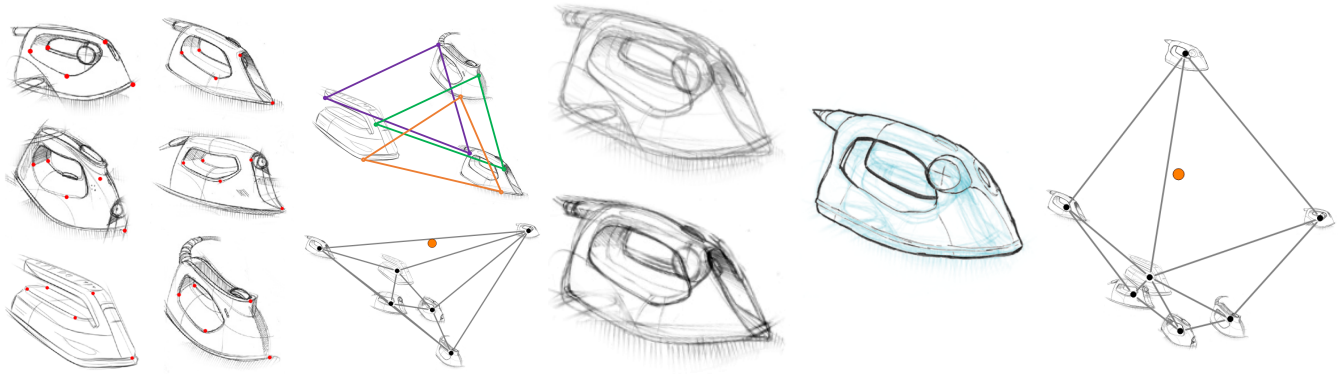
SketchSoup: Exploratory Ideation using Design Sketches

R. Arora[1,2,3] and I. Darolia[2] and V. P. Namboodiri[2] and K. Singh[1] and A. Bousseau[3]

[1]DGP Lab, University of Toronto, Canada
[2]Department of Computer Science, IIT Kanpur, India
[3]GraphDeco team, Inria Sophia-Antipolis, France



(a) *Input sketch images* **(b)** *Registration (top) and interpolation space (bottom)* **(c)** *Warping (top) and non-uniform blending (bottom)* **(d)** *Oversketching* **(e)** *Augmenting the interpolation space*

Figure 1: *SketchSoup takes an unstructured set of sketches as input, along with a small number of correspondences (shown as red dots)* ***(a)****, registers the sketches using an iterative match-warp algorithm harnessing matching consistency across images (****(b)****, top), and embeds the sketches into a 2D interpolation space based on their shape differences (****(b)****, bottom). Users can explore the interpolation space to generate novel sketches, which are generated by warping existing sketches into alignment(****(c)****, top), followed by spatially non-uniform blending (****(c)****, bottom). These interpolated sketches can serve as underlay to inspire new concepts* ***(d)****, which can in turn be integrated into the interpolation space to iteratively generate more designs* ***(e)****. (Some sketches courtesy Spencer Nugent and Mike Serafin.)*

Abstract

A hallmark of early stage design is a number of quick-and-dirty sketches capturing design inspirations, model variations, and alternate viewpoints of a visual concept. We present SketchSoup, a workflow that allows designers to explore the design space induced by such sketches. We take an unstructured collection of drawings as input, along with a small number of user-provided correspondences as input. We register them using a multi-image matching algorithm, and present them as a 2D interpolation space. By morphing sketches in this space, our approach produces plausible visualizations of shape and viewpoint variations despite the presence of sketch distortions that would prevent standard camera calibration and 3D reconstruction. In addition, our interpolated sketches can serve as inspiration for further drawings, which feed back into the design space as additional image inputs. SketchSoup thus fills a significant gap in the early ideation stage of conceptual design by allowing designers to make better informed choices before proceeding to more expensive 3D modeling and prototyping. From a technical standpoint, we describe an end-to-end system that judiciously combines and adapts various image processing techniques to the drawing domain – where the images are dominated not by color, shading and texture, but by sketchy stroke contours.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms

1. Introduction

The early ideation stage of conceptual design is dominated by rapidly drawn freehand sketches whereby designers externalize their imagination into an evolving design (Figure 2). Such sketches are sparse and focus on capturing the essence of the shape being designed. They allow designers to quickly create and communicate a mental design space to peers and clients, by expressing model variations in the drawings and by showing different viewpoints; often leaving regions of the drawing ambiguous and subject to interpretation [ES11]. Automatically turning this mental design space into a computationally explicit one is the important but largely unaddressed problem for which we present our solution – *SketchSoup* (Figure 1).

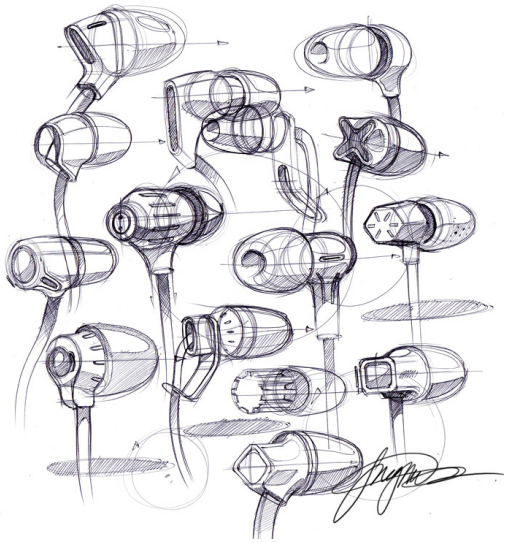


Figure 2: *Designers explore the shape of a concept by drawing many variations from different viewpoints. Drawing by Spencer Nugent on sketch-a-day.com*

Design exploration at the ideation stage has great significance as it can catch and avoid design problems that are costly later in the design pipeline. Yet, while post-ideation conceptual modeling has been well studied [NISA07, BBS08, SBSS12, XCS*14], there is little research, barring tools for architectural drawing [DXS*07, PKM*11], that supports the earlier ideation stage of the design process. Conceptual modeling tools require the designer to follow drawing principles and create simplified vector sketches, slowing down a designer and distracting from the central ideation goal of exploring a design space quickly.

Our sketches are dominated by many imperfect and often incomplete strokes – an artifact of the efficiency with which they are executed – and further convey the early and unfinished nature of the design. Vectorizing and lifting these sketches into 3D for the purpose of constructing a 2D design space is thus both difficult [OK11, BC13] and unnecessary. While crystallized product design concepts can be represented by a cleaned-up network of sketched strokes with coherent geometric properties [XCS*14], this is not an assumption we can make of arbitrary drawings at the ideation stage. Moreover, ideation and early concept drawings are often executed on paper, as and when inspiration strikes, necessitating the need for handling raster inputs. However, image-based modeling and rendering techniques designed for natural [CDSHD13] or even cartoon-like images [RID10] are not directly applicable. Unlike natural images, ideation sketches are sparse, with only a small minority of pixels providing information for image matching. Such sketches are also noisy, and may represent inaccurate or inconsistent projections of a depicted shape, confounding view calibration methods such as structure-from-motion [SSS06]. Model variations depicted in sketches further requires estimating non-rigid transformations between the sketches.

