ToonCap: A Layered Deformable Model for Capturing Poses From Cartoon Characters

Xinyi Fan
Princeton University

Amit H. Bermano
Princeton University

Vladimir G. Kim
Adobe Research

Jovan Popović
Adobe Research

Szymon Rusinkiewicz
Princeton University

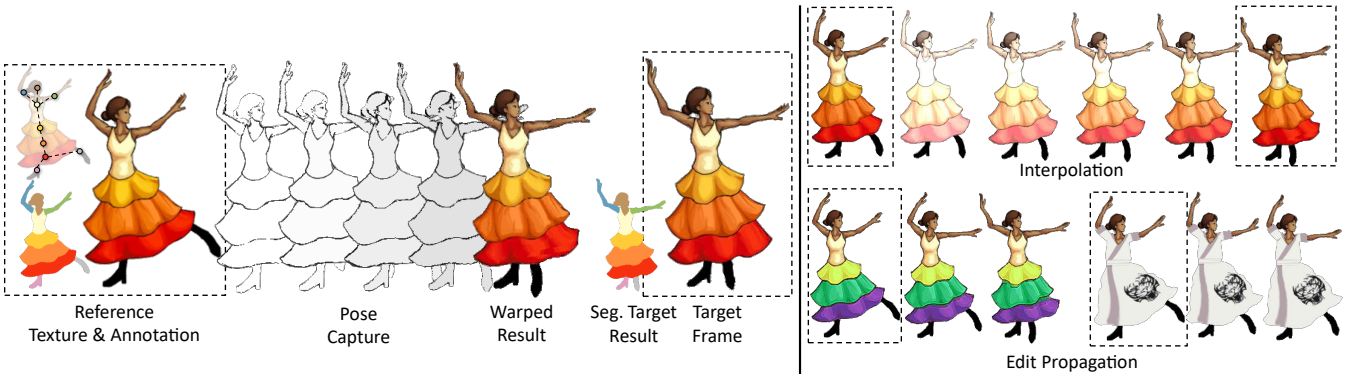

Figure 1: Given an input puppet constructed from a reference frame with annotated layers, joints, and handles, we capture poses in novel unlabeled images by registering the puppet to each image. This enables editing other frames by altering the puppet's artwork and synthesizing novel animations by interpolation. We demonstrate the utility of this approach by animating static characters from books and by re-targeting motion to new characters (see Section 7). Dashed boxes mark user input.

ABSTRACT

Characters in traditional artwork such as children's books or cartoon animations are typically drawn once, in fixed poses, with little opportunity to change the characters' appearance or re-use them in a different animation. To enable these applications one can fit a consistent parametric deformable model — a *puppet* — to different images of a character, thus establishing consistent segmentation, dense semantic correspondence, and deformation parameters across poses. In this work, we argue that a *layered* deformable puppet is a natural representation for hand-drawn characters, providing an effective way to deal with the articulation, expressive deformation, and occlusion that are common to this style of artwork. Our main contribution is an automatic pipeline for fitting these models to unlabeled images depicting the same character in various poses. We demonstrate that the output of our pipeline can be used directly for editing and re-targeting animations.

Expressive '18, August 17–19, 2018, Victoria, BC, Canada

ACM ISBN 978-1-4503-5892-7/18/08…$15.00
https://doi.org/10.1145/3229147.3229149

CCS CONCEPTS

• **Computing methodologies** → **Animation**;

KEYWORDS

character animation, correspondence, segmentation, registration

ACM Reference Format:
Xinyi Fan, Amit H. Bermano, Vladimir G. Kim, Jovan Popović, and Szymon Rusinkiewicz. 2018. ToonCap: A Layered Deformable Model for Capturing Poses From Cartoon Characters. In *Expressive '18: The Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering, August 17–19, 2018, Victoria, BC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3229147.3229149

1 INTRODUCTION

Cartoon characters are a popular genre of art, with examples spanning graphical novels, animated movies, and illustrations in children's books. Beginning from cartoons' hand-drawn roots, recent advances in digital modeling have simplified content creation and re-use. For instance, if an animation is created with a posable digital character, one can easily add new visual elements to change its shape or texture (Figure 1, right). One can also transfer its motion to novel posable characters with compatible rigs.

Unfortunately, most artist-created data does not come in this consistently-parameterized form. Artwork typically consists of

characters that are hand-drawn independently in each pose, providing a high-quality final result, but offering little opportunity for re-use. *The main motivation for our work is to enable artists and amateurs to re-use existing artwork by fitting a consistent parametric representation to a collection of unlabeled images.* Using such a system, static characters from cartoon strips and illustrations in children's books can come to life, by interpolating between available static poses.

Existing techniques for cartoon registration use either a single-layer deformable model [Sýkora et al. 2009a], which is limited in its ability to represent the piecewise-rigid motion and self-occlusion common in articulated characters, or a multi-layer rigid model [Zhu et al. 2016], which fails to match more expressive artwork. In contrast, we pick a *layered deformable puppet* as our parametric model. While this model is more general (and subsumes previous work as special cases), it is also much more difficult to fit, and our main technical contribution is an automatic and robust method for registering layered deformable puppets to input images.

Our model (Figure 1, extreme left) is defined by a hierarchy of layers that are connected via joints and can be deformed by handle manipulation. *Layers* are textured meshes that represent meaningful character parts (e.g., head, torso, arms, and legs for a biped). They are organized in a parent-child hierarchy and are sorted by depth. *Handles* are control points on each layer associated with rigid-body transformations. Interpolating the handle transformations enables the characters to deform. *Joints* are fixed points at which a positional or rotational constraint is enforced between a layer and its parent. This model naturally captures piecewise-rigid articulations as well as free-form deformations by simply adjusting the number of handles per layer and their degrees of freedom. It also enables our method to model self-occlusions via layering, a common technique used in traditional animation.

Fitting a layered deformable puppet to a set of input frames proceeds in two stages. The first stage, consisting of fitting the puppet to a single reference frame, leverages a small amount of user input together with an adaptation of existing segmentation and rigging tools. The second stage, however, involves solving for the pose of every frame other than the reference, which requires jointly reasoning about semantic correspondence, segmentation, deformation, and occlusion. To tackle this challenge, we observe that the layered deformable puppet model provides effective priors for plausible deformations, allowing us to develop a novel discrete-continuous optimization that jointly solves for these parameters in an interleaved manner. We also demonstrate that results can be further improved by optimizing across all frames simultaneously instead of matching the puppet to each frame independently.

We evaluate our method on a benchmark constructed by animating puppets using state-of-the-art digital puppeteering tools. We show that our deformable layered model provides the best accuracy, in comparison to existing non-layered or piecewise-rigid alternatives. We further demonstrate the utility of our approach for altering the content of existing animations, such as textures and shapes, as well as retargeting animations to novel characters and animating characters from static media, e.g., children books.

2 RELATED WORK

Digital Models for Character Animation. Many models have been developed for digital character animation, which saves the artist from having to draw each frame of an animation sequence. These techniques typically require the artist to create one deformable puppet, which is further animated by manually prescribing deformation parameters for the key poses, and then automatically interpolating between these parameters to create the final animation. These puppets are typically represented as triangular or tetrahedral meshes with associated artwork. The user prescribes deformations at a small set of handles (e.g., skeleton joints or control points), which are further automatically extrapolated to the rest of the shape [Jacobson et al. 2011]. Significant effort has gone into producing deformations that preserve geometry of a reference shape (e.g., [Igarashi et al. 2005; Poranne and Lipman 2014; Chien et al. 2016]) or multiple references [Wampler 2016], and are fast to compute (e.g., [Jacobson et al. 2012; Wang et al. 2015]).

While modeling 2D characters can be as easy as triangulating a 2D sketch, deforming the entire character as a single mesh poses challenges in modeling occlusions. Several techniques have been proposed to address this limitation, including multi-layered characters [Catmull 1978], assigning depth [Sýkora et al. 2010] or local layering order [McCann and Pollard 2009] to different mesh regions, and leveraging 3D proxies [Jain et al. 2012; Sýkora et al. 2014]. In this work we focus on multi-layered representations, since they effectively model occlusions, enable artists to prescribe complex relationships between layers [Willett et al. 2017], and are commonly supported by commercial tools [Adobe 2018]. We leverage these recent advances in digital puppets and demonstrate that deforming a layered puppet to images of the same character in different poses enables applications in editing and creating animations.

