Supplementary Material for "Winding Number Features for Vector Sketch Colorization"

SUBMISSION ID: 1013

Contents

1 Background: k-means Clustering

With winding number features in hand, our method applies a tailored k-means++ clustering to determine the multi-region segmentation. Thus, we briefly review this classic method here, starting with k-means. The input is a cloud of points $P = \{q_0, q_1, \ldots, q_{|P|-1}\}$ in $\mathbb{R}^n$, and k cluster centroids $\mu_0^0, \ldots, \mu_{k-1}^0$. At each ith iteration, $\mathbb{R}^n$ is partitioned into k parts via the Voronoi diagram seeded by the centroids $\mu_0^i, \ldots, \mu_{k-1}^i$, and these are then updated to the center of mass of the points in their Voronoi cell $B(\mu_j^i)$:

$$\mu_j^{i+1} = \frac{\sum_{v \in B(\mu_j^i)} v}{\left| B(\mu_j^i) \right|} \tag{1}$$

Iterations continue until the movement of the cluster points stabilizes and falls below some user-specified threshold $\varepsilon > 0$.

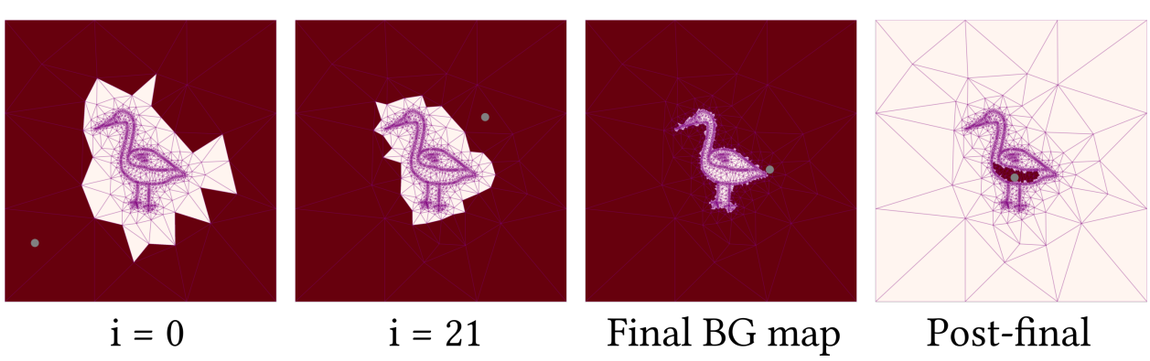
2 Additional details on automatic cluster intialization

k-means requires the specification of initial cluster centroids, or uses random centroids. We have formulated an initialization strategy described in §4.2.3 of the main text. Below, we clarify additional details.

We note that the background mask referenced in our automatic method deters oversegmentation of the background *but need not be very accurate to serve this purpose.* As such, we propose a straightforward approach to obtaining such a mask, noting that this approach is un-optimized and secondary in importance to the actual selection of seeds. To this end, the background mask, T_{BG}, is generated via an iterative process. Starting from $i = 0$:

1. Pick the largest un-selected triangle, t_s^i.
2. Identify all triangles $T_{BG}^i \subseteq T$ for which a path satisfying a *bottleneck constraint* exists from t_s to each $t \in T_{BG}^i$.
3. Proceed forward once $T_{BG}^i \not\subseteq T_{BG}^{i-1}$, taking T_{BG}^{i-1} as our background mask.

Intuitively, this process grows the background mask by progressively "stepping in" from the corners of the sketch until a point *inside* the sketch is discovered. See the following illustrative example:

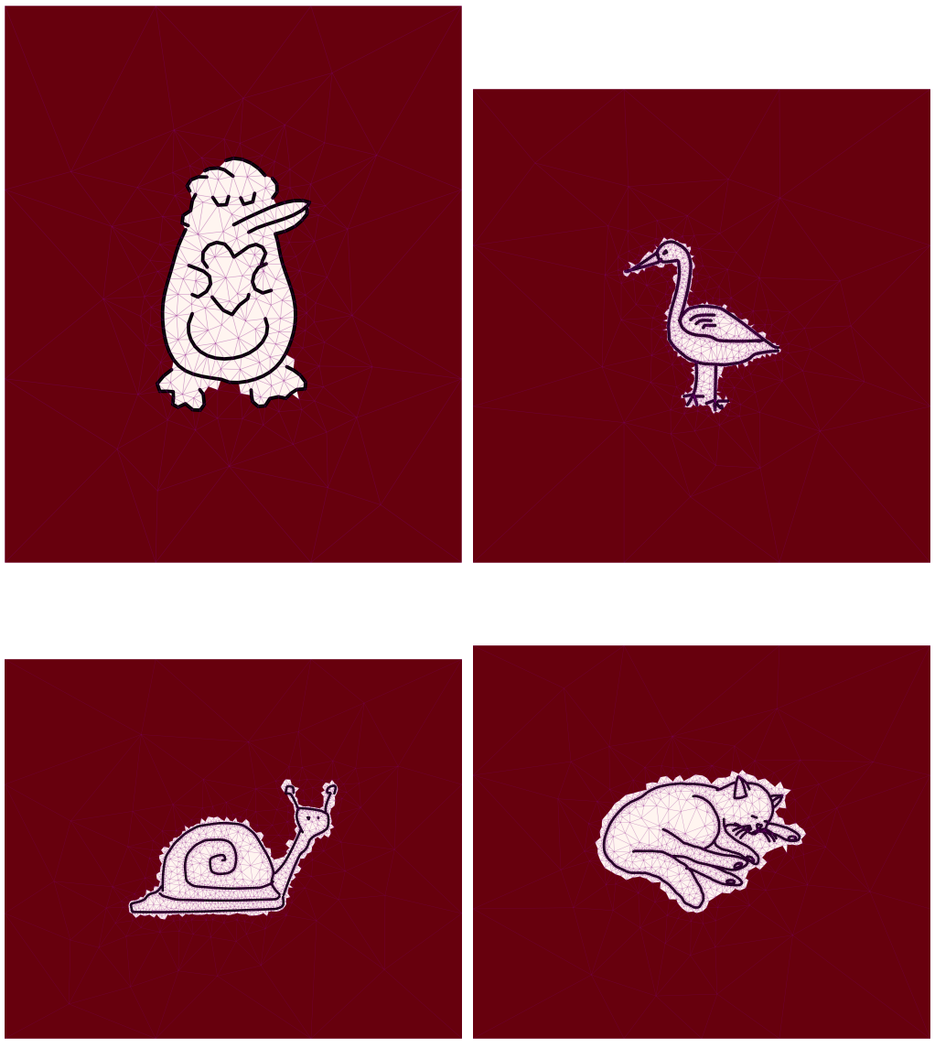


The background mask at various stages of the iterative process. The barycenter of t_s^i is plotted in grey. Left to right: $i = 0$ BG mask, $i = 22$ BG mask, final BG mask, and "post-final" mask that causes the algorithm to terminate at step (3).

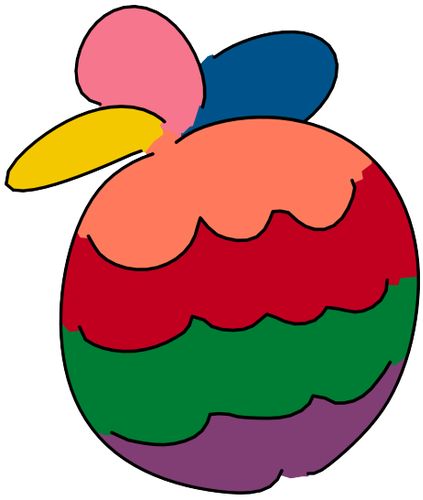
In generating a map initialized at $t_s \in T$, we say that there is a valid path P from a source triangle to a destination triangle if P satisfies our *bottleneck constraint*. Let each $e \in P$ represent a directed edge between two triangles on the path: $e = (t_1, t_2, l)$, where t_1 and t_2 are the source and destination triangles and l is the length of the edge **shared** by t_1 and t_2. Additionally, let $equilateral(t)$ be the side length of an equilateral triangle with the area of t: $equilateral(t) = \sqrt{\frac{4}{\sqrt{3}} t_a}$. Then, e satisfies the bottleneck constraint if $e_l \geq \frac{equilateral(t_2)}{\alpha}$ with α as an additional tuning parameter. By varying α over a large interval in experiments, we found $\alpha = 2.5$ to be a reliable choice with strong performance over a wide variety of triangulations. As such, $\alpha = 2.5$ is left as default in all our results and does **not** require modification by the user.

This bottleneck constraint was chosen via experimentation on diverse sketches, but we note that it is dependent on the triangulation and may fail in rare cases. If our *iterative* background map initialization fails (i.e. the result covers too much of the sketch), we instead generate four maps initialized at each of the *corners* of our sketch, and take the union as our background map. This conservative strategy offers good performance as an alternative.

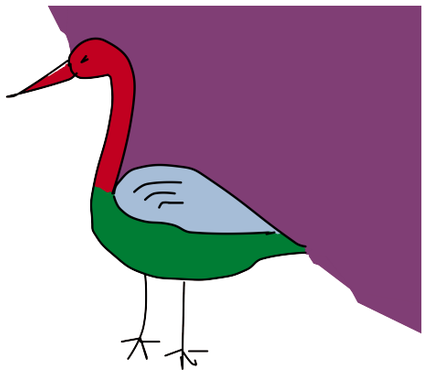
Below, we show the background masks for several sketches:



Lastly, we show sample automatic results for sketches in which the background mask is **not** used as proof of its secondary importance to the seed-selection mechanism. Note that the results are largely consistent with those found elsewhere in our work.

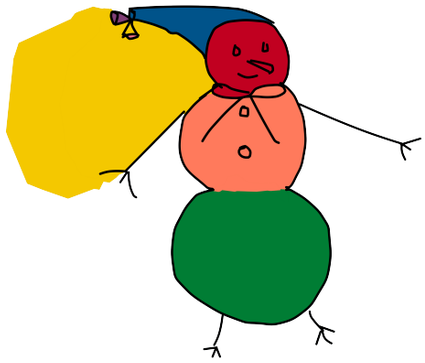


(a) $k = 8$, no BG mask

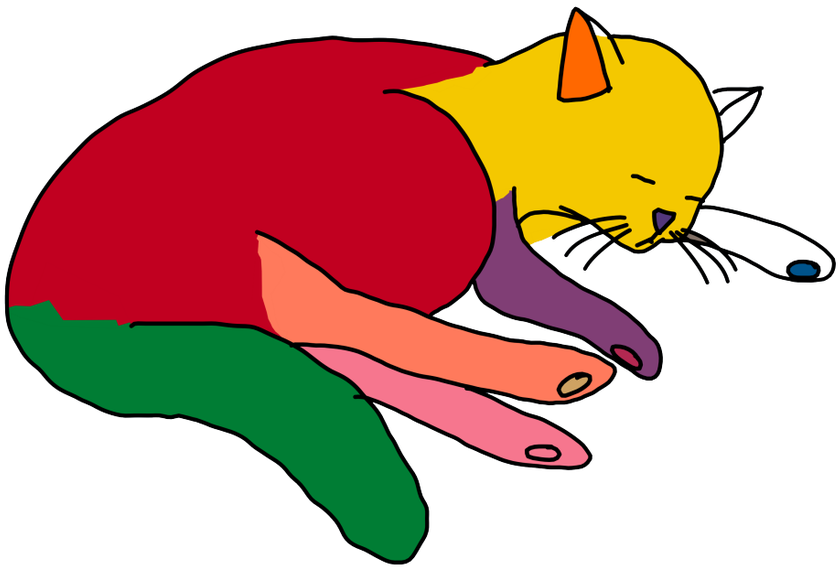


(b) $k = 5$, no BG mask

(a) $k = 8$, no BG mask



(b) $k = 6$, no BG mask

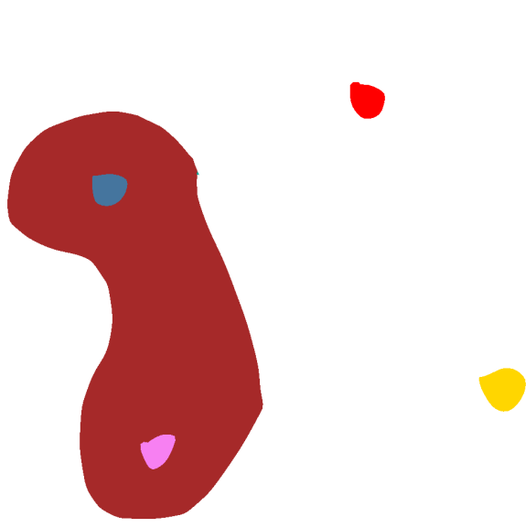


(a) $k = 9$, no BG mask

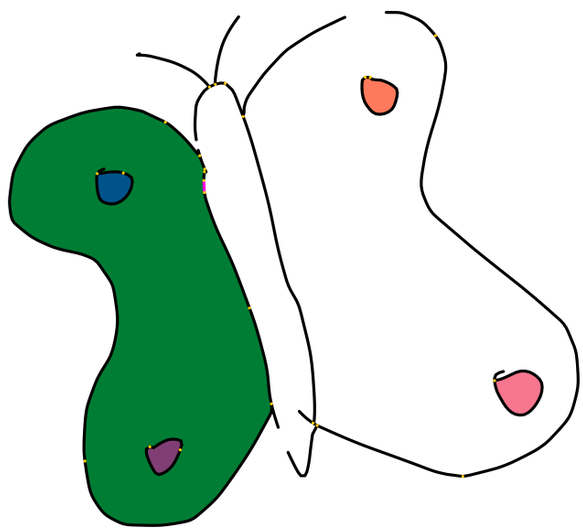
3 Quantitative Experiment: Additional Details

Here, we elaborate on the quantitative experiment outlined in §5.1.1 of the main text. To generate the ground truth, small gaps in the original inputs were completed with the Live Paint tool of Adobe Illustrator. The boundaries of the resulting closed sketches were then destroyed at random by introducing gaps. Along each stroke/polyline $s = \{r_0, r_1, \ldots, r_n\}$, we sample $rand(1, \frac{|s|}{16})$ points at which to create holes. The size of each hole is sampled from a Gaussian distribution with mean proportional to n for each stroke. We note the exact parameters (mean, standard deviation) in our dataset. After running our method and that of [YLL*22] on the destroyed, gappy inputs, we rasterize each output along with the ground truth segmentation map at 300 DPI and extract region contours via Canny edge detection [Can86]. Finally, each method's processed output was scored against the contours of the ground truth segmentation using the software of [YVG20].

We note that in addition to detecting closed loops/fill regions, the reference implementation of [YLL*22] displays attempted junction closures (see below). As we focus on segmentation rather than curve completion in this work, we compare **only** to the fill regions generated by their method.



(a) Yin and Liu et al. (2022), fill regions only

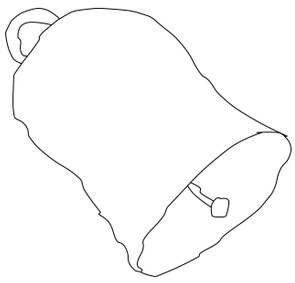


(b) Yin and Liu et al. (2022), fill regions + junction closures

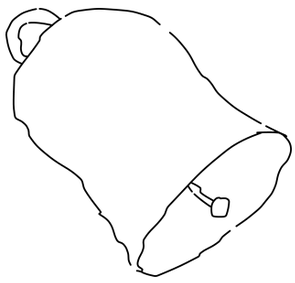
It is likely that overlaying the junction closures and input strokes on their fill regions would result in improved metrics: see results in §3.1, in which [YLL*22] successfully close some junctions that ultimately do not end up forming a fill region. However, this would be incompatible with our method, which is not focused on an explicit notion of stroke completion.

3.1 Results on Sketches Used in Quantitative Comparison

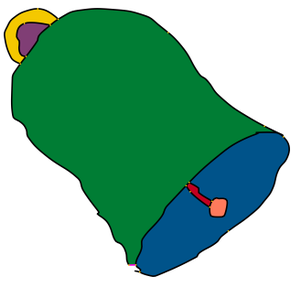
Below, we include results for all sketches along with the ground truth and "fragmented" inputs.



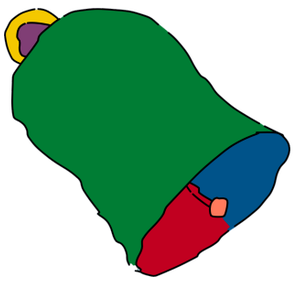
(a) Ground truth



(b) Input with gaps

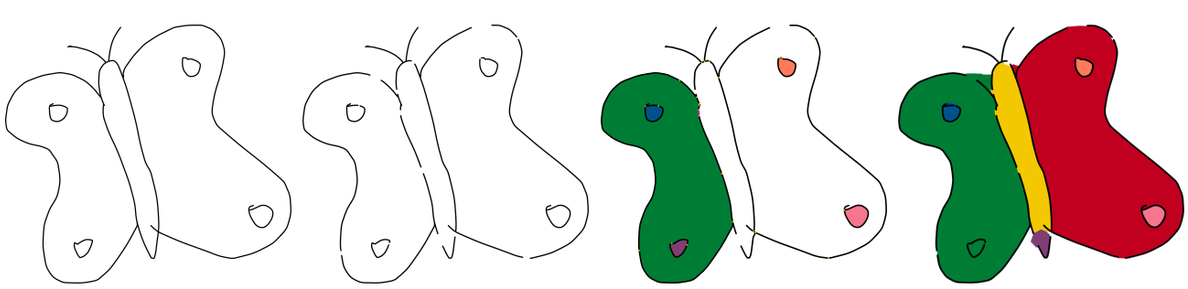


(c) Yin and Liu et al. (2022)

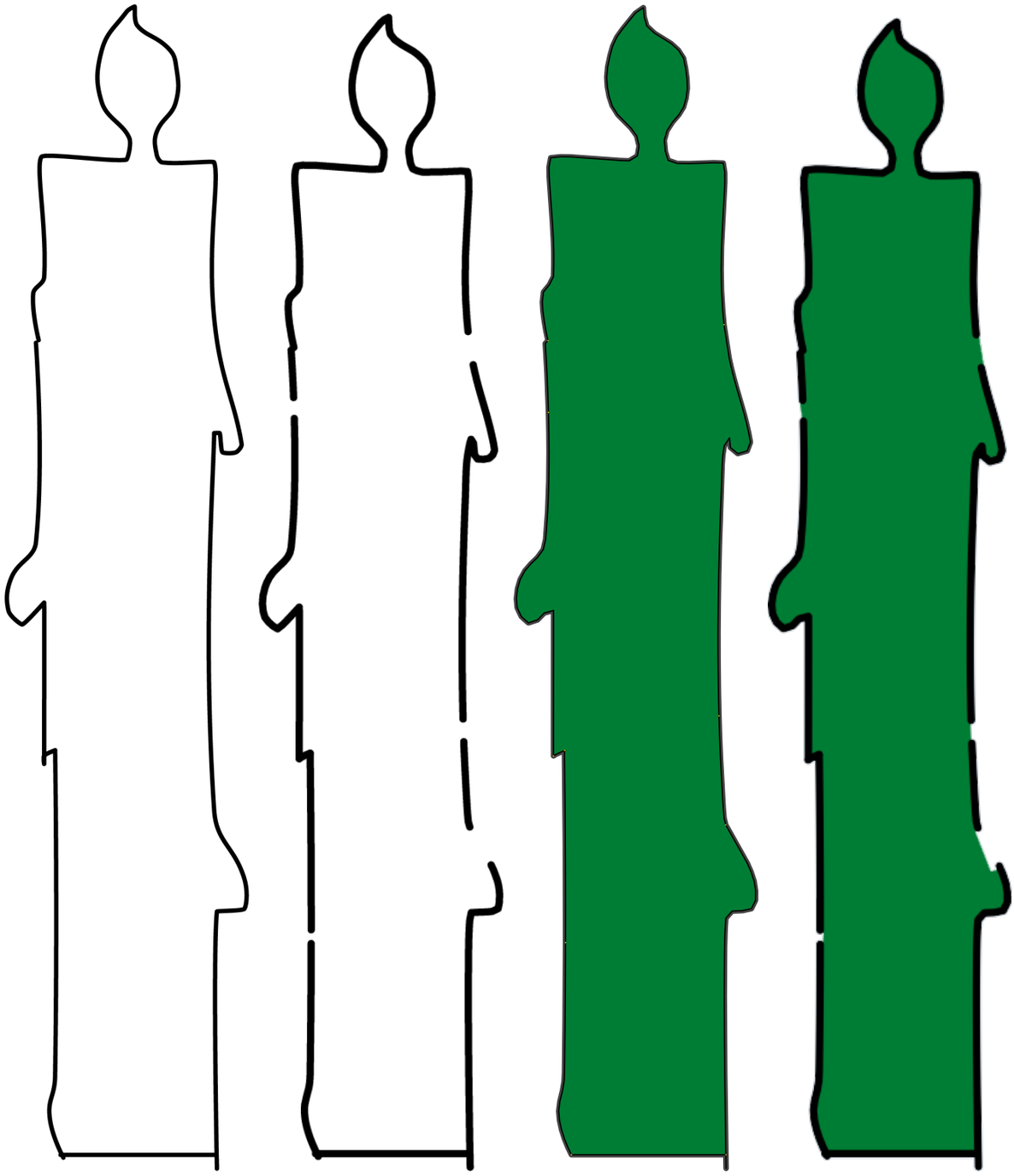


(d) Ours ($k = 7$)

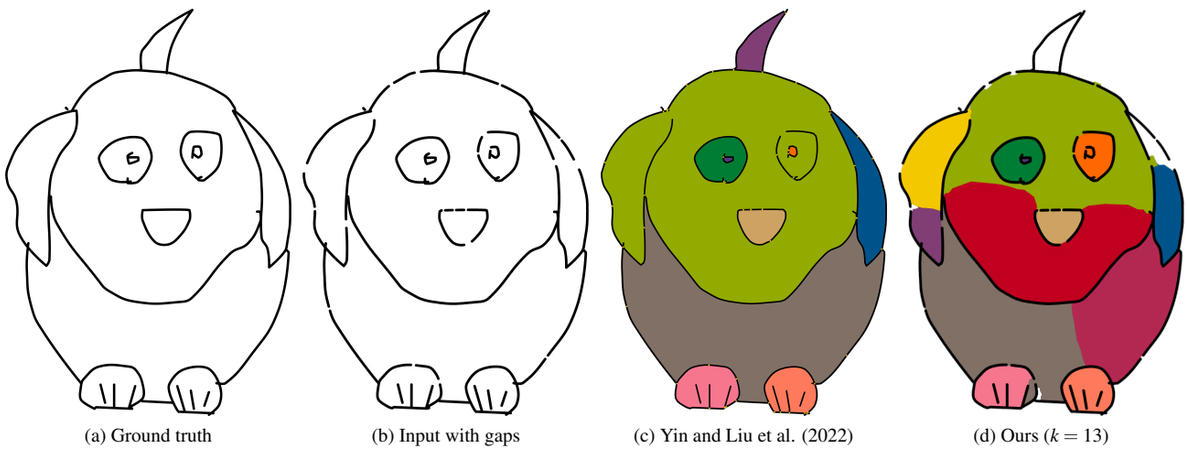
(a) Ground truth (b) Input with gaps (c) Yin and Liu et al. (2022) (d) Ours ($k = 6$)



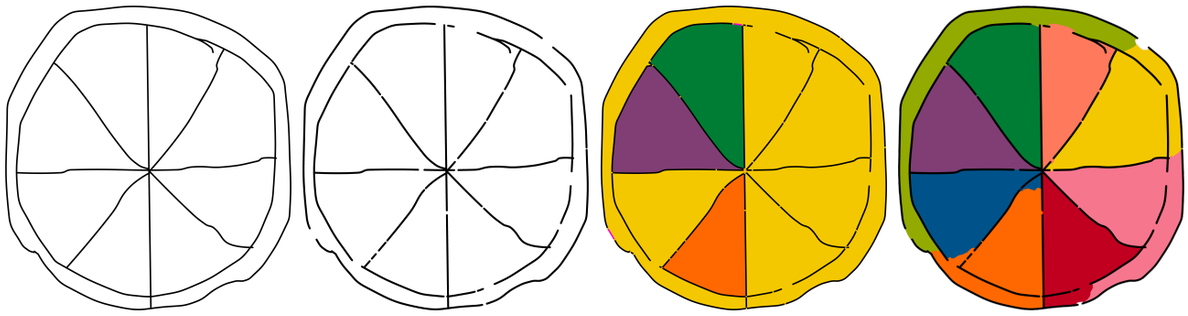
(a) Ground truth (b) Input with gaps (c) Yin and Liu et al. (2022) (d) Ours ($k = 8$)



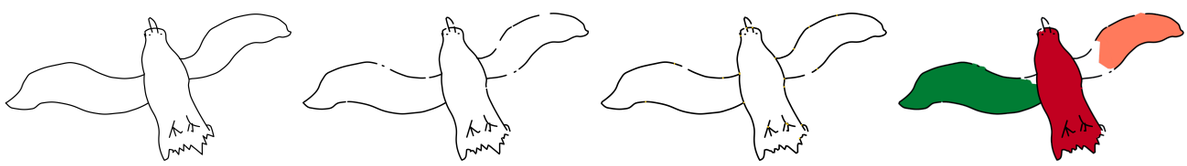
(a) Ground truth (b) Input with gaps (c) Yin and Liu et al. (2022) (d) Ours ($k = 2$)



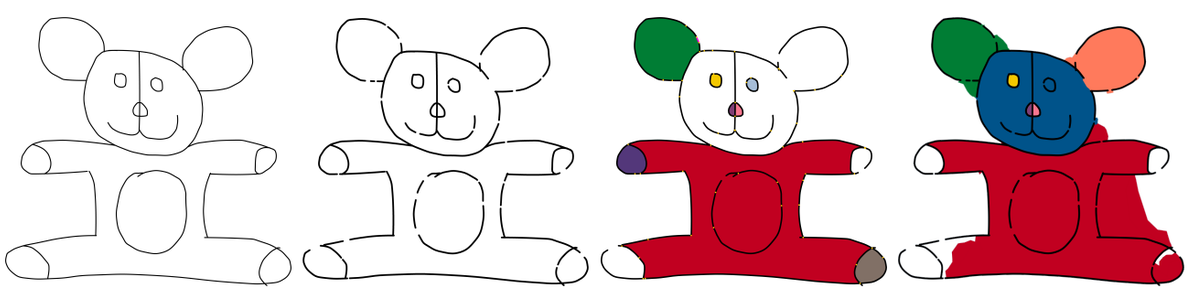
(a) Ground truth (b) Input with gaps (c) Yin and Liu et al. (2022) (d) Ours ($k = 13$)



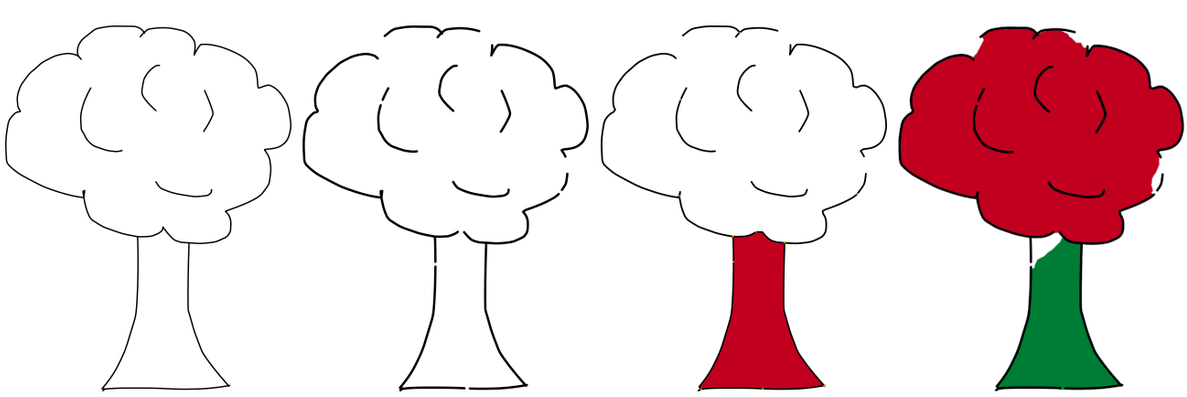
(a) Ground truth (b) Input with gaps (c) Yin and Liu et al. (2022) (d) Ours ($k = 10$)



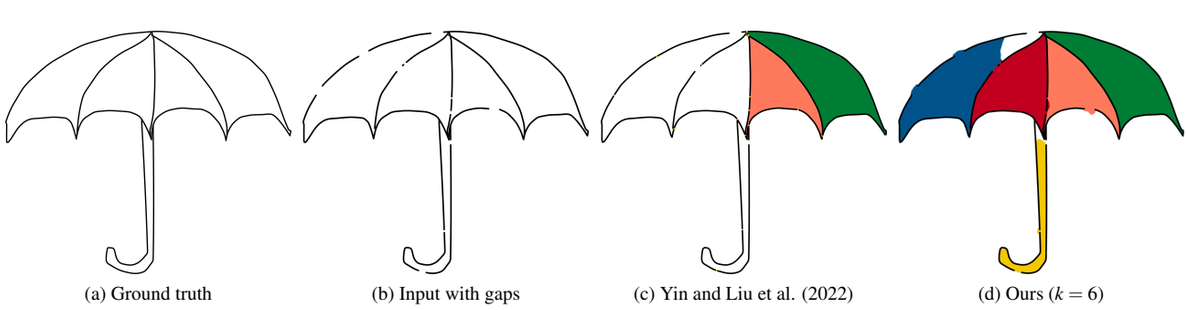
(a) Ground truth (b) Input with gaps (c) Yin and Liu et al. (2022) (d) Ours ($k = 4$)



(a) Ground truth (b) Input with gaps (c) Yin and Liu et al. (2022) (d) Ours ($k = 6$)



(a) Ground truth (b) Input with gaps (c) Yin and Liu et al. (2022) (d) Ours ($k = 3$)

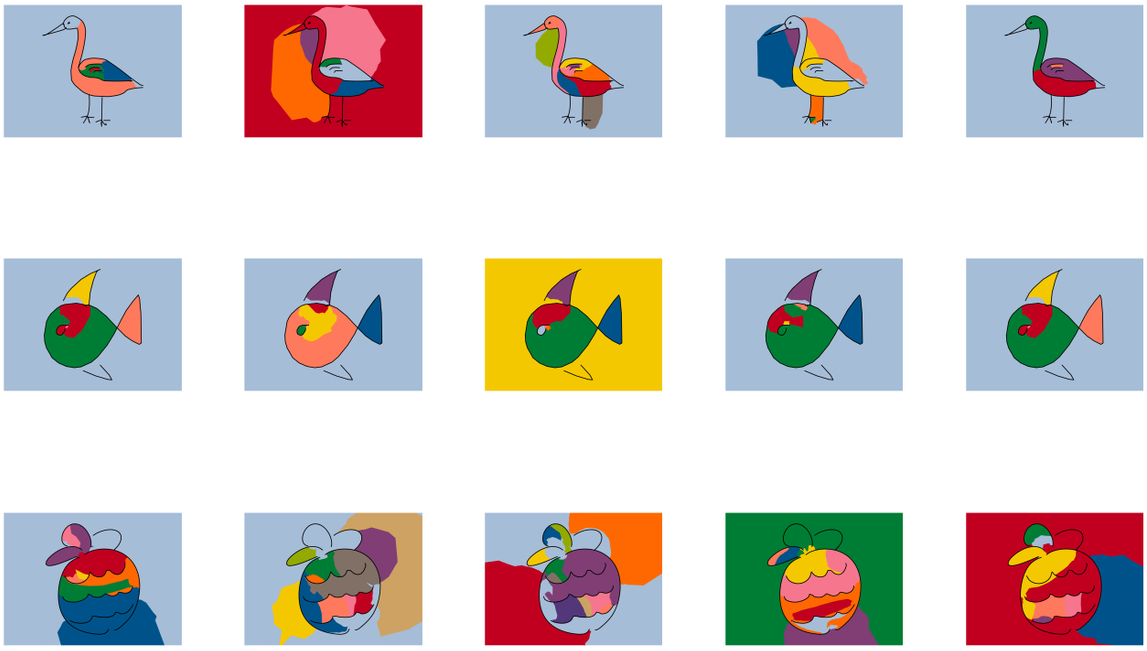


(a) Ground truth (b) Input with gaps (c) Yin and Liu et al. (2022) (d) Ours ($k = 6$)

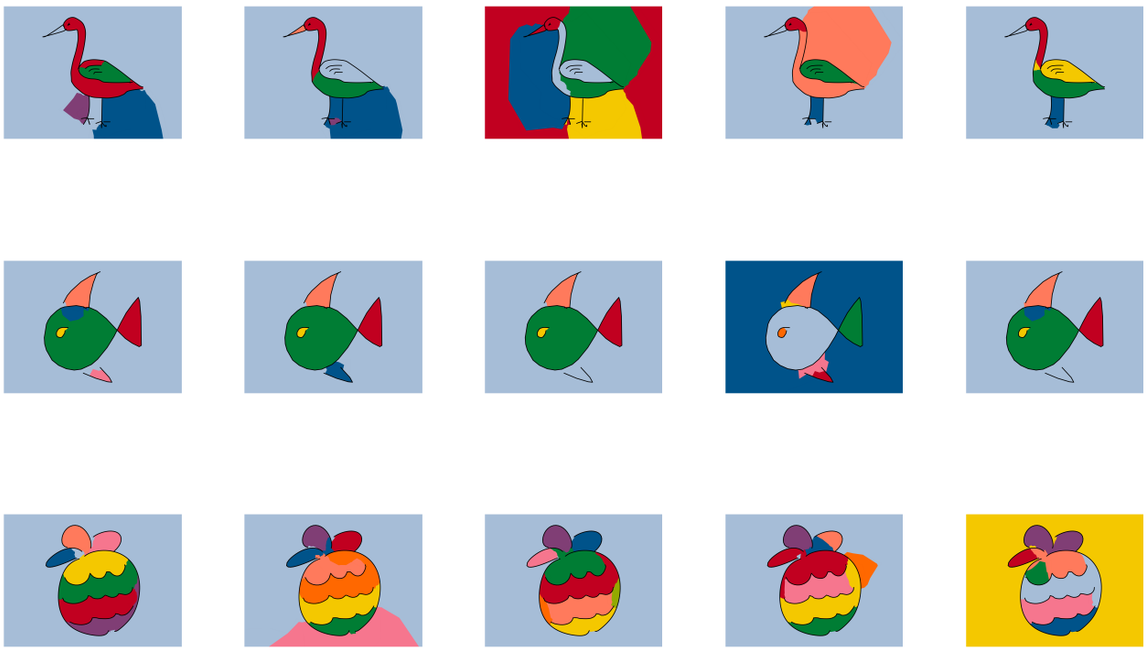
4 Choosing the number of features

In §4.2.1 of the main text, we note that our method is not particularly sensitive to the choice of m, the number of random features on which to perform clustering. Broadly speaking, our experiments suggest that there exists some "minimum" number of features needed to differentiate fill regions — in practice, this number is quite small (on the order of tens of features). Increasing the number of features beyond this minimum threshold may increase the consistency of automatic results to a point, but gains are minimal once we have reached hundreds of features for simple sketches (such as those shown in our results). To provide a better sense of this analysis, we run our automatic method 5 times for different choices of m on 3 sample sketches (crane: $k = 5$; fish: $k = 6$; fruit: $k = 8$).

