The Visual Computer (2019) 35:609–622
https://doi.org/10.1007/s00371-018-1490-1

ORIGINAL ARTICLE

Importance-based approach for rough drawings

Heekyung Yang[1] · Kyungha Min[1]

Published online: 8 March 2018

Abstract
We present a framework for producing rough drawings from photographs. Depicting a scene using a series of lines is one of the most effective methods of visual communication. Our framework for rough drawing is comprised of three steps: extracting lines from images, estimating line importance, and producing strokes that express various styles. To extract lines, we employ the widely used difference-of-Gaussian filter approach to devise a fault-correcting line shift scheme. Line importance is estimated by combining gradient and saliency. To obtain an efficient saliency estimation, we propose a stochastic content-based method. Various styles of rough drawings are produced by convoluting adaptive stroke texture segments, which are prepared by sampling real stroke texture images. We test our framework on various images and compare our results with real artwork and other schemes.

Keywords Rough drawing · Pencil · Charcoal · Convolution · Saliency · DoG

1 Introduction

Throughout history, rough drawing has been recognized as an important genre of fine art. Many artists express rough drawing using monochrome strokes created using stick-shaped media, such as a pencil or charcoal. They draw contours and salient features of scene objects by applying various strokes; they adjust tone using a series of hatching strokes. Owing to its ability to achieve high abstraction of objects and scenes, rough drawing is regarded as a very effective method of visual communication. The upper row of Fig. 1 illustrates several rough drawings produced by professional artists.

We present a scheme that produces rough drawings of target images by applying various styled strokes according to the importance estimated for the extracted feature curves. In the lower row of Fig. 1, we illustrate various styles of rough drawings created with our scheme, and then we compare them with those from the upper row.

We follow the artists' process of producing a rough drawing. Artists draw lines by placing various stylistic strokes along feature locations using the media they have chosen. Among the feature locations, artists tend to draw thick and salient strokes to emphasize the most important features.

✉ Kyungha Min
minkh@smu.ac.kr

[1] Department of Computer Science, Sangmyung University, Seoul, South Korea

We begin by mimicking an artist's decision of where to draw lines by estimating the importance of the feature curves. Importance is estimated by combining gradient and saliency, which are already widely used to analyze images. Then, we produce importance-guided strokes by simulating the artists media. Per our survey results, pencil and charcoal implements are the most frequently used media for expressing the necessary strokes.

To extract features from an image, difference-of-Gaussian (DoG) filter algorithms [7,11,24,25] are used as an important supporting technology. Among the features, we employ a flow-based DoG algorithm [11] that applies a weighted averaging filter for tangent vectors. However, the DoG scheme is seriously limited: the features extracted do not match the precise object borders. To achieve our goal of employing gradient and saliency to estimate feature importance, this mismatch becomes a serious obstacle. Therefore, we devise an algorithm to shift the lines back to the correct borders. With this strategy, we estimate the importance of the lines and distinguish more important features, such as contours, from less important features, such as interior lines.

To simulate the media used to produce strokes, we employ a convolution-based framework, which is widely used for simulating stick-shaped media such as pencil [9,17,26]. The convolution-based framework distributes black noise to mimic the distribution of pigments. The density of the noise distribution corresponds to the thickness of the produced stroke patterns. In related works, pixel-scale noise is used to

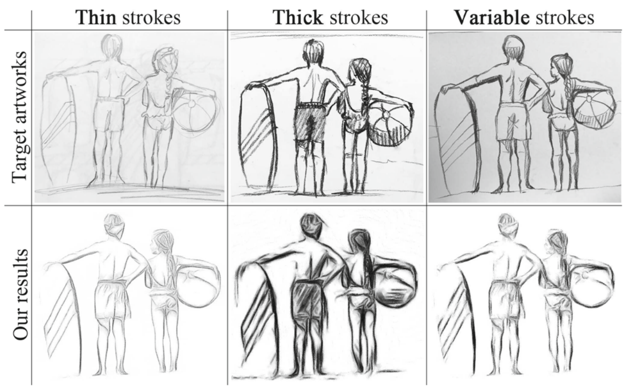


Fig. 1 Teaser: rough drawings with comparison. Upper row presents target works of art drawn by professionals. The left image is drawn with thin strokes using a B pencil. The middle image is drawn with thick strokes using charcoal. The right image uses strokes of various styles to depict a body. The contour is drawn using bold strokes, and the details are drawn using thin strokes. Our algorithm mimics these styles by estimating the importance of features and by controlling stroke styles

mimic pencil strokes [9,26,27] and rectangle-shaped noise is used to mimic pastel strokes [28]. Both approaches are not appropriate for our framework, since the stroke patterns in rough drawings have a wide range of variations.

To control the density of noise for rough drawing, we pioneer to sample real stroke textures and capture segments of various sizes. We scatter these texture segments over the line locations and then apply a convolution filter to integrate them. The strategy of employing stroke textures for noise distribution has the following benefits.

1. The density of noise distribution is controlled by employing stroke textures of various widths and lengths.
2. The stroke textures produced by various media such as H pencils and B pencils are employed to express the roughness of the strokes.

By controlling the widths and lengths of the segments, we simulate various stick-shaped media, such as sharp-tipped pencils and thick-tipped charcoal. We also control the stroke styles by perturbing the texture segments size and orientation. Because the existing convolution-based stroke-producing schemes [9,17,26] generate only thin strokes from pixel-scaled noise, we can produce thick and bold strokes using densely deposited noise from texture segments.

Our rough drawing scheme is distinguished from existing methods by two points.

1. Existing works concentrate on extracting prominent feature lines from scenes embedded in an image, whereas we also estimate importance of the feature lines. The importance values are incorporated to mimic an artists' drawing strategy.
2. Existing works employ convolution frameworks or texture-overlapping to mimic rough drawing strokes, whereas we devise a hybrid scheme that convolutes sampled stroke textures to express various styles by mimicking diverse media, from high-H pencils to charcoal.

This paper is organized as follows. We first summarize related works in Sect. 2, and we then present an outline of our work in Sect. 3. In Sects. 4 and 5, we present a line extraction scheme and a line stylization scheme, respectively. We illustrate and compare our results with others in Sect. 6. Finally, we draw conclusions and suggest future work in Sect. 7.

2 Related works

2.1 Lines on images

2.1.1 Extracting lines

For image processing, detecting and extracting important edges and lines is one of the most important research issues [19]. According to Papari and Petkov [19], edge detection schemes exploit various background techniques, such as local pattern analysis, global methods, contextual methods, and multiresolution approaches. In computer graphics, many researchers present schemes that convey objects salient shapes embedded in an image. In the early days, the Canny edge detector [2] was used in many research examples. Salisbury et al. [20] used the Canny edge detector to present a system of interactive pen-and-ink illustrations. Litwinowicz [16] also employed the Canny edge detector to clip paintbrush strokes to present an impressionist style. DeCarlo and Santella [4] combined the Canny edge detector and a mean shift algorithm to develop an image abstraction algorithm. Recently, the DoG filter has been widely used to extract salient lines from images and videos. Gooch et al. [7] used the DoG filter to extract edges from facial photographs to create illustrations and caricatures. Winnemoeller et al. [25] used the DoG filter to extract salient edges from a video, incorporating its stylization in real-time. Kang et al. [11] extended the DoG filter by using a flow-based approach, applying a bilateral filter to present a very convincing line-drawing algorithm to create coherent lines. Recently, a deep convolutional network is employed to extract lines from manga-style images [15]. Their network has a conv-deconv structure where the conv part is designed based on AlexNet. For training, they collected hundreds of examples from various artists. They aim to remove frequently used screen patterns from manga image to extract salient structure lines. Even though they successfully extracts the structure lines, it has a limitation to be applied to ordinary photographs, since their network was trained to remove only frequently used screen patterns such as stipples and hatching lines.

2.1.2 Stylizing lines

After extracting lines, several researchers presented schemes for stylization. Son et al. [21] presented an abstract line-drawing algorithm that extracts lines using a likelihood-function estimation, rendering extracted lines by applying stroke textures. Various illustration styles are provided by controlling detail, focus, and style of the lines. AlMeraj et al. [1] captured human pencil-drawing strokes to reproduce hand-drawn pencil lines. For this purpose, they conducted huge user studies and constructed a pencil-drawing stroke database. They decorated their lines with various styles by applying stroke textures captured from real strokes. Winnemoeller [24] extended the DoG algorithm to illustrate an image with various styles, including line-drawing, rough drawing, and black-and-white abstraction. Lu et al. [17] produced pencil-drawing lines by applying gradient filters to an image to extract pixels from the line. To mimic hand-drawn strokes, they quantized the lines by convoluting the pixels with eight-directional line detectors. Resultant quantized lines were decorated with stroke textures. Yang et al. [26] captured salient lines from an image with a flow-based DoG filter, generating noise close to the lines using a blue noise generation algorithm, and by applying a convolution filter to the noise to produce lines similar to pencil strokes.