Our contribution is the first approach to enable the interactive exploration of the continuous design space induced by a collection of rough ideation sketches. Smooth view and shape transitions help understand the relationships between different design alternatives [HR07], and the interpolated drawings can serve either as design alternatives themselves or as inspirational underlays for the creation of new concepts.

From a technical standpoint, SketchSoup is enabled by the careful design of an end-to-end solution that accounts for properties of design sketches to successfully perform sketch filtering, registration, warping and blending. We employ a multi-image matching algorithm to identify the common features present in several sketches while neglecting the strokes that are specific to each sketch. Designers can optionally refine these matches. We then exploit this information to guide both image warping and image blending. In particular, we propose a novel image blending scheme that adjusts the contribution of the strokes according to the number of images where they appear: strokes that are present in many sketches are persistent throughout interpolation, while strokes that are only present in one sketch disappear quickly. This scheme reduces ghosting artifacts when creating novel sketches from existing ones. Finally, we embed the sketches in a 2D interpolation space using multi-dimensional scaling, where the relative distance between the sketches reflects their similarity as measured by their motion field magnitude. When dealing with multi-view sketches, motion due to shape variation is typically smaller in magnitude than that due to changes of viewpoints. The embedding thus provides a plausible estimate of relative camera positions, which gives designers the illusion of performing small rotations of the object in 3D.

2. Related Work

Our approach brings together the areas of sketch-based modeling and image-based rendering. We briefly discuss these two domains in the context of SketchSoup.

Sketch-based modeling and interpolation. Sketch based modeling systems aim at creating consistent 3D models from drawings [OSSJ09]. Multi-view approaches adopt an iterative workflow where users progressively sketch strokes over existing or transient surfaces [NISA07, BBS08]. Single-view algorithms strive to estimate a complete 3D model from a clean vector drawing [XCS*14]. We see our approach as a preliminary step for sketch-based modeling by allowing designers to explore a range of design concepts using rough unstructured preparatory drawings in 2D, well before

attempting to model anything in 3D. Sketch-based 3D modelers also assume and construct a unique and definitive 3D shape representation for any sketch, while our method is meant to iteratively explore a collective of *variations* on a design concept, from which to select one or more to take forward along the design pipeline.

Our approach is closer in spirit to drawing interpolation methods. In particular, Rivers et al. [RID10] describe a system to produce view interpolations from cartoon drawings that resemble ours. However, their method relies on vector drawings that are deformed by the user in each keyframe, which implicitly provides perfect correspondences. We also share the motivation of Baxter and Anjyo [BA06], who introduce a *doodle space* to create new drawings by interpolating a few examples. However, their system takes, as input, vector drawings executed using the same drawing sequence, which makes their stroke-matching algorithm inapplicable to the rough bitmap sketches we target. Similarly, the later system by Baxter et al. [BBA09] only registers the outer boundary of 2D shapes, while we also register inner strokes. Our goal is also related to the one of Shao et al. [SLZ*13] who interpret sketches of objects composed of articulated parts. Here again, user intervention is necessary to model the object with simple geometric shapes. We instead strive to interpolate between 2D sketches without requiring any explicit 3D reconstruction. Our approach thus shares natural similarities with the problem of 2D cartoon registration and inbetweening [SDC09, WNS*10, XWSY15], such as the concept of alternating between feature matching and shape-preserving regularization to warp one image onto another [SDC09]. However, while successive keyframes of cartoon animations have many features in common, we aim at registering design sketches that have similar content overall but different details at the stroke level. Finally, one of the applications enabled by SketchSoup is to generate morphed sketches that designers can use as underlays to draw new concept. This use of underlays as guidance is similar in spirit to the ShadowDraw system [LZC11], although we aim at registering and morphing a small number of user-provided sketches rather than blending a large number of roughly aligned photographs.

Image-based rendering and morphing. Image-based rendering methods vary wildly in terms of their target input complexity [LH96, CW93, CDSHD13]. Our method is most comparable to approaches which use implicit geometric information, that is, 2D correspondences between images [SD96]. Our work is also greatly inspired by the PhotoTourism system [SSS06] that provides interactive view interpolation as a way to explore unstructured photo-collections of touristic sites. Similarly, our system offers a novel way to experience the design space captured by an unstructured set of sketches. However, while PhotoTourism exploits structure-from-motion to register photographs of a static scene, we face the challenge of registering line drawings that contain variations of viewpoints, shapes, and even styles. We thus share the motivation of Xu et al. [XWL*08] and Averbuch-Elor et al. [AECOK16] who register and order multiple photographs of similar objects to create animations of animal motion and view and shape transitions respectively. However, while Xu et al. and Averbuch-Elor et al. select a subset of images to form a smooth 1D animation path, we seek to order all the provided drawings over a 2D interpolation space. In addition, they only register the outer boundaries of the objects, while our algorithm also builds correspondences between interior parts.

Recent methods have improved on the classical morphing technique by computing dense correspondences using optical flow [MHM*09], or by computing a half-way image for better alignment [LLN*14]. However, these techniques rely on pixel neighborhoods to compute dense motion fields over natural images, which is poorly suited to sparse line drawings where many neighborhoods look alike. Our approach instead builds upon a recent multi-image matching algorithm [ZJLYE15], which we adapt to work well with sketch input.

3. Registering Concept Sketches

High quality morphing requires high quality correspondences between images. However, design sketches contain many sources of variations that make this requirement a challenge: they are drawn from different viewpoints, they represent slightly different shapes, and they are often polluted with decorative lines and hatching.

Fortunately, all the sketches in a collection represent a similar object. The key to the success of our system is to leverage the redundancy offered by all the sketches to be robust to the variations specific to each sketch. In particular, we build on the concept of *cycle consistency*, as introduced in related work on shape and image matching [NBCW*11, ZKP10, ZJLYE15]. This concept can be summarized as follow: if point p in sketch $\mathcal{S}_i$ matches point q in sketch $\mathcal{S}_j$, and if point q in sketch $\mathcal{S}_j$ matches point r in sketch $\mathcal{S}_k$, then by transitivity point p should also match point r. Cycle consistency offers a way to detect high quality matches, as well as to improve low quality matches by replacing them with better candidates through transitivity.

Our approach follows the general methodology of the FlowWeb algorithm [ZJLYE15], which optimizes an initial set of matches by iterating three steps:

1. **Cycle-consistency score**. For each pair of images, for each match, count how many other images confirm the match as part of a consistent cycle.
2. **Cycle-consistency improvement**. For each pair of images, for each match, replace it with an alternative if it increases its cycle-consistency score.
3. **Spatial propagation**. For each pair of images, update the matches by propagating information from high quality matches to their low-quality spatial neighbors.

