E-StopMotion: Digitizing Stop Motion for Enhanced Animation and Games

Anamaria Ciucanu
University of Bath
Bath, UK
ac884@bath.ac.uk

Naval Bhandari
University of Bath
Bath, UK
N.Bhandari@bath.ac.uk

Xiaokun Wu
University of Bath
Bath, UK
X.Wu@bath.ac.uk

Shridhar Ravikumar
University of Bath
Bath, UK
shridharravikumar@gmail.com

Yong-Liang Yang
University of Bath
Bath, UK
y.yang@cs.bath.ac.uk

Darren Cosker
University of Bath
Bath, UK
D.P.Cosker@bath.ac.uk

ABSTRACT

Stop Motion Animation is the traditional craft of giving life to handmade models. The unique look and feel of this art form is hard to reproduce with 3D computer generated techniques. This is due to the unexpected details that appear from frame to frame and to the sometimes choppy appearance of the character movement. The artist's task can be overwhelming as he has to reshape a character into hundreds of poses to obtain just a few seconds of animation. The results of the animation are usually applied in 2D mediums like films or platform games. Character features that took a lot of effort to create thus remain unseen. We propose a novel system that allows the creation of 3D stop motion-like animations from 3D character shapes reconstructed from multi-view images. Given two or more reconstructed shapes from key frames, our method uses a combination of non-rigid registration and as-rigid-as-possible interpolation to generate plausible in-between shapes. This significantly reduces the artist's workload since much fewer poses are required. The reconstructed and interpolated shapes with complete 3D geometry can be manipulated even further through deformation techniques. The resulting shapes can then be used as animated characters in games or fused with 2D animation frames for enhanced stop motion films.

CCS CONCEPTS

• **Computing methodologies** → **Animation**; • **Applied computing** → **Media arts**;

KEYWORDS

stop motion animation, 3D games, animation reconstruction, non-rigid registration, as-rigid-as-possible interpolation

Figure 1: Artist at Fat Pebble [21] working on characters for the game Clay Jam [22].

ACM Reference Format:
Anamaria Ciucanu, Naval Bhandari, Xiaokun Wu, Shridhar Ravikumar, Yong-Liang Yang, and Darren Cosker. 2018. E-StopMotion: Digitizing Stop Motion for Enhanced Animation and Games. In *MIG '18: Motion, Interaction and Games (MIG '18), November 8–10, 2018, Limassol, Cyprus.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3274247.3274505

1 INTRODUCTION

Stop motion refers to a special type of animated film, in which physical characters are placed in different poses for each frame by hand (see Figure 1). It is also common for these poses to be modeled separately, when a particular shape is desired per frame. The results are usually a sequence of images, which are further combined to create a 2D animation [23]. Compared to digital animation based on computer graphics (CG) techniques, its style is more photorealistic, since frames are obtained by taking photographs of real world characters.

However, the whole process is time consuming and solely depends on the skills and efforts of the stop motion artist. The resulting animations often exhibit non-smooth transitions between frames due to the limited number of poses created. This effect is sometimes desired, given the unique signature look of the stop motion art form. Nevertheless, when enhancements of the animation are required, it is very hard to edit an existing stop motion animation sequence. This is due to capturing the character poses as 2D images, without

having the corresponding 3D geometry for making further refinements. Moreover, it is not trivial to bring the handmade character animation into a 3D environment, like a game. Thus qualities of the real model like texture, volume, custom animation style, may be lost when 2D mediums are the preferred method of visualization.

To address the above limitations, we propose a novel system that allows the artist to reconstruct and refine handmade animation on the computer. Our method chiefly benefits scenarios where multiple character poses need to be modeled individually, as it reduces the artist's workload significantly. Even when a single physical character is used to shape all the poses in an animation, our digitization pipeline can help bring it to life in a 3D environment.

The main idea is to digitize a few *key poses* of a physical character and use interpolation to generate plausible in-between poses. This largely simplifies the process since fewer physical poses are required for smoother animations. Also, the digitization enables flexible post-production editing of the animation based on existing CG techniques, such as shape deformation, image and shape composition etc. Note that we do not intend to diminish the virtues of stop motion animation, but rather make it more versatile and accessible to the virtual worlds of animation and 3D games.

