becomes impossible once composited into a raster image. In this work, we propose LayerD, a method to decompose raster graphic designs into layers for re-editable creative workflow. LayerD addresses the decomposition task by iteratively extracting unoccluded foreground layers. We propose a simple yet effective refinement approach taking advantage of the assumption that layers often exhibit uniform appearance in graphic designs. As decomposition is ill-posed and the ground-truth layer structure may not be reliable, we develop a quality metric that addresses the difficulty. In experiments, we show that LayerD successfully achieves high-quality decomposition and outperforms baselines. We also demonstrate the use of LayerD with state-of-the-art image generators and layer-based editing. Code and models are publicly available [1].

[1]https://github.com/CyberAgentAILab/LayerD

1. Introduction

In the creative workflow, designers create and edit graphic designs at the *layer* level, which is a basic unit of visual objects, such as text or images, and is commonly seen in design authoring tools like Adobe Photoshop or PowerPoint. Once the workflow is complete, authoring tools composite these layers into a final image and deliver it to a display device or print media, such as social media posts, flyers, and posters. Composite raster images do not retain layer information, making it difficult for designers to edit or retouch a raster graphic design. Precise decomposition of raster artwork into layers, *i.e.*, the inverse problem of composition, addresses this situation and enables a workflow that uses existing raster artwork assets to create new artwork.

In this work, we investigate graphic layer decomposition, aiming to automatically decompose a raster graphic design into a composable sequence of raster layers. Since designers create graphic designs in a layered format, we can view this task as restoring the original layered representation. Layer decomposition involves several computer vision tasks, such as object localization, segmentation, order estimation, and image inpainting. Unlike natural images, graphic design is a mixture of various elements, including typography, embellishments, vector art, illustrations, and even natural image materials (Fig. 1). Naively applying image decomposition approaches [51, 59, 61] tuned for the natural image domain results in unintended decomposition (*e.g*., objects in a photo material are decomposed) or undesirable artifacts (*e.g*., background lighting affects solid-color vector-art), which are prohibitive for creative work. Graphic layer decomposition is also inherently ill-posed; there are multiple possible solutions, and a layer can be arbitrarily divided into multiple layers. This can be problematic, particularly when ensuring consistent evaluation.

We propose a method for fully automatic graphic layer decomposition, **LayerD**, which we formulate as iterative *top-layer* matting and background completion. We define a top-layer by objects appearing on the front without occlusion in the raster image, and in graphic designs, they typically contain typography at the beginning, followed by embellishment behind texts or photo materials in later iterations. We learn a top-layer matting model from a high-quality graphic design dataset to ensure that the layer granularity aligns with humans, and together with an off-the-shelf inpainting model and a simple-yet-effective heuristic refinement to remove artifacts, we build a complete layer decomposition pipeline for graphic designs. There have been a few similar attempts at fully automatic image decomposition into layer representations [5, 51], where they build a modular decomposition pipeline consisting of components for each subproblem, such as object detection [27, 57], segmentation [38], ordering [22, 37], and inpainting [41]. While the stacked pipeline approach can take advantage of pre-trained models at each stage, component stacking cannot avoid error accumulation throughout the pipeline; *e.g*., segmentation can fail when object detection contains overlapping bounding boxes or there is a large hall in the region. LayerD unifies detection, segmentation, and layer ordering by an iterative matting model to reduce the error accumulation while improving the efficiency. In addition, we introduce a refinement approach for both foreground and background layers utilizing the domain prior that graphic design often consists of texture-less flat regions, which improves the final decomposition quality.

As layer decomposition can have multiple solutions and even humans are not consistent on the granularity of layers, we propose qualitative metrics for evaluation based on edit distance and visual quality between layer sequences aligned by dynamic time warping (DTW) [34], which account for the inconsistency of the ground truth layers. We compare LayerD with several baselines and demonstrate that our method achieves the highest quality.

We summarize our contributions as follows:

- We propose LayerD, a fully automatic framework for layer decomposition from raster graphic designs. LayerD unifies the subtasks inherent in layer decomposition into iterative top-layer extraction and leverages domain priors to improve the final decomposition quality.
- We propose a consistent evaluation protocol for layer decomposition based on the edit distance and appearance quality between aligned layer sequences, which accounts for the ambiguity in the ground truth layer structure.
- We empirically show that LayerD achieves the highest quality compared to baselines and decomposed layers can be used for downstream graphic design editing.

2. Related Work

2.1. Image Layer Decomposition

Image layer decomposition is a task to decompose an image into a sequence of layers, which are composable with a specific compositing function (e.g., alpha compositing) to reproduce or approximate the original image [36]. Color segmentation represents an image with semi-transparent color layers, targeting digital paintings [49] and natural images [1, 2, 50]. Koyama *et al*. [19] propose to handle non-linear color blending functions, followed by the efficient deep learning-based extension [12].

There have been many studies on decomposing natural scenes at the object level [15, 30, 33, 51, 59, 61]. For instance, PCNet [59] decomposes a scene image into object layers by estimating the order of objects and the RGB of occluded parts. While PCNet assumes the object modal mask is given, Zhang *et al*. [61] create layered data including occluded parts in indoor scenes and decompose the image by training instance segmentation, depth estimation, and background completion. Text2Layer [60] extracts salient objects from natural images using matting and generates training data for layered image generation. Recently, MULAN [51] decomposes natural images, including outdoor scenes where obtaining ground truth data is difficult, by combining the latest off-the-shelf open vocabulary object detection models [57], zero-shot segmentation [18], depth estimation [37], and instance ordering [22] with heuristics. While the above studies mainly focus on object decomposition, Yang *et al*. [55] decompose physical object effects (*e.g*., shadows or reflections) as well.