We refer the interested reader to [ZJLYE15] for a detailed description of the FlowWeb algorithm. We now describe how we tailored this algorithm to the specifics of design sketches.

3.1. Sketch pre-processing

Hatching removal. Our goal is to build correspondences between the sketch lines that represent contours and surface discontinuities. However, these lines are often polluted by repetitive hatching lines that convey shading and shadows. As a pre-process, we blur out hatching lines by applying the rolling guidance filter [ZSXJ14], a

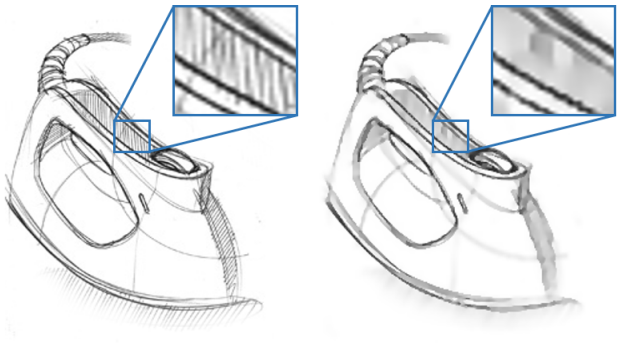
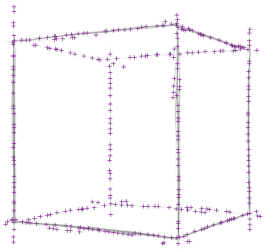


Figure 3: *The rolling guidance filter prevents sketch details like hatching (left) from contributing to the sketch's inferred shape by selectively blurring them out (right).*

filter that has been specifically designed to remove repetitive textures in images while preserving other details like contours. Figure 3 shows the effect of this filter on a sketch.

Contour thinning and sampling. The original FlowWeb algorithm is designed to compute dense correspondence fields between natural images. However, concept sketches are predominantly composed of sparse contours rather than dense shading and texture areas. We thus designed our algorithm to build sparse correspondences between point samples distributed along the *contours* of the drawing. Since contours can have varying thickness and sketchyness, we first locate their centerline by applying a small blur followed by non-maximum suppression and hysteresis thresholding. This filtering is similar in spirit to the Canny edge detector, except that we process the image intensity rather than its gradient since the contours in the drawing already represent the edges we want to thin.



We then distribute candidate point matches over the sketch by sampling the thinned contours. Our implementation generates point samples using a uniform grid of cell size 10×10 pixels. For each cell, we first try to find contour junctions by detecting Harris corners [HS88]. For cells with no such junctions, a sample is positioned on the edge pixel closest to the center (if any). On the left, we show the resulting sampling on a typical sketch. This subsampling increases processing speed by allowing the later use of a warping mesh that is coarser than the image resolution. Nevertheless, alternative warping approaches based on dense regularization [SMW06, NNRS15] could alleviate the need for sub-sampling.

3.2. Initializing the matches

Given two sketches represented as sets of points distributed along the contours, our goal is now to find, for each point in one sketch, the most similar point in the other sketch. Similar to prior work [XWL*08, CCT*09], we use shape context [BMP02] as a local descriptor to measure similarity between points in different sketches. This descriptor encodes a log-polar histogram of the points around the point of interest. Shape contexts are intrinsically invariant to translations. They are easily made scale invariant by normalizing all radial distances by the mean distance between the n^2 point pairs in the shape. Moreover, the coarse histogram representation makes shape contexts robust against small geometric perturbations, occlusions, and outliers typical of concept sketches.

Since shape contexts are histogram-based distributions, the cost of matching two points is computed with a χ^2 test. In order to take local appearance into account, we augment the shape context cost with a patch appearance cost computed as the Gaussian weighted intensity difference between small (9×9 pixels) patches around the two points. Following Belongie et al. [BMP02], we linearly combine the two costs, and then use a bipartite matching algorithm known as the Hungarian algorithm [Mun57] to compute the best matching between the two point sets. Further, to avoid strong distortions, we only keep *spatially local* matches by pruning out matches where the source and target points are farther than $\lceil \min(h,w)/10 \rceil$ pixels from each other, where h and w are the height and width of the sketch respectively. While we also experimented with other descriptors (SIFT, SSD) and found them less reliable than shape context, many alternatives exist such as recent deep-learned features.

3.3. Computing and improving cycle consistency

Given an initial set of matches between all pairs of sketches, we build on the FlowWeb algorithm to improve the matches by encouraging cycle consistency.

Cycle-consistency score. We denote T_{ij}^{pq} the motion vector between point p in sketch $\mathcal{S}_i$ and its matching point q in sketch $\mathcal{S}_j$. The cycle-consistency score $\mathcal{C}(T_{ij}^{pq})$ denotes the number of other sketches $\mathcal{S}_k$ that confirm this match by transitivity, up to a small error tolerance ε

$$\tilde{\mathcal{C}}(T_{ij}^{pq}) = card\{\mathcal{S}_k \notin \{\mathcal{S}_i, \mathcal{S}_j\} \mid \|T_{ij}^{pq} - (T_{ik}^{pr} + T_{kj}^{rq})\| < \varepsilon, r \in \mathcal{S}_k\}. \quad (1)$$

We fix ε to $0.02 \times \max(h,w)$ pixels in our implementation.

Note that we only consider cycles formed by triplets of images. While higher degree cycles could be considered, 3-cycles have been shown sufficient while remaining tractable [ZJLYE15].

Cycle-consistency improvement. Once each match has been assigned a consistency score, the second step of the FlowWeb algorithm is to improve the matches by replacing them with alternatives that have a higher score. These alternatives are found by transitivity. Given a match T_{ij}^{pq}, the algorithm considers all other sketches $\mathcal{S}_k \notin \{\mathcal{S}_i, \mathcal{S}_j\}$ and any alternative matching point q' with motion vector $T_{ij}^{pq'} = T_{ik}^{pr} + T_{kj}^{rq'}$. The score of an alternative match is given by the number of sketches $\mathcal{S}_l \notin \{\mathcal{S}_i, \mathcal{S}_j, \mathcal{S}_k\}$ that confirm both segments T_{ik}^{pr} and $T_{kj}^{rq'}$. If the alternative match obtains a higher score then T_{ij}^{pq} is replaced by $T_{ij}^{pq'}$.

3.4. Propagation with shape-preserving warp

The last step of the algorithm is to propagate information from the high-quality matches to their low-quality neighbors, where we identify high-quality matches as the ones with a consistency score greater than 2. The original FlowWeb algorithm works on dense, pixel-wise correspondences and relies on a consistency-weighted Gaussian filter to propagate motion vectors in image space. However, such a filtering approach does not impose any regularization on the resulting motion field, which in our experience can result in strong distortions. We thus propose to perform spatial propagation using an iterative matching/warping scheme similar to the as-rigid-as-possible registration algorithm of Sýkora et al. [SDC09].

