Gestural Inputs as Control Interaction for Generative Human-AI Co-Creation

JOHN JOON YOUNG CHUNG, University of Michigan, USA
MINSUK CHANG, Naver AI LAB, Republic of Korea
EYTAN ADAR, University of Michigan, USA

While AI-powered generative systems offer new avenues for art-making, directing the algorithms remains a central challenge. Current methods for steering have focused on conventional interaction techniques (widgets, examples, etc.) In this position paper, we argue that the intersection of user needs in creative contexts and algorithmic capabilities requires re-thinking the interactions with generative AI. We propose that rough gestural inputs, such as hand gestures or sketching, can enhance the experience of human-AI co-creation–even for text. First, the undetermined and ambiguous nature of gestural inputs corresponds to the purpose and the capability of generative systems. Second, rough gestural can be intuitive and expressive, facilitating iterative co-creation. We discuss design dimensions for inputs of artifact-creating systems, then characterize existing and proposed input interactions with those dimensions. By analyzing existing tools and proposing possible input designs, we highlight how gestural inputs can expand the control interaction for generative systems. We hope future generative human-AI co-creation systems actively adopt gestural inputs so that user intentions are best supported while maximizing the perceived efficacy of generative algorithms.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: generation, controllability, gestural input

ACM Reference Format:
John Joon Young Chung, Minsuk Chang, and Eytan Adar. 2018. Gestural Inputs as Control Interaction for Generative Human-AI Co-Creation. In *3rd Workshop on Human-AI Co-Creation with Generative Models, March 22, 2022, Virtual*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Technologies such as Generative adversarial network (GAN) [14] and pretrained language models (PLM) [4] have the potential to enable human-AI co-creation. These algorithms are attractive in creative contexts as they can generate novel creations–something the human hadn't thought of. However, this can turn into a potential downside, as novelty and surprises can misalign with the user's intention and preference. For example, produced text can suddenly take a turn from what the author wants. This uncertainty leads these algorithms to be iteratively used, re-running the algorithm until the user gets the desired results. Controllability can be a key to minimizing this iterative use: it can steer the behavior of algorithms while managing the user's expectations of the algorithms. In this paper, we propose that rough gestural 'sketches' coupled with abstract representations of content (i.e., information visualizations) can facilitate control interaction for generative algorithms.

Our proposal is motivated by limitations in existing control interactions for generative algorithms. There have been several approaches—from inputting a a simple number (e.g., *have a violin play with the maximum amount of vibrato, by*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
Manuscript submitted to ACM

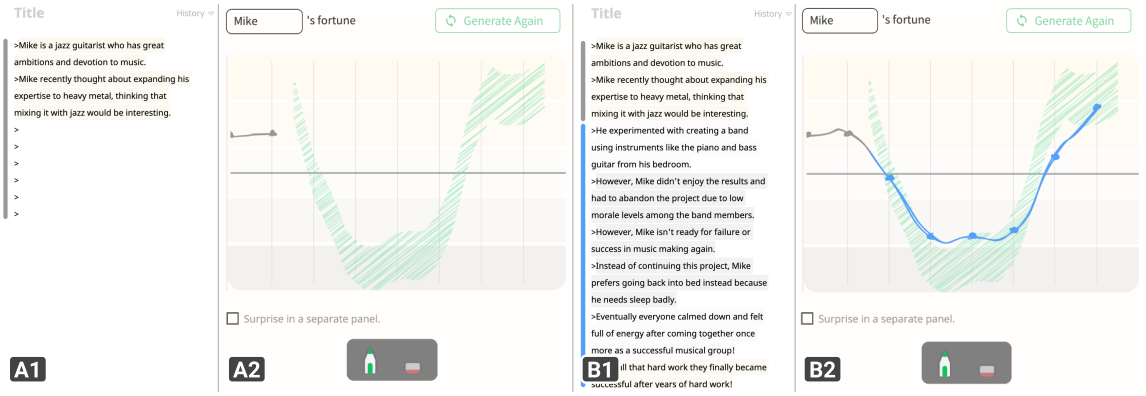

Fig. 1. TaleBrush facilitate human-AI story co-creation by allowing users to intuitively control and sensemake story generation with a line sketching interaction. Specifically, users can control the protagonist's fortune with line sketching. In TaleBrush, the writer can write a portion of a story (A1) and sketch the change of the protagonist's fortune as control input (A2, green shaded area). In the sketched line, the x and y positions stand for the chronological story position and the protagonist's fortune, respectively. For the protagonist's fortune, the higher the position, the better the fortune. In the sketch, the width shows the possible variance in the fortune of the generated sentences. With the given line sketch, TaleBrush generates story sentences (B1, indicated with blue). These sentences are then visualized upon the original sketch (B2, the blue line and dots).