Animating Digital Characters Based on Artwork. Even with a deformable puppet, defining key poses is challenging for users who lack an animation background. One can borrow motions from motion capture data [Hornung et al. 2007]; however, they are usually limited to skeletal motions of real humans and thus lack the expressiveness and variation of cartoon animations. To address this limitation, several methods have been proposed to capture and transfer deformations from cartoon data [Bregler et al. 2002; de Juan and Bodenheimer 2006], enabling more expressive animations. They typically focus on cartoon videos and leverage existing tools [Shi and Tomasi 1994; Wirtz et al. 2004] to track the motion. Sýkora et al. [2009a] proposed a more general image registration technique for cartoon characters, alternating between correspondence estimation and as-rigid-as-possible mesh deformation to align the character to the target frame. In this work, we propose a similar approach for layered deformable puppets. To leverage layering, we incorporate a segmentation step into our pipeline. We also design our approach to be more reliable in the presence of strong deformations and higher variance in character depiction: we leverage multiple features to estimate sparse correspondence, guide our correspondence with layer and segmentation information, reinforce self-consistent correspondences within a frame and across multiple frames, and jointly optimize for regularization and correspondence

alignment. Our comparisons demonstrate that this pipeline is better at handling stronger deformations, occlusions, and multi-layer relationships.

Layered Models for Cartoon Analysis. Traditional cel animation has motivated layer-aware methods in cartoon analysis [Petrović et al. 2000; Corrêa et al. 1998]. Zhang et al. [2011] proposed EXCOL, a method for decomposing cartoons into foreground and background layers. Their model mostly assumes rigid motions, with the exception of foreground characters, which they register as a single deformable mesh using the method of Sýkora et al. [2009a]. Zhu et al. [2016] further propose a global optimization for segmentation and piecewise-rigid registration of segments, and demonstrate that their approach provides higher accuracy than EXCOL. Due to the success of these techniques, we sidestep the foreground/background segmentation problem and focus on improving the fitting of foreground characters. We make improvements including allowing per-layer deformation, maintaining joint constraints, and solving for highly-accurate deformation parameters aligning each layer of the puppet to each frame in the animation. This allows us to explore applications such as re-targeting motion, editing layer shapes, and animating static frames in children books.

3 LAYERED DEFORMABLE MODEL

Our layered deformable puppet model is designed to provide a flexible framework that can accommodate both rigid-body and deformable motion. Conceptually, a puppet is initialized from a single reference frame, using a small amount of manual annotation. The model is then fit onto all other available frames of the character, where the deformation parameters are automatically inferred. As part of this process, areas occluded in the reference frame are filled in from other frames.

In this section, we introduce our layered deformable puppet framework and explain what input is needed to initialize it from a reference frame. Section 4 describes how we solve the problem of deforming the reference frame to best match another frame, effectively fitting the pose of that frame to the puppet. We find that fitting all frames at once can lead to better results, and we describe this "global" optimization in Section 5.

Puppet Model. A puppet $P(\Theta)$ is parameterized by deformation parameters Θ, which define the final textured geometry. The puppet is composed of a tree-structured hierarchy of multiple layers $\mathcal{L} = \{L_1, ..., L_{|\mathcal{L}|}\}$, where each layer L_l is assigned a depth d_l, a mesh M_l, a set of deformation handles H_l, and joints J_l. Assigning a single integer depth d_l per layer offers a simple model for occlusion, which we nevertheless found sufficient for all the examples presented in this paper. More complex self-occlusion could be accommodated by having a scalar per-point and per-frame depth function.

The handles defining the character's deformation are defined per-layer, $H_l = \{h_{l,1}, ..., h_{l,|H_l|}\}$. This choice simplifies the deformation model by constraining the influence of each handle to a single corresponding segment, allowing us to control the amount of deformation in each layer independently (e.g., a dress might be floating freely, while the limbs are mostly piecewise-rigid). Each handle $h_{l,i}$ corresponds to a vertex of the mesh and is assigned a transformation defined by 3 parameters: $[\theta, t_x, t_y]$, where θ is the angle of rotation, and t_x, t_y is translation. Note that multiple handles can lead to non-rigid deformations. The union of all per-handle transformations defines the deformation parameters Θ. We propagate these transformations to the rest of the mesh vertices on that layer via linear blend skinning, with per-vertex weights assigned using the method of Jacobson et al. [2012].

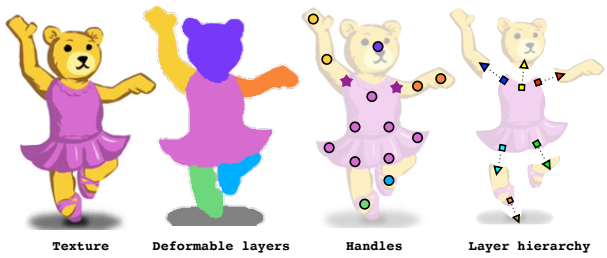

Figure 2: Our puppet is composed of a textured image decomposed into deformable layers, with annotated handles (circles) and joints (stars), organized into a parent-child hierarchy.

Joints J_l are used to model the kinematic structure of articulated characters. Each joint consists of a pair of corresponding points, one on the layer itself and one on its parent. The interpretation is that the child is constrained to have the same position (hinge joint) or position and rotation (welded joint) as the parent layer at the corresponding point. We assume that parent-child relationships are consistent and organize our layers into a tree, which prescribes the order in which deformations are estimated.

Finally, each mesh also has a corresponding texture image I_l along with an alpha channel A_l, specifying visibility. Figure 2 illustrates the parameters of our layered deformable model.

Initialization. To initialize our puppet we ask the user to annotate a single reference frame denoted by f_0. The user segments the reference frame using standard techniques (e.g. [Boykov et al. 2001; Sýkora et al. 2009b]), and we use Triangle [Shewchuk 2002] to create a mesh for each layer M_l. The user then clicks on handles and joints, and prescribes the depth ordering and parent-child relationships between the layers.

4 CAPTURING THE POSE OF A FRAME

Given a deformable puppet $P(\Theta)$ initialized from a reference frame f_0, our goal is to capture the pose of the puppet in a new frame f_i, by optimizing deformation parameters Θ_i to maximize similarity between $P(\Theta_i)$ and f_i. Note that the presence of multiple frames can improve the accuracy of this method, as discussed in Section 5.

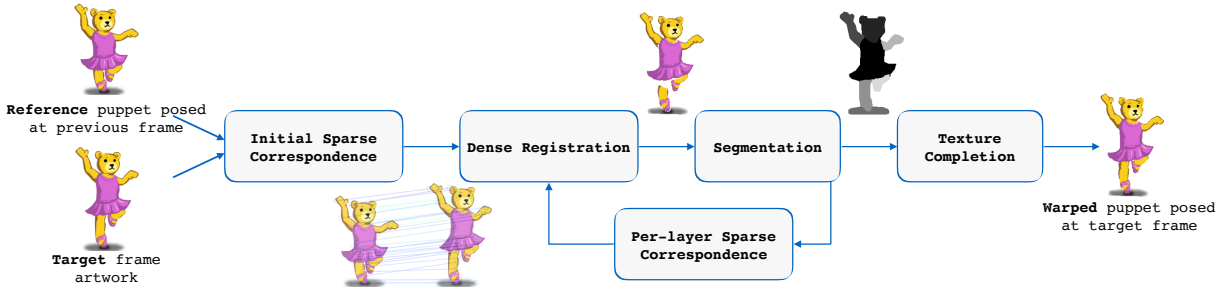

Figure 3: Our fitting pipeline begins with the puppet posed at the previous frame, together with a new target frame. It finds initial (sparse) correspondences (Section 4.1), and uses them to bootstrap dense registration (Section 4.2) and segmentation for layer assignment (Section 4.3). These stages are iterated until convergence, and then any disoccluded regions are used to fill the texture in the puppet (Section 4.4). The result is the puppet posed at the new frame.