Compared to the natural image decomposition, graphic design decomposition has to deal with different granularities of *objects*; *e.g*., a corporate logo in a graphic design consists of an illustration and a text, and whether they should be decomposed into parts depends on the context. Considering the nature of the task, we propose a simple and effective method and a new quantitative evaluation protocol for inconsistent ground-truth. A concurrent work [5] tackles the same task as ours with a stacked pipeline approach

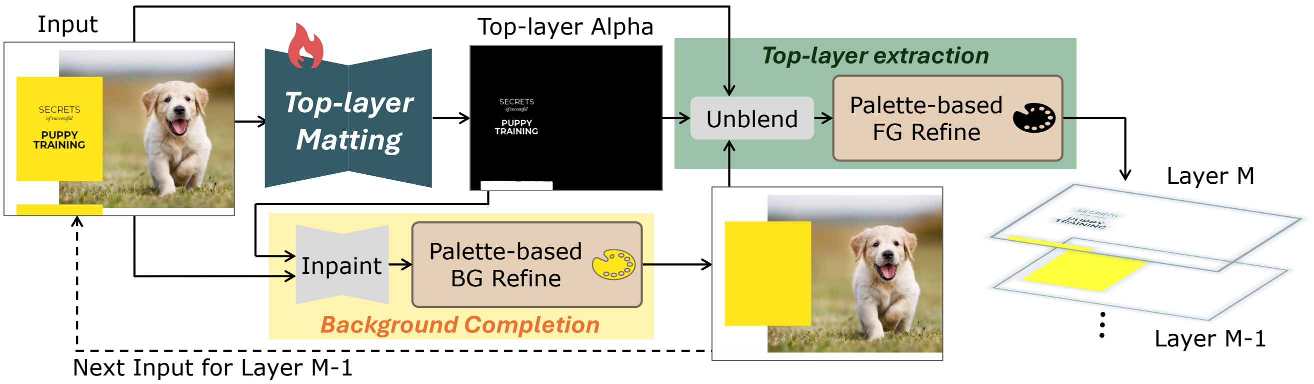


Figure 2. LayerD decopmoses raster graphic designs into layers by iteratively extracting the top-layer and completing the background. Our training target is the top-layer matting model. Figs. 3 and 4 illustrate details of the top-layer extraction and background completion.

using a VLM trained on closed data. LayerD's pipeline is overwhelmingly simple and leverages domain knowledge to refine the final quality. We compare LayerD with a VLM-based pipeline in our experiment.

2.2. Image Vectorization

Related to layer decomposition, image vectorization converts an image or a part into a set of parameters of a specific drawing function, rather than layer images. Our layer decomposition approach can be useful for vectorization as a pre-processing step to extract part-based raster images. Du *et al.* [8] and Favreau *et al.* [9] obtain a sequence of linear gradient layers that approximate the original image by optimization using alpha blending. Several works attempt to generate SVG-based representation from raster images [4, 32, 39, 40, 42, 44], where they typically assume vector art, cleanly masked images, or clean segmented images as input. A few specifically focus on typographic representation in graphic design, where they estimate text rendering parameters [5, 43].

2.3. Image Matting and Foreground Extraction

Image matting is a task to estimate alpha mattes of objects in an image, and together with other tasks such as background inpainting, forms the layer decomposition task. Matting approach often assumes user-specified trimap [7, 46, 53, 56], and a few trimap-free methods have been reported recently [24, 62]. LayerD mainly uses network architectures used in matting [62] to extract unoccluded top layers.

While matting estimates alpha mattes, foreground color estimation involves determining the color of the foreground that is mixed with the background. There are energy-based methods [3, 6, 23] and their efficient versions [10, 11], as well as deep learning-based methods [31] that estimate the foreground color given the alpha. Hou *et al.* and Li *et al.* simultaneously estimate the alpha map and foreground color given an image and a trimap [13, 25]. The foreground color is deterministic when the background color and foreground alpha are given. In our setup, we obtain the foreground matte from our trained model and the background color from high-quality background inpainting [47], and then calculate the foreground color.

3. Problem Formulation

Graphic layer decomposition is the task of decomposing a raster graphic design image $\boldsymbol{x} \in [0,1]^{H \times W \times 3}$ into a sequence of layers $Y = (\boldsymbol{l}_k \in [0,1]^{H \times W \times 4})_{k=0}^{K}$. Here, H and W represent the height and width of the image, respectively. $\boldsymbol{x}$ is an RGB image, and $\boldsymbol{l}_k$ is an RGBA image, with 3 and 4 channels, respectively. k represents the blending order of the layer, *i.e.*, the z-index. $\boldsymbol{l}_{k>0}$ is the foreground layer, and $\boldsymbol{l}_0$ is the background layer.

The layer sequence Y is composited by the following recursive process from $k = 1$ to $k = K$ ($\boldsymbol{x} = \boldsymbol{x}_K$):

$$\boldsymbol{x}_k^{\mathrm{C}} = \mathrm{B}(\boldsymbol{l}_k, \boldsymbol{x}_{k-1}^{\mathrm{C}}), \tag{1}$$

$$= \boldsymbol{l}_k^{\mathrm{C}} \odot \boldsymbol{l}_k^{\mathrm{A}} + \boldsymbol{x}_{k-1}^{\mathrm{C}} \odot (1 - \boldsymbol{l}_k^{\mathrm{A}}). \tag{2}$$

Here, the superscript A represents the alpha channel, and C represents one of the RGB channels. $\mathrm{B}(\cdot)$ is the alpha blending function, $\odot$ is element-wise multiplication, and $\boldsymbol{x}_k$ is the k-th blended image.

In this study, we solve the inverse problem of the above, *i.e.*, layer decomposition that estimates the layer sequence Y from the raster image $\boldsymbol{x}$. The granularity of the layer depends on the dataset, and in this study, we treat the human-made graphic designs in the dataset as ground truth.

4. Approach

LayerD solves the decomposition task by iterative extraction of *top-layers*, which are not occluded by any other layers, and background completion (Fig. 2). Our formulation integrates the subtasks of layer decomposition, which prior methods [5, 51] separately address, into a single task, leading to a simplified implementation and performance gain by the simple training goal. Additionally, we refine the final decomposition quality by leveraging the domain prior that graphic designs often contain simple, texture-less elements or backgrounds.

