Automated Accessory Rigs for Layered 2D Character Illustrations

Jingyi Li
Stanford University

Wilmot Li
Adobe Research

Sean Follmer
Stanford University

Maneesh Agrawala
Stanford University

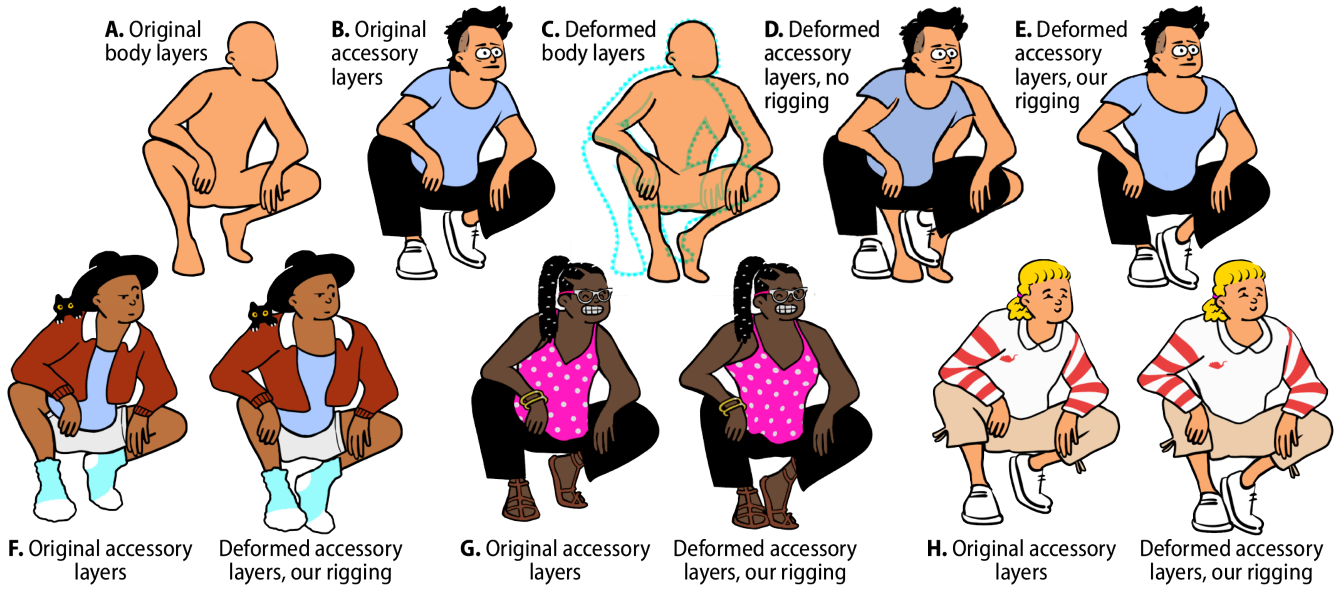

A. Original body layers
B. Original accessory layers
C. Deformed body layers
D. Deformed accessory layers, no rigging
E. Deformed accessory layers, our rigging
F. Original accessory layers
Deformed accessory layers, our rigging
G. Original accessory layers
Deformed accessory layers, our rigging
H. Original accessory layers
Deformed accessory layers, our rigging

Figure 1: Given a collection of layered artwork comprised of "body layers" (A) and "accessory layers" (B) and a user specified deformation to the body layers (C, original overlaid in cyan for comparison), current tools leave the accessory layers unchanged, leading to visual artifacts where the accessories no longer cover the body (D). Our system automatically rigs the accessories with constraints so they properly adapt to the deformation (E). Mix-and-match character creation data sets like Open Peeps [24] provide various accessory layers; our system enables the different accessories to automatically adapt to the same underlying body deformation (F–H).

ABSTRACT

Mix-and-match character creation tools enable users to quickly produce 2D character illustrations by combining various predefined accessories, like clothes and hairstyles, which are represented as separate, interchangeable artwork layers. However, these accessory layers are often designed to fit only the default body artwork, so users cannot modify the body without manually updating all the accessory layers as well. To address this issue, we present a method that captures and preserves important relationships between artwork layers so that the predefined accessories adapt with the character's body. We encode these relationships with four types of constraints that handle common interactions between layers: (1) occlusion, (2) attachment at a point, (3) coincident boundaries, and (4) overlapping regions. A rig is a set of constraints that allow a motion or deformation specified on the body to transfer to the accessory layers. We present an automated algorithm for generating such a rig for each accessory layer, but also allow users to select which constraints to apply to specific accessories. We demonstrate how our system supports a variety of modifications to body shape and pose using artwork from mix-and-match data sets.

UIST '21, October 10–14, 2021, Virtual Event, USA
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8635-7/21/10…$15.00
https://doi.org/10.1145/3472749.3474809

CCS CONCEPTS

• **Computing methodologies** → **Graphics systems and interfaces**; • **Human-centered computing** → *Interactive systems and tools.*

ACM Reference Format:
Jingyi Li, Wilmot Li, Sean Follmer, and Maneesh Agrawala. 2021. Automated Accessory Rigs for Layered 2D Character Illustrations. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3472749.3474809

1 INTRODUCTION

Illustrated 2D characters are prevalent across many visual applications, including storytelling, advertising, animation, and games. Recently, "mix-and-match" 2D character creation tools have emerged that make character design more accessible to a broader range of users. Instead of starting from scratch, tools and templates like Open Peeps [24], Blush [6], Character Creator [11], and Picrew [33] allow users to assemble characters by choosing from discrete sets of predefined variations for attributes such as body shape, skin color, and clothing. These variations are represented as 2D artwork layers that are composited together to produce the final character appearance. We refer to the artwork that represents variations of the character's unadorned body—such as different shapes, poses, and skin colors—as *body layers*. Likewise, we define *accessory layers* as the various objects (e.g., clothing, hairstyles, facial features, accoutrements, etc.) that adorn the base layers. With mix-and-match tools, users can quickly explore a range of character designs by selecting different combinations of body and accessory layers.

