AN INTELLIGENT WEB APPLICATION TO ACCELERATE ART REFERENCE DISCOVERY USING POSE ESTIMATION AND GENERATIVE IMAGE SYNTHESIS

Keyang (Kerry) Liu[1], Andrew Park[2]

[1]Claremont High School, 1601 N Indian Hill Blvd, Claremont, CA 91711
[2]University of California, Irvine, Irvine, CA 92697

ABSTRACT

Artists depend on reference images to study proportion, lighting, and movement, yet locating a reference that matches a specific visual intention remains slow because mainstream image search matches keywords rather than visual structure. This paper presents Reference Gallery, a web application that lets artists retrieve and generate art references through three channels: a written description, a full-body pose built on a draggable mannequin, and a color chosen from a color wheel. Library poses are indexed with a pose-estimation model and ranked by joint distance, while colors are ranked by distance from each image's dominant palette. When no stored reference is suitable, a moderated generation service synthesises a new one and adds it to the shared library. An internal ranking study over ten queries produced a mean relevance score of 7.2 out of 10, with pose queries scoring a perfect 10.0 and description and color queries averaging 6.0, indicating that structural channels outperform semantic ones here.

KEYWORDS

Art Reference Retrieval, Pose Estimation, Generative Image Synthesis, Content-Based Image Retrieval, Creativity Support Tools

1. INTRODUCTION

Reference images are a working tool of visual art practice. Artists consult them to study lighting, shading, proportion, and movement before committing marks to a surface, and interviews with novice visual artists identify quicker and better access to references as one of the central unmet needs of the early creative process [1]. The growth of online image collections has made hundreds of millions of pictures reachable from a browser, but abundance has not produced accessibility. Mainstream platforms retrieve images by matching query terms against captions, filenames, and surrounding page text, an approach whose mismatch with what a viewer actually perceives in a picture has been documented across two decades of image-retrieval research [2]. An artist looking for a figure in a particular stance, or for a composition in a particular color key, must therefore translate a visual intention into words, scan long result grids, discard most of what is returned, and repeat the cycle across several sites.

The motivation for this project came from preparing an AP 2-D Art and Design portfolio. Before beginning a drawing, the author routinely searched for references constrained by subject, human pose,

David C. Wyld et al. (Eds): BMLI, CACIT, EDUAIT, InWES – 2026
pp. 43-55, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.161804

and lighting condition at once. Keyword search handles the first constraint tolerably and the other two badly, because pose and palette are structural properties that resist description in a short text query. The cost is measured in time that would otherwise be spent drawing. This burden falls on a large and varied population: students, illustrators, concept artists, animators, and designers all begin from reference, and novice artists in particular report relying on it while they are still building visual vocabulary [11].

Two developments make a better approach practical. First, generative image models built on latent diffusion can now synthesise a plausible reference on demand rather than requiring that one already exist in a corpus [5], and text-image embedding models such as CLIP have made it routine to relate language and pictures in a shared representation [6]. A controlled study of DesignAID found that designers rated automatically generated images as significantly more inspirational, enjoyable, and useful than images obtained through a conventional Pinterest-based search [3]. Second, visual similarity search offers a way to surface colors, textures, shapes, and compositions that a user cannot easily name, as demonstrated by the Camera Obscurer system, which retrieves generated images from a seed photograph rather than a text string [4].

Pose is the constraint that benefits most from this shift. On-device pose estimators such as BlazePose recover thirty-three body keypoints from a single image in real time [7], which makes it feasible to index an entire reference library by body configuration. On the generation side, spatial conditioning methods including ControlNet [8] and pose-specialised adapters such as Stable-Pose [9] accept a skeleton as an explicit control signal, so a pose the user constructs can drive synthesis directly. Any system that generates images for users, however, inherits a safety obligation, since text-to-image models can be induced to produce unsafe imagery [10].

This paper presents **Reference Gallery**, a web application that combines these capabilities into a single search surface. It offers three ways to reach a reference: a written description, a full-body pose composed by dragging the joints of a mannequin, and a color chosen from a color wheel. Each channel first ranks the images already held in the library, and, when nothing stored is suitable, generates a new reference through a moderated backend and adds it to the shared collection. By replacing repeated keyword guessing across several sites with one query expressed in the terms artists already think in, the system aims to shorten the search step and return results that sit closer to the user's intention.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Keeping a Draggable Skeleton Anatomically Consistent

The pose channel depends on a mannequin whose joints a user can move freely while the figure continues to read as a human body. An early approach built the model from a segmented skeleton illustration, treating each limb as an independent element. Rotating one joint then displaced the parts attached to it, and limbs stretched or detached as the figure was manipulated. A workable model would need every part bound to a fixed pivot and every bone held at a constant length, so that dragging a hand rotates the forearm about the elbow instead of extending it. An SVG mannequin with explicit pivot points and enforced bone lengths could satisfy both requirements, and the resulting rig could be validated by posing it and measuring the agreement between the requested and rendered configuration.