setting the parameter value of 1.0 [28]) to providing natural language prompts (e.g., *produce an image of a dragon sitting on a castle* [23]) or providing examples (e.g., *make this photograph look like this example from Picasso* [13]). However, we argue that these approaches are limited in different ways. Numerical inputs imply a high level of control, and the user's expectation about this precise input would not match with the algorithmic capability. Natural language prompts and examples are potentially problematic if the end-user needs to iterate on what the system generated. They may not understand why the algorithm did what it did, how to correct for it, or may simply be challenged to find or create a new example or prompt. The cost of iterative practice may make generative algorithms unappealing in practice.

In answer to many challenges for generative propose, we propose that rough gestural inputs, such as sketching, may be a sweet spot for human-AI co-creation. First, the gestural input conveys imprecise and ambiguous intentions [15, 18], which corresponds to the nondeterministic nature of generative algorithms. This doubly reduces the expectation on what the algorithm will produce–the drawing surface is in an abstract space and the region is both produced and seen as 'sketchy.' Second, because impreciseness is allowed (e.g., simple brush strokes [9]), the interaction of specification would be easier. With easier interactions, the end-user may not need to think as carefully about the examples or prompts they are generating, thus allowing for more rapid iteration.

While gestural inputs have already been used as materials for generation (e.g., generating photo-realistic images from low fidelity sketches [6]), our specific suggestion is on expanding their roles to control "how the generation should be done." Put another way, sketching has been largely used as *input* whereas we propose to utilize the interaction as a mode of *control*. This is not to imply that there is a clear separation between the two, as they are often inexorably connected as mechanisms to have a system produce the desired output. Rather, it is the case that in implementation, we often use the sketch as input (e.g., make a photorealistic version of this Dragon I scribbled) and other widgets for control (e.g., by adjusting this slider I am indicating how to bias color selection).

One example of 'sketch-as-control' is our controllable story generation system, TaleBrush (Figure 1). Here, TaleBrush leverages *abstract visual representation* of the character's fortune to control the generation of story. The canvas is a 2D

plane that allows for the specification of the protagonist's fortune (y-axis) along the progression of the story (x-axis). In such interface, the specification interaction is as simple as a single stroke of line. It is much easier to specify and iterate compared to alternatives, such as having multiple sliders for different parts of the story. Moreover, the ambiguous and imprecise nature of sketching would correspond to the user's ambiguous intentions and the algorithm's uncertainty.

In this paper, we expand on this idea. We first introduce the design dimensions for inputs of artifact-creation systems. They include the types of support one wants with an AI tool as well as considerations of algorithmic uncertainty, precision of input, and ease of iteration on algorithms and inputs. With the design dimensions, we characterize different existing input types for generative human-AI co-creation. We specifically discuss our example system, TaleBrush [8], how it adopts sketched inputs to facilitate iterative human-AI co-creation in story writing. We believe that considering sketching and gestural inputs as mechanisms of control will enable new ways to support human-AI co-creation.

2 DESIGN DIMENSIONS OF ARTIFACT-CREATION SUPPORT

We first consider possible dimensions for designing control interactions for creation support. In this context, we scope "creation support" to those systems that help creatives directly implement artifacts. We do not consider those tools that are more indirect tools (e.g., systems that critique a work). This boundary is something we have previously considered in surveying the range of tools in the creative space [7]. We propose a focus on five high level aspects: 1) the *type of support*, 2) a system's *algorithmic uncertainty*, 3) *algorithmic iteration*, 4) *input precision*, and 5) *input iteration*.

2.1 Type of Support

Creativity support tools (CSTs) are a broad category–even when restricting to those that directly influence artifact creation [7]. Within this high-level definition, we find tools that are intended to augment, transfer, or generate. While the last two are perhaps the most relevant to our proposal, we briefly cover the full range.

CSTs that provide *augmentation* support often enhance a task the creative is already doing through computational means. Many direct manipulation creation tools fall into this category. The least 'intelligent' of these tools simply replicate existing tools in a digital format. For example, a digital painting canvas has various types of digital brushes. Other augmentation tools provide some limited automation. For example, a bucket tool will flood fill a closed area in a sketch. Most of these tools are highly deterministic–they are "predictable" and more naturally correspond to the user's mental model of what the system will do. When using augmentation-focused tools, the user is firmly in control over both the idea and style of the final artifact.