However, this convenience comes at the cost of flexibility and expressiveness. While there are typically dozens of accessories to choose from, there are many fewer body shape and pose variations (often just one). For a given application, both end-users and the original character designers may want to modify a character's body shape or pose away from the predefined options. For example, for the character shown in Fig 1, a user may want to explore different torso proportions to better match the character to their own body, or animate an "idle" squatting animation when making the character playable in a video game. The problem is that current mix-and-match tools do not support such modifications. Since the character's accessory layers are only designed to fit with the predefined body layers, edits to the body shape or pose produce artifacts in the accessory layers. For example, in Fig 1D, we see the new body shape protrude out of the clothing—since the independent accessory layers are not constrained to the body layers in any way, edits to the body do not propagate to them. To fix these artifacts, users would have to manually edit or redraw all the accessory layers to fit the edited body.

The goal of our work is to make mix-and-match character creation tools more flexible by allowing users to modify a character's body layers while automatically adapting the accessory layers to fit. Our approach is to automatically generate a *rig*, a set of constraints that specify how changes in the body layers should transfer to the accessory layers, for each accessory layer. Each rig captures and preserves important relationships between artwork layers, such as where and how they should remain attached with respect to each other under deformation. Specifically, we introduce four types of constraints that represent common layer-to-layer interactions: (1) occlusions (e.g., arm against sleeve), (2) attachment at a point (e.g., brooch on sweater), (3) coincident boundaries (e.g., lapel on jacket), (4) overlapping regions (e.g., sleeve to bodice near the armpit). Our rigs model these interactions via inter-layer spatial constraints that cause accessory layers to deform as the body layers change. This approach allows users to customize a character's body shape and pose while still making use of all the accessory artwork that was designed for the default body shapes and poses. Fig 1F–H shows different accessories adapting to the same body deformation specified in Fig 1C.

Our main contribution is the definition of the constraints and automated techniques for rigging accessories with these constraints such that accessory layers appropriately adapt to deformations on the body layers. We use our approach to automatically rig a wide range of character accessories derived from existing mix-and-match data sets and show the results for various modifications to the shape and pose of the character. In addition, we demonstrate two applications where our rigs enable continuous edits that propagate to all the accessories: an interactive character customization interface where users modify various body shape parameters, and animations that change the body shape and pose over time.

2 RELATED WORK

2.1 Mix-and-Match Assembly-Based Modeling

Mix-and-match assembly-based modeling is a popular authoring paradigm for novice users. Most prior research in this domain focuses on creating 3D output and addresses topics such as decomposing existing model collections into interchangeable parts [12, 27], inferring appropriate part combinations [7, 8, 30], and optimizing how parts fit together geometrically [19, 32, 38]. Mix-and-match templates for creating 2D characters have uses from creating personalized avatars [29] to generating character assets [24]. The goal of our work is to make 2D mix-and-match character creation tools more expressive by automatically propagating user edits across all the predefined artwork layers. While some existing tools support such behavior (e.g., customized avatars that can be animated in video games), they typically require all assets to be pre-rigged by hand to encode the relevant inter-part relationships. In contrast, we provide automated rigging tools that facilitate propagation across collections of unrigged 2D artwork.

2.2 Rigging

There is a large corpus of previous research on rigging character models, along with numerous commercial products. A rig is a set of constraints that defines how a character should behave under an external deformation. Many existing commercial systems provide interactive tools that allow users to directly instrument 2D or 3D models with constraints, like handles, joints and bones [1, 3, 5]. Some research systems [20, 34] allow users to create higher level rigs to achieve specific styles of secondary motion effects. Other tools incorporate rig creation into the modeling process, which produces parameterized digital characters by construction [9, 13]. However, this prior work does not apply to our problem setting. We aim to automate the rigging process as much as possible, rather than provide interactive rigging tools. Moreover, existing techniques focus on rigging the character itself rather than accessories.

Two common approaches to automatic rigging are rig transfer and rig synthesis. Rig transfer starts with a pre-rigged reference model (or a small set of template models) and computes correspondences that define how to transfer the rig to the target geometry. Most previous work has focused on creating skeletal rigs that control the primary motion of a humanoid character by transferring an input template rig to new 3D [22, 25] and 2D [14] output characters. These existing techniques are not suitable for rigging 2D character accessories, which require different types of rig constraints than humanoid characters. In addition, since 2D accessories exhibit such a wide range of variations in shape, appearance, and layer decomposition, it is difficult to produce a small set of pre-rigged template models that cover the variations well enough to compute the necessary correspondences for rig transfer.

An alternative approach is to synthesize rigs directly based on the source geometry, which does not require pre-defined templates. Some methods produce traditional skeletal rigs for 3D characters [4, 35] by analyzing the model geometry and determining where to place joints and bones. In contrast, our work aims to synthesize rigs that constrain accessories to the character's body rather than directly control the character's motion. There are prior techniques that infer geometric constraints between object parts [23, 27], but these methods focus on 3D models of mechanical objects, which is a very different domain from ours. The types of spatial relationships and constraints between mechanical parts are distinct from those between accessories and body parts in layered 2D characters.

2.3 Deformation

Previous work introduces a variety of techniques for deforming 2D and 3D geometry. Many of these methods take user-specified constraints as input (e.g., move a specified point to a specified location) and solve for an overall deformation that satisfies the constraints while preserving attributes like smoothness or similarity to the undeformed rest shape [2, 15, 16]. Our approach leverages such methods as an enabling technology. More specifically, we present accessory rigs that use low-level pin constraints from bounded-biharmonic weights (BBW) [16] to form higher-level constraints that specify how changes to the character's base layers propagate to accessory layers. The main contribution of our work lies in the definition of these higher-level constraints and automated rigging algorithms for applying the constraints to layered artwork so that accessory layers properly adapt to deformations of the body layers.