2.2 Lines of 3D models

Dooley and Cohen [5] pioneered a 3D surface illustrator with a series of lines, whose normal vectors were perpendicular to viewing directions, denoted as silhouettes and contours. Lake et al. [13] stylized silhouette edges by applying stroke textures to produce a stylization rendering scheme for real-time animation. Hata et al. [9] presented a suggestive contour algorithm by extending the contour concept, estimating their likelihood by perturbing the viewing direction and facilitating the contour with potential lines. This method conveyed the shape of 3D models with improved details. Lee et al. [14] applied pencil stroke textures to the exterior lines of 3D meshes. A single pencil texture was randomly connected to mimic various exterior human strokes drawn. Cole et al. [3] studied how artists draw lines to convey specific 3D shapes. According to this study, 75% of artists draw overlapping lines. Other lines overlap large gradients of the image intensity. Through this study, algorithms predicted where artist will draw lines for new scenes. Recently, Kim et al. [12] presented a stereoscopic rendering of 3D shapes using line-drawings. Stereo-coherent lines extracted from 3D shapes were rendered with stroke textures to present stereoscopic views of the shapes.

2.3 Importance-based representation

DeCarlo and Santella [4] employed an eye-tracking system to pinpoint the region where users pay the most attention. Then, they controlled the abstraction details per the higher-magnitude areas of attention. For this purpose, they constructed a hierarchical image structure and presented a scheme that controls the representation of a scene according to importance. However, because importance depends on users' eye-tracking, the scheme that estimates the importance of an image by analyzing its information has not been studied. Guo et al. [8] presented a primal sketch graph that selects important lines from the various lines extracted from an image. Using the primal sketch graph, they conveyed salient shape information of objects embedded within a scene. Zeng et al. [30] improved the primal sketch graph to present an image-parsing technique that organizes a hierarchical structure from an image, determining the size of stroke textures for producing an oil-painting effect. Because the stroke size adapts to the importance of an image, their method presented visually pleasing oil-painting results. In their work, however, importance information is used only to preserve details of the input image. Unlike rough drawings, their scheme does not emphasize key details and ignores unwanted details. Hata et al. [9] estimated saliency to detect distinguished regions from an image, using it to express the region using pencil strokes. Their scheme mimics artistic techniques that ignore unimportant regions, leaving them blank. Their use of saliency information is very similar to our technique. However, they only aimed to remove unwanted regions, whereas we can both emphasize important regions and ignore unimportant regions. Recently, Spicker et al. [22] presented a scheme that emphasizes close coherent lines using thick strokes, ignoring far lines that use thin strokes. Line distance was estimated based on depth computation for the 3D scene. Importance, based on depth information, mimicked atmospheric perspective drawing techniques of fine art. Their scheme, however, was limited: the importance of similar depth ranges is not estimated.

3 Outline of the algorithm

The outline of our algorithm is presented in Fig. 2. We separate our process for rough drawing into importance-embedded feature line extraction and line stylization. For line extraction, we extract feature lines using a flow-based DoG filter [11]. Extracted lines are vectorized and corrected using a line shift process. After correction, we estimate line importance using gradient and saliency. As a result, we produce a series of importance-embedded feature lines. For stylization of the extracted lines, we sample stroke texture segments from real stroke textures and apply them along the lines.

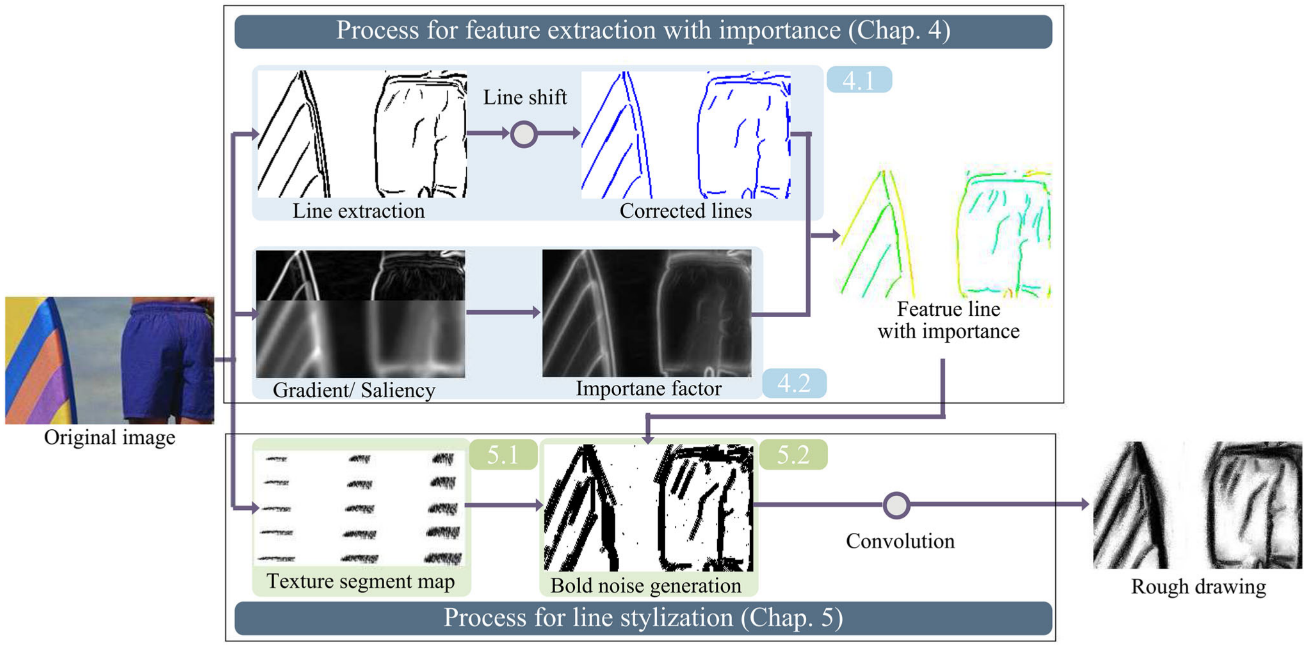


Fig. 2 An overview of the algorithm

Then, we apply a convolution algorithm to produce long stroke marks along the feature lines.

4 Line extraction

Lines depict the important features and contours of objects in a scene. Many researchers present various line extraction schemes in computer graphics and image processing. However, only several efforts evaluate line importance. For example, Spicker et al. [22] computed a depth map from an image and assigned depth information to the lines.

4.1 Extracting correct lines

4.1.1 Extracting lines

We apply the flow-based DoG scheme to extract lines from an input image. Because the DoG filter produces a line represented as a set of pixels, we apply a vectorization scheme [26], calculating a set of skeleton curves that pass through the center of each extracted line. These curves are comprised of sets of serial points.

Unfortunately, the extracted DoG-based feature lines migrate toward the darker region of the exact contour. We illustrate the background of this mismatch in Fig. 3a. The gradient, $|G|$, which follows Gaussian smoothing, is estimated by a formula, $|G| = \sqrt{G_x^2 + G_y^2}$. DoG, D, is computed as $D(\hat{x}, \sigma, k, I) = G(\hat{x}, \sigma, l) - G(\hat{x}, k\sigma, l)$, where $G(\hat{x}, \sigma, I)$ is a linear integral. Because the pixels whose D is below a threshold are extracted as feature lines, the center line (blue line) passing through these feature lines does not match the center of the gradient (red line). See Fig. 3a. This incorrect extraction evokes a mismatch from the other estimates, such as gradient and saliency, which are located exactly on the contour of an object. See Fig. 3b.

4.1.2 Correcting lines

To remedy this problem, we correct the vectorized lines to simultaneously preserve smoothness and correctness. There are many optimization schemes to resolve this problem [29]. Because the extracted line is smooth, the vectorized line that passes through the center of the line preserves smoothness. Therefore, using the vectorized line as an initial model enables us to concentrate on increasing the correctness. We suggest a simple scheme that shifts the line along its orthogonal direction to increase the correctness.

