Legible Compact Calligrams

Changqing Zou, Junjie Cao, Warunika Ranaweera, Ibraheem Alhashim, Ping Tan, Alla Sheffer, Hao Zhang

SIGGRAPH 2016

Introduction

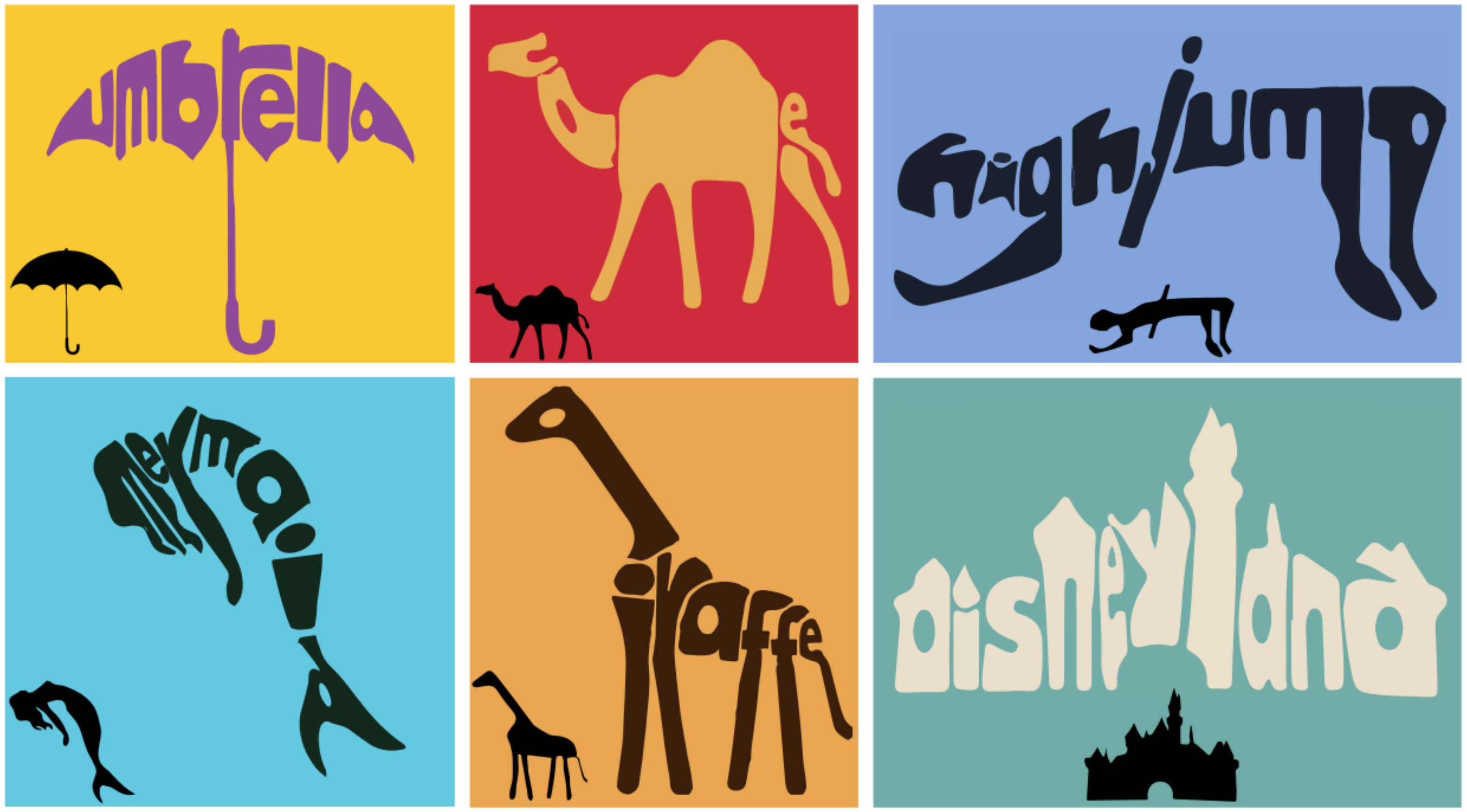
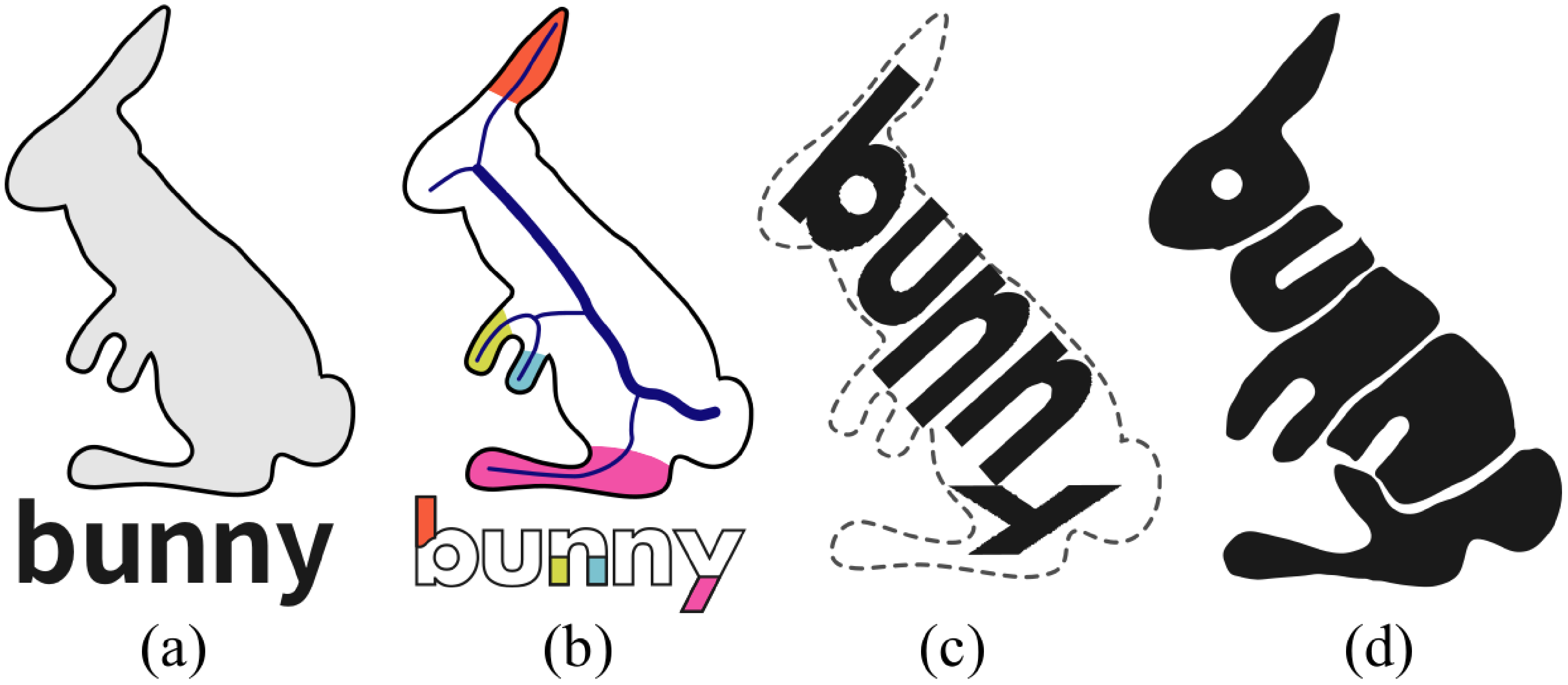


