AttribIt: Content Creation with Semantic Attributes

Siddhartha Chaudhuri *
Princeton University
sidch@cs.princeton.edu

Evangelos Kalogerakis *
U. Mass. Amherst
kalo@cs.umass.edu

Stephen Giguere
U. Mass. Amherst
sgiguere@cs.umass.edu

Thomas Funkhouser
Princeton University
funk@cs.princeton.edu

ABSTRACT

We present ATTRIBIT, an approach for people to create visual content using relative semantic attributes expressed in linguistic terms. During an off-line processing step, ATTRIBIT learns semantic attributes for design components that reflect the high-level intent people may have for creating content in a domain (e.g., adjectives such as "dangerous," "scary," or "strong") and ranks them according to the strength of each learned attribute. Then, during an interactive design session, a person can explore different combinations of visual components using commands based on relative attributes (e.g. "make this part more dangerous"). Novel designs are assembled in real-time as the strength of selected attributes are varied, enabling rapid, in-situ exploration of candidate designs. We applied this approach to 3D modeling and web design. Experiments suggest this interface is an effective alternative for novices performing tasks with high-level design goals.

Author Keywords

content creation, exploratory interfaces, high-level attributes, relative attributes, semantic attributes, assembly-based modeling, interactive modeling

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation (e.g., HCI): User Interfaces - Interaction styles; I.2.10 Artificial Intelligence: Vision and Scene Understanding - Shape

General Terms

Algorithms; Design; Human Factors.

INTRODUCTION

The ubiquity of visual media, and the growing interest in new applications such as collaborative virtual worlds and 3D printing, has led to a demand for interfaces suitable for novice users to engage in computer-aided visual design to produce their own content. This is especially true for three-dimensional content, which is notoriously difficult for novices to create [33]. A successful interface must make it as easy as possible for novice users to map a design goal to intuitive interactions so that they quickly reach the objective and feel confident that it has been achieved.

Figure 1: *A user explores novel virtual creatures by changing the strength of semantic attributes reflecting high-level design intent. The attributes, and their relative strengths for different parts, are learned from crowdsourced training data.*

Several types of state-of-the-art interfaces, such as drawing and sculpting tools [15,27], drag-and-drop component assembly [6,24] and design galleries [22,23,35] have been proposed to address this need. However, such interfaces have a common limitation: they do not provide a way to directly express a high-level design goal (e.g. "create a cute toy"). Instead, the goal must be reached by careful planning and execution of a series of low-level selection and editing commands — which requires previsualization, dexterity and time — or serendipitiously through largely unstructured exploration. The gap between how a person thinks about what she wants to create, and how she can interact with a computer to get there, is a barrier for the novice.

This paper introduces ATTRIBIT, an approach for quickly realizing high-level design goals by directly capturing design intent in *linguistic* terms. For example, the designer may want

*S. C. and E. K. contributed equally to this work.

Figure 2: *Exploration with semantic attributes for 3D modeling. Top: Exploring less (left) to more (right) aerodynamic fuselages. Bottom: Exploring less (left) to more (right) scary heads. The adjectives "aerodynamic" and "scary," and the ranking of components under these attributes were learned from crowdsourced training data. The displayed components, left to right, have ∼0%, 33%, 66% and 100% of the respective attribute strengths. All components are automatically positioned and glued.*

to make a virtual creature more "scary", or a web page more "artistic". Such requirements are natural for humans, yet cannot be directly expressed in current interfaces. Nor are they represented in the pioneering work of Coyne and Sproat [8] which focuses, by contrast, on configurational specifications ("the cat is facing the wall"). We seek a human-centric interface that lets us specify such objectives directly, without having to translate to computer-friendly abstractions.

ATTRIBIT is a data-driven interface for content creation that associates *relative semantic attributes* with design elements. A relative attribute [29] is one whose strength varies continuously (e.g. "strong"), as opposed to binary attributes, which are either present or absent (e.g. "quadruped"). ATTRIBIT directly supports actions such as "show me scarier heads" or "show me more artistic webpages". Of course, it is infeasible to capture all possible design objectives as individual attributes. However, complex concepts can usually be expressed in terms of simpler ones [25] (e.g. an airplane designed to win dogfights is typically "aerodynamic", "fast" and "military"). Hence, ATTRIBIT supports a wide range of design objectives beyond those corresponding to individual supported attributes.

Further, ATTRIBIT is an exploratory interface, supporting the notion of design as informed choice [3, 14]. It facilitates *targeted exploration* of the design space in terms of human-friendly, semantic axes of variation. ATTRIBIT uses domain knowledge learned from a database of exemplar designs to automatically assemble coherent designs from selected components. The designer can rapidly iterate over a large number of relevant candidate designs to find a suitable one, rather than spend time in painstaking geometric manipulation tasks to assemble each design. Instead of trying to visualize the result of assembling components, she can simply evaluate the finished design in-situ.

A key challenge in enabling this type of interaction is to develop automatic or semi-automatic techniques to infer high-level semantic attributes for design components. A second key challenge is to develop interfaces that allow targeted exploration of the design space based on these attributes.

To infer semantic attributes, we first crowdsource descriptive adjectives for design components from Amazon Mechanical Turk. Since real-world applications will have too many exemplar components for complete manual annotation, we leverage techniques from information retrieval, machine learning and computer vision to learn functions that map component appearance to attribute strengths, e.g. how scary an animal head is, or how artistic a web page title appears. The functions are learned from a training set of pairwise comparisons also gathered via Mechanical Turk. Finally, we introduce a statistical model for multiple components to be changed simultaneously and coherently as attribute strength is varied.