At each point, $\mathbf{p}^i$ on a vectorized line, $\mathbf{C}$, which is $\mathbf{C} = (\mathbf{p}^0, \ldots, \mathbf{p}^n)$, we sample k candidate points along the orthogonal direction to the vectorized line, $\mathbf{C} = (\mathbf{p}^i_0, \ldots, \mathbf{p}^i_k)$, where $\mathbf{p}^i_{k/2}$ is $\mathbf{p}^i$.

$$\mathbf{p}^i_j = \mathbf{p}^i + s\mathbf{n},$$

where $s = \delta(jk/2)/k$, $\mathbf{n}$ is the normal direction to the line at $\mathbf{p}^i$, and $0 \le j \le k$. δ is the lines width. See Fig. 4a, b. We define the correctness of a point, $\mathbf{p}^i_j$, as the sum of gradient, $G(\mathbf{p}^i_j)$, and saliency, $S(\mathbf{p}^i_j)$. Then, the following formula estimates the correctness, $E(\mathbf{C})$, of the line, $\mathbf{C}$:

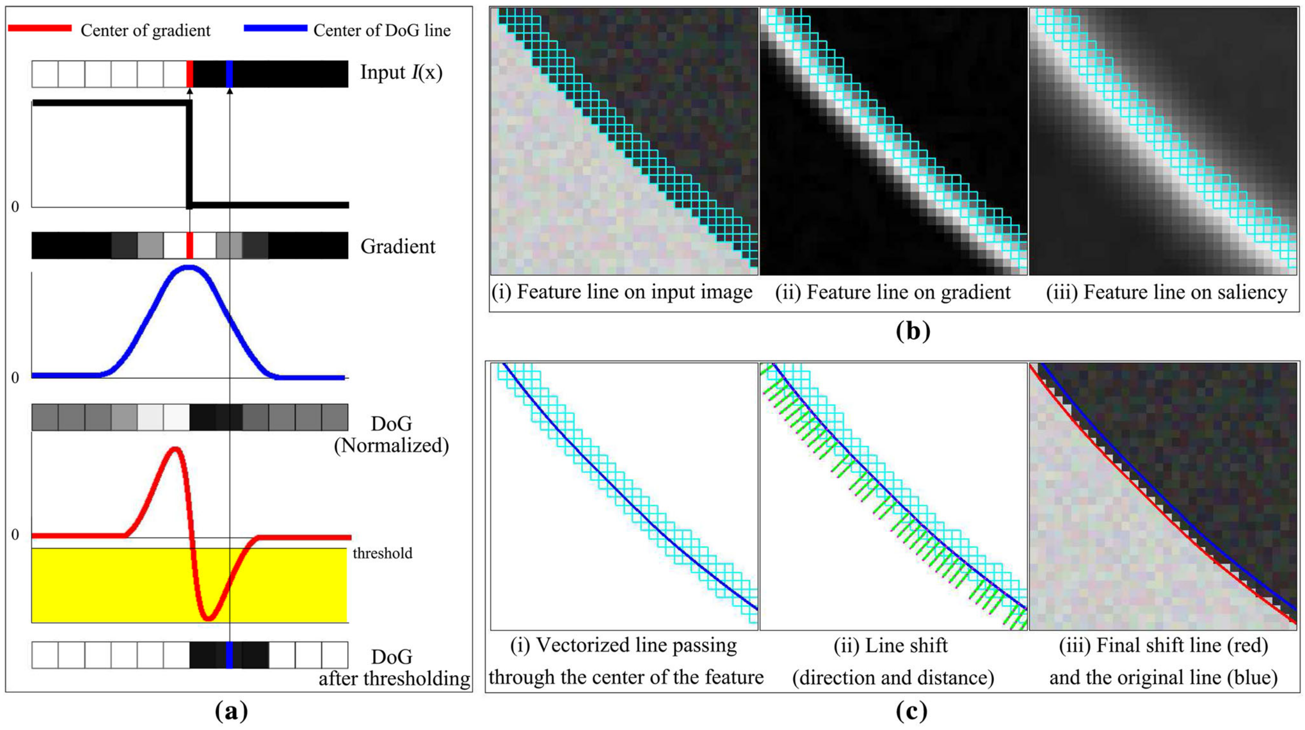

Center of gradient
Center of DoG line
Input $I(x)$
0
Gradient
0
DoG (Normalized)
0
threshold
DoG after thresholding
(a)
(i) Feature line on input image
(ii) Feature line on gradient
(iii) Feature line on saliency
(b)
(i) Vectorized line passing through the center of the feature
(ii) Line shift (direction and distance)
(iii) Final shift line (red) and the original line (blue)
(c)

Fig. 3 Line shifting. **a** Incorrect feature line with the exact contour, salience, and gradient. **b** The results of line shift process

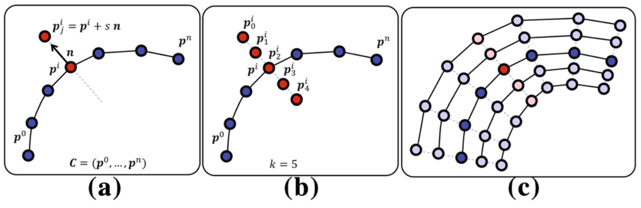


Fig. 4 The process of correction: **a** computing $\mathbf{p}^i_j$, **b** sampling five candidate points, **c** five candidate vectorized lines. The line whose $E(\mathbf{C})$ is maximum is then chosen

$$E(\mathbf{C}) = \sum_{i=0}^{n} (G(\mathbf{p}^i) + S(\mathbf{p}^i)).$$

Because we have $(k+1)$ candidate curves, we find a jth line whose $E(\mathbf{C}_j)$ is maximum for $0 \le j \le k$. See Fig. 4c. The result of line shifting is illustrated in Fig. 3c.

```
dart_throwing ( )
{
    sample_count = 0;
    sample_list = EMPTY;
    while ( n < max_sample ) {
        reject_count = 0;
        do {
            generate a new sample at x;
            if ( reject_count > reject_threshold ) {
                rx = k * rx;
                reject_count = 0;
            }
            reject_count++;
        } while ( dist(x) < rx );
        add x to a sample_list;
        sample_count++;
    }
    return sample_list;
}
```

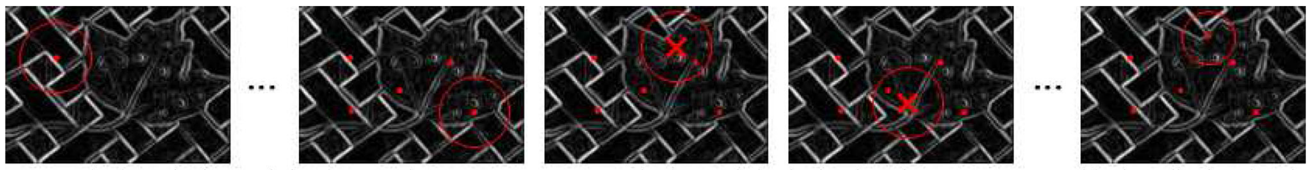
Fig. 5 Pseudocode for sampling using dart-throwing

4.2 Measuring importance

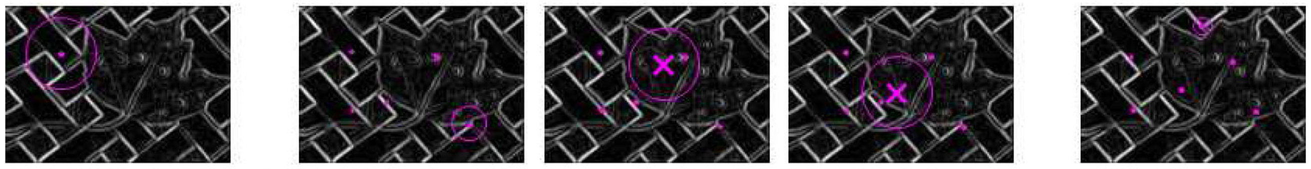
We define line importance as the magnitude of attention it attracts. People tend to pay attention to distinguished parts of any scene. In image processing, saliency, which is estimated from Luv color space, measures how distinguishable a pixel is from others in an image. High saliency means that the pixel is very distinguishable, whereas low saliency means that the pixel is similar to the others. Among various saliency-estimating schemes, we apply a content-based approach [6], which is recognized as one of the most effective.

4.2.1 Stochastic saliency estimation

Goferman et al. [6] constructs uniformly sampled patches and builds a metric to estimate the similarity among patches. Pixel saliency is inversely proportional to the similarities among all

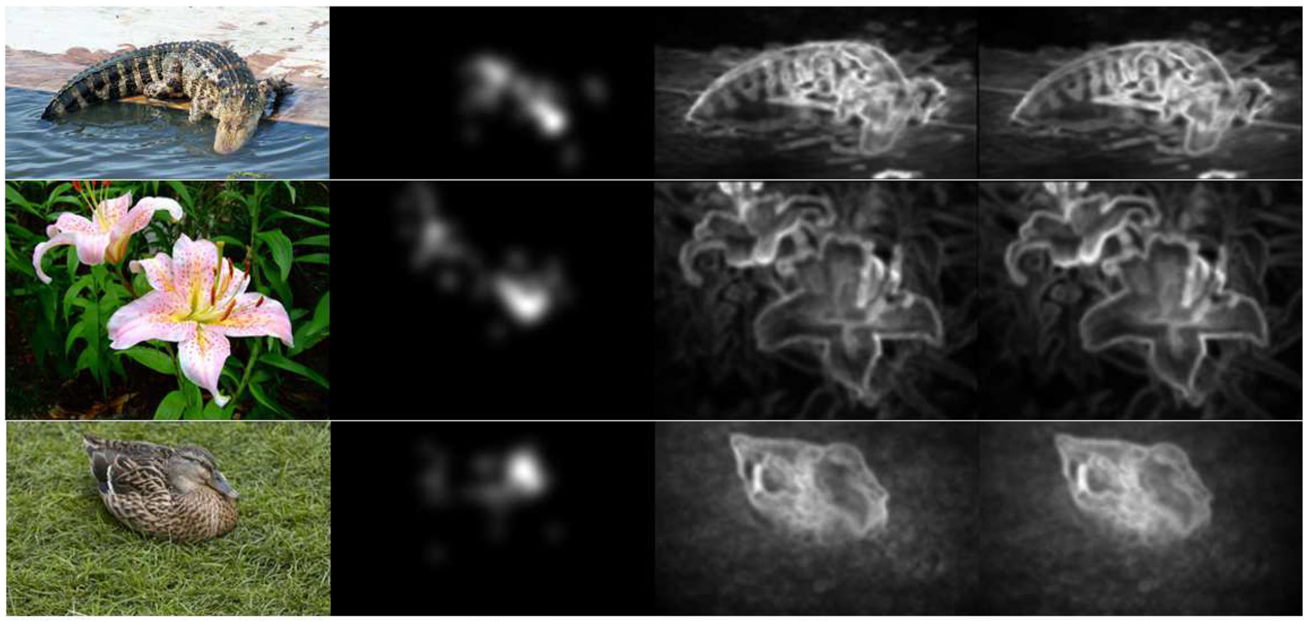

(1-2) Anoise is sampled and accepted (3-4) Sampled noise is rejected (5) The radius is reduced