The second and third types of support, *transfer* and *generation*, use variants of generative algorithms. In contrast to the augmentation category, the end-user is ceding some creative control to the tool. Though, of course, the human maintains ultimate control over what makes it into the final artifact.

Transfer tools are designed to turn one artifact into another. A common feature is that one input to such tools is some kind of 'original' artifact (e.g., a picture, a piece of text, a sketch, etc.). The tool will then act on this input to generate a variant. There is a wide range of tools that fall into this category, and are often modality-specific. For example, in the visual design/art space we see tools that transfer one visual art piece's style to another image [13]. Other tools in the space will transform rough sketches into photorealistic images [6]. In writing, we observe algorithms that change the style of the text. Just as with the images, we can rewrite a sentence using a particular author's style [27]. Not all transfer tools need a target 'style.' In music, for example, there are tools that add transform some input piece of music by adding effects like delay or compression [26].

Finally, we observe tools focused on *generation*. With these, content is generated from the algorithm from inputs that are incomplete or come from a different modality. For example, an input might be some previous part of the music, story, or some portion of drawings. The algorithm's purpose is not to change this initial input, but rather to add to them. In music and text, these algorithms would either continue the user-given content or infill between the user-given start and end context [3, 5, 10, 20]. In visual arts, we find algorithms that fill in empty spaces in an image [11, 19]. Note that many tools sit somewhere between transfer and generation and may depend partially on how the underlying task is defined. For example, we might have a tool that automatically colors a part of an image. From the perspective of the whole image, this may be *transfer* (especially if the input is some kind of color palette or color model). However, because we are also *generating* new colors, we might treat the colorization task as generative support.

Depending on the type of tool, we will certainly expect different types of controls to direct the underlying systems.

2.2 Algorithmic Uncertainty

The algorithmic pipeline of a tool can differ depending on the level of uncertainty in their behaviors. *Deterministic algorithms* are one extreme in that users are able to predict the result when using these algorithms. Direct manipulation implementations are, naturally, one example. When a box is dragged with a mouse cursor, the end-user knows where it will end up. However, automated algorithms with clear rules are also deterministic. For example, with flood fill (e.g., a bucket tool), the user knows that they will fill closed areas. If something goes wrong, the user is able to isolate the problem.

On the other extreme are *non-deterministic algorithms* which represent many machine learning algorithms. In some situations, the uncertainties in these algorithms will lead to failures. For example, in comic colorization 'flatting' is the process of automatically creating colored polygons under different parts of the linear art (e.g., one for the face, one for the shirt, etc.). The algorithm for automated flatting contains inference (i.e., the shirt does not need to be closed, the system infers the shape of it). The uncertainty here will lead to unexpected bleeding [29]–a failure case.

However, in the creation setting, and specifically for generative algorithms, uncertainty need not be a failure. Or more precisely, the line between a novel, desirable result, and an error are not necessarily clear cut. There is rarely a single gold standard for what should be generated, and the user might subjectively decide whether it fits their goal.

2.3 Ease of Algorithmic Iteration

Iterative design is important in creating artifacts. This is due largely to the explorative nature of the task. How easy it is to iterate depends on the algorithm. There are various algorithmic properties that will impact the user's ability to iterate. First among these is latency–the time taken by the algorithm for each cycle. The lower the latency, the easier to iterate. For example, when moving a box with a mouse cursor, the iteration is in real-time as the box's position updates instantly with the user's movement. On the other hand, many generative algorithms tend to take significant time to generate artifacts, significantly slowing down iteration.

A second algorithmic aspect that impacts iteration is scope. Here, we define scope as relating to what parts of the artifact are iterated on. For example, with the direct manipulation of the box's position, only the box's position is changing, but not the color or size. At the other extreme are style transfer algorithms [24]. Here, every time we run the algorithm, the entire image (or many parts of it) will change.

Iteration naturally connects back to the algorithmic uncertainty. If the user better understands the behavior of the algorithm (e.g., what the transfer algorithm changes, and how), iteration may become easier. With high uncertainty, the user may need to iterate many times to get the effect or artifact they want.

2.4 Input Precision

While there are numerous input approaches for artifact-creation systems, they vary on the spectrum of precision. These varying levels are helpful in different contexts. The most traditional type of widgets receives one specific value. For example, a number in the slider or a category in a dropdown box. With this precise control, users will expect the output to precisely react in the output.