Consider a pair of sketches, we first embed the point samples of the first sketch into a triangular warping mesh. We then warp the mesh using the high quality matches as guidance, subject to a regularization term that seeks to preserve the shape of the mesh triangles. This warp aligns the first sketch to be closer to the second one. We finally use the warped sketch to update the Shape Context descriptor of each sample and to update the low-quality matches by running the Hungarian algorithm with these new descriptors, as described in Section 3.2. We repeat this process for 3 iterations.

Various strategies for shape preserving warps have been described in image morphing and image-based rendering [ZCHM09, LGJA09, SDC09]. We follow the formulation of [LGJA09], which minimizes an energy functional composed of two terms:

$$E = w_c E_c + w_s E_s. \tag{2}$$

The first term tries to satisfy sparse correspondence constraints, while the second term penalizes strong distortions. Note that while we rely on a triangle mesh to compute these two terms, alternative meshless methods [SMW06, NNRS15] could also be used to generate a dense regularized propagation.

Correspondence constraints. In our context, we set the correspondence constraints as the set of matches $\mathcal{P}$ that have a consistency score greater or equal to 2. For each such match $T_{ij}^{pq} \in \mathcal{P}$, we have a point p in sketch $\mathcal{S}_i$ and a corresponding point q in sketch $\mathcal{S}_j$. The warp should, therefore, satisfy $W_{ij}(p) = q$[†]. Let the triangle in which p is contained be formed of the vertices (u,v,w) and let $\alpha(p),\beta(p),\gamma(p)$ be the barycentric coordinates of p w.r.t the triangle. The least-squares energy term for the correspondence constraint is therefore

$$E_c(p) = \tilde{C}(T_{ij}^{pq}) \| (\alpha(p)u, \beta(p)v, \gamma(p)w) - q \|^2, \tag{3}$$

where we weight each match by its consistency score $\tilde{C}(T_{ij}^{pq})$. Points with cyclic consistency less than 2 are weighted by their combined shape context and intensity score normalized to $[0,1]$, ensuring that such matches have a lower level of influence. We sum this energy term over all matches $T_{ij}^{pq} \in \mathcal{P}_i$.

Triangle shape constraints. Consider a mesh triangle $t = (u,v,w)$ and attach a local orthogonal frame to it: $\{v-u, R_{90}(v-u)\}$, where R_{90} is a counterclockwise rotation by 90 degrees. Assume that u is the origin of the frame. Now, in the frame, v is simply $(1,0)$ and let $w = (a,b)$. To preserve the shape of this triangle, we need to ensure that the transformation it goes through is as close as possible to a similarity transformation. Thus, we try to ensure that the local frame remains orthogonal and the coordinates of the vertices remain the same. The energy to express this constraint, with (u',v',w') denoting the warped coordinates of the triangle, is

$$E_s(t) = \| w' - (u' + a(v'-u') + b(R_{90}(v'-u'))) \|^2. \tag{4}$$

We sum this energy term over all triangles of the mesh.

Energy minimization. All the energy terms are quadratic, and the system of equations is overdetermined. This results in a standard least-squares problem which we solve with QR decomposition. In our experiments, we set $w_c = 5$ and $w_s = 6$.

3.5. User guidance

The algorithm described above converges to high-quality matches for simple, similar sketches. However, real-world sketches contain many variations of shape, viewpoint and line distribution which can disturb automatic matching. We improve accuracy by injecting a few user-provided correspondences in the algorithm, typically 3 to 5 annotations per image. The user only needs to provide correspondences between the most representative sketch and other sketches. We use these to automatically assign cycle-consistent correspondences to all other sketch pairs using transitivity.

We keep the user-provided correspondences fixed during all steps of the algorithm, and assign them the highest possible consistency score of $N-2$ (N is the number of input sketches) to ensure that they impact other matches during cycle-consistency improvement and spatial propagation.

4. Blending Concept Sketches

Once all sketches are registered, we interpolate them using warping and blending. While applications involving images with similar appearance and topology, such as image-based rendering [CDSHD13] and natural image morphing [LLN*14] are served reasonably well by uniform alpha-blending, blending concept sketches from disparate sources proves more challenging as misaligned strokes produce severe ghosting.

We first describe how we compute a generalized warp function from one sketch to a combination of other sketches. We follow up with a description of our blending method, and the user interface to control it. All these computations are performed in real-time.

4.1. Generalized warping function

Consider the N sketches input to the system, $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N$. The registration algorithm generates a family of pairwise warp functions $\mathbf{W} = \{W_{ij}\}, 1 \le i,j \le N$, such that applying the function W^{ij} on a pixel $p \in \mathcal{S}_i$ moves it towards its matched position in $\mathcal{S}_j$.

[†] $W_{ij}(p)$ and T_{ij}^{pq} both represent the motion vector at point p – while $W_{ij}(p)$ encodes the motion of sketch $\mathcal{S}_i$ towards $\mathcal{S}_j$ for all pixels, T_{ij}^{pq} encodes the motion of the sparse point set.

To move sketch $\mathcal{S}_i$ in-between multiple sketches $\mathcal{S}_{j=1..N}$, we follow [LWS98] that computes a generalized warp function as a linear combination of the pairwise warps

$$\widehat{W}_i(p) = \sum_{j=1}^{N} \mathbf{c}_j W_{ij}(p),$$

where the contribution weights $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N$ satisfy $\mathbf{c}_i \in [0,1]$, and $\sum \mathbf{c}_i = 1$. The interpolated sketch is given by the application of $\widehat{W}_i$ on $\mathcal{S}_i$, denoted as $\widehat{W}_i \bullet \mathcal{S}_i$. We compute the generalized warp function for all the sketches in real time so that users can interactively explore the interpolation space and create arbitrary combinations of sketches.

4.2. Consistency weighted non-linear alpha blending

Analogous to the warping weights $\mathbf{c}_i$ which determine the shape of the interpolated sketch, one can define blending weights $\alpha_1, \alpha_2, \ldots, \alpha_N$ to combine the color information of the warped sketches. The appearance of the resulting interpolated image $\mathcal{S}$ is determined as

$$\mathcal{S} = \sum_{i=1}^{N} \alpha_i (\widehat{W}_i \bullet \mathcal{S}_i).$$

Existing work on image blending often choose α_i to be the same as $\mathbf{c}_i$. In order to reduce the ghosting caused by this simple function, we modify it to allow for non-uniform blending across the image space, and to vary α_i as a non-linear function of $\mathbf{c}_i$. The idea is to follow linear alpha-blending for sketch contours that have good matches, but to quickly suppress contours that have poor matches as the contribution weight of their parent sketch decreases.

We utilize the combination of two sigmoid-like functions to achieve this. We first define the matching confidence of pixels sampled in Section 3.1 using their cyclic consistency.