2.2. Obtaining Varied, Usable References from a General-Purpose Image Model

The generation channel relies on an external image model that was not built specifically for art reference. Early prompts produced narrow output: monochrome line sketches in a single house style, repetitive enough that generating a second image added little to the first. References of that kind are of limited value to an artist exploring alternatives, and this echoes a broader finding that generative inspiration boards yield quantity without matching diversity. Addressing this would require treating the prompt and the generation settings as parameters to be tuned rather than fixed, iterating on phrasing, quality, and style hints until the service returned varied, multicolored references across a range of rendering styles instead of one repeated sketch idiom.

2.3. Providing Affordable Authentication, Storage, and Content Safety

A public site that stores user accounts and calls a paid generation API needs authentication, a database, file storage, hosting, rate limiting, and content safety. Implementing these from scratch is slow and error-prone, and a fixed-capacity deployment would incur cost during the early period when the user base is too small to justify it. A managed backend billed by actual usage would address both concerns, letting the project store credentials and generated images without exposing private data while scaling cost with demand. Because generation is user-driven, the same layer would also have to enforce per-user quotas and screen every generated image before publishing it to the shared library.

3. SOLUTION

Reference Gallery is a React single-page application backed by Firebase for authentication, database, file storage, hosting, and server-side logic. It connects three major components: pose search, color search, and a database layer that unifies them. Opening the site presents the reference library, which the user can browse or filter by keyword. If no stored image is suitable, the user may generate one; because generation calls a paid external service, it requires a signed-in account so that quotas and history can be enforced. The pose and color pages provide the two structural search channels. In every case a newly generated image is written back into the shared library, so the collection grows as the site is used. Figure 1 shows how the three components relate.

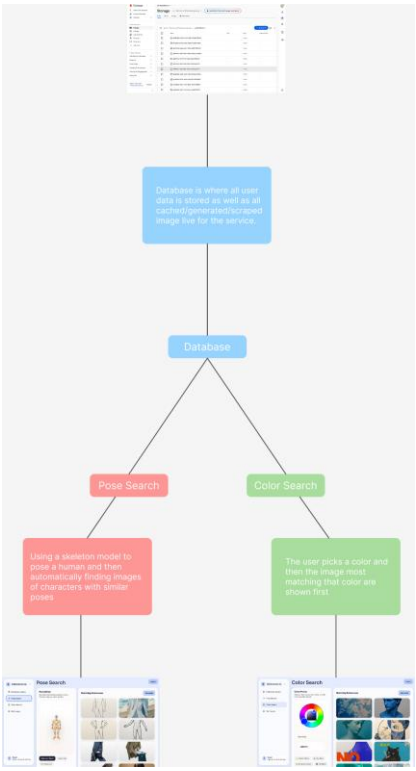

Figure 1. Overview of the solution

Pose Search lets a user state a body configuration directly instead of describing it. The page presents a draggable mannequin; the program compares the resulting configuration against pose data extracted from every image in the library and orders results by similarity. Joint positions for library images are recovered with MediaPipe, and the comparison reduces each figure to a set of named joints so that two poses can be scored by how far apart their corresponding joints lie. Signed-in users may also generate a new reference from the pose they have built. Figure 2 shows the page with a pose in progress and the ranked matches beside it.