Clearly, not all inputs need to be precise. *Natural language prompt* is one example and can handle a wider range of input precision [16, 22]. Roughly specified language would be imprecise, but at the same time allow a high degree of freedom in how it can be interpreted. For example, asking for a "rough texture" can mean many things—anything from Jackson Pollock's chaotic style to Van Gogh's impressionism. However, language *can* support finer control. For example, we can say "move the selected square 3 pixels left," does not leave a lot of room for misinterpretation.

At the imprecise end, we often find *Examples* as inputs [12, 24]. With examples, it is up to the algorithm to determine, if it can, which aspects of the input should be followed closely and which are only suggestions. For example, when transferring the style of visual arts, with Van Gogh's The Starry Night, it is not clear if the user wants its color to be replicated or textures to be the focus of the generation. Adding more examples might make the target clearer. However, it may be hard for the user to determine which attributes overlap between the examples and which are left ambiguous. The interaction with uncertain algorithms makes this problem even more complex as it is not clear if the issue is the input or the inherent ambiguity of the system.

As with language prompts, *Gestural inputs*, such as sketches or hand gestures, can also have a wide range of precision. For example, gestural inputs for direct manipulation require outputs to exactly follow the given input. When resizing a box in graphics editors, users expect the box to follow the cursor they are moving. However, they also can be used for the input of low precision–for example, sketches are used for expressing flexible and lightweight ideas with their roughness, ambiguity, and uncertainty [15, 18]. Similarly, hand gestures have been used to provide imprecise but intuitive and flexible inputs, such as serving as rough scaffolds in 3D modeling [17]. As we see in these examples, gestural inputs can be designed to provide high intuitiveness and flexibility and be traded off against precision.

We note that input precision is often related to input difficulty. As we know from psychophysical properties such as Fitts's Law [21], certain input precision comes at the cost of time or difficulty. Lower precision interactions, such as gestures, can often lower the difficulty of interaction.

2.5 Ease of Input Iteration

Just as we consider the iterative cost at the algorithmic level, it is worth considering it at the input level as well. Though these two might be tied, in many situations a tool may have relatively small–or at least limited–iterative cost in the back-end, but with widely diverging front-end costs. Thus, different input approaches vary greatly in how well they can be iterated with.

Traditional input widgets, such as numerical values on sliders, are relatively easy to iterate with. With a single slider, the control options given to the user are tightly restricted and a change in value does not require much effort. Only if there are numerous input parameters and when the user needs to change (or decide if to change) many of them, the iteration cost increases.

Other types of inputs, such as natural language prompts or examples, increase iterative costs. This is largely due to the vast space of options for these modalities. For natural language prompts, the user would need to come up with better wordings or more specific details on the prompts. This can be tricky if the user is to express differences in degree

(e.g., how to express the level of roughness of the texture the user intended?). Similarly, iterating with examples is difficult because the user needs to search for more adequate examples. If such an example can't be found or created the user will not be able to iterate.

Gestural inputs can potentially be used to maximize the benefits in iteration. While these come at the cost of precision, gestural input can facilitate flexibility and intuitiveness. When the specification does not need to be precise and may not have high cognitive demands, it can be quick. For example, the user can erase and redraw a portion of the sketches to quickly change the specifications.

2.6 Input Directness

The last spectrum we consider for our design space is input directness. That is, does the input method target the artifact directly or indirectly [7]. With direct input, the end-user indicates the artifact or subject 'target.' Because of this directness, the input is usually in the same medium as the target artifact. In some situations, a portion of the artifact can also be used for high input directness. For example, we can select a portion of the image or the story. At the other extreme are those inputs that do not directly impact the artifact, but may give broad instructions on how the tool should implement something. The simplest example might be a slider control for some parameters. The user isn't touching the artifact directly (i.e., the story or image) but the change in the slider guides the tool. The modality of indirect input can be far from the medium (e.g., visual arts as artifacts and numbers as inputs). As with our introductory example, abstract visual encodings can also be used for indirect inputs. In that example, the end-user was drawing in the character 'alignment' space to produce text.

3 DESIGNING GENERATIVE CO-CREATION TOOLS

The five design dimensions above represent a large design space. From here, we can begin to consider points in the space that are either required, or are more suitable, for co-creation tools.

3.1 Requirements for Generative Co-creation Tools

As we argued above, generative algorithms are usually used to support transfer or generation. Recent trends have certainly leaned towards machine-learning based approaches. Thus, we are largely working in the non-deterministic space. We argue that non-determinism implies a couple of key requirements for tools.