To enable exploration with high-level attributes, we introduce an interface that allows users to initialize a design with a coherent combination of components from a database, select a subset of these components, interactively increase or decrease the strength of an attribute using sliders, and observe changes to the whole design in real time as new database components corresponding to the updated attribute strengths are swapped in. The components are automatically assembled into a coherent design.

In this paper, we discuss ATTRIBIT chiefly in the context of modeling three-dimensional shapes: virtual creatures, aircraft and ships. To demonstrate that the method generalizes to other (non-3D) domains, we also describe an application to web page design. We present results of experiments with novice users, showing that the interface is easily mastered (in 10 minutes, by an 11-year-old) and can be used to rapidly construct compelling visual content. Finally, we discuss possible future applications of high-level attributes for design.

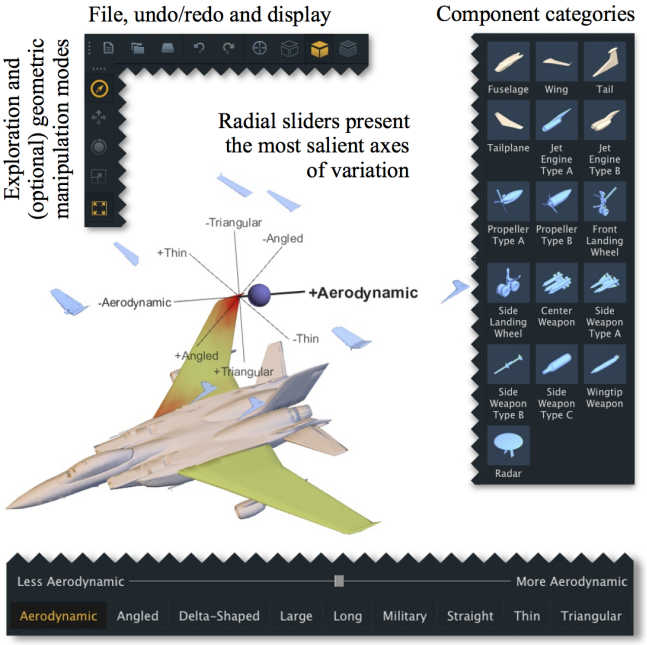


Figure 3: *Modeling interface. Sliders at the bottom and in a radial menu allow exploring novel designs by increasing (+) or decreasing (−) the strength of semantic attributes. Each radial axis approximates the direction of corresponding geometric variation: aerodynamic, angled and triangular wings are stubby and swept back, those with opposite properties are long and perpendicular to the fuselage. Here the user drags the violet handle towards "+Aerodynamic" to explore more aerodynamic wings. Red highlights indicate regions of high geometric variation under the current attribute — the tips are most salient because aerodynamic wings are swept back.*

INTERFACE

ATTRIBIT employs a database of exemplar designs, assumed to be compatibly segmented and labeled. E.g. creatures might be decomposed into head, tail, torso and legs, and airplanes into wings, fuselage, tailplane and engines. Novel designs are created by recombining components from different exemplars [6, 11, 17]. Repurposing successful elements from prior designs can be more efficient than reinventing them from scratch [13]. The space of possible designs is exponentially large: in comparison, the subset fulfilling a given design objective is infinitesimally small. The goal of ATTRIBIT is to facilitate access to this subset of suitable designs.

When the interface is first launched, the artist is presented with buttons on the right corresponding to categories of components from the domain (e.g., heads, torsos, legs, tails for animals; fuselages, wings, tails, engines for airplanes), and a modeling area in the center (Figures 1, 3). She can click buttons in any sequence to add initial components from the corresponding categories to the modeling area. Each initial component is chosen so that the assembled design is plausible, according to a statistical shape prior [17]. They are automatically positioned and stitched into a coherent, seamless design. Components are selected by clicking; selected components are removed with the Delete key. Components may be added and removed at any time, in any sequence. To allow fine adjustment in positioning if desired, our interface also provides translation, rotation and scaling controls.

Exploration Widgets

To explore semantic axes of variations, the user selects one or more components. Upon selection, exploration widgets appear on the screen. The first such widget is a tabbed panel of sliders at the bottom of the screen. Each tab is labeled with the name of an attribute that applies to at least one of the selected components (e.g. "Aerodynamic", "Dangerous", "Fast"). As the user drags a slider up (or down) the range, new components corresponding to higher (resp. lower) strengths of the attribute replace the current ones (Figure 2). Replacement, repositioning and re-stitching of assembled components happens automatically and in real time (Appendix).

The second widget is a set of sliders displayed as spokes of a wheel, positioned directly on the most recently selected component. The spokes have equal angular spacing ($45°$). Each pair of opposed spokes maps to increasing and decreasing directions of an attribute slider. For instance, the wheel might display "+Fast" and "−Fast" sliders along opposed spokes, corresponding to increasing or decreasing the strength of the "Fast" attribute respectively. One clicks and drags the central handle to explore replacements along any axis, or return to the center to pick a new one.

The wheel allows designers to efficiently evaluate the utility of different attributes with a single click (and subsequent dragging), instead of the click-heavy process of selecting and interacting with individual slider tabs. It displays the 4 attributes with largest geometric variation at the point c at the wheel center (the attribute of the current tab is always included for consistency). To estimate this variation for an attribute, we pre-align all parts in its ordered sequence to the current part using non-rigid ICP [1]. This lets us track c through the preceding and succeeding parts in the sequence of components ordered according to their attribute strength. The local geometric variation for the attribute is computed as the average displacement of the point through successive replacements, limited to K preceding and K succeeding parts in the sequence (we use $K = 3$). The length of this vector ("attribute saliency") determines inclusion in the wheel, and its screen-space direction determines the preferred orientation of the corresponding pair of opposed spokes (Figure 3).