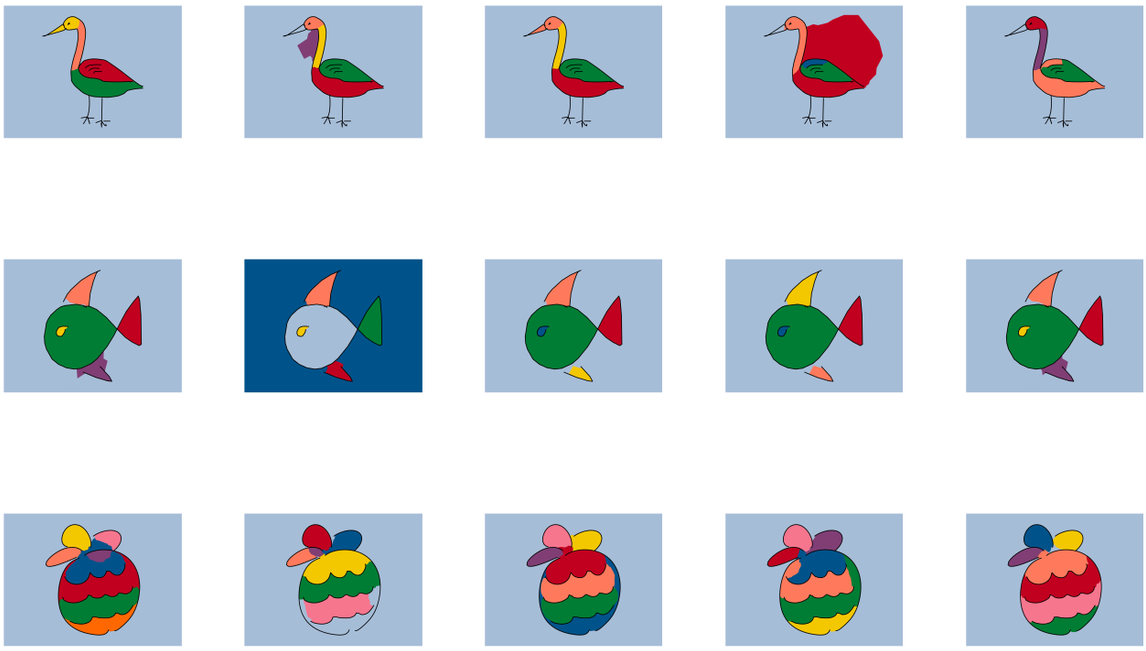
4.1 $m = 1$



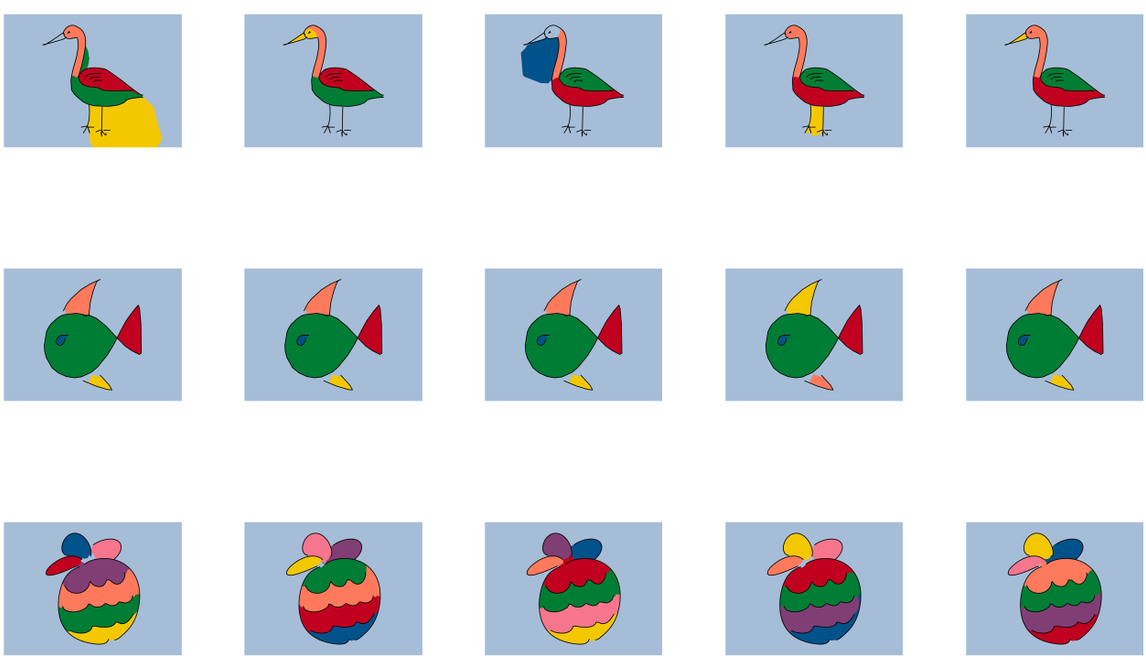
4.2 $m = 5$



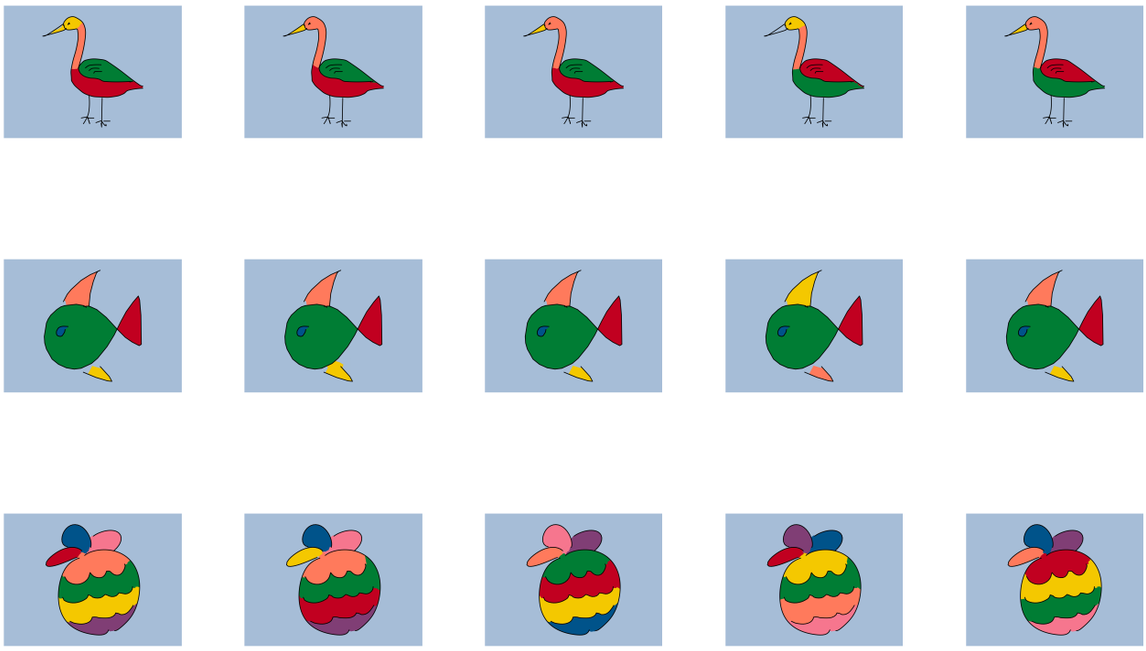
4.3 $m = 10$



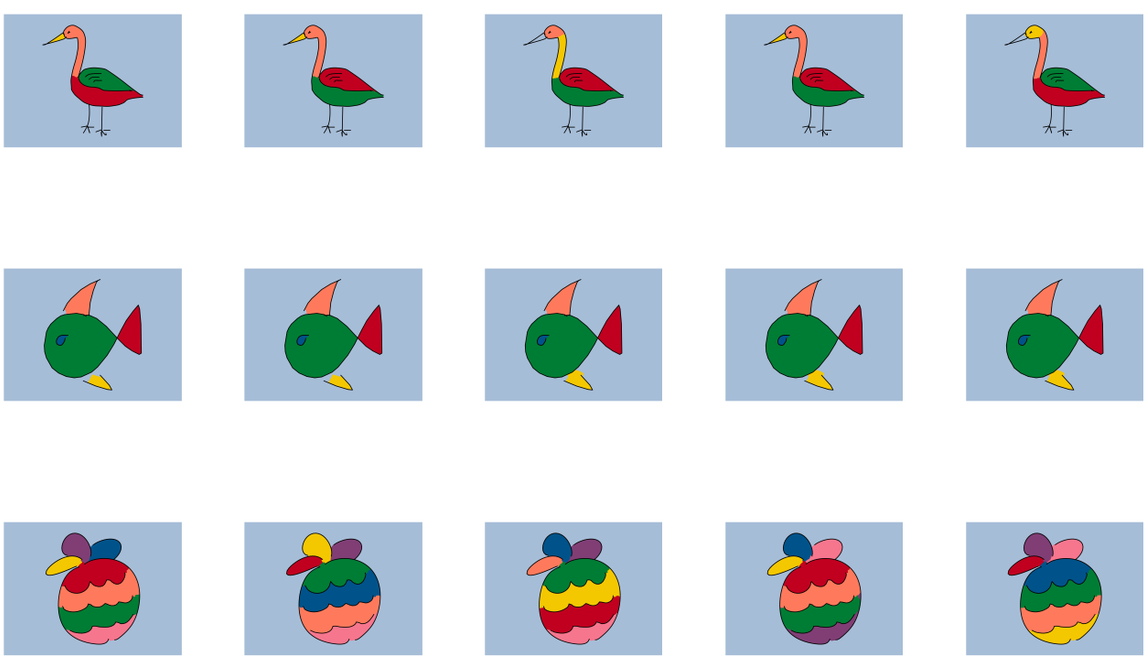
4.4 $m = 20$



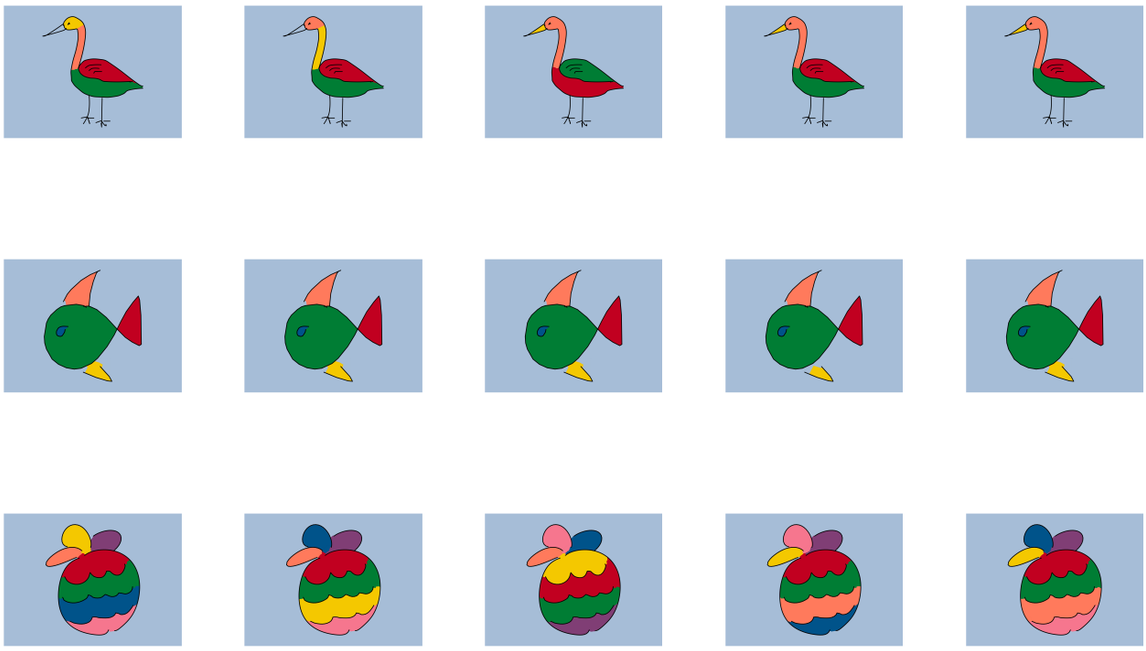
4.5 $m = 50$



4.6 $m = 100$



4.7 $m = 200$



5 Power Radius Adjustment

Requiring users to specify the exact power radius of each cluster centroid would prove to be a tedious task. We incorporate this parameter as an intuitive element of our user interface by allowing users to drag a unitless slider to adjust the "strength" of each color hint. In the background, this is implemented as a multiplier for a base unit, δ, by which the power radius is incremented or decremented whenever the user moves the slider. We set δ according to the size of the m-dimensional embedding space $\mathscr{E}$. Let b_1 and b_2 be the minimum and maximum coordinate tuples of the axis-aligned bounding box that encloses $\mathscr{E}$. Then, δ is calculated as follows:

$$\delta = \frac{\|b_1 - b_2\|^2}{250} \tag{2}$$

Another possible choice would be to initialize δ on a per-cluster basis according to the average squared distance between vertices and their nearest centroids. We found that the first initialization of δ worked well in practice, producing modifications to the boundary that were neither too incremental nor too drastic.

6 Post-processing

6.1 Cluster Refinement

Here, we elaborate on the post-processing procedure referenced in §4.2.4 of the main text. Each cluster is first split into a series of connected components. The largest component is always retained, as are all sufficiently large components (in our case, any component with area at least $\frac{1}{2}$ of the largest component). For remaining components, we evaluate the length of the shared boundary with all neighboring clusters and merge accordingly. Crucially, some fully disconnected components of the mesh may not share boundary with *any* other cluster. These correspond to trivial fill regions that would be detected via traditional flood-filling. Such regions are spun off as their own unique clusters; therefore, the user need not specify regions of the sketch that are fully closed, as these will be detected automatically at this stage. Neither is it harmful for the user to provide manual hints for these regions, as the winding number will provide strong segmentation information. Fig. 4 in the main text of our paper demonstrates a characteristic result of this process.

6.2 Boundary Extraction

In order to produce flat-filled color regions, we extract cluster boundaries from the embedding of M in the feature space $\mathscr{E}$. First, for all "mixed edges" in the triangulation — edges $(p_i, p_j) \in E$ in which the endpoints are associated with different clusters — we compute the interior point p^* with equal power to cluster centroids c_i, c_j having radius r_i, r_j. This is the point of intersection between the edge and the hyperplane separating the two clusters, and satisfies the following geometric property:

$$(c_1 + c_2 - 2p^* - s)^T (c_2 - c_1) = 0 \tag{3}$$

s is a corrective factor that accounts for the radii of the two hyperplanes. We may calculate s as a linear combination of the vectors resulting from the hyperplane's normal line. Specifically:

$$s = t(c_i - c_j) + (d - t)(c_j - c_i) \tag{4}$$

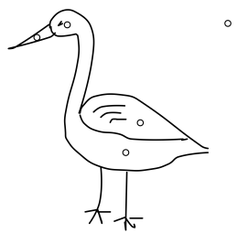
$$t = \frac{\|c_i - c_j\|^2 + r_i^2 - r_j^2}{2\|c_i - c_j\|} \tag{5}$$

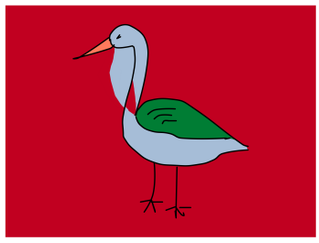
For simple mixed triangles in which any two points share the same cluster, we trace the boundary between the intersection points along each mixed edge. For complex mixed triangles in which all three points are identified with a different cluster, we must trace all three pairwise boundaries and find their point of intersection. If this point of intersection lies inside the complex triangle, we use it to divide the triangle into three regions. Otherwise, we fall back on the triangle's barycenter and proceed with an identical decomposition. A more robust approach would be to recalculate cluster boundaries on neighboring triangles when the point of intersection is outside the triangle, but we find that this is unnecessary for our goal of simply achieving the correct topology.

7 Total Variation Ablation Study

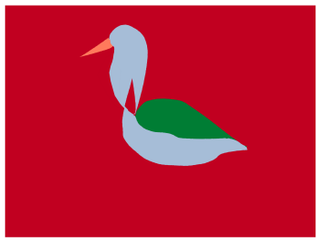
To demonstrate the effectiveness of total variation weights, we perform a small ablation study. Here, we aim to consider the effect of weighting each feature by its relative total variation: see §4.2.1 of the main text for the precise formulation. We fix a set of random stroke orientations and a set of user hints. For clarity, we visualize the resulting clusters *without* the post-processing referenced in §6.1.

7.1 Sketch 1: Crane

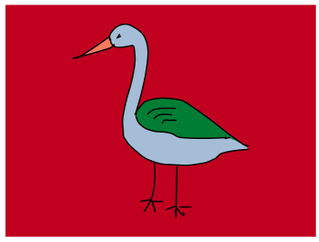




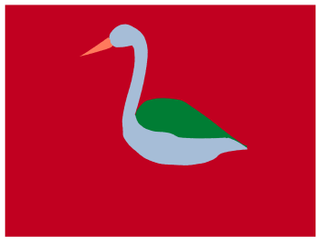
(a) No TV weights



(b) No TV weights (clusters)



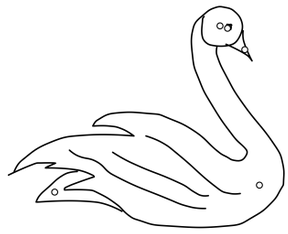
(a) With TV weights

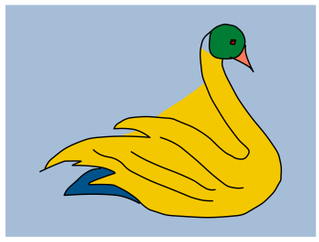


(b) With TV weights (clusters)

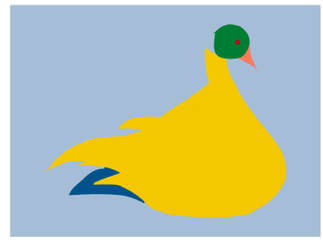
Figure 1: With TV weights

7.2 Sketch 2: Swan

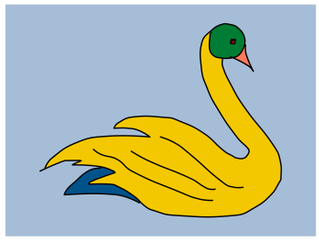




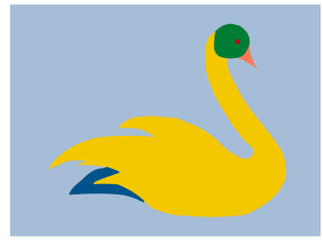
(a) No TV weights



(b) No TV weights (clusters)



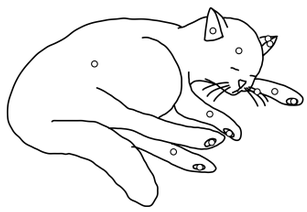
(a) With TV weights

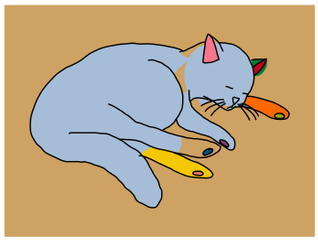


(b) With TV weights (clusters)

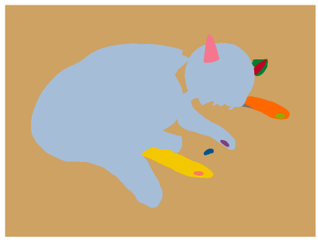
Figure 2: With TV weights

7.3 Sketch 3: Cat

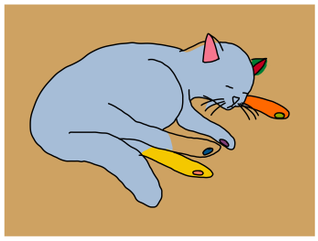




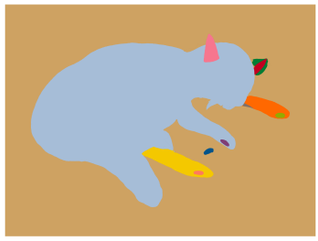
(a) No TV weights



(b) No TV weights (clusters)



(a) With TV weights



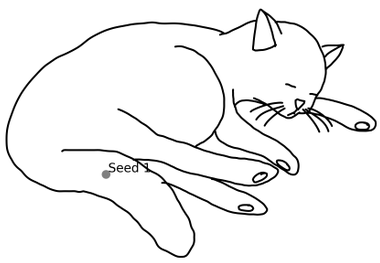
(b) With TV weights (clusters)

Figure 3: With TV weights

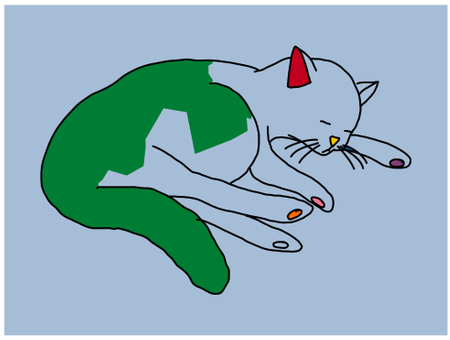
8 Varying k in Automatic Method

The choice of k in our automatic method corresponds to the number of desired fill regions and is discussed in further detail in §4.2.3 of the main text. To build intuition, we vary k for 3 sample sketches and show the results alongside the seed points obtained in our automatic initialization. The set of stroke configurations (and thus winding number features) does not change between runs.