First, **iteration should be easy and fast.** In creative tasks, iteration and exploration are necessary as they expose the artist to more options and, eventually, a concretization of 'direction' [1, 2]. Thus, users of creative tools often *want* to be able to iterate, which is well-aligned with the reality that to use non-deterministic tools, they *need* to iterate. Unfortunately, sometimes the cost of iteration becomes high. Thus, tools should either act to speed up the number of iterations, and if that is not possible, to reduce them. In both situations, reducing the iteration cost is critical.

Second, **algorithmic uncertainty should match the user's expectations.** With standard algorithms, we would only need to worry about the user's expectations in how their input and deterministic output relate. For example, that dragging an icon into the trash would lead to it being deleted. However, with non-deterministic algorithms, the user doesn't only need to model the specific output but also the range of possible outputs. Without this understanding, end-users are likely to be dissatisfied with the results. They will also find it difficult to model how their input choices will lead to a better, or more certain, output. Our advantage in creative tools is that some degree of uncertainty is actually a desired property. Our goal is not necessarily to make the tool appear deterministic as creativity often requires 'surprise.' Thus, users both want and expect some level of (controlled) uncertainty. A user may be willing to make a

rough specification and understand and expect that the tool will have some degrees of freedom within that space. Note that none of this is to say that we need to force the algorithms to match the user's expectations. In some cases, it is feasible to identify places where users do not have well-defined expectations. In other cases, we may be able to change the user's expectations.

3.2 Algorithmic Design

On the algorithmic side, there are various ways to approach the iteration and uncertainty problems. One is by adding extensive controllability. This would provide the user with fine-grained controls when steering the behavior of the generative algorithms. This requires building algorithms that can actually accept all these controls.

Such control may reduce the number of iterations, at least from the perspective of the algorithm. That is, fine-grained controls would reduce the ambiguity of the input and enable the generative system to produce a more targeted response. Detailed controls also work to 'teach' the end-user how to model and direct the underlying algorithm. Their expectations of system capabilities would come to be more in alignment with reality with fewer iterations. Of course, reducing the latency of the algorithm would also facilitate the ease of iteration. Clever designs, such as using smaller models, before executing more costly larger ones may help here.

However, shifting the responsibility of satisfying our requirements to the algorithmic side exclusively is not realistic. Regardless of the algorithm, many bottlenecks for iteration are from the interaction side. Increasing the number of controls may be cognitively costly for the end-user. This is not to say that fewer controls or simpler inputs, such as examples or prompts, reduce cognitive cost. In fact, the cognitive cost of figuring out how to change or create an example can be equally bad. When coupled with the specific demands of creative applications–that we want some iteration and some surprise–achieving a 'sweet-spot' through algorithmic means alone seems implausible. Put another way, simply changing the algorithm can't solve our problem if the interface costs are high or the user's requirements for a creative tool are unmet.

3.3 Input Design

To achieve our requirements, we argue that interaction is a critical factor. We focus on possible input approaches. These will naturally range based on the type of input directness.

3.3.1 Direct Inputs: A Small Space of Design. Direct input and control of the artifact are usually done in the same medium as the artifact. For example, when the target artifact is visual, we might use low-fidelity sketches on the drawing surface that will then be transferred to high-fidelity images on what is essentially the same surface/encoding [6]. In other cases, the algorithm may simply append elements to the sketch [11]. This type of input serves as the 'material' for the generation—where the transfer is applied or what the generative algorithms build upon.

While direct inputs may depend on the application domain, their specific type may be largely constrained to a small design space. In large part, this is because the representation depends on the target artifact's medium (e.g., a drawing canvas). Additionally, the interactions are largely constrained by the underlying algorithm. For example, we may train an algorithm to produce a photo-realistic image given a low-resolution sketch. Such datasets are more readily available and easier to produce. The user-facing input modality and form are thus constrained to something that looks like the training data. Finally, direct inputs create a set of expectations for the end-user that need to be maintained in the interactive controls. Because of these constraints, which may limit our design space options, we move to consider indirect inputs.

3.3.2 Indirect Inputs: Approaches and Their Limitations. Indirect inputs serve as instructions for both *what* and *how* to generate. Unlike direct inputs, they are not dependent on the medium of the artifact. One can work in abstract spaces or through abstract representations. Thus, there is often more freedom on what can be designed for indirect inputs. The consequence of freeing ourselves somewhat from the constraints of the domain also enables us to consider additional algorithm types. This flexibility further affords a better ability to match the end-user's high-level intentions rather than forcing them to work within the constraint of their algorithm and interface. However, this is not to say that all indirect inputs are good ones. A novel indirect interaction might be further from the target artifact's modality and thus might be harder to master when the mapping is complex. A poorly designed indirect interaction can also increase cognitive costs and reduce the ability to iterate. All together, indirect inputs open up a vast space of possibilities but also introduce various pitfalls.