The wheel can be repositioned by clicking at a different point on the selected part, whereupon the most salient attributes are recomputed at that point. We visualize regions of high geometric variation for the active attribute as a saliency map, highlighting more salient regions in red. This visualization provides information on how *semantic* variation manifests as *geometric* variation over a shape (Figures 1, 3). For instance, fast airplane wings have their tips swept back, so related attributes have high saliency at wingtips. The wheel is initially positioned at the most salient point.

The accompanying video shows several example interaction sessions with the ATTRIBIT interface.

Figure 4: *Ordering of animal heads according to the "dangerous" attribute (top-left is the least dangerous head, sequence is in row-major order). The orderings are organized on sliders in the* ATTRIBIT *interface (Figure 1).*

LEARNING ATTRIBUTES

Recent computer vision research has shown that low-level image features can be mapped to "visual attributes" that correspond to properties observable in images and have human-designated names (e.g., smiling, natural, man-made) [2, 9, 10, 28, 31]. These visual attributes can serve as semantic cues in various image analysis problems. Our work is particularly inspired by the idea of "relative attributes" introduced by Parikh and Grauman [29].

ATTRIBIT uses relative semantic attributes to explore the design space. To achieve this, it learns mappings from features of design components to attribute strengths, based on crowdsourced training data. For 3D shapes, the features include histograms of curvature and shape diameter, and view-based descriptors. We compute non-linear transformations of these descriptors as additional higher-level features. For details, please refer to the Appendix. By applying these learned mappings from features to attributes, ATTRIBIT automatically ranks components by attribute strength.

Attribute Selection

Our first task is to find adjectives describing attributes of each category of design component. Rather than picking attributes ourselves, we resorted to crowdsourcing to quickly find such adjectives. For each component category (e.g. "head", "fuselage"), we created web pages showing images of two randomly selected designs with highlighted components. For each such pair, we asked respondents in Amazon Mechanical Turk to complete one of the following statements with a suitable adjective (Type 1 questions):

- The left part is more ____ than the right part.
- The left part is ____ than the right part.
- The right part is more ____ than the left part.
- The right part is ____ than the left part.

We also asked about the role of the parts in influencing attributes of the whole design (Type 2 questions):

- The left part makes the shape look more ____ than the right part.
- The left part makes the shape look ____ than the right part.
- The right part makes the shape look more ____ than the left part.
- The right part makes the shape look ____ than the left part.

To obtain a consistent set of salient adjectives across a large sample of respondents, we followed Schelling [32]. Respondents were asked to suggest adjectives they thought *other* people would also select. In addition, to reduce spurious entries, we disallowed the adjectives "better," "worse," "brighter," and "darker." Our survey collected a total of 21580 comparative adjectives for 32 component categories. Then, we merged synonyms and corrected typos manually. For each component category, we selected the top 10 most common adjectives, omitting those selected < 5 times. For example, the top adjectives for torsos were: *large*, *long*, *thin*, *strong*, *rounded*, *heavy*, *dangerous*, *smooth*, *curved* and *muscular*.

Pairwise Comparison

After finding suitable attributes, we compute complete rankings of all components of each category for each attribute. Asking humans to manually rank thousands of components would be impractical. However, it is relatively easy for a person to compare two objects [29]. Hence, we collect a few pairwise comparisons as training data and automatically infer full rankings. We again created web pages showing pairs of designs with highlighted components. If an attribute was associated with Type 1 questions during attribute selection, we asked Mechanical Turk users to indicate which component in each pair has more of the attribute:

- The left part is more [adjective] than the right part.
- The left part is less [adjective] than the right part.
- The left part is as [adjective] as the right part.
- Neither part is [adjective].

If an attribute was associated with Type 2 questions, we asked which component contributes more to the global attribute:

- The left part makes the shape look more [adjective] than the right part.
- The left part makes the shape look less [adjective] than the right part.
- The left part makes the shape look as [adjective] as the right part.
- Neither part makes the shape look [adjective].

We posted at most 100 pairs for each category and for each adjective, for a total of 87650 comparisons. Each respondent compared 25 pairs and repeated the process for consistency. We rejected respondents whose answers for more than a third of the pairs differed across repetitions. Three different people evaluated each pair, yielding a per-pair confidence measure.

Learning to Rank Components

Given training pairs, we learn a ranking function that infers the strength of an attribute for each associated component. The function maps features of a component to its ranking "score," and thus generalizes to components not in the training comparisons.

For each component category and each associated adjective m, the input to the learning step is a set of training pairs $O_m = \{i, j\}$ in which component i has more of an attribute than j, and a set of training pairs $S_m = \{i, j\}$ in which components i and j have comparable attribute strengths. For each training pair and relative ordering thereof, we also have a confidence measure $c_{ij} \in [0, 1]$ defined as the fraction of respondents who agreed on this ordering. Each component i is represented in $\mathbb{R}^n$ by a feature vector $\mathbf{x}_i$ (for more details on the features used, please refer to the Appendix).