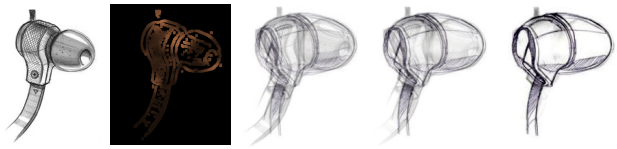
$$\texttt{conf}(p \in \mathcal{S}_i) = \frac{1}{N-2} \sum_{j=1}^{N} \mathbf{c}_j \times \tilde{\mathcal{C}}(T_{ij}^{pq}),$$

where p is matched to q in $\mathcal{S}_j$, and $1/(N-2)$ is the normalization factor. This confidence score is then propagated to all other pixels via linear interpolation (see Figure 4a). Our blending function for a pixel p in $\mathcal{S}_i$ is defined as

$$\alpha_i(p, \mathbf{c}_i, k) = \begin{cases} \frac{m_1}{1+\exp(-\mathtt{a}(p,k)\times(\mathbf{c}_i - 2/3))} + n_1 & \text{if } i = \text{argmax}_j(\mathbf{c}_j) \\ \frac{m_2}{1+\exp(-\mathtt{a}(p,k)\times(\mathbf{c}_i - 2/3))} + n_2 & \text{otherwise,} \end{cases}$$

where $\mathtt{a}(p,k) = (\texttt{conf}(p))^{-k}$. The values m_1, n_1, m_2, and n_2 are fixed such that $\alpha_i(p,0,k) = 0$, $\alpha_i(p,1,k) = 1$, and both the cases of the function evaluate to $2/3$ at $\mathbf{c}_i = 2/3$. The non-linearity of the blending function results in images with the highest contribution $\mathbf{c}_i = \max_j(\mathbf{c}_j)$ contributing more strongly to the final image appearance as compared to standard alpha blending. In addition, the non-uniformity of the function in image space ensures that well-matched regions smoothly transition into other images, while regions with poor matching "pop" in and out of existence based on which image has a high contribution.

The parameter $k \in [0,5]$ is controlled by using a slider in our interface. This allows the user to choose between mimicking alpha blending at one extreme ($k = 0$), and drawing contours predominantly from the image contributing the most to the interpolated shape at the other extreme ($k = 5$). The resulting difference in the interpolated image can be observed in Figure 4b. Finally, we normalize the contrast of the interpolated image to avoid contrast oscillation during interactive navigation.



(a) $\mathcal{S}$ *and* $conf(\mathcal{S})$ **(b)** *Varying the blending parameter k*

Figure 4: *Visualizing the spatial distribution of matching confidence over a sketch* ***(a)****, with brighter regions depicting higher matching confidence; and an example to show the impact on blending as the parameter k increases from left to right* ***(b)****. Notice that the poorly matched regions such as those with texture but no contours disappear first.*

4.3. User interface

We present the user with a planar embedding of the sketches, which represents a 2D interpolation space. The embedding is computed using metric multi-dimensional scaling [CC00] on average spatial distance between pairwise sketch correspondences. The average distance between any two sketches $\mathcal{S}_i$ and $\mathcal{S}_j$ is given by

$$d_{ij} = \frac{1}{2}\left(\frac{\sum_{p\in\mathcal{S}_i} \|W_{ij}(p) - p\|_2}{|\mathcal{S}_i|} + \frac{\sum_{q\in\mathcal{S}_j} \|W_{ji}(q) - q\|_2}{|\mathcal{S}_j|}\right).$$

The embedding places similar sketches close to each other on the plane. Users can also invert the embedding about either axis, or switch the two axes with each other if they feel those variants represent a more natural orientation. When the sketches are very similar in shape but differ in viewpoint, the arrangement gives an approximation of the 3D camera positions of the sketch viewpoints, as illustrated in Figure 11 (cars, planes). Exploration of the design space then gives the impression of 3D-like rotations around the concept, as shown in the accompanying video.

For rendering, we compute a Delaunay triangulation of the embedding. The contribution of each sketch is determined by the triangle under the user's mouse pointer. The sketches corresponding to the three vertices of this triangle contribute according to the barycentric coordinates of the point, while all other sketches have zero contribution. We also provide an alternate interface in which a user can simply choose the contributions of the sketches by manipulating sliders, one for each sketch. This interface also allows the user to turn off the blending equation, and use color information from a single sketch of her choice. While the interpolation space helps understand the relationship between various sketches, the slider-based interface gives users full control on which sketches they want to combine. Using either interface, a user can save the current interpolation at any time and oversketch it using traditional or digital methods to initiate the next design iteration with an augmented design space.

5. Evaluation

We now compare our algorithmic components with alternative solutions designed for other types of images than sketches. We provide animated comparisons of morphing sequences as supplemental material.

Figure 5 shows the matches produced by the original FlowWeb algorithm and by our adaptation. Since the original FlowWeb was designed to build dense correspondences between natural images, it also attempts to match points in the empty areas of the drawings. For the sake of visualization, we only show the matches where both points lie on a stroke, while we remove all other matches, which contain one or more point in an empty area. This visualization reveals that our algorithm obtains more stroke-to-stroke matches, and of higher quality than the matches found by the original algorithm.

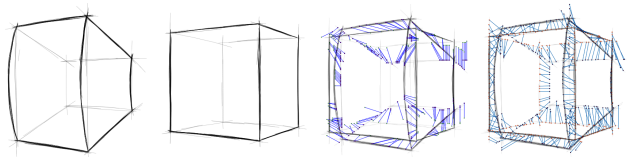


Figure 5: *Matches obtained by the original FlowWeb algorithm, compared to our adaptation for sketches. Notice the improvement in quality as well as quantity of stroke-to-stroke matches.*

Figure 6 compares our approach with our implementation of the registration algorithm of Sýkora et al. [SDC09]. While similar in spirit, our algorithm is customized for design sketches by using the ShapeContext descriptor, by sampling feature points along strokes rather than over the entire image, and by using FlowWeb for cycle consistency between multiple images. These different ingredients yield a better alignment of the design sketches overall. We also show the effect of applying a dense registration algorithm [GKT*08] after aligning the sketches. This post-process was used with success by Sýkora et al. to refine the registration of cartoon images that mostly differed in pose but shared similar details. In contrast, the design sketches we target often contain very different pen strokes and as such cannot be aligned with sub-pixel accuracy. The resulting local distortions are especially noticeable on animated sequences since this post-process is applied on a per-frame basis, which produces temporal artifacts (please refer to the accompanying video). In cases where sketches have minor shape differences, this post-process can be helpful (Figure 7), but still leads to loss of interactivity and temporal artifacts (see accompanying video). While their method does not explicitly support user guidance, we augmented it to take advantage of user specified correspondences, if available. Similar to our own algorithm, we enforce these correspondences across all iterations.

Figure 6 also provides a comparison of our method with the recent halfway-domain morphing algorithm [LLN*14] using the same user-provided correspondences. Similarly to the original FlowWeb algorithm, the halfway-domain morphing builds dense correspondences between the images, which often yields erroneous matches between stroke pixels and empty areas, as also illustrated in the accompanying video. In contrast, our method ensures that stroke samples match to other stroke samples, up to the shape-preserving regularization.