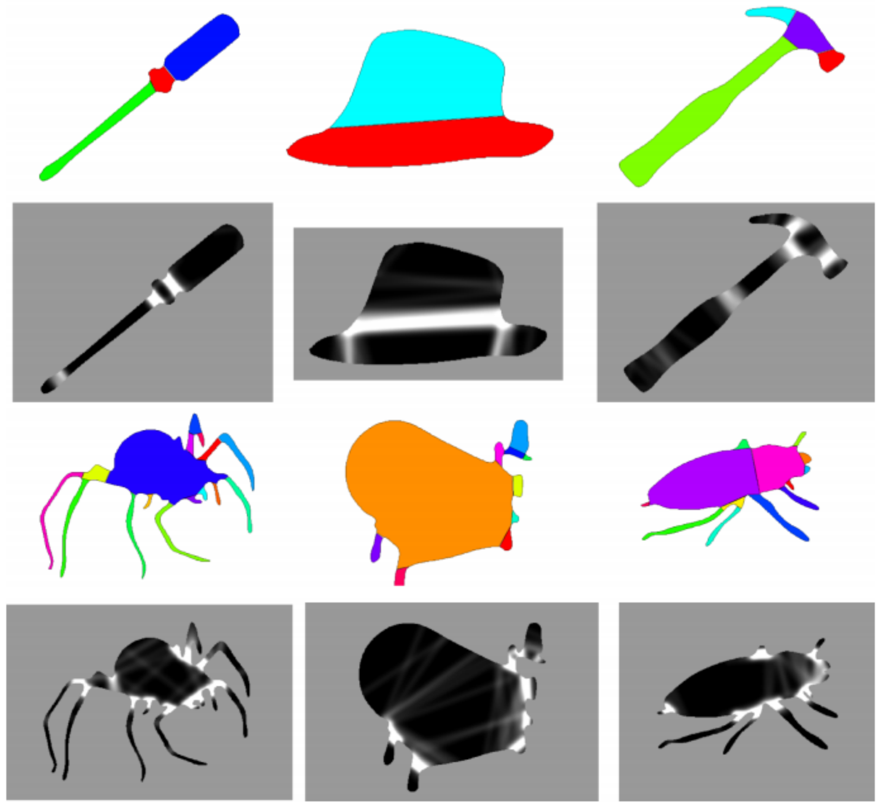
Figure 1: *A few compact calligrams generated by our algorithm fully automatically. Input images are shown as insets.*

Overview



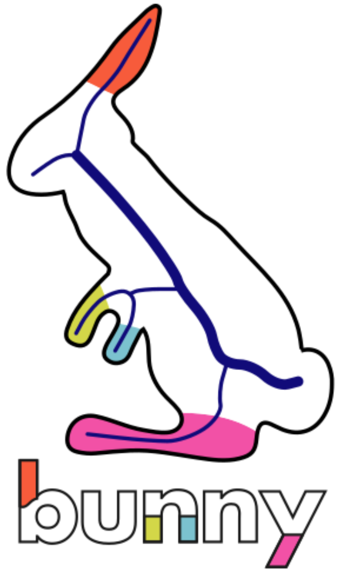
Method

- Decomposition
- A Computational Model of the Short-Cut Rule for 2D Shape Decomposition



Method

- Protrusion detection
- $Area_S$: input shape area
  N : letter count
- Filter out protrusion area > $\frac{2 \cdot Area_S}{N}$ or < $\frac{0.2 \cdot Area_S}{N}$
- Filter out protrusion bounding box when $\frac{W}{H} > \frac{3}{4}$