The main challenge in fitting a deformable puppet to a single frame is that the initial puppet $P(\Theta_0)$ might appear to be substantially different from the appearance in the target frame, and thus naive gradient descent over $\|P(\Theta_i) - f_i\|$ is prone to getting stuck in local minima. To remedy this, our approach leverages both sparse correspondences that are robust to strong deformation and dense correspondences that enable us to estimate deformation parameters Θ_i precisely. In the presence of layering, it is also important to reason about visible parts and segments in the target frame.

We propose an iterative pipeline that first finds a sparse set of correspondences between the reference and target frames (Subsection 4.1), and then uses these correspondences to initialize and regularize an optimization to find a dense alignment (Subsection 4.2). This dense alignment is further used to reason about the segmentation of the target frame (Subsection 4.3), and this segmentation in turn is used to improve correspondences in the next iteration, by constraining them to corresponding layers (see Figure 3).

4.1 Sparse Correspondence

Given initial deformation parameters Θ_0, we treat the resulting deformed puppet $f_0 = P(\Theta_0)$ as an image and estimate a sparse image-to-image correspondence to the frame f_i. As shown in Figure 3, the same method will later be used to compute layer-to-layer correspondences. Cartoons, unlike natural images, often lack textures, which makes it difficult to rely on a single type of feature. We therefore design a method that leverages multiple weak cues such as color, texture, and outline similarity, and then use a voting scheme to select a larger set of self-consistent correspondences.

Feature Detection and Matching. We first detect sparse feature points $Q_j \subset f_j$, which include corners [Harris and Stephens 1988], blobs [Lowe 2004], and uniform samples on detected edges [Canny 1986]. We then extract several feature descriptors for each point, capturing the local texture and shape:

- **Color histogram** is computed within a circular neighborhood around each keypoint to make it rotation-invariant (96 features, 32 bins × 3 colors).
- **SIFT** [Lowe 2004] provides cues about texture similarity (128 features).
- **Shape context** [Belongie et al. 2002] provides cues about outline similarity. We modify it to be rotation-invariant by applying a 1D Fourier transform within each radius bin and discarding phase [Zahn and Roskies 1972] (72 features, 6 radius bins × 12 angle bins).
- **Location offset** from the center of mass of the character encourages rough spatial alignment (2 features).

We concatenate the descriptors into one vector, and normalize each dimension to have zero mean and unit variance (across all available images for multi-frame analysis). We denote the resulting descriptor space as $\mathcal{F} : Q_i \to \mathbb{R}^{|\mathcal{F}|}$. Note that although this similarity metric leverages multiple local cues, it is not sufficient to establish reliable correspondences for cartoons, where the absence of shading and lighting cues yields very similar-looking local patches (see Figure 7, left).

Correspondence Aggregation with Voting. For reliable sparse matching, we leverage local consistency cues, so that correspondences that agree with other nearby correspondences are treated as more likely to be reliable. This is still not trivial because puppets undergo non-rigid deformations, thus they are unlikely to have a single low-dimensional transformation that aligns all the regions. Inspired by techniques for aligning 3D non-rigid shapes, we employ a voting scheme that casts votes proportionally to the fraction of the puppet that aligned well [Lipman and Funkhouser 2009]. The voting routine needs several decisions: which feature points to use to define a transformation to align two shapes, how to compute the vote score, and which pairs of points receive the vote.

We pick a random quadruplet of points $q_1, q_2 \in Q_0$ and $p_1, p_2 \in Q_i$, where $\mathcal{F}(q_i)$ and $\mathcal{F}(p_i)$ are *mutually-closest* points in feature space and q_1, q_2 are not too far or too close in image space ($\delta_l < \|q_1 - q_2\|_2 < \delta_u$). We use L_2 distance metric over both feature space: $\|\mathcal{F}(q) - \mathcal{F}(p)\|_2$ and image space: $\|q - p\|_2$. We set $\delta_l = 2\%$, $\delta_u = 10\%$ of character's bounding box diameter. Based on these points, we find the similarity transformation $\mathcal{T}$ such that $p_i = \mathcal{T} q_i$.

Next, we estimate the fraction of puppet that was aligned well under this transformation. Due to vast texture-less regions we only consider alignments between compatible feature points. For each point $q \in Q_0$ we define a compatible set $\mathcal{N}(q) \subset Q_i$ as the K nearest points in feature space. For each pair of compatible points $q, p \in \mathcal{N}(q)$, we define their post-alignment proximity weight (clamped to the $[0, 1]$ interval):

$$w_{\mathcal{I}}(q, p) = \text{clamp}\left(\frac{\delta_u - \|\mathcal{T} q - p\|_2}{\delta_u - \delta_l}, [0, 1]\right). \quad (1)$$

We now measure weighted distances in image and feature space, summed across all feature points:

$$D_{\mathcal{I}} = \frac{1}{Z} \sum_{q \in Q_0, p \in \mathcal{N}(q)} w_{\mathcal{I}}(q,p) \|p - \mathcal{T}q\|_2, \quad (2)$$

$$D_{\mathcal{F}} = \frac{1}{Z} \sum_{q \in Q_0, p \in \mathcal{N}(q)} w_{\mathcal{I}}(q,p) \|\mathcal{F}(p) - \mathcal{F}(q)\|_2, \quad (3)$$

where Z is the normalizing factor: $Z = \sum_{q \in Q_0, p \in \mathcal{N}(q)} w_{\mathcal{I}}(q,p)$. We finally define the vote score:

$$w_{\mathcal{T}} = e^{\left(-D_{\mathcal{I}}^2/\sigma_{\mathcal{I}}^2\right)} e^{\left(-D_{\mathcal{F}}^2\sigma_{\mathcal{F}}^2\right)}. \quad (4)$$

Where $\sigma_{\mathcal{I}}$ is 1% of the character's diameter and $\sigma_{\mathcal{F}}^2$ is 10% of the feature vector's length. We cast the vote for every pair of compatible points ($q \in Q_0, p \in \mathcal{N}(q)$), weighted by post-alignment proximity $w_{\mathcal{I}}(q,p)$, and store it in a correspondence matrix $\omega_c \in \mathbb{R}^{|Q_0| \times |Q_i|}$. We cast 8000 votes for a pair of frames. After voting, we estimate the final corresponding points as $corr(q) = \arg\max_{p \in \mathcal{N}(q)} \omega_c(q,p)$. We summarize this process in Algorithm 1, and demonstrate the impact of the voting in Figure 7.

Algorithm 1 Correspondence voting

Input: feature keypoints sets Q_0, Q_i and feature descriptors $\mathcal{F}$.
  for iterations = 1, ... N **do**
    Sample random quadruplet (q_1, q_2, p_1, p_2) s.t. $\mathcal{F}(q_i), \mathcal{F}(p_i)$ are mutually closest and $\delta_l < \|q_1 - q_2\|_2 < \delta_u$
    Compute transformation $\mathcal{T}$ s.t. $p_i = \mathcal{T}q_i$
    Compute vote $w_{\mathcal{T}}$ ▷ Equation 4
    for all pairs $(q \in Q_0, p \in \mathcal{N}(q))$ **do**
      $\omega_c(q,p) \leftarrow \omega_c(q,p) + w_{\mathcal{T}} \cdot w_{\mathcal{I}}(q,p)$ ▷ Cast votes
    end for
  end for
  Set $corr(q) = \arg\max_{p \in \mathcal{N}(q)} \omega_c(q,p)$

4.2 Dense Registration

The sparse correspondences estimated with the voting provide opportunity to relate even strongly deformed regions, however, they do not consider the puppet rig, and thus might yield unnatural warps if used directly. Moreover, since we assume that the target frame can be produced with a deformation of the puppet, besides coarse correspondence at feature points, we expect dense pixel-wise consistency to provide us with a better guidance once we are close to a good alignment. Thus, we formulate dense alignment as a continuous optimization with respect to free handle parameters Θ minimizing an objective that takes all priors into account:

$$E(\Theta) = \tau_{\text{corrs}} E_{\text{corrs}} + \tau_{\text{color}} E_{\text{color}} + \tau_{\text{joint}} E_{\text{joint}} + \tau_{\text{arap}} E_{\text{arap}}, \quad (5)$$

E_{corrs} favors alignment of sparse correspondences, E_{color} favors dense pixel-wise alignment, and $E_{\text{joint}}, E_{\text{arap}}$ are regularization terms to ensure that the articulated puppet structure is preserved. We now describe these terms in more details.