To better understand which aspects may help or hinder, we focus on three types of inputs—traditional input widgets, natural language prompts, and gestural inputs. Traditional input widgets, such as sliders for numerical inputs, represent the simplest option. If we are able to define these, it often means that we can provide an interface that matches the user's expectations to the actual behavior of the system. For example, for melody generation, the user would be able to input the degree of how much the melody should be in major scale or minor scale, in a single pitch-perfect value [20]. Note, however, that this depends on the widget representing some construct understood by the end-user. If the label on the slide is ambiguous (e.g., this will control the 'brightness' of the text) or novel (e.g., this will control the 'certitudeness' of the text), the user may struggle with the control. Though, this may improve with continued use and calibration. More critically, when a user only has a rough idea of what they want to generate, a standard input widget may be insufficient. As critically, the algorithms themselves may not be able to deliver on the precision of the input. Thus, the interface is over-promising. The ease of iteration with input widgets largely depends on the complexity of the interface. A single slider is simple, but many controls and their interactions will naturally become more challenging.

Recent generative co-creation systems have enabled the use of indirect natural language prompts as input. For example, natural language prompts can steer vision-language models to generate visual images [16, 22]. Examples are often used in the context when the user wants to guide the model to apply styles similar to the given examples to other artifacts [25]. As these approaches can be used with imprecision or ambiguity, they are useful for giving high-level specifications on generations. As the prompt becomes more specific, the underlying algorithm may not be able to produce satisfactory results. Furthermore, the vast space of possible inputs makes learning 'what to say' challenging and will hamper iteration.

Surprisingly, few human-AI co-creation tools have used gestural or sketch interactions for indirect control. We argue that this is a missed opportunity as there are a number of benefits to this approach. In the next section, we expand on this possibility and why it may be appropriate.

4 GESTURAL INDIRECT INPUTS FOR GENERATIVE CO-CREATION

We believe that gestural or sketch based interfaces for indirect specification satisfy our requirements for co-generation tools. At the very least, this approach may *complement* other input controls. Simple gestural interactions, like producing a rough sketch, are easy to iterate on. This characteristic can complement more effortful controls such as prompts or examples. Moreover, the multi-dimensional characteristics of sketches and gestures can reduce the effort to interact with multiple attributes simultaneously. For example, 2D sketching coupled with pressure and speed recognition can be used to simultaneously encode multiple parameters. This flexibility also means that we can work in abstract visual encodings. Finally, as we have argued before, gestural inputs convey the sense of being rough and flexible. This strongly

aligns with the non-determinism of the algorithm, the ambiguity of the user's intent, and to convey the 'unfinished' nature of the generative process.

As a demonstration of the feasibility of this approach we describe our system TaleBrush [8] (Figure 1). TaleBrush is a human-AI story co-creation tool that generates story sentences according to the specifications of the protagonist's fortune. For example, if we were describing Cinderella's fortune we might say that her fortune started low (with he stepmother and sisters), improved greatly as she went to the ball, collapsed as she was forced to flee, and then improved again when she was found by the prince.

TaleBrush allows the user to first input a portion of the story (direct input) in the text box (Figure 1A1). Then, they can sketch out how the protagonist's fortune in a 2-dimensional line sketches as in Figure 1A2. This is roughly a standard time series with the x and y axes standing for sequence position and fortune levels respectively. Using this sketched line (which is actually represented as a sketch rendering), TaleBrush will generate a story (Figure 1B1). Because the underlying algorithm is ambiguous, and may not exactly match the desired fortune sketch, the best matching generation is also displayed in the visualization (Figure 1B2).

With TaleBrush, the benefits of gestural inputs hold. First, iteration on the generation is easy. The user only needs to redraw parts of the sketch. Additionally, a single drawn line expresses two dimensions simultaneously: where in the story, and at what fortune level, should the sentence be. Notably, the first (position) is a direct input whereas the fortune level represents an indirect one. In reality, we also use the speed at which the sketched line is drawn to indicate how much ambiguity the user will tolerate in the generated result. This is visually represented in the thickness of the line. A thinner line indicates the user wants a better match. Internally, this is implemented by regenerating the sentences multiple times and finding the one that best matches that desired fortune level. This visualized boundary further emphasizes the ambiguity and non-determinism of the algorithm. Note that a 'sketch' does not necessarily mean a 'sketchy appearance.' However, we have opted to use this type of rendering to further lower the user's expectations that the algorithm should be precise [15, 18].