Method

- Path computation
- Matlab thinning tool

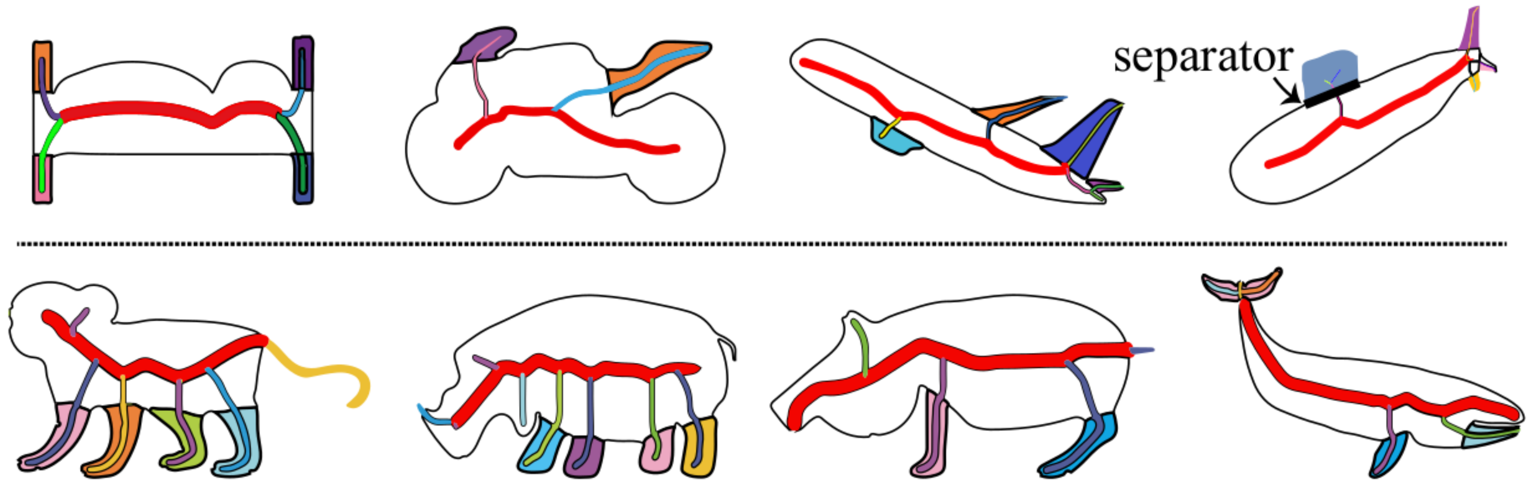
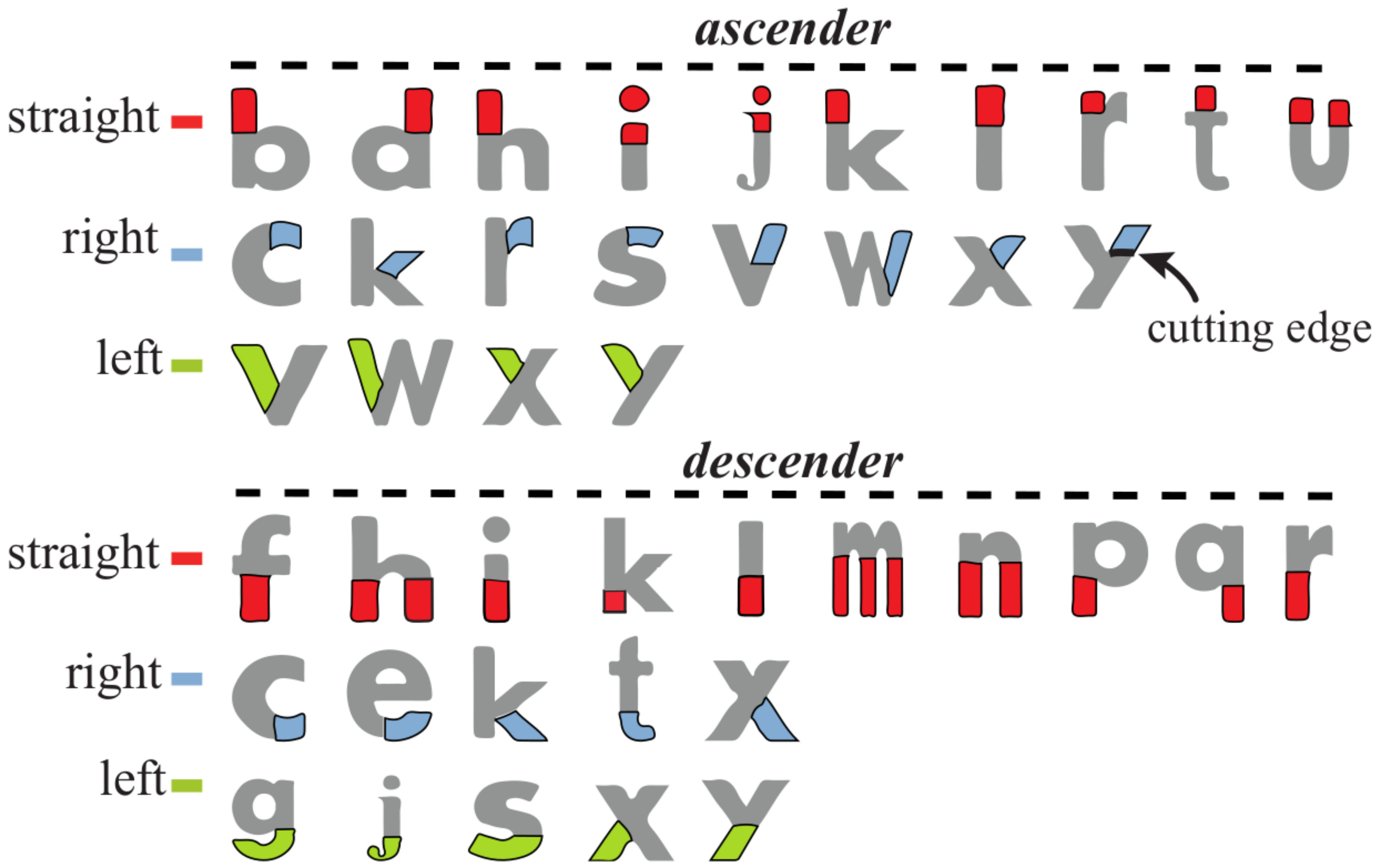