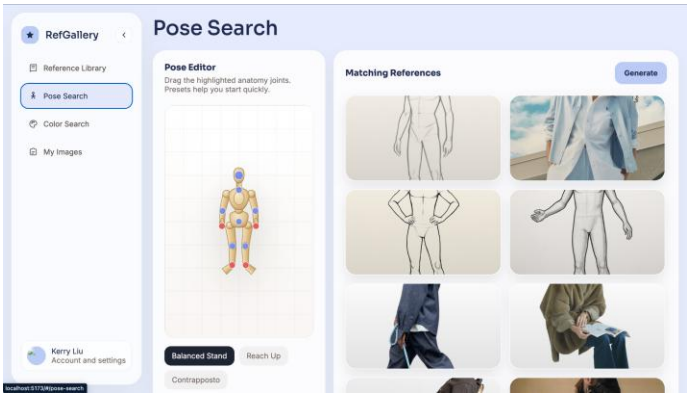

Figure 2. The Pose Search page, showing the pose editor and ranked matches

```
function getSkeletonRig(points) {
  const torsoVector = getUnitVector(points.neck, points.hip);
  const sideVector = { x: -torsoVector.y, y: torsoVector.x };
  const leftShoulder  = offsetPoint(points.neck, sideVector,  0.135, 0.034);
  const rightShoulder = offsetPoint(points.neck, sideVector, -0.135, 0.034);
  const leftHip  = offsetPoint(points.hip, sideVector,  0.092, 0.032);
  const rightHip = offsetPoint(points.hip, sideVector, -0.092, 0.032);
  const leftElbow = fixedEndpoint(leftShoulder, points.leftElbow,
      skeletonBoneLengths.upperArm, { x: -0.9, y: 0.35 });
  const rightElbow = fixedEndpoint(rightShoulder, points.rightElbow,
      skeletonBoneLengths.upperArm, { x: 0.9, y: 0.35 });
  const leftWrist = fixedEndpoint(leftElbow, points.leftHand,
      skeletonBoneLengths.forearm, getUnitVector(leftShoulder, leftElbow));
  const rightWrist = fixedEndpoint(rightElbow, points.rightHand,
      skeletonBoneLengths.forearm, getUnitVector(rightShoulder, rightElbow));
  const leftKnee = fixedEndpoint(leftHip, points.leftKnee,
      skeletonBoneLengths.thigh, { x: -0.2, y: 1 });
  const rightKnee = fixedEndpoint(rightHip, points.rightKnee,
      skeletonBoneLengths.thigh, { x: 0.2, y: 1 });
  // ... remaining joints omitted
```

Figure 3. Excerpt of getSkeletonRig() in App.jsx, which derives the mannequin rig

The excerpt in Figure 3 runs whenever the mannequin is initialised and again each time a joint is dragged. `getSkeletonRig` takes the current control points and returns a complete rig. It first calls `getUnitVector` to obtain a normalised direction from the neck to the hip, giving the torso axis; `sideVector` is that axis rotated a quarter turn, providing a lateral direction. `offsetPoint` then displaces a point along a vector, which is how the shoulders and hips are placed symmetrically about the torso centre line rather than stored as independent coordinates. `fixedEndpoint` places a point in the direction of a target at an exact distance from its origin, so each call fixes a limb segment to its entry in `skeletonBoneLengths`. This is what prevents the stretching that the segmented-image approach suffered from: the elbow is constrained to lie one upper-arm length from the shoulder no matter where the user drags, and the wrist one forearm length from the elbow. The trailing vectors supply a fallback direction when the target coincides with the origin and no direction can be derived.

Color Search applies the same ranking idea to palette. The user selects a hue on a color wheel, types a hex value, or starts from a named preset; the program compares the selection against the dominant colors recorded for each library image, computes a color distance, and orders results from nearest to furthest. As with pose, a signed-in user can generate a new image built around the selected color when the library holds nothing suitable. Figure 4 shows the page with a warm yellow selected.