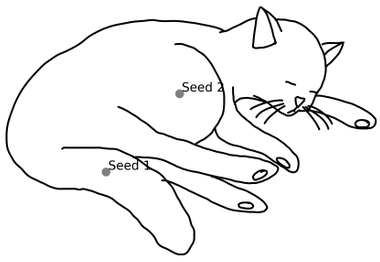
8.1 Sketch 1: Cat



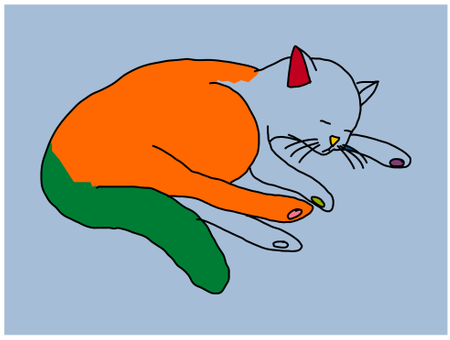
(a) $k = 2$, seeds



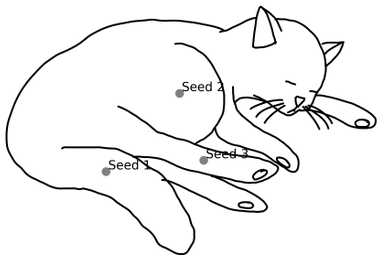
(b) $k = 2$, regions



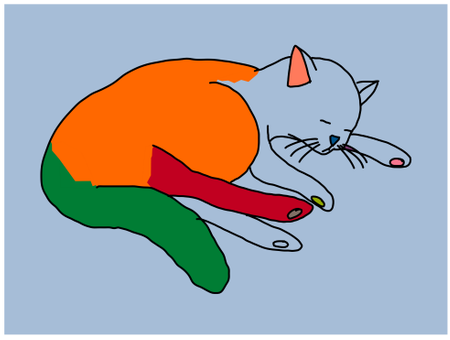
(a) $k = 3$, seeds



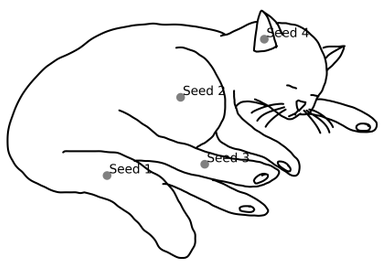
(b) $k = 3$, regions



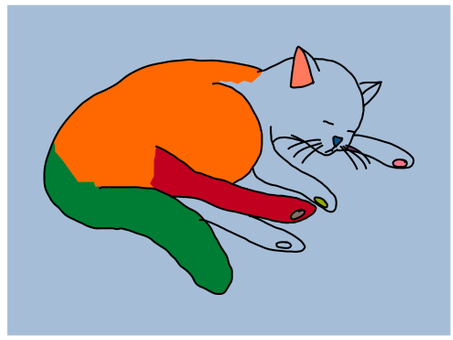
(a) $k = 4$, seeds



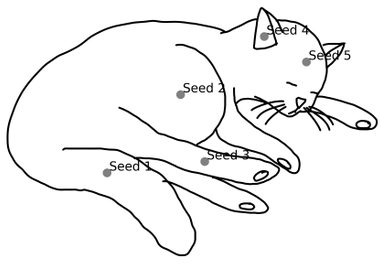
(b) $k = 4$, regions



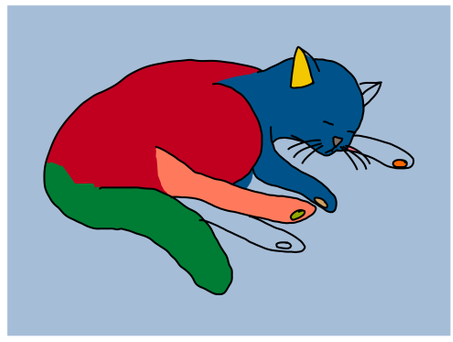
(a) $k = 5$, seeds



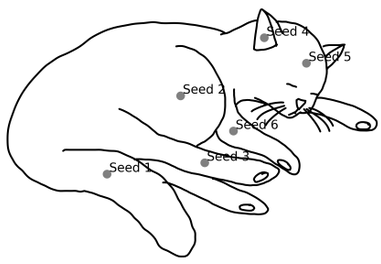
(b) $k = 5$, regions



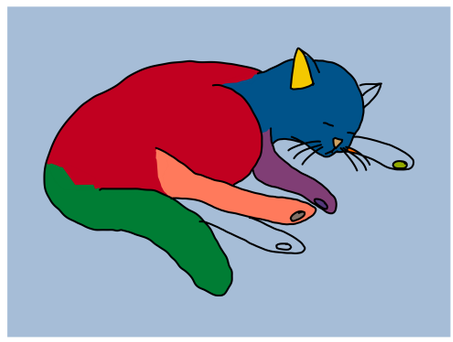
(a) $k = 6$, seeds



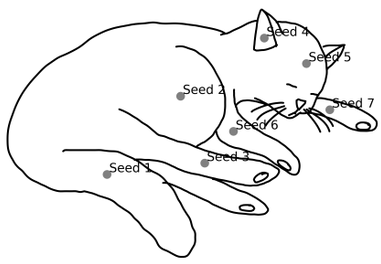
(b) $k = 6$, regions



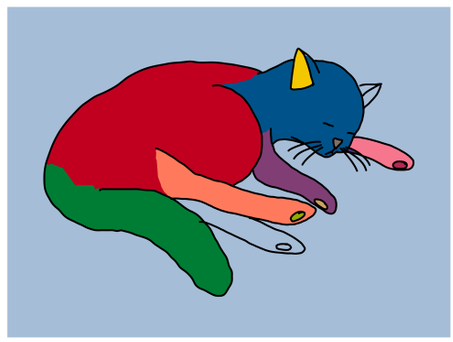
(a) $k = 7$, seeds



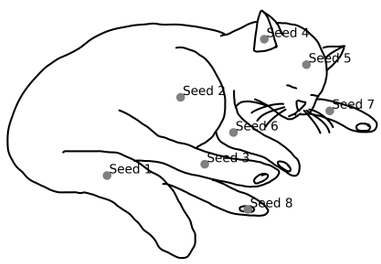
(b) $k = 7$, regions



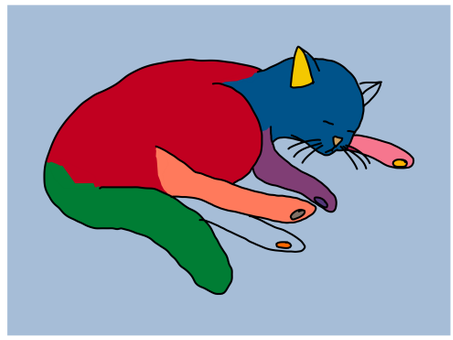
(a) $k = 8$, seeds



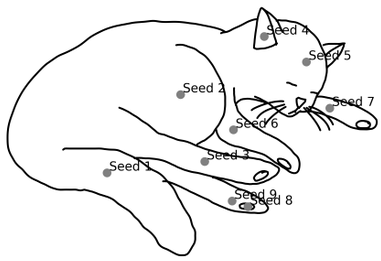
(b) $k = 8$, regions



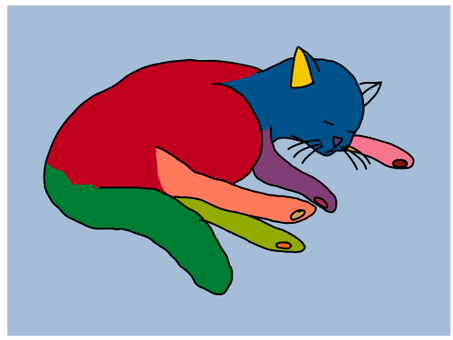
(a) $k = 9$, seeds



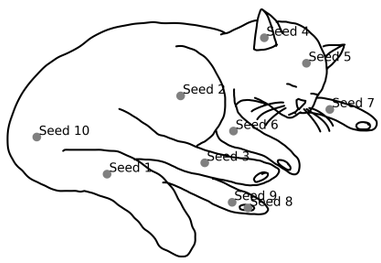
(b) $k = 9$, regions



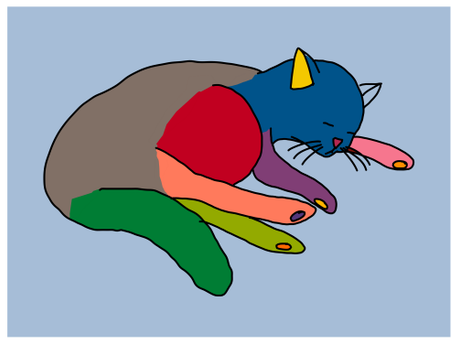
(a) $k = 10$, seeds



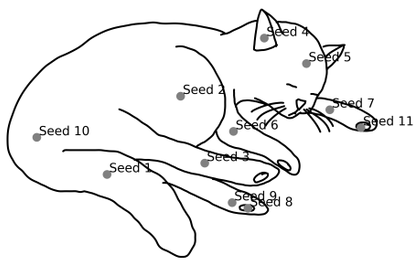
(b) $k = 10$, regions



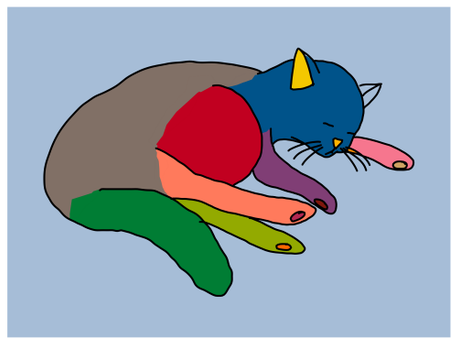
(a) $k = 11$, seeds



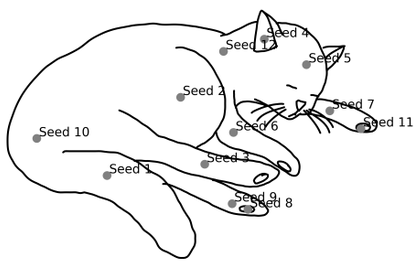
(b) $k = 11$, regions



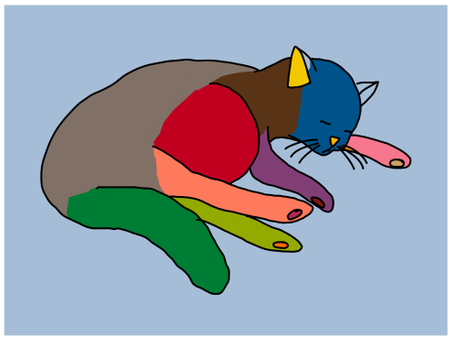
(a) $k = 12$, seeds



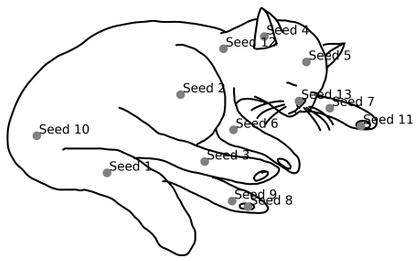
(b) $k = 12$, regions



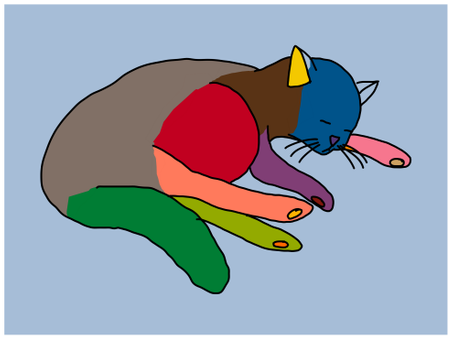
(a) $k = 13$, seeds



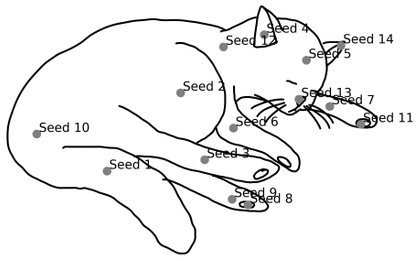
(b) $k = 13$, regions



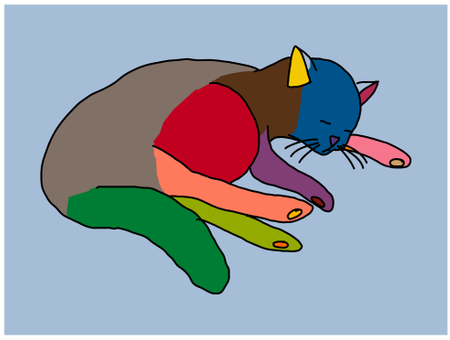
(a) $k = 14$, seeds



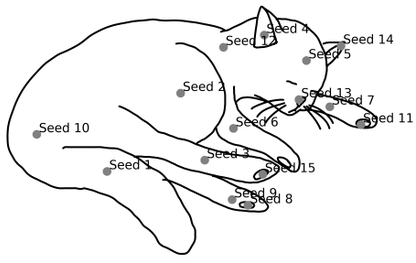
(b) $k = 14$, regions



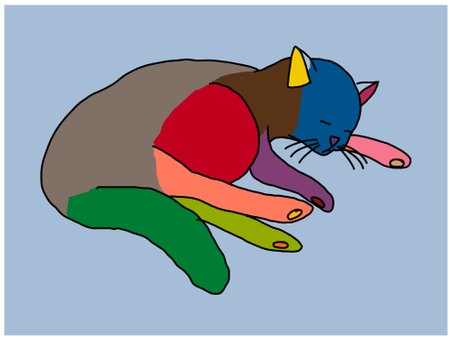
(a) $k = 15$, seeds



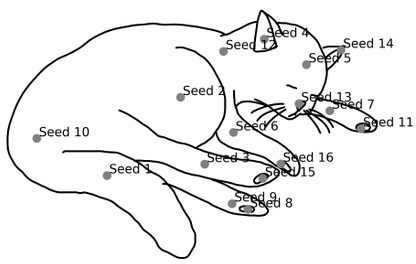
(b) $k = 15$, regions



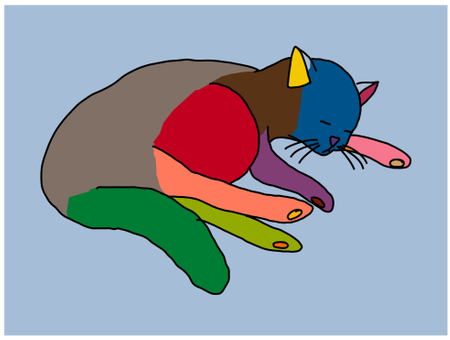
(a) $k = 16$, seeds



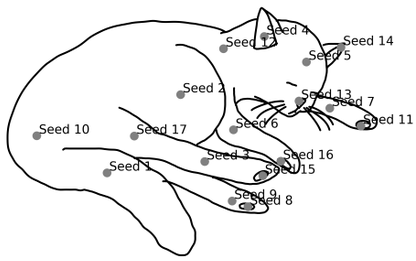
(b) $k = 16$, regions



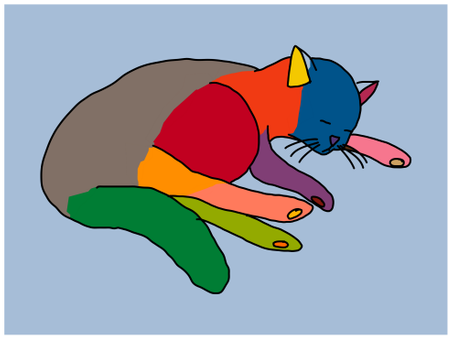
(a) $k = 17$, seeds



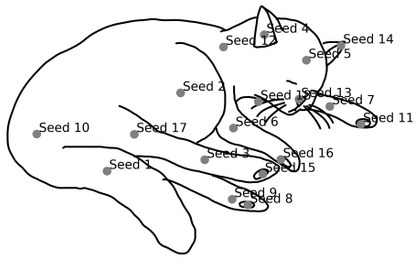
(b) $k = 17$, regions



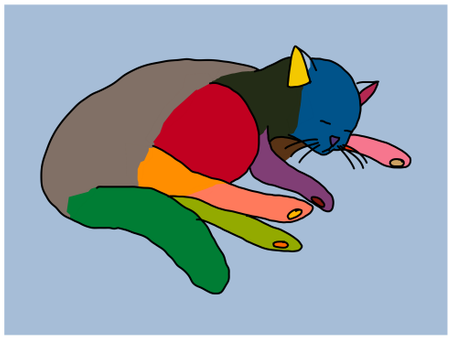
(a) $k = 18$, seeds



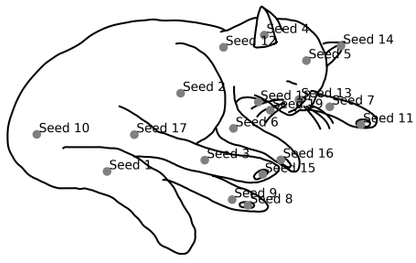
(b) $k = 18$, regions



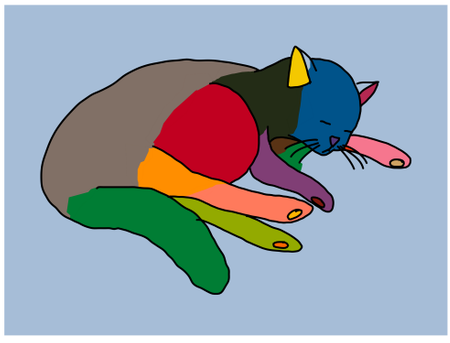
(a) $k = 19$, seeds



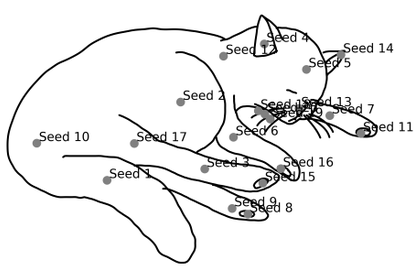
(b) $k = 19$, regions



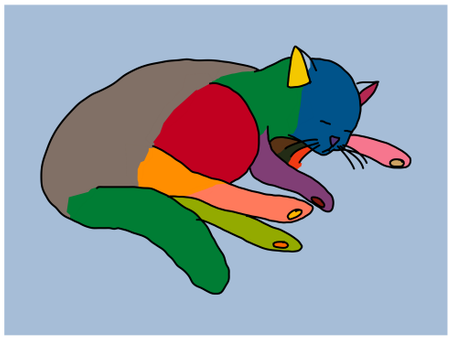
(a) $k = 20$, seeds



(b) $k = 20$, regions

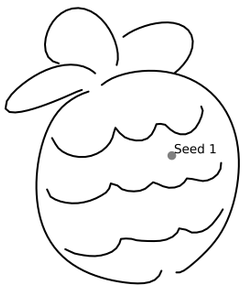


(a) $k = 21$, seeds

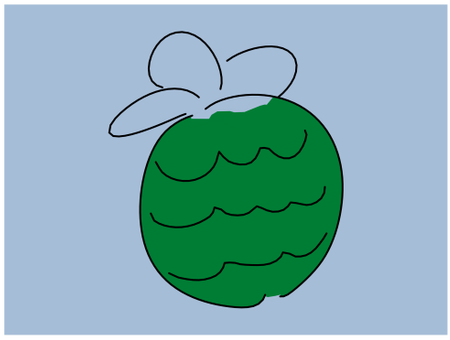


(b) $k = 21$, regions

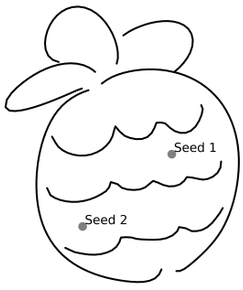
8.2 Sketch 2: Fruit



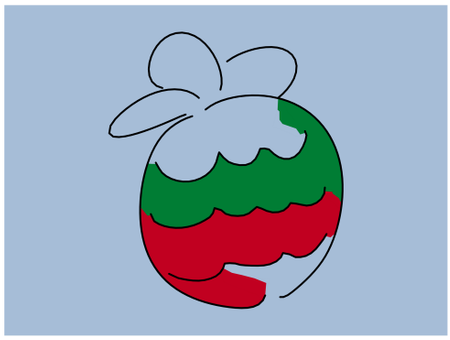
(a) $k = 2$, seeds



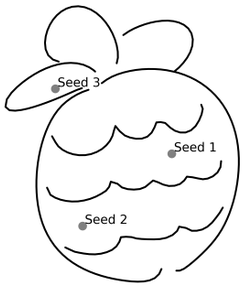
(b) $k = 2$, regions



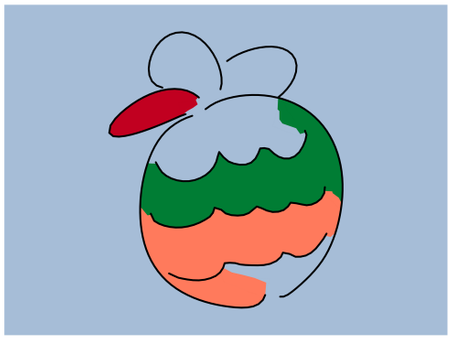
(a) $k = 3$, seeds



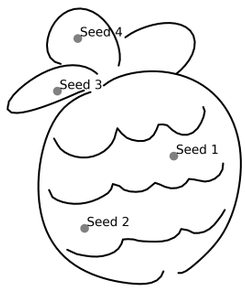
(b) $k = 3$, regions



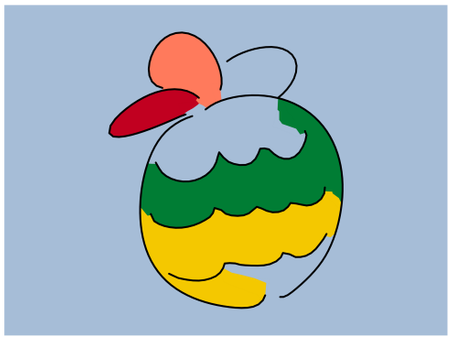
(a) $k = 4$, seeds



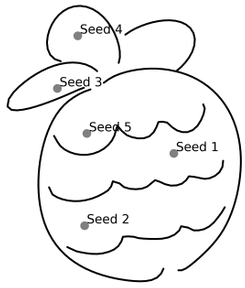
(b) $k = 4$, regions



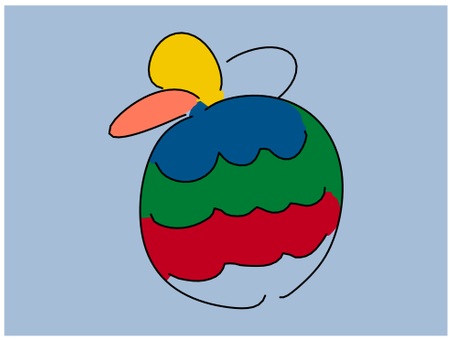
(a) $k = 5$, seeds



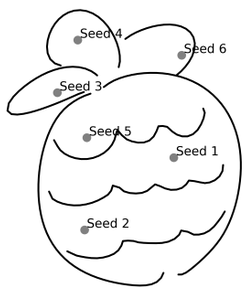
(b) $k = 5$, regions



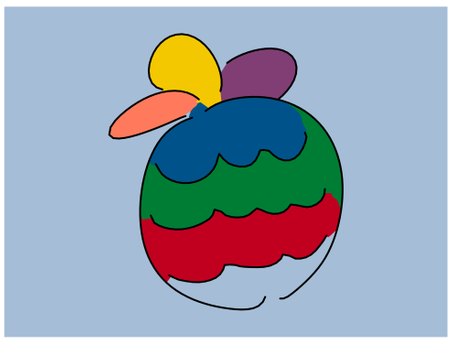
(a) $k = 6$, seeds



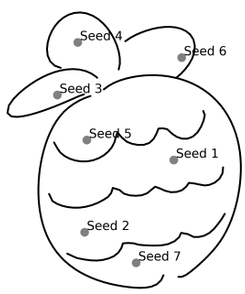
(b) $k = 6$, regions



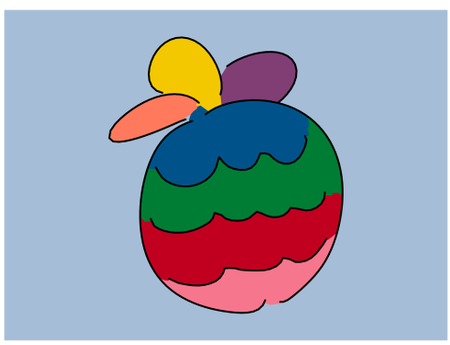
(a) $k = 7$, seeds



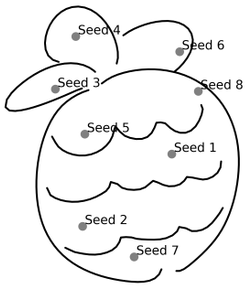
(b) $k = 7$, regions



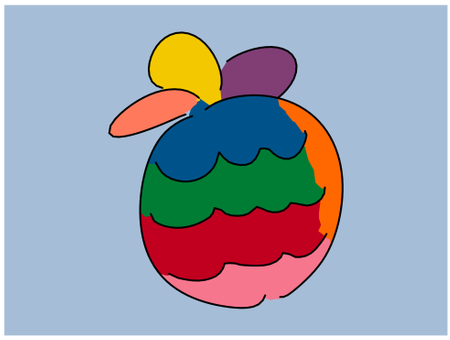
(a) $k = 8$, seeds



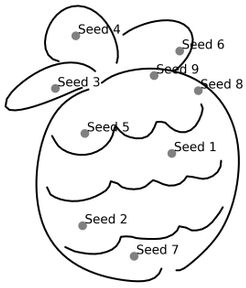
(b) $k = 8$, regions



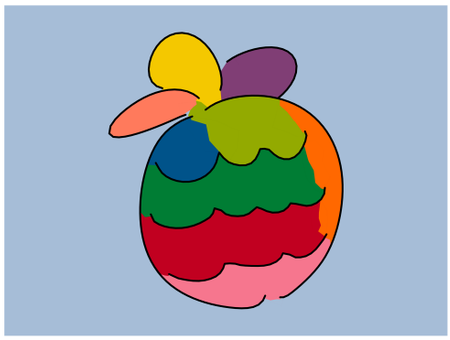
(a) $k = 9$, seeds



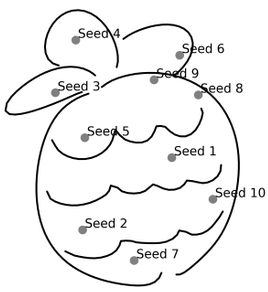
(b) $k = 9$, regions



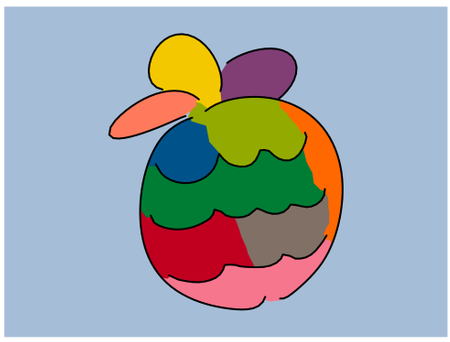
(a) $k = 10$, seeds



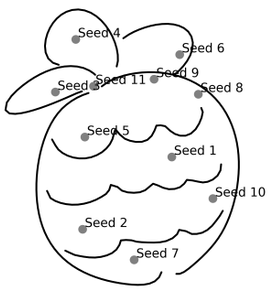
(b) $k = 10$, regions



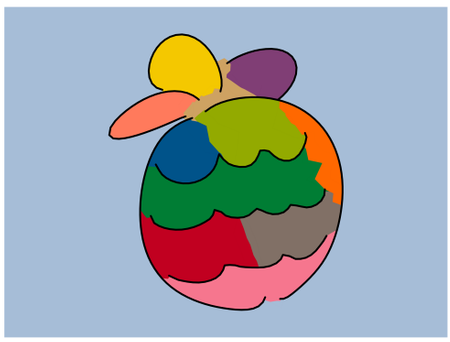
(a) $k = 11$, seeds



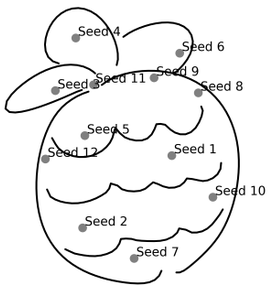
(b) $k = 11$, regions



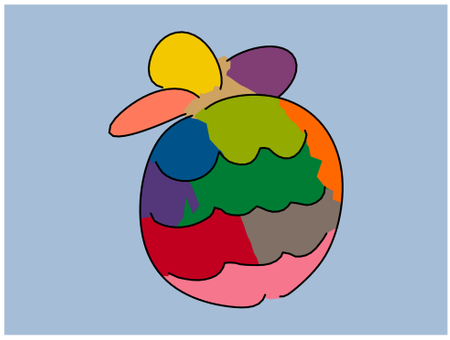
(a) $k = 12$, seeds



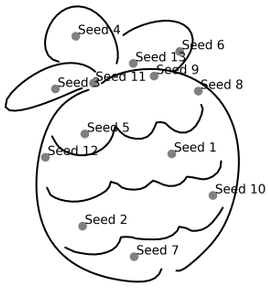
(b) $k = 12$, regions



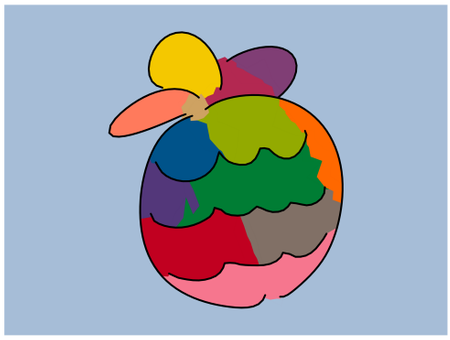
(a) $k = 13$, seeds



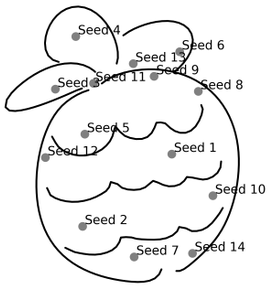
(b) $k = 13$, regions



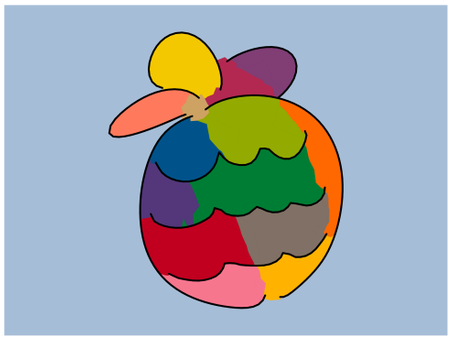
(a) $k = 14$, seeds



(b) $k = 14$, regions



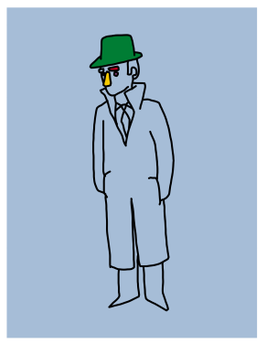
(a) $k = 15$, seeds



(b) $k = 15$, regions

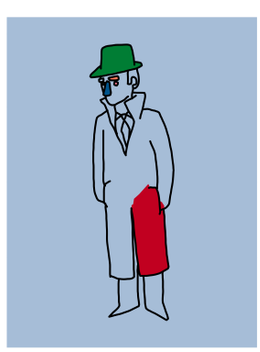
8.3 Sketch 3: Noir

(a) $k = 2$, seeds



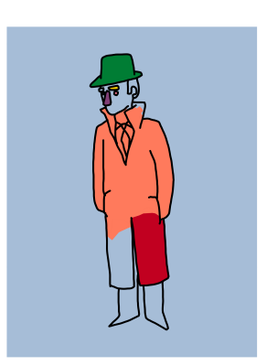
(b) $k = 2$, regions

(a) $k = 3$, seeds



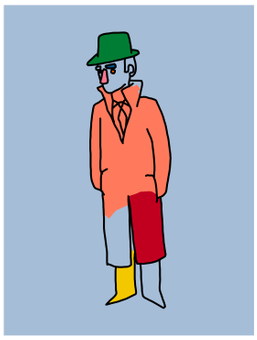
(b) $k = 3$, regions

(a) $k = 4$, seeds



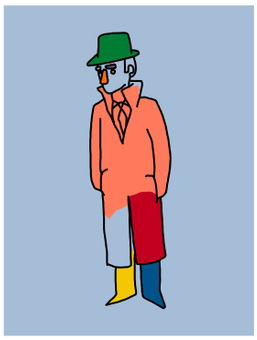
(b) $k = 4$, regions

(a) $k = 5$, seeds



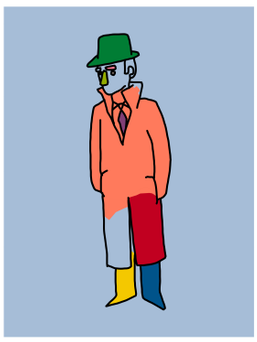
(b) $k = 5$, regions

(a) $k = 6$, seeds

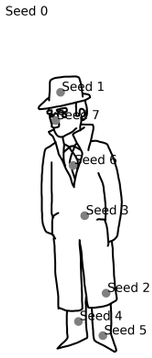


(b) $k = 6$, regions

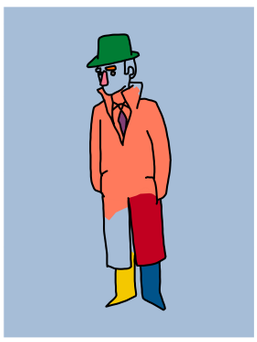
(a) $k = 7$, seeds



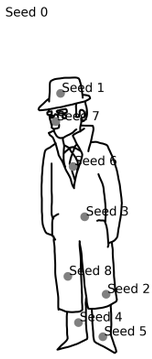
(b) $k = 7$, regions



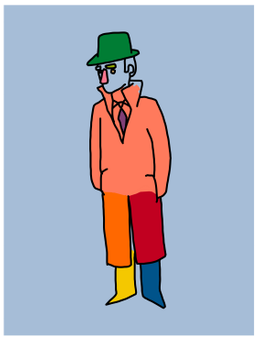
(a) $k = 8$, seeds



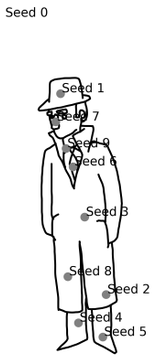
(b) $k = 8$, regions



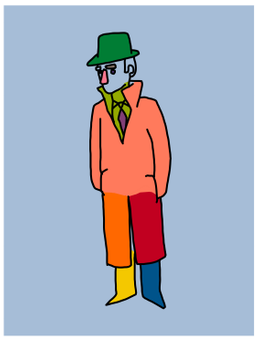
(a) $k = 9$, seeds



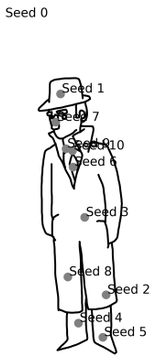
(b) $k = 9$, regions



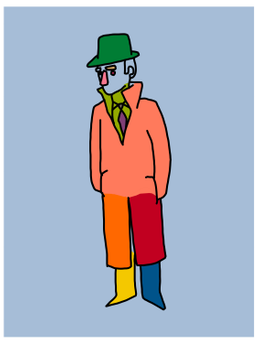
(a) $k = 10$, seeds



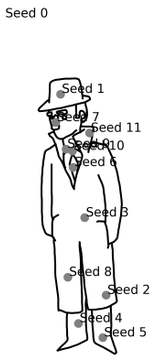
(b) $k = 10$, regions



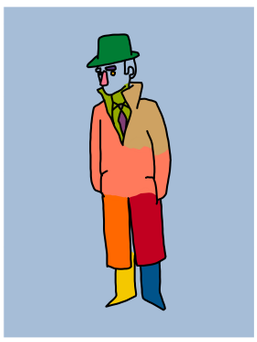
(a) $k = 11$, seeds



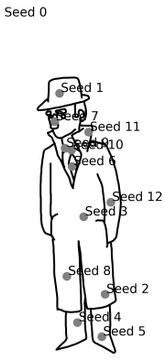
(b) $k = 11$, regions



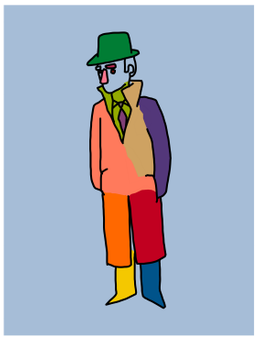
(a) $k = 12$, seeds



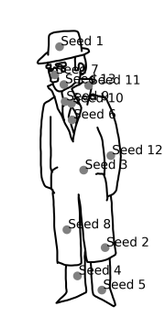
(b) $k = 12$, regions



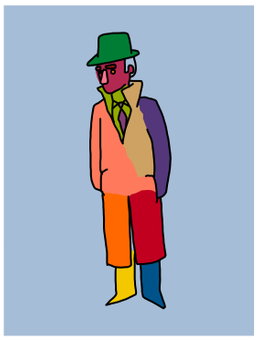
(a) $k = 13$, seeds



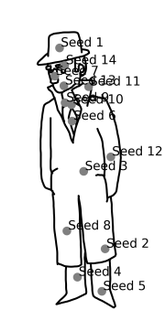
(b) $k = 13$, regions



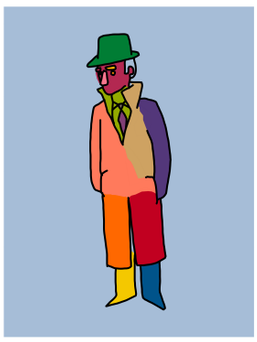
(a) $k = 14$, seeds



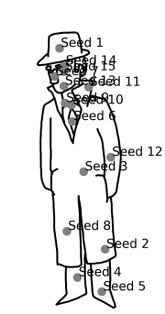
(b) $k = 14$, regions



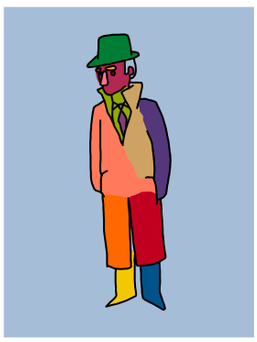
(a) $k = 15$, seeds



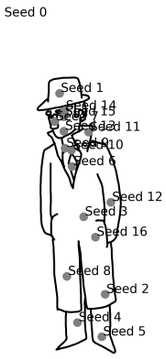
(b) $k = 15$, regions



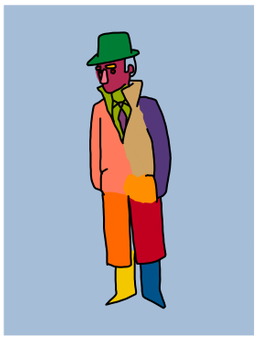
(a) $k = 16$, seeds



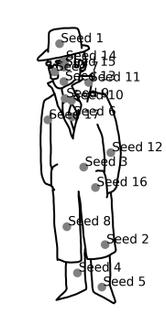
(b) $k = 16$, regions



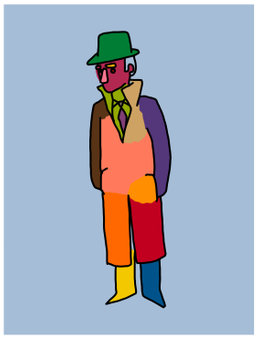
(a) $k = 17$, seeds



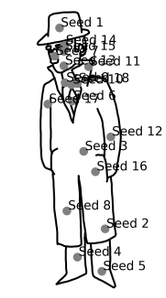
(b) $k = 17$, regions



(a) $k = 18$, seeds

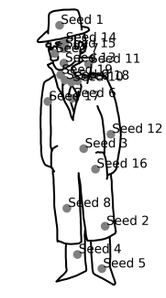


(b) $k = 18$, regions

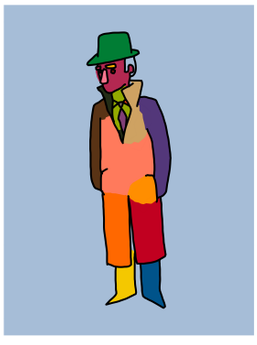


(a) $k = 19$, seeds

(b) $k = 19$, regions



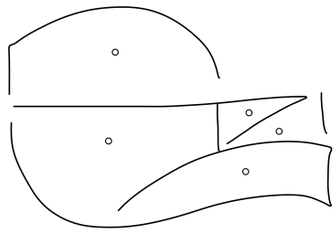
(a) $k = 20$, seeds

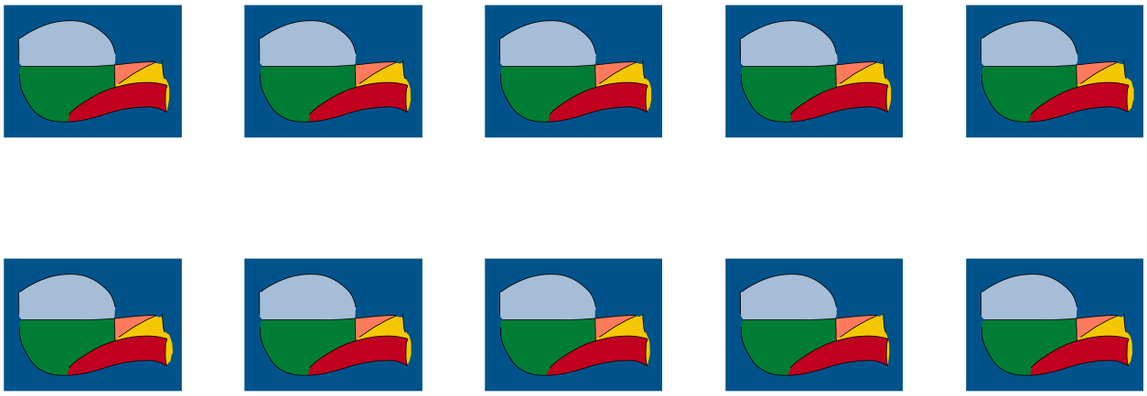


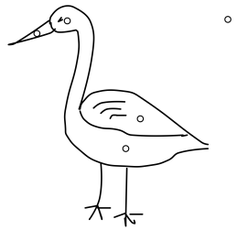
(b) $k = 20$, regions

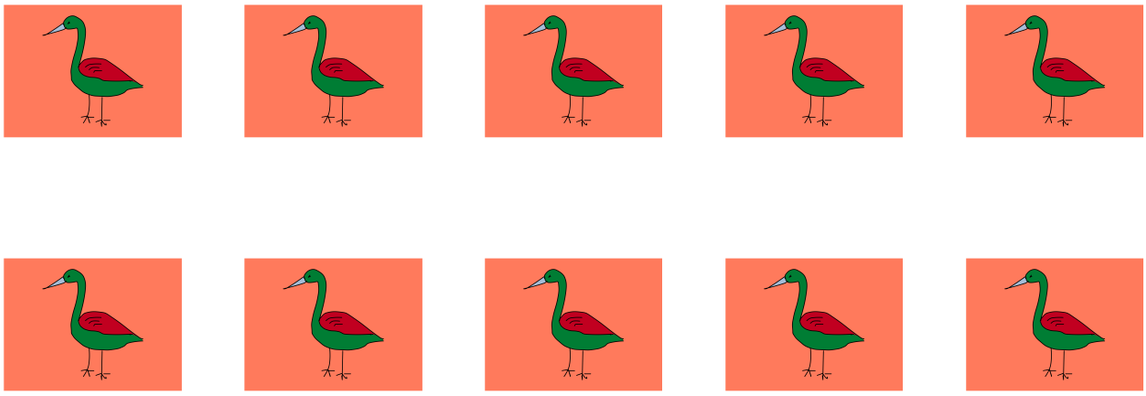
9 Varying Randomized Features