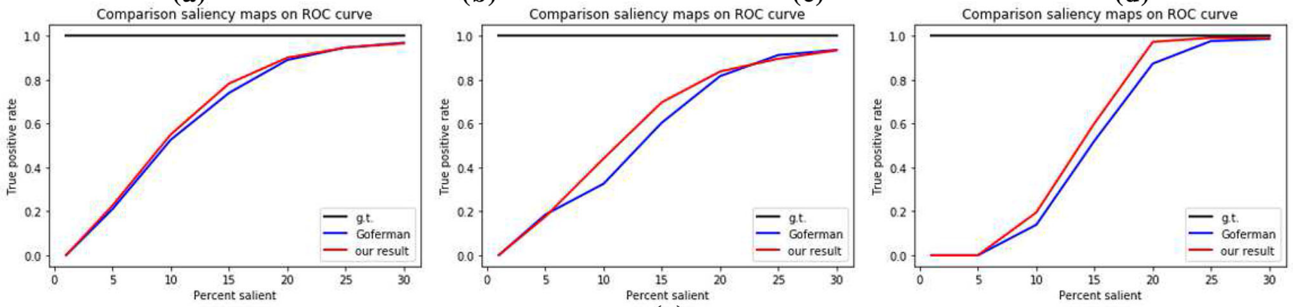
(a)


(1-2) A noise of adaptive radius is sampled and accepted (3-4) Sampled noise is rejected (5) The radius is reduced

(b)

Fig. 6 Comparison of processes: the conventional process, **a** accepts a newly generated sample if there are no other samples in its radius, (`rx`). If other sample exist, it is rejected and a new position is sampled. If the number of rejections exceeds a given threshold, the radius is then reduced by multiplying $k < 1.0$. In our process, **b** the radius of a newly generated sample inversely corresponds to the gradients magnitude at the given position


(a) **(b)** **(c)** **(d)**



(e)

Fig. 7 Comparison of the saliency from Goferman et al. [6] and ours

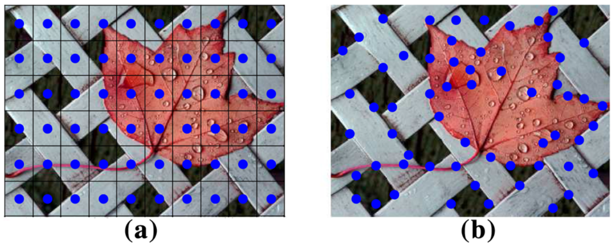


Fig. 8 Comparison of the approaches: Goferman et al.'s [6] and ours

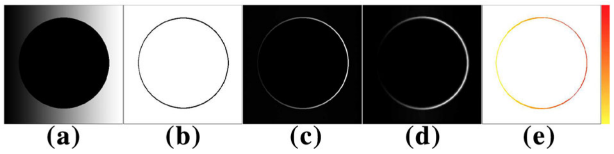


Fig. 9 Importance measuring process. **a** Input image, **b** Coherent line, **c** Saliency, **d** Gradient, and **e** Importance measure

Table 1 Comparison of performance between Gofermans et al. [6] and ours (in second)

	Top	Middle	Bottom
Goferman et al. [6]	80.6	81.5	71.2
Our scheme	16.5	16.6	15.4

patches. This scheme produces improved results compared to existing schemes, but it suffers from the computational inefficiency of comparing pairwise patches. Thus, this scheme is limited in practically resolving target images.

We propose a stochastic approach to sample pixels according to their importance. This approach produces both a dense sample distribution in more distinguished regions of an image, and a sparse distribution in less distinguished regions. We employ the dart-throwing algorithm to preserve the Poisson distribution of any prefix of the generated noise sequences [18]. The dart-throwing algorithm works by generating a sample at a random position. If no other samples exist in a given radius, the sample is accepted. Otherwise, it is rejected. During the sample generation process, a sample can be rejected more often than a predefined threshold (`reject_threshold`). In this case, radius is reduced by multiplying $k < 1.0$ to minimize the probability of rejection. We summarize this process in Fig. 5. In the code, `dist(x)` denotes the minimum distance from $\mathbf{x}$ to all sampled positions, computing and maintaining them using a distance map.

The identical radius, (`rx`), produces evenly distributed samples in the original dart-throwing algorithm [18]. To control the density of samples per distinguished region, we modify the algorithm by assigning a radius to the distinguished area. For this purpose, we employ the gradient as an initial guess of the saliency. We control the radius inversely proportional to the gradient of the candidate pixel. Our $r(\mathbf{x})$ is defined as follows:

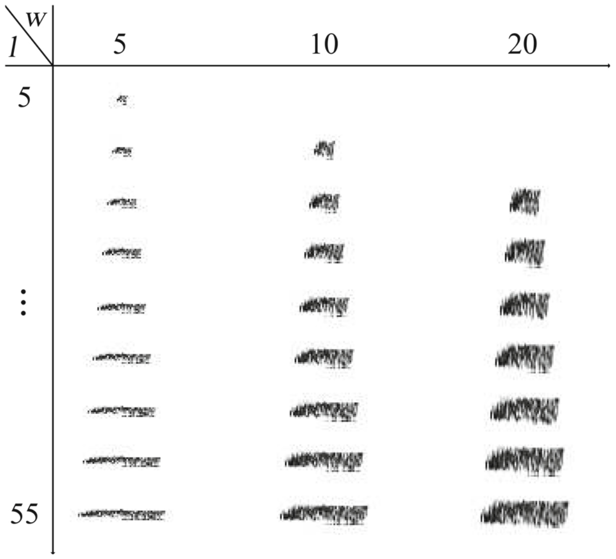


Fig. 10 Texture segments of various widths and lengths (units are pixels)

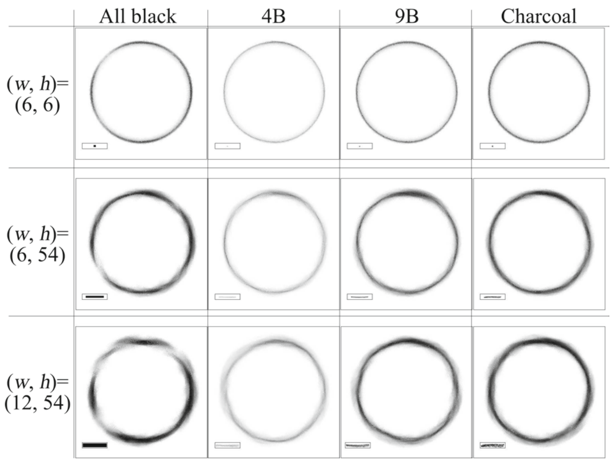


Fig. 11 Stroke variations by texture segments from various media. In the small box, we present the texture segment used to produce the lines: a black bold line segment, a 4B pencil, a 9B pencil, and a charcoal pencil (from left to right)

$$r(\mathbf{x}) = \frac{r0}{G(\mathbf{x}) + \epsilon},$$

where $r0$ denotes the initial radius, $G(\mathbf{x})$ is the gradient at $\mathbf{x}$, and ϵ is a small value, avoiding a divide-by-zero dilemma. We compare the sample generation processes in Fig. 6.

We employ the gradient as an initial guess of the saliency. The strategy assumes the density of the sampled pixels corresponds to their gradients, allowing both correctness and efficiency in saliency estimation. Figure 7 compares the saliency estimated from [6] and our stochastic scheme. We compare the distributions from uniform sampling and adaptive sampling in Fig. 8. Although some samples are located in

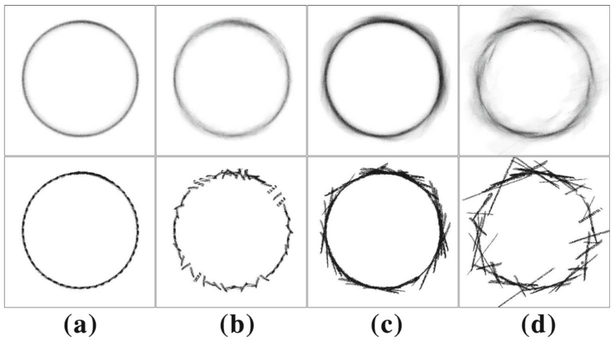


Fig. 12 Stylization with perturbation. **a** Non-perturbation on orientation and size, **b** perturbation on orientation, **c** perturbation on size, and **d** perturbation on orientation and size

non-salient regions of Fig. 8b, they are sparsely distributed, whereas the samples in salient region are densely distributed.