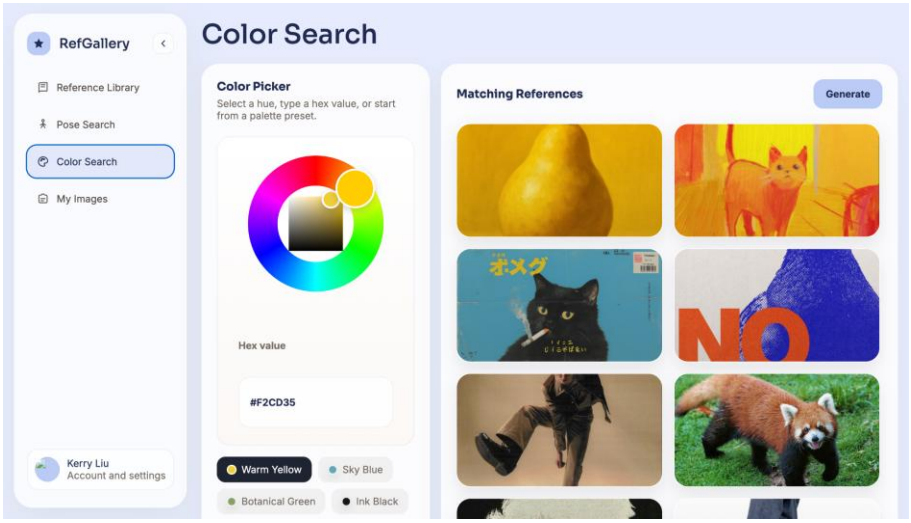

Figure 4. The Color Search page with a selected hue and ranked matches

```
function ColorSearchPage({ requireAuth }) {
  return (
    <div className="page">
      <PageHeading
        title="Color Search"
        subtitle="Pick a color to find reference images with matching
palettes."
      />
      <section className="pose-layout">
        <div className="panel">
          <div className="panel-header">
            <div>
              <h3>Color Picker</h3>
              <p>Select a hue, type a hex value, or start from a palette
preset.</p>
            </div>
          </div>
          <ColorPicker value={selectedColor} onChange={setSelectedColor} />
          <div className="preset-row">
            {colorPresets.map((preset) => (
              <button
                className={`preset-button color-preset-button ${
                  preset.value.toLowerCase() === selectedColor.toLowerCase() ?
'active' : ''
                }`}
                key={preset.id}
                onClick={() => { setSelectedColor(preset.value); }}
                type="button"
              >
```

Figure 5. Excerpt of ColorSearchPage() in App.jsx, which builds the color search interface

Figure 5 runs when the user opens the color page and re-runs on every interaction with the picker. `ColorSearchPage` constructs the page and holds the current selection in `selectedColor`. The `ColorPicker` element is the wheel itself, and it is a controlled component: it receives the current

value and reports changes back through `setSelectedColor`, so the wheel and the ranked results always agree on a single value. The `preset-row` block maps over `colorPresets`, the named swatches shown beneath the wheel, and renders a button for each. Its template literal compares the preset against the current selection case-insensitively and applies an `active` class when they match, which is what marks the chosen swatch in the interface. Clicking a preset simply writes its value into the same state the wheel writes to, so the two input methods are interchangeable.

The database and generation backend is the layer that ties the two search channels together. Firebase stores reference images, user records, extracted pose data, and saved collections, and security rules restrict each user to the records they own; saved images appear on a “My Images” page. Figure 6 shows the storage bucket holding the published reference images. Generation itself is not performed in the browser. It runs in a Firebase Cloud Function, which holds the API credentials, enforces quotas, and screens output before anything reaches the shared library.