4.1. Iterative Decomposition

We obtain layer predictions $\hat{Y} = (\hat{\boldsymbol{l}}_m \in [0,1]^{H\times W\times 3})_{m=0}^{M}$ from an input image $\boldsymbol{x}$ by iterative processes from front ($m = M$) to back ($m = 1$) as follows:

$$\hat{\boldsymbol{l}}_m^{\mathrm{A}} = F_\theta(\hat{\boldsymbol{x}}_m) \tag{3}$$

$$\hat{\boldsymbol{x}}_{m-1} = G_\phi(\hat{\boldsymbol{x}}_m, \hat{\boldsymbol{l}}_m^{\mathrm{A}}) \tag{4}$$

$$\hat{\boldsymbol{l}}_m^{\mathrm{C}} = \mathrm{B}^{-1}(\hat{\boldsymbol{x}}_{m-1}^{\mathrm{C}}, \hat{\boldsymbol{x}}_m^{\mathrm{C}}, \hat{\boldsymbol{l}}_m^{\mathrm{A}}) \tag{5}$$

$$= (\hat{\boldsymbol{x}}_m^{\mathrm{C}} - \hat{\boldsymbol{x}}_{m-1}^{\mathrm{C}} \odot (1 - \hat{\boldsymbol{l}}_m^{\mathrm{A}})) \oslash \hat{\boldsymbol{l}}_m^{\mathrm{A}} \tag{6}$$

where $\hat{\boldsymbol{x}}_M = \boldsymbol{x}$, $\hat{\boldsymbol{l}}_0 = \hat{\boldsymbol{x}}_0$, and $\oslash$ is an element-wise division. The superscripts A and C denote the alpha channel and one of the RGB channels, respectively. $F_\theta(\cdot)$ is a model that takes an image as input and outputs an alpha map, which is the same as the trimap-free matting task, as long as the matting target is the top-layers. The output alpha contains all top layers; they are decomposed in one iteration. $G_\phi(\cdot)$ is a background completion model that takes an image and a target mask obtained from the top-layers alpha map as input and outputs an image with the target area completed. The background completion model should not insert new objects. We tried several inpainting approaches, including generative model-based completion [20], and found that generative approaches often insert unnecessary objects. We use LaMa [47] for G_ϕ, which satisfies our inpainting requirement. $\mathrm{B}^{-1}(\cdot)$ is a process that estimates the RGB values from the completed background and the alpha map of top-layers. Since we know the alpha map and the completed background, we can calculate the RGB values of top-layers by simple arithmetic as the inverse process of alpha blending $B(\cdot)$. Note that existing methods [5, 51] replace the alpha of the original image with the predicted soft or hard segmentation mask. For pixels with an alpha of 1, our method estimates the same RGB values as these naive methods. However, for other transparent pixels, our method estimates the RGB values while considering blending with the background. This primarily improves the quality of layer boundaries where soft blending is applied. We terminate the iteration (*i.e.*, $m = M$) when there are no pixels above a certain threshold in the matting result $\hat{\boldsymbol{l}}_m^{\mathrm{A}}$.

4.2. Training

In LayerD, we utilize two learnable models: top-layer matting model F_θ and background inpainting model $G_\phi(\cdot)$. Since image inpainting is a general task and reasonably performant models are available in the public [47], we use an off-the-shelf pretrained model without fine-tuning. For training the top-layer matting model, we prepare pairs of an input RGB image $\boldsymbol{x}$ and a target alpha map of top-layers $\boldsymbol{l}^{\mathrm{A}}$ from Crello [54] for supervised learning. We first check the occlusion of each layer based on the layer information and integrate the alpha maps of non-occluded layers into a single alpha map. This clear target definition eliminates ambiguity in training. Similarly to LayerD's pipeline, we create multiple pairs per design sample by recursively performing the same process on the remaining background.

We follow the prior study [62] and define the loss function as below:

$$\mathcal{L}(\hat{\boldsymbol{l}}^{\mathrm{A}}, \boldsymbol{l}^{\mathrm{A}}) = \lambda_{\mathrm{BCE}}\mathcal{L}_{\mathrm{BCE}}(\hat{\boldsymbol{l}}^{\mathrm{A}}, \boldsymbol{l}^{\mathrm{A}}) + \lambda_{\mathrm{IoU}}\mathcal{L}_{\mathrm{IoU}}(\hat{\boldsymbol{l}}^{\mathrm{A}}, \boldsymbol{l}^{\mathrm{A}}) + \lambda_{\mathrm{SSIM}}\mathcal{L}_{\mathrm{SSIM}}(\hat{\boldsymbol{l}}^{\mathrm{A}}, \boldsymbol{l}^{\mathrm{A}}), \tag{7}$$

where $\mathcal{L}_{\mathrm{BCE}}(\cdot)$, $\mathcal{L}_{\mathrm{IoU}}(\cdot)$, and $\mathcal{L}_{\mathrm{SSIM}}(\cdot)$ are binary cross-entropy, Intersection-over-Union (IoU) loss, and structural similarity index (SSIM) loss, respectively, and λ_{BCE}, λ_{IoU}, and λ_{SSIM} are weights for each loss term. We train the matting model using all loss functions at the early steps and then use only the SSIM loss to improve the boundary quality.

During inference, the model takes the background completion result ($\hat{\boldsymbol{x}}_m$) as input instead of the clean intermediate composite result, except for the first iteration. This gap between training and inference can degrade the decomposition quality. To make the matting model robust to the inpainting artifacts, we include training examples where the top-layer regions are completed by the background completion model. Since the completion in areas spanning the back layers can alter their shape, leading to inconsistencies with the ground truth, we do not complete such areas when making training data.

4.3. Palette-based Refinement

Graphic designs often contain flat elements or backgrounds with few textures, such as decorations, texts, and vector shapes. Based on this observation, we introduce a simple refinement approach that greatly improves the resulting appearance at the end of each iteration.