The sparse correspondence term is based on weighted image-space distance:

$$E_{\text{corrs}} = \sum_{q \in Q_0} \omega_c\left(q, corr(q)\right) \|P(\Theta, q) - corr(q)\|_2^2, \quad (6)$$

where $P(\Theta, q)$ is the location of feature point q after puppet is warped with handle parameters Θ.

To favor accurate pixel-wise alignment we measure the color similarity between pixels in the texture of the source layer and the pixel that it aligned to after the warp:

$$E_{\text{color}} = \sum_{x \text{ s.t.} A_{l,0}(x)=1} E_{\text{pixel}}(x), \quad \text{where} \quad (7)$$

$$E_{\text{pixel}}(x) = \begin{cases} \|I_{l,0}(x) - I_{l,i}\left(P(\Theta, x)\right)\|_2^2 & \text{if} \quad A_{l,i}(x) = 1, \\ I_{\max} & \text{if} \quad A_{l,i}(x) = 0. \end{cases} \quad (8)$$

We use 3D RGB vectors to compare pixel values in the [0...255] range. Note that we only consider pixels with opaque alpha in the source, and we assign penalty $I_{\max} = 255$ if they match to empty regions in the target. In the first iteration, we do not have layer-wise segmentation of the target frame, and thus both source and target are treated as a single layer.

We treat joints as soft constraints, which makes our approach less sensitive to puppet initialization and improves convergence (since joint constraints play smaller role until the puppet is coarsely aligned). We represent joints with a set of pairs $(j^p, j) \in J$, where j^p is a point on parent layer and j is a point on a child layer. We measure the error based on distances between these parent and child joints under the puppet deformation:

$$E_{\text{joint}} = \sum_{(j^p, j) \in J} \|P(\Theta, j^p) - P(\Theta, j)\|_2^2 \quad (9)$$

For welded joints, we estimate the entire transformation matrix at joint positions, and measure distance between vectors that include translation and rotation: $[x, y, \theta]$.

Finally, we favor preservation of the reference puppet shape using as-rigid-as-possible (ARAP) deformation energy [Sorkine and Alexa 2007]:

$$E_{\text{arap}} = \sum_{l=1..|\mathcal{L}|} \sum_{f \in M_l} \lambda_f^2 + \lambda_f^{-2} + \gamma_f^2 + \gamma_f^{-2} \quad (10)$$

where f is a face on a mesh M_l, and λ, γ are singular values of a deformation gradient at face f induced by $P(\Theta)$.

To optimize for parameters Θ, we use a gradient descent-based method [Agarwal et al. 2018] to minimize $E(\Theta)$. We use the built-in autodiff function for efficient gradient computation. We also leverage the layer structure, as we optimize one layer at a time, keeping the parent layer fixed for joint energy. Finally, since the color term is not expected to be reliable in early iterations (before coarse shape is matched well), we gradually increase its influence over the course of iterations. In particular we set $\tau_{\text{corrs}} = \tau_{\text{arap}} = \tau_{\text{joint}} = 1$, $\tau_{\text{color}} = 3it$, where it is the iteration number.

4.3 Segmentation for Layer Assignment

Understanding the layer structure of the target frame is important to properly reason about occlusions and correspondences. On the other hand, segmenting the target into layers requires correspondence as guidance. To address this inter-dependency, we employ an

alternating optimization strategy. As discussed above, we start by estimating coarse and dense correspondence, treating both shapes as a single layer. We use this information to segment the target frame, and restrict the correspondence computation to the corresponding regions, as dictated by the resulting segmentation.

Note that the segmentation limits the solution space of the correspondence estimation, leading to poor results if one commits to inaccurate region divisions too early. To mitigate this problem, we propose a two-stage process. First, we produce a hard segmentation, using the common graph cut approach. Then, we create a *soft segmentation* by assigning multiple labels to the same pixel. This enables the correspondence algorithms to map points between several viable options.

Hard Segmentation. The dense registration provides strong cues for segmentation, but it does not address texture boundaries in the target frame. Thus, we formulate a graph cut problem [Boykov et al. 2001], with unary terms which are derived from dense correspondence, and binary terms from the target image.

To define the unary term $E_u(p,l)$, we observe that if a pixel $p \in f_i$ is of label l, p and its immediate surroundings are likely to match in color when compared to their counterparts in $P(\Theta_i)$. Based on this observation, we compute a confidence score $\alpha_l(p)$ for every non-empty pixel in each target frame layer, considering only the relevant puppet layer. The score is computed as the color difference within a window of radius n around the pixel p (where correlating empty and non-empty pixels receive a maximum difference penalty). Of course, in $P(\Theta_i)$ some pixels from different layers can overlap, yielding a few layers with high confidence scores for the same region. To resolve these ambiguities, we give higher priority to upper layers. We compute the confidence $\alpha_{l_j}(p)$ from the top most layer l_1 to the back most one $l_{|\mathcal{L}|}$, according to:

$$E_u(p,l_j) = w_{\text{unary}} \cdot \left(1 - \text{clamp}_{[0,1]}\left(\sum_{k=1}^{j-1} \alpha_{l_k}(p)\right)\right) \cdot \alpha_{l_j}(p). \quad (11)$$

In other words, we reduce the confidence of back layers, if the confidence of frontal ones is high. In all our experiments, we set $w_{\text{unary}} = 1$, and as previously described:

$$\alpha_l(p \in f_i) = \frac{1}{Z_n(p)} \sum_{q \in L_l,\, \|P(\Theta,q)-p\|_2 \le n} \|f_0(q) - f_i(P(\Theta,q))\|, \quad (12)$$

where $Z_n(p)$ is the number of pixels in the kernel, and $n = 5$ in all our experiments.

To define pairwise potentials we evaluate how smooth the texture is in a particular region (i.e., how similar a pixel p is to a neighboring pixel q). Since cartoons often have thick outlines around layers, we incorporate two adaptations: we down-weigh penalties near edge pixels and introduce longer-range connections (with penalty scaled according to the distance). More specifically, given a pair of points p, q, we define the pairwise penalty if two points are within $n = 3$ pixels of each other, and received different labels:

$$E_{\text{pair}}(p,q) = w_{\text{pair}} \exp\left(\frac{\|f_i(p) - f_i(q)\|_2^2}{\sigma_c^2}\right) \cdot \exp\left(\frac{\max\{C(p),C(q)\}^2}{\sigma_e^2}\right) \cdot \exp\left(\frac{\|p-q\|_2^2}{\sigma_d^2}\right), \quad (13)$$

where $C(p)$ is the probability that p lies on a contour [Canny 1986]. In all our experiments, we set $w_{\text{pair}} = 7, \sigma_c = 0.5, \sigma_e = 1$, and $\sigma_d = 1$. The final graph cut optimizes the following energy over labelings of points L:

$$E(L) = \sum_{p \in I_i} E_u(p, L(p)) + \sum_{\substack{(p,q) \in I_i, \text{ s.t.} \\ L(p) \neq L(q), \text{ and} \\ \|p-q\| < n}} E_{\text{pair}}(p,q). \quad (14)$$

Multi-Label Segmentation. To mitigate the effects of inaccurate labeling in challenging cases, we utilize an inclusive approach that does not limit the following iterations to the hard choices made by the segmentation step. We compute layer-wise and point-wise confidences, and form masks for every layer, $A_{l,i}$, which include the initial segmentation augmented by regions that may have been segmented incorrectly, and are plausible candidates. The insight is that the upcoming steps benefit from the additional options much more than they stand to lose from them.

