

Generative Whimsy Jigsaws - Supplementary Material

JEREMY CHEW, ETH Zürich, Switzerland
 KENJI TOJO, ETH Zürich, Switzerland
 NOBUYUKI UMETANI, University of Tokyo, Japan
 BERND BICKEL, ETH Zürich, Switzerland

ACM Reference Format:

Jeremy Chew, Kenji Tojo, Nobuyuki Umetani, and Bernd Bickel. 2026. Generative Whimsy Jigsaws - Supplementary Material. In *SIGGRAPH Asia 2026 Conference Papers (SA Conference Papers '26)*, December 01–04, 2026, Kuala Lumpur, Malaysia. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3829340.3842213>

1 Properties of the Generalized Voronoi Formulation

We prove that our generalized Voronoi formulation (parameterized by the slowness field f) satisfies constraints (i) and (iii). That is, cells (pieces) form a single connected region, and adjacent cells exactly share common edges with no gaps in between.

Setup. Let $\Omega \subseteq \mathbb{R}^2$ be the puzzle domain, and $f : \Omega \rightarrow [f_{\min}, f_{\max}]$ the slowness field with $0 < f_{\min} \leq f_{\max} < \infty$. For sites $\{\mathbf{p}_s\}_{s=1}^S$, the Eikonal solution $u^s(\mathbf{x})$ gives the geodesic distance $d_f(\mathbf{p}_s, \mathbf{x})$ from $\mathbf{p}_s$ to $\mathbf{x}$ under the metric induced by f . This is the shortest-path cost on a discretized grid. The generalized Voronoi cells are given by $V_i = \{\mathbf{x} \in \Omega : u^i(\mathbf{x}) \leq u^j(\mathbf{x}) \forall j\}$. Since $\{u^s(\mathbf{x})\}_{s=1}^S$ is a finite set and attains its minimum at every $\mathbf{x}$, we have $\bigcup_i V_i = \Omega$.

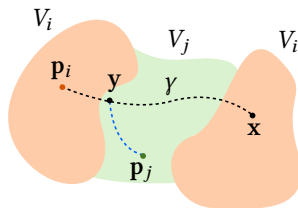
Exact Boundaries. We first show that adjacent cells must share common edges. Suppose we have $\mathbf{x} \in V_i \cap V_j$ for $i \neq j$. Then, by definition, we have $u^i(\mathbf{x}) \leq u^j(\mathbf{x})$ and $u^j(\mathbf{x}) \leq u^i(\mathbf{x})$, implying $u^i(\mathbf{x}) = u^j(\mathbf{x})$, and both $u^i(\mathbf{x})$ and $u^j(\mathbf{x})$ are $\leq u^k(\mathbf{x}) \forall k$. Conversely, any $\mathbf{x}$ satisfying $u^i(\mathbf{x}) = u^j(\mathbf{x}) \leq u^k(\mathbf{x}) \forall k$ is in V_i and V_j .

Therefore, we have

$$V_i \cap V_j = \{\mathbf{x} \in \Omega : u^i(\mathbf{x}) = u^j(\mathbf{x}) \leq u^k(\mathbf{x}) \forall k\}, \quad (1)$$

i.e. the bisector under d_f . Combined with the fact that $\bigcup_i V_i = \Omega$, we have that adjacent pieces share exact common edges with no gaps. $\square$

Per-cell Connectedness. We show that each V_i is connected by contradiction. Suppose V_i splits into multiple components, with $\mathbf{p}_i$ in one of them. Let $\mathbf{x}$ lie in a **different** component of V_i . Let γ be the shortest path from $\mathbf{p}_i$ to $\mathbf{x}$,



Authors' Contact Information: Jeremy Chew, ETH Zürich, Zürich, Switzerland, jechew@ethz.ch; Kenji Tojo, ETH Zürich, Zürich, Switzerland, tojo@arch.ethz.ch; Nobuyuki Umetani, University of Tokyo, Tokyo, Japan, n.umetani@gmail.com; Bernd Bickel, ETH Zürich, Zürich, Switzerland, bickelb@ethz.ch.



This work is licensed under a Creative Commons Attribution 4.0 International License. *SA Conference Papers '26, Kuala Lumpur, Malaysia*
 © 2026 Copyright held by the owner/author(s).
 ACM ISBN 979-8-4007-2842-6/2026/12
<https://doi.org/10.1145/3829340.3842213>

so $\text{length}(\gamma) = u^i(\mathbf{x})$. Since $\mathbf{x}$

is not in the same component as $\mathbf{p}_i$, γ must exit V_i at some point, passing through $\mathbf{y}$, where a different site j satisfies $u^j(\mathbf{y}) < u^i(\mathbf{y})$.