Background refinement (Fig. 3) We first divide the alpha map into connected regions and process each area. We calculate the color gradients of the surrounding area of the completion target area. If the area with zero color gradients is dominant, we assume that the completion target region is a flat-paint area with few textures. We extract the dominant colors, *i.e.*, the palette, of the region based on percentiles and assign the completed RGB values to the nearest palette color in the Lab color space. The background completion model can make rough predictions for such flat backgrounds even with noticeable artifacts, allowing our simple

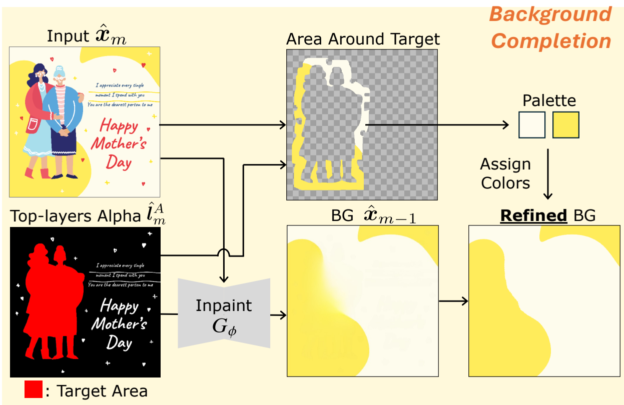


Figure 3. Background completion with palette-based refinement. We first complete the area of the predicted alpha map, then refine the target connected region based on the color palette of the surrounding area. We select the target area based on the color gradient of the surrounding area (shown in red).

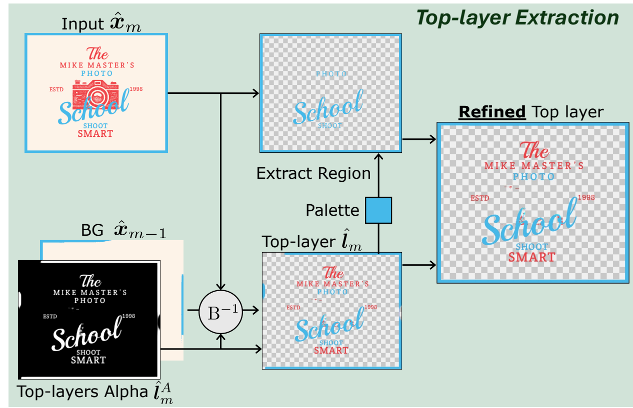


Figure 4. Palette-based foreground refinement. First, we estimate the RGB values of the top-layer using the input image, the top-layer's alpha, and the background by the unblend function. Next, we extract the color palette of the connected components of the top-layer, extract the region that matches the color from the original image, integrate the connected color region with a large overlap with the predicted alpha map, and use it as a new alpha map. Note that the missing blue edge is refined in this figure.

refinement to work effectively. Our refinement eliminates such artifacts.

Foreground refinement (Fig. 4) Similar to background refinement, we first divide the alpha map into connected regions. Next, we calculate the color gradients within each region and classify regions where the area with zero color gradients dominates as flat regions. We extract the region that matches the palette color from the input image ($\boldsymbol{x}$) or intermediate completed background ($\hat{\boldsymbol{x}}_m$) and select the region if the overlap with the predicted alpha exceeds a threshold. Then, we integrate the selected regions as a new mask. Since the obtained mask is binary, we calculate the alpha value around the mask using the palette color and the background color to derive the final alpha value. We often observe that this refinement improves the quality of the boundary parts and thin decoration layers (*e.g.*, lines and frames), which the plain matting model frequently fails.

5. Decomposition Metrics

There are two problems in evaluating the quality of predicted layers $\hat{Y}$ against the ground truth Y. First, the number of layers in the ground truth and the predicted layers can differ, making it non-trivial to compare directly. To address this, we apply order-aware layer alignment using DTW [34]. Second, the quality of the predicted layers can be evaluated from two perspectives: visual quality and granularity. If we do not consider these aspects separately, we may underestimate the quality of the predicted layers due to differences in granularity, even if they are practically useful. We measure granularity by the number of edits required to align the two; we allow merging adjacent layers in the z-index and report both the number of edits and the visual quality after the editing operations.

Layer alignment As pre-processing, we first group the ground truth and predicted layers based on visibility. Specifically, we extract layers whose visible regions (*i.e.*, alpha values greater than zero) are not occluded by any other higher layers in z-index, blend them into a single layer, and repeat the same operation with the remaining layers. This operation never affects the appearance of the composite image and forms what we refer to as a top-layer.

Next, we find alignment between the two layer sequences with different lengths using DTW, which considers the sequence order even if the lengths differ. We obtain a set of pairs $P = \{(k_s, q_s) | s = 0, 1, \ldots, S\}$, where k_s and q_s represent the layer indices, and S is the number of pairs. Note that resolved pairs satisfy the monotonicity condition, *i.e.*, k_s and q_s are increasing sequences; in other words, layers cannot be shuffled during alignment (see Supp. Sec. F.1). We define the distance metric for the layer pair as the sum of the negative value of the alpha's soft IoU and L1 distance of the RGB channels weighted by the ground-truth alpha, as introduced in [48].

Finally, we compute the quality metric between the two layer sequences as follows:

$$\mathcal{E}(\hat{Y}, Y) = \frac{1}{S} \sum_{s=0}^{S} e(\hat{\boldsymbol{l}}_{k_s}, \boldsymbol{l}_{q_s}), \tag{8}$$

where $e(\cdot)$ is an arbitrary function that measures the similarity or distance between layers. We use the weighted L1 distance of the RGB channels and the soft IoU of the alpha channel as $e(\cdot)$, similar to DTW's distance metric.

Layer merge Due to the ill-posed nature of layer decomposition, decomposition results sometimes do not align well with the ground truth. In this work, we relax the alignment constraints by allowing *edits*. The idea is inspired by minimum edit distance [52], which is commonly used for string alignment. We define a specific edit operation set for layers, and report both the maximum number of allowed edits and the distance metric used in DTW after edits. This gives a straightforward insight into how many layer-level edits are required for good alignment.