Figure 4: *Detected protrusions (with colors different from the main body) and layout paths (red) on a few shapes.*

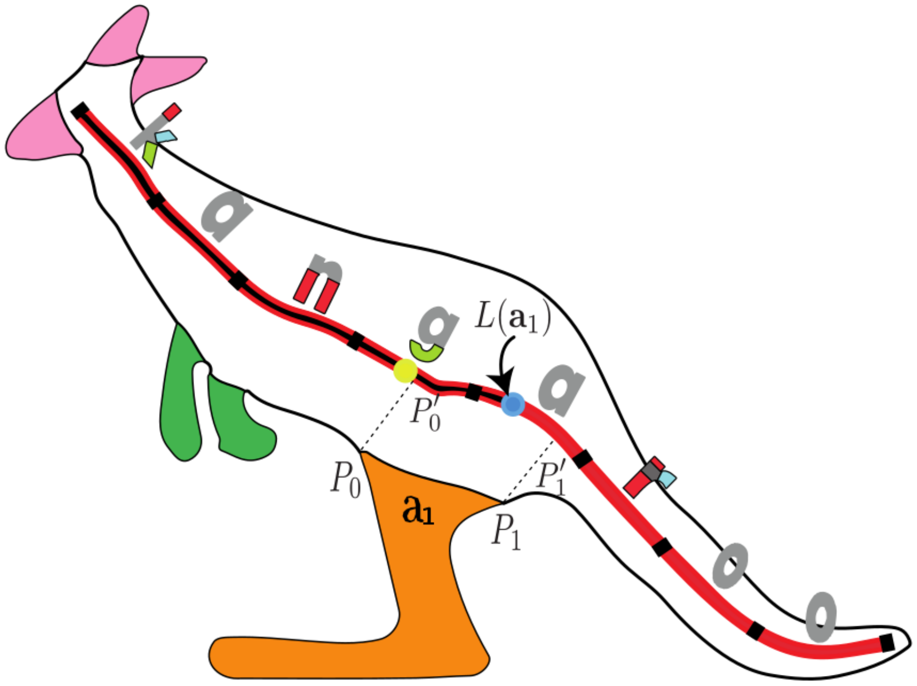
Letter anchors



Positional Compatibility

- $L(s_j)$: Letter's optimal position
  $L(a_i)$: Approximate location of the protrusion

$$P_{loc}(\mathbf{a}_i, \mathbf{s}_j) = \exp\left(-\left(\frac{L(\mathbf{a}_i) - L(\mathbf{s}_j)}{2 \cdot W/N}\right)^2\right)$$



Type and Orientation Compatibility

- Between letter anchor and protrusion

$$\overbrace{0\,,1\,,0.5,\,0.75}^{P_{simm}}$$

Compatibility cost

$$c(\mathbf{a}_i, \mathbf{s}_j) = 1 - P_{loc}(\mathbf{a}_i, \mathbf{s}_j) \cdot P_{sim}(\mathbf{a}_i, \mathbf{s}_j)$$

$$c(\mathbf{A}, \mathbf{S}) = \sum_{i=1}^{M} c(\mathbf{a}_i, \mathbb{M}(\mathbf{a}_i)),$$

$$s.t. \quad L(\mathbb{M}(\mathbf{a}_i)) < L(\mathbb{M}(\mathbf{a}_j)), \text{ if } \quad L(\mathbf{a}_i) < L(\mathbf{a}_j$$

Layout energy

$$score_l = \sum_{i=1}^{N} \left| \ln \frac{r_d(l_i)}{r_o(l_i)} \right| / N$$

$$score_g = \left| \ln \frac{Area_{\mathcal{S}}}{\overline{Area}_{(L,\mathcal{S})}} \right|$$

$$score_w = \omega_z Var_s(L) + \omega_o Var_o(L) + \omega_b \Sigma_i \frac{Overlap_{(l_i, l_{i+1})}}{Area_{\mathcal{S}}}$$