To prove the correctness of our scheme, we employ the ground truth data from Judds et al. work [10].

4.2.2 Importance estimation

However, because saliency is estimated based on color, the salient information based on intensity may be ignored. Therefore, we also employ gradient as an assistant measure of importance. The following formula estimates importance term:

$$Imp(\mathbf{x}) = \kappa \tilde{S}(\mathbf{x}) + (1-\kappa)\tilde{G}(\mathbf{x}),$$

where $\tilde{S}(\mathbf{x})$ and $\tilde{G}(\mathbf{x})$ are the normalized saliency and gradient, respectively. Users can determine the weight of $\tilde{S}(\mathbf{x})$ and $\tilde{G}(\mathbf{x})$. In our approach, we set κ to 0.5. Fig. 9 illustrates our process for measuring importance of a feature line. In Fig. 9a, a black circle is suggested, and the background shows a graduation from black to white. Even though the circles boundary is clearly extracted as a line in Fig. 9b, we see that the right part of the line shows a clearer boundary, and the left part shows an unclear boundary. This difference is not shown in the line in Fig. 9b. By combining gradient and saliency in Fig. 9c, e, we can visualize the importance factor in Fig. 9e, which distinguishes the important part of the line from the less important parts (Table 1).

5 Line stylization

To stylize lines, many researchers employ sampled stroke textures [1,13,14] or execute convolution filtering [17,26]. Stroke texture approaches present visually pleasing line-drawings, but they suffer from inconvenient shape control. The convolution approach allows users to create various line shapes, but they cannot mimic hand-drawn strokes. In our approach, we express line strokes by convoluting stroke texture segments distributed on the pixels near target lines.

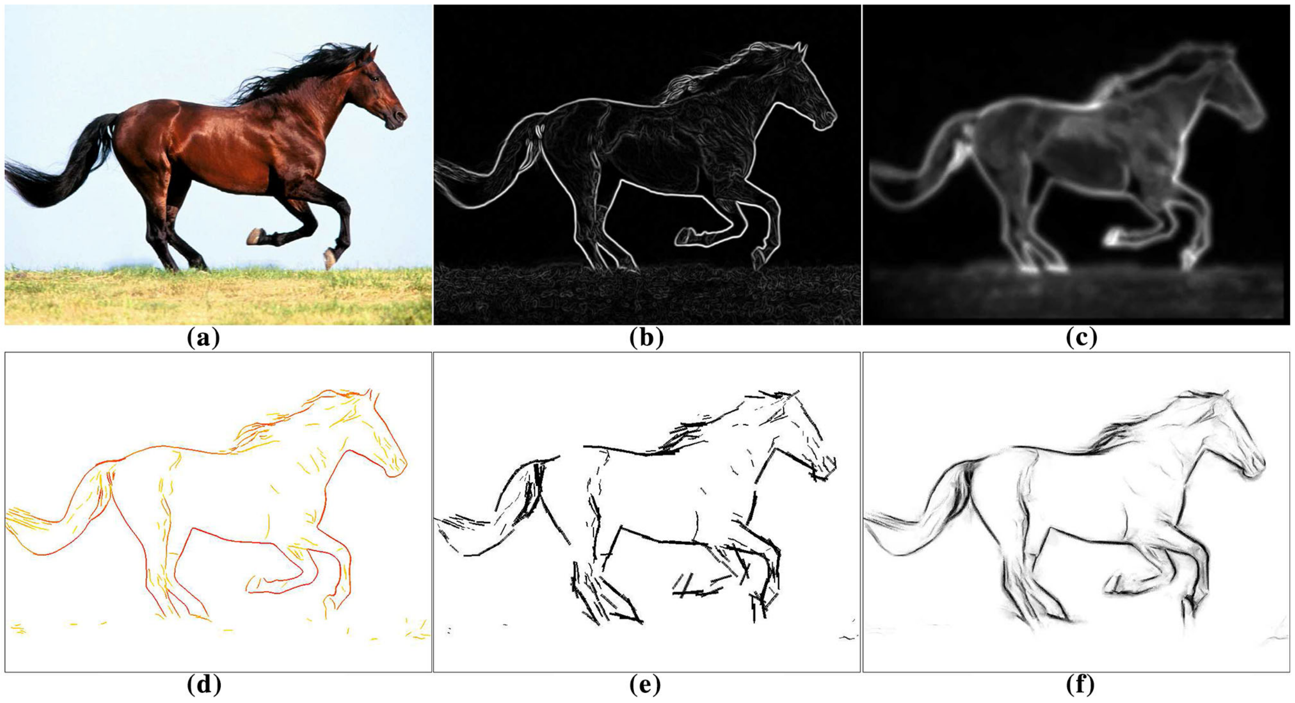


Fig. 13 The milestones of our framework. The importance in **d** is estimated from the gradient in **b** and the saliency in **c**. Noise is generated with length, (1–70), and width, (1–7)

Table 2 Parameters for the figures in our paper

		Length	Width	Perturbation
Figure 1	Thin	1	1	0
	Thick	50	5	0
	Variable	1–100	1–7	0
Figure 14	Girl	1–60	1–4	0–50
	Woman	1–70	1–7	0
	Dancer	1–60	1–4	50
	City	1–80	1–3	2
	Boat	1–100	1–5	2
Figure 15	Thin	15	1	0
	Thick	35	3	0
	Variable	0–70	1–7	0
Figure 16	Thin	15	1	0
	Thick	35	5	0
	Variable	15–35	1–5	0
Figure 17	Thin	15	1	0
	Thick	35	3	0
	Variable	1–70	1–7	0

5.1 Building texture segments

We sample a series of strokes drawn by various stick-shaped media such as H pencils, B pencils, and charcoal pencils. Then, we classify stroke textures and segments by width and length. The width varies in (5, 20) and the length in (5, 50), with a resolution of 5. In Fig. 10, we illustrate widths and lengths of texture segments, drawing lines by employing the stroke texture segments in Fig. 11. These texture segments are further processed to various widths and lengths for our stylization.

5.2 Stylizing the strokes

The first step of stylization determines the target media and mimics it by controlling the width and height of the texture segment. To allow variation of stroke patterns for expression, we assign the minimum and maximum parameters. The next step stylizes strokes by perturbing the lengths and widths of texture segments, including the directions along which they are aligned. For low perturbations, our strokes mimic long stroke marks drawn by a single stroke method. Using high perturbations, our strokes mimic multiple short stroke marks drawn by several stroke methods. Various results from the