For simplicity, we define a single edit operation; `Merge`, which merges two consecutive layers in z-index, when the edit yields the highest positive distance improvement. We apply edits iteratively until no further improvements are possible or the number of layers is reduced to 2. The ground truth is also mergeable. Visual examples of the edit process can be found in Supp. Figs. G and H.

6. Experiments

6.1. Datasets

We use the Crello dataset [54], which is a collection of graphic design templates, for both training and evaluation. We obtain pairs of input images and their ground-truth layer sequences from the layer structure in design templates. We follow the data split of v5.1.0 to obtain 19,478 / 1,852 / 1,971 samples for train, validation, and test split, respectively. We resize all images to maintain their aspect ratio, with the shorter side set to 512. In this work, we exclude transparent layers from the evaluation since neither our method nor the baselines primarily focuses on accurate transparency estimation. For all methods, we conduct training and validation on the train and validation split, respectively, and report the results on the test split. For training of LayerD as described in Sec. 4.2, we generate input / ground-truth pairs, and finally obtain 48,725 and 4,674 pairs for the train and validation, respectively. As typography is one of the unique domain properties, we prepare the full dataset and the variant that excludes all text layers for evaluation.

6.2. Baselines

Although there are a few methods comparable to LayerD, none of them have publicly available code or models. We design the following baseline and implement an existing approach with minor modifications to fit our setting. Additionally, we evaluate all methods using Hi-SAM [58], a state-of-the-art text segmentation, for initial layer extraction.

YOLO baseline We design a naive baseline that combines state-of-the-art object detection and pretrained segmentation models. First, since most graphic designs contain text on the top, we segment text using Hi-SAM [58] and complete the background using LaMa [47]. Next, we detect bounding boxes of layers from the remaining background. To this end, we extract bounding boxes from the layer structure of Crello and fine-tune YOLO [17] using them. We determine the z-index of layers based on heuristics in graphic design, assuming that the smaller box is in front when the boxes overlap. Then, we obtain the segmentation masks of the topmost boxes using a pretrained SAM2 [38] and perform background completion. We repeat this process, except for text segmentation, until the number of detections becomes zero. We obtain the final layers by replacing the alpha of the input or completion image with the predicted segmentation mask and blacking out the color of pixels with an alpha close to zero.

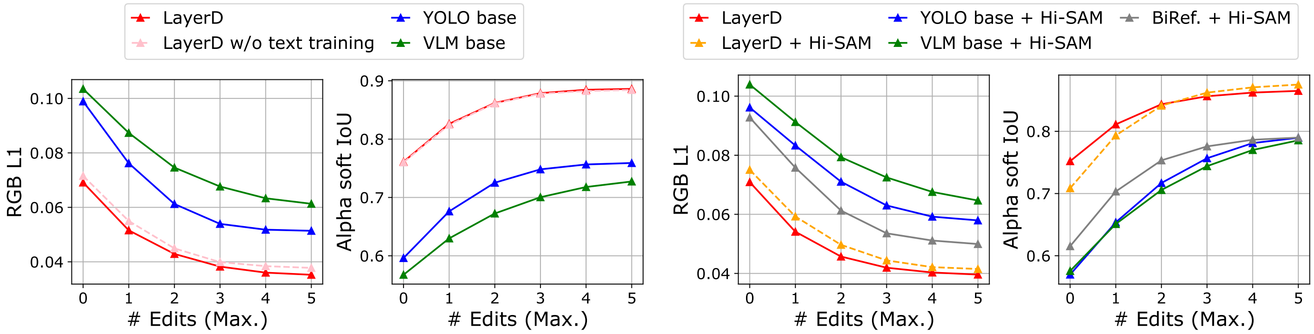
VLM baseline We also consider a baseline that follows the approach of Accordion [5]. Accordion generates layered graphic design by combining raster-based image generation [21] and layer decomposition, where VLM first takes an image as input and generates a decomposition plan with a JSON-like description of bounding boxes and z-indices of layers, and then applies segmentation and background completion in a front-to-back order to obtain the layer sequence. Since the model and training data are not publicly available, we reproduce this with minor modifications in our experiment. We use PaliGemma2 [45] as the backbone VLM and fine-tune on the Crello train set, Hi-SAM for text detection, and SAM2 for other elements. For background completion, we use LaMa [47] to ensure fairness with other methods.

6.3. Implementation Details

We use BiRefNet [62] with Swin-L [29] pre-trained on natural image object segmentation for the top-layers matting model, and train it on Crello for 60 epochs with a batch size of 12. We set the maximum number of iterations for decomposition to 3 for LayerD and the YOLO baseline. The maximum number of colors for palette-based refinement is set to 10 for the foreground and 2 for the background. For evaluation, we change the maximum number of allowed edits from 0 to 5, and report the visual quality for each case.

6.4. Quantitative Evaluation

Baseline comparison We compare LayerD with baselines with and without text layers in Fig. 5b and Fig. 5a, respectively. In all metrics, LayerD generates layer sequences close to the ground truth with fewer edits. Our simplified pipeline and training objective are effective in layer decomposition. Moreover, in the results for all layers (Fig. 5b), LayerD alone shows higher performance than LayerD + Hi-SAM, which replaces the first iteration with Hi-SAM.



(a) Without text layers

(b) All layers

Figure 5. Baseline comparisons. We show visual quality metrics (RGB L1, Alpha IoU) as the maximum number of allowed edits increases. The left two are the results when we exclude text layers from the dataset, and the right two are the results when all layers are included. "w/o text training" indicates the case where text layers are not included during training.

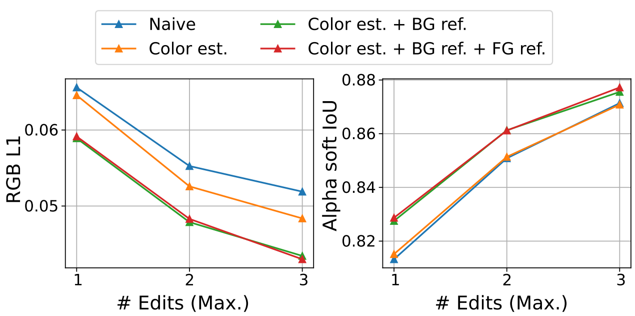


Figure 6. Ablation results of foreground color estimation and refinement.