The ranking function r_m maps features to attribute strength:

$$r_m(\mathbf{x}) = \mathbf{w}_m \cdot \mathbf{x},$$

where $\mathbf{w}_m$ is a parameter vector, s.t. the maximum number of dissimilarity and similarity constraints is satisfied:

$$\begin{aligned} \forall (i,j) \in O_m : \ & \mathbf{w}_m \cdot \mathbf{x}_i > \mathbf{w}_m \cdot \mathbf{x}_j \\ \forall (i,j) \in S_m : \ & \mathbf{w}_m \cdot \mathbf{x}_i = \mathbf{w}_m \cdot \mathbf{x}_j \end{aligned}$$

Learning the optimal ranking function is NP-hard [16]. However, we can introduce slack variables to relax the problem, as in Support Vector Machine (SVM) classification [29].

$$\begin{aligned} \text{minimize} \ \ & \frac{1}{2}\|\mathbf{w}_m\|_2^2 + \alpha \sum_{i,j} (\mathcal{L}(\xi_{ij}) + \mathcal{L}(\gamma_{ij})) \\ \text{s.t.} \quad \mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j) \ & \geq \ 1 - \xi_{ij}, \quad \forall (i,j) \in O_m \\ |\mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j)| \ & \leq \ \gamma_{ij}, \quad \forall (i,j) \in S_m \\ \xi_{ij} \geq 0 \quad & \text{and} \quad \gamma_{ij} \geq 0 \end{aligned}$$

where ξ_{ij} , γ_{ij} are slack variables, $\mathcal{L}(\xi_{ij})$, $\mathcal{L}(\gamma_{ij})$ are loss functions to progressively penalize misranked training examples, and α weights the approximated error. Various loss functions can be employed — for example, in [29] quadratic loss functions $\mathcal{L}(\xi_{ij}) = \xi_{ij}^2$, $\mathcal{L}(\gamma_{ij}) = \gamma_{ij}^2$ were used. This problem is equivalent to solving the following unconstrained optimization problem [5]:

$$\begin{aligned} \text{minimize} \ \ \|\mathbf{w}_m\|_2^2 + \alpha & \sum_{i,j \in O_m} \ell(1 - \mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j)) \\ & + \alpha \sum_{i,j \in S_m} \ell(|\mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j)|) \end{aligned}$$

where the first term serves as a regularization term favoring sparse parameter vectors, and $\ell(\cdot)$ is the loss function penalizing misranked examples, e.g. quadratic loss: $\ell(t) = \max(0, t)^2$, or hinge loss: $\ell(t) = \max(0, t)$. In practice, we found the sigmoid loss function [4, 30] performed slightly better (see *Ranking Quality* section). We also consider the confidence c_{ij} for each pair, and weight the loss functions for similarity and dissimilarity constraints differently. Thus, the optimization problem we solve is the following:

$$\begin{aligned} \text{minimize} \ \ \|\mathbf{w}_m\|_2^2 + \ \mu & \sum_{i,j \in O_m} c_{ij}(1 - \sigma(\mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j))) \\ & + \nu \sum_{i,j \in S_m} c_{ij}\sigma(|\mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j)|) \end{aligned}$$

where μ, ν are weighting coefficients and $\sigma(t) = 1/(1 + \exp(-\lambda t))$ is the sigmoid function. The sigmoid loss function is not convex, so we start optimizing the parameters $\mathbf{w}_m$ with a quadratic loss function that is convex and can be minimized with Newton's method [5]. Then, we refine the parameters with a subgradient method for the objective function using the sigmoid loss, which practically converges as also shown in [12]. The weights λ, μ, ν are estimated using 5-fold cross-validation to minimize the number of wrong pairwise rankings in the validation sets.

Using Attributes as Additional Features for Learning

Some attributes, such as "strong", can depend on other simpler attributes, such as "large," "heavy," "thin," or other higher-level attributes, such as "dangerous". ATTRIBIT takes into account these inter-dependencies between attributes to learn them more effectively. Specifically, we first learn a model $r_m(\cdot)$ for each attribute using the base features. Then, we learn a new model $r'_m(\cdot)$ by enriching the feature vector with the scores of all other attributes inferred from the first trained models. We repeat the procedure to learn another model $r''_m(\cdot)$ using the attribute scores from the second trained models. This procedure further decreased test errors (see *Ranking Quality* section). Subsequent repetitions did not decrease errors significantly. Interestingly, many natural inter-dependencies of attributes are successfully captured in this way. E.g. the attribute scores for "strong" heads are strongly correlated with "larger," "heavier," more "dangerous" and less "thin" heads.

MULTIPLE COMPONENTS AND DESIGN PRIORS

ATTRIBIT also supports exploring variation in multiple components simultaneously. When more than one component is selected, some will possess the active high-level attribute and some will not. For example, the attribute "scary" exists for animal heads, but not for legs. We define a joint ranking score for multiple components that share the same active attribute, and infer plausible replacements for components that do not have the attribute.

Joint Ranking of Multiple Components

When multiple components $P = \{p_1, p_2, \ldots, p_n\}$, from categories $q_1, q_2, \ldots, q_n$ respectively, share a single high-level attribute m, we define a joint ranking score over them. First, the ranking scores r''^{q}_m for the components in each associated category q are normalized to $[0, 1]$. Let $R^{q_i}_m(p_i)$ the normalized scores for each component p_i in their associated category q_i. The combined score is defined as the sum

$R_m(P) = \sum_{i=1}^{n} R_m^{q_i}(p_i)$. Starting from the current configuration P, we greedily find the single component replacement that will improve the ranking score, defined above, by the smallest increment. We continue this process until no further replacements are possible — the resulting configuration has the maximum attribute strength under our definition. Similarly, we extend the ranking in the other direction, decreasing the score by minimal increments until we reach the lowest-strength configuration. The resulting ordering ranks configurations of n components under the attribute. We found this simple approach sufficient for our needs. More sophisticated techniques for combining attribute scores (e.g. with learned weights, or nonlinearly) may be considered in future work.

Inference

When a component does not share the active attribute, we perform inference to find a suitable replacement, given context from the rest of the design. This can be thought of as enforcing a design prior. For 3D shapes, we follow Kalogerakis et al. [17] to learn a probability distribution over the shape space that captures relationships between shape components. The distribution is defined over discrete and continuous properties, such as shape type, component styles, number and geometric features of components from each category.