Split γ at $\mathbf{y}$. The prefix from $\mathbf{p}_i$ to $\mathbf{y}$ is itself a shortest path, with length $u^i(\mathbf{y})$, and the suffix from $\mathbf{y}$ to $\mathbf{x}$ has length $u^i(\mathbf{x}) - u^i(\mathbf{y})$. We construct a path from $\mathbf{p}_j$ (the source point of the cell that γ passes through) to $\mathbf{x}$ as $\mathbf{p}_j \rightarrow \mathbf{y} \rightarrow \mathbf{x}$. This has length $u^j(\mathbf{y}) + u^i(\mathbf{x}) - u^i(\mathbf{y}) < u^i(\mathbf{y}) + u^i(\mathbf{x}) - u^i(\mathbf{y}) = u^i(\mathbf{x})$ since $u^j(\mathbf{y}) < u^i(\mathbf{y})$. Hence, we have $u^j(\mathbf{x}) < u^i(\mathbf{x})$, contradicting $\mathbf{x} \in V_i$, and therefore, V_i must be connected. $\square$

Discrete Setting. In the discrete setting, only constraint (iii) (jigsaw-fit) is guaranteed because the argmin assigns exactly one index per pixel. For the other two constraints, constraint (i) (piece-wise area-connectivity) can technically be violated due to how the argmin breaks ties using the lowest index. Constraint (ii) (global area-connectivity) is also not guaranteed, but the background sources provide a strong heuristic. In practice, however, we observe **zero** constraint violations in all (35+) experiments.

2 Recognizability

We attempt to quantify recognizability by showing each silhouette to a Vision-Language Model (VLM) (Claude Sonnet 4.6) and asking it to accept or reject candidate labels. The incorrect candidate labels were generated by picking from the target category's confusion pool (for example, a silhouette labeled "mouse" could reasonably be confused as "guinea pig" or "hamster"), filtered by a WordNet rule that collapses synonyms to a single answer. The prompt given to the VLM was: "This image is a single solid black silhouette on a plain white background. Does the shape look like a **{noun}**? Answer with exactly one word: yes or no."

Across our examples, the model correctly accepted the correct label 313/358 times, and correctly rejected the wrong label 1176/1278 times (around 90% balanced accuracy). Figure 1 shows some examples of where the VLM failed to say **yes** to the correct label, as well as where it failed to say **no** to the wrong labels.

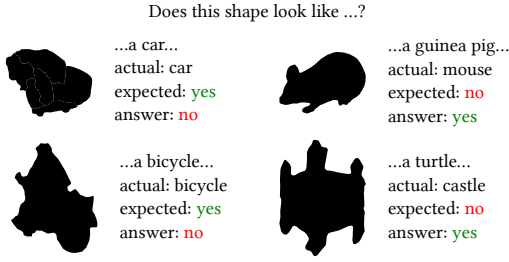


Fig. 1. **Wrong Labels.** We show some examples where the VLM was unable to correctly recognize what the silhouette was trying to depict. *Left:* The VLM does not accept the correct labels. *Right:* The VLM accepts the incorrect labels

3 Eikonal Forward Pass

ALGORITHM 1: Eikonal solve forward pass

Input: Source locations p_i , slowness field f
Output: Travel times u^i
Initialize $u^i \leftarrow \infty$ everywhere;
// Compute step size, assume grid spans $[-1,1]^2$
 $h \leftarrow 2/(N-1)$;
for each source point p_i **do**
 for each neighboring cell (m, n) **do**
 Set $u_{m,n}^i \leftarrow \text{dist to } p_i \times f_{m,n}$;
 end
end
for $it = 1$ to $4N$ **do**
 for each grid cell (m, n) **do**
 Skip if (m, n) is a source corner;
 // Get smallest neighbors
 $a \leftarrow \min(u_{m,n-1}^i, u_{m,n+1}^i)$;
 $b \leftarrow \min(u_{m-1,n}^i, u_{m+1,n}^i)$;
 if $|a - b| \geq hf_{m,n}$ **then**
 $u_{m,n}^i \leftarrow \min(a, b) + hf_{m,n}$;
 else
 $u_{m,n}^i \leftarrow \frac{1}{2}(a + b + \sqrt{2(hf_{m,n})^2 - (a - b)^2})$;
 end
 end
end