Sampling many random stroke configurations and weighting features by total variation both allow us to maintain consistency in our random feature generation. Below, we provide results from 10 random stroke configurations on 3 sample sketches, which we observe are highly stable. We fix a set of user hints for each sketch.

















10 Varying User Hints

We analyze the effect of perturbing a given set of user-provided hints, and observe that the results are robust to slight variations in user input. Hints from the base set are specified on triangle barycenters; to perturb hints, we select a random neighboring triangle's barycenter. 10 results are shown for each of 3 sample sketches.

10.1 Sketch 1: Abstract

10.2 Sketch 2: Crane

10.3 Sketch 3: Snowman

11 Spectral Clustering

A natural suggestion is to implement basic spectral clustering on the sketches. We find that this does not perform well, as the algorithm seeks to identify clusters that are approximately the same size. This undermines the method's ability to segment the sketch based on stroke configurations, leading to oversegmentation of the background and 'bleeding' of regions into neighboring ones.

Based on testing conducted using the cotangent Laplacian and running k-means++ clustering on k+1 eigenfunctions, we can see that this method does not suit our purpose. Note that the cotangent Laplacian has eigenvalue 0 with multiplicity equal to the number of connected components of the sketch, so spectral clustering will always identify connected regions as clusters, regardless of the size of the region. However, with 'gappy' sketches, the regions are subject to the algorithm's preference towards clusters of the same size. Intuitively, this can be explained by the relationship between spectral clustering and graph partitioning algorithms, as is detailed in Section 5 of [Lux07]. In particular, spectral clustering with the cotangent Laplacian, normalized by an area component of a vertex, is analogous to normalized cut, which aims to partition the graph into portions of approximately the same size. The examples below show how this leads to undesirable segmentation.

(a) k=16

(b) k=8

In the left hand result, the strokes defining the face, hair, and background are ignored in order to prioritize evenly sized clusters, which in turn causes oversegmentation of the background. Note that the connected components that do not contain the background — the right hand, buttons, and collar — are not subject to this balanced cluster behavior because they have eigenvalue 0, and are independent of the other regions. The right hand result shows bleeding between 'gappy' regions and the inability of the method to distinguish between the background and internal part of the sketch.

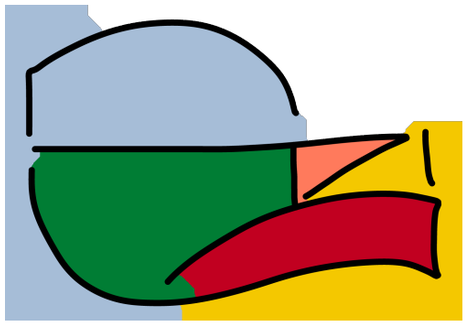
12 Additional Results

12.1 Semi-Automatic Methods

12.1.1 Methods with stroke-based input

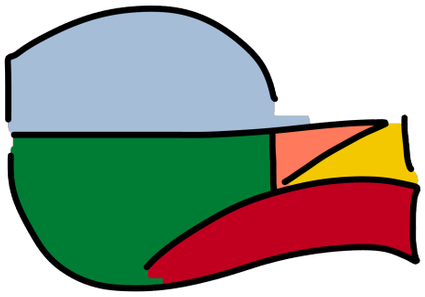
Below are comparison results for our semi-automatic method against the stroke-based methods of [SDC09] and [FTR18] on the remaining sketches that are not included in the main paper (elephant, penguin, snowman are in Fig. 8).

(a) Krita (Sýkora et al. 2009), hints

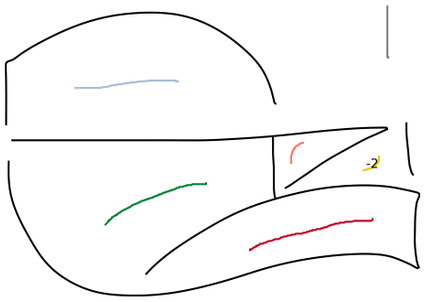


(b) Krita (Sýkora et al. 2009), results

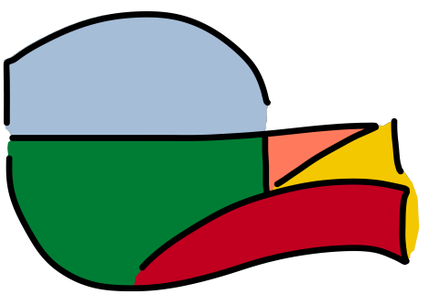
(a) G'MIC (Fourey et al. 2018), hints



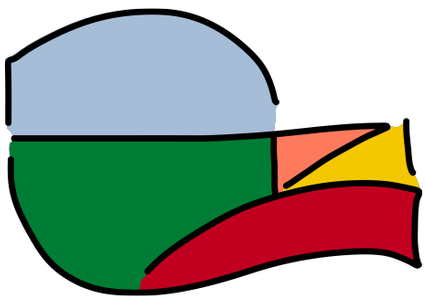
(b) G'MIC (Fourey et al. 2018), results



Ours, hints



(a) Ours, results (no strengths)



(b) Ours, results (with strengths)

(a) Krita (Sýkora et al. 2009), hints

(b) Krita (Sýkora et al. 2009), results

(a) G'MIC (Fourey et al. 2018), hints

(b) G'MIC (Fourey et al. 2018), results

Ours, hints

(a) Ours, results (no strengths)

(b) Ours, results (with strengths)

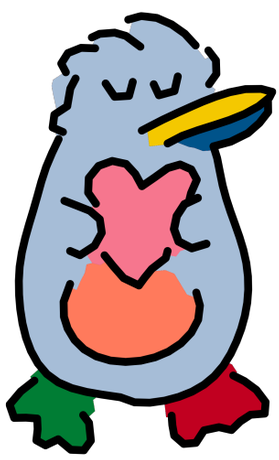
(a) Krita (Sýkora et al. 2009), hints

(b) Krita (Sýkora et al. 2009), results

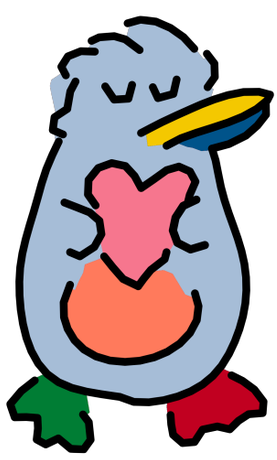
(c) G'MIC (Fourey et al. 2018), hints

(d) G'MIC (Fourey et al. 2018), results

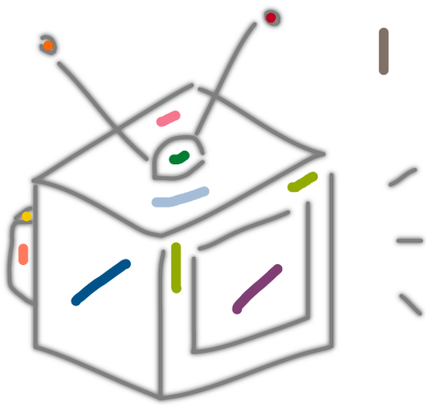
Ours, hints



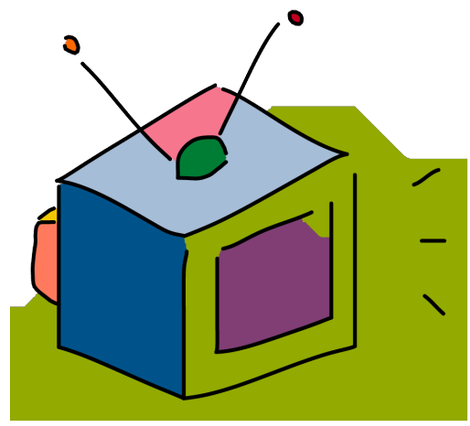
(a) Ours, results (no strengths)



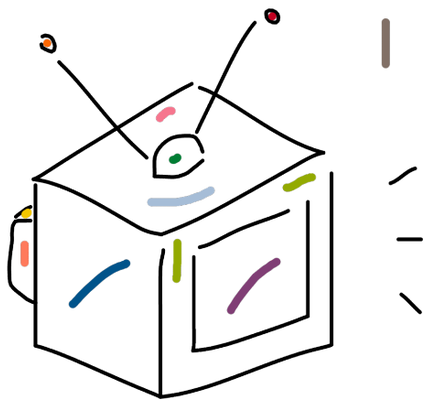
(b) Ours, results (with strengths)



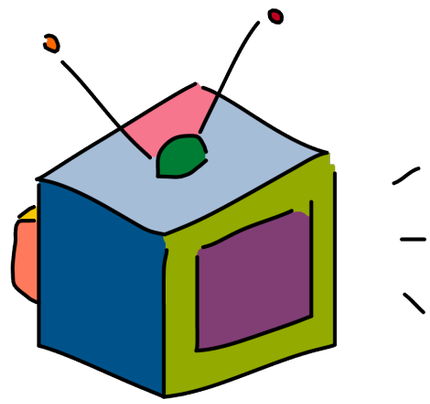
(a) Krita (Sýkora et al. 2009), hints



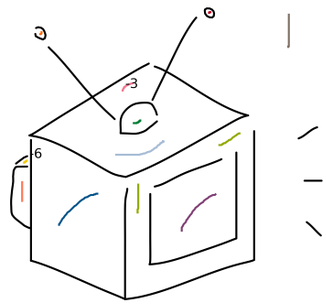
(b) Krita (Sýkora et al. 2009), results



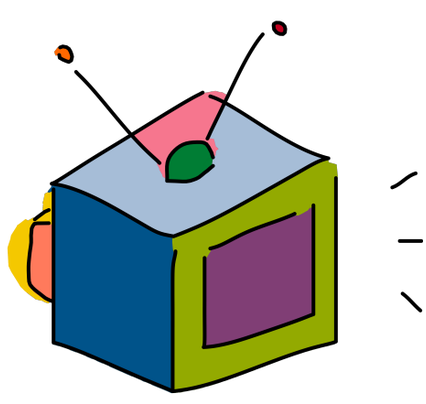
(a) G'MIC (Fourey et al. 2018), hints



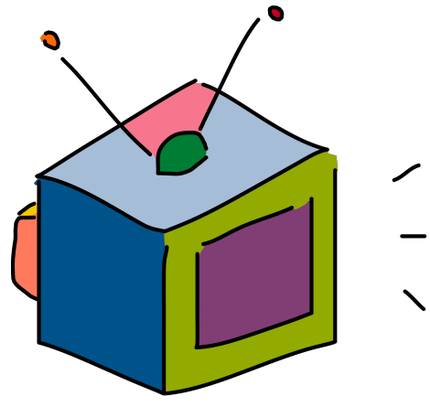
(b) G'MIC (Fourey et al. 2018), results



Ours, hints



(a) Ours, results (no strengths)



(b) Ours, results (with strengths)

(a) Krita (Sýkora et al. 2009), hints

(b) Krita (Sýkora et al. 2009), results

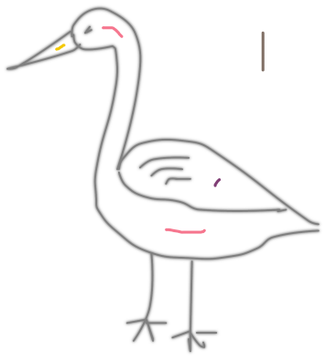
(a) G'MIC (Fourey et al. 2018), hints

(b) G'MIC (Fourey et al. 2018), results

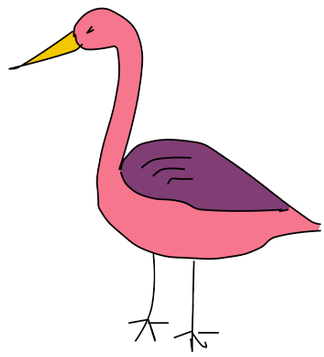
Ours, hints

(a) Ours, results (no strengths)

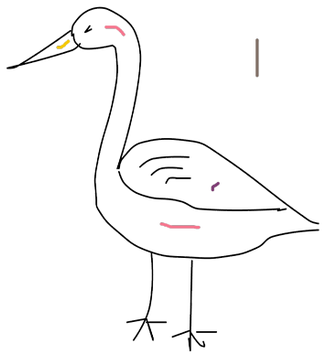
(b) Ours, results (with strengths)



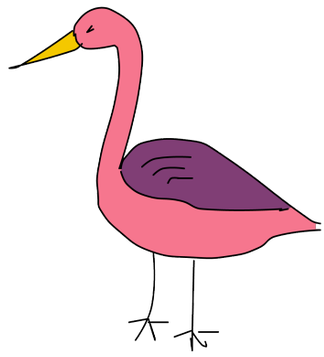
(a) Krita (Sýkora et al. 2009), hints



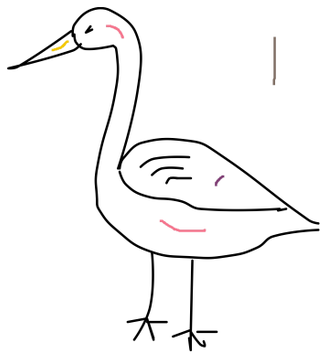
(b) Krita (Sýkora et al. 2009), results



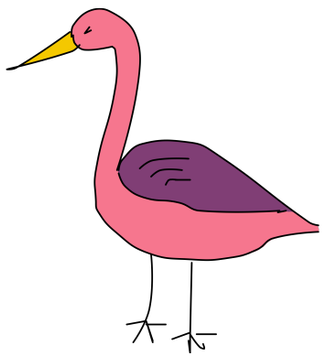
(a) G'MIC (Fourey et al. 2018), hints



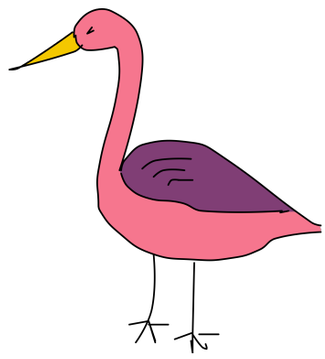
(b) G'MIC (Fourey et al. 2018), results



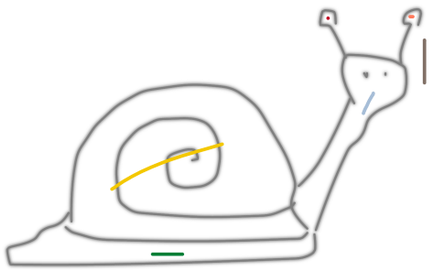
Ours, hints



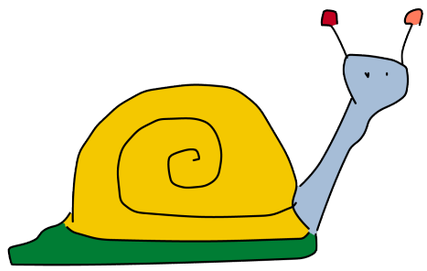
(a) Ours, results (no strengths)



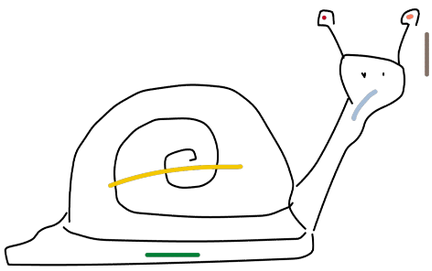
(b) Ours, results (with strengths)



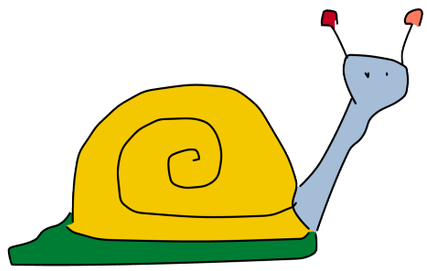
(a) Krita (Sýkora et al. 2009), hints



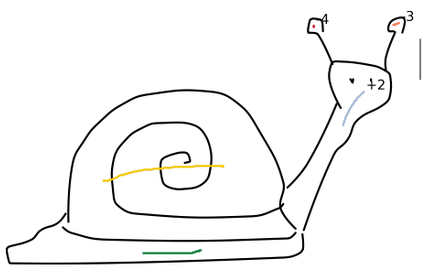
(b) Krita (Sýkora et al. 2009), results



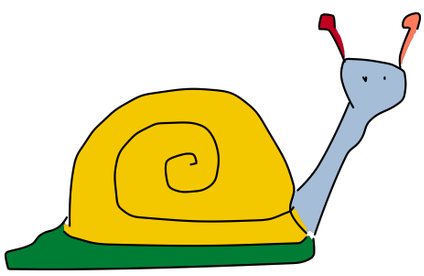
(a) G'MIC (Fourey et al. 2018), hints



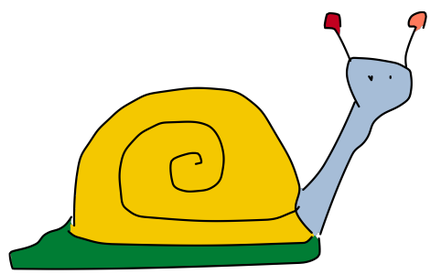
(b) G'MIC (Fourey et al. 2018), results



Ours, hints

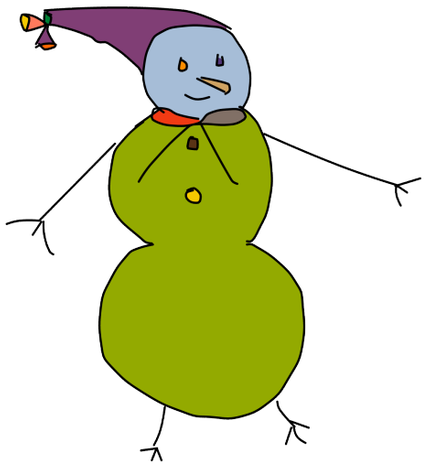


(a) Ours, results (no strengths)



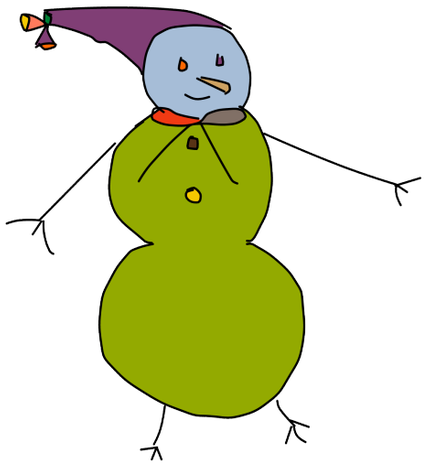
(b) Ours, results (with strengths)

(a) Krita (Sýkora et al. 2009), hints



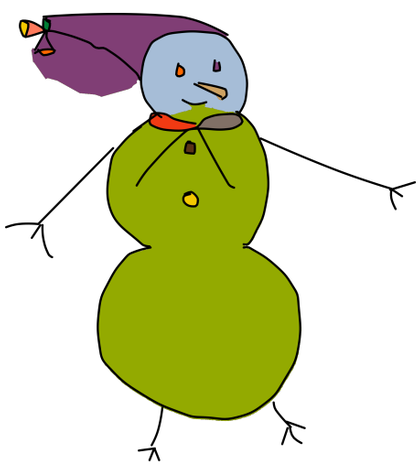
(b) Krita (Sýkora et al. 2009), results

(a) G'MIC (Fourey et al. 2018), hints

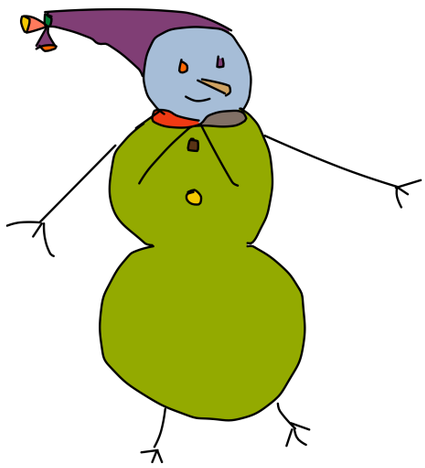


(b) G'MIC (Fourey et al. 2018), results

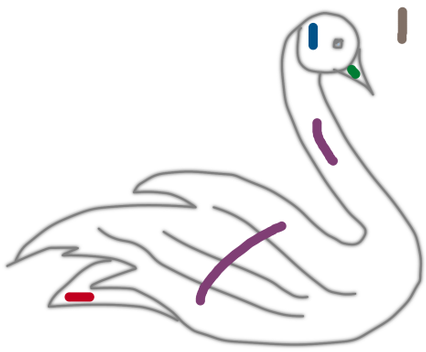
Ours, hints



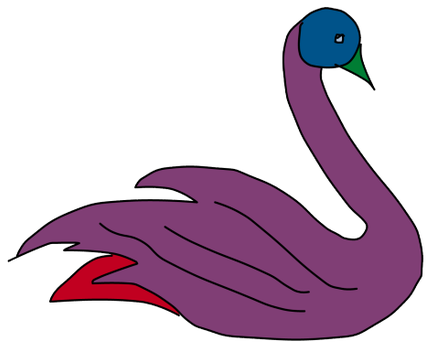
(a) Ours, results (no strengths)



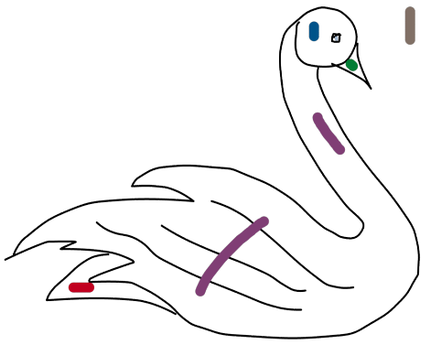
(b) Ours, results (with strengths)



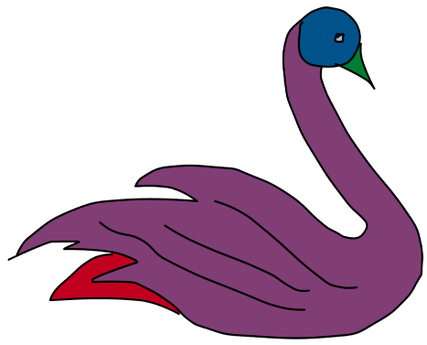
(a) Krita (Sýkora et al. 2009), hints



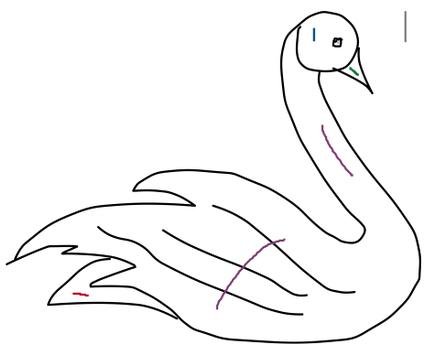
(b) Krita (Sýkora et al. 2009), results



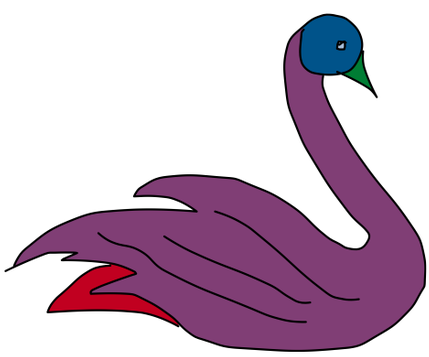
(a) G'MIC (Fourey et al. 2018), hints



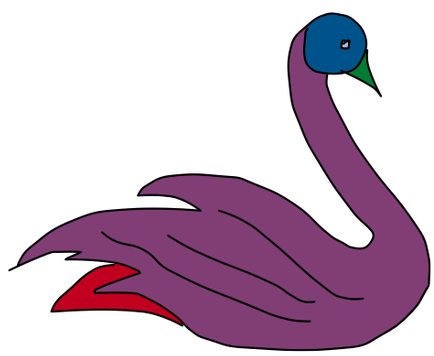
(b) G'MIC (Fourey et al. 2018), results



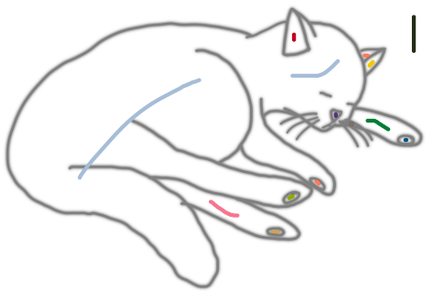
Ours, hints



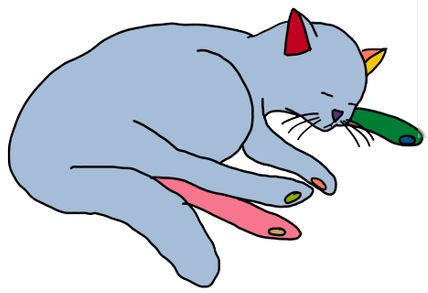
(a) Ours, results (no strengths)