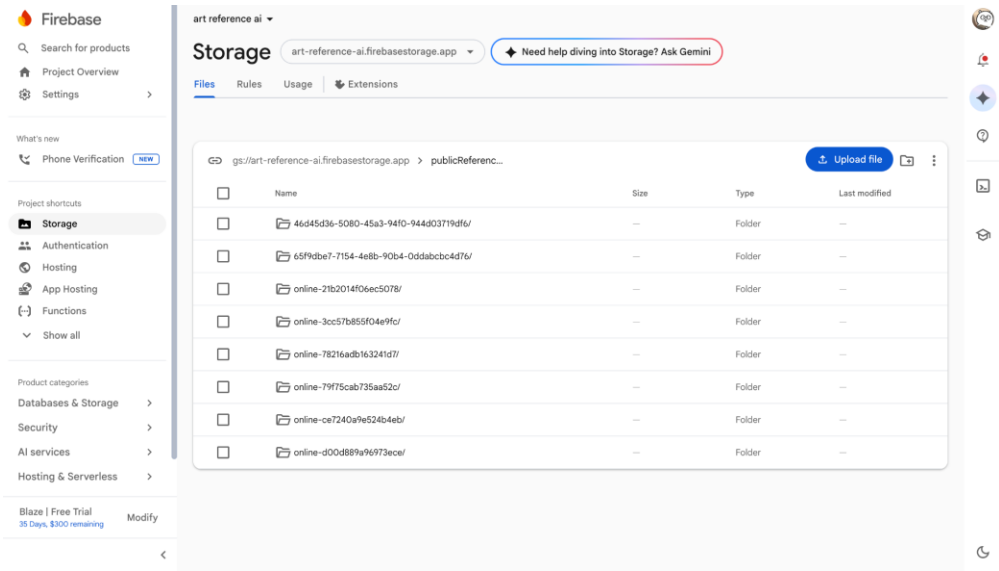

Figure 6. Firebase Storage holding the published reference images

```
const { initializeApp } = require('firebase-admin/app');
const { FieldValue, getFirestore } = require('firebase-admin/firestore');
const { getStorage } = require('firebase-admin/storage');
const { createHash, randomUUID } = require('node:crypto');
const { defineSecret } = require('firebase-functions/params');
const { HttpsError, onCall } = require('firebase-functions/v2/https');
const { onObjectFinalized } = require('firebase-functions/v2/storage');

initializeApp();

const openAiApiKey = defineSecret('OPENAI_API_KEY');
const moderationModel = process.env.OPENAI_MODERATION_MODEL || 'omni-moderation-
latest';
const reviewScoreThreshold = Number(process.env.OPENAI_MODERATION_REVIEW_THRESHOLD
|| 0.35);
const imageGenerationModel = process.env.OPENAI_IMAGE_MODEL || 'gpt-image-1-mini';
const generationDailyLimit = Number(process.env.GENERATION_DAILY_LIMIT || 3);
const generationCooldownSeconds = Number(process.env.GENERATION_COOLDOWN_SECONDS
|| 30);
const allowedSearchTypes = new Set(['text', 'color', 'pose']);
const allowedCategories = new Set([
  'Figure', 'Animals', 'Hands & Feet', 'Faces',
  'Still Life', 'Environment', 'Clothing', 'Plants',
]);
const imageSupportedCategories = [
  'sexual', 'sexual/minors', 'violence', 'violence/graphic', 'self-harm',
];
```

Figure 7. Excerpt of the Cloud Function in functions/index.js, showing its configuration

Figure 7 shows the head of the Cloud Function, which is invoked when a user requests a generation and again when a new object is finalised in storage. The first block imports the Firestore, Storage, and Functions bindings and initialises the admin client, which runs with server privileges rather than the caller's. `defineSecret` keeps the API key out of the source and out of the browser, so a user can trigger generation without ever holding the credential. The configuration constants encode the project's policy decisions as values rather than scattered logic: which models to call, a moderation review threshold of 0.35, a limit of three generations per user per day, a thirty-second cooldown, the three permitted search types, the subject categories a reference may be filed under, and the five moderation categories that will block publication. Every environment variable is read with a fallback, so the function still runs with sane defaults if a value is missing. It is worth being explicit about one limit: pose data is used to rank images that already exist, and the generation path conditions on the pose only through the text prompt built from it, not through a skeleton passed to the image model as a spatial control signal.

4. EXPERIMENT

4.1. Ranking Accuracy Across Description, Pose, and Color Queries

The value of the application rests on whether its three channels actually place the right reference near the top of the results, so the first experiment measured ranking quality directly. Ten queries were prepared, four written descriptions, three poses, and three colors, and for each query the images in the library that should have ranked first were designated in advance. Each query was then run against the

live site and scored from 0 to 10, where 0 means no returned image matched the intent and 10 means the results were ordered exactly as designated. Intermediate values reflect the proportion of returned images that were valid: a search for "cat" that returned three images of which one was a red panda scored 6. Scoring against a pre-designated ranking rather than a subjective impression keeps the judgement consistent across the three channels, which is what makes them comparable.