In practice, we face two main challenges. Firstly, in stop motion animation, characters are static and have to be manually manipulated by an artist. The scale of deformation can vary significantly from one key pose to another. Thus it becomes more difficult to identify correspondences between the respective scans. This is different from continuous capture of deformable shapes like humans, where the significant amount of available data makes it possible to reconstruct the shape and movement reliably ([26]). Secondly, since the available key poses can be sparse, the deformation between them is often large. Generating plausible intermediate poses is not straightforward, as common techniques like linear interpolation may result in artefacts.

We propose to use non-rigid registration with a few user picked landmark points to establish dense correspondences between key poses. The real world models used are made of plasticine and present high flexibility of movement. This non-rigidity is encoded in our algorithm by locally defined affine transformations to cope with possible elastic and plastic deformations. It is more flexible than registration methods based on local rigidity, isometry and conformality. Also, we extend a normal equation based as-rigid-as-possible shape interpolation from 2D to 3D to interpolate satisfactory intermediate poses. We test our pipeline on various plasticine characters from the game Clay Jam [22] by Fat Pebble [21] to show its effectiveness. We also demonstrate two applications for stop motion animation and games that benefit from our system.

In summary, our paper makes two main contributions: i) a novel system that digitizes stop motion based on robust 3D registration and interpolation, and ii) two major applications for enhanced stop motion animation and games.

2 PREVIOUS WORK

2.1 Animation Reconstruction

Animation reconstruction aims at capturing and constructing animated content from moving objects in the physical world [12]. This involves capturing data from a deforming model as it evolves over time. The available data for these scenarios is usually dense and the deformations between frames are small, making it easier to find reliable correspondences.

Common approaches make use of a template shape which is fitted to the point clouds at each frame of animation [13, 26]. Although this approach offers a significant prior, it can restrict the quality of animation as argued in [30]. Alternative approaches have been proposed to reconstruct animation without a template shape. Tevs et al. [30] propose a cartography inspired method of reconstructing a shape and its movement from partial scans and by imposing isometric deformations. This restriction would be an impediment in our case, since plasticine can deform nonisometrically and irreversibly.

Wand et al. [35] present an efficient pipeline for animation reconstruction, which involves the minimization of an energy term considering distances between data points, acceleration and velocity, volume change and surface smoothness. Their method is complex to implement, however, and only works accurately for small deformations between consecutive frames. Sharf et al. [24] reconstruct motion by making use of a 3D grid that embodies the whole character. Although volume preservation is a desirable feature for plasticine models, the use of a volumetric model for reconstructing animation is time consuming and needs an expensive rendering mechanism for visualization.

We propose a specialized animation reconstruction approach that is suitable in a new scenario, where only sparse key poses are available. Our argument stands in the advantages of a stop motion animation reconstruction pipeline for the animation and game industries. We first choose a non-rigid registration technique that takes into account the nonisometric nature of plasticine deformation. After obtaining a suitable registration, an as-rigid-as-possible interpolation technique is used to preserve the shape of the character between the source and obtained target pose.

2.2 Non-rigid Registration

Several surveys discuss shape matching [16], registration [28] and finding correspondences [7, 31]. Most of the work focuses on isometric or nearly isometric shape deformations, which preserve the local features as accurately as possible. Given that plasticine can move irreversibly in any direction, we focus on techniques that allow nonisometric deformation to some extent. Also, the available data from our scans consists of triangular meshes. Therefore we studied methods relevant to this form of model representation.

Non-rigid registration is a combination of the correspondence problem and the trajectory problem. The better the correspondences are between the source and target shapes, the easier it is to find a deformation trajectory between them. Likewise, when the deformation trajectory is known, good correspondences are easy to establish [12].

As specified in [32], the characteristics of good correspondences are bijectivity, continuity and similarity matching between salient points. This is given in the context of nonisometric deformations between shapes, where preserving geodesic or diffusion distances between pairs of correspondences fails. Approaches that involve stochastic methods [32, 33], region correspondence or functional maps [10, 19] have been very effective in mapping between topologically similar, but geometrically different shapes. However, the

complexity of implementation, time costs or the reliance on features like surface curvature have made it difficult to opt for such methods in our case. Given that plasticine characters can change their local curvature dramatically, it becomes difficult to rely on more than lower frequency Laplace-Beltrami eigenfunctions when matching between similar regions of the shape.