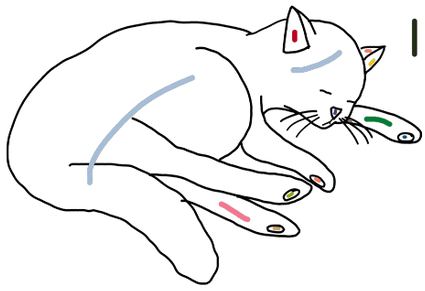
(b) Ours, results (with strengths)



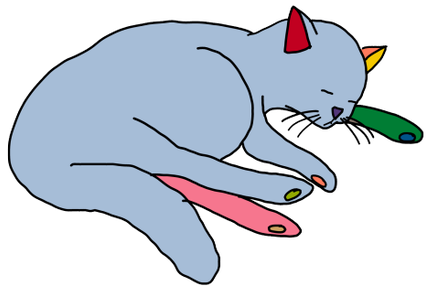
(a) Krita (Sýkora et al. 2009), hints



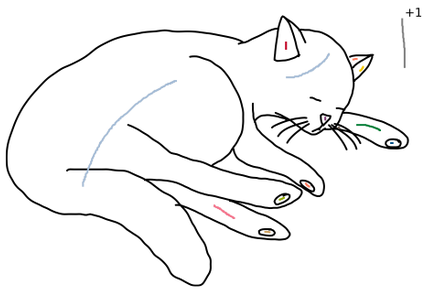
(b) Krita (Sýkora et al. 2009), results



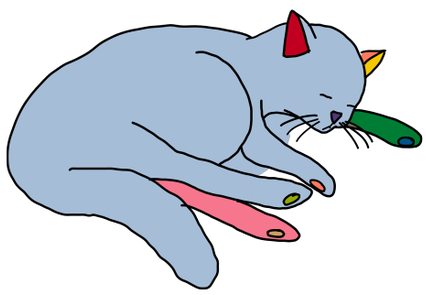
(a) G'MIC (Fourey et al. 2018), hints



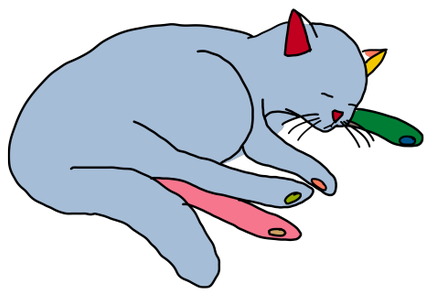
(b) G'MIC (Fourey et al. 2018), results



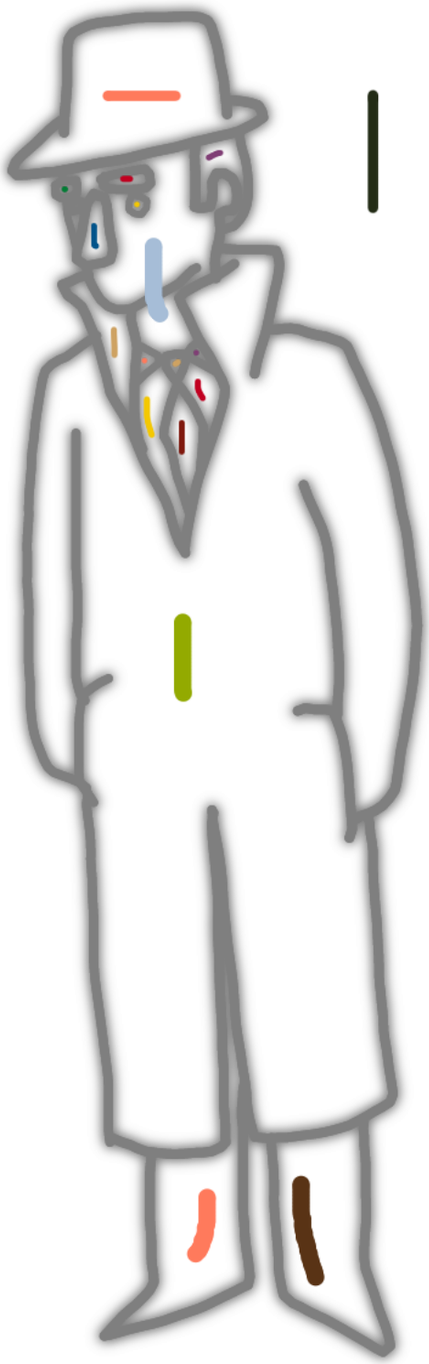
Ours, hints



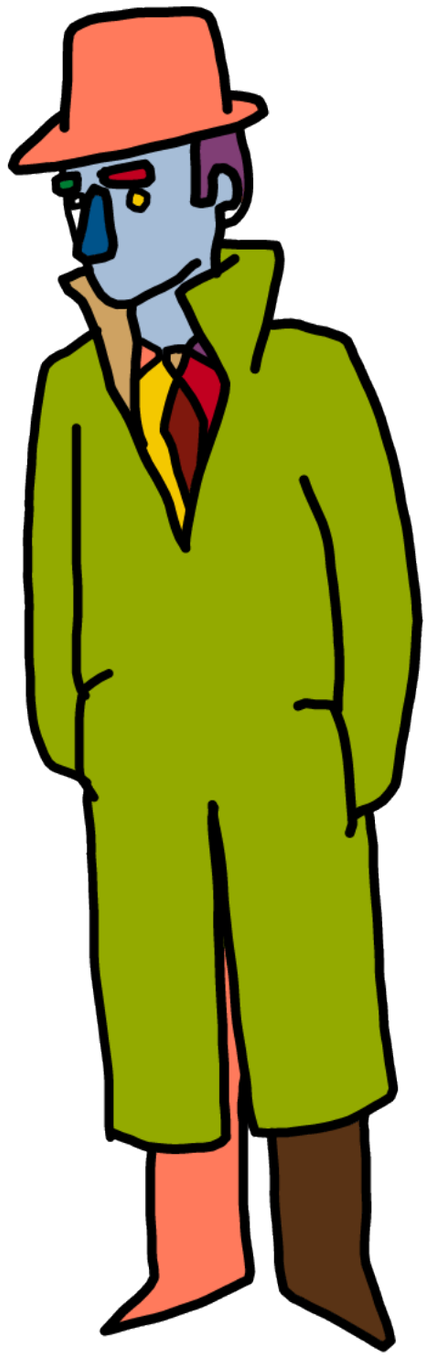
(a) Ours, results (no strengths)



(b) Ours, results (with strengths)

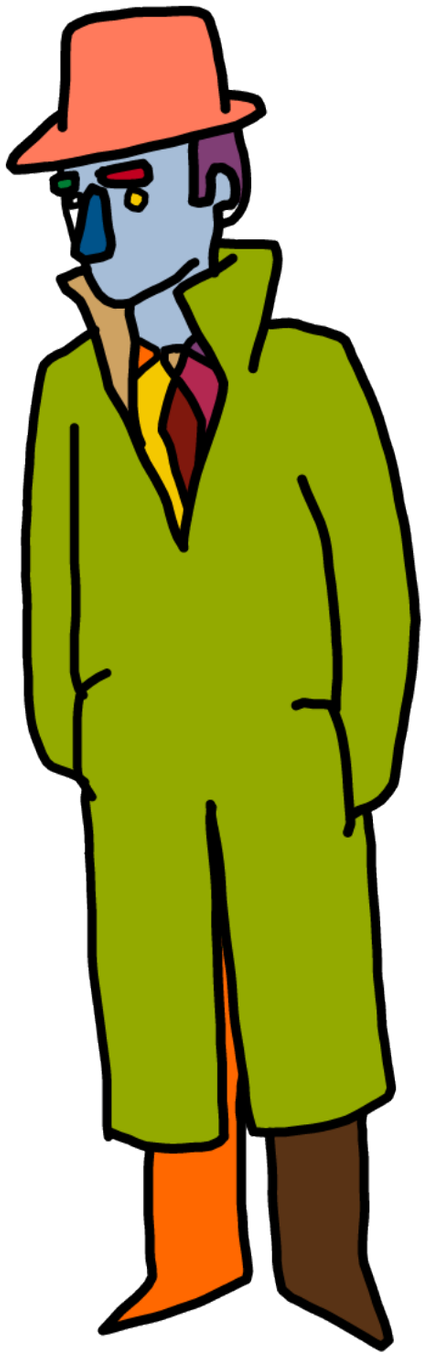


(a) Krita (Sýkora et al. 2009), hints



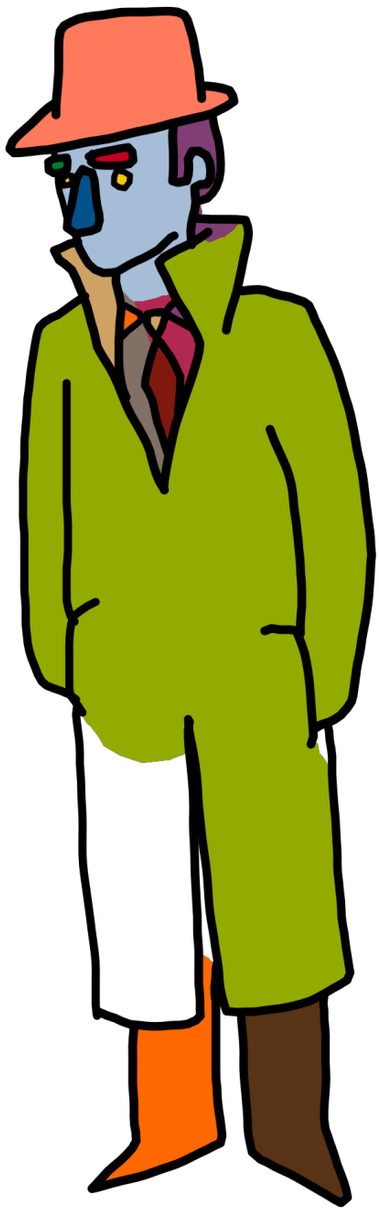
(b) Krita (Sýkora et al. 2009), results

(a) G'MIC (Fourey et al. 2018), hints

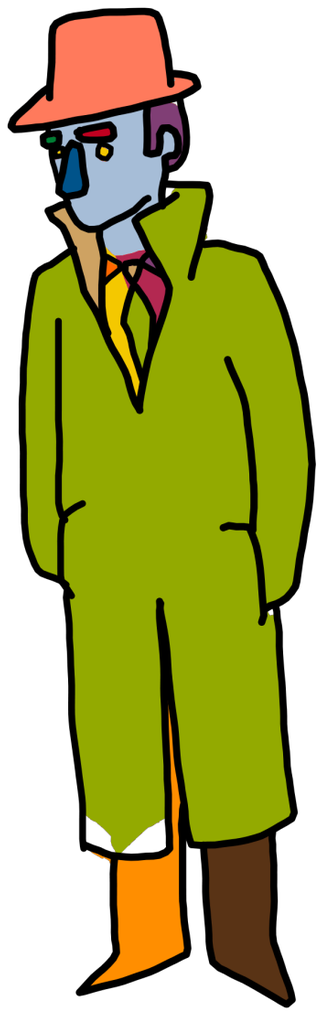


(b) G'MIC (Fourey et al. 2018), results

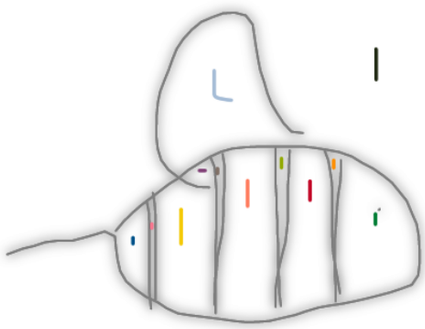
Ours, hints



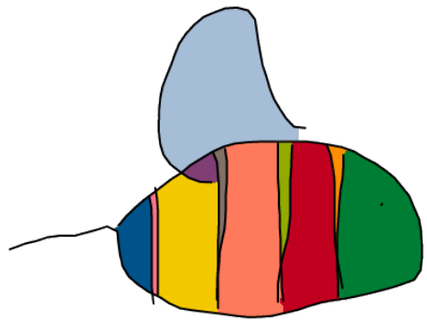
(a) Ours, results (no strengths)



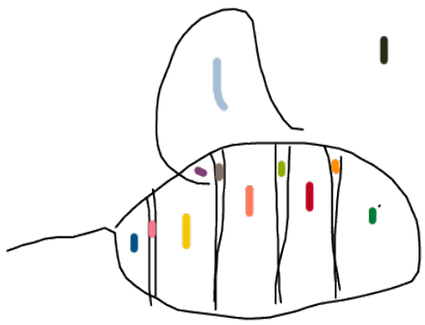
(b) Ours, results (with strengths)



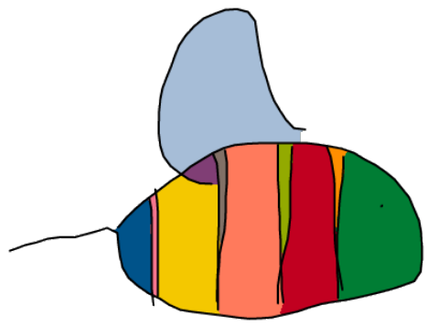
(a) Krita (Sýkora et al. 2009), hints



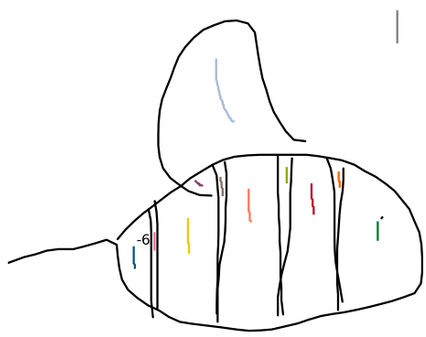
(b) Krita (Sýkora et al. 2009), results



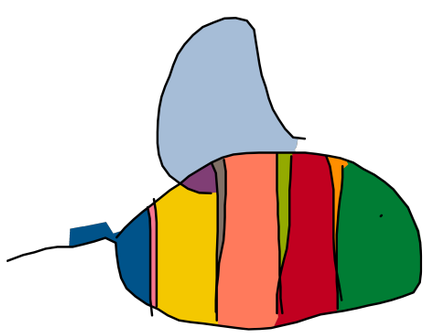
(a) G'MIC (Fourey et al. 2018), hints



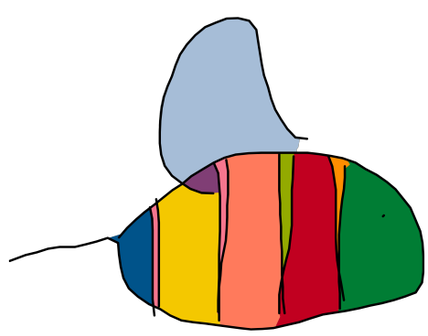
(b) G'MIC (Fourey et al. 2018), results



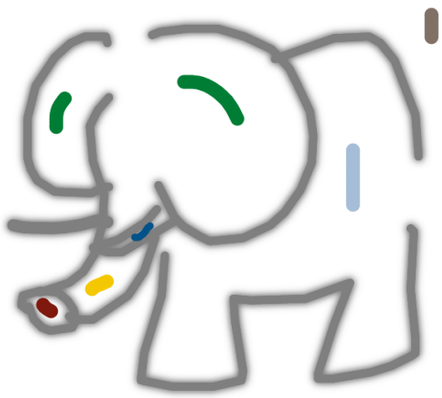
Ours, hints



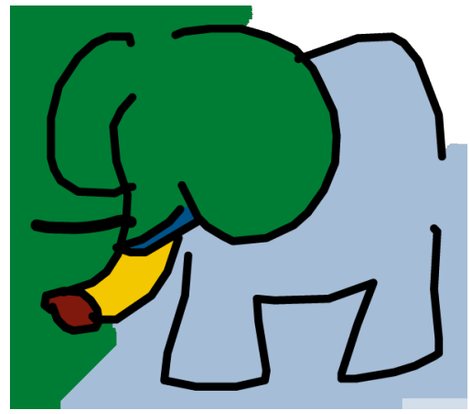
(a) Ours, results (no strengths)



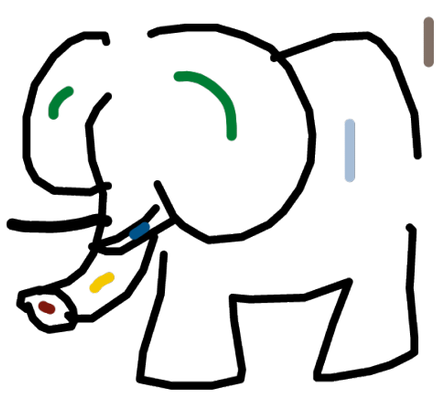
(b) Ours, results (with strengths)



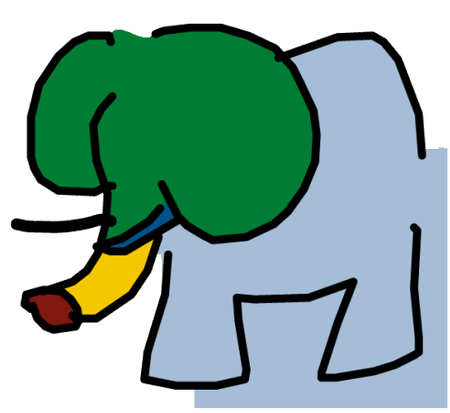
(a) Krita (Sýkora et al. 2009), hints



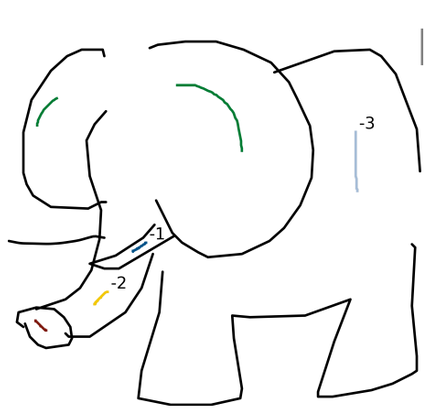
(b) Krita (Sýkora et al. 2009), results



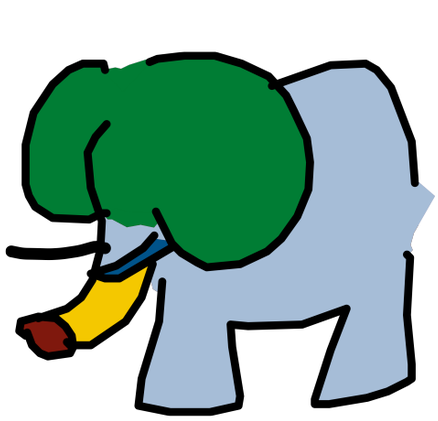
(a) G'MIC (Fourey et al. 2018), hints



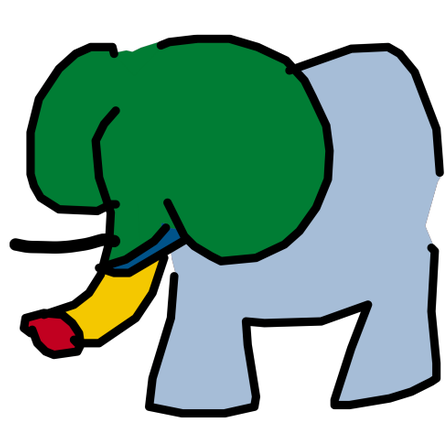
(b) G'MIC (Fourey et al. 2018), results



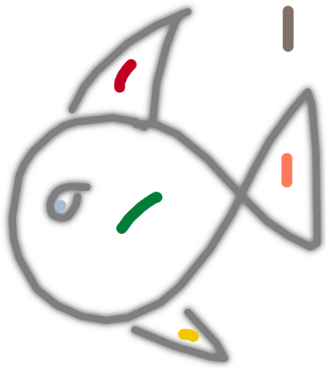
Ours, hints



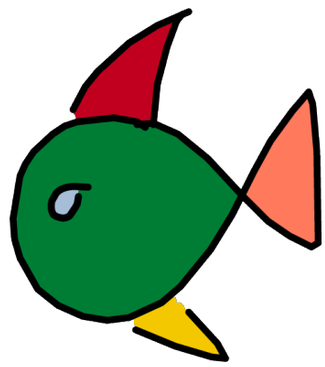
(a) Ours, results (no strengths)



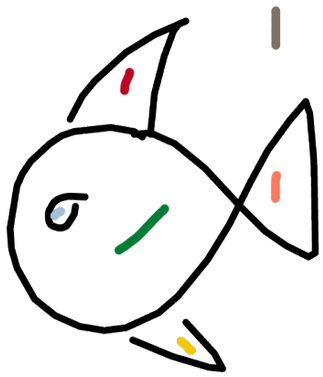
(b) Ours, results (with strengths)



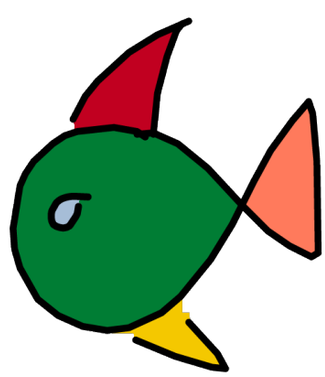
(a) Krita (Sýkora et al. 2009), hints



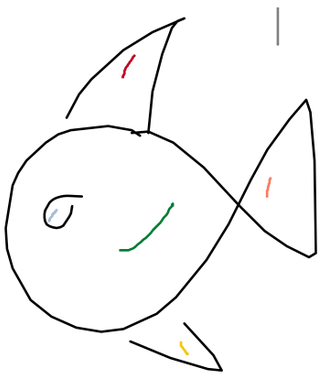
(b) Krita (Sýkora et al. 2009), results



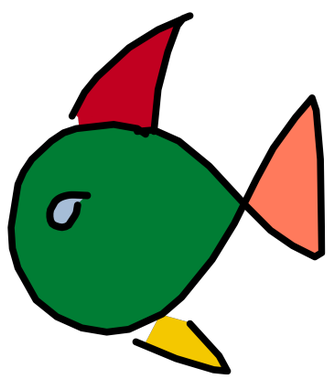
(a) G'MIC (Fourey et al. 2018), hints



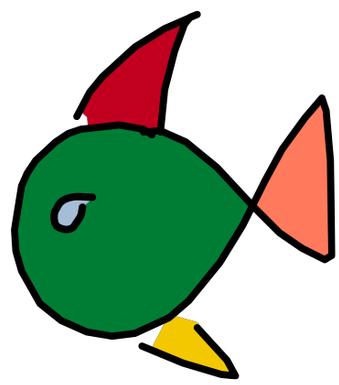
(b) G'MIC (Fourey et al. 2018), results



(a) Ours, hints

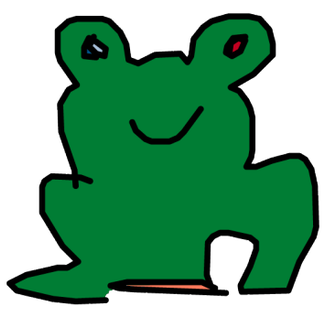


(b) Ours, results (no strengths)

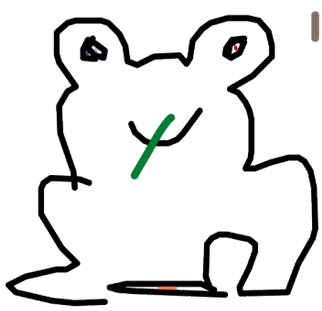


(c) Ours, results (with strengths)

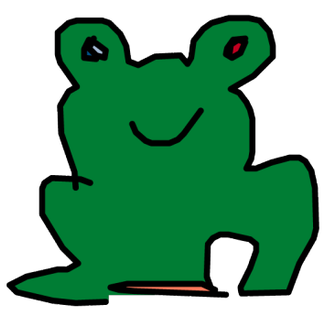
(a) Krita (Sýkora et al. 2009), hints



(b) Krita (Sýkora et al. 2009), results



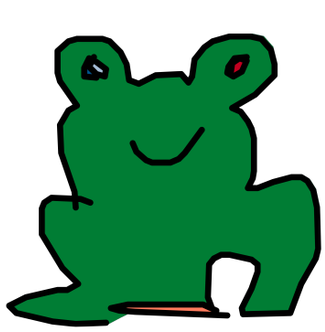
(a) G'MIC (Fourey et al. 2018), hints



(b) G'MIC (Fourey et al. 2018), results

Ours, hints

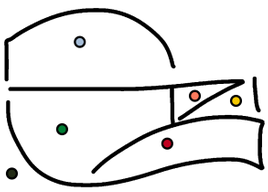
(a) Ours, results (no strengths)



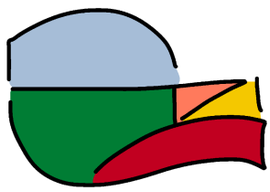
(b) Ours, results (with strengths)

12.1.2 Methods with point-based input

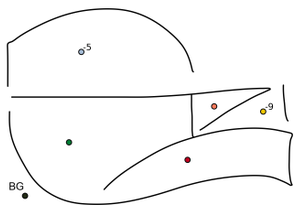
Below are comparison results for our semi-automatic method against the point-and-click based method of [PMC22] on the remaining sketches that are not included in the main paper (swan, snowman, television are in Fig. 9).



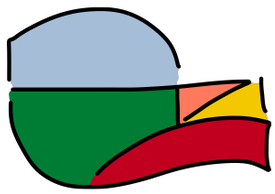
(a) Parakkat et al. (2022), hints



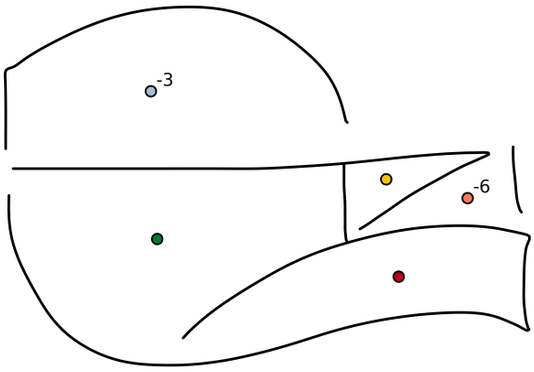
(b) Parakkat et al. (2022), results



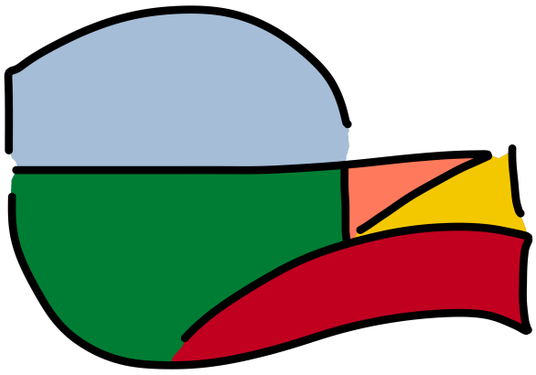
(c) Ours, hints (with strengths)



(d) Ours, results



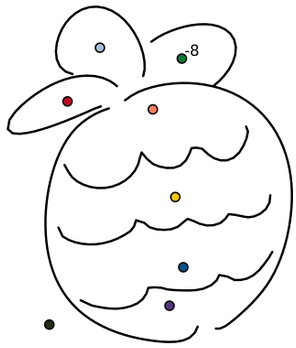
(a) Ours, hints (alt.)



(b) Ours, results (alt.)

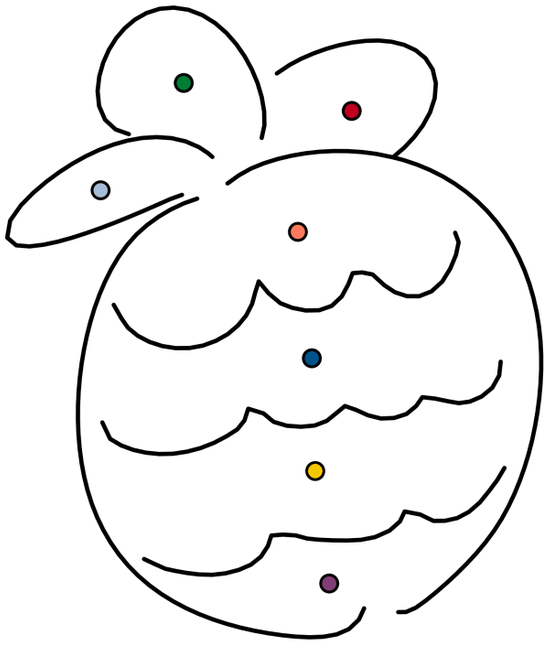
(a) Parakkat et al. (2022), hints

(b) Parakkat et al. (2022), results

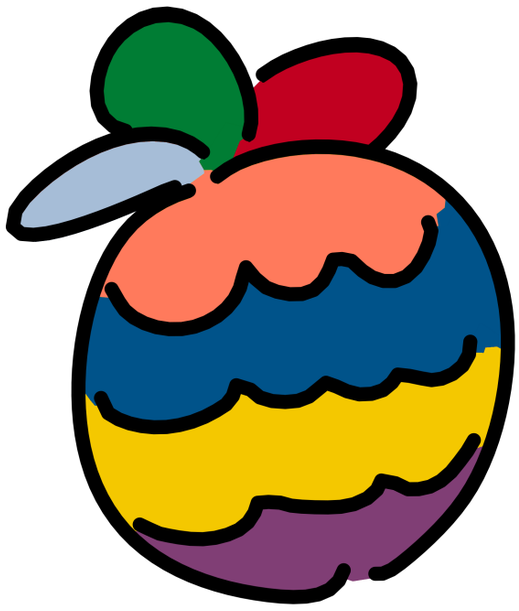


(c) Ours, hints (with strengths)

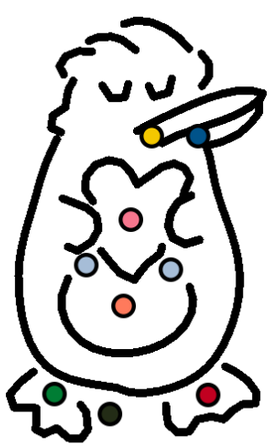
(d) Ours, results



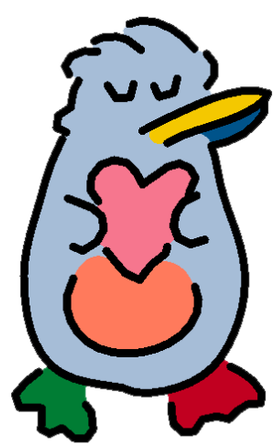
(a) Ours, hints (alt.)



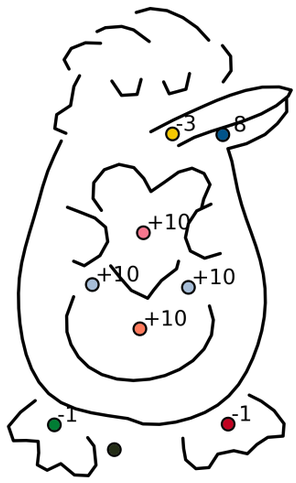
(b) Ours, results (alt.)