Real-Time Inference

The 3D shape prior of Kalogerakis et al. [17] was designed for sampling novel shapes (not for inference). We developed a real-time inference technique for this model. Given a category q, the technique finds the component from q most compatible with the rest of the shape.

Inference is performed as follows. Let $\mathbf{X}_q$ be the model variables sufficient to identify the most probable component from category q, $\mathbf{X}_e$ be observed features of components from other categories existing in the shape, and $\mathbf{X}_u$ be unobserved shape variables. The unobserved variables include latent shape and component styles, and features from categories not observed in the current shape. The goal of inference is to compute the posterior

$$P(\mathbf{X}_q \mid \mathbf{X}_e = \mathbf{e}) = \frac{\int\limits_{\mathbf{u}} P(\mathbf{X}_q, \mathbf{X}_e = \mathbf{e}, \mathbf{X}_u = \mathbf{u})}{\int\limits_{\mathbf{u},\mathbf{q}} P(\mathbf{X}_q = \mathbf{q}, \mathbf{X}_e = \mathbf{e}, \mathbf{X}_u = \mathbf{u})},$$

where the integral is replaced by a sum for discrete variables. Since the denominator does not differentiate between different components (i.e. is a normalizing factor), we only need to compute the marginal distribution in the numerator. Computing the marginal distribution however requires summing over an exponential number of joint assignments to the variables $\mathbf{X}_u$, which makes its direct computation intractable. We resort to loopy belief propagation to approximate the marginal distribution [20]. Each factor in the model is mapped to a node in a "cluster graph". Belief propagation maintains a set of beliefs (marginals) over these nodes by passing messages along edges to exert the influence of one variable on another. If we mutually connect all nodes including a variable X, the messages would cycle endlessly, resulting in incorrect beliefs. Hence, we introduce auxiliary nodes involving individual variables that are connected to nodes containing the factors in which they appear. The resulting cluster graph has Bethe-style structure and guarantees there is only a single, non-cyclic path along which messages about a variable can flow. In each belief propagation iteration, messages are sent from all nodes to their neighbors synchronously. The beliefs typically converge after 5-10 iterations in under 50ms, allowing real-time approximation of the marginal distribution.

Organizing Components on Sliders

Components are placed on the slider for an associated attribute according to their attribute strength. To increase ease of use, the slider intervals are equally spaced (even if the ranking strengths are not). Second, to increase visual continuity and better organize components in the slider according to their types (so that, e.g., canine heads precede dinosaur heads for the "scary head" attribute, even if MTurk users judged some canine heads to be scarier than some dinosaur heads), the ordering is adjusted to group components by visual style, found by the design prior above. For this purpose, we define attribute strengths for each visual style as the average of the attribute strengths of components associated with this style. We place components on the slider based first on attribute strength of their style, then on their individual strength.

DATASETS

We implemented ATTRIBIT on three domains of 3D models: animals, airplanes and ships. The models were obtained from publicly available commercial libraries (Digimation, Dosch 3D). They were segmented and labeled semi-automatically [17]. Statistics for the datasets are shown in Table 1.

	Airplanes	Animals	Ships
# of source shapes	100	69	42
# of categories	7	11	14
# of components	881	593	639
# of attributes	59	81	21

Table 1: *Input datasets.*

COMPARING ATTRIBIT TO A CATALOG-DRIVEN INTERFACE

We ran an experiment to test the following hypothesis for 3D content creation: high-level attributes help accomplish certain types of content creation tasks more effectively than an alternative data-driven interface where components have to be located by searching catalogs. We chose this comparison for two reasons. The first reason is normalization: the two interfaces can operate on identical input databases, differing only in the semantic exploration aspects introduced in this paper. The second reason is that catalog-driven drag-and-drop interfaces have been shown to be highly accessible to first-time designers in both industrial [24] and academic [6] settings. We chose the state-of-the-art "Modeling by Assembly" (MBA) interface of Chaudhuri et al. [6], which leverages context-sensitive component suggestions. We restricted the experiment to novice users to discount the influence of prior experience with particular modeling interfaces, and because a goal of both ATTRIBIT and MBA is to be as accessible as possible to non-professionals.

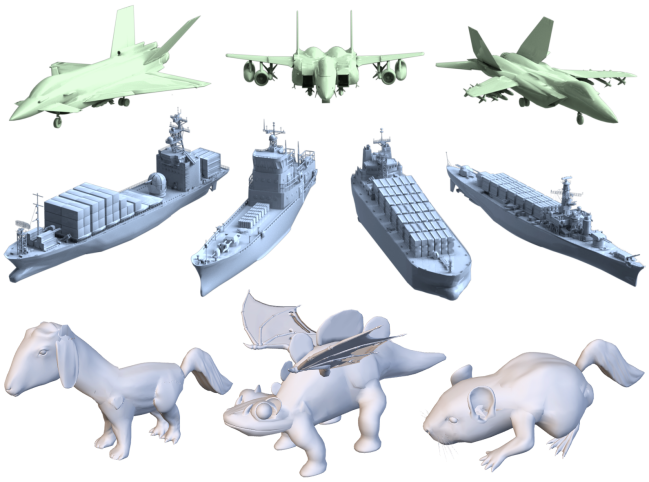


Figure 5: *Dogfight airplanes (top), high-security cargo ships (middle) and cute animals (bottom) created by novice modelers in 15 minutes with* ATTRIBIT. *Images of all 68 models created in the study are provided in supplementary material.*

Method

We recruited 37 persons ranging in age from 11 to 57 . Each reported zero prior 3D modeling experience. We asked them to perform the following content creation tasks:

Airplane. Create an airplane that is best suited to win a dogfight (one-on-one aerial combat) in a computer game.