Our technique differs from prior work that uses deformations to modify character poses. Some previous methods warp an image of a character using a single deformation field to adjust the pose and proportions of the subject [18, 37]. In contrast, our method constrains and propagates deformations across multiple accessory layers, which allows overlapping accessories to deform independently and offers greater flexibility to handle pose changes while avoiding unwanted coupling effects between layers. Related work by Yang et al. [36] supports deformations of multi-layered 2D artwork by connecting pairs of layers using a single rigid link. However, such links do not provide sufficient degrees of freedom to deform accessory layers to match the silhouettes of edited body layers, while our constraints define finer grained relationships between layers.

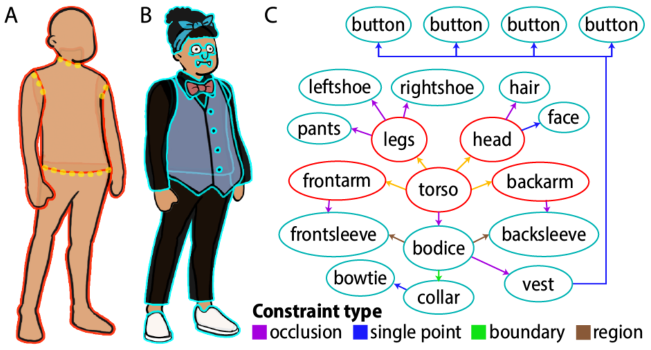


Figure 2: Input artwork for the body layers (A) and accessory layers (B) from the Open Peeps [24] data set. The artwork is organized into a DAG that describes how body layers (red) and accessory layers (cyan) should be constrained to one another. Each edge in the DAG denotes a constraint between a parent layer (arrow tail) and a child layer (arrow head). The color of the edge denotes the type of constraint between layers. The DAG is also annotated with directed edges (orange) that denote adjacent body layers (e.g. head is child of torso). For each pair of adjacent body parts, our tool explicitly maintains the region of spatial overlap that defines where the layers are attached.

3 METHOD

Given a collection of mix-and-matchable layered artwork, our goal is to propagate user-specified modifications of the character's body to all the accessories. Our approach is to encode important spatial relationships between layers with constraints that define how layers should adapt with respect to each other. In particular, we propose four types of constraints that capture common layer-to-layer relationships: (1) occlusion, (2) attachment at a point, (3) coincident boundaries, and (4) overlapping regions. A set of constraints for a given accessory forms a rig that determines how to transfer edits specified on the body to that accessory. We present automated algorithms to generate a rig for every accessory by analyzing the layer geometry and adding the relevant constraints.

3.1 Input Layered Artwork

Our system takes as input a collection of named layers containing all the artwork for the character. The layers are represented as bitmaps, which we automatically convert into textured triangle meshes [28]. As previously mentioned, each layer is either a body layer or an accessory layer. For example, the body layers for the character in Fig 2A include multiple layers (e.g., one for the torso, head, arms respectively) to enable independent control over the shape and pose for each part. Accessory layers (Fig 2B) occlude various body layers. In the simplest case, each accessory is represented as a single layer that is composited on top of the body layer, such as the bow tie. But more complex accessories like the shirt may consist of multiple layers to capture part and occlusion relationships (e.g., the sleeve and collar of the shirt are different layers of the shirt accessory that appear in front of the vest accessory, while the bodice of the shirt is a third shirt layer that appears behind the vest accessory). This layered decomposition of artwork is common for existing mix-and-match tools, since accessories need to be represented as separate, interchangeable assets, and the layers themselves can usually be exported directly from artwork creation tools. Furthermore, accessories are often further separated into multiple layers representing semantically different parts such as the vest buttons in Fig 2B, or the lapels of a jacket.

Finally, users annotate which body layer serves as the root of directed acyclic graph described in Section 3.2 (usually the torso). This creates directed edges between adjacent body layers (dotted orange in Fig 2C, such as between the legs and torso). For each of these edges, our system maintains the region of spatial overlap that defines where the adjacent body layers are attached (e.g., the leg layer is attached to the torso layer at the hip region where the layers overlap). These overlapping regions of geometry are used to disambiguate where to place constraints between accessory layers that may occlude multiple regions, as described in the end of Section 3.3.

3.2 Layer DAG

Given the input layered artwork, our system propagates modifications of the character's body layers to the accessory layers. Doing so requires determining (1) *which* layers should propagate changes to which other layers (e.g., in Fig 2, edits to the frontarm should affect the frontsleeve but not the backsleeve), and (2) *how* the changes should propagate across coupled layers (e.g., as the frontarm gets thicker, the silhouette of the frontsleeve should expand so that it still occludes the arm). Here, we explain how our system addresses the first of these two questions, while Section 3.3 describes the rig constraints that define how changes propagate across layers.

We use a directed acylic graph (DAG) to specify which layers should be constrained to each other. Fig 2C shows the DAG for a simple character with one full set of accessories. Body layers (in red) are always the roots of the DAG. Edges in the DAG specify which pair of layers should be constrained, with the edge direction indicating the child (arrow head) should adapt to changes made in the parent (arrow tail). For example, in Fig 2C, the legs parent layer is connected to the pants child layer, meaning changes in the legs will propagate to the pants. Note that a given layer may have multiple parents. For example, the frontsleeve of the black shirt is constrained to both the frontarm and bodice, since modifications to either of those layers affect the frontsleeve.

Our system automates the DAG creation process as follows. For each accessory layer, we check if it overlaps every other layer (body and accessory). The overlap is calculated as the area of intersection of the pixels of both layers. If just a single layer overlaps, we create an edge from said layer to the accessory (e.g., the face and shoes in Fig 2B). If multiple layers overlap, we consider all layers within some threshold ϵ (in practice, $\epsilon = 15\%$) range of the maximum overlapping area. We then create an edge from the layer that has the closest z-ordering to the accessory layer. Using the z-ordering of layers avoids cycles and helps resolve ambiguity—for instance, in Fig 1G, the bracelet completely overlaps the front arm, shirt, and torso, but is attached to the front arm as that is the closest layer. An evaluation of this algorithm is under each result in Section 4.