Finally, all the above algorithms have been developed for matching and morphing pairs of images, and it is unclear how to generalize them to multiple images.

6. Results

We have applied our approach on a number of real-world design sketches. Figure 11 shows the planar embeddings we obtain for several sets of sketches, as well as some of the interpolations generated by our algorithm during interactive exploration. The accompanying video shows how the shape and view transitions produced by our morphing provide a vivid sense of continuous navigation within the design space. In the field of data visualization, such animated transitions have been shown to improve graphical perception of inter-related data as they convey transformations of objects, cause-and-effect relationships and are emotionally engaging [HR07].

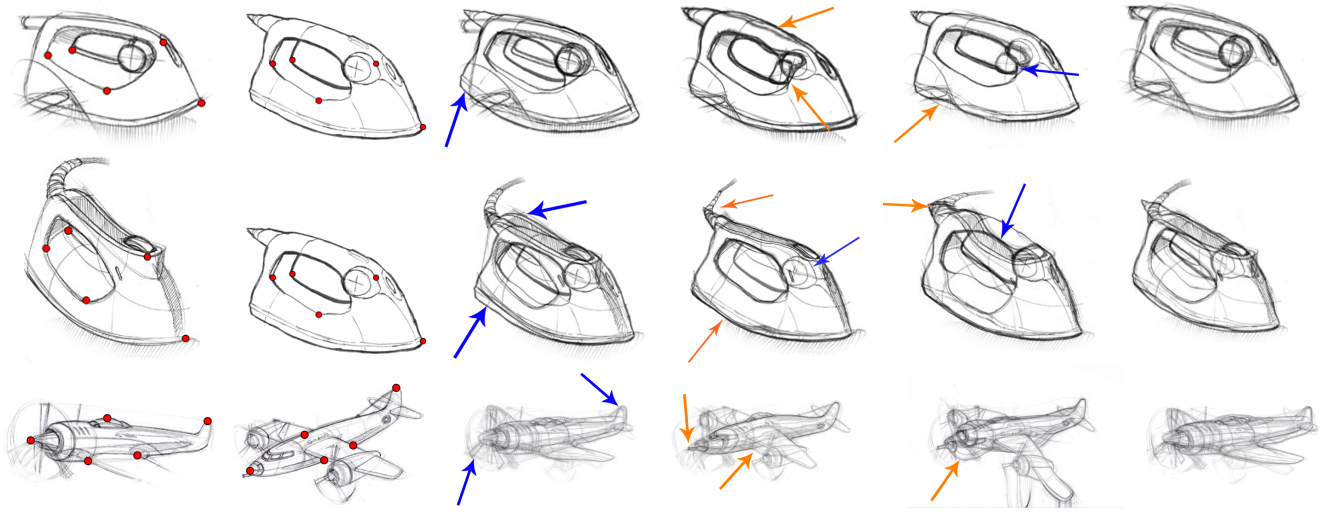
Figures 8 and 9 illustrate two application scenarios where our interpolated sketches support the creation of new concepts. In Figure 8, a designer aligns two sketches of airplanes using our tool before selectively erasing parts of each sketch to create a new mix-and-match airplane with more propellers. In Figure 9, a designer uses interpolated sketches as inspirational underlays to draw new shapes. Once created, these new drawings can be injected back in our algorithm to expand the design space for iterative design exploration. The accompanying video shows how a design student used this feature to design an iron. While informal, this user test confirmed that blended sketches form an effective guidance for design exploration.

At the core, our algorithm relies on good quality shape context matches for sketch alignment. In cases where the difference in shapes of the drawn concepts is too high or the viewpoints are very far apart, our algorithm fails to produce acceptable results, as shown in Figure 10. Providing manual correspondences, or adding more drawings to the dataset, can alleviate this limitation by relating very different sketches via more similar ones.

7. Conclusion

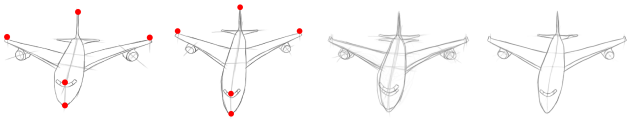
We have adapted and reformulated a number of 2D image processing techniques to enable matching, warping and blending design sketch input. These ideas can impact other problems in the space of sketch-based interfaces and modeling. In the future, we also plan to explore the use of non-rigid 3D reconstruction algorithms [VCAB14] to recover the 3D object depicted by multiple concept sketches along with the variations specific to each sketch.

While there is a rich body of work in exploring collections of well-defined 3D shapes, there are few tools to support the unstructured and messy space of design ideation. SketchSoup thus fills a void, empowering designers with the ability to continuously explore and refine sparsely sampled design sketch spaces in 2D itself, at the same level of detail and sophistication as the input sketches. By registering and morphing unstructured sketch collections, automatically or with minimal user interaction, SketchSoup further allows designers to present their design spaces better to others, via



(a) *Source image* **(b)** *Target image* **(c)** *Sýkora et al. [SDC09]* **(d)** *[SDC09] + [GKT*08]* **(e)** *Liao et al [LLN*14]* **(f)** *Ours*

Figure 6: *Comparison of our approach for sketch alignment* ***(f)*** *with an existing natural image alignment algorithm [LLN*14]* ***(e)****; and with a cartoon image alignment method [SDC09], the latter both with* ***(d)*** *and without* ***(c)*** *a dense alignment post-process. Red dots on the input images show user correspondences. Fine-level distortions and misalignments have been indicated using orange and blue arrows, respectively. Please refer to the accompanying video for animated versions of these comparisons.*



Source sketch *Target sketch* *Our result* *Post-processed*

Figure 7: *Applying a dense registration method [GKT*08] as a per-frame post-process on our method can improve alignment when the shape difference between sketch pairs is small.*

a layout that reflects design similarity and interpolated animations that can better mimic animated view transitions and convey part relationships between designs.

Acknowledgements

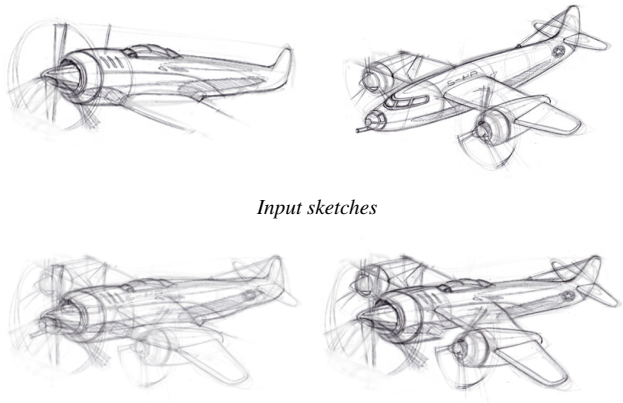
The authors thank Candice Lin for narrating the video. We also thank Mike Serafin and Spencer Nugent for allowing us to use their inspiring design sketches, and to Chris de Pauli for testing our system to create new sketches (shown in Figure 9). This work was partially supported by research and software donations from Adobe.