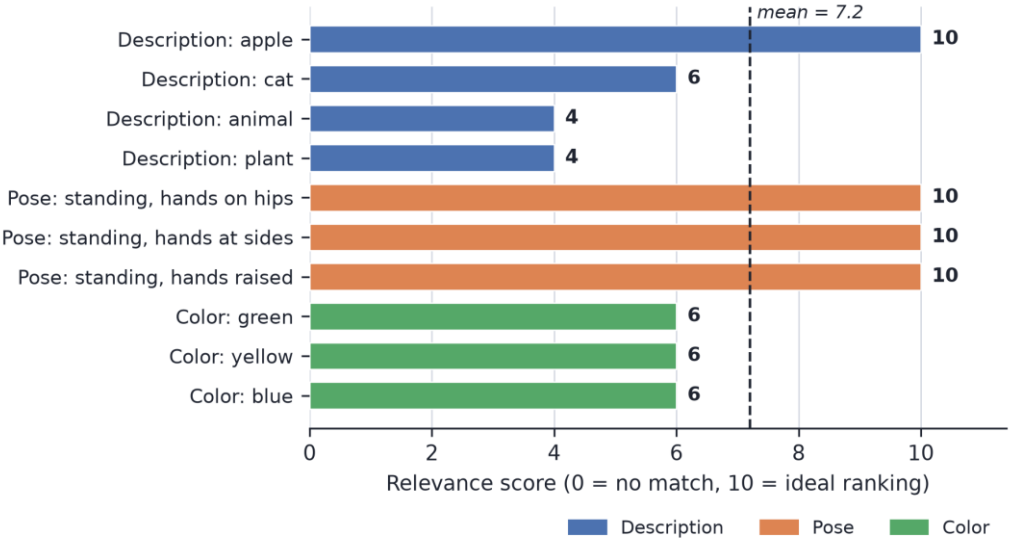

Figure 8. Relevance score for each of the ten test queries

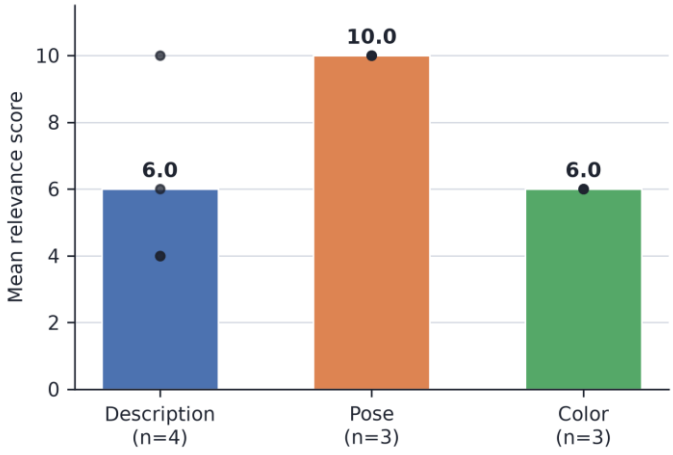

Figure 9. Mean relevance score by search channel

Scores ranged from 4 to 10 with a mean of 7.2 and a median of 6. The three pose queries each scored 10, giving that channel a mean of 10.0, while description and color queries both averaged 6.0. The split is the most informative part of the result. Pose search compares numeric joint coordinates against numeric joint coordinates, a well-posed geometric comparison, and it performed accordingly on the standing poses tested. Description search performed worst on the two broadest terms, "animal" and

"plant", which scored 4 apiece, against 10 for the narrow term "apple". Broad categories fail because the library is small and unevenly populated, so a category term matches few stored images and the ranking is forced to fall back on weak partial matches. Color search scored a uniform 6, consistent with ranking by a single dominant color: an image whose overall palette matches can still be dominated by a different subject, which the distance measure cannot detect. The results indicate that the structural channel is the strongest part of the system and that description accuracy is bounded by library coverage rather than by the ranking method.

4.2. Adversarial Testing of the Generation Moderation Pipeline

The second blind spot is safety. Because any signed-in user can generate an image that is then published into the shared library, a moderation failure would expose every visitor to the result, so the screening layer needs to be tested rather than assumed. The experiment was designed as a red-teaming exercise: rather than uploading a prepared set of benign and objectionable images, the tester attempts to elicit objectionable output from the generation endpoint itself using adversarial prompts spanning harassment, violence, and adult content, and records the proportion of attempts the moderation layer blocks. Red-teaming is the appropriate design here because it tests the path an actual abuser would take, through prompt phrasing, instead of testing the classifier on images that never pass through the prompt interface. Figure 10 shows the policy the pipeline enforces, read from the deployed configuration.

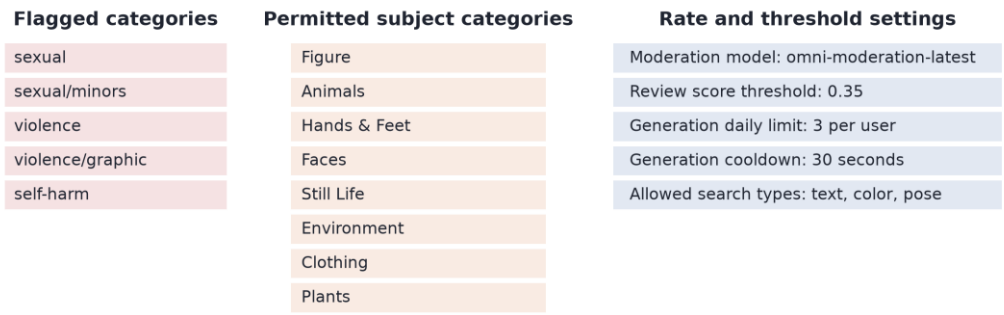

Figure 10. Moderation policy enforced by the generation backend

The pipeline screens generated output against five categories, blocks publication above a review score of 0.35, and additionally constrains abuse volume through a three-per-day generation limit and a thirty-second cooldown, so a determined attacker is rate-limited even where the classifier is imperfect. It must be reported plainly that this experiment was designed but not completed: no trial counts or block rates were recorded, and no quantitative result for the moderation layer is available. The figure therefore documents the configuration under test, not an outcome. Completing the trial is the most immediate piece of outstanding work on the project, since the defence it would measure is the one protecting a shared, publicly visible library.