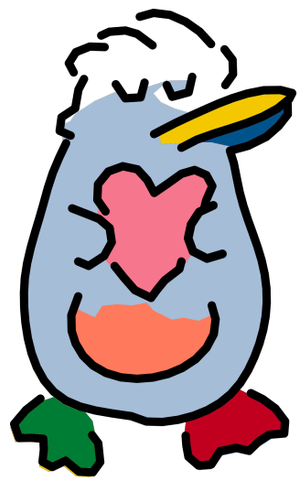
(a) Parakkat et al. (2022), hints



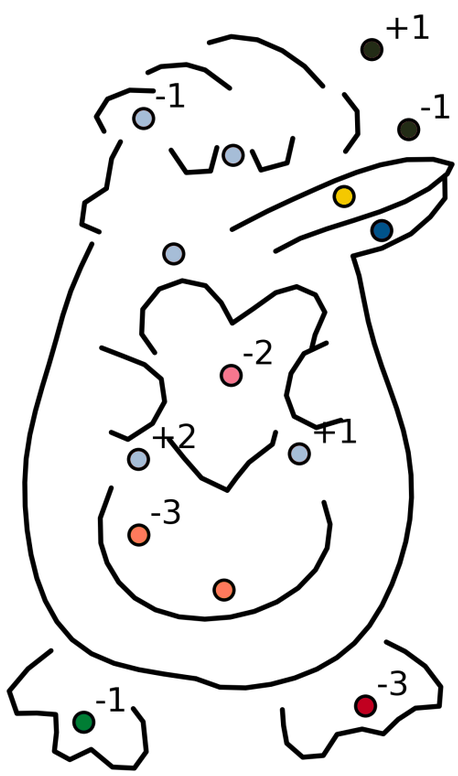
(b) Parakkat et al. (2022), results



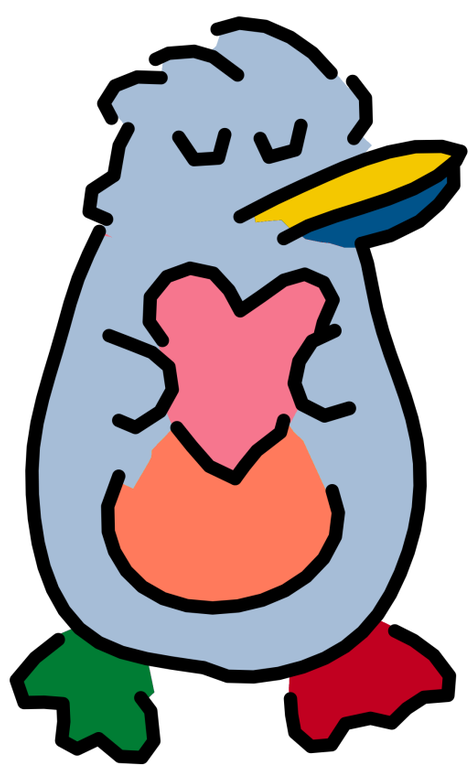
(c) Ours, hints (with strengths)



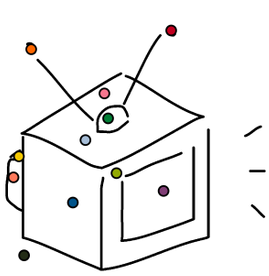
(d) Ours, results



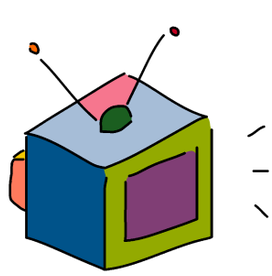
(a) Ours, hints (alt.)



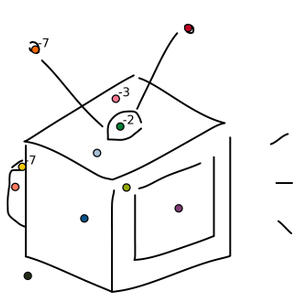
(b) Ours, results (alt.)



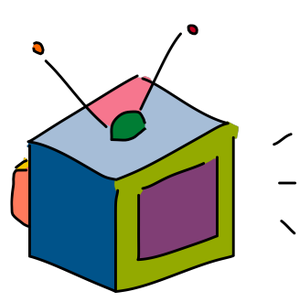
(a) Parakkat et al. (2022), hints



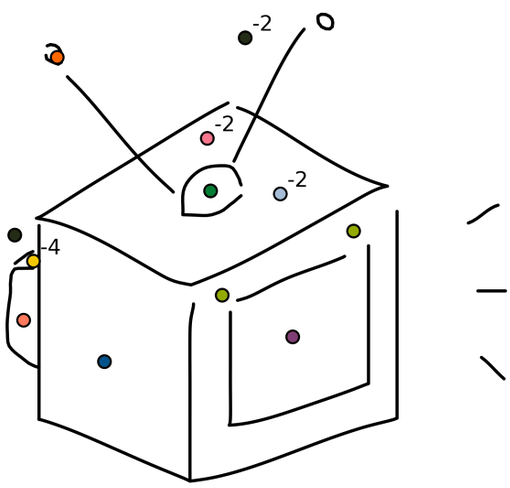
(b) Parakkat et al. (2022), results



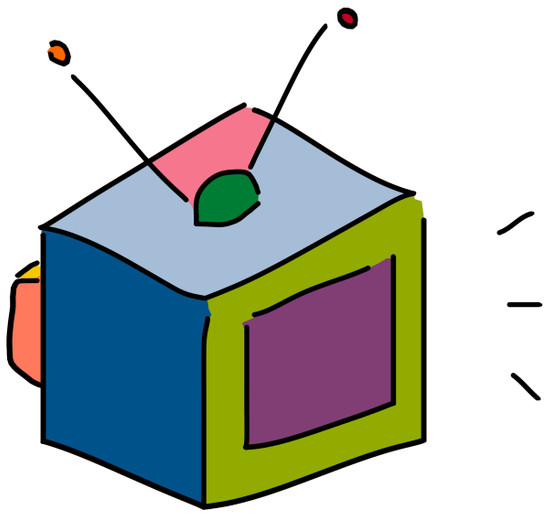
(c) Ours, hints (with strengths)



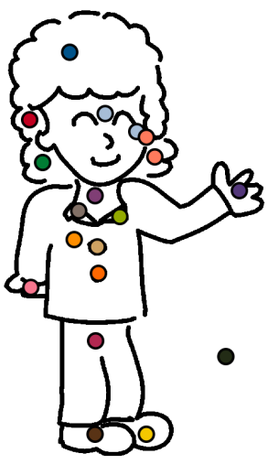
(d) Ours, results



(a) Ours, hints (alt.)

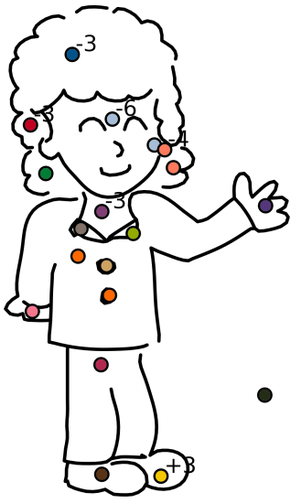


(b) Ours, results (alt.)



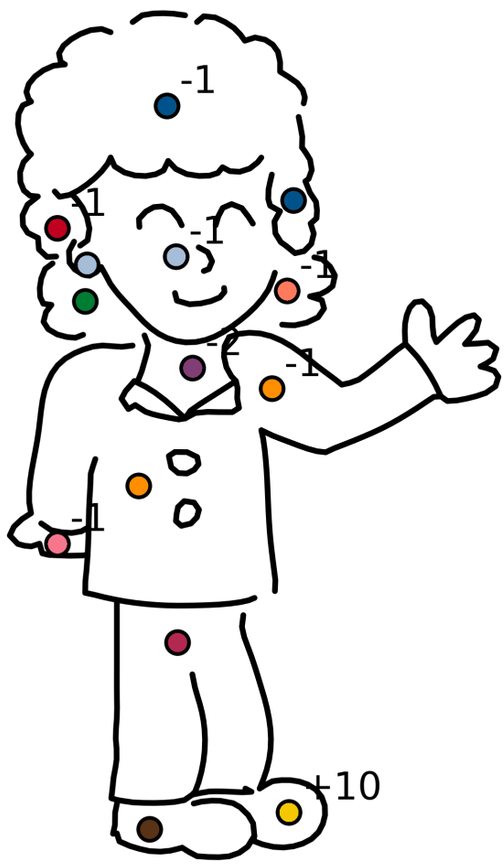
(a) Parakkat et al. (2022), hints

(b) Parakkat et al. (2022), results



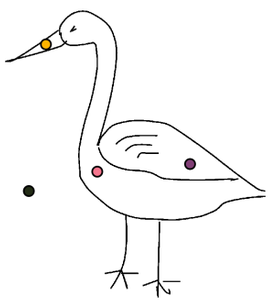
(c) Ours, hints (with strengths)

(d) Ours, results

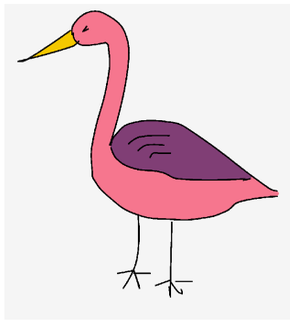


(a) Ours, hints (alt.)

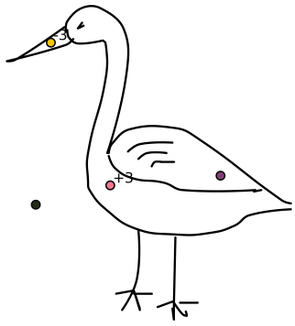
(b) Ours, results (alt.)



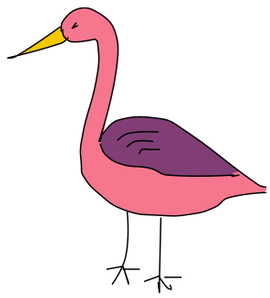
(a) Parakkat et al. (2022), hints



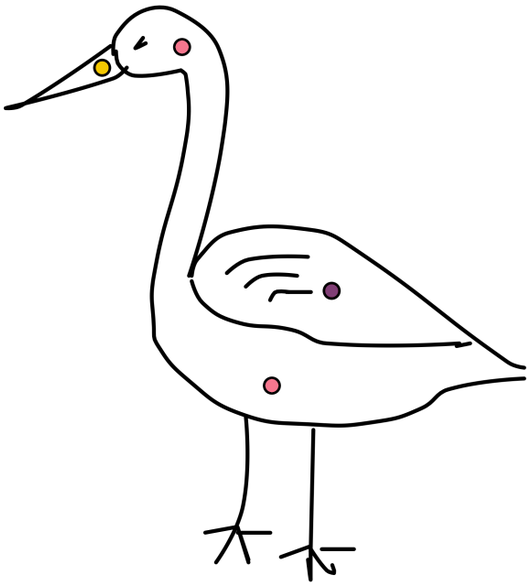
(b) Parakkat et al. (2022), results



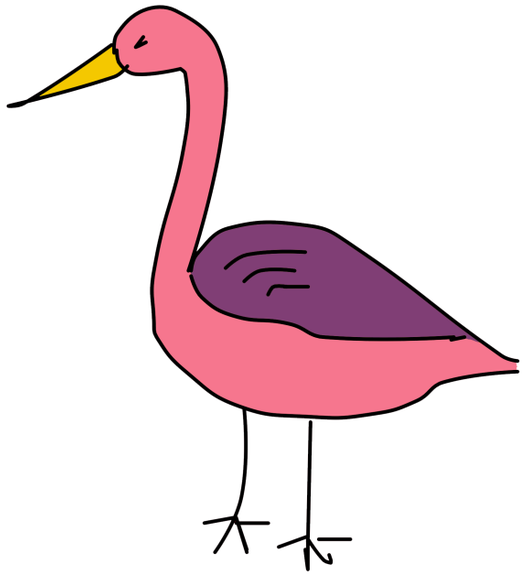
(c) Ours, hints (with strengths)



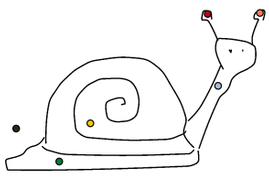
(d) Ours, results



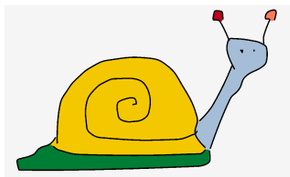
(a) Ours, hints (alt.)



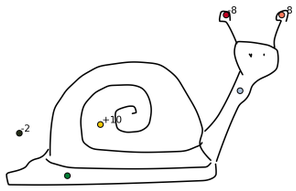
(b) Ours, results (alt.)



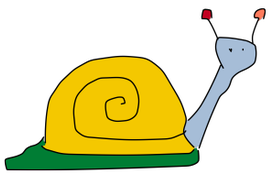
(a) Parakkat et al. (2022), hints



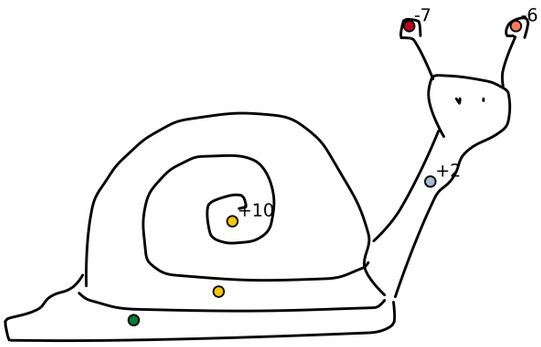
(b) Parakkat et al. (2022), results



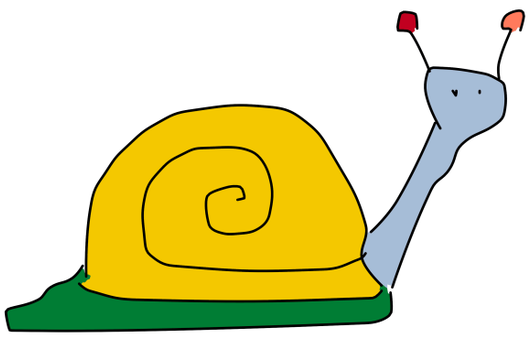
(c) Ours, hints (with strengths)



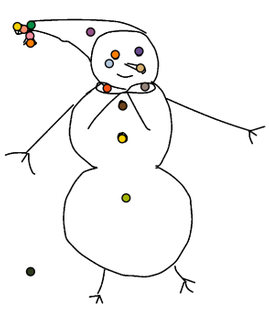
(d) Ours, results



(a) Ours, hints (alt.)



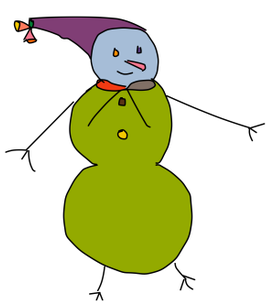
(b) Ours, results (alt.)



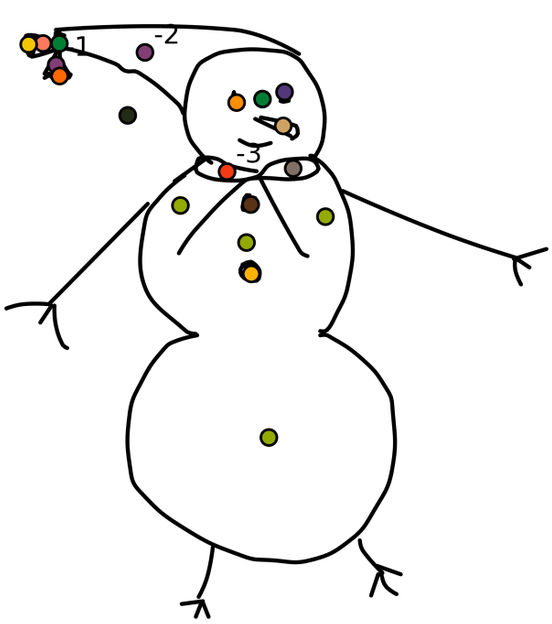
(a) Parakkat et al. (2022), hints

(b) Parakkat et al. (2022), results

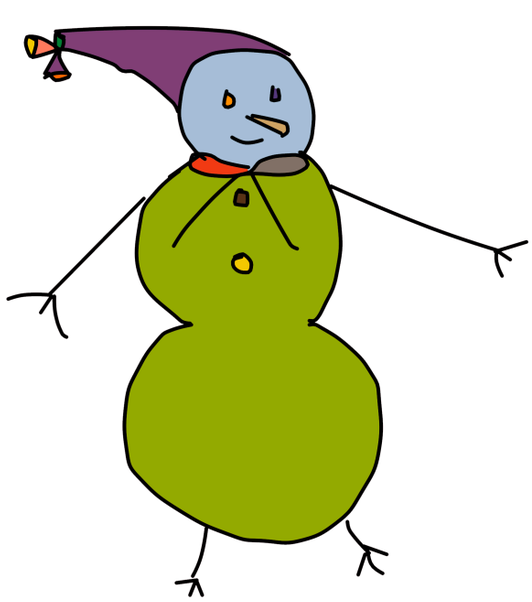
(c) Ours, hints (with strengths)



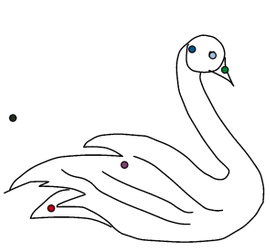
(d) Ours, results



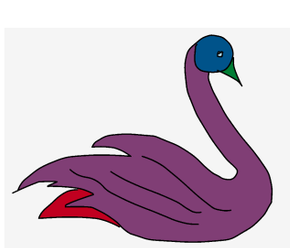
(a) Ours, hints (alt.)



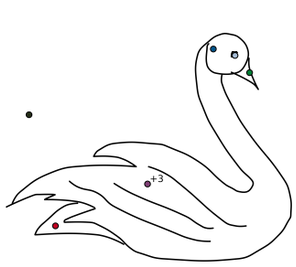
(b) Ours, results (alt.)



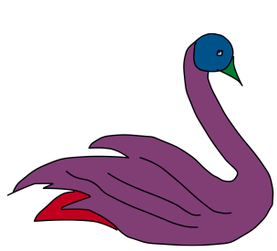
(a) Parakkat et al. (2022), hints



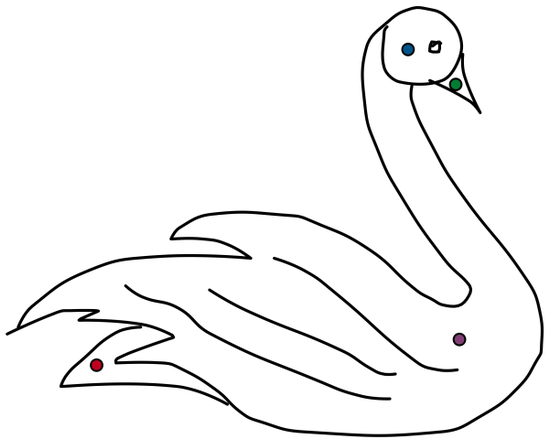
(b) Parakkat et al. (2022), results



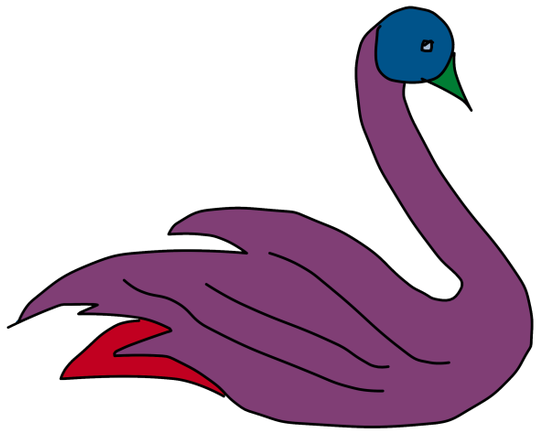
(c) Ours, hints (with strengths)



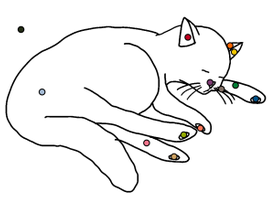
(d) Ours, results



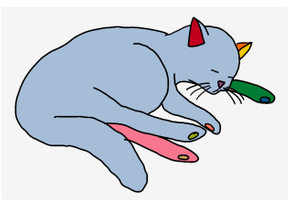
(a) Ours, hints (alt.)



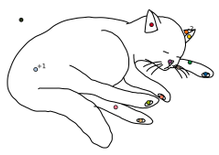
(b) Ours, results (alt.)



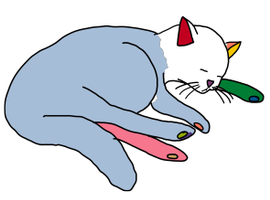
(a) Parakkat et al. (2022), hints



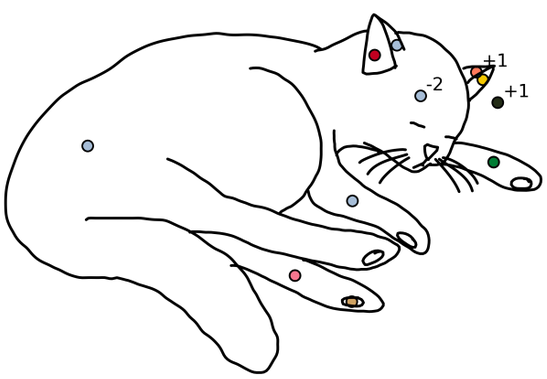
(b) Parakkat et al. (2022), results



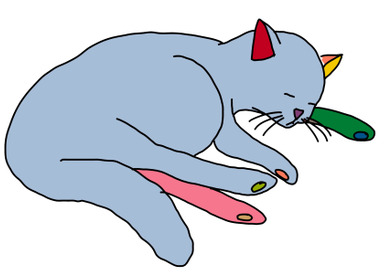
(c) Ours, hints (with strengths)



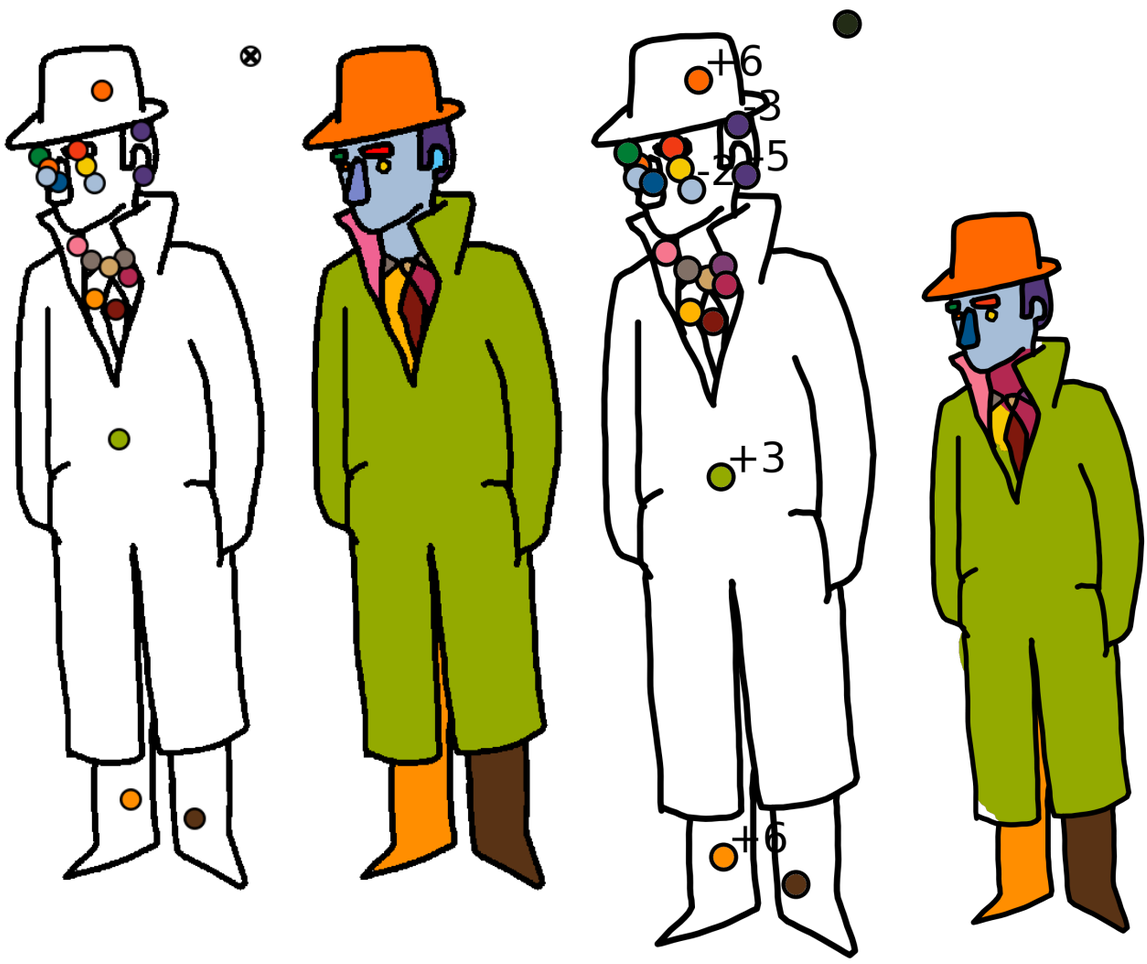
(d) Ours, results



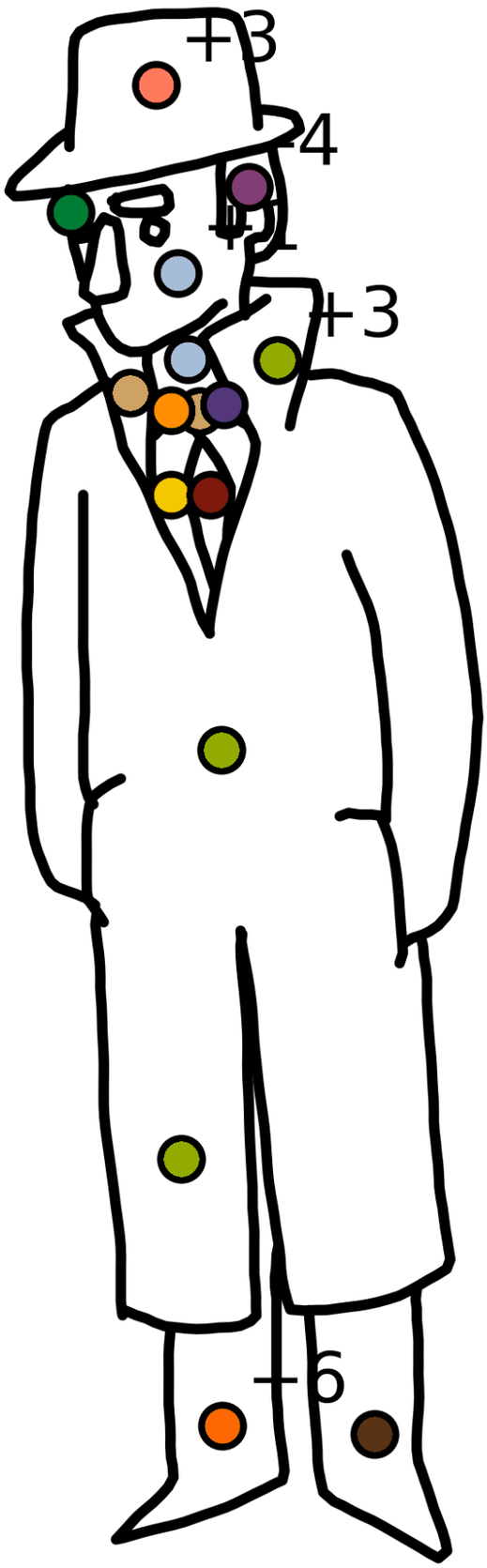
(a) Ours, hints (alt.)



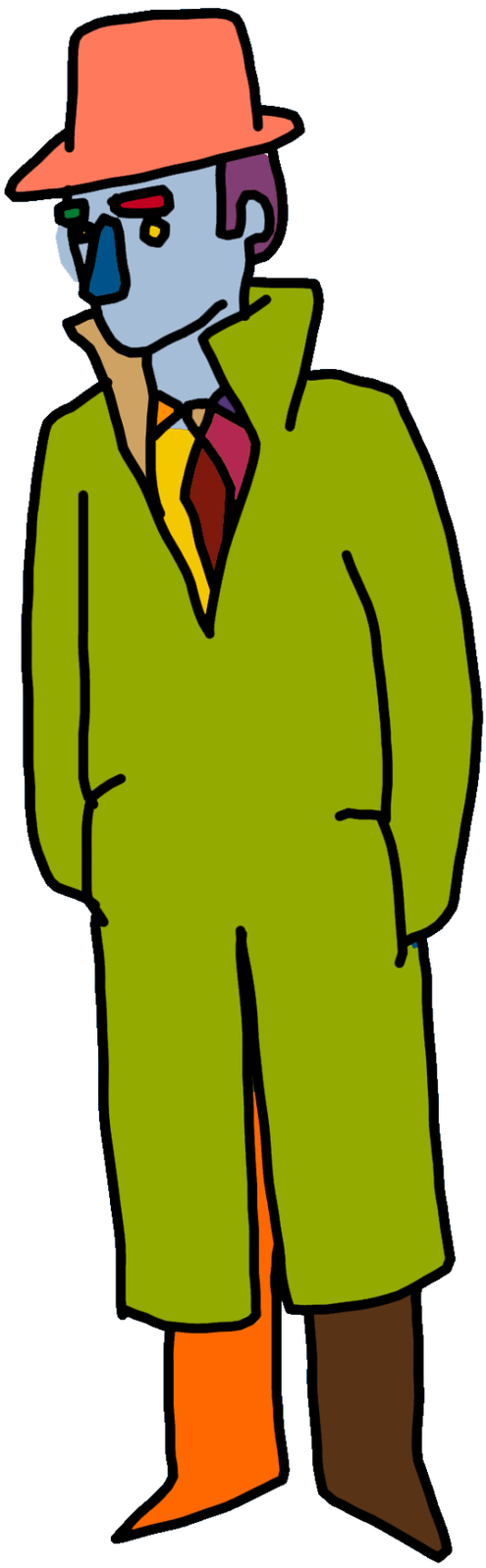
(b) Ours, results (alt.)



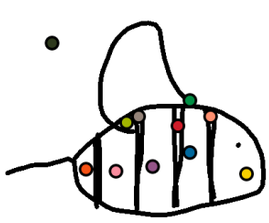
(a) Parakkat et al. (2022), hints (b) Parakkat et al. (2022), results (c) Ours, hints (with strengths) (d) Ours, results



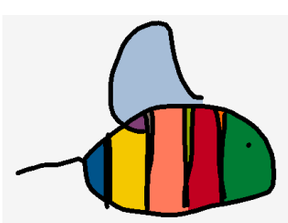
(a) Ours, hints (alt.)



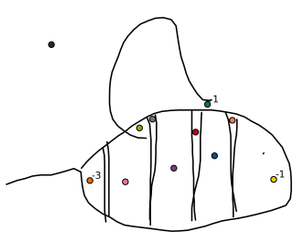
(b) Ours, results (alt.)