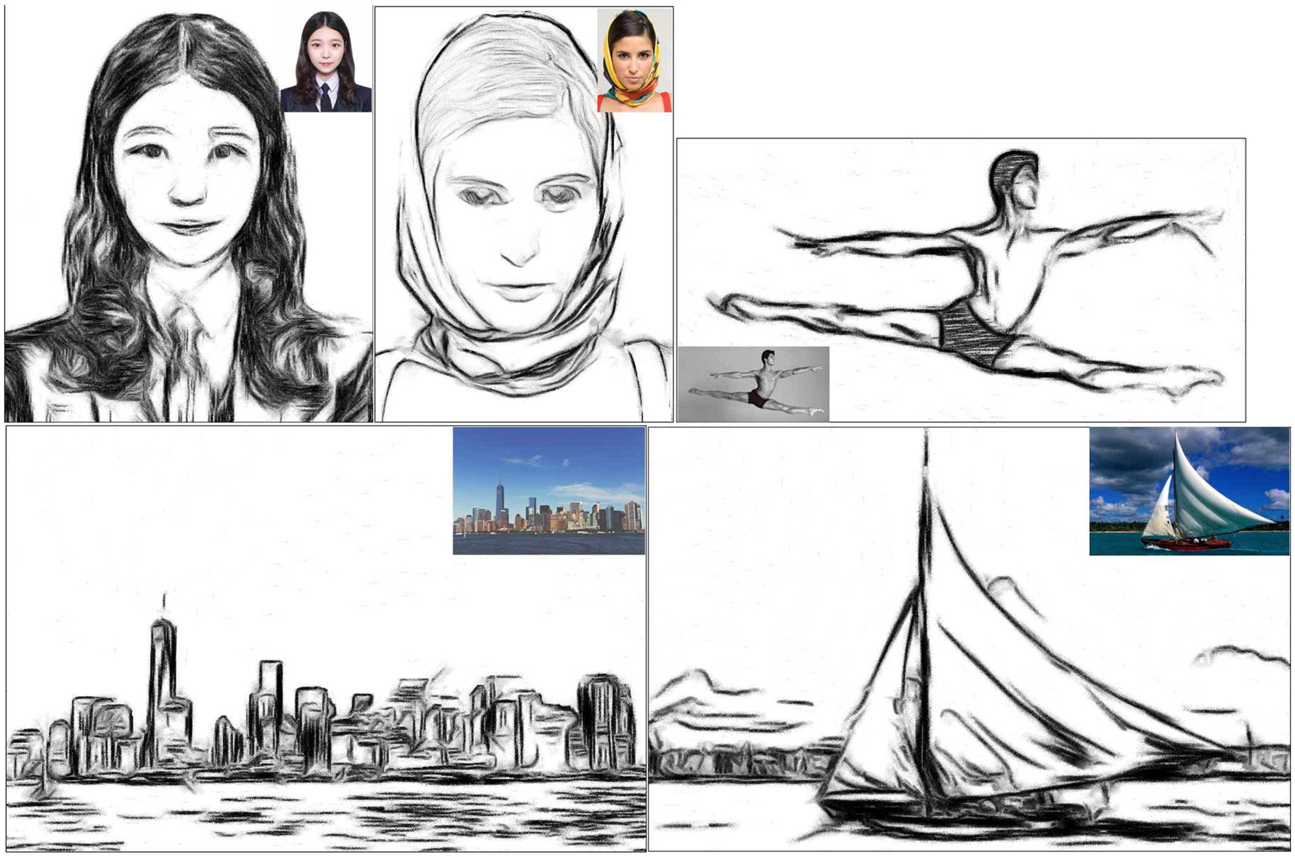


Fig. 14 The results of our algorithm

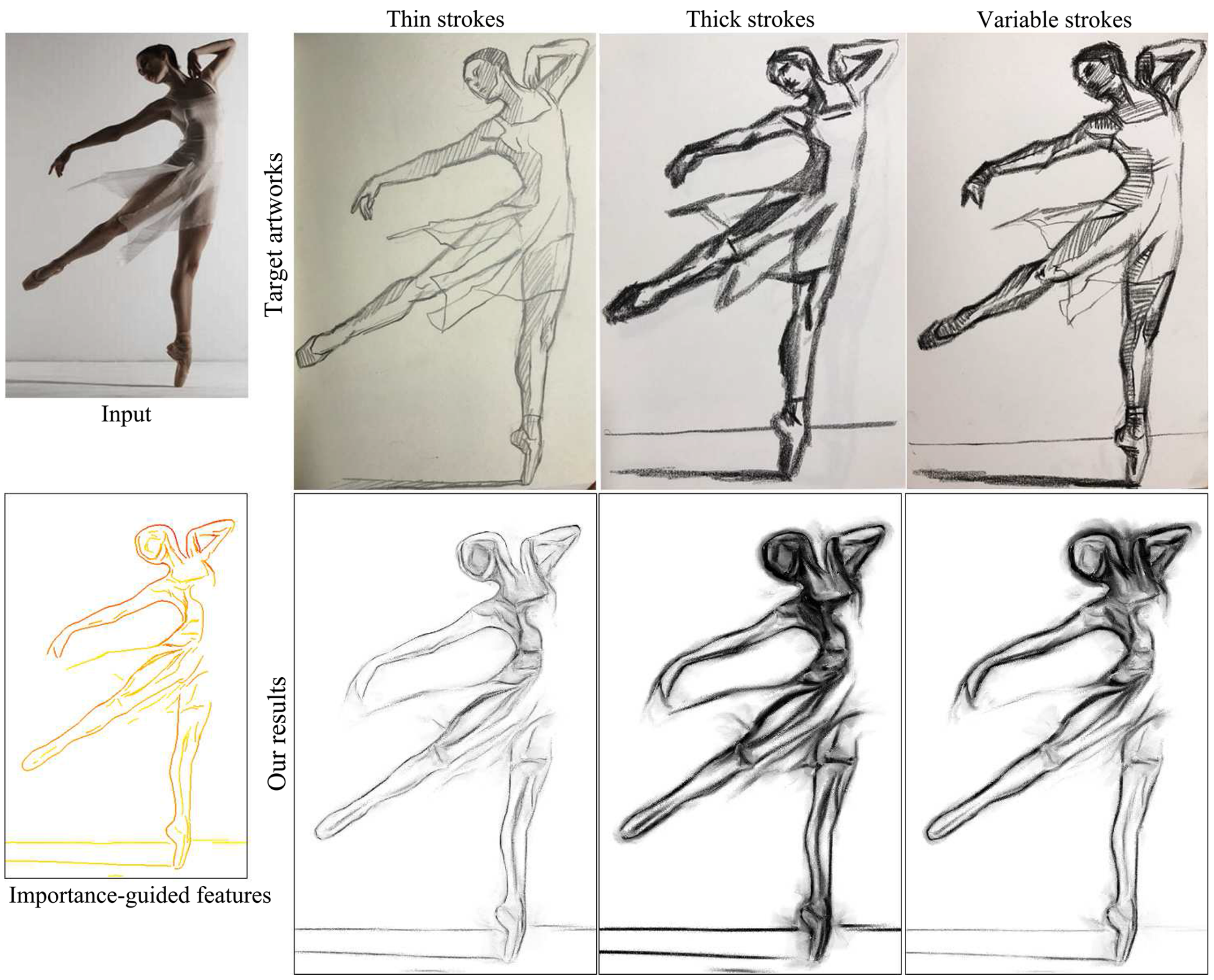


Fig. 15 Comparison to real artwork I: we compare rough dynamic human gesture-drawings of various styled strokes to real artwork. Our importance-guided features are in the left corner

perturbations are illustrated in Fig. 12. Length and width are proportional to importance, and perturbation is determined inversely proportional to importance. The formula for length, (l), width, (w), and perturbation, (p), are shown as follows:

$$l = l_{\min} + Imp(x)(l_{\max} - l_{\min})$$
$$w = w_{\min} + Imp(x)(w_{\max} - w_{\min})$$
$$p = p_{\min} + (1 - Imp(x))(p_{\max} - p_{\min})$$

Pixels underlying the texture segments embed the intensity of the corresponding segment position. These intensities portray noise using a convolution-based approach. Note that the noise deposition becomes very dense as texture segments increase. The dense deposition of noise allows mimicking thick strokes from high-B pencils or charcoal. We apply a line interval convolution-based filter to determine the value of a pixel in the stroke by integrating the noise embedded in the filter kernel.

6 Implementation and results

We implemented the rough drawing framework using a PC with a Pentium i7 processor with 16 GB main memory. We tested our framework on images under various situations to prove that our framework produces visually pleasing rough drawings that mimic real artwork. We also compared our works with existing famous works to show the excellence of our framework. Figure 13 illustrates the milestones of our framework. Table 2 present the parameters we used.

6.1 Experiments

We apply our scheme to images of portrait, gesture, and landscape formats, as shown in Fig. 14. We produce rough drawings using different stroke styles, shown in Figs. 15, 16 and 17. In those figures, we produce B pencil strokes:

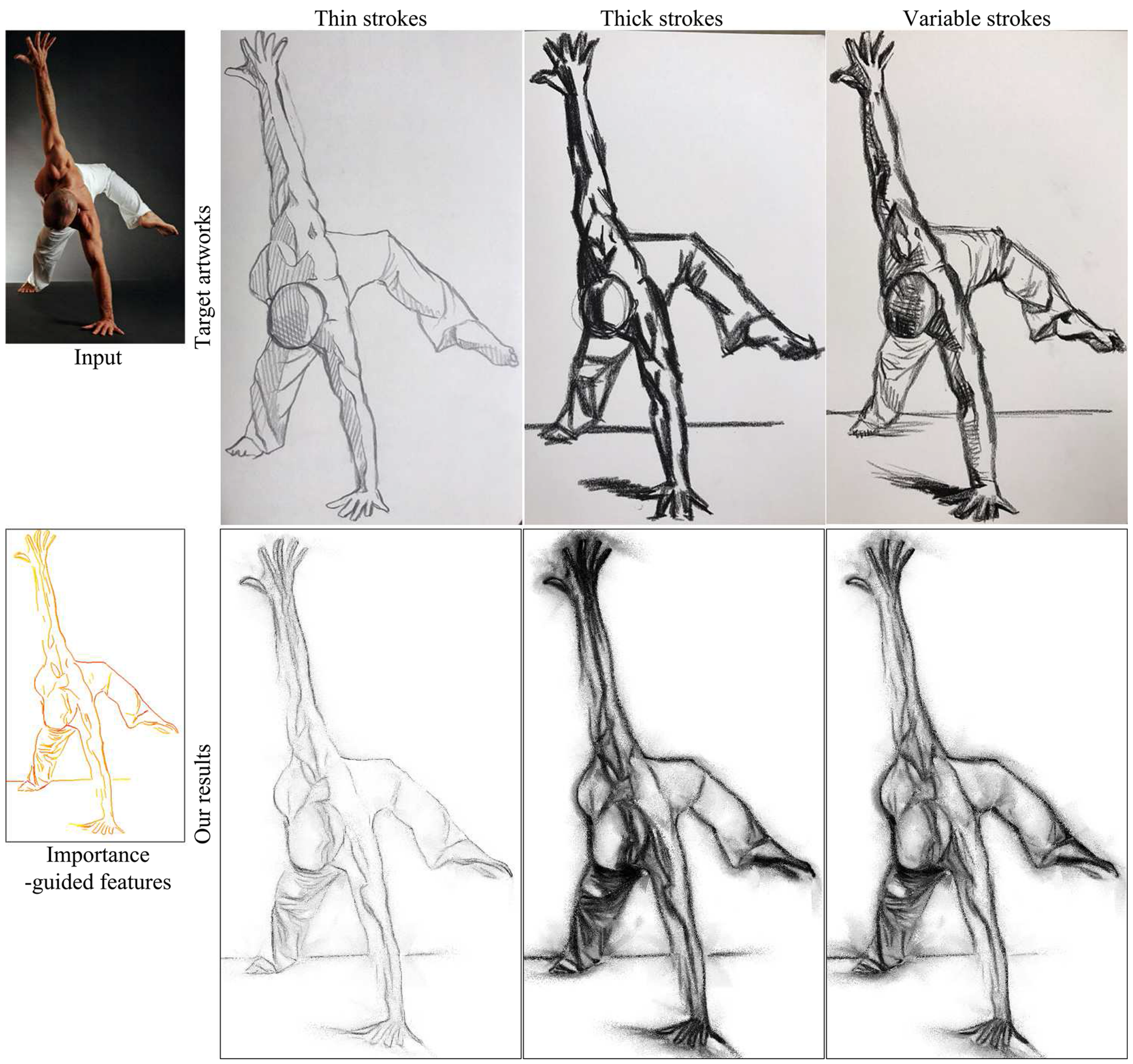


Fig. 16 Comparison with real artwork II: we compare rough dynamic human gesture-drawings of various styled strokes to real artwork

thin, with long and narrow stroke segments; and charcoal strokes: with thicker, longer, and bolder lines and segments. The left and center columns of Figs. 15, 16 and 17 correspond to thin and thick strokes, respectively. To produce variable strokes, we use a mapping to determine the size of the segments from their estimated importance. In Figs. 15 and 16, our importance estimation scheme assigns higher importance to the contours of the body and lower importance to the body details. In Fig. 17, our scheme assigns higher importance to the lines of the house in the center. Therefore, the lines depicting contours are expressed with thicker strokes, and the lines depicting details are expressed with thinner strokes.

6.2 Comparisons

6.2.1 Comparisons with real artworks

As we provide comparisons to real artwork in Fig. 1, we also present comparisons in Figs. 15, 16 and 17, where two-gesture and single-landscape drawings are created and compared. Gesture drawing is a good example of rough drawing, because the body contours are drawn with thick strokes and the body details are drawn with thin strokes. Landscape drawing is another good example of rough drawing, because many artists omit the fine details, but emphasize salient lines to roughly convey the scene.

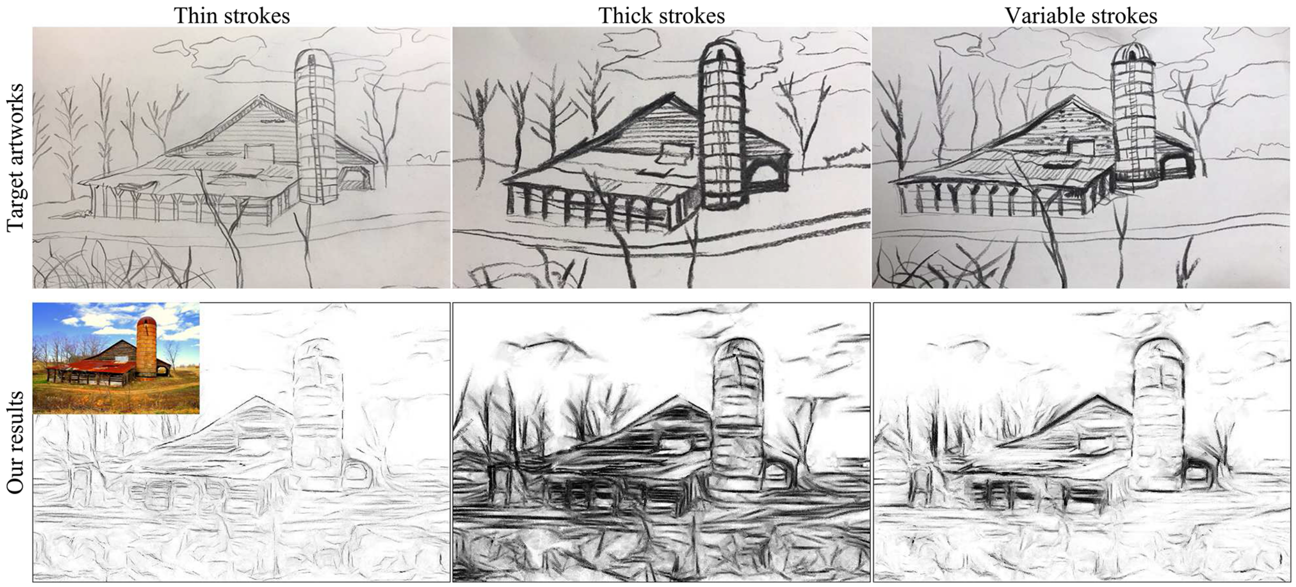


Fig. 17 Comparison with real artwork III: we compare rough landscape drawings of various styled strokes to real artwork

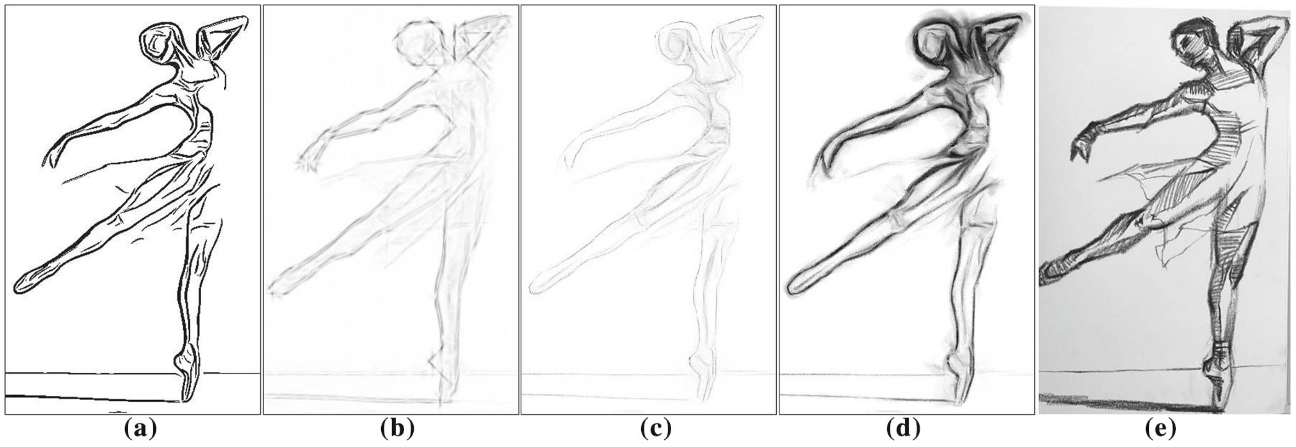


Fig. 18 We compare our results with those from related works

To compare results, we hire professional artists to draw rough drawings with three different strokes: thin, thick, and variable. For each real artwork, we tune the parameters of our system to produce similar rough drawings. As seen in the right column of Figs. 15, 16 and 17, our work resembles the real rough drawings that convey the salient shape of the scene. One limitation of our work is the hatching marks that convey the scenes abstract tone.

6.2.2 Comparisons with existing works

Our target artwork, shown in Fig. 18d, is drawn with adaptive strokes. Body contours are drawn with thick and bold strokes, but the body details are drawn with thin strokes. Existing works, [11,17,26], presenting line-based depictions do not present lines of adaptive styles per the targets importance. The coherent line, [11], in Fig. 18a, does not adapt lines according to their importance. Pencil-mimicking artwork, [17,26], presents styles that differ from Kang et al. [11]. Both employ convolution strategies to produce lines. Because Lu et al. [17] applied the convolution in eight quantized directions, the lines are more linear than those from Yang et al. [26]. The limitation of these works derive from point-sized noise, which is convoluted to produce stroke marks. In our work, we generate noise from stroke texture segments, from which densely deposited noise can be generated. This is required to mimic graphite media using thick and bold strokes, such as with charcoal pencils. By combining this strategy with importance, we can mimic the artwork in Fig. 18e with importance-adaptive strokes.