Concerning the trajectory problem, a lot of the literature focuses on iteratively determining the deformation of the source shape in parallel to finding correspondences with the target shape [3, 4, 6, 15, 36]. Amberg et al. [4] propose a non-rigid iterative closest point registration (NRICP) algorithm, where an affine transformation is used per vertex. Affine transformations per vertex give the mesh freedom to rotate, translate, scale and shear. Surface smoothness is also ensured by using a global stiffness, which decreases as the source shape approaches the target shape. The smaller the stiffness becomes, the more malleable the surface is. This imitates the physics of plasticine when undergoing tensile stress. The amount of stretch in a material is directly proportional to its propensity to stretch further until a breaking point.

Enhanced versions of the NRICP are common approaches for gradually matching a source to a target shape. One example of an enhancement for NRICP is the coarse-to-fine technique. Deformations are separated into levels of complexity, from large to small. Deformations graphs are sometimes used to approximate the source shape and cover the larger deformations that it may undergo [14, 20]. Zell and Botsch [37] use non-rigid registration with an added smoothing step. This is done by embedding the template and target shapes into a simpler domain through a Laplacian smoothing procedure. Although nonisometric non-rigid deformations are allowed, their work is more suitable for face models than for limbed characters. Nevertheless, one of the examples in their paper is matching a sequence of clay character scans, which strengthens our desire to pursue the non-rigid registration path.

In our approach we use the non-rigid registration method proposed by [4] as it allows enough freedom of movement for our plasticine characters. This is considered a sufficient approach for our needs since the deformations are moderate. Future helpers that take into account the nature of the material (volume preservation, stiffness) and the character structure (skeletons, coarse-to-fine helpers) might be considered for characters displaying large deformations.

2.3 Interpolation

A popular technique with interpolation is to use multiple shapes to guide the deformation [9, 34]. Although our registration algorithm generates a small set of meshes with the same connectivity, we argue that this approach can constrain the deformations more than it is desirable. Since our models are made of plasticine, we expect their deformation to flow in a physically plausible way. Maintaining the smoothness of the surface is also important, since all of the generated shapes can be part of the final animation. In this respect we find it suitable to adopt an as-rigid-as-possible approach to our interpolation [2, 25]. The idea of considering individual cells of the surface rigid can also link to the surface tension property that plasticine displays [8].

Alexa et al. [2] propose an algorithm that allows the interpolation between two sets of meshes using triangles in 2D or tetrahedra in 3D by minimizing a quadratic error function. They account for translation by fixing a vertex using linear interpolation, which, depending on the vertex chosen, can give different results. Building on the work of [2], Liu et al. [17] address this problem by generating a set of surface tetrahedra on the fly as opposed to needing a fixed initial and end volumetric mesh. No vertex needs to be fixed, as translation is taken into account in their minimization function.

Baxter et al. [5] provides an improvement on several algorithms for interpolating between 2D shapes by incorporating normal equations. They provide a mathematically equivalent solution to minimising the quadratic error. They do not tackle the need to fix a vertex, instead using it to their advantage to enable more control over the trajectories taken by the vertices by using Lagrange multipliers. Their solution does not extend into 3D, and for it to be mathematically equivalent to the method of [2], it would require the use of volumetric tetrahedral meshes.

In our approach we extend on the works of [5] and [17] and use normal equations and dynamically generated tetrahedra to create as-rigid-as-possible deformations between shapes.

3 OVERVIEW

Given a limited number of key poses for a physical character made of plasticine, the goal of our system is to digitize these shapes and create plausible in-between poses for animation and game applications. Figure 3 shows an overview of our E-StopMotion pipeline. It consists of four components: shape acquisition is based on an existing multi-view reconstruction approach, two major algorithmic components (non-rigid registration, and as-rigid-as-possible interpolation) and applications that find novel uses for the reconstructed and interpolated shapes in stop motion animation and games.

The acquisition component is responsible for capturing and reconstructing the geometry and texture of the key poses of a plasticine character created by an artist (Figure 2). The multi-view 3D reconstruction approach, as implemented in Agisoft Photoscan [1], is employed to generate the digital shapes represented by triangular meshes. The images used in the reconstruction process are captured with an Alphashot 360 machine [18]. The resulting meshes display most of the character's details in their key poses, depending on the desired resolution. Artefacts like holes and overlapping triangles also make an appearance. A clean up procedure is needed to make the meshes watertight, 2-manifolds. Texture remapping is also used to redefine the UV and texture maps.