4.1 Design Approaches for Gestural Indirect Inputs

Implementing TaleBrush has given us some insight about what may work well and more poorly for gestural indirect inputs.

4.1.1 Ease of Iteration on Input.

Combine direct and indirect input if possible. For some generation tools, indirect inputs may be sufficient. For example, if the tool generates any character biographies based on the good-evil nature of the character, then it might not require direct inputs (as in our introductory example). However, as with TaleBrush, certain tasks require control that can only come from direct input. That is, the user needs to be able to indicate, "where the generation should be done" (e.g., where in the story a certain fortune level should exist or what does the start of the story look like?). In some cases, as we did with TaleBrush, the indirect and direct controls can be combined into a single gestural sketch. That is, with a single brushstroke, the sequential position (x position–the 'where') and the level of the protagonist's fortune (y position–the 'how') are both specified.

Complement hard-to-iterate control inputs (language, examples). Spatial positions by themselves do not necessarily convey meaning. They are meaningful when combined with semantic structures that can be put on a continuous scale. For example, TaleBrush takes a restricted set of numerical semantics: whether the character's fortune is good or ill.

However, this design can be extended to receive qualitative inputs as the endpoints of the axes. For example, the user can give natural language prompts or examples on each end, and explore the confined space with gestural inputs. This complements the limitations and features of different input approaches. Language prompts and examples lack the ease of iteration, which is the strength of gestural inputs. On the other hand, gestural inputs lack semantics, which language prompts and examples can convey.

4.1.2 *Matching Input Precision with Algorithmic Uncertainty.*

Match the algorithmic precision with the input precision. To have better expectations of the algorithmic behaviors, the user should ideally be aware of the precision of the algorithm. The gestural input can be designed to convey this information. For example, in TaleBrush, this level of precision is conveyed through the width of the sketched line. This was designed to match the median error from the test dataset that was used during the development. Thus, the interaction and representation can be used in ways that reduce ambiguity and help to match expectations.

Controlling the precision. When using gestural inputs, the user's intention regarding the precision might vary. For example, in TaleBrush, the user might have wanted the algorithm to follow the generation more tightly when they draw the line with more care. To reflect such intentions, the system can be designed to leverage other input dimensions to control the precision. In TaleBrush, the sketching speed was used to decide how tightly generation should be done. That is, if the user slowly drew the sketch, we considered that the user indicated that they wanted a better fit (represented as a thinner error envelope). In this way, gestural interactions and representations can be used to align input precision with system capabilities.

5 CONCLUSION

In this position paper, we have explored where generative algorithms sit in the overall design space of co-creative tools. We have further isolated those properties that are desirable, and potentially required, for supporting human-AI co-creation. Our focus was specifically on how inputs (both the 'what' and the 'how') can interact with underlying algorithms. Our focus on enabling iteration and managing expectations allowed us to consider the pros and cons of different input types. Ultimately, we argued that gestural and sketch-based interactions would work well for the control of generative algorithms. We showcased the benefits of this approach with a specific example, TaleBrush. We believe that there are significant possibilities opened up by using abstract visual representations when coupled with these novel interaction types.

REFERENCES

[1] Teresa M Amabile. 1983. The social psychology of creativity: A componential conceptualization. *Journal of personality and social psychology* 45, 2 (1983), 357.

[2] Teresa M Amabile. 2012. Componential theory of creativity. (2012).

[3] Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O. Riedl. 2020. Automated Storytelling via Causal, Commonsense Plot Ordering. *CoRR* abs/2009.00829 (2020). https://arxiv.org/abs/2009.00829

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[5] Chin-Jui Chang, Chun-Yi Lee, and Yi-Hsuan Yang. 2021. Variable-Length Music Score Infilling via XLNet and Musically Specialized Positional Encoding. arXiv:2108.05064 [cs.SD]

[6] Shu-Yu Chen, Wanchao Su, Lin Gao, Shihong Xia, and Hongbo Fu. 2020. DeepFaceDrawing: Deep Generation of Face Images from Sketches. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2020)* 39, 4 (2020), 72:1–72:16.

[7] John Joon Young Chung, Shiqing He, and Eytan Adar. 2021. The Intersection of Users, Roles, Interactions, and Technologies in Creativity Support Tools. In *Conference on Designing Interactive Systems*. ACM, 1817–1833.