$$S = \lambda score_l + score_g + \gamma score_w$$

Letter legibility

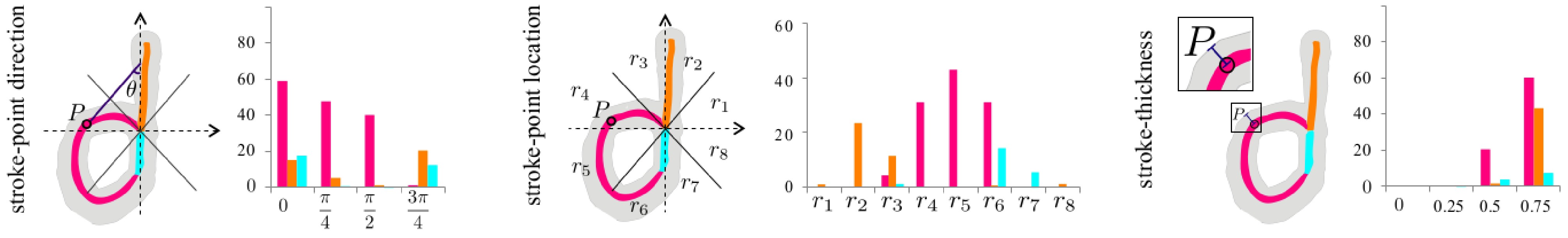
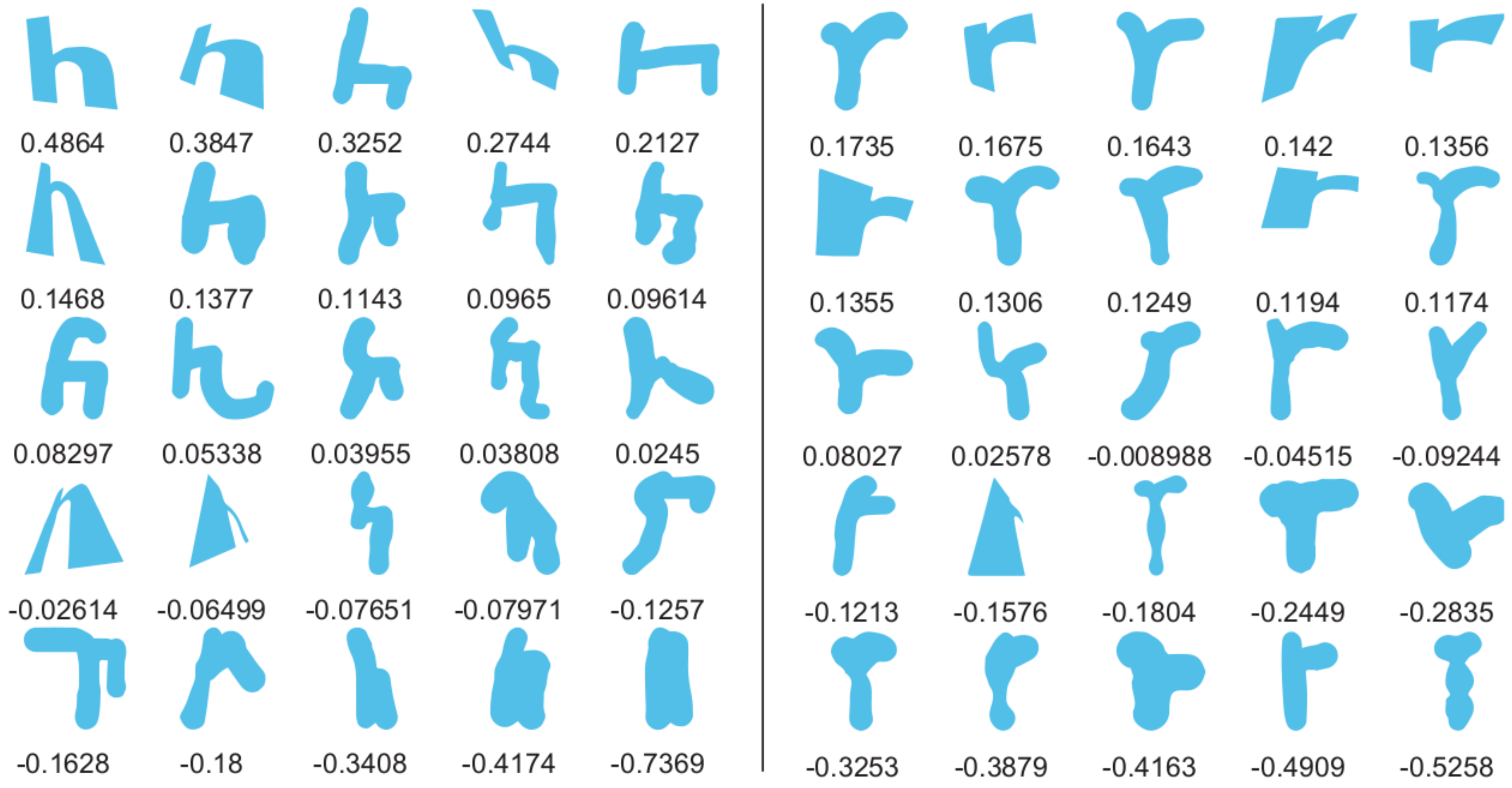


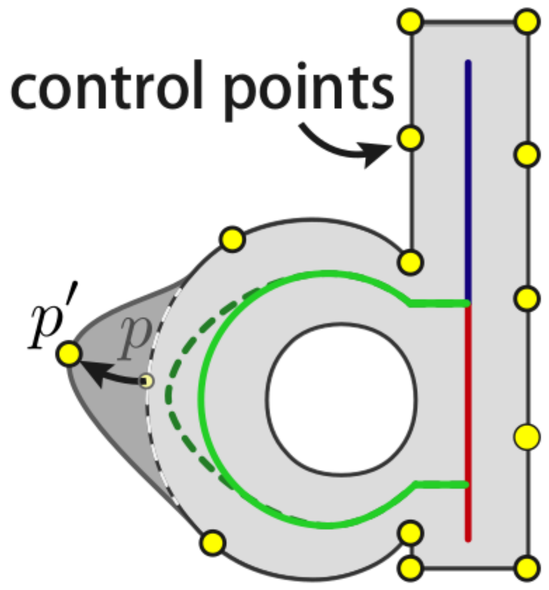
Figure 8: *Stroke features. From left to right, they are the histogram features of stroke orientation, position, and thickness. Histograms bins are colored according to the color of the corresponding stroke in the letter ‘d’.*