LayerD, which is specifically trained for graphic design, is more effective than Hi-SAM, which is trained for text segmentation without being limited to graphic design. More interestingly, in the case of decomposition without text layers, as shown in Fig. 5a, LayerD trained with text layers exhibits slightly better performance than LayerD trained without text, even though the decomposition targets contain no text. We suspect that text is essentially a variant of vector shapes, and training with text layers improves the decomposition performance of these elements. Our method outperforms the BiRefNet without additional training in Fig. 5b, indicating the importance of training on top-layer matting.

Refinement ablation Fig. 6 shows the ablation results of foreground color estimation and refinement. First, the color estimation by the inverse blending (Eq. (6)) reduces the RGB L1 compared to the "Naive", which simply replaces the alpha with the predicted mask. Background refinement significantly improves the RGB L1, resulting in better subsequent layer decomposition, as indicated by the improvement in Alpha IoU. The foreground refinement slightly improves the quality of the alpha map. Although the quantitative improvement is slight, we observe that foreground refinement improves the quality of boundary regions.

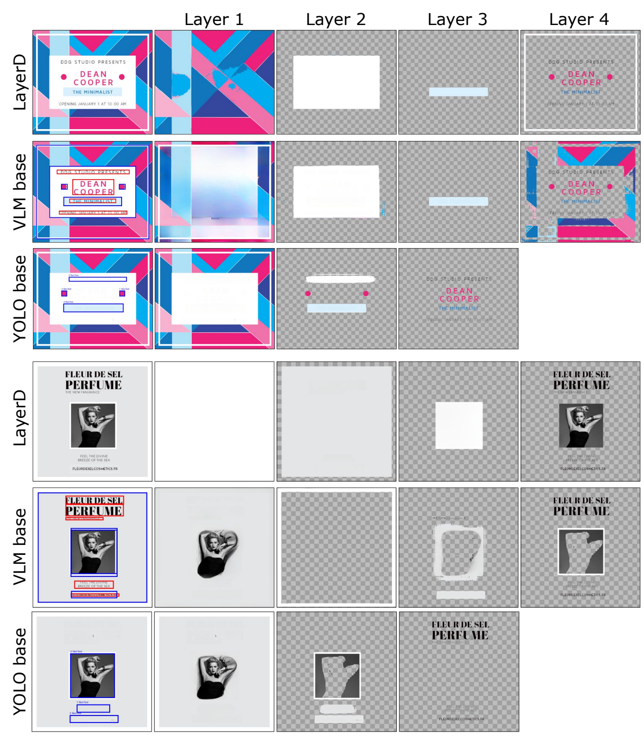


Figure 7. Comparison of decomposition results by LayerD and baselines. The leftmost images are the input image (LayerD), object detections (VLM), and text-removed input (YOLO). Red rectangles indicate text, and blue rectangles indicate other elements.

6.5. Qualitative Evaluation

Baseline comparison Fig. 7 shows the qualitative comparison of LayerD with VLM and YOLO baseline. The VLM baseline, which relies on bounding boxes, struggles

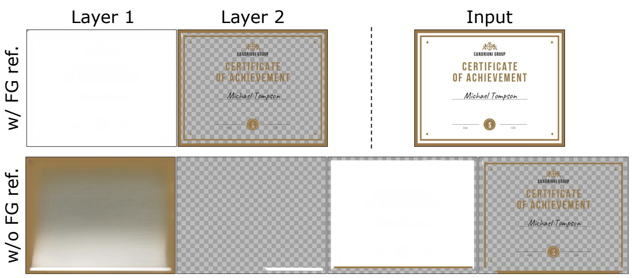


Figure 8. Example with or without foreground refinement. Without refinement, layers tend to have a collapsed boundary.

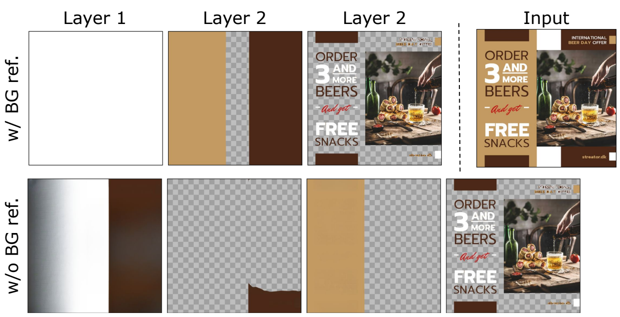


Figure 9. Example with or without background refinement. Our refinement prevents the background from being divided into segments or filled with unexpected colors.

with proper decomposition when detection fails or when the detected boxes overlap. The bounding box of a layer with a large hole like a donut includes the elements of the entire image, making it difficult for the subsequent segmentation (Fig. 7 top sample). The YOLO baseline suffers from false negatives in detection, resulting in incomplete decomposition. In contrast, LayerD directly extracts layers without relying on bounding boxes, resulting in clean decomposition results in all cases.

Refinement effect Fig. 8 shows an example of foreground refinement. The foreground refinement recovers the large missing gold decoration with a large hole, which is a typical failure case of the top-layers matting model. In Fig. 9, we show an example of background refinement. Background refinement successfully completes the missing part of the background. Since layer decomposition is recursive, failures in the previous iteration have a negative impact on subsequent iterations. Our refinement contributes not only to the quality of the target layer itself but also to the quality of the subsequent layers, *i.e.*, the overall quality of the layer decomposition.

6.6. Applications

In this section, we demonstrate that LayerD enables a few applications out of the box.