5. Related Work

Pinterest is the platform most artists reach for first, and it is effective at what it does: it aggregates an enormous user-contributed collection, lets people save and combine images into boards, and surfaces

visually related pins from a selected image [18]. Its limitation for this use case is that discovery is bounded by what has already been posted and indexed, and it offers no way to specify a pose or a palette as a query, nor any generative fallback when the collection holds nothing suitable. The controlled comparison in DesignAID is instructive here: designers rated generated images as more inspirational, enjoyable, and useful than the results of a Pinterest-based search, though the same body of work cautions that generative boards can deliver quantity while remaining narrower in diversity than traditional methods [3][12]. Reference Gallery keeps a browsable library but adds pose and color as first-class query types and a generation path when retrieval returns nothing usable.

The closest tool to the pose channel is an open pose-search site that matches drawing references by manipulating a skeleton model [19]. It works from a large image collection and demonstrates that structural query is viable for this audience. Its interaction cost is higher, however: the user must both move a part of the skeleton and select the corresponding body part in the search field, and a query fails if either step is incomplete. It also appears to match on individual parts rather than whole-body configurations. Reference Gallery matches on the full rig produced by dragging the mannequin, with no separate part selection, which is possible because library poses are indexed automatically with a keypoint estimator of the kind introduced by BlazePose [7]. Research on pose-conditioned synthesis such as Stable-Pose points to the further step of passing the skeleton to the generator as an explicit spatial control [9].

A third approach abandons photographs entirely. JustSketchMe supplies posable three-dimensional figures and props, letting an artist assemble a scene and view it from any angle [20]. For exact anatomical control it is stronger than any retrieval system, since the artist builds the pose rather than searching for it, but it offers no generative component and a limited set of textures and materials, so the output is a mannequin render rather than a rendered reference. Work on pose-guided synthesis has shown that a posed figure can instead drive photorealistic generation while preserving appearance [17]. Reference Gallery occupies the middle ground, using a posable figure as a query rather than as the final artefact. Positioning matters as much as capability: studies of how designers and novice artists receive such systems find acceptance depends on whether the tool is experienced as a partner in ideation rather than a replacement for the artist [13][14][15][16].

6. CONCLUSIONS

This project set out to shorten the gap between an artist's visual intention and a usable reference image, and it did so by making pose and color first-class queries alongside description, with moderated generation as a fallback when retrieval fails. Testing across ten queries produced a mean relevance score of 7.2 out of 10. The result divided sharply by channel: pose search scored a perfect 10.0, while description and color each averaged 6.0. Structural queries, where a numeric configuration is compared against numeric configurations, proved the most reliable part of the system, whereas description accuracy was limited by how thinly the library covers broad categories and color accuracy by the coarseness of ranking on a single dominant hue. The second experiment, an adversarial test of the moderation pipeline, was designed but not carried out, so no safety measurement is claimed here.

Several limitations follow directly from those findings. The library is small and unevenly distributed across subjects, which is the main constraint on description search and the first thing more time would be spent on, both by broadening the collection and by improving the scraper that populates it. Color ranking on one dominant hue should give way to a comparison over a full palette so that an image is not surfaced merely because its background matches. The two-dimensional mannequin restricts poses

to a plane and offers no control over extremities; replacing it with a three-dimensional rig would permit foreshortening, viewpoint changes, and finer articulation such as individual fingers. On the generation side, the pose currently reaches the image model only as text, and passing the skeleton as an explicit spatial control would tie generated output to the requested pose far more tightly. Completing the moderation experiment remains the highest priority.

Reference Gallery shows that an artist's query does not have to be a sentence. A pose built by dragging a figure and a color chosen from a wheel are both precise statements of intent that keyword search cannot express, and the pose results in particular suggest that when a query can be posed structurally, it should be. The system is usable today for pose and color lookup, and its weakest results point at fixable causes rather than at the approach itself.