The following two algorithmic components are devised to match the clean, scanned key poses and generate plausible in-between poses through interpolation (see Sections 4 and 5). This can help reduce the number of manually manipulated poses, which facilitates the traditional stop motion creation process and benefits subsequent animation and game applications. We utilize non-rigid registration [4] with local affine transformations to align the key pose scans. Since the deformations between the meshes vary from small to large, we allow the user to pick a few landmark points to guide the registration. Based on the resulting surface correspondences, we then make use of our as-rigid-as-possible interpolation method to obtain pleasing intermediate shapes. Our approach to

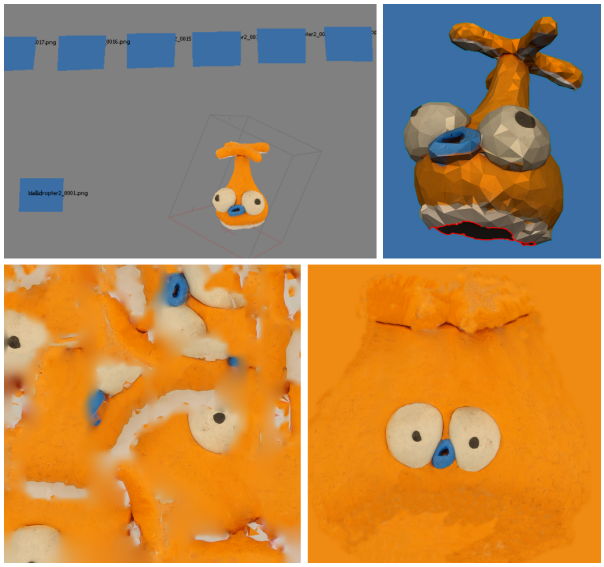

Figure 2: Acquisition component for Hellidropter consists of photogrammetry in Photoscan (top left), mesh cleanup (top right) and texture remapping (bottom).

interpolation extends an existing normal equation based shape interpolation technique from 2D to 3D [5, 17]. Also it does not require consistent tetrahedral meshing between key poses, thus can save computational time while achieving plausible results.

The scanned key poses and interpolated intermediate poses can next be imported into existing animation and game development software for applications. To demonstrate the use of our system, we import the digitized poses into a popular animation software and a game engine (Blender and Unity respectively) to generate enhanced stop motion animation and stop motion-like games (see Section 6).

4 E-STOPMOTION ALGORITHMS

In this section, we elaborate the two algorithmic components of our system.

4.1 Non-rigid Registration

The non-rigid registration component is adapted from [4]. The algorithm is based on an iterative procedure similar as the rigid ICP [6]. It also uses the closest point correspondences to match a source mesh M_s as closely as possible to a target mesh, M_t. The difference is that local admissible affine transformation is employed instead of a global rigid-body motion to enable registering two shapes with non-rigid deformation in between, which meets the requirement in our context.

Due to the high number of degrees of freedom given by the vertex affine transformations, a smoothness regularization term is involved to make the problem solvable. Each iteration of the registration is solved by finding the optimum affine transformations per vertex, which minimize the energy functional described in Eqn. 1. A decreasing global stiffness is further used to constrain the amount of movement each vertex can undergo. The closer the source shape gets to the target shape, the higher the probability that the chosen correspondences are accurate. To prevent wrong correspondences from influencing the match, the stiffness starts off as high and decreases as the distance between the source and target meshes becomes smaller.

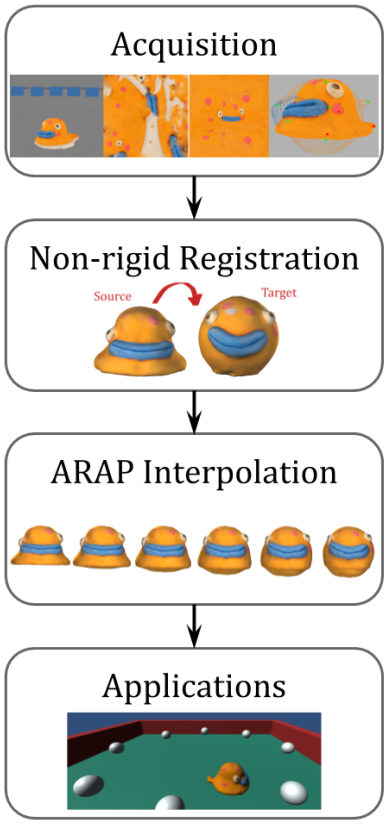

Figure 3: The major components of our E-StopMotion pipeline: Acquisition - involves scanning and cleaning the plasticine models; Non-rigid Registration - finds the correspondences and transformations between the clean models; ARAP Interpolation - interpolates between the registered shapes as-rigidly-as-possible; Applications - highlights the uses of our pipeline in 2D animation and games.