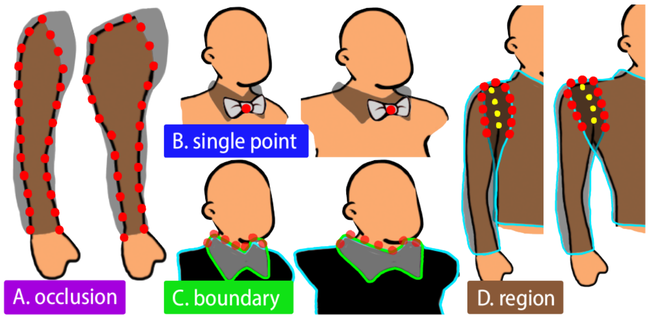


Figure 3: Our four different kinds of constraints between layers, with color labels carried over from Fig 2. For each group, the left is the input rig and the right is the rig under deformation. Occlusion constrains (A) pin two layers along the boundary of the parent layer (arm) when it lies within the child layer (sleeve). The bow tie is constrained at a single point (B) to the shirt collar and translates rigidly with it. Constraints along coincident boundaries (C) like the neckline of the shirt collar and bodice ensure the boundaries remain coincident after deformation. Constraints over a region (D) like between the sleeve and bodice maintain the region's overlap after deformation.

3.3 Rig Constraints

To transfer modifications made on the character's body to the accessories, we must preserve specific relationships between connected layers in the input DAG. For example, given the edit shown in Fig 1C, we must ensure that the shirt, pants and shoes still occlude the relevant body layers in order to avoid the artifacts shown in Fig 1D. We propose four types of constraints that encode common inter-layer relationships. Each constraint "pins" the child and parent layers together at specific points, which constrains how the layers can move relative to each other. By configuring the pins in different ways, our constraints preserve different relationships between the layers.

A. Occlusion. Many of the occlusion relationships between layers are critical to the character's appearance. For example, in Fig 2, the various accessory layers must occlude the relevant body parts (e.g., sleeves occluding arms, shirt bodice occluding the torso, pants and shoes occluding legs) to make it appear as if the character is wearing the clothes. To maintain such occlusions between a child layer (occluder) and a parent layer (occludee), we apply a constraint that places pins along the parent layer's boundary wherever it passes underneath the child layer (Fig 3A). This configuration of pins ensures that changes to the occluded layer's position or shape (which will necessarily transform its boundary) propagate to the occluding layer in a way that prevents any previously occluded portion of the parent layer from becoming visible.

B. Attachment at a Single Point. Some accessory layers represent objects that should move rigidly with their parent layers, such as the bowtie and vest buttons in Fig 2. We encode this relationship with a constraint that simply pins a child layer to its parent layer at a single point. When the parent is modified, the child layer undergoes a rigid body translation based on the position of the pin. To implement this constraint, our system pins the parent and child layers together at the point within the overlapping region that is closest to the centroid of the child (Fig 3B).

C. Coincident Boundaries. Some pairs of accessory layers have coincident boundaries that should always remain coincident. Such boundaries often occur where clothing folds over on itself, as in Fig 2B where the shirt collar and shirt bodice layers align at the neck, or at the folded line between a lapel and jacket. In other cases, layers from different accessories may align visually along a boundary, like the hem line of an untucked shirt matching that of a cardigan. To preserve such alignments, we apply a constraint that pins the child and parent layers together along the coincident boundary (Fig 3C).

D. Overlap over a Region. Similar to coincident boundaries, there are cases where the entire region of overlap between layers should be preserved as the layers are modified. For example, in Fig 2B, the regions where the sleeves overlap with the shirt bodice should remain overlapping even as the sleeves and bodice adapt to changes in the character's body. Another scenario is a ponytail layer that overlaps with the hair layer that sits directly on top of the head. To preserve the overlap between a pair of layers, we apply a constraint that pins the two layers together along the entire boundary of the overlapping region (Fig 3D).

Our system provides automated algorithms for constructing each type of constraint between any given pair of layers. The occlusion, coincident boundary, and overlap constraints all involve placing pins along specific boundaries: for occlusion, we detect all occluded portions of the parent layer boundary; for coincident boundary, we identify portions of the child layer boundary that are within 5px of the parent layer boundary; and for overlap, we extract the boundaries of the overlapping regions between the child and parent layers. We then place pins at regularly sampled locations along the relevant boundaries to constrain the two layers together. For constraints at a single point, we place a pin within the overlapping region of the two layers at the point that is closest to the centroid of the child layer.

In some cases, the overlap constraint can result in unwanted pins that overly restrict the motion of the parent and child layers. Specifically, if the parent and child layers overlap in multiple regions, we may only want to preserve a subset of these overlaps. For example, if a sleeve layer overlaps with a shirt bodice layer near the shoulder and also near the hand where the sleeve passes in front of the bodice, we may only want to preserve the overlap near the shoulder. To handle such cases, we first check whether the two overlapping layers are also connected to adjacent body layers (e.g., overlapping sleeve and bodice layers are connected to adjacent arm and torso body layers, respectively). If so, we only place pins around overlapping regions that are within 20px of the attachment region between the adjacent body parts.

3.4 Automated Rigging

The previous section describes how we automate the application of our four constraint types, but constructing a complete rig for all accessories also requires choosing which constraints to apply for each pair of connected layers in the DAG. We provide an automated rigging algorithm that infers which constraints to apply based on a set of heuristics.

For each DAG edge e, we use the following inference rules:

- If e connects a body layer to an accessory layer, we apply an occlusion constraint, which covers the common cases where an accessory fits over a body part (e.g., sleeve on an arm).
- If e connects two accessory layers, we apply both coincident boundary and overlap constraints, which handles scenarios where accessory layers overlap and/or align along edges.
- The preceding rules may fail to apply any constraints, if we do not detect an occluded parent layer boundary or a coincident boundary or a valid region of overlap. This often occurs for small accessory layers that are connected to larger parent layers with no shared boundaries or nearby attachment regions (e.g., bowtie on torso, button on vest). In such cases, we apply a single point constraint.