We compute the point-wise confidence as defined in Equation 12. We also generate a confidence value in the other direction — measuring how well f_i matches $P(\Theta_i)$ instead of how well $P(\Theta_i)$ matches f_i:

$$\alpha_l(q \in f_0) = \frac{1}{Z_n(q)} \sum_{\substack{p \in f_i,\, L(p)=l, \\ \|P^{-1}(\Theta_i,p)-q\|_2 \le n}} \|f_0\left(P^{-1}(\Theta_i,p)\right) - f_i(p)\|, \quad (15)$$

where P^{-1} maps pixels from the target frame to the reference frame according to the deformation prescribed in Θ_i. The layer-wise confidence is then computed as the average of the pixel-wise confidence values, in a bi-directional manner:

$$\alpha(l) = \frac{1}{Z(l,0)} \sum_{q \in f_{l,0}} \alpha_l(q) + \frac{1}{Z(l,i)} \sum_{p \in f_i,\, L(p)=l} \alpha_l(p), \quad (16)$$

where $Z(l,i)$ measures area of the layer l in frame i.

Based on this, we set $A_{l,i}(p) = 1$ if $L(p) = l$, or if the pixel is of low confidence and is close enough to the label l, i.e. $\alpha(p) < 0.75$ and $d(p,L) < \exp\left(-\frac{\alpha(l)}{\sigma_{\text{ml}}}\right)$. $d(p,L)$ is the minimum distance between the point p and any of the non-empty pixels in L, and $\sigma_{\text{ml}} = 0.25$ in all our experiments. The process is demonstrated in Figure 4.

4.4 Texture Completion

As described in Section 3, each layer of our puppet is assigned a depth order to handle occlusion between layers. During deformation, it is possible that gaps will appear at layer boundaries, exposing regions that were occluded in the reference frame. We exploit these gaps to expand our layered representation, and complete the texture in regions of a puppet's layer that were previously unknown. To perform this *texture completion* operation, one must solve two problems: determining which regions in the source need

Figure 4: Generating a soft layer mask for the dancer's right shoe. From left: The blended image includes the warped reference and the target frame. Incorrect correspondences were generated at this iteration, causing the hard segmentation to incorrectly place the right leg in the same layer as the dress. However, the confidence of this segmentation is low, so the right leg receives an expanded mask allowing correct correspondence in the next iteration.

to be textured and assign appropriate texture. To answer the former, we find pixels in the target frame that are not covered by the warped source, and were not seen in any frame prior to the current one. Formally, given a pixel that satisfies both $p \in f_i$ and $p \notin P(\Theta)$, we examine $P^{-1}(\Theta, p)$ on the layer $L(p)$. For each pixel in every frame, we store whether it has been viewed (i.e, not occluded by more frontal layers) in any of the frames so far. If $P^{-1}(\Theta, p)$ has not been seen before, we assign the color from the target frame $f_i(p)$ to pixel $P^{-1}(\Theta, p)$ updating the puppet texture $I_{L(p),0}$, and mark it as textured. See the Bear in Figure 11.

5 MULTI-FRAME GLOBAL OPTIMIZATION

While the "pairwise" pose capture algorithm described in Section 4 can be directly used to align the puppet to every frame, it lacks important context that can be provided by multiple frames. In particular, for animation sequences, neighboring frames are likely to have small relative deformations, and this observation can be leveraged to provide a good initial guess for the subsequent fitting. In addition, joint reasoning across multiple frames enables us to regularize based on cycle consistency (i.e., if a sequence of deformations returns to the original frame, concatenating all the deformations should yield the identity).

Fitting Order. The convergence of our iterative fitting procedure can be made more robust if an initial guess is available. In other words, when fitting a puppet to some target frame f_i, we make the process easier by starting with an existing fit to some frame $P(\Theta_j)$ that is as similar as possible to f_i, instead of always beginning with the reference frame $P(\Theta_0)$. This initialization affects all steps of the fitting process, including computation of feature points and descriptors. To decide on the order in which frames are fit, we first construct a graph where each image is a vertex and edge weights are L_2 distances computed between all pairs of images. We compute a minimum spanning tree of the graph, rooted at the reference frame (see Figure 5) to define the fitting order. We always use the puppet warp parameters computed at the parent within this tree to define the initial state for the fitting to the child frame.

Synchronization for Cycle-Consistency. An effective regularization commonly used in multi-frame correspondence estimation is

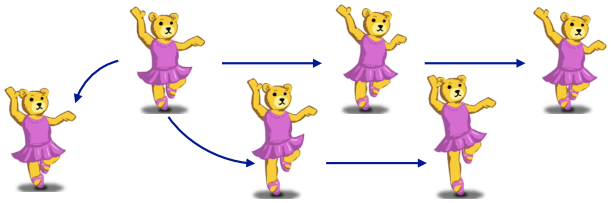

Figure 5: Spanning tree constructed among the frames of an animation, based on L_2 image differences.

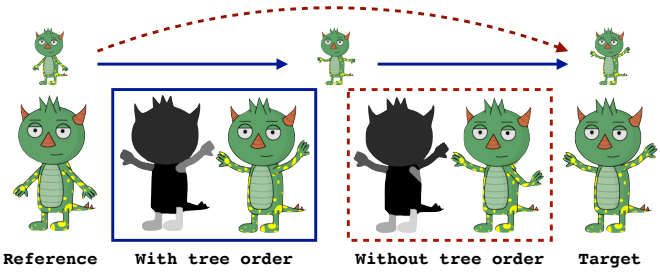

Figure 6: Following the computed fitting order (blue path, top) yields a successful pose capture (blue box, bottom). On the other hand, trying to match the reference (left) to the target (right) directly results in a poor local minima (bottom, red dashed box).

cycle consistency. It is based on the common-sense observation that a cyclic path of mapping frames should be the identity. To formalize this observation, we compute a sparse pairwise correspondence matrix $\omega_c^{i,j}$, as described in Section 4.1 for all pairs of frames, and concatenate these matrices into one matrix $\Omega \in \mathbb{R}^{|\mathbb{Q}|\times|\mathbb{Q}|}$, where Q is the set of all feature points from all frames. Next, we project the matrix Ω onto its low-rank approximation $AA^T \approx \Omega$, using an off-the-shelf non-negative matrix factorization [Solomon et al. 2016]. This matrix is used to produce a new sparse correspondence matrix $\omega_c^{\text{improved}}$ This step can be performed every time any of the pairwise frame-to-frame correspondences are updated, but we found this to be unnecessary; we update only at the first iteration for each new frame.

6 EVALUATION

In this section we evaluate the performance of our system on various types of characters and deformations, and compare our method to the state of the art. See the Appendix (Table 1) for information about the characters we consider, as well as run-times.

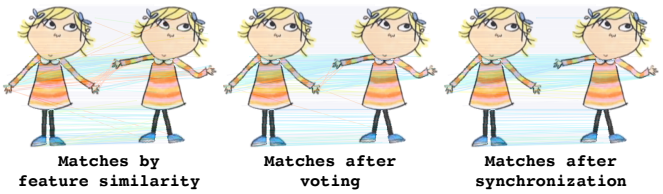

Figure 7: Matches on two poses of Lola [Child 2013] are depicted as lines. Accurate correspondences are washed out, while inaccurate ones are colored from most confident (blue) to least (red).

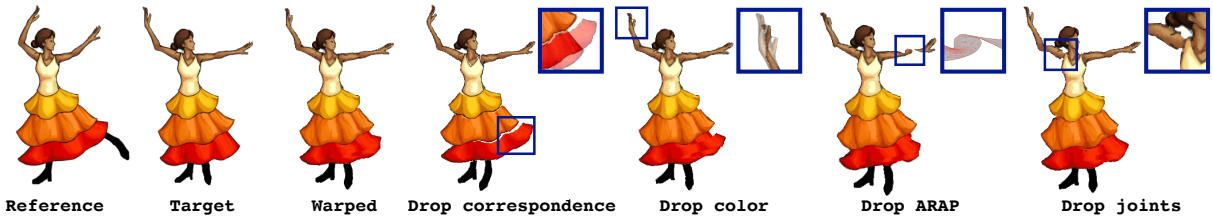

Figure 8: The effects of the different energy terms in Equation 5. From left to right: the Reference pose is deformed to match the Target. The full energy term yields the Warped results. For every term dropped, we show the deformed result (with zoomed-ins), laid over the target frame for comparison. For the ARAP term, we visualize the deformed mesh (due to flipping)

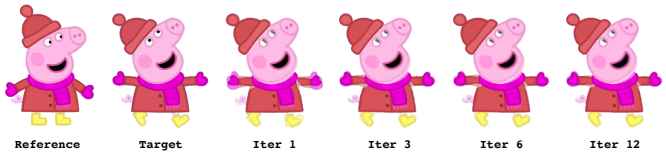

Figure 9: The reference frame (left) is warped to match the target frame over 12 iterations. The result of each iteration is laid over the target frame for comparison. [Astley Baker Davies 2003]

6.1 System Components Evaluation

We first evaluate the effect of our central modules on the end result quality. Namely we examine the sparse correspondence and dense registration modules, and their iterative interaction. The other parts of our system are either well established (e.g. graph-cut formulation for image segmentation), or their importance has already been demonstrated in previous sections (e.g. Figures 4 and 6).