An example of non-rigid registration to align the source and target shape is shown in Figure 4, and the mathematical formulation and registration procedure are detailed in the following paragraphs.

4.1.1 *Energy Minimization in a Single Iteration.*

$$E(X) = E_d(X) + \alpha E_s(X) + \beta E_l(X) \quad (1)$$

$X = [X_1, ..., X_N]^T$ is the transformation matrix that minimizes the total energy $E(X)$. It contains N 4x4 affine matrices X_i, one per source vertex. E_d is the energy term that measures the sum of squared distances between correspondences, E_s is the smoothness regularization term between neighbouring vertices, and E_l is the landmark energy term which allows the user to pick a few corresponding landmark points to guide the registration.

Figure 4: Blob Fish source (left) and target (middle) meshes with landmarks and the registered mesh (right).

$$E_d(X) = \sum_{v_i \in M_s, u_i \in M_t} w_i \,||\, X_i v_i - u_i \,||^2 \tag{2}$$

$$E_s(X) = \sum_{(i,j)\in\mathcal{E}} ||\, (X_i - X_j)\, G \,||^2 \tag{3}$$

$$E_l(X) = \sum_{(v_i,l)\in\mathcal{L}} ||\, v_i - l \,||^2 \tag{4}$$

The weights w_i from (2) are 0 or 1, signifying whether the correspondence pair (v_i, u_i) is reliable or not. A correspondence pair may be filtered out as unreliable when the angle between their respective normals is greater than 90°.

The smoothness of the transformation between reliable correspondences is ensured by the term in (3). The idea is to minimize the differences in transformation between neighbouring vertices. The pair (i, j) represents an edge between vertices v_i and v_j, while $\mathcal{E}$ is the list of edges of the source mesh M_s. Also, matrix G from (3) is a diagonal matrix of the form $G = diag\{1, 1, 1, \gamma\}$, which can influence the scale and rotation contribution of the regularization through the term γ.

The pair (v_i, l) represent a landmark correspondence pair and is considered separately from the closest point correspondences. These pairs of vertices are deemed to be essential in guiding the deformation and are usually weighted higher than the global stiffness at the beginning ($\beta > \alpha$). After a certain threshold of α, it is considered that the landmark influence is no longer necessary ($\beta = 0$). This is due to the implication of the source mesh being close enough to the target mesh, which results in enough reliable correspondences.

Landmarks can also be seen as helpers that guide the deformation. They work best when chosen in regions with high displacement between the source and the target meshes. We found that 10-15 landmarks, distributed noncoplanarly across the surface, are sufficient for guiding the deformation and avoiding artefacts. Coplanar landmarks produce flattening effects and too few or too many landmarks may cause the solution to get stuck in local minima.

Other helpers that will be considered more thoroughly in the future are inspired by the structure or material of the character. Characters with limbs may benefit from skeletons to guide the large deformations (Figure 12). Also, local stiffness, instead of global stiffness would allow some regions to remain stiff, while other regions deform ([11]).

4.1.2 Matrix Notation. In practice, to facilitate implementation, it is more convenient to rewrite the energy functional from Eqn. 1 using matrix notation. For a full description of the matrix notation please refer to the work of [4]. The following representation of the energy function in matrix notation is the final form used in our implementation. Note that $||\cdot||_F$ indicates the Frobenius norm.

$$\overline{E}(X) = \left\| \begin{bmatrix} \alpha M \otimes G \\ WD \\ \beta D_{\mathcal{L}} \end{bmatrix} X - \begin{bmatrix} 0 \\ WU \\ U_{\mathcal{L}} \end{bmatrix} \right\|_F^2 \tag{5}$$

Matrix M is the directed node-arc matrix, representing whether there is an edge between two vertices. The direction of the respective edge is indicated by ± 1. $W = \{w_1, ..., w_N\}$ contains the weights applied to the source to target correspondences stored in matrices D and U respectively. Similarly, $D_{\mathcal{L}}$ and $U_{\mathcal{L}}$ contain the landmark correspondences, weighed by β.