While automation provides a convenience and may reduce tedious manual labor, forms of automation that prevent users of visual art systems from having direct aesthetic control over the final results may also prevent them from developing aesthetically refined artifacts [21]. While we provide automated rigging algorithms, we acknowledge a lot of results are a matter of subjective preferences. Thus, the system gives users the freedom to refine the automatic rigs by overriding the inferred choice of constraints. For example, the automatic rig may apply an overlap constraint to a pocket on a shirt, which would cause the pocket to stretch if the character's body gets wider. Users can choose a different constraint type for the pocket to obtain different behavior. Applying a single point constraint between the pocket and the shirt would cause the pocket to stay the same size as the body expands. If they need more fine-grained control over the rigs, users can also click to manually add and delete pins.

3.5 Deforming Accessories

Given a complete set of accessory rigs, our system propagates modifications of the character's body by deforming accessory layers based on the rig constraints. We leverage bounded biharmonic weights (BBW) [16] to implement the constrained deformation. For each rig constraint, we represent the pins as BBW handles whose positions are "written" by the parent layer (i.e., the handles move as the parent deforms) and "read" by the child layer (i.e., the handles define how the child deforms). When the geometry of any body layer changes, we traverse the DAG in topological order and update the BBW handles for every visited constraint, deforming each accessory layer in the process. At the end of the traversal, all accessories will have been deformed to satisfy the rig constraints.

Our approach does not require the body layer modifications to be specified in a particular format. In our examples, we use BBW handles to edit the shape and pose of body layers, but in general, we can support any edit, as long as it is possible to update the pin positions for accessory layers that are constrained directly to the body layers. For instance, if the user redraws a body layer, we could compute correspondences between the original and unmodified layers (e.g., via as-rigid-as-possible image registration [31]) to move the pins.

4 RESULTS

We present results made using our automated rigs across three different mix-and-match data sets: Open Peeps [24] (Figs 1, 4), illustrated faces (Fig 5), and cartoon ponies (Fig 6). These data sets demonstrate a diversity of artwork styles, and they include different kinds of accessory and body layers—Open Peeps highlights clothing accessories on the full body, the illustrated faces focus on swappable facial features and hairstyles, and the cartoon ponies demonstrate how our methods extend to non-humanoid characters. In addition to the static results in the paper and supplemental website, our video shows an interactive interface with high-level sliders for exploring different body shapes (Fig 8), and animations created in our system. Both of these applications leverage our automated rigs to deform accessories dynamically with the body. Furthermore, we conducted informal usability studies with two participants who provided and manipulated their own original character art. Overall, the results demonstrate that our rigs preserve important spatial relationships between artwork layers across a range of edits to the shape and pose of the character.

4.1 Open Peeps

Open Peeps [24] provides a diverse set of mix-and-match artwork, including 13 body poses and over 40 accessories (across shirts, hairstyles, etc.) that can be combined in different ways. We created several modifications to the shape and pose of the predefined Open Peeps bodies, and for each edit applied our rigs to deform all the accessory artwork. Fig 4 shows a subset of the resulting deformed accessories for four such edits. Note that we manually modified the skin tone of some body layers to achieve greater diversity in the appearance of the characters. Our supplemental website includes the full set of results.

For all of the edits, our method preserves the relevant occlusion relationships between the accessory and body layers. For example, when the belly and shoulders expand in Figs 4A–B, the shirts, jackets, and sweaters stretch accordingly. Similarly, the sleeves and pants deform to occlude the arms and legs as the character's pose changes in Fig 4D. In addition, our coincident boundary and single point attachment constraints enable multilayered accessories to deform naturally. For example, the collar of the white shirt in Fig 4B is separate from the underlying shirt bodice, which enables layering sweaters above the bodice and below the collar (as in Fig 4C). As the shirt deforms with the shoulders, our system preserves the shared boundary between the bodice and collar at the character's neckline. The bowtie is also a separate layer that is attached to the underlying artwork via a single point constraint. Note that it moves rigidly with the character in Figs 4C–D.

Our DAG construction algorithm successfully parents most accessories for the desired visual deformations. Since it only creates a single edge per accessory, we have to manually add additional edges parenting the sleeves to the bodice for multi-layer shirts (2 edges out of around 20 per character). Some failure cases also involve specific interactions between accessories; for instance, the short skirt parents to the kimono (instead of the legs) as the long fabric of the kimono overlaps more with the skirt than the legs do, while longer pants and shorter shirts do not have this issue.

4.2 Illustrated Faces

While the Open Peeps results demonstrate the variety of accessories, poses, and kinds of deformations we can support, it does not provide individual variations within a character's face (all facial features are grouped in a single layer, as shown in the DAG in Fig 2C). Many mix-and-match systems (e.g., Picrew [33]) do allow users to combine individual facial features like the eyes and mouth. To evaluate how our rigs work for such artwork, we authored our own mix-and-match data set with various interchangeable facial features.

Fig 5 shows the results for an edit that widens the face. Our constraint method automatically applies occlusion constraints for the hairstyle layers and single point constraints for all facial features. These constraints deform the hair to fit the wider head and move the facial features rigidly to accommodate the extra space within the face (middle row). As an alternative, the user may prefer the facial features to stretch horizontally along with the face. To make this change in our system, the user can select the overlap constraint for the facial features, which updates the rigs and produces the result shown in the bottom row. Our constructed DAG identifies the correct parent layer for every accessory layer except for the back of the blue hair, which it connects to the body instead of the head, as the long hair overlaps more with the body. We manually fix this.

4.3 Cartoon Ponies

Our method can be used to rig accessories for non-humanoid characters. To demonstrate, we created a data set of cartoon ponies with interchangeable wings, manes, and tails inspired by the 3D Pony Creator [26]. Fig 6 shows the results from an example edit that scales down the torso and limbs to create more squat, "chibi-style" body proportions. Our rig constraints produce plausible deformations for the accessories. For example, the brown saddle deforms to fit the shortened body, and the tails and wings move to the appropriate locations in the edited results. Our automated DAG construction algorithm correctly connects 16 out of 17 accessory layers for correct visual effects. It incorrectly connects the hair to the back wing of the pink pony instead of the head, which we manually fix.