References

[AECOK16] AVERBUCH-ELOR H., COHEN-OR D., KOPF J.: Smooth image sequences for data-driven morphing. *Computer Graphics Forum (Proc. Eurographics) 35*, 2 (2016). 3

[BA06] BAXTER W., ANJYO K.-I.: Latent Doodle Space. *Computer Graphics Forum* (2006). 3

[BBA09] BAXTER W., BARLA P., ANJYO K.-I.: N-way morphing for 2D Animation. *Computer Animation and Virtual Worlds 20*, 2 (2009). 3



Input sketches

Alignment (left) and selective combination (right)

Figure 8: *Mixing parts of two sketches. Top: The two original sketches. Bottom: Their aligned versions, and novel sketch generated by selective erasing parts of each drawings.*

[BBS08] BAE S.-H., BALAKRISHNAN R., SINGH K.: ILoveSketch: As-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (Monterey, CA, USA, 2008), UIST '08, ACM, pp. 151–160. 2

[BC13] BONNICI A., CAMILLERI K.: A circle-based vectorization algorithm for drawings with shadows. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (Anaheim, California, 2013), SBIM '13, ACM, pp. 69–77. 2

[BMP02] BELONGIE S., MALIK J., PUZICHA J.: Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine*

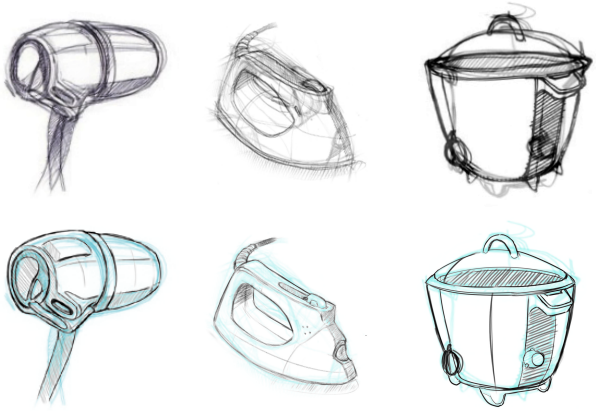


Figure 9: *Interpolations generated by our method (top) guide the creation of polished novel sketches (bottom, with faded blue guidance).*

Figure 10: *Limitation. Our algorithm fails to align sketches properly when the shape or view difference is too large.*

Intelligence, IEEE Transactions on 24, 4 (4 2002), 509–522. 4

[CC00] COX T. F., COX M. A.: *Multidimensional scaling*. CRC Press, 2000. 6

[CCT*09] CHEN T., CHENG M.-M., TAN P., SHAMIR A., HU S.-M.: Sketch2photo: Internet image montage. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 28*, 5 (2009). 4

[CDSHD13] CHAURASIA G., DUCHÊNE S., SORKINE-HORNUNG O., DRETTAKIS G.: Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics 32* (2013). 2, 3, 5

[CW93] CHEN S. E., WILLIAMS L.: View interpolation for image synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA, 1993), SIGGRAPH '93, ACM, pp. 279–288. 3

[DXS*07] DORSEY J., XU S., SMEDRESMAN G., RUSHMEIER H., MCMILLAN L.: The mental canvas: A tool for conceptual architectural design and analysis. In *The 15th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2007), IEEE Computer Society, IEEE Computer Society. 2

[ES11] EISSEN K., STEUR R.: *Sketching: The Basics*. BIS, 2011. 2

[GKT*08] GLOCKER B., KOMODAKIS N., TZIRITAS G., NAVAB N., PARAGIOS N.: Dense image registration through MRFs and efficient linear programming. *Medical image analysis 12*, 6 (2008), 731–741. 7, 8

[HR07] HEER J., ROBERTSON G.: Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov. 2007), 1240–1247. 2, 7

[HS88] HARRIS C., STEPHENS M.: A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference* (Manchester, UK, 1988), Organizing Committee AVC 88, pp. 147–151. 4

[LGJA09] LIU F., GLEICHER M., JIN H., AGARWALA A.: Content-preserving Warps for 3D Video Stabilization. *ACM Trans. Graph. 28*, 3 (7 2009), 44:1–44:9. 5

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 31–42. 3

[LLN*14] LIAO J., LIMA R. S., NEHAB D., HOPPE H., SANDER P. V., YU J.: Automating image morphing using structural similarity on a halfway domain. *ACM Trans. Graph. 33*, 5 (Sept. 2014), 168:1–168:12. 3, 5, 7, 8

[LWS98] LEE S., WOLBERG G., SHIN S. Y.: Polymorph: morphing among multiple images. *Computer Graphics and Applications, IEEE 18*, 1 (Jan 1998), 58–71. 6

[LZC11] LEE Y. J., ZITNICK C. L., COHEN M. F.: Shadowdraw: Real-time user guidance for freehand drawing. *ACM TOG (Proc. SIGGRAPH) 30*, 4 (July 2011), 27:1–27:10. 3

[MHM*09] MAHAJAN D., HUANG F.-C., MATUSIK W., RAMAMOORTHI R., BELHUMEUR P.: Moving gradients: A path-based method for plausible image interpolation. *ACM Trans. Graph. 28*, 3 (July 2009), 42:1–42:11. 3

[Mun57] MUNKRES J.: Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics 5*, 1 (1957), 32–38. 4

[NBCW*11] NGUYEN A., BEN-CHEN M., WELNICKA K., YE Y., GUIBAS L.: An optimization approach to improving collections of shape maps. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 1481–1491. 3

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: Designing freeform surfaces with 3d curves. *ACM Trans. Graph. 26*, 3 (7 2007). 2

[NNRS15] NGUYEN C. H., NALBACH O., RITSCHEL T., SEIDEL H.-P.: Guiding image manipulations using shape-appearance subspaces from co-alignment of image collections. *Computer Graphics Forum (Proc. Eurographics) 34*, 2 (2015). 4, 5

[OK11] ORBAY G., KARA L. B.: Beautification of design sketches using trainable stroke clustering and curve fitting. *IEEE Transactions on Visualization and Computer Graphics 17*, 5 (2011), 694–708. 2

[OSSJ09] OLSEN L., SAMAVATI F. F., SOUSA M. C., JORGE J. A.: Technical section: Sketch-based modeling: A survey. *Comput. Graph. 33*, 1 (Feb. 2009), 85–103. 2

[PKM*11] PACZKOWSKI P., KIM M. H., MORVAN Y., DORSEY J., RUSHMEIER H., O'SULLIVAN C.: Insitu: Sketching architectural designs in context. *ACM Transactions on Graphics 30*, 182 (12/2011 2011). 2

[RID10] RIVERS A., IGARASHI T., DURAND F.: 2.5d cartoon models. *ACM Trans. Graph. 29*, 4 (July 2010), 59:1–59:7. 2, 3