Sparse correspondences. The sparse correspondence step gives the system robustness to large deformations. In Figure 7, we visualize the matches obtained in a controlled experiment, with ground truth correspondences available. This allows us to wash out perfect matches from the visualization, and focus on the relatively few inaccurate ones. As can be seen on the left, our raw feature descriptor similarity yields accurate estimations for most cases. In the middle image, we demonstrate the effect of local aggregation through voting (Algorithm 1) (e.g. some bad correspondences in the hair are resolved). Large similar regions are not handled well by this process, for example, points on the right palm being matched to the left one. On the right, we take advantage of multiple frames through synchronization, and resolve these last issues.

Dense Registration terms. Next, we carry out an ablation study to test the performance of each term in our objective function in Equation 5. Figure 8 demonstrates the issue introduced by dropping each term, respectively. As can be seen, ignoring the sparse correspondences causes the optimization to fall into local minima, while dropping the color term reduces local fine-scale accuracy. It is further demonstrated how dropping the ARAP term causes extreme deformations and overlaps. Finally it can be observed how the joints are necessary for enforcing articulated behavior.

Iterations. Lastly we demonstrate how alternating between the continuous optimization of the deformation parameters and the discrete optimization of the sparse correspondence and segmentation improves the quality of the results. As can be seen in Figure 9, from iteration to iteration the deformed puppet matches the (overlaid) target better and better, especially visible in the limb.

6.2 Pose Capture with Various Character Types

We evaluate the expressiveness of our model by demonstrating the range of deformations it can handle, including articulation, non-rigid deformation, occlusion, and their combinations. We also evaluate how the amount of deformation and texture affects performance. Many results in this section are presented as rows of four images, where each row includes the puppet reference pose, the target frame, the warped puppet, and the segmentation of the target frame. Note that the former two are considered as inputs, while the latter are generated.

Articulation. Our deformable layered model naturally handles various kinds of articulations. Joints ensure that the kinematic structure of the character is preserved. For example, one can easily model simple characters used in low-budget TV shows, such as Peppa Pig (Figure 10), where motions have an almost piecewise-rigid cutout style. Note how Peppa's head is rotating independently from the body without introducing any distortion to the scarf, which would be hard to achieve if the character were modeled with a single mesh (see comparison in Section 6.3 below).

Deformation. At the extreme opposite end of the spectrum from articulated motion lies free-form deformation, which is also common for some characters. For example, the Ghost in Figure 10 cannot be handled with a simple kinematic model, and thus we model its body with several evenly-distributed handles, allowing expressive deformations.

Mixed Deformation. Most real characters exhibit a mixture of various deformations. For example, the Wilk and Fox animations shown in Figure 10 have arms that move near-rigidly in some poses, but bend more significantly in others. The Snowman animation undergoes nonrigid deformation of the body, but the arms are articulated through the use of multiple segments per arm. The Dancer mostly has a traditional kinematic structure, but her dress is composed of multiple non-rigid layers. These mixed models are easily accommodated within our framework by giving the user the flexibility to select the most appropriate rig, including the number of layers and the number of handles and joints per layer.

Occlusion. One of the key advantages of our layered model is that it naturally deals with self-occlusion. Our method can handle a previously visible part being occluded (see the Fox in Figure 11). However, an even more challenging case is when parts get revealed,

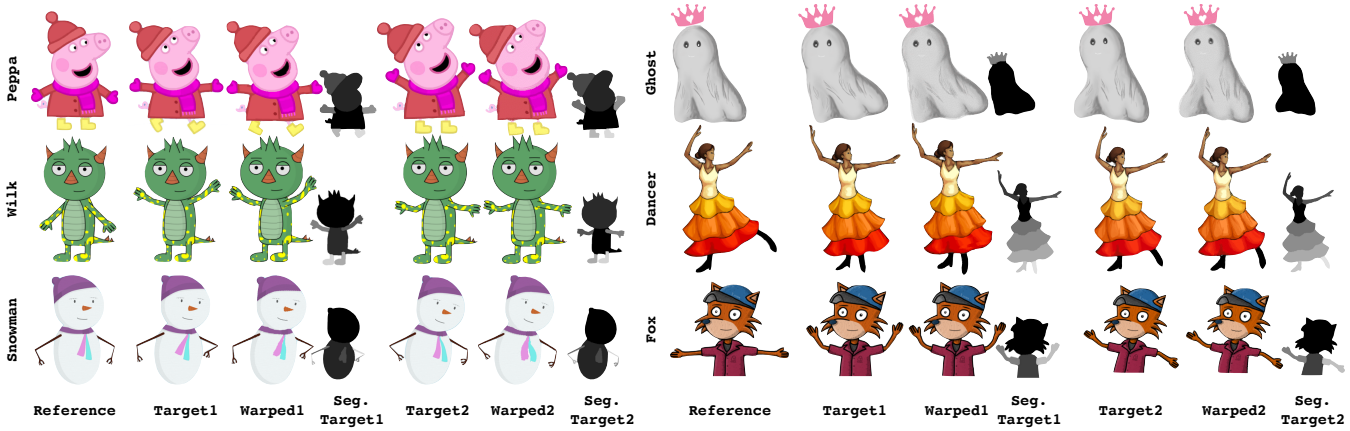

Figure 10: Our model naturally accommodates characters with multiple deforming layers. For every demonstrated example, the Reference pose is warped to match two input Target frames. The Warped result is depicted, along with the computed Segmentation of the target frame. (Peppa Pig [Astley Baker Davies 2003])

Figure 11: Our system handles occlusion across layers, when parts are occluded (Fox) and revealed (Bear).

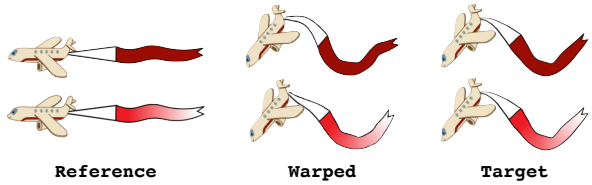

Figure 12: With little variability in texture, our system may have difficulty in computing correspondences (top). Adding texture or color variation improves quality (bottom).

which is common since it is often impossible to find a reference frame in which all parts are visible. Our texture completion enables us to properly fill the missing texture, as demonstrated by the Bear in Figure 11.

Amount of Texture. Feature descriptors are expected to struggle for images containing less texture and strong contour deformation, as happens for the Airplane banner (Figure 12). Indeed if we paint the banner with a constant color (top), the quality of the alignment decreases. In contrast, adding some color variation to the banner helps the alignment (bottom).

6.3 Comparisons

We now compare our method to previous relevant techniques. We first compare directly to a single-layer ARAP-based registration method [Sýkora et al. 2009a] (which is the most similar method to ours). We then demonstrate the disadvantages of existing multi-layered models that use rigid transformations per layer.

Cartoon Registration via a Single Deformation. We compare our result to the cartoon registration technique of Sýkora et al [2009a], by running both methods on the same pair of reference and target images (Figure 13). Since their method operates on a mesh constructed from a regular grid, we test it with both coarser and finer grids. As discussed before, a non-layered model is not expressive enough to represent some animations (e.g., the Dancer's leg moves independently under the dress, which cannot be modeled with a single layer). Their correspondence estimation technique can also struggle for stronger deformations, especially if regions lack texture or good alignment of features within the grid cells (see the misalignment on Peppa's foot).