Ranking Function

$$r_l(\mathbf{x}) = \mathbf{w}_l^T \mathbf{x}$$



Letter Deformation

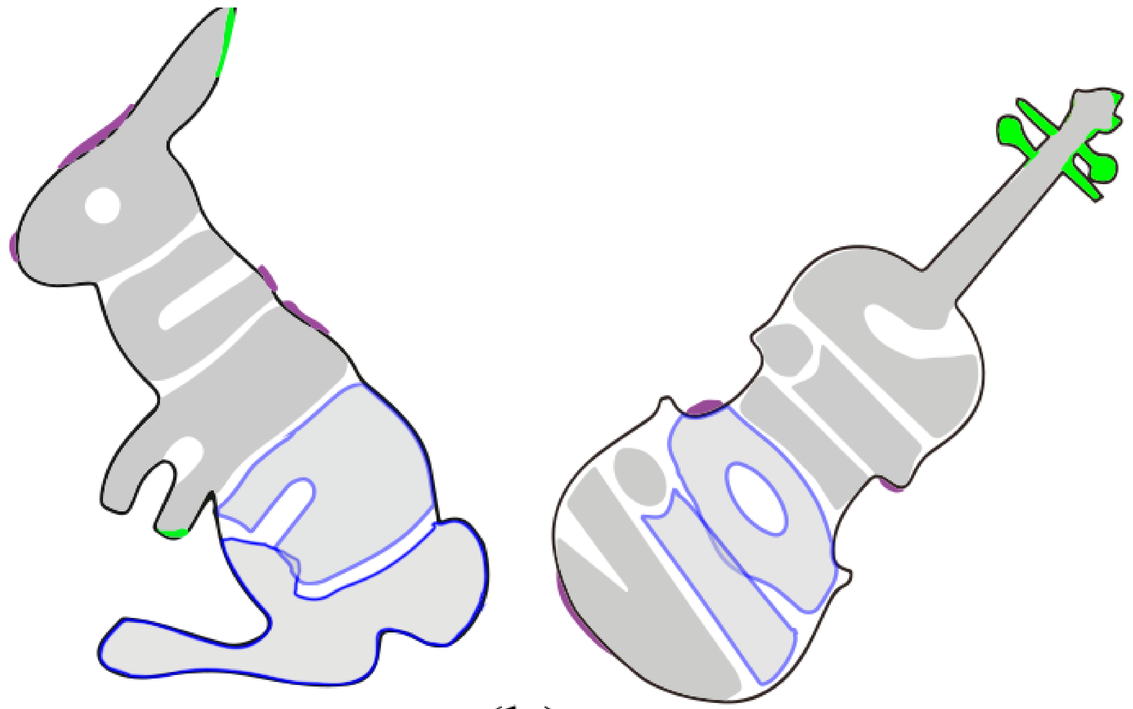


(a)

$\boldsymbol{\mu} = [10, 5, 3]$ pixels at different iterations

$$\hat{score}_l = (1 - \sum_{i=1}^{N} (r(l_i) + 1)/2N)$$

Post-processing


(b)