Animal. Create the cutest possible toy for a small (∼7-year old) child.

Ship. Create the best possible ship for carrying large amounts of valuable cargo across a sea infested by modern-day pirates.

The tasks were chosen to represent real-world open-ended design situations which did not directly correspond to maximizing any single attribute (e.g. no animal attribute was "cute" or any synonym thereof).

Each participant completed one task using MBA, and one using ATTRIBIT, with no indication of which interface was "test" and which was "control". We skinned the two interfaces identically and supplied them with identical segmented and labeled datasets. We also let MBA use the state-of-the-art probabilistic model of [17] (an improvement on [6]) and our real-time inference algorithm for component suggestions.

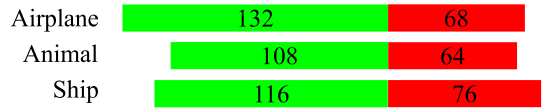
The choice and order of both tasks and interfaces were randomized. No participant was assigned the same task twice, to eliminate the influence of memory of available parts in designing new shapes. Each participant was given a 10 minute orientation with each interface and then allowed 15 minutes of unsupervised modeling time. In all, we collected 68 models: 13 airplanes, 13 animals and 8 ships with each interface (to balance the numbers, the last participant was assigned tasks and interfaces deterministically). A selection of the created models are shown in Figure 5. Images of all models are provided in the supplementary material.

Task	ATTRIBIT	MBA	Both good	Both bad
Airplane	**278**	142	163	19
Animal	**265**	196	53	102
Ship	**226**	166	150	96

(a) *Model better accomplishes goal of task.*

Task	ATTRIBIT	MBA	Both good	Both bad
Airplane	**199**	97	213	27
Animal	**192**	152	65	137
Ship	**183**	119	279	31

(b) *Model is more plausible.*



(c) *Model is better in both goal and plausibility. Green:* ATTRIBIT, *red:* MBA. *Statistics obtained from comparisons of the same pair of models under both questions.*

Table 2: *Distribution of votes cast in blind evaluations of user-created models by MTurk respondents. Overall, raters favoured* ATTRIBIT *models over those created with* MBA *[6].*

Evaluation

To evaluate how well the interfaces helped people accomplish tasks, we asked 71 Mechanical Turk respondents to do blind comparisons of the created models. Each respondent was asked 25 questions via a web survey. Each page of the survey showed renderings of two shapes. One shape was randomly selected from models created with ATTRIBIT, and the other from models for the same task created with MBA. The order was randomized. The respondent was asked to compare either how well the two models accomplished the task (better dogfighter, cuter toy, etc.), or how plausible they were. The rater could say one shape was better than the other, or both were equally good, or both were equally bad. After the rater had answered all 25 questions, (s)he was presented the same questions again in a different order, to check for consistency. Raters with ≥ 8 inconsistent responses were discarded.

The results are in Table 2. Shapes created with ATTRIBIT were preferred by proportions of 52.6% to 94.1% in terms of both goal achievement and plausibility (Table 2c). All preferences for ATTRIBIT in Table 2 are statistically significant according to a two-tailed Mann-Whitney U test with $p < 0.05$.

To see if participants actually found the semantic attributes useful for accomplishing tasks, we logged how often individual attributes were used for exploration. Table 3 shows the top attributes used to select the parts present in the final user models. The common attributes ("faster fuselages", "less scary heads", "more modern hulls") correspond to natural associations of adjectives with the tasks, which indicates that users benefited from semantic controls to achieve design goals.

We also observed that participants could more easily explore designs with ATTRIBIT, because parts do not need to be manually positioned. This is supported by statistics from program

Task	Part	Most Popular Attributes
Airplane	Fuselage	Fast, Military, Aerodynamic
	Wing	Delta-Shaped, Military, Aerodynamic
	Tail	Aerodynamic, Military, Less-Delta-Shaped
	Engine	Advanced, Fast, Strong
	Weapon	Dangerous, Less-Big, Pointed
Animal	Head	Less-Scary, Less-Big, Rounded
	Torso	Curved, Rounded, Large
	Leg	Less-Agile, Stubby, Long
	Tail	Curled, Less-Big, Bushy
	Ear	Curved, Long, Floppy
Ship	Hull	Fast, Less-Rounded, Modern
	Weapon	Large, Advanced
	Cabin	Modern
	Antenna	Military, Powerful, Tall
	Funnel	Less-Complex

Table 3: *Most popular attributes used to select final parts in user models. Part categories are ordered by decreasing frequency of replacements explored using their top attributes. Attributes for a part are ordered by decreasing frequency of replacements initiated using them. "Less-" indicates chosen parts came from the lower half of the ranking for the attribute (no prefix indicates parts were chosen from the upper half).*

logs: on average, participants saw 243.4 distinct designs with ATTRIBIT vs 24.6 designs with MBA, a 9.9x improvement.

Two participants told us that without ATTRIBIT, they would not have been able to fulfill the airplane and ship tasks as they lacked the necessary domain knowledge to assemble suitable models. Informal feedback from others also showed encouraging support for ATTRIBIT.

RANKING QUALITY AND COMPUTATIONAL COMPLEXITY

We evaluated our ranking method by test ranking error: the fraction of test pairs where the predicted relative ordering contradicts ground truth. We used leave-one-out mode, iteratively omitting one element from the training set and testing on all pairs containing that element. As a baseline, the human inconsistency (fraction of times a human ranking for a pair contradicted the majority vote) was 14.0%. The lowest average test ranking error (20.4%) was obtained with the sigmoid loss function and attribute strengths as additional features. Using only the sigmoid, hinge and quad loss functions, the average error was 20.9%, 21.1% and 21.1% respectively.