Piecewise-Rigid Layered Model. Two recent techniques for tracking cartoons from video sequences use layered models. Note that their goals are different, as they aim to decompose the entire sequence into foreground and background layers (instead of decomposing a single character). EXCOL [Zhang et al. 2011] uses the method of Sykora et al. [2009a], as discussed in the previous paragraph, to track foreground characters, and thus suffers from similar shortcomings in the context of our problem. GOTT [Zhu et al. 2016] treats layers as independent segments that can be aligned with a rigid transformation across multiple frames. However, their method cannot directly leverage an input puppet. It only outputs consistent segmentations of frames, but does not provide dense registration. To demonstrate this model, we simulate a piecewise-rigid model in our framework by restricting each layer to have only a single handle.

Figure 14 shows a comparison between rigid and nonrigid deformation. We see that for models that undergo significant deformation, such as the Airplane, the rigid model is obviously too limiting. However, even for characters such as the Bear, for which the deformation might at first appear to be mostly rigid, much of the expressiveness is conveyed through subtle deformations and secondary motion, such as in the shape of the dress. The piecewise-rigid model cannot capture these effects.

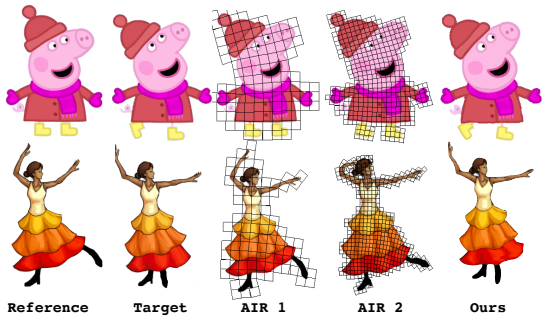

Figure 13: Comparison with two resolution configurations of Sýkora et al. [2009a]. The single-layer-based method cannot handle extreme variation in deformation, e.g. local rotation (such as the foot of Peppa [Astley Baker Davies 2003]), and sliding (such as the Dancer's foot). To enable a fair comparison, no intermediate frames were used for our results.

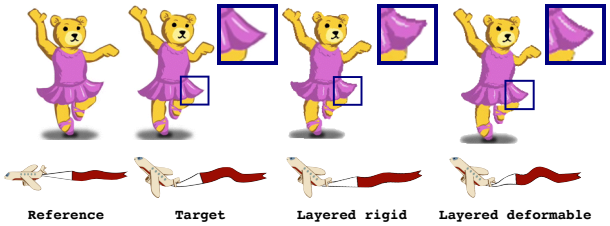

Figure 14: Comparison of rigid vs. deformable (ours) layers. Even for modest deformation (top), a rigid model fails to capture these important subtleties. The airplane's banner (bottom) requires significant deformation.

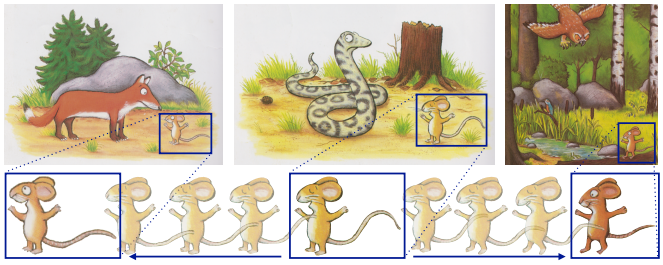

Figure 15: By fitting our puppet to different character poses in a kids' book [Donaldson and Scheffler 1999], we can animate static characters by interpolating deformations.

7 APPLICATIONS

Our dense registration of cartoon characters can be used in a wide range of applications for altering and re-purposing cartoon artwork. We prototype a few examples in this section.

Animating Children's Books. Many children's books have the same character in various poses. The ability of our method to handle strong deformations enables us to densely register all appearances of a character and produce in-betweens. This may be used to automatically animate digitized copies of otherwise static classics (see Figure 15 and supplemental video).

Figure 16: Animation retargeting. The captured Fox sequence (top) is used to animate (bottom) the Lola character [Child 2013], which has a similar handle layout.

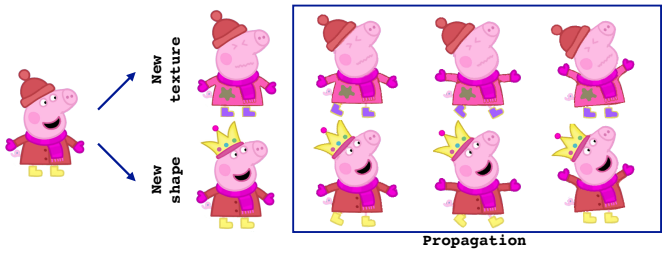

Figure 17: Edit propagation. The reference frame (left) is altered, and the change is propagated to the rest of the captured sequence. (Peppa Pig [Astley Baker Davies 2003])

Motion Transfer. As motivated by Bregler et al. [2002], we can re-use professionally-created motions and apply them to new puppets. In this case, the user needs to prescribe the correspondence between handles and layers of the source and target puppets, and we directly transfer transformations between corresponding handles (see Figure 16).

Layer Editing. Dense registration across frames in a cartoon video can also be used for effective content editing. Instead of redrawing every frame, the artist only needs to update the puppet's textures or layer shapes, and the edits are automatically propagated to the entire set of frames (Figure 1 and 17). While texture edits are trivial to accommodate, changing the shapes of layers is less trivial, since portions of each target frame may need to be filled in with new content. To find pixels that must be filled in, we identify pairs of points $(p, q) \in f_0$ that stem from different layers, are adjacent in f_0, and are not adjacent in $P(\Theta)$. The domain spanned by such consecutive pairs identifies the region to be filled.

8 DISCUSSION

We propose a method for dense registration of deformable layered characters. Our method estimates per-pixel correspondence and deformation parameters for a puppet while aligning it to unlabeled images. We demonstrate that our method can effectively handle articulations, free-form deformations, and occlusions, and provides a better model for cartoon registration than existing alternatives. Finally, we demonstrate its use for applications such as animating children's books, editing of video sequences, and motion transfer.

Even though we use a combination of multiple descriptors that consider color, texture, and shape, as well as local aggregation,

we nevertheless observe difficulty in establishing correspondences between hand-drawn characters, especially with little texture.

Results for more examples are demonstrated in the supplemental video, along with basic animations between fitted frames (via interpolation of the computed deformation parameters). While some artifacts can be observed in our animations, most of these are caused by imperfect texture filling (e.g. Peppa's armpits, Dancer's dress layers, Wilk's armpits and Charlie's collar and fingertips). State-of-the-art texture synthesis techniques, e.g. [Fišer et al. 2017], can be incorporated to improve the visual quality.

In the future, we would like to combine our system with automatic foreground/background segmentation using existing cartoon tracking techniques. Another venue for future work is to infer the puppet rig (including layering and handle placement) automatically based on observed deformations and correspondences across images. Choosing an appropriate parametric model to explain underlying data without over-fitting is a long-standing problem in machine learning, and it would be interesting to extend it to the domain of cartoons. These two enhancements would open the door to fully-unsupervised cartoon analysis, enabling indexing, editing, and re-using the plethora of unorganized cartoon data available via free online video streaming resources. We believe that this holds tremendous potential in democratizing visual storytelling.

ACKNOWLEDGEMENTS

We would like to thank all the people who have provided helpful suggestions, encouragement, and feedback for this project, particularly Nora Willett, Wilmot Li, Kevin Wampler, Daniel Sýkora, Marek Dvoroznak, Haichao Zhu, and the members of the Princeton Graphics Group. We also thank all the reviewers for their constructive comments. This work is supported by NSF grants CHS-1617236 and IIS-1421435.

REFERENCES

Adobe. 2018. Adobe Character Animator. https://www.adobe.com/products/character-animator.html

Sameer Agarwal, Keir Mierle, and Others. 2018. Ceres Solver. https://code.google.com/p/ceres-solver/

Astley Baker Davies. 2003. Peppa Pig. Astley Baker Davies Ltd / Entertainment One UK Ltd. http://www.peppapig.com/

S. Belongie, J. Malik, and J. Puzicha. 2002. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Trans. PAMI* 24, 4 (April 2002), 509–522.