4.1.3 Iterative Registration Procedure. As described in Algorithm 1, two loops are involved in an iterative process to incrementally align the source shape to the target. The outer loop decreases the global stiffness for each iteration, allowing the current mesh more freedom of movement. The inner loop corresponding to each iteration has two steps. The first step is to find correspondences between the source and the target meshes. The second step is to calculate the transformation matrix X that would bring the current source mesh closer to the desired target mesh. This loop finishes when the differences between two consecutive transformation matrices is lower than a threshold ε.

Algorithm 1: Non-rigid Registration

Input: Clean source mesh M_s and clean target mesh M_t with a small set of landmarks $\mathcal{L}$ between them.
$\alpha = MAX\_STIFFNESS$
Output: The source mesh registered to the target mesh, M_{s-t}
Align M_s and M_t globally using a Procrustes analysis.
Initialize the transformation matrix X_0.
while $\alpha > MIN\_STIFFNESS$ **do**
  while $||X_i - X_{i-1}|| < \varepsilon$ **do**
    Find correspondences u_i between current source mesh $M_s^{'}$ and target mesh M_t.
    Calculate X_i as the solution to a linear system of the form $AX = B$, where A is the matrix described in the previous section.
  end
end

4.2 Interpolation

Our as-rigid-as-possbile (ARAP) interpolation approach builds on the work of [17] and [5]. It employs a local-global optimization strategy to compute the topologically consistent intermediate meshes between a source and a target mesh. Locally it computes the affine transformation for each local shape element from the source to the target and obtains an expected intermediate interpolation first. The algorithm then globally solves for the intermediate mesh by minimizing the sum of differences between all expected local transformations and the corresponding actual transformations.

Unlike the original as-rigid-as-possible method [2], we do not require consistent tetrahedral meshes, and instead calculate surface

tetrahedra from triangle meshes. Consistent tetrahedral meshes are difficult to create [17], meaning that we can sidestep this issue. We also formulate the problem using normal equations, allowing us to address issues that are carried over from 2D shape interpolation in 3D. This gives an elegant solution to symmetric interpolation and adds constraints and control to the interpolated results. It also allows for easy correction of rotational inconsistencies.

4.2.1 Energy Terms. The mesh is constructed of several tetrahedra, many of them sharing vertices. An interpolated affine transformation for one tetrahedron will not necessarily be the same affine transformation for another tetrahedron with a common vertex. We wish to minimise the global error between every expected affine transformation A_T, and the actual one B_T. We let $V_T(t)$ be our set of unknown interpolated vertex positions, and can define the relationship between $V_T(t)$ and the source tetrahedron V_T^* as $B_T = V_T^* V_T(t)$. This gives us our minimisation equation:

$$E(t) = \sum_{T=1}^{M} a_T ||B_T - A_T(t)||_F^2, \tag{6}$$

where a_T is the area of the original triangle of a tetrahedron V_T, and $|| \cdot ||_F$ is the Frobenius norm. We use the area to weigh the importance of the particular triangle, as it allows for large differences in tessellation of a mesh, whilst maintaining the same interpolation results.

4.2.2 Matrix Notation. We calculate an affine transformation A_T from a source tetrahedron V_T to a target U_T. Following the approach used by [5]; for each tetrahedron, we calculate V_T^*. This is the inverse of V_T, giving us the affine transformation: $A_T = V_T^* U_T$. We build two matrices $-H$ and $G(t)$, using them to construct Equation 6:

$$-H_T = a_T (V_T^*)^T V_T^*, \tag{7}$$

$$G_T(t) = a_T (V_T^*) A_T(t), \tag{8}$$

where we can then place the values of $-H_T$ and $G_T(t)$ into $-H$ and $G(t)$, at the position of the index of each vertex. $-H$ is a sparse $(N + M) \times (N + M)$ matrix, where the values of $-H_T$ are summed at corresponding indices in both directions of $-H$. $G(t)$ is an $(N + M) \times 3$ matrix, where the values of $G_T(t)$ are summed in rows corresponding to the vertex indices. N and M are the number of the vertices and triangles, respectively.

It is important to note that $-H$ is independent of t, and thus only needs to be calculated once, before interpolation. As $G(t)$ relies on each $A_T(t)$, it must be calculated at every time step. Using the approach proposed by [2] we solve $V(t)$ as follows:

$$V(t) = -H^{-1} G(t). \tag{9}$$

5 RESULTS

We applied our pipeline to a set of characters from the game Clay Jam[22]) created by Fat Pebble ([21]). One of the models was even recreated by hand, since the original character had been lost (Figure 13). The difference between frames was moderate, with large amounts of overlap. In some cases, nonisometries made the matches difficult, but we found that landmark correspondences corrected the artefacts to some extent. We aimed at a minimal set of handpicked landmarks (10-15), however, since it can be a tedious process for the artist.