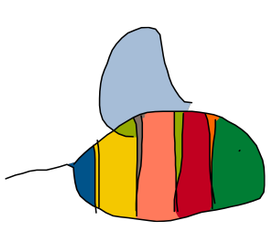
(a) Parakkat et al. (2022), hints



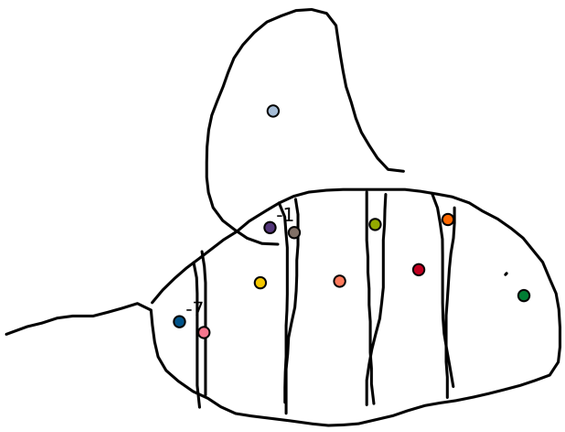
(b) Parakkat et al. (2022), results



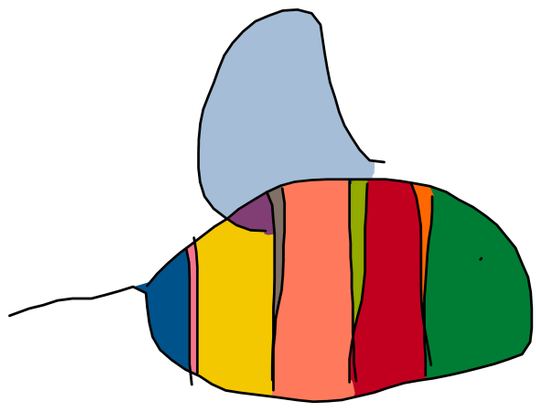
(c) Ours, hints (with strengths)



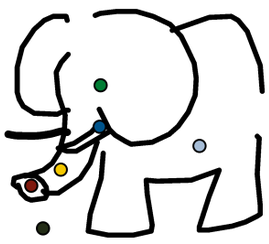
(d) Ours, results



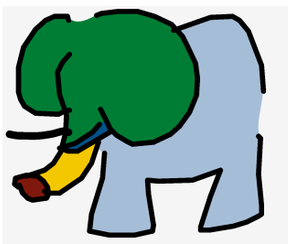
(a) Ours, hints (alt.)



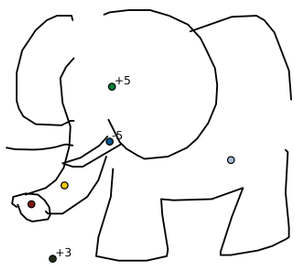
(b) Ours, results (alt.)



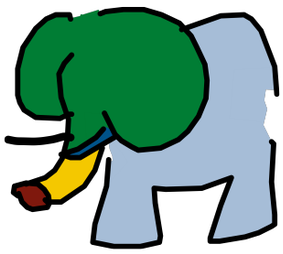
(a) Parakkat et al. (2022), hints



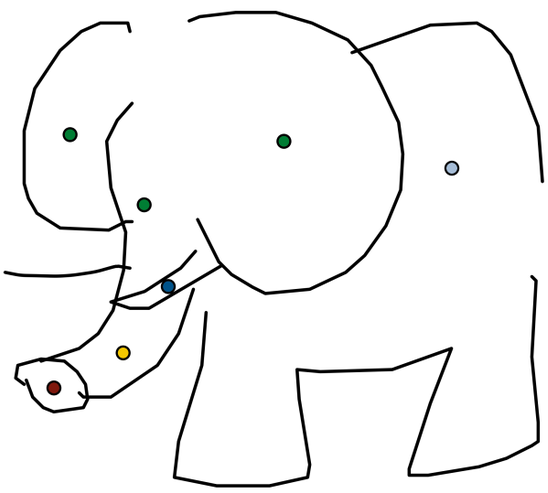
(b) Parakkat et al. (2022), results



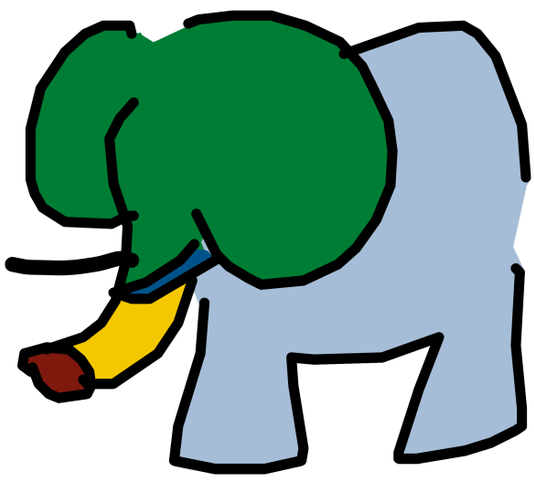
(c) Ours, hints (with strengths)



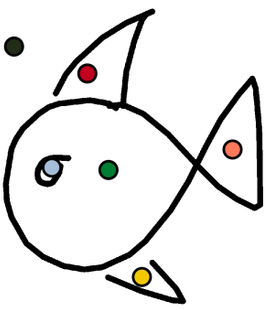
(d) Ours, results



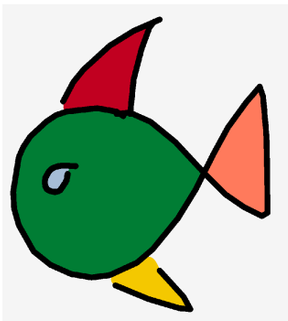
(a) Ours, hints (alt.)



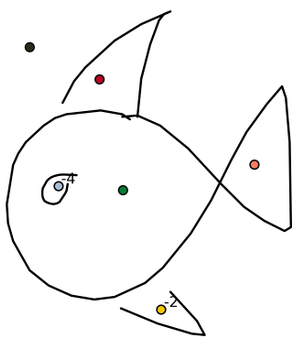
(b) Ours, results (alt.)



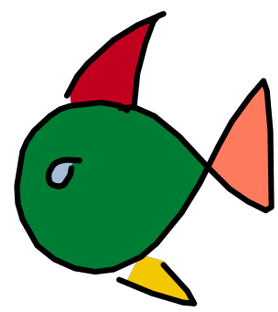
(a) Parakkat et al. (2022), hints



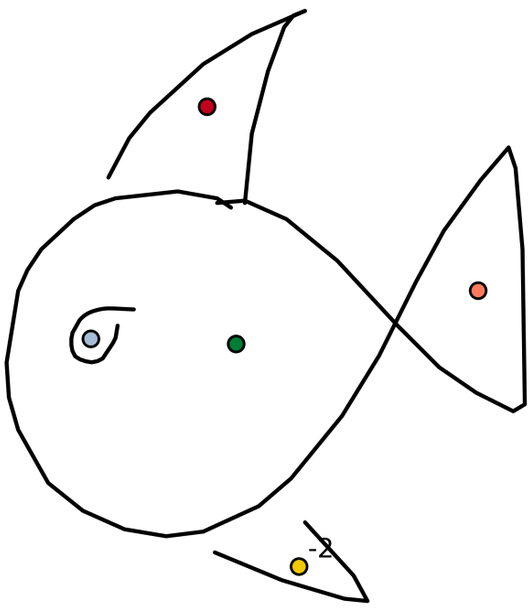
(b) Parakkat et al. (2022), results



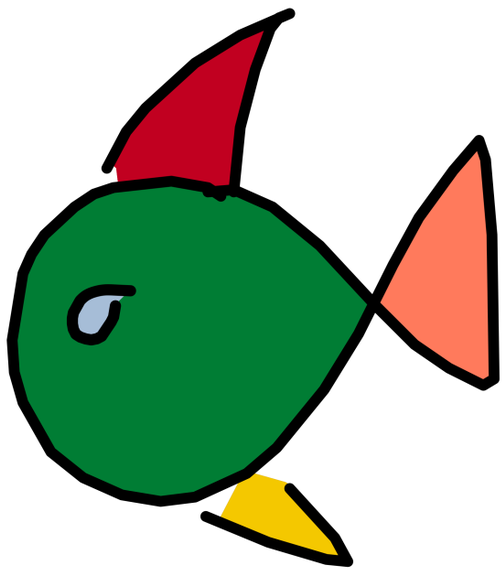
(c) Ours, hints (with strengths)



(d) Ours, results

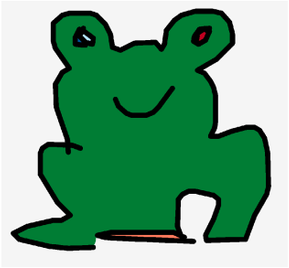


(a) Ours, hints (alt.)

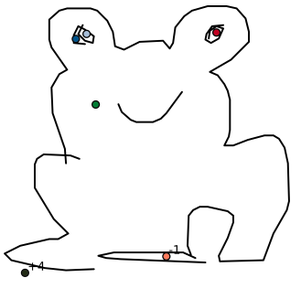


(b) Ours, results (alt.)

(a) Parakkat et al. (2022), hints

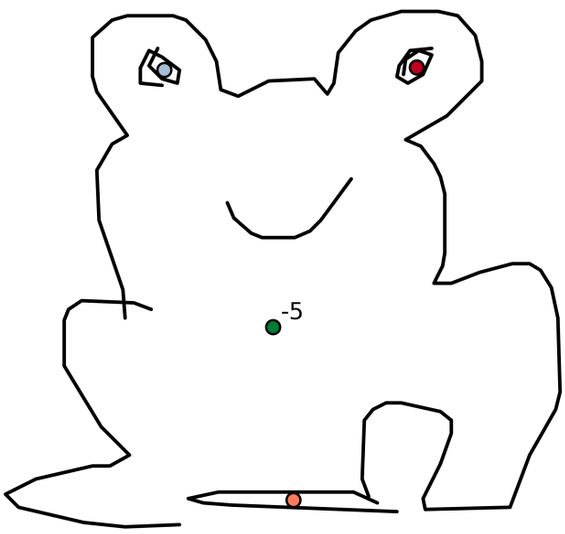


(b) Parakkat et al. (2022), results

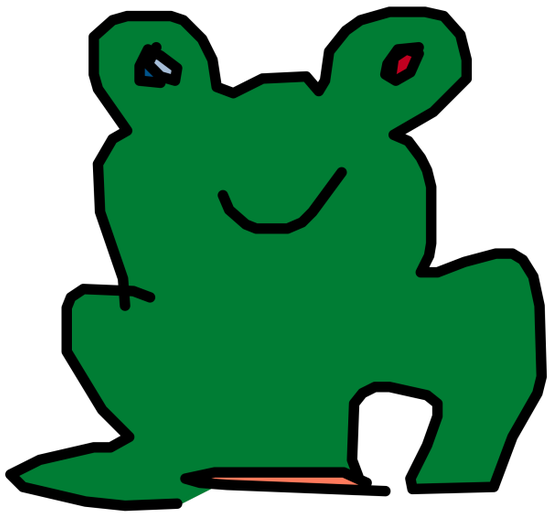


(c) Ours, hints (with strengths)

(d) Ours, results



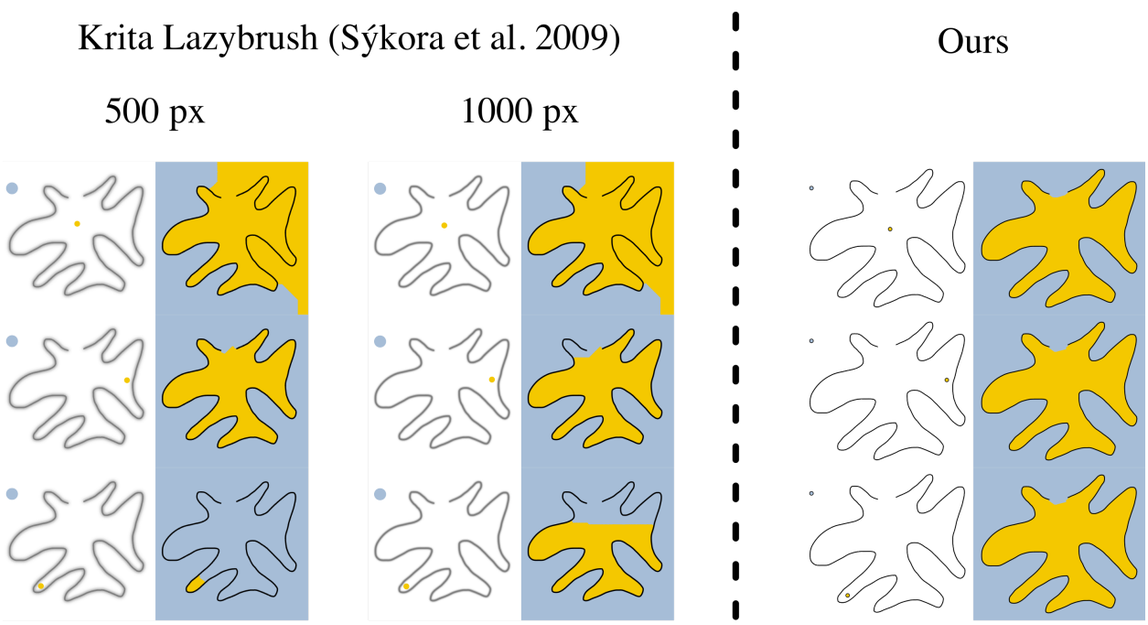
(a) Ours, hints (alt.)



(b) Ours, results (alt.)

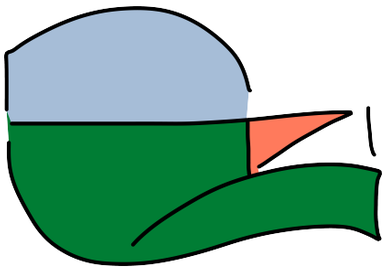
12.1.3 Comparison to [SDC09]

In §2.1 of the main text, we reference Fig. 8 of [SDC09] in which the authors demonstrate limitations of their method. Specifically, in (B), their algorithm does not effectively fill a single-stroke drawing that contains highly concave features. In the following figure, we replicate this input and run their method as well as ours, varying (1) the placement of user hints and (2) the resolution at which the input is rasterized for use with Krita. We observe that, on this highly concave input, their method demonstrates sensitivity to both properties, whereas ours does not.

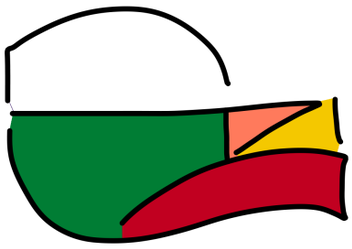


12.2 Automatic Methods

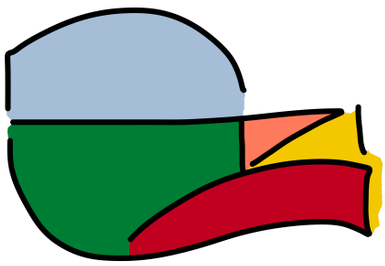
Below are comparison results for our automatic method against [YLL*22] and [PMC22] on the remaining survey sketches that are not included in the main paper (fish, fruit, penguin are in Fig. 6).



(a) Yin and Liu et al. (2022)



(b) Parakkat et al. (2022)

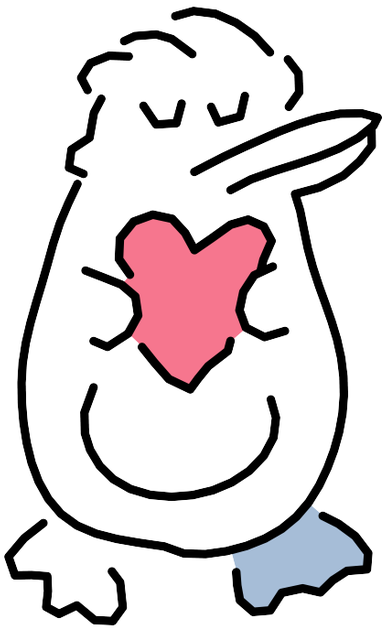


(c) Ours ($k = 6$)

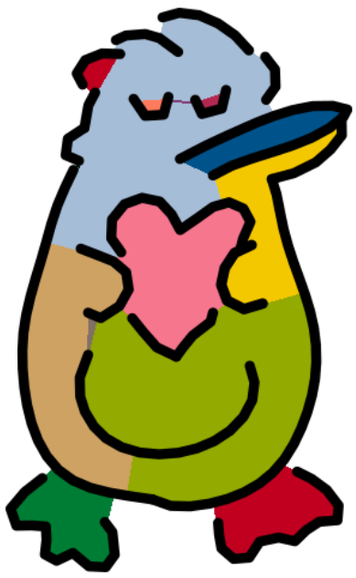
(a) Yin and Liu et al. (2022)

(b) Parakkat et al. (2022)

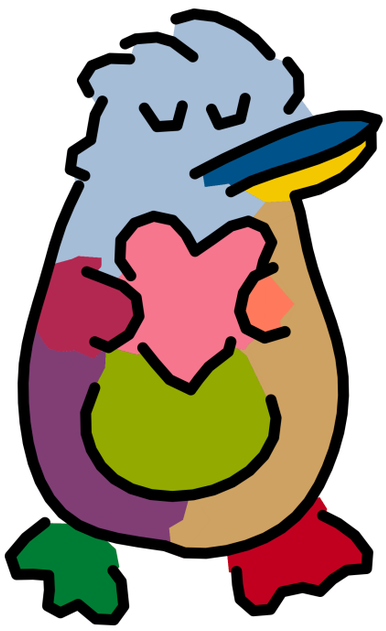
(c) Ours ($k = 8$)



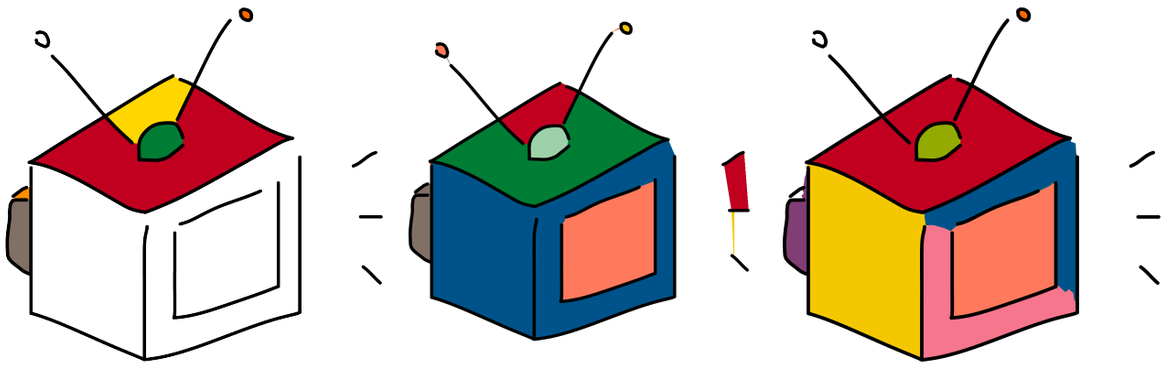
(a) Yin and Liu et al. (2022)



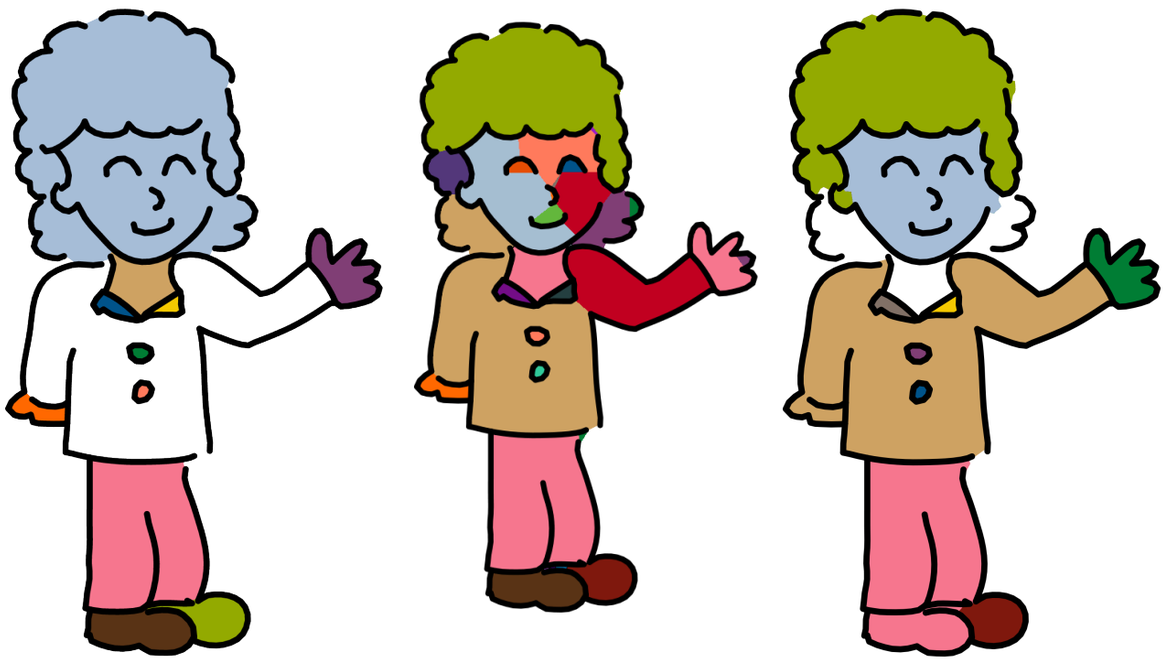
(b) Parakkat et al. (2022)



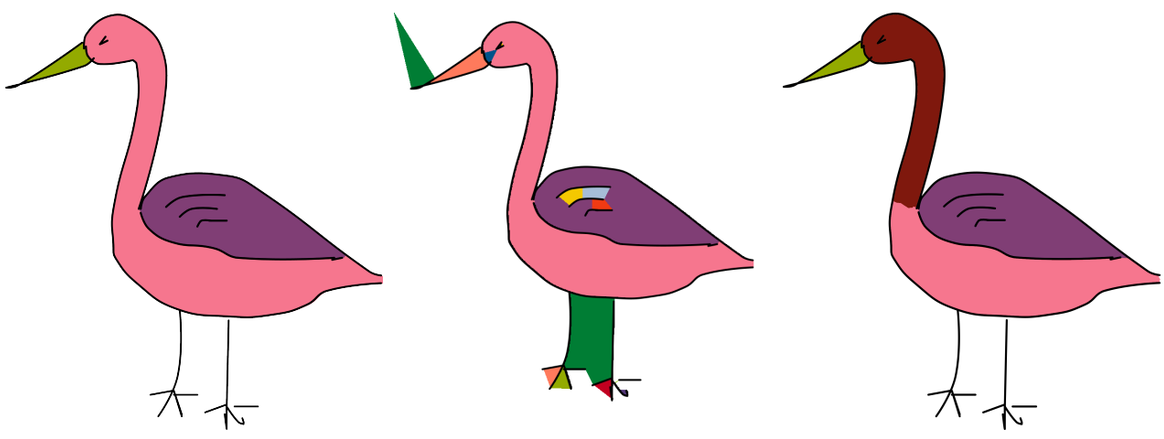
(c) Ours ($k = 12$)



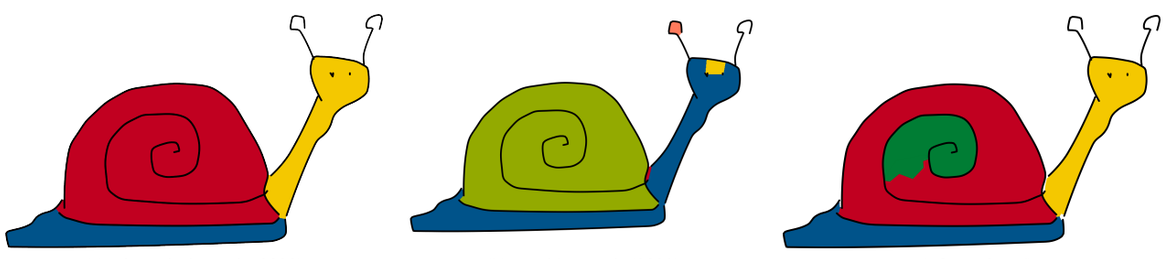
(a) Yin and Liu et al. (2022) (b) Parakkat et al. (2022) (c) Ours ($k = 8$)



(a) Yin and Liu et al. (2022) (b) Parakkat et al. (2022) (c) Ours ($k = 8$)



(a) Yin and Liu et al. (2022) (b) Parakkat et al. (2022) (c) Ours ($k = 5$)

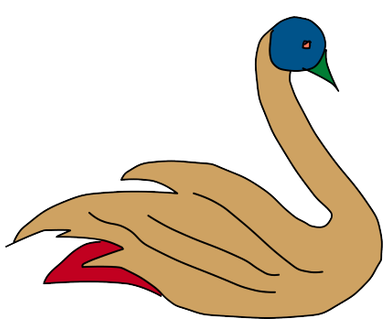


(a) Yin and Liu et al. (2022) (b) Parakkat et al. (2022) (c) Ours ($k = 5$)

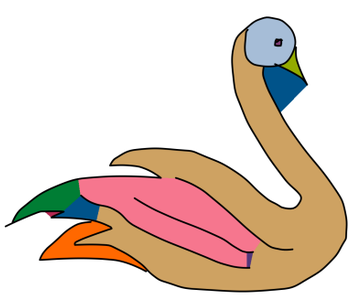
(a) Yin and Liu et al. (2022)

(b) Parakkat et al. (2022)

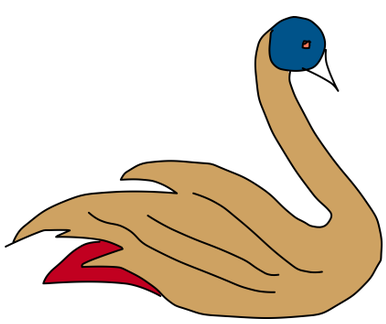
(c) Ours ($k = 6$)



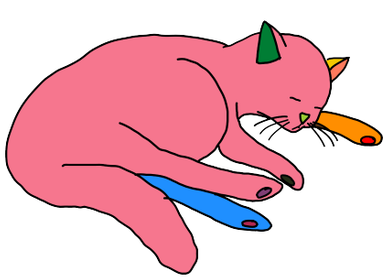
(a) Yin and Liu et al. (2022)



(b) Parakkat et al. (2022)

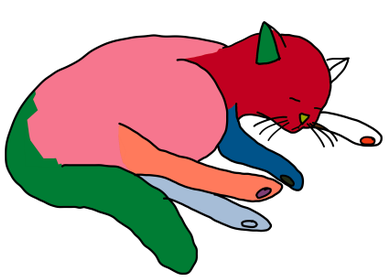


(c) Ours ($k = 4$)

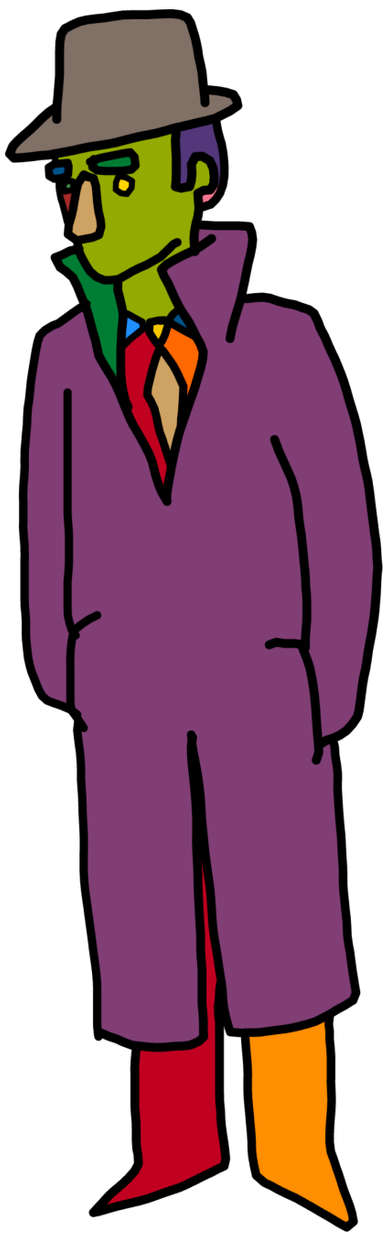


(a) Yin and Liu et al. (2022)

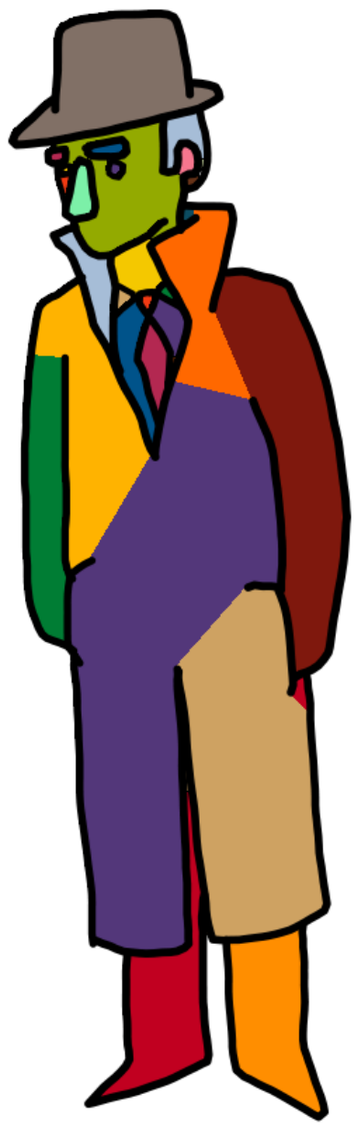
(b) Parakkat et al. (2022)



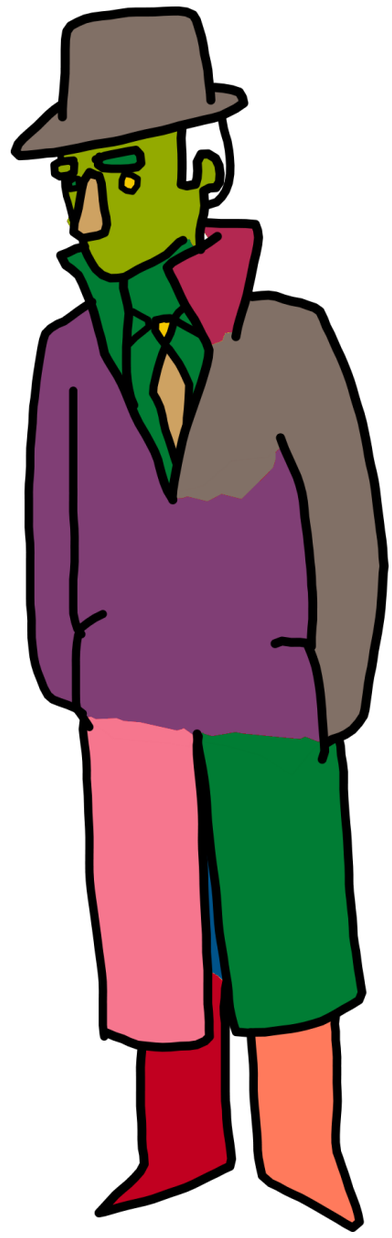
(c) Ours ($k = 9$)



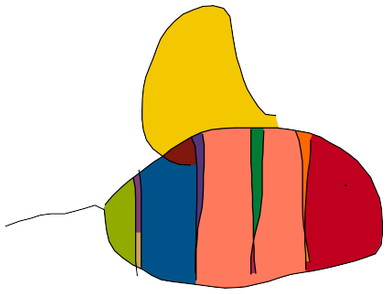
(a) Yin and Liu et al. (2022)



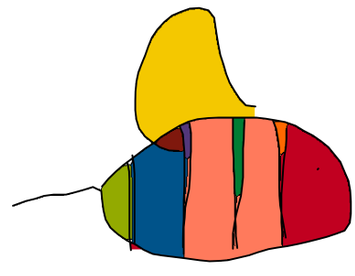
(b) Parakkat et al. (2022)