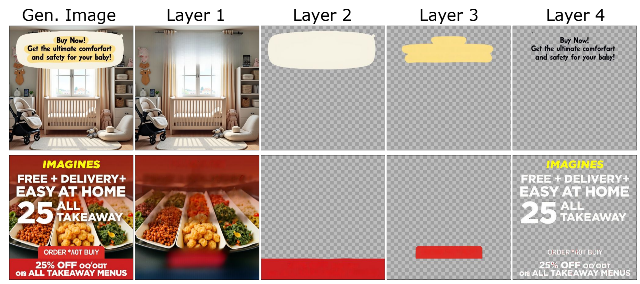


Figure 10. Decomposition results of LayerD on raster images generated by FLUX.1 [dev] [21].

Decomposing generated images Fig. 10 shows the decomposition results of LayerD on graphic design generated by FLUX.1 [dev] [21], which is one of the state-of-the-art text-to-image generator, using prompts from DESIGNERINTENTION-v2 benchmark [16]. Note that quantitative evaluation is not possible in this setup as ground truth layers are not available. The results suggest that LayerD can generalize and successfully decompose generated images, enabling an editable workflow for raster image generators.

Image editing We show the image editing examples using the decomposed layers by LayerD in Fig. 1 and Supp. Sec. A. Here, we only perform simple operations such as color conversion, translation, and resizing at the layer level. Although the layers obtained by LayerD are grouped by top-layers, we can easily divide them into connected components for more granular editing. Our decomposition enables flexible and intuitive layer-based editing. Note also that there is no significant artifact in the editing results.

7. Conclusion

We present LayerD for decomposing raster graphic designs, where we propose the iterative extraction of unoccluded layers and background completion as well as refinement tailored for graphic materials. We propose an evaluation protocol for the ill-posed decomposition task, where we introduce the notion of layer edit to quantify the difference from the unreliable ground truth. The experiment showed that LayerD led to solid improvement over baselines.

Our approach aims at decomposition, but it might be interesting to further consider vectorization [4, 32, 39, 44], which would expand the possible creative workflow. In the other direction, our method could help learn a layered design generation model [14, 16] as a preprocessing component, or be combined with recent automatic layered design editing [26, 35], or animation generation [28] for interesting applications.

References

[1] Naofumi Akimoto, Huachun Zhu, Yanghua Jin, and Yoshimitsu Aoki. Fast soft color segmentation. In *CVPR*, 2020. 2

[2] Yağiz Aksoy, Tunç Ozan Aydin, Aljoša Smolić, and Marc Pollefeys. Unmixing-based soft color segmentation for image manipulation. *ACM TOG*, 36(2), 2017. 2

[3] Yagiz Aksoy, Tunc Ozan Aydin, and Marc Pollefeys. Designing effective inter-pixel information flow for natural image matting. In *CVPR*, 2017. 3

[4] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. In *NeurIPS*, 2020. 3, 8

[5] Jingye Chen, Zhaowen Wang, Nanxuan Zhao, Li Zhang, Difan Liu, Jimei Yang, and Qifeng Chen. Rethinking layered graphic design generation with a top-down approach. *arXiv preprint arXiv:2507.05601*, 2025. 2, 3, 4, 6

[6] Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. Knn matting. *IEEE TPAMI*, 35(9), 2013. 3

[7] Yung-Yu Chuang, Brian Curless, David H Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *CVPR*, 2001. 3

[8] Zheng-Jun Du, Liang-Fu Kang, Jianchao Tan, Yotam Gingold, and Kun Xu. Image vectorization and editing via linear gradient layer decomposition. *ACM TOG*, 42(4), 2023. 3

[9] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM TOG*, 36(6), 2017. 3

[10] Marco Forte. Approximate fast foreground colour estimation. In *ICIP*, 2021. 3

[11] Thomas Germer, Tobias Uelwer, Stefan Conrad, and Stefan Harmeling. Fast multi-level foreground estimation. In *ICPR*, 2021. 3

[12] Daichi Horita, Kiyoharu Aizawa, Ryohei Suzuki, Taizan Yonetsuji, and Huachun Zhu. Fast nonlinear image unblending. In *WACV*, 2022. 2

[13] Qiqi Hou and Feng Liu. Context-aware image matting for simultaneous foreground and alpha estimation. In *ICCV*, 2019. 3

[14] Naoto Inoue, Kento Masui, Wataru Shimoda, and Kota Yamaguchi. OpenCOLE: Towards Reproducible Automatic Graphic Design Generation. In *CVPRW*, 2024. 8

[15] Phillip Isola and Ce Liu. Scene collaging: Analysis and synthesis of natural images with semantic layers. In *ICCV*, 2013. 2

[16] Peidong Jia, Chenxuan Li, Zeyu Liu, Yichao Shen, Xingru Chen, Yuhui Yuan, Yinglin Zheng, Dong Chen, Ji Li, Xiaodong Xie, et al. COLE: A hierarchical generation framework for graphic design. *arXiv preprint arXiv:2311.16974*, 2023. 8

[17] Rahima Khanam and Muhammad Hussain. YOLOv11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*, 2024. 6

[18] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 2

[19] Yuki Koyama and Masataka Goto. Decomposing images into layers with advanced color blending. *CGF*, 37(7), 2018. 2

[20] Black Forest Labs. FLUX.1 fill [dev]. `https://huggingface.co/black-forest-labs/FLUX.1-Fill-dev`, 2024. Last accessed 7 March, 2025. 4

[21] Black Forest Labs. FLUX.1 [dev]. `https://huggingface.co/black-forest-labs/FLUX.1-dev`, 2024. Last accessed 7 March, 2025. 6, 8

[22] Hyunmin Lee and Jaesik Park. Instance-wise occlusion and depth orders in natural scenes. In *CVPR*, 2022. 2

[23] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *IEEE TPAMI*, 30(2), 2007. 3

[24] Jiachen Li, Jitesh Jain, and Humphrey Shi. Matting anything. *arXiv: 2306.05399*, 2023. 3

[25] Xiaodi Li, Zongxin Yang, Ruijie Quan, and Yi Yang. Drip: Unleashing diffusion priors for joint foreground and alpha prediction in image matting. In *NeurIPS*, 2024. 3