Yuri Boykov, Olga Veksler, and Ramin Zabih. 2001. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. PAMI* 23, 11 (Nov. 2001), 1222–1239.

Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. 2002. Turning to the Masters: Motion Capturing Cartoons. *ACM Trans. Graphics* 21, 3 (July 2002), 399–407.

John Canny. 1986. A Computational Approach to Edge Detection. *IEEE Trans. PAMI* 8, 6 (Nov 1986), 679–698.

Edwin Catmull. 1978. The Problems of Computer-Assisted Animation. In *Proc. SIGGRAPH*. 348–353.

Edward Chien, Renjie Chen, and Ofir Weber. 2016. Bounded Distortion Harmonic Shape Interpolation. *ACM Trans. Graphics* 35, 4, Article 105 (July 2016).

Lauren Child. 2013. Charlie and Lola. Tiger Aspect Productions Ltd. https://www.tigeraspect.co.uk/productions/animation-childrens/charlie-and-lola-2/

Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, and Adam Finkelstein. 1998. Texture Mapping for Cel Animation. In *Proc. SIGGRAPH*. 435–446.

Christina N. de Juan and Bobby Bodenheimer. 2006. Re-using Traditional Animation: Methods for Semi-Automatic Segmentation and Inbetweening. In *Proc. Symposium on Computer Animation*. 223–232.

Julia Donaldson and Axel Scheffler. 1999. *The Gruffalo.* Macmillan Children's Books.

P.D. Eastman. 1960. *Are You My Mother?* Random House.

Jakub Fišer, Ondřej Jamriška, David Simons, Eli Shechtman, Jingwan Lu, Paul Asente, Michal Lukáč, and Daniel Sýkora. 2017. Example-Based Synthesis of Stylized Facial Animations. *ACM Trans. Graphics* 36, 4, Article 155 (July 2017).

Chris Harris and Mike Stephens. 1988. A Combined Corner and Edge Detector. In *Proc. Alvey Vision Conference*. 147–151.

Alexander Hornung, Ellen Dekkers, and Leif Kobbelt. 2007. Character Animation from 2D Pictures and 3D Motion Data. *ACM Trans. Graphics* 26, 1, Article 1 (Jan. 2007).

Takeo Igarashi, Tomer Moscovich, and John F. Hughes. 2005. As-Rigid-As-Possible Shape Manipulation. *ACM Trans. Graphics* 24, 3 (July 2005), 1134–1141.

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. 2012. Fast Automatic Skinning Transformations. *ACM Trans. Graphics* 31, 4, Article 77 (July 2012).

Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. 2011. Bounded Biharmonic Weights for Real-Time Deformation. *ACM Trans. Graphics* 30, 4, Article 78 (July 2011).

Eakta Jain, Yaser Sheikh, Moshe Mahler, and Jessica Hodgins. 2012. Three-Dimensional Proxies for Hand-Drawn Characters. *ACM Trans. Graphics* 31, 1, Article 8 (Feb. 2012).

Yaron Lipman and Thomas Funkhouser. 2009. Möbius Voting for Surface Correspondence. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28, 3, Article 72 (Aug. 2009).

David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60, 2 (Nov. 2004), 91–110.

James McCann and Nancy Pollard. 2009. Local Layering. *ACM Trans. Graphics* 28, 3, Article 84 (July 2009).

Lena Petrović, Brian Fujito, Lance Williams, and Adam Finkelstein. 2000. Shadows for Cel Animation. In *Proc. SIGGRAPH*. 511–516.

Roi Poranne and Yaron Lipman. 2014. Provably Good Planar Mappings. *ACM Trans. Graphics* 33, 4, Article 76 (July 2014).

Jonathan Richard Shewchuk. 2002. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Comput. Geom. Theory Appl.* 22, 1-3 (May 2002), 21–74.

Jianbo Shi and Carlo Tomasi. 1994. Good Features to Track. In *Proc. CVPR*. 593–600.

Justin Solomon, Gabriel Peyré, Vladimir G. Kim, and Suvrit Sra. 2016. Entropic Metric Alignment for Correspondence Problems. *ACM Trans. Graphics* 35, 4, Article 72 (July 2016).

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Proc. SGP*. 109–116.

Daniel Sýkora, John Dingliana, and Steven Collins. 2009a. As-Rigid-As-Possible Image Registration for Hand-Drawn Cartoon Animations. In *Proc. NPAR*. 25–33.

Daniel Sýkora, John Dingliana, and Steven Collins. 2009b. LazyBrush: Flexible Painting Tool for Hand-Drawn Cartoons. *Computer Graphics Forum* 28, 2 (April 2009), 599–608.

Daniel Sýkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. 2014. Ink-and-Ray: Bas-Relief Meshes for Adding Global Illumination Effects to Hand-Drawn Characters. *ACM Trans. Graphics* 33, 2, Article 16 (April 2014).

Daniel Sýkora, David Sedlacek, Sun Jinchao, John Dingliana, and Steven Collins. 2010. Adding Depth to Cartoons Using Sparse Depth (In)equalities. *Computer Graphics Forum* 29, 2 (May 2010), 615–623.

Kevin Wampler. 2016. Fast and Reliable Example-based Mesh IK for Stylized Deformations. *ACM Trans. Graphics* 35, 6, Article 235 (Nov. 2016).

Yu Wang, Alec Jacobson Jernej Barbič, and Ladislav Kavan. 2015. Linear Subspace Design for Real-Time Shape Deformation. *ACM Trans. Graphics* 34, 4, Article 57 (July 2015).

Nora S. Willett, Wilmot Li, Jovan Popović, Floraine Berthouzoz, and Adam Finkelstein. 2017. Secondary Motion for Performed 2D Animation. In *Proc. UIST*. 97–108.

Stefan Wirtz, Bernd Fischer, Jan Modersitzki, and Oliver Schmitt. 2004. Superfast Elastic Registration of Histologic Images of a Whole Rat Brain for 3D Reconstruction. In *Proc. SPIE 5370, Medical Imaging 2004: Image Processing*. 328–334.

C. T. Zahn and R. Z. Roskies. 1972. Fourier Descriptors for Plane Closed Curves. *IEEE Trans. Comput.* C-21, 3 (March 1972), 269–281.

Lei Zhang, Hongbo Fu, and Hua Huang. 2011. EXCOL: An EXtract-and-COmplete Layering Approach to Cartoon Animation Reusing. *IEEE Trans. Visualization and Computer Graphics* 18, 7 (July 2011), 1156–1169.

Haichao Zhu, Xueting Liu, Tien-Tsin Wong, and Pheng-Ann Heng. 2016. Globally Optimal Toon Tracking. *ACM Trans. Graphics* 35, 4, Article 75 (July 2016).

APPENDIX

We list all the characters that we consider, together with statistics about the characters and run-time.

Table 1: Details of the puppet characters presented in this paper and the accompanying video. For each puppet, we list the number of layers, the total number of handles and joints, the reference frame resolution, and the average end-to-end processing time per target frame (including correspondence, registration, segmentation, and texture completion).

Character Name	Layers	Handles	Joints	Resolution	Runtime (min)
Dancer [Willett et al. 2017]	9	21	4	290×375	10
Bear[1] [Willett et al. 2017]	6	31	2	205×240	9
Lola [Child 2013]	6	13	4	262×382	20
Peppa Pig [Astley Baker Davies 2003]	7	22	2	239×302	9
Ghost [Willett et al. 2017]	2	18	0	277×275	4
Wilk [Adobe 2018]	6	12	2	330×450	18
Fox [Adobe 2018]	4	17	2	450×330	14
Snowman [Adobe 2018]	7	15	6	351×405	21
Airplane [Adobe 2018]	2	6	1	450×330	6
The Hatchling Bird [Eastman 1960]	8	23	0	300×240	6
The Gruffalo Mouse [Donaldson and Scheffler 1999]	7	25	6	304×255	10

[1] For illustration purpose, we don't show all the handles for the bear in Figure 2.