4 Eikonal Backward Pass

ALGORITHM 2: Eikonal solve backward pass

Input: Source locations p_i , slowness field f , solution from forward pass u^i , initial adjoint $du^i = \frac{dL}{du^i}$
Output: Gradient with respect to the slowness field f' and source positions p'_i
Initialize $\lambda \leftarrow du^i$ and $\frac{dL}{df} \leftarrow 0$;
 $h \leftarrow 2/(N-1)$;
for $it = 1$ to $4N$ **do**
 for each grid cell (m, n) **do**
 Skip if (m, n) is a source corner;
 // Get and clear current adjoint
 $l \leftarrow \lambda_{m,n}$;
 $\lambda_{m,n} \leftarrow 0$;
 if $l = 0$ **then** continue;
 // Recompute neighbors
 $a \leftarrow \min(u_{m,n-1}^i, u_{m,n+1}^i)$;
 $b \leftarrow \min(u_{m-1,n}^i, u_{m+1,n}^i)$;
 // Get indices
 $i_a \leftarrow \text{argmin}(u_{m,n-1}^i, u_{m,n+1}^i)$;
 $i_b \leftarrow \text{argmin}(u_{m-1,n}^i, u_{m+1,n}^i)$;
 if $|a - b| \geq hf_{m,n}$ **then**
 // Gradient only flows in one direction
 if $a < b$ **then** $\lambda_{i_a} \leftarrow \lambda_{i_a} + l$;
 else $\lambda_{i_b} \leftarrow \lambda_{i_b} + l$;
 $f'_{m,n} \leftarrow f'_{m,n} + hl$;
 else
 // Gradient is split
 $d \leftarrow \sqrt{2(hf_{m,n})^2 - (a - b)^2}$;
 $\delta_a \leftarrow \frac{1}{2}(1 - \frac{a-b}{d})$;
 $\delta_b \leftarrow \frac{1}{2}(1 + \frac{a-b}{d})$;
 $\delta_f \leftarrow \frac{h^2 f_{m,n}}{d}$;
 $\lambda_{i_a} \leftarrow \lambda_{i_a} + l\delta_a$;
 $\lambda_{i_b} \leftarrow \lambda_{i_b} + l\delta_b$;
 $f'_{m,n} \leftarrow f'_{m,n} + l\delta_f$;
 end
 end
end
for each source p_i **do**
 $p'_i \leftarrow 0$;
 for each neighboring cell (m, n) **do**
 $l \leftarrow \lambda_{m,n}$;
 // distance derivative wrt source position
 $p'_i \leftarrow p'_i + \text{direction to } p_i \times lf_{m,n}$;
 end
end

5 Full Prompts

For the prompts used in our method, we augment them by appending the suffix: “*trending on artstation. black silhouette on plain white background. high quality.*”. Additionally, we also use the following prompt as negative conditioning: “*blurry. low-detail. low quality. extra limbs. deformed. disfigured.*”.

5.1 Figure 1: Generative Whimsy Jigsaws

From left to right:

(1) **FISH A** (12 pieces)

Overall *fish*

Individual *fish*

(2) **NATURE A** (12 pieces)

Overall *tree*

Individual Clockwise starting from 12 o'clock: *tree, leaf, bird, leaf, leaf, bird, tree, tree, bird, leaf, bird, tree*

(3) **ZODIAC A** (12 pieces)

Overall *circle*

Individual Clockwise starting from 12 o'clock: *mouse, cow, tiger, rabbit, chinese dragon, snake, horse, goat, monkey, chicken, dog, pig*

(4) **ANIMAL TREE** (7 pieces)

Overall *tree*

Individual Clockwise starting from 12 o'clock: *fish, cat, bird, chicken, dog, bear, mouse*

5.2 Figure 7: Textured Puzzles

From left to right:

(1) **HUMANS B**

Texture *Generate minimal vector line-art textures for this UV map (possibly just adding eyes and other minimal details) consisting of side profiles of humans. Each UV island represents a different shape. DO NOT change the positions of the UV islands. DO NOT go beyond the UV islands.*

(2) **DOG A**

Texture *Generate black-and-white minimal vector line-art dog textures for this UV map (possibly just adding eyes and other minimal details). Each UV island represents a different shape. DO NOT change the positions of the UV islands. DO NOT go beyond the UV islands.*

(3) **ANIMAL TREE**

Texture *Generate minimal vector line-art textures for this UV map (possibly just adding eyes and other minimal details) following the labels in red. Each UV island represents a different shape. DO NOT change the positions of the UV islands. DO NOT go beyond the UV islands.*

5.3 Figure 8: Failure Cases

From left to right:

(1) **FISH F** (20 pieces)

Overall *fish*

Individual *fish*

(2) **CRAB F** (8 pieces)

Overall *crab*

Individual *crab*

5.4 Figure 9: Ablations

From left to right:

(1) Explicit

Overall *cat*

Individual *cat*

(2) Implicit

Overall *bird*

Individual *bird*

(3) Slowness field f (no FFN)

Overall *human portrait*

Individual *male portrait (x4), female portrait (x4)*

5.5 Figure 11: VLM Generation

We used the same prompt for all three VLM models.