4.4 Usability Study

In addition to generating a diverse set of results to evaluate the quality of our automated constraints, we also collected initial qualitative usability feedback from two participants who uploaded and manipulated their own original character art with our tool. Both participants thought our tool was faster than hand redrawing and reported the deformed accessories looked how they expected. One participant (who drew the woman, Fig 7 right) did not change any automated constraints, while the other (who drew the blue animal, Fig 7 left) removed a point of contact where the leaf occluded the eyes which they wanted to be free-flowing. Overall, these initial results suggest our system and simple set of constraints are usable.

5 LIMITATIONS AND FUTURE WORK

While our constraint types cover a broad range of body shape and pose changes on a variety of characters, the main limitation to our approach is that because we rely on deformations to propagate

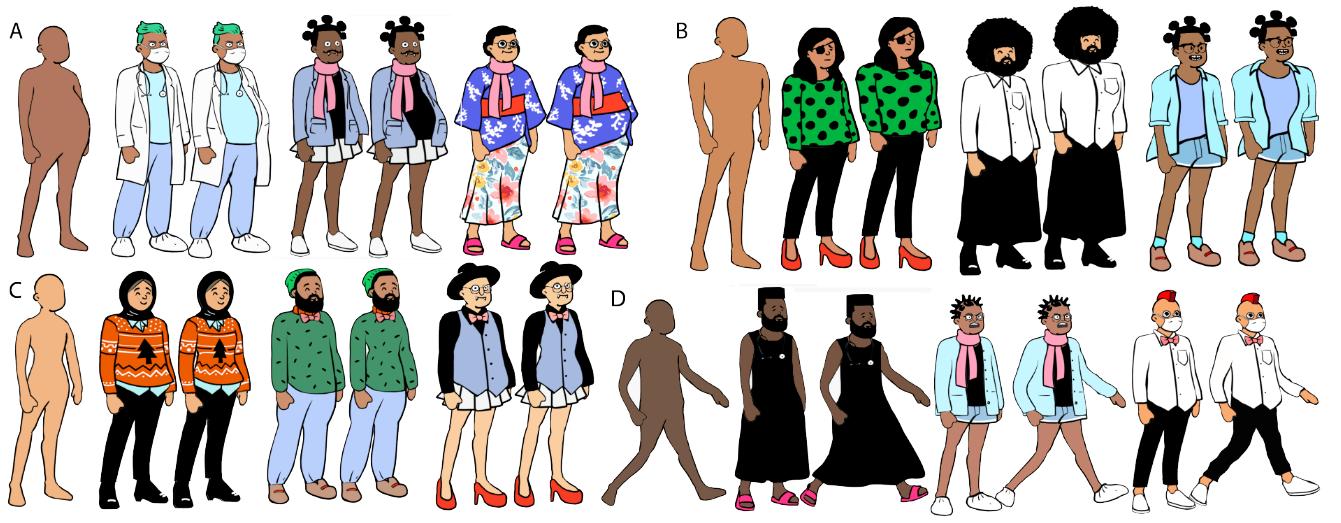


Figure 4: A sample of results from the Open Peeps dataset. For each pair of clothed results, the left is the input while the right is the deformed character. A. Widening the belly. B. Broadening the shoulders. C. Creating an hourglass figure. D. Changing the pose of the limbs. For additional accessories, deformations, and input poses, please see our supplemental materials.

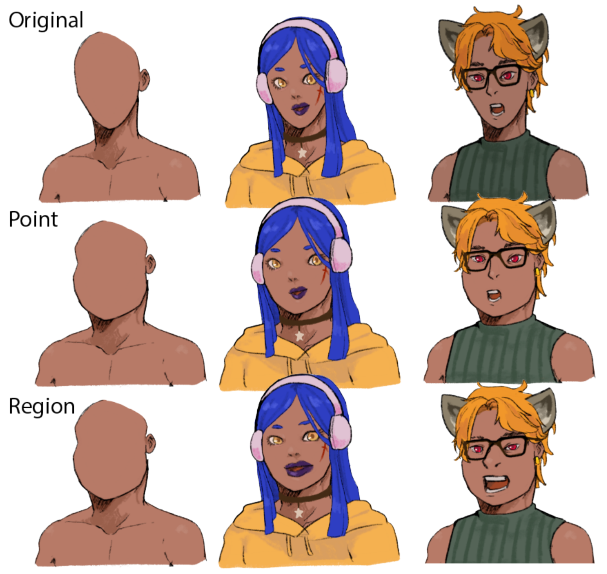


Figure 5: Results from widening the face of a data set emulating Picrew's avatar creation options. Top row: input artwork. Middle row: constrained at a single point, the facial features remain rigid. Bottom row: constrained along their regions of overlap with the face, the facial features deform.

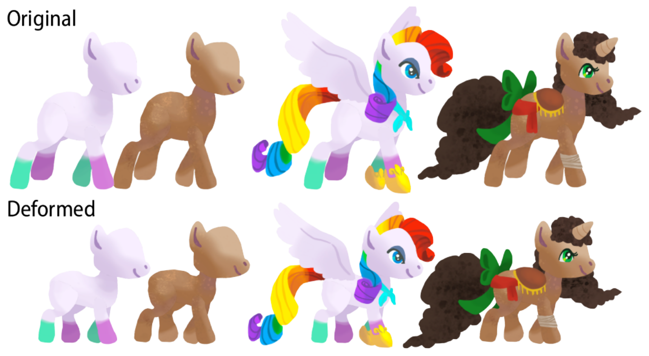


Figure 6: Our rigs also apply to non-humanoid body layers, such as these ponies.