[26] Jiawei Lin, Shizhao Sun, Danqing Huang, Ting Liu, Ji Li, and Jiang Bian. From elements to design: A layered approach for automatic graphic design composition. *arXiv preprint arXiv:2412.19712*, 2024. 8

[27] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 2

[28] Vivian Liu, Rubaiat Habib Kazi, Li-Yi Wei, Matthew Fisher, Timothy Langlois, Seth Walker, and Lydia Chilton. LogoMotion: Visually grounded code generation for content-aware animation. *arXiv preprint arXiv:2405.07065*, 2024. 8

[29] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 6

[30] Zhengzhe Liu, Qing Liu, Chirui Chang, Jianming Zhang, Daniil Pakhomov, Haitian Zheng, Zhe Lin, Daniel Cohen-Or, and Chi-Wing Fu. Object-level scene deocclusion. In *ACM SIGGRAPH Conference Papers*, 2024. 2

[31] Sebastian Lutz and Aljosa Smolic. Foreground color prediction through inverse compositing. In *WACV*, 2021. 3

[32] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *CVPR*, 2022. 3, 8

[33] Tom Monnier, Elliot Vincent, Jean Ponce, and Mathieu Aubry. Unsupervised layered image decomposition into object prototypes. In *ICCV*, 2021. 2

[34] Meinard Müller. *Information Retrieval for Music and Motion*. Springer Verlag, 2007. 2, 5

[35] Sohan Patnaik, Rishabh Jain, Balaji Krishnamurthy, and Mausoom Sarkar. Aesthetiq: Enhancing graphic layout design via aesthetic-aware preference alignment of multi-modal large language models. In *CVPR*, 2025. 8

[36] Thomas Porter and Tom Duff. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, 1984. 2

[37] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE TPAMI*, 44(3), 2020. 2

[38] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. SAM 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 2, 6

[39] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In *CVPR*, 2021. 3, 8

[40] Juan A Rodriguez, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images. *arXiv preprint arXiv:2312.11556*, 2023. 3

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2

[42] I-Chao Shen and Bing-Yu Chen. Clipgen: A deep generative model for clipart vectorization and synthesis. *IEEE TVCG*, 28(12), 2021. 3

[43] Wataru Shimoda, Daichi Haraguchi, Seiichi Uchida, and Kota Yamaguchi. De-rendering stylized texts. In *ICCV*, 2021. 3

[44] Yiren Song, Xuning Shao, Kang Chen, Weidong Zhang, Zhongliang Jing, and Minzhe Li. Clipvg: Text-guided image manipulation using differentiable vector graphics. In *AAAI*, 2023. 3, 8

[45] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, et al. PaliGemma 2: A family of versatile VLMs for transfer. *arXiv preprint arXiv:2412.03555*, 2024. 6

[46] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. In *ACM SIGGRAPH Conference Papers*. 2004. 3

[47] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *WACV*, 2022. 3, 4, 6

[48] Tomoyuki Suzuki, Kotaro Kikuchi, and Kota Yamaguchi. Fast sprite decomposition from animated graphics. In *ECCV*, 2024. 5

[49] Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. Decomposing images into layers via rgb-space geometry. *ACM TOG*, 36(1), 2016. 2

[50] Jianchao Tan, Jose Echevarria, and Yotam Gingold. Efficient palette-based decomposition and recoloring of images via rgbxy-space geometry. *ACM TOG*, 37(6), 2018. 2

[51] Petru-Daniel Tudosiu, Yongxin Yang, Shifeng Zhang, Fei Chen, Steven McDonagh, Gerasimos Lampouras, Ignacio Iacobacci, and Sarah Parisot. Mulan: A multi layer annotated dataset for controllable text-to-image generation. In *CVPR*, 2024. 2, 4

[52] Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *JACM*, 21(1), 1974. 6

[53] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *CVPR*, 2017. 3

[54] Kota Yamaguchi. CanvasVAE: Learning to generate vector graphic documents. In *ICCV*, 2021. 4, 6

[55] Jinrui Yang, Qing Liu, Yijun Li, Soo Ye Kim, Daniil Pakhomov, Mengwei Ren, Jianming Zhang, Zhe Lin, Cihang Xie, and Yuyin Zhou. Generative image layer decomposition with visual effects. *arXiv preprint arXiv:2411.17864*, 2024. 2

[56] Jingfeng Yao, Xinggang Wang, Shusheng Yang, and Baoyuan Wang. Vitmatte: Boosting image matting with pretrained plain vision transformers. *Information Fusion*, 103, 2024. 3

[57] Lewei Yao, Jianhua Han, Xiaodan Liang, Dan Xu, Wei Zhang, Zhenguo Li, and Hang Xu. Detclipv2: Scalable open-vocabulary object detection pre-training via word-region alignment. In *CVPR*, 2023. 2

[58] Maoyuan Ye, Jing Zhang, Juhua Liu, Chenyu Liu, Baocai Yin, Cong Liu, Bo Du, and Dacheng Tao. Hi-sam: Marrying segment anything model for hierarchical text segmentation. *arXiv preprint arXiv:2401.17904*, 2024. 6

[59] Xiaohang Zhan, Xingang Pan, Bo Dai, Ziwei Liu, Dahua Lin, and Chen Change Loy. Self-supervised scene deocclusion. In *CVPR*, 2020. 2

[60] Xinyang Zhang, Wentian Zhao, Xin Lu, and Jeff Chien. Text2layer: Layered image generation using latent diffusion model. *arXiv preprint arXiv:2307.09781*, 2023. 2

[61] Chuanxia Zheng, Duy-Son Dao, Guoxian Song, Tat-Jen Cham, and Jianfei Cai. Visiting the invisible: Layer-by-layer completed scene decomposition. *IJCV*, 129, 2021. 2

[62] Peng Zheng, Dehong Gao, Deng-Ping Fan, Li Liu, Jorma Laaksonen, Wanli Ouyang, and Nicu Sebe. Bilateral reference for high-resolution dichotomous image segmentation. *CAAI Artificial Intelligence Research*, 3, 2024. 3, 4, 6