Prompt *First generate a silhouette of a cat, and partition it into 9 pieces such that each partition is in the shape of a cat, not necessarily the same shape. Then, write the SVG code for each partition boundary.*

5.6 Figure 12: Gallery (Single Prompt)

From top to bottom, left to right:

(1) **BEAR** (8 pieces)

Overall *bear*

Individual *bear*

(2) **BIRD A** (8 pieces)

Overall *bird*

Individual *bird*

(3) **BIRD B** (10 pieces)

Overall *bird*

Individual *bird*

(4) **CASTLE** (10 pieces)

Overall *castle*

Individual *castle*

(5) **CAR A** (6 pieces)

Overall *car*

Individual *car*

(6) **CAR B** (10 pieces)

Overall *car*

Individual *car*

(7) **CAT A** (12 pieces)

Overall *cat*

Individual *cat*

(8) **CAT B** (15 pieces)

Overall *cat*

Individual *cat*

(9) **DINOSAUR** (12 pieces)

Overall *dinosaur*

Individual *dinosaur*

(10) **DOG A** (12 pieces)

Overall *dog*

Individual *dog*

(11) **DOG B** (12 pieces)

Overall *dog*

Individual *dog*

(12) **DRAGON** (8 pieces)

Overall *dragon*
Individual *dragon*
 (13) **FISH B** (12 pieces)
Overall *fish*
Individual *fish*
 (14) **FISH C** (20 pieces)
Overall *fish*
Individual *fish*
 (15) **FLOWER A** (6 pieces)
Overall *flower*
Individual *flower*
 (16) **FLOWER B** (8 pieces)
Overall *flower*
Individual *flower*
 (17) **LEAF** (10 pieces)
Overall *leaf*
Individual *leaf*
 (18) **RABBIT A** (10 pieces)
Overall *rabbit*
Individual *rabbit*
 (19) **RABBIT B** (10 pieces)
Overall *rabbit*
Individual *rabbit*
 (20) **TREE** (8 pieces)
Overall *tree*
Individual *tree*

(7) **VEHICLES C** (8 pieces)
Overall *car*
Individual Top to bottom, left to right: *airplane, airplane, boat, boat, helicopter, helicopter, car, car*
 (8) **VEHICLES D** (8 pieces)
Overall *car*
Individual Top to bottom, left to right: *car, car, helicopter, helicopter, boat, boat, airplane, airplane*
 (9) **JAPAN A** (6 pieces)
Overall *circle*
Individual Top to bottom, left to right: *himeji castle, torii gate, cherry blossom flower, koi fish, koi fish, koi fish*
 (10) **JAPAN B** (6 pieces)
Overall *circle*
Individual Top to bottom, left to right: *himeji castle, torii gate, cherry blossom flower, koi fish, koi fish, koi fish*
 (11) **BIRD TREE** (8 pieces)
Overall *tree*
Individual Top to bottom, left to right: *tree, tree, tree, tree, tree, tree, tree, bird*
 (12) **NATURE B** (12 pieces)
Overall *tree*
Individual Top to bottom, left to right: *tree, tree, tree, tree, leaf, leaf, leaf, leaf, bird, bird, bird, bird*

5.7 Figure 13: Gallery (Multi-prompt)

From top to bottom, left to right:

(1) **USA** (6 pieces)
Overall *bald eagle*
Individual Top to bottom, left to right: *cactus, turkey, bald eagle, statue of liberty, teepee, cowboy hat*
 (2) **ZODIAC B** (12 pieces)
Overall *circle*
Individual Top to bottom, left to right: *rabbit, cow, chicken, pig, snake, dog, horse, mouse, chinese dragon, tiger, goat, monkey*
 (3) **HUMANS A** (8 pieces)
Overall *human portrait*
Individual Top to bottom, left to right: *male portrait, male portrait, male portrait, male portrait, female portrait, female portrait, female portrait, female portrait*
 (4) **HUMANS B** (8 pieces)
Overall *human portrait*
Individual Top to bottom, left to right: *male portrait, male portrait, male portrait, male portrait, female portrait, female portrait, female portrait, female portrait*
 (5) **VEHICLES A** (5 pieces)
Overall *car*
Individual Top to bottom, left to right: *helicopter, boat, train, car, airplane*
 (6) **VEHICLES B** (6 pieces)
Overall *car*
Individual Top to bottom, left to right: *motorcycle, bicycle, <>, helicopter, airplane, car*