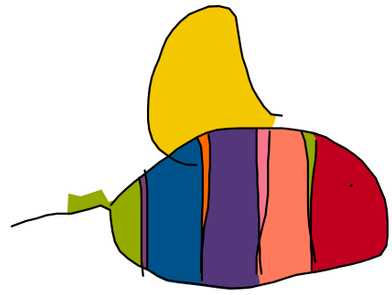
(c) Ours ($k = 15$)



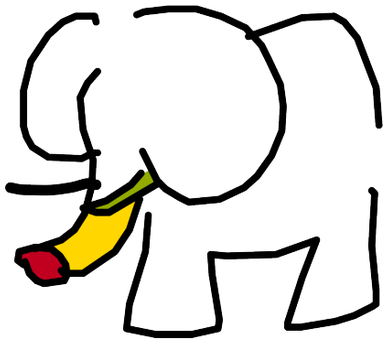
(a) Yin and Liu et al. (2022)



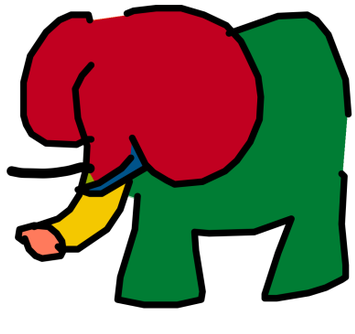
(b) Parakkat et al. (2022)



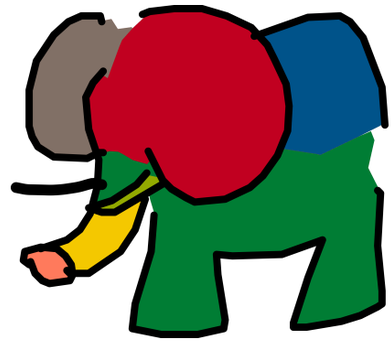
(c) Ours ($k = 11$)



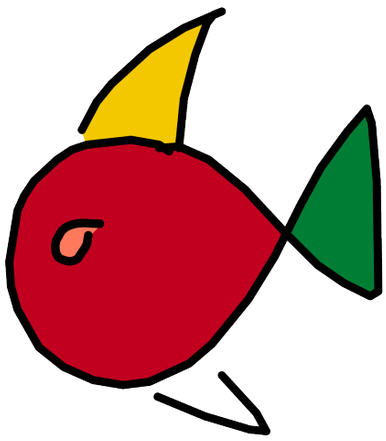
(a) Yin and Liu et al. (2022)



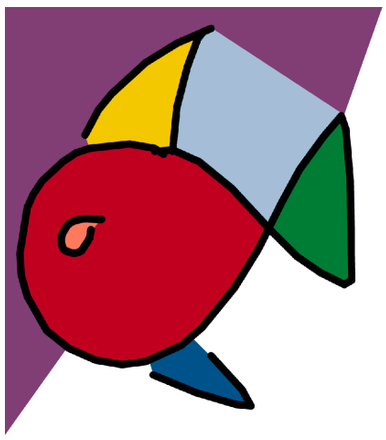
(b) Parakkat et al. (2022)



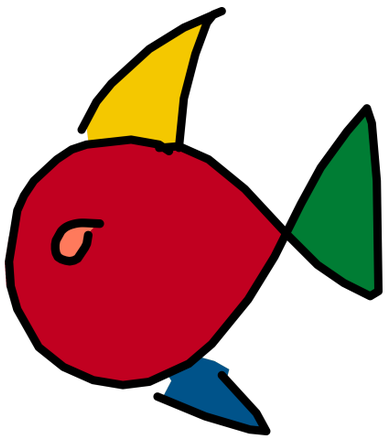
(c) Ours ($k = 8$)



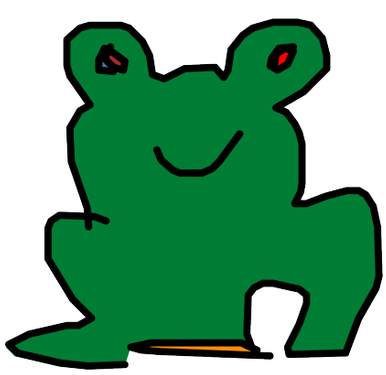
(a) Yin and Liu et al. (2022)



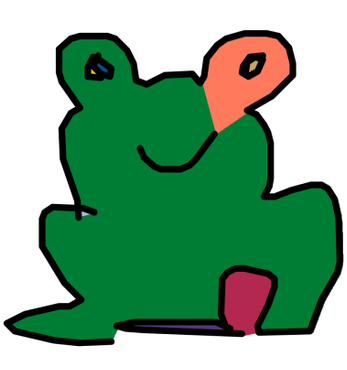
(b) Parakkat et al. (2022)



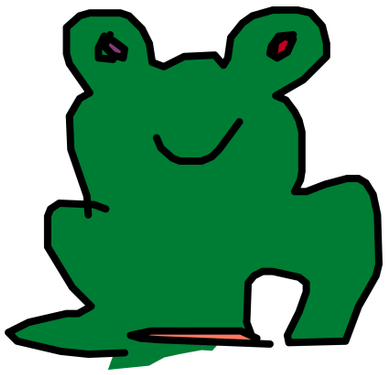
(c) Ours ($k = 6$)



(a) Yin and Liu et al. (2022)



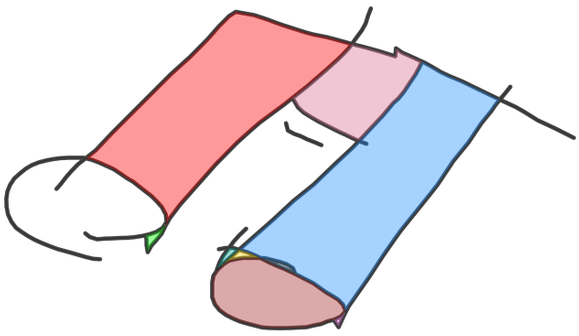
(b) Parakkat et al. (2022)



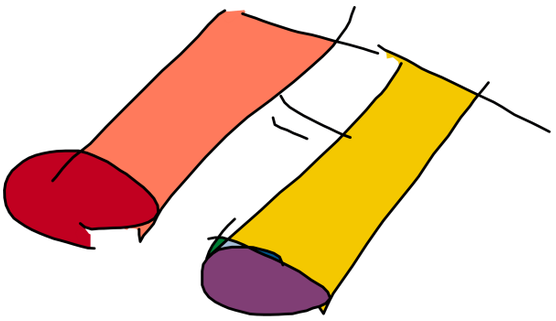
(c) Ours ($k = 5$)

12.2.1 Additional "gappy" inputs

The results of our survey (Fig. 7 of main text) suggest that our method is particularly strong on gappy inputs. To provide further evidence for this claim, we include 10 additional results from the Quick, Draw! [Goo17] dataset relative to the work of [YLL*22].

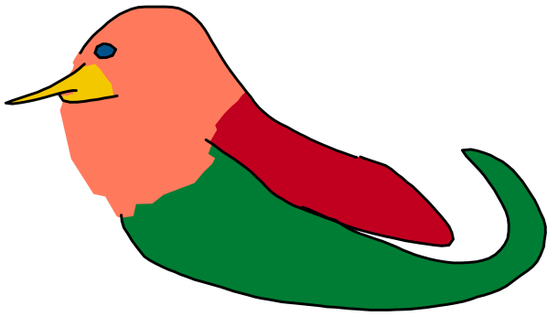


(a) Yin and Liu et al. (2022)

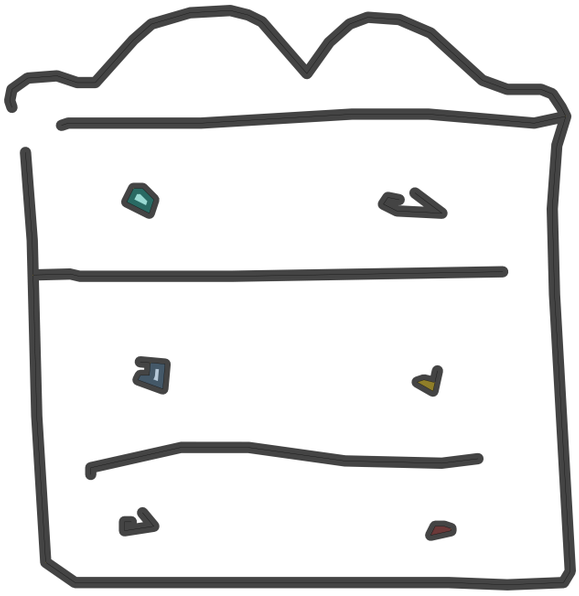


(b) Ours ($k = 5$)

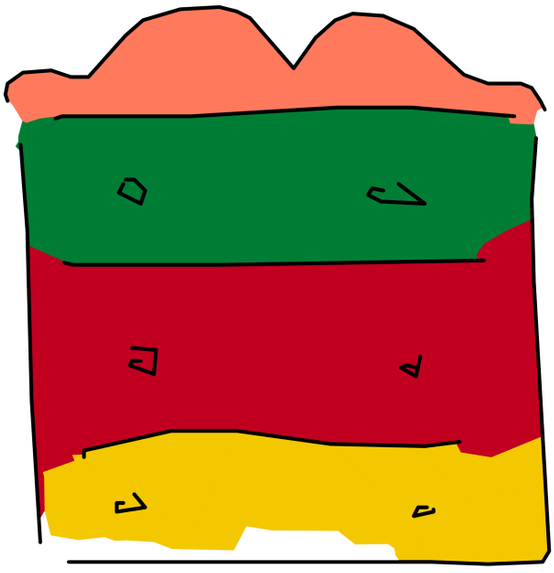
(a) Yin and Liu et al. (2022)



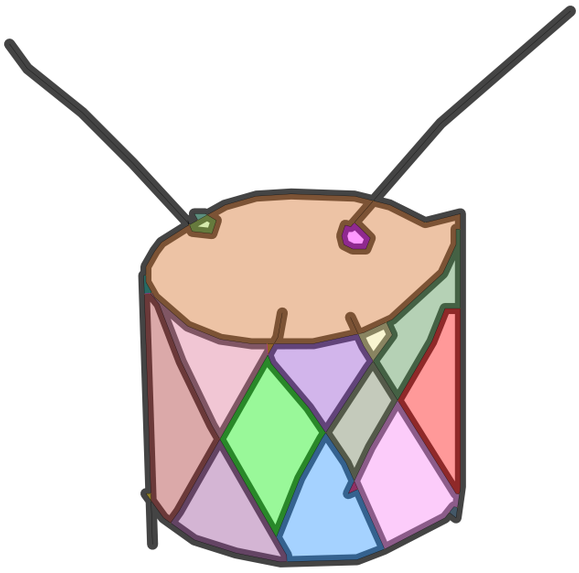
(b) Ours ($k = 6$)



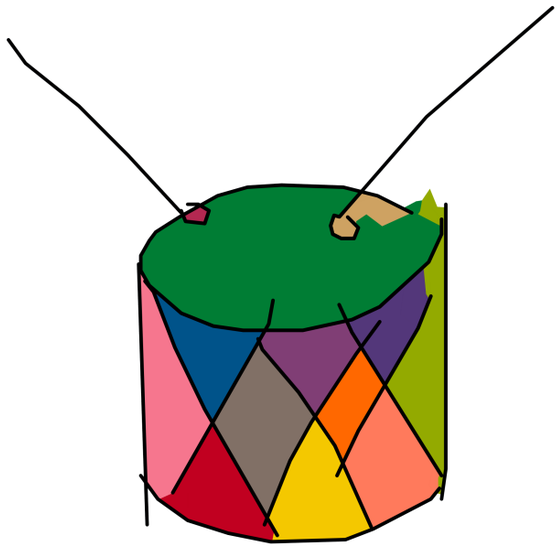
(a) Yin and Liu et al. (2022)



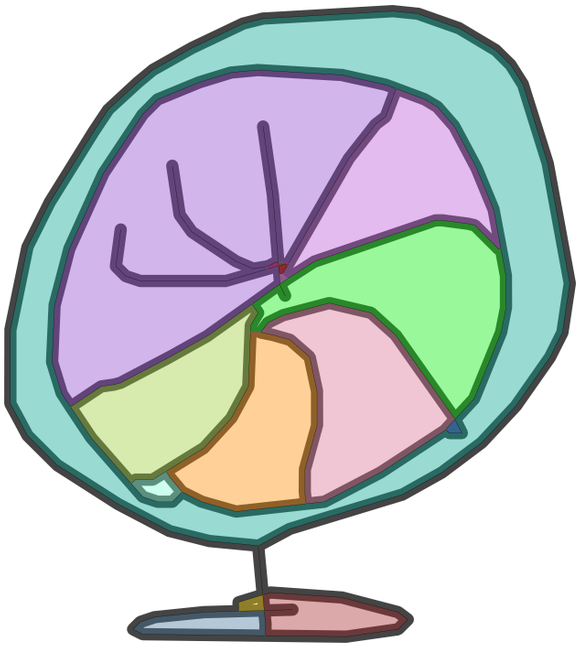
(b) Ours ($k = 5$)



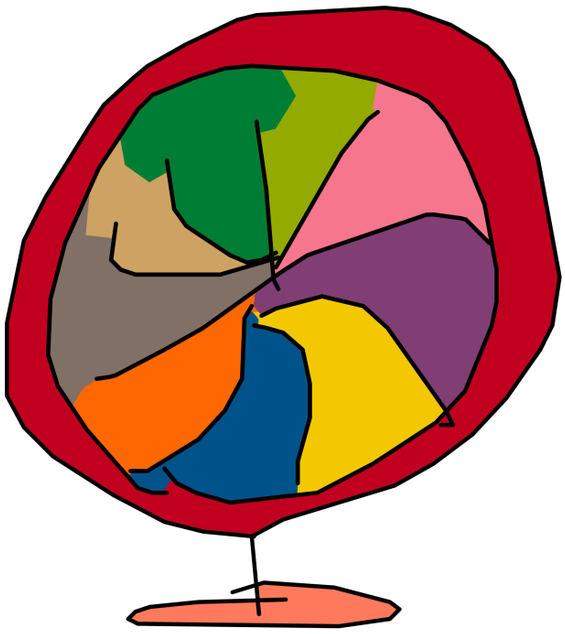
(a) Yin and Liu et al. (2022)



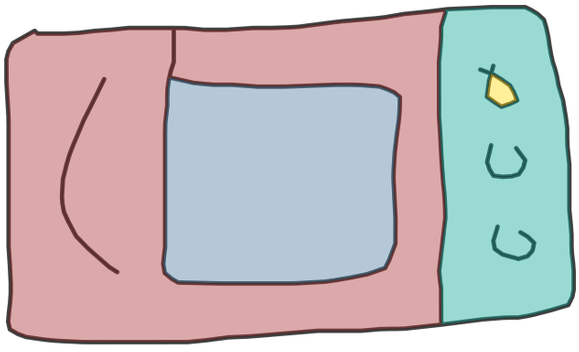
(b) Ours ($k = 13$)



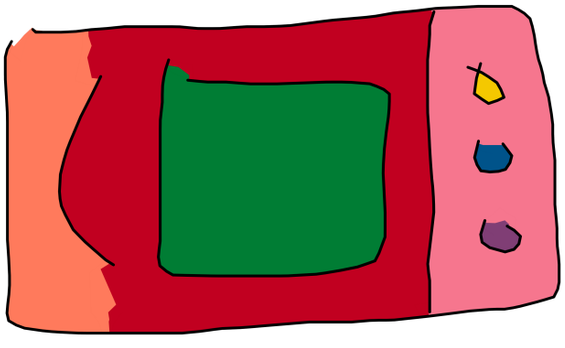
(a) Yin and Liu et al. (2022)



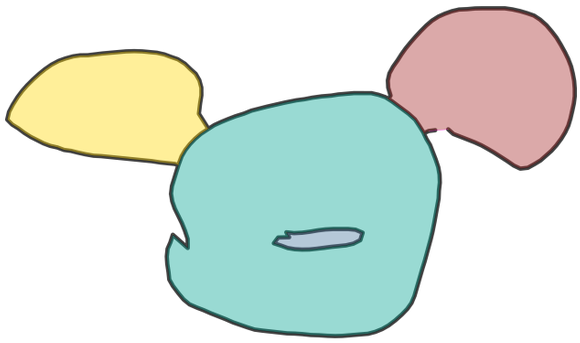
(b) Ours ($k = 14$)



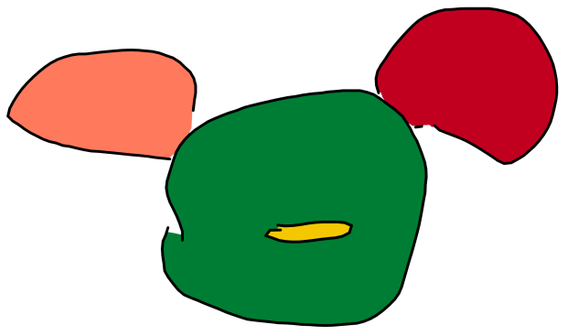
(a) Yin and Liu et al. (2022)



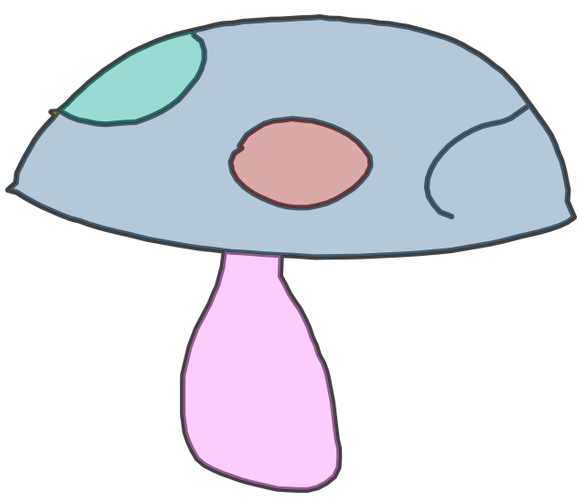
(b) Ours ($k = 7$)



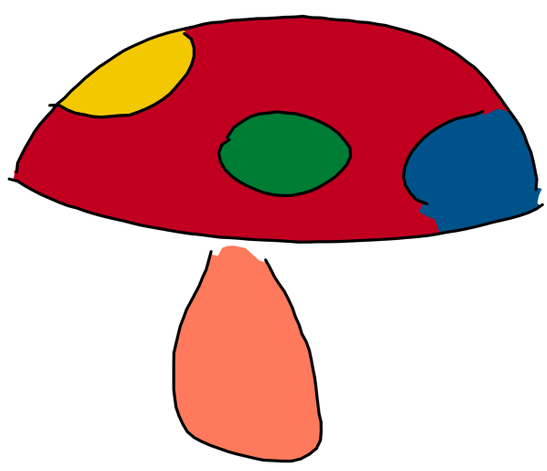
(a) Yin and Liu et al. (2022)



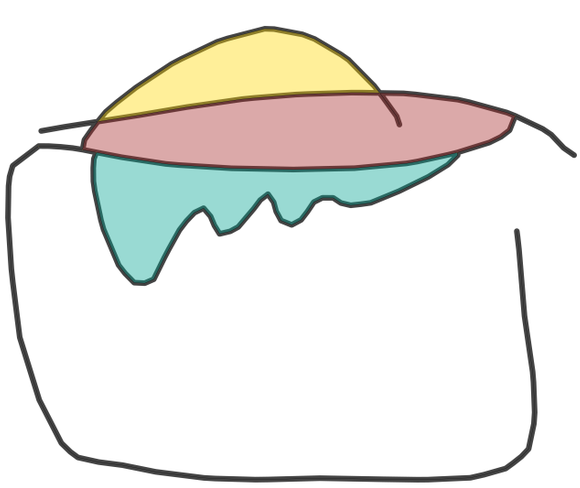
(b) Ours ($k = 5$)



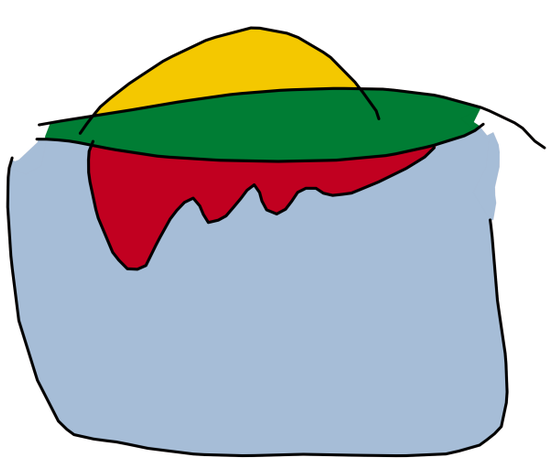
(a) Yin and Liu et al. (2022)



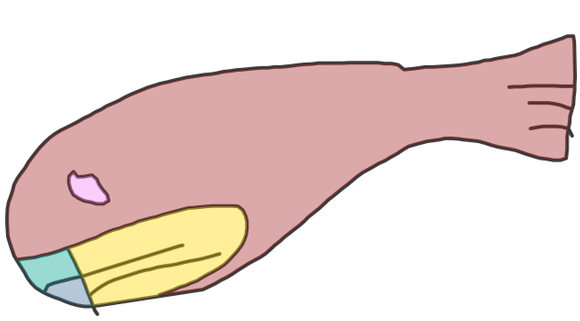
(b) Ours ($k = 6$)



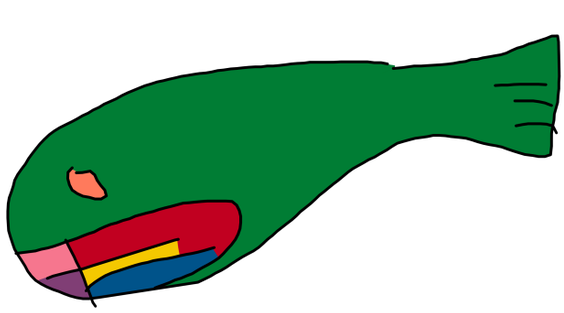
(a) Yin and Liu et al. (2022)



(b) Ours ($k = 4$)



(a) Yin and Liu et al. (2022)



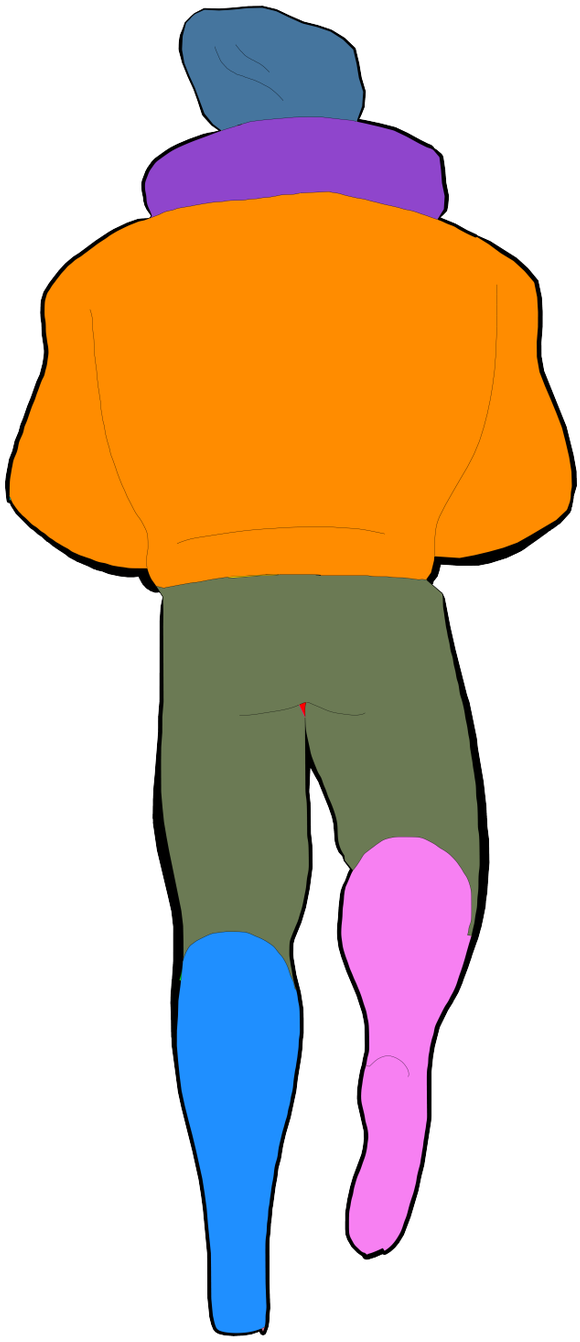
(b) Ours ($k = 8$)

12.2.2 Comparison on sketches in Yin and Liu et al. (2022)

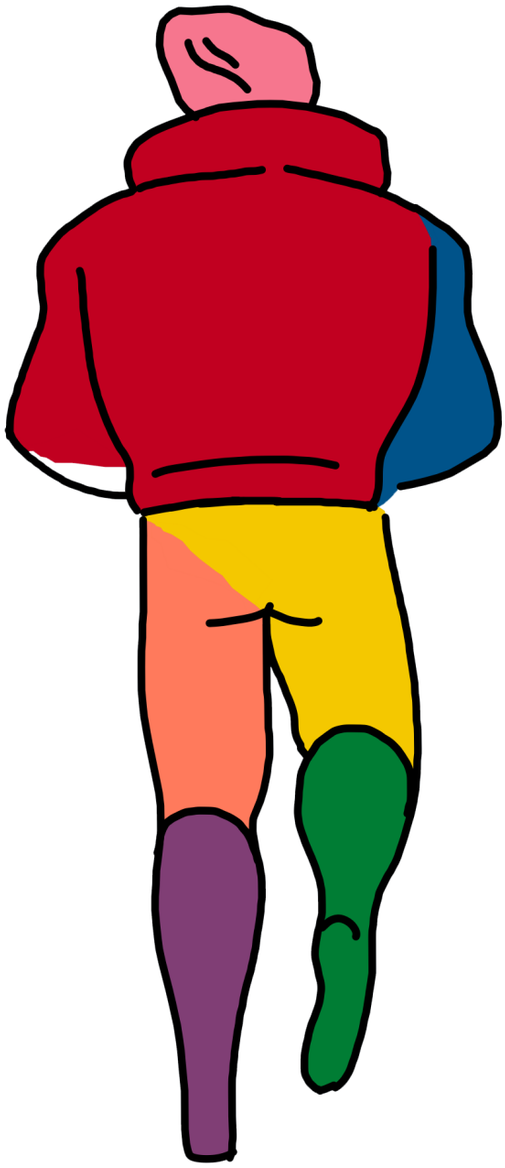
The automatic comparisons presented in our paper suggest that our method's advantages are most apparent on loose, gappy sketches. In order to compare faithfully against the genre of sketches which the work of [YLL*22] features more prominently, in which the size of gaps is smaller on average and less varied, we include the results of five additional sketches from their training/validation dataset.

(a) Yin and Liu et al. (2022)

(b) Ours ($k = 15$)



(a) Yin and Liu et al. (2022)



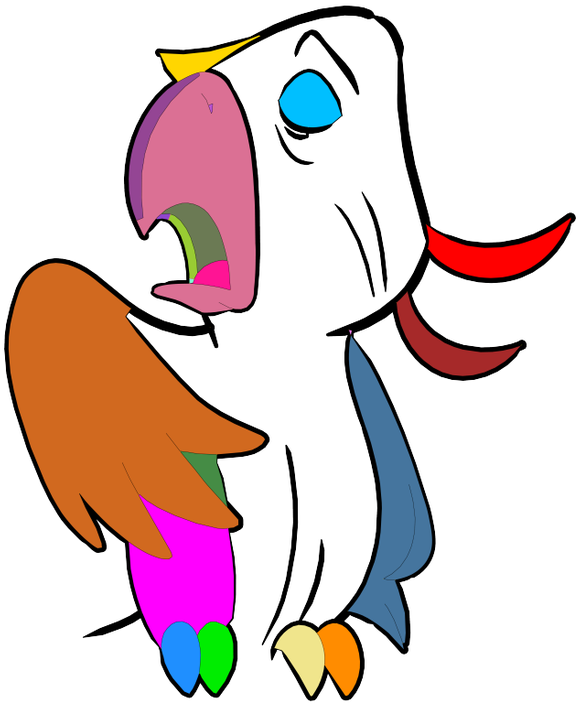
(b) Ours ($k = 8$)

(a) Yin and Liu et al. (2022)

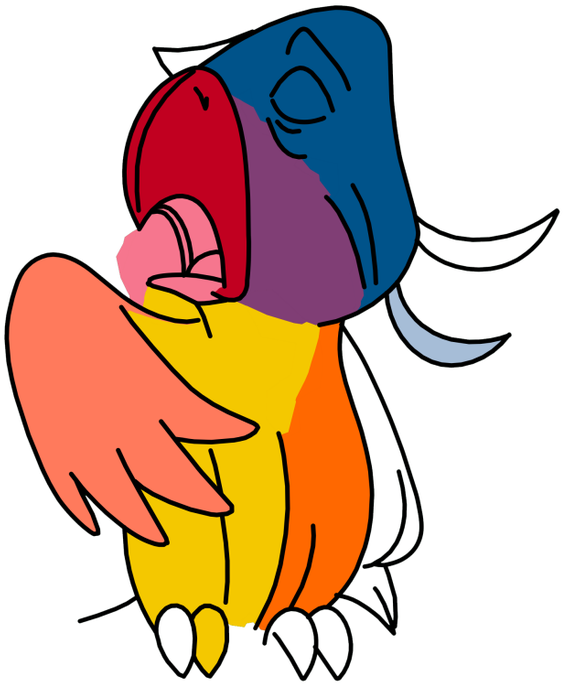
(b) Ours ($k = 15$)

(a) Yin and Liu et al. (2022)

(b) Ours ($k = 20$)



(a) Yin and Liu et al. (2022)



(b) Ours ($k = 8$)

References

[Can86] CANNY J.: A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 6 (1986), 679–698. doi:10.1109/TPAMI.1986.4767851.

[FTR18] FOUREY S., TSCHUMPERLÉ D., REVOY D.: A Fast and Efficient Semi-guided Algorithm for Flat Coloring Line-arts. In *Vision, Modeling and Visualization* (2018), Beck F., Dachsbacher C., Sadlo F., (Eds.), The Eurographics Association.

[Goo17] GOOGLE: Quick, draw! the data. https://quickdraw.withgoogle.com/data, 2017.

[Lux07] LUXBURG U.: A tutorial on spectral clustering. *Statistics and Computing 17*, 4 (dec 2007), 395–416.

[PMC22] PARAKKAT A. D., MEMARI P., CANI M.-P.: Delaunay painting: Perceptual image colouring from raster contours with gaps. *Computer Graphics Forum 41*, 6 (2022), 166–181.

[SDC09] SÝKORA D., DINGLIANA J., COLLINS S.: LazyBrush: Flexible painting tool for hand-drawn cartoons. *Computer Graphics Forum 28*, 2 (2009), 599–608.

[YLL*22] YIN J., LIU C., LIN R., VINING N., RHODIN H., SHEFFER A.: Detecting viewer-perceived intended vector sketch connectivity. *ACM Transactions on Graphics 41* (2022).

[YVG20] YAN C., VANDERHAEGHE D., GINGOLD Y.: A benchmark for rough sketch cleanup. *ACM Transactions on Graphics (TOG) 39*, 6 (Nov. 2020).