Result

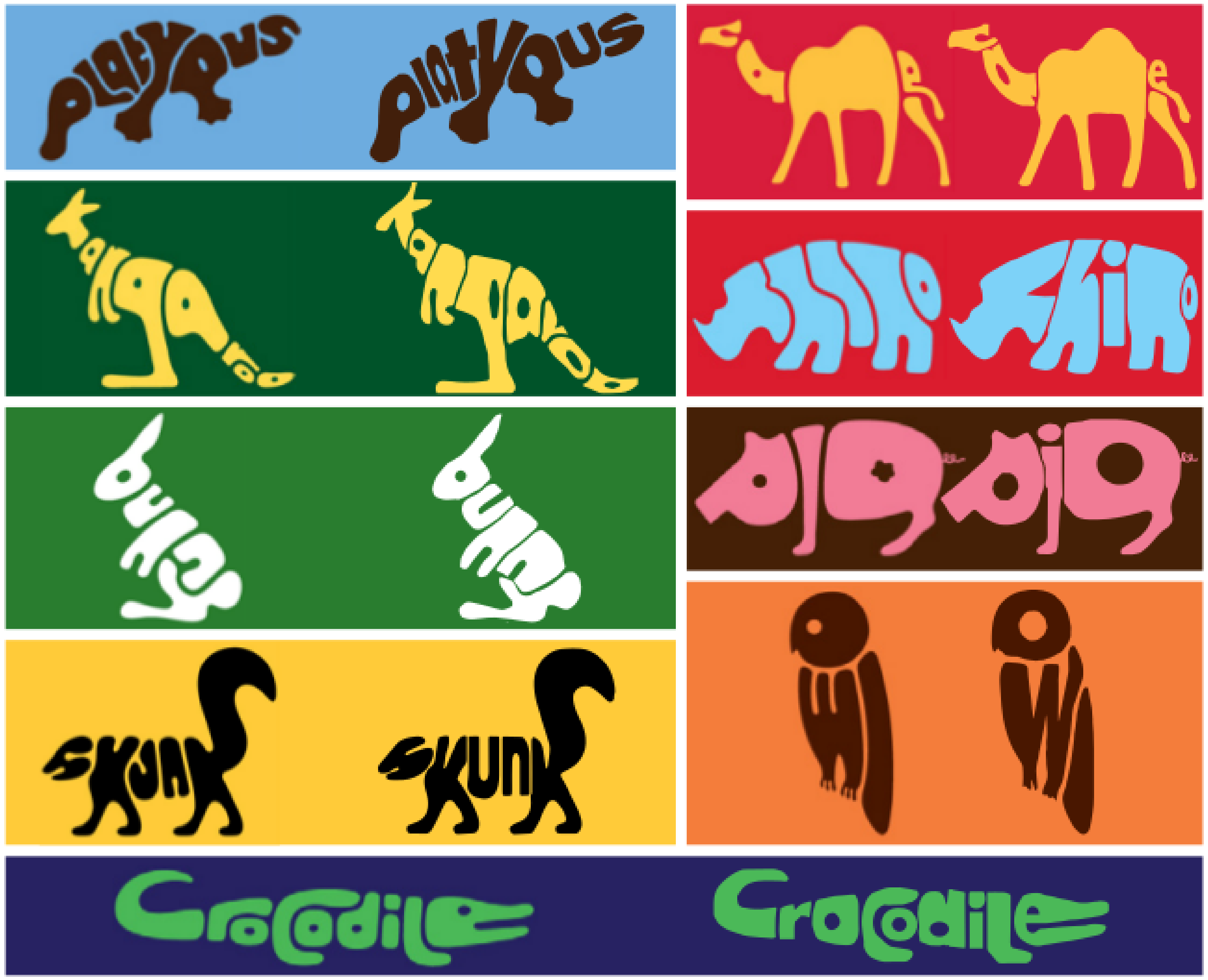


Figure 12: *A gallery of word animals generated by our automatic algorithm (right image in each pair), compared to designs from Dan Fleming (images taken with permission). In most cases, the overall quality of the results are quite close.*

Result

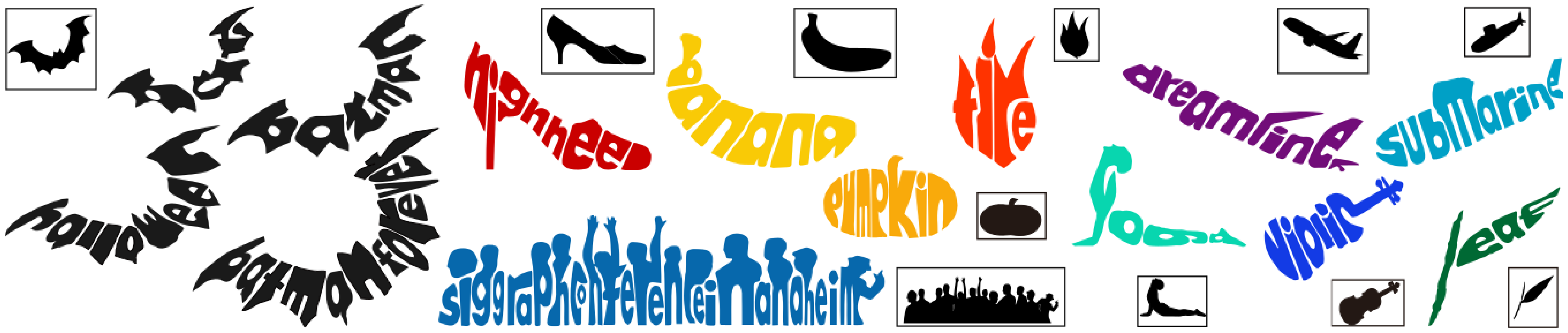
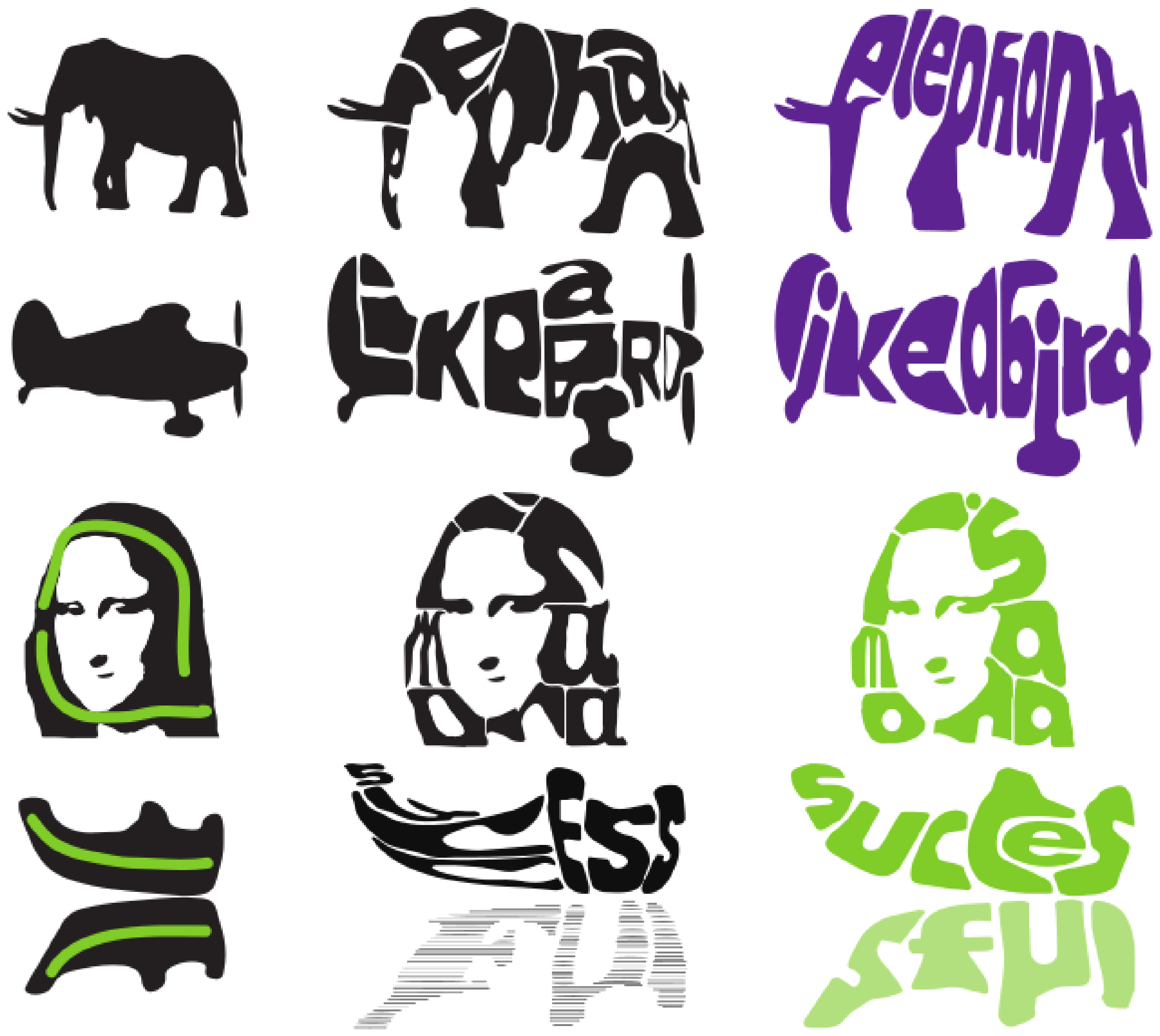