[SBSS12] SHAO C., BOUSSEAU A., SHEFFER A., SINGH K.: Crossshade: Shading concept sketches using cross-section curves. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012) 31*, 4 (2012). 2

[SD96] SEITZ S. M., DYER C. R.: View morphing. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 21–30. 3

[SDC09] SÝKORA D., DINGLIANA J., COLLINS S.: As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of International Symposium on Non-photorealistic Animation*

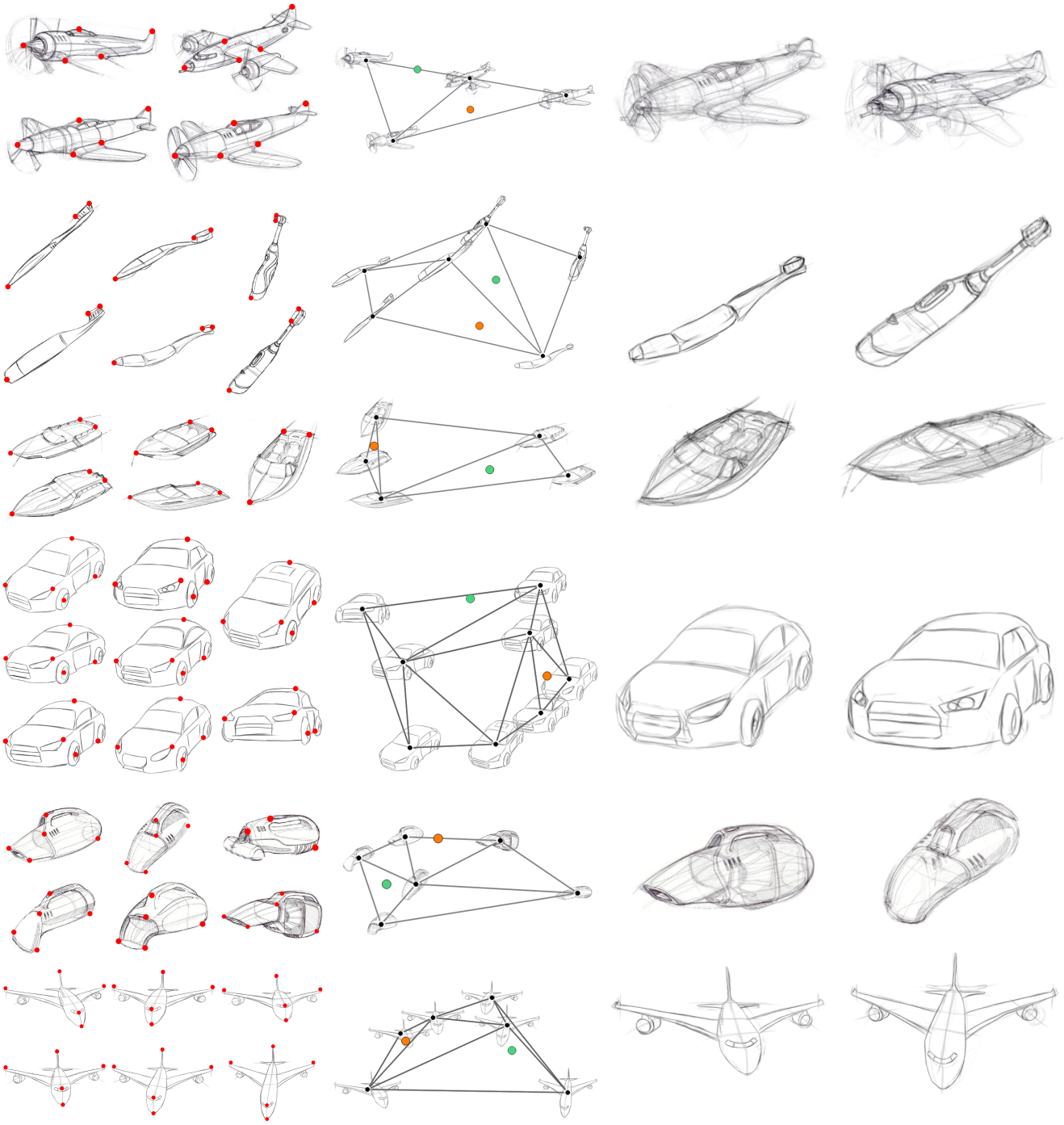


Figure 11: *2D interpolation spaces and selected interpolated sketches for different concepts. Some sketches by Spencer Nugent.*

and Rendering (New Orleans, Louisiana, USA, 2009), ACM, pp. 25–33. 3, 5, 7, 8

[SLZ*13] SHAO T., LI W., ZHOU K., XU W., GUO B., MITRA N. J.: Interpreting concept sketches. *ACM Trans. Graph. 32*, 4 (July 2013), 56:1–56:10. 3

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006). 4, 5

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*

(New York, NY, USA, 2006), ACM Press, pp. 835–846. 2, 3

[VCAB14] VICENTE S., CARREIRA J., AGAPITO L., BATISTA J.: Reconstructing pascal voc. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Columbus, Ohio, June 2014), IEEE Computer Society. 7

[WNS*10] WHITED B., NORIS G., SIMMONS M., SUMNER R., GROSS M., ROSSIGNAC J.: Betweenit: An interactive tool for tight inbetweening. *Comput. Graphics Forum (Proc. Eurographics) 29*, 2 (2010), 605–614. 3

[XCS*14] XU B., CHANG W., SHEFFER A., BOUSSEAU A., MCCRAE J., SINGH K.: True2form: 3d curve networks from 2d sketches via selective regularization. *ACM Transactions on Graphics (Proc. SIGGRAPH 2014) 33*, 4 (2014). 2

[XWL*08] XU X., WAN L., LIU X., WONG T.-T., WANG L., LEUNG C.-S.: Animating animal motion from still. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 27*, 5 (December 2008), 117:1–117:8. 3, 4

[XWSY15] XING J., WEI L.-Y., SHIRATORI T., YATANI K.: Autocomplete hand-drawn animations. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 169:1–169:11. 3

[ZCHM09] ZHANG G.-X., CHENG M.-M., HU S.-M., MARTIN R. R.: A shape-preserving approach to image resizing. *Computer Graphics Forum 28*, 7 (2009), 1897–1906. 5

[ZJLYE15] ZHOU T., JAE LEE Y., YU S. X., EFROS A. A.: Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, MA, USA, June 2015), IEEE Computer Society. 3, 4, 7

[ZKP10] ZACH C., KLOPSCHITZ M., POLLEFEYS M.: Disambiguating visual relations using loop constraints. In *Computer Vision and Pattern Recognition (CVPR)* (San Fransisco, CA, USA, June 2010), IEEE Computer Society, pp. 1426–1433. 3

[ZSXJ14] ZHANG Q., SHEN X., XU L., JIA J.: Rolling guidance filter. In *Computer Vision âĂŞ ECCV 2014*, Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), vol. 8691 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 815–830. 3