Training an attribute is $O(P \cdot D)$ where P is number of training pairs and D is feature vector size. Evaluating the ranking score per attribute is $O(N \cdot D)$, where N is component category size. Practically, training the model and ranking the parts for each attribute took $\sim$10 minutes in Matlab on an i7.

APPLICATION TO WEB DESIGN

We believe high-level attributes constitute a viable design paradigm in many different visual (and possibly non-visual) domains. To demonstrate that our approach generalizes, we applied it to an important non-3D domain: web page design.

We collected 30 WordPress blog templates (`http://wordpress.com`), recreated by hand to have compatible DOM structure (background, body, sidebar, title font, header image). Since we focused on exploring attribute variations, we did not experiment with more sophisticated style transfer methods [21] to work with unmodified HTML sources: this forms an interesting avenue for future work.

Web page components were of two types: images and (styled) text. Their feature vectors comprised image and font descriptors respectively, as described in the Appendix. We set up Mechanical Turk surveys analogously to 3D shapes, replacing "shape" with "web page" in the survey text. The attributes learned for web page components were: *artistic*, *casual*, *cheerful*, *colorful*, *creative*, *cute*, *elegant*, *emphatic*, *modern*, *professional*, *romantic*, *simple* and *welcoming*.

At runtime, components were swapped into the current template by exchanging matched CSS/HTML blocks as attributes were varied with sliders. Joint ranking was performed on all components using a simple handwritten model that tries to maintain compatibility with dominant design elements (e.g. background) or between associated elements (e.g. title font and header image). The wheel widget was not relevant for this domain and was omitted. Two sequences of novel web designs created with ATTRIBIT are shown in Figure 6. The "casual" sequence progresses from bare, formal pages to bright, colorful pages, matching our intuition of "casual". A similarly meaningful progression holds for "artistic".

DISCUSSION

This paper introduces an approach for people to create visual content by exploring an assembly-based space of designs with commands based on linguistic terms. We also demonstrate a method to learn relative attributes for 3D shapes. During experiments, we find it is more effective for 3D modeling with high-level design goals compared to related prior work.

Although our initial results are encouraging, there are limitations of the current implementation and questions for future research. First, our interface based on sliders to control relative attributes is a first prototype. It will be interesting to explore other interactions for adjusting semantic attributes, e.g. using voice input and natural language processing.

Second, our current system only allows creation of new content by combining components of existing examples. It will be interesting to study how relative attributes can be mapped back to design parameters to synthesize completely novel designs, for example by incorporating attributes directly in probabilistic models of shapes and design patterns [17, 34].

Third, attributes that do not increase monotonically in the selected feature space cannot be learned effectively with the current method and would require a more complex approach.

Fourth, our assembly-based system leverages highly structured databases of examples containing consistent segmentations and alignments of components. While many data sets are available of this type, it would be nice to relax this constraint by inferring such structures automatically, for example using methods described in [18, 19, 21]).

(a) *Less (left) to more (right) "casual" novel web pages.*

(b) *Less (left) to more (right) "artistic" novel web pages.*

Figure 6: *Novel web page designs created with* ATTRIBIT *using components from 30 exemplars. Attributes learned for web pages were:* artistic, casual, cheerful, colorful, creative, cute, elegant, emphatic, modern, professional, romantic, simple, welcoming.

Finally, our study has considered design tasks only in the domains of 3D modeling and web page design. It will be interesting to investigate the range of application domains for which design with semantic attributes is possible.

ACKNOWLEDGMENTS

Discussions with Vladlen Koltun, Sergey Levine, Scott Klemmer, Jerry Talton and Karan Singh helped improve the paper. Mixamo Inc. provided the GUI artwork. This research was supported by funding from Google, Adobe, Intel (ISTC-VC) and the National Science Foundation (CCF-0937139, CNS-0831374).

REFERENCES

1. Amberg, B. Optimal step nonrigid ICP algorithms for surface registration. In *CVPR* (2007).
2. Berg, T. L., Berg, A. C., and Shih, J. Automatic attribute discovery and characterization from noisy web data. In *ECCV* (2010).
3. Buxton, B. *Sketching User Experiences: Getting the Design Right and the Right Design*. Interactive Technologies. Morgan Kaufmann, 2010.
4. Carvalho, V. R., Elsas, J. L., Cohen, W. W., and Carbonell, J. G. A meta-learning approach for robust rank learning. In *SIGIR* (2008).
5. Chapelle, O. Training a support vector machine in the primal. *Neural Comput. 19*, 5 (2007).
6. Chaudhuri, S., Kalogerakis, E., Guibas, L., and Koltun, V. Probabilistic reasoning for assembly-based 3D modeling. In *SIGGRAPH* (2011).
7. Chen, D.-Y., Tian, X.-P., Shen, Y.-T., and Ouhyoung, M. On visual similarity based 3D model retrieval. *Computer Graphics Forum 22*, 3 (2003).
8. Coyne, B., and Sproat, R. WordsEye: an automatic text-to-scene conversion system. In *SIGGRAPH* (2001).
9. Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. Describing objects by their attributes. In *CVPR* (2009).
10. Ferrari, V., and Zisserman, A. Learning visual attributes. In *NIPS* (2007).
11. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., and Dobkin, D. Modeling by example. In *SIGGRAPH* (2004).
12. Guillory, A., Chastain, E., and Bilmes, J. Active learning as non-convex optimization. *J. Machine Learning Research 5* (2009).
13. Hartmann, B., Wu, L., Collins, K., and Klemmer, S. R. Programming by a sample: rapidly creating web applications with d.mix. In *UIST* (2007).
14. Hartmann, B., Yu, L., Allison, A., Yang, Y., and Klemmer, S. R. Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. In *UIST* (2008).
15. Igarashi, T., Matsuoka, S., and Tanaka, H. Teddy: a sketching interface for 3D freeform design. In *SIGGRAPH* (1999).
16. Joachims, T. Optimizing search engines using clickthrough data. In *SIGKDD* (2002).
17. Kalogerakis, E., Chaudhuri, S., Koller, D., and Koltun, V. A probabilistic model for component-based shape synthesis. In *SIGGRAPH* (2012).
18. Kalogerakis, E., Hertzmann, A., and Singh, K. Learning 3D mesh segmentation and labeling. In *SIGGRAPH* (2010).
19. Kim, V., Li, W., Mitra, N., Chaudhuri, S., DiVerdi, S., and Funkhouser, T. Learning part-based templates from large collections of 3D shapes. In *SIGGRAPH* (2013).