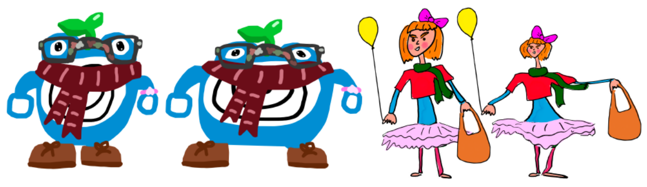
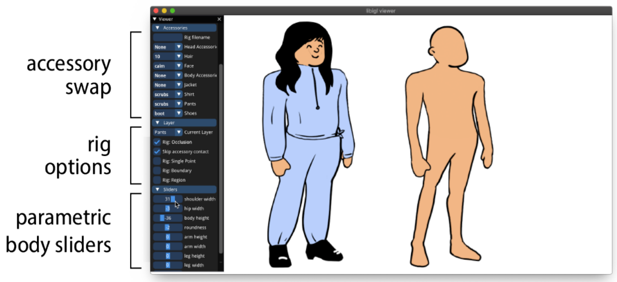


Figure 7: Original artwork created by our participants and then manipulated in our system.

edits, we cannot generate new geometries or textures. Our system does not support motions that are out of plane, such as modifying the body layers to a new pose where the character has been rotated with respect to the viewer. Similarly, textured accessories like the pink tank top with white dots in Fig 1G might show distortion artifacts under deformation. One future direction could address this issue by applying some form of texture synthesis [10] when the accessory undergoes an "extreme" enough deformation.

Our four constraint types are derived from simple geometric heuristics. While they have of the advantage of being relatively easy to implement and understand, our modeling of inter-layer relationships currently lacks any notion of real-world physics. Many accessories are affected by gravity, such as a long ponytail or areas of clothing that drape significantly like an apron or the sleeve in



Rig filename
None Head Accessories
10 Hair
calm Face
None Body Accessories
None Jacket
scrubs Shirt
scrubs Pants
boot Shoes

Pants Current Layer
✓ Rig: Occlusion
✓ Skip accessory contact
Rig: Single Point
Rig: Boundary
Rig: Region

31 shoulder width
-3 hip width
-36 body height
-2 roundness
0 arm height
0 arm width
0 leg height
0 leg width

Figure 8: Our system's user interface, built in libigl [17]. In addition to displaying the character and its body layers, the interface also allows users to mix-and-match accessories, set rig options, and use sliders to explore body deformations at a high level.

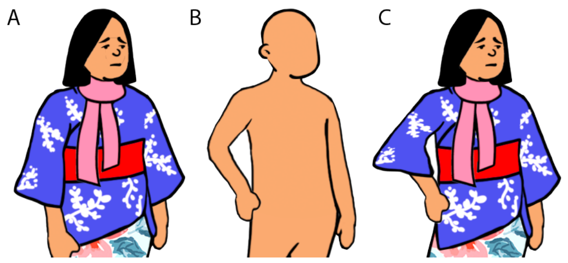


Figure 9: A limitation in our approach is the lack of real world physics. When accessories that should feel the effects of gravity like the kimono sleeve (A) undergo deformation (B), they should maintain their droopiness. Currently, the kimono sleeve protrudes outward (C) instead.

Fig 9. Future work could investigate ways to incorporate the effects of gravity into the appearance of such accessories.

Additionally, our constraint inference heuristics may produce undesired results. For example, the boundary of a jacket pocket may align with the silhouette of the jacket bodice, resulting in a coincident boundary constraint and large deformations to the silhouette could cause the pocket to stretch significantly. Here, users may prefer to manually change to a single point constraint, which preserves the pocket shape.

6 CONCLUSION

This paper presents a method for propagating user-specified modifications of a 2D illustrated character's body to all the accessories. We propose four types of constraints that preserve common relationships between layers: (1) occlusion, (2) attachment at a point, (3) coincident boundaries, and (4) overlapping regions. These constraints staple layers together to ensure outcomes like occlusions staying occluded after deformation. We provide an algorithm on how to automate both the creation of such constraints and how to infer them through heuristics from layer geometry. Applying a set of these constraints as a rig for each accessory layer, we show how they enable modifying a wide variety of artwork, across different input body layers, deformations, and kinds of accessories; we also demonstrate how users may choose to change the kinds of constraints applied for different visual effects. We hope our methods enable another dimension of customization—changing body shape and pose—while leveraging the existing collection of accessories in mix-and-match character customization interfaces towards more expressive visual communication.

ACKNOWLEDGMENTS

The authors would like to thank the character designers and study participants interviewed; Kevin Wampler and Alec Jacobson for their insightful conversations about this work; and Eric Rawn for his assistance in generating the supplemental materials. A Brown Institute for Media Innovation Magic Grant supported this work.