REFERENCES

[1] Peng, Cindy, Alice Qian, Lingqing Jin, Jiayin Chen, Erica X. Han, Paul Pu Liang, Hong Shen, Haiyi Zhu, and Jane Hsieh. "Exploring Opportunities to Support Novice Visual Artists' Inspiration and Ideation with Generative AI." arXiv preprint arXiv:2509.24167 (2025).

[2] Datta, Ritendra, Dhiraj Joshi, Jia Li, and James Z. Wang. "Image Retrieval: Ideas, Influences, and Trends of the New Age." ACM Computing Surveys 40.2 (2008): 1-60.

[3] Cai, Alice, Steven R. Rick, Jennifer L. Heyman, Yanxia Zhang, Alexandre Filipowicz, Matthew K. Hong, Matt Klenk, and Thomas W. Malone. "DesignAID: Using Generative AI and Semantic Diversity for Design Inspiration." Proceedings of the ACM Collective Intelligence Conference (2023): 61-72.

[4] Singh, Dilpreet, Nina Rajcic, Simon Colton, and Jon McCormack. "Camera Obscurer: Generative Art for Design Inspiration." Computational Intelligence in Music, Sound, Art and Design (EvoMUSART 2019), Lecture Notes in Computer Science 11453 (2019): 51-68.

[5] Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022): 10684-10695.

[6] Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. "Learning Transferable Visual Models From Natural Language Supervision." Proceedings of the 38th International Conference on Machine Learning, PMLR 139 (2021): 8748-8763.

[7] Bazarevsky, Valentin, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. "BlazePose: On-device Real-time Body Pose Tracking." arXiv preprint arXiv:2006.10204 (2020).

[8] Zhang, Lvmin, Anyi Rao, and Maneesh Agrawala. "Adding Conditional Control to Text-to-Image Diffusion Models." Proceedings of the IEEE/CVF International Conference on Computer Vision (2023): 3836-3847.

[9] Wang, Jiajun, Morteza Ghahremani, Yitong Li, Björn Ommer, and Christian Wachinger. "Stable-Pose: Leveraging Transformers for Pose-Guided Text-to-Image Generation." Advances in Neural Information Processing Systems 37 (2024).

[10] Qu, Yiting, Xinyue Shen, Xinlei He, Michael Backes, Savvas Zannettou, and Yang Zhang. "Unsafe Diffusion: On the Generation of Unsafe Images and Hateful Memes From Text-To-Image Models." Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (2023): 3403-3417.

[11] Zhang, Shuning, and Shixuan Li. "'Confrontation or Acceptance': Understanding Novice Visual Artists' Perception towards AI-assisted Art Creation." arXiv preprint arXiv:2410.14925 (2024).

[12] Ranscombe, Charlie, Linus Tan, Mark Goudswaard, and Chris Snider. "Inspiration or Indication? Evaluating the Qualities of Design Inspiration Boards Created Using Text to Image Generative AI." Proceedings of the Design Society 4 (2024): 2207-2216.

[13] Yu, Yi-Chen. “Research on Emotion-Based Inspiration Mechanism in Art Creation by Generative AI.” Mathematics 13.16 (2025): 2597.
[14] Vechtomova, Olga. “Between Predictability and Randomness: Seeking Artistic Inspiration from AI Generative Models.” arXiv preprint arXiv:2506.12634 (2025).
[15] Kim, Kyungsun, Jeongyun Heo, and Sanghoon Jeong. “Tool or Partner: The Designer’s Perception of an AI-Style Generating Service.” Artificial Intelligence in HCI (HCII 2021), Lecture Notes in Computer Science 12797 (2021): 241-259.
[16] Chung, John Joon Young. “Artistic User Expressions in AI-powered Creativity Support Tools.” Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (2022): 1-4.
[17] Sun, Wei, Jawadul H. Bappy, Shanglin Yang, Yi Xu, Tianfu Wu, and Hui Zhou. “Pose Guided Fashion Image Synthesis Using Deep Generative Model.” arXiv preprint arXiv:1906.07251 (2019).
[18] Pinterest. Pinterest, Inc. Accessed 18 July 2026. https://www.pinterest.com/
[19] x6ud. “Pose Search.” Accessed 18 July 2026. https://x6ud.github.io/pose-search/
[20] JustSketchMe. Accessed 18 July 2026. https://justsketch.me/

©2026 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.