[8] John Joon Young Chung, Wooseok Kim, Kang Min Yoo, Hwaran Lee, Eytan Adar, and Minsuk Chang. 2022. *TaleBrush: Sketching Stories with Generative Pretrained Language Models*. Association for Computing Machinery, New York, NY, USA.

[9] Mathias Eitz, James Hays, and Marc Alexa. 2012. How Do Humans Sketch Objects? *ACM Trans. Graph. (Proc. SIGGRAPH)* 31, 4 (2012), 44:1–44:10.

[10] Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for Structuring Story Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 2650–2660. https://doi.org/10.18653/v1/P19-1254

[11] Judith E. Fan, Monica Dinculescu, and David Ha. 2019. Collabdraw: An Environment for Collaborative Sketching with an Artificial Agent. In *Proceedings of the 2019 on Creativity and Cognition* (San Diego, CA, USA) *(C&C '19)*. Association for Computing Machinery, New York, NY, USA, 556–561. https://doi.org/10.1145/3325480.3326578

[12] Emma Frid, Celso Gomes, and Zeyu Jin. 2020. Music Creation by Example. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376514

[13] L. A. Gatys, A. S. Ecker, and M. Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, USA, 2414–2423. https://doi.org/10.1109/CVPR.2016.265

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[15] Mark D. Gross and Ellen Yi-Luen Do. 1996. Ambiguous Intentions: A Paper-like Interface for Creative Design. In *ACM Symposium on User Interface Software and Technology*. ACM, 183–192.

[16] Forrest Huang and John F. Canny. 2019. Sketchforme: Composing Sketched Scenes from Text Descriptions for Interactive Applications. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 209–220. https://doi.org/10.1145/3332165.3347878

[17] Yongkwan Kim, Sang-Gyun An, Joon Hyub Lee, and Seok-Hyung Bae. 2018. *Agile 3D Sketching with Air Scaffolding*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173812

[18] J.A. Landay and B.A. Myers. 2001. Sketching interfaces: toward more human interface design. *Computer* 34, 3 (2001), 56–64. https://doi.org/10.1109/2.910894

[19] Yuyu Lin, Jiahao Guo, Yang Chen, Cheng Yao, and Fangtian Ying. 2020. It Is Your Turn: Collaborative Ideation With a Co-Creative Robot through Sketch. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376258

[20] Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie J. Cai. 2020. Novice-AI Music Co-Creation via AI-Steering Tools for Deep Generative Models. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376739

[21] I. Scott MacKenzie. 1992. Fitts' Law as a Research and Design Tool in Human-Computer Interaction. *Hum.-Comput. Interact.* 7, 1 (mar 1992), 91–139. https://doi.org/10.1207/s15327051hci0701_3

[22] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. arXiv:2112.10741 [cs.CV]

[23] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-Shot Text-to-Image Generation. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8821–8831. https://proceedings.mlr.press/v139/ramesh21a.html

[24] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. 2018. Avatar-Net: Multi-scale Zero-shot Style Transfer by Feature Decoration. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. 1–9.

[25] Jing Shi, Ning Xu, Haitian Zheng, Alex Smith, Jiebo Luo, and Chenliang Xu. 2021. SpaceEdit: Learning a Unified Editing Space for Open-Domain Image Editing. arXiv:2112.00180 [cs.CV]

[26] Christian J. Steinmetz and Joshua D. Reiss. 2021. Steerable discovery of neural audio effects. arXiv:2112.02926 [eess.AS]

[27] Bakhtiyar Syed, Gaurav Verma, Balaji Vasan Srinivasan, Anandhavelu Natarajan, and Vasudeva Varma. 2020. Adapting Language Models for Non-Parallel Author-Stylized Rewriting. arXiv:1909.09962 [cs.CL]

[28] Yusong Wu, Ethan Manilow, Yi Deng, Rigel Swavely, Kyle Kastner, Tim Cooijmans, Aaron Courville, Cheng-Zhi Anna Huang, and Jesse Engel. 2021. MIDI-DDSP: Detailed Control of Musical Performance via Hierarchical Modeling. arXiv:2112.09312 [cs.SD]

[29] Chuan Yan, John Joon Young Chung, Kiheon Yoon, Yotam Gingold, Eytan Adar, and Sungsoo Ray Hong. 2022. *FlatMagic: Improving Flat Colorization through AI-driven Design for DigitalComic Professionals*. Association for Computing Machinery, New York, NY, USA.