REFERENCES

[1] Adobe. 2021. *Character Animator.*
[2] Adobe Photoshop. 2010. *Puppet Warp.*
[3] Autodesk. 2021. *Maya.*
[4] Ilya Baran and Jovan Popović. 2007. Automatic Rigging and Animation of 3D Characters. *ACM Trans. Graph.* 26, 3 (July 2007), 72–es. https://doi.org/10.1145/1276377.1276467
[5] Blender Foundation. 2021. *Blender.*
[6] Blush Design. 2021. *Blush.* https://blush.design/
[7] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. 2011. Probabilistic Reasoning for Assembly-Based 3D Modeling. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 35, 10 pages. https://doi.org/10.1145/1964921.1964930
[8] Siddhartha Chaudhuri and Vladlen Koltun. 2010. Data-Driven Suggestions for Creativity Support in 3D Modeling. *ACM Trans. Graph.* 29, 6, Article 183 (Dec. 2010), 10 pages. https://doi.org/10.1145/1882261.1866205
[9] Marek Dvorožňák, Daniel Sýkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. 2020. Monster Mash: A Single-View Approach to Casual 3D Modeling and Animation. *ACM Trans. Graph.* 39, 6, Article 214 (Nov. 2020), 12 pages. https://doi.org/10.1145/3414685.3417805
[10] Alexei A. Efros and William T. Freeman. 2001. Image Quilting for Texture Synthesis and Transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 341–346. https://doi.org/10.1145/383259.383296
[11] Frederic Guimont. 2020. *The Character Creator.* https://charactercreator.org/
[12] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. 2004. Modeling by Example. In *ACM SIGGRAPH 2004 Papers* (Los Angeles, California) *(SIGGRAPH '04)*. Association for Computing Machinery, New York, NY, USA, 652–663. https://doi.org/10.1145/1186562.1015775
[13] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. 2008. Real-Time Motion Retargeting to Highly Varied User-Created Morphologies. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 1–11. https://doi.org/10.1145/1360612.1360626
[14] Alexander Hornung, Ellen Dekkers, and Leif Kobbelt. 2007. Character Animation from 2D Pictures and 3D Motion Data. *ACM Trans. Graph.* 26, 1 (Jan. 2007), 1–es. https://doi.org/10.1145/1189762.1189763
[15] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. 2005. As-Rigid-as-Possible Shape Manipulation. *ACM Trans. Graph.* 24, 3 (July 2005), 1134–1141. https://doi.org/10.1145/1073204.1073323
[16] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. 2011. Bounded Biharmonic Weights for Real-Time Deformation. *ACM Trans. Graph.* 30, 4, Article 78 (July 2011), 8 pages. https://doi.org/10.1145/2010324.1964973
[17] Alec Jacobson and Daniele Panozzo. 2017. libigl: prototyping geometry processing research in C++. In *SIGGRAPH Asia 2017 courses*. 1–172.
[18] Arjun Jain, Thorsten Thormählen, Hans-Peter Seidel, and Christian Theobalt. 2010. MovieReshape: Tracking and Reshaping of Humans in Videos. *ACM Trans. Graph.* 29, 6, Article 148 (Dec. 2010), 10 pages. https://doi.org/10.1145/1882261.1866174
[19] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. 2012. A Probabilistic Model for Component-Based Shape Synthesis. *ACM Trans. Graph.* 31, 4, Article 55 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185551
[20] Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. Motion Amplifiers: Sketching Dynamic Illustrations Using the Principles of 2D Animation. In *Proceedings of the 2016 CHI Conference on Human Factors in*

Computing Systems (CHI '16). Association for Computing Machinery, New York, NY, USA, 4599–4609. https://doi.org/10.1145/2858036.2858386

[21] Jingyi Li, Sonia Hashim, and Jennifer Jacobs. 2021. What We Can Learn From Visual Artists About Software Development. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/110.1145/3411764.3445682

[22] Christian Miller, Okan Arikan, and Don Fussell. 2010. Frankenrigs: Building Character Rigs from Multiple Sources. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Washington, D.C.) *(I3D '10)*. Association for Computing Machinery, New York, NY, USA, 31–38. https://doi.org/10.1145/1730804.1730810

[23] Niloy J. Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. 2010. Illustrating How Mechanical Assemblies Work. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 58, 12 pages. https://doi.org/10.1145/1833349.1778795

[24] Pablo Stanley. 2020. *Open Peeps*. https://www.openpeeps.com/

[25] Martin Poirier and Eric Paquette. 2009. Rig Retargeting for 3D Animation *(GI '09)*. Canadian Information Processing Society, CAN, 103–110.

[26] PonyLumen. 2021. *Pony Creator*. https://ponylumen.net/games/3d-pony-creator/

[27] Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-amorn, and Wojciech Matusik. 2014. Design and Fabrication by Example. *ACM Trans. Graph.* 33, 4, Article 62 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601127

[28] Jonathan Richard Shewchuk. 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Workshop on Applied Computational Geometry*. Springer, 203–222.

[29] Snap Inc. 2021. *Bitmoji*. https://www.bitmoji.com/

[30] Minhyuk Sung, Hao Su, Vladimir G. Kim, Siddhartha Chaudhuri, and Leonidas Guibas. 2017. ComplementMe: Weakly-Supervised Component Suggestions for 3D Modeling. *ACM Trans. Graph.* 36, 6, Article 226 (Nov. 2017), 12 pages. https://doi.org/10.1145/3130800.3130821

[31] Daniel Sýkora, John Dingliana, and Steven Collins. 2009. As-Rigid-as-Possible Image Registration for Hand-Drawn Cartoon Animations. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering* (New Orleans, Louisiana) *(NPAR '09)*. Association for Computing Machinery, New York, NY, USA, 25–33. https://doi.org/10.1145/1572614.1572619

[32] Kenshi Takayama, Ryan Schmidt, Karan Singh, Takeo Igarashi, Tamy Boubekeur, and Olga Sorkine. 2011. GeoBrush: Interactive Mesh Geometry Cloning. *Computer Graphics Forum* (2011). https://doi.org/10.1111/j.1467-8659.2011.01883.x

[33] TetraChroma Inc. 2021. *Picrew*. https://picrew.me/

[34] Nora S. Willett, Wilmot Li, Jovan Popovic, Floraine Berthouzoz, and Adam Finkelstein. 2017. Secondary Motion for Performed 2D Animation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 97–108. https://doi.org/10.1145/3126594.3126641

[35] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. 2020. RigNet: Neural Rigging for Articulated Characters. *ACM Trans. Graph.* 39, 4, Article 58 (July 2020), 14 pages. https://doi.org/10.1145/3386569.3392379

[36] Wenwu Yang, Jieqing Feng, and Xun Wang. 2012. Structure Preserving Manipulation and Interpolation for Multi-element 2D Shapes. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2249–2258.

[37] Shizhe Zhou, Hongbo Fu, Ligang Liu, Daniel Cohen-Or, and Xiaoguang Han. 2010. Parametric Reshaping of Human Bodies in Images. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 126, 10 pages. https://doi.org/10.1145/1833349.1778863

[38] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Renjiao Yi, and Hao Zhang. 2018. SCORES: Shape Composition with Recursive Substructure Priors. *ACM Trans. Graph.* 37, 6, Article 211 (Dec. 2018), 14 pages. https://doi.org/10.1145/3272127.3275008