20. Koller, D., and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

21. Kumar, R., Talton, J. O., Ahmad, S., and Klemmer, S. R. Bricolage: example-based retargeting for web design. In *CHI* (2011).

22. Lee, B., Srivastava, S., Kumar, R., Brafman, R., and Klemmer, S. R. Designing with interactive example galleries. In *CHI* (2010).

23. Marks, J., Andalman, B., Beardsley, P. A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfister, H., Ruml, W., Ryall, K., Seims, J., and Shieber, S. Design galleries: a general approach to setting parameters for computer graphics and animation. In *SIGGRAPH* (1997).

24. Maxis Software. *Spore*. 2008. `http://www.spore.com`.

25. Murphy, G. L. *The Big Book of Concepts*. Bradford Books, 2004.

26. Oliva, A., and Torralba, A. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. J. Comput. Vision 42*, 3 (2001).

27. Olsen, L., Samavati, F., Sousa, M. C., and Jorge, J. Sketch-based modeling: a survey. *Computers & Graphics 33*, 1 (2009).

28. Parikh, D., and Grauman, K. Interactively building a discriminative vocabulary of nameable attributes. In *CVPR* (2011).

29. Parikh, D., and Grauman, K. Relative attributes. In *ICCV* (2011).

30. Perez-Cruz, F., Navia-Vazquez, A., Figueiras-Vidal, A. R., and Artes-Rodriguez, A. Empirical risk minimization for support vector classifiers. *Trans. Neur. Netw. 14*, 2 (2003).

31. Russakovsky, O., and Li, F.-F. Attribute learning in large-scale datasets. In *ECCV* (2010).

32. Schelling, T. C. *The Strategy of Conflict*. Harvard University Press, 1960.

33. Sproull, R. F. Parts of the frontier are hard to move. *Computer Graphics Forum 24*, 2 (1990), 9.

34. Talton, J., Yang, L., Kumar, R., Lim, M., Goodman, N. D., and Měch, R. Learning design patterns with Bayesian grammar induction. In *UIST* (2012).

35. Talton, J. O., Gibson, D., Yang, L., Hanrahan, P., and Koltun, V. Exploratory modeling with collaborative design spaces. In *SIGGRAPH Asia* (2009).

APPENDIX

Automatic Placement and Stitching

It can be tedious to assemble designs from components manually. For 3D shapes, we developed a placement algorithm to automatically readjust the assembly to accommodate newly added parts (re-scaled to have comparable relative scales [17]). A preprocessing step records "slots" for each part that connect it to others in its source shape. At runtime, we sort all parts in the assembly by decreasing average part scale of their categories, yielding a sequence $p_1, p_2, p_3, \ldots$. The first (canonically largest) part p_1 is held fixed. The second part p_2 is placed relative to it by identifying valid connection points between the two parts. If slots with matching labels (e.g. leg-torso) exist, they constitute these connection points. Else, we check if any part p_1' from the category of p_1 exists in the source mesh of p_2. If the check succeeds, we map the connection points between p_1' and p_2 to p_1 via non-rigid registration and use them to attach p_2 to p_1. If the check fails, we try with p_1 and p_2 exchanged. The process repeats with the next part in the sorted order. Parts are smoothly stitched together using the method described by Chaudhuri et al. [6].

Features for 3D Shape Components

The feature vector for a part comprises: i) low-level geometric features (same as [17]): part scales, histograms of curvature, shape diameter, and light-field descriptor values [7] — the dimensionality is reduced by PCA; ii) high-level features representing nonlinear transformations of the low-level features. Specifically, the high-level features are probabilities for the part to belong to each part and shape "style" estimated by a statistical model [17]. The probabilities characterize soft memberships of the part in these visual "styles", and also take into account the context of the parent shape of the part. This context provides additional useful information for the part (e.g. that it belonged to a dinosaur).

Features for Web Page Components

The feature vector for a web page component that is an image (background or header) includes: i) color histograms in CIE L*a*b space (4 bins per channel) and ii) the GIST descriptor [26] with a 4×4 spatial resolution, where each bin contains the corresponding image region's average response to steerable filters at 6 orientations and 4 scales.

The feature vector for a web page component containing text (sidebar/body/title) includes: i) histograms of curvature of character outlines (10 bins), ii) 32 Fourier coefficients of the signal representing the distance of the outlines to their centroids, iii) area coverage of characters w.r.t. their bounding rectangle, iv) aspect ratio of characters, and v) width of pairs of letters divided by their height. These features are measured for each character individually: we use the median values as final features. In addition, we include features from the CSS style: vi) a boolean indicating whether the font is italicized, vii) a boolean indicating whether the font is bold, and viii) the font size. Finally, each feature vector is augmented with nonlinear transformations of the low-level features from a probabilistic model, analogously to 3D shape components.