Figure 14: *A gallery of compact calligrams obtained by our method on input shapes obtained via Google image search.*

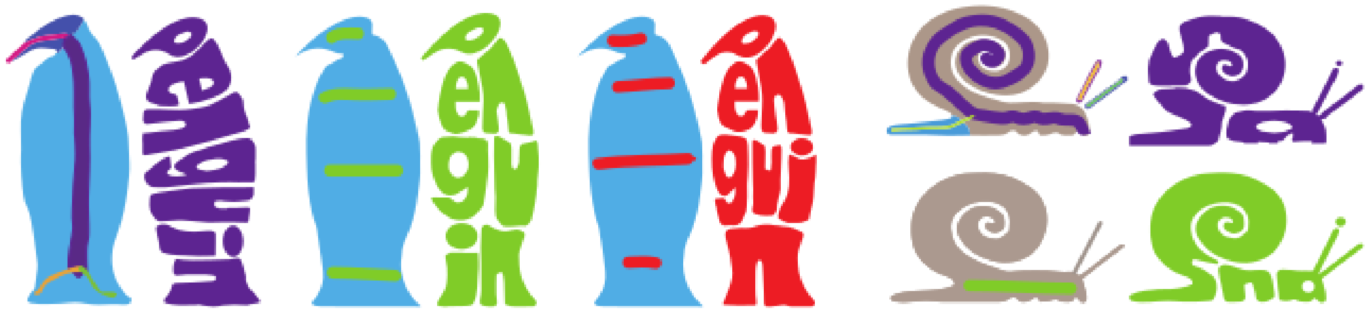
Comparison to Calligraphic Packing



Timing

- Except for the final deformation step, take less than 30 seconds to execute
- Average run time for the final deformation of 10 examples was 11.8 minutes

Limitation



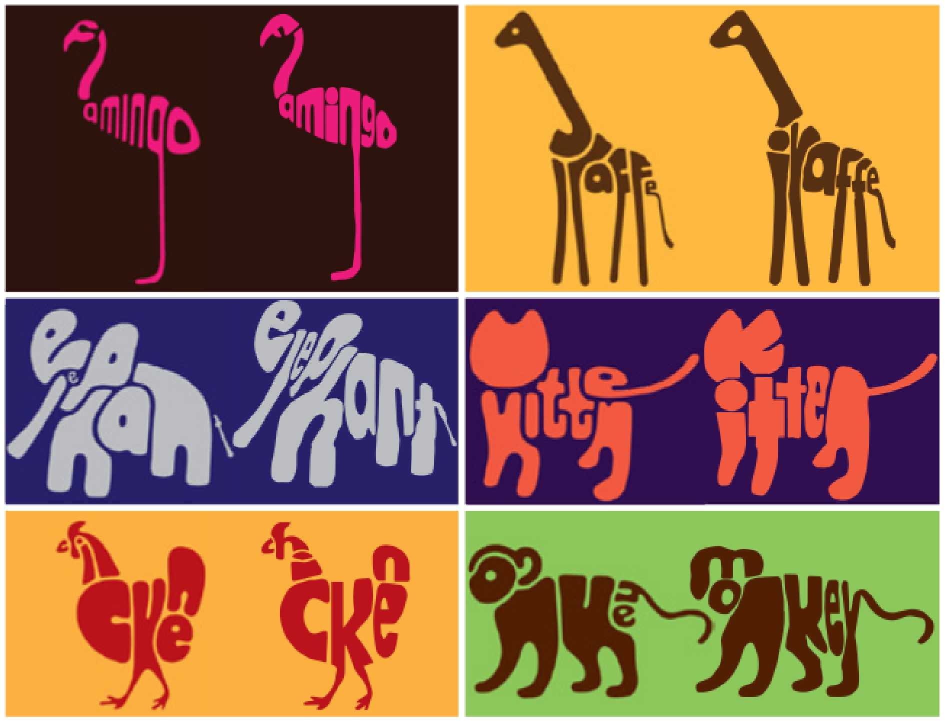


Figure 13: *Additional examples comparing algorithmic generation with expert design, showing more noticeable discrepancies. Yet, our results are still seen as providing reasonable alternatives.*

Future Works

- A user may only provide a textual description, e.g., “a jumping tiger”, for the image.
- Morphing between two word calligrams, shape-to-shape and letter-to-letter, in a visually pleasing way.