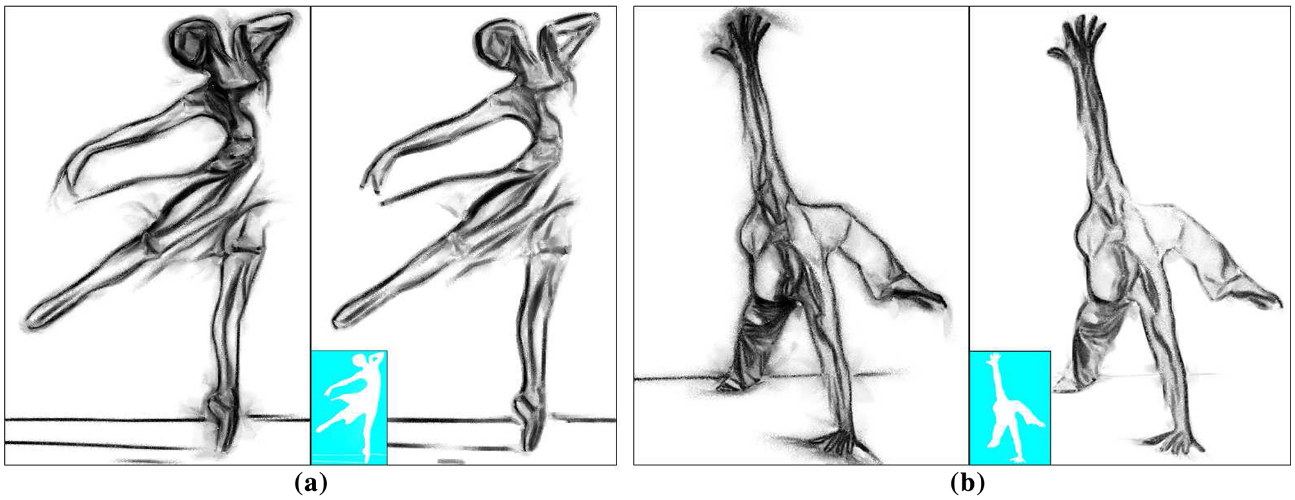


Fig. 19 Avoiding blurry artifacts by user-guided segmentation: The convolution is not executed in the background (cyan pixels). **a** and **b** are from the result of the thick strokes in Figs. 15 and 16, respectively

6.3 Discussion

We produce rough drawings using different stroke styles. We analyze the importance using gradient and saliency to mimic artists drawing styles. Thickness and boldness of the strokes are simulated by controlling the stroke texture segments. Even though we present an improved framework for rough drawings, our framework still possesses the following limitations.

1. The DoG-based approach sometimes loses important details. For example, the DoG approach ignores detail extraction from the ballerinas face in Fig. 17. Other feature extraction schemes, such as convolutional neural network, present interesting alternatives.
2. We focus on line-drawing approach for rough drawing, but we cannot present a scheme to convey the tone. In the target artworks, the darker regions are expressed by a series of hatching patterns with various widths and spacing. Since our rough drawing approach is based on the salient features, we cannot produce rough drawings on the regions where salient features are not detected. In our future work, we analyze the schemes that produce hatching on tonal region [23] and build a scheme that produce the tonal expression of the target artworks.
3. The blurry artifacts in Figs. 15 and 16 are another limitation of our framework. This blurry marks come from the convolution that produces thick and bold stroke patterns from the stroke textures deposited along the feature lines. Since our convolution integrates the values deposited in the pixels that belong to its convolution boundary, the convolution on the pixels outside an object integrates the value stored on the pixels in the features. Therefore, the blurry patterns outside a figure are produced. We suggest a simple scheme to avoid this blurry artifacts by employing a user-controlled segmentation that separates the object from its background. The convolution is not executed outside the object, which reduces most of blurry artifacts (See Fig. 19).

7 Conclusion and future work

We have presented a framework that produces rough drawings by extracting lines with importance estimation and by applying stroke texture segments. Lines were extracted using a DoG scheme, and importance was estimated using gradient and saliency. To express lines of various styles, we applied a convolution algorithm for the stroke texture segments attached to the lines of an image. To extend and improve our work, we plan to apply advanced feature-extraction schemes based on deep learning. We will also employ machine-learning algorithms to analyze line-drawing information.

Acknowledgements This research was supported by the grant NRF-2017R1D1A1B03034137 and NRF-2015R1D1A1A01061415 from National Research Foundation (NRF) of Korea.

References

1. AlMeraj, Z., Wyvill, B., Isenberg, T., Gooch, A., Richard, G.: Automatically mimicking unique hand-drawn pencil lines. Comput. Graph. **33**(4), 496–508 (2009)
2. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intel **8**(6), 679–698 (1986)
3. Cole, F., Golovinskiy, A., Limpaecher, A., Barros, H., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S.: Where do people draw lines? ACM Trans. Graph. **27**(3), 88 (2008)

4. DeCarlo, D., Santella, A.: Stylization and abstraction of photographs. Proc. Siggraph **2002**, 769–776 (2002)
5. Dooley, D., Cohen, M.: Automatic illustration of 3d geometric models: surfaces. IEEE Comput. Graph. App. **13**(2), 307–314 (1990)
6. Goferman, S., Zelnik-Manor, L., Tal, A.: Context-aware saliency detection. IEEE Trans. Pattern Anal. Mach. Intel **34**(10), 1915–1926 (2012)
7. Gooch, B., Reinhard, E., Googh, A.: Human facial illustrations: creation and psychophysical evaluation. ACM Trans. Graph. **23**(1), 27–44 (2004)
8. Guo, C., Zhu, S.C., Wu, Y.N.: Primal sketch: integrating texture and structure. J. Comput. Vis. Imag. Under **106**(1), 5–19 (2007)
9. Hata, M., Toyoura, M., Mao, X.: Automatic generation of accentuated pencil drawing with saliency map and LIC. Vis. Comput. **28**(6–8), 657–668 (2012)
10. Judd, T., Ehinger, K., Durand, F., Torralba, A.: Learning to predict where humans look. Proc. ICCV **2009**, 2106–2113 (2009)
11. Kang, H., Lee, S., Chui, C.: Coherent line drawing. Proc. NPAR **2007**, 43–50 (2007)
12. Kim, Y., Lee, Y., Kang, H., Lee, S.: Stereoscopic 3d line drawing. ACM Trans. Graph. **32**(4), 57 (2013)
13. Lake, A., Marshall, C., Harris, M., Blackstein, M.: Stylized rendering techniques for scalable real-time 3d animation. Proc. NPAR **00**, 13–20 (2000)
14. Lee, H., Kwon, S., Lee, S.: Real-time pencil rendering. Proc. NPAR **06**, 37–45 (2006)
15. Li, C., Liu, X., Wong, T.T.: Deep extraction of manga structural lines. ACM Trans. Graph. **36**(4), 117 (2017)
16. Litwinowicz, P.: Processing images and video for an impressionist effect. Proc. Siggraph **97**, 407–414 (1997)
17. Lu, C., Xu, L., Jia, J.: Combining sketch and tone for pencil drawing production. Proc. NPAR **2012**, 65–73 (2012)
18. McCool, M., Fiume, E.: Hierarchical poisson disk sampling distributions. Proc. Graph. Interface **92**, 94–105 (1992)
19. Papari, G., Petkov, N.: Edge and line oriented contour detection: state of the art. Imag. Vis. Comput. **29**(2), 79–103 (2011)
20. Salisbury, M., Anderson, S., Barzel, R., Salesin, D.: Interactive pen-and-ink illustration. In: Proceedings of the Siggraph, vol. 94, pp. 101–108 (1994)
21. Son, M., Kang, H., Lee, Y., Lee, S.: Abstract line drawings from 2d images. Proc. Pac. Graph. **2007**, 333–342 (2007)
22. Spicker, M., Kratt, J., Arellano, D., Deussen, O.: Depth-aware coherent line drawings. In: Proceedings of Sigraph Asia Technical Briefs 2015, p. 1 (2015)
23. Suarez, J., Belhadj, F., Boyer, V.: Real-time 3d rendering with hatching. Vis. Comput. **33**(10), 1319–1334 (2017)
24. Winnemoeller, H.: Xdog: advanced image stylization with extended difference-of-Gaussians. Proc. NPAR **2011**, 147–156 (2011)
25. Winnemoeller, H., Olsen, S., Gooch, B.: Real-time video abstraction. ACM Trans. Graph. **25**(3), 1221–1226 (2006)
26. Yang, H., Kwon, Y., Min, K.: A stylized approach for pencil drawing from photographs. Comput. Graph. Forum **31**(4), 1471–1480 (2012)
27. Yang, H., Min, K.: Feature-guided convolution for pencil rendering. KSII Trans. Internet Inf. Syst. **5**(7), 1311–1328 (2011)
28. Yang, H., Min, K.: A multi-layered framework for color pastel painting. KSII Trans. Internet Inf. Syst. **11**(6), 3143–3165 (2017)
29. Ye, C., Zheng, Y.: A survey on image segmentation using geometric active contour models. Proc. ICEICE **2012**, 233–236 (2012)
30. Zeng, K., Zhao, M., Xiong, C., Zhu, S.: From image parsing to painterly rendering. ACM Trans. Graph. **29**(1), 2:1–2:11 (2009)

Heekyung Yang is a Ph. D candidate at Department of Computer Science, Graduate School, Sangmyung University. She got her B.S. at Division of Digital Media, Sangmyunv University, and her M.S. at Department of Computer Science, Graduate School, Sangmyung University. Her research interest includes computer graphics, image processing and machine learning. She especially studies non-photorealistic rendering techniques that produce various artistic effects.

Kyungha Min is a professor at Department of Computer Science, Sangmyung Univerisity. He got his Ph.D. at Department of Computer Science and Engineering, POSTECH. He got his B.S. at Department of Computer, KAIST and his M.S. at Department of Computer Science and Engineering, POSTECH. His research interest includes computer graphics and machine learning.