Figures 10 - 14 show the in-between poses obtained from the original source and registered source scans of a few plasticine characters. The exception is figure 12, which has the target shape generated by rigging and deforming the source shape. This was done to exemplify the future use of skeleton helpers for registration and the difference between linear and as-rigid-as-possbile (ARAP) interpolation when rotation produces nonlinear deformations. These pairwise results were extended to more key poses to create varied animation sequences (see supplementary video).

Figures 10, 11 and 12 display a comparison between linear and ARAP interpolation. Figures 10 and 11 do not display significant differences in shape, due to the deformations being approximated well by both linear and ARAP interpolation. Figure 5 shows the small shape difference for the Blob Fish character. The difference between the two types of interpolation methods becomes visible when rotation produces nonlinearities that cannot be approximated well with the linear approach. Figure 12 shows how linear interpolation shrinks the volume of the Party Onion character when a large rotation is introduced, while ARAP interpolation manages to preserve the shape better.

Tables 1 and 2 show the average area and volume distortions and standard deviations for our characters in both types of interpolation. The area and volume distortions are relative to the original mesh area and volume respectively. Notice how the average area distortions and standard deviations are generally smaller for the ARAP interpolation, compared to the linear one. The average volume distortions and standard deviations are close in values, with ARAP displaying smaller distortion than linear interpolation when nonlinear deformations are more common. The difference is especially noticeable in the case of Party Onion from figure 12. While the linear interpolation shrinks the area and volume of the model, the ARAP interpolation distortion is significantly smaller.

Figure 5: The contours of three iterations of the Blob Fish interpolations. Linear interpolation is shown in grey, while ARAP interpolation is red. Notice the small difference in shape between the two types of interpolation in this case.

Figure 6 shows a normal distribution of the area and volume distortions for Tick (Figure 13) and Party Onion (Figure 11). In case of the Tick (top part of the image), although the average and standard deviation of the area and volume distortions are smaller for the ARAP interpolation, we see the frequency of smaller distortions is higher for the linear interpolation (red), than for the ARAP interpolation (blue). The ARAP distortion, however, seems more uniformly distributed than the skewed linear one. This signifies that distortion evolves more smoothly in the former case.

Table 1: Average relative area distortions for linear and as-rigid-as-possible (ARAP) interpolations and the corresponding standard deviations. The area ratios are calculated per interpolation step, relative to the original surface area. Negative values signify that the mesh area is preponderantly shrinking during the interpolation, while positive values signify the area is growing. The meshes considered can be seen in figures 10 to 14 respectively. The landmarks specified are used in the non-rigid registration step.

Character	Faces	Landmarks	Linear Dist.	Linear σ	ARAP Dist.	ARAP σ
Blob Fish	3482	12	-0.0537	0.0245	-0.0208	0.0157
Party Onion	3600	13	0.0224	0.0185	**4.6793 E-04**	0.0020
Party Onion Rigged	3600	N/A	-0.0299	0.0208	**9.2850E-05**	**1.3018E-04**
Tick	2750	13	0.0835	0.0568	-0.0329	0.0235
Hellidropter	1454	10	0.0202	0.0196	-0.0206	0.0142

Table 2: Average relative volume distortions for linear and as-rigid-as-possible (ARAP) interpolations and the corresponding standard deviations. The volume ratios are calculated per interpolation step, relative to the original surface volume. Negative values signify that the mesh volume is preponderantly shrinking during the interpolation, while positive values signify the volume is growing. The meshes considered can be seen in figures 10 to 14 respectively. The landmarks specified are used in the non-rigid registration step.

Character	Faces	Landmarks	Linear Dist.	Linear σ	ARAP Dist.	ARAP σ
Blob Fish	3482	12	0.1025	0.0759	0.1663	0.0822
Party Onion	3600	13	-0.0334	0.0207	0.0357	0.0222
Party Onion Rigged	3600	N/A	-0.0134	0.0077	**8.7708E-04**	**5.6068E-04**
Tick	2750	13	0.0674	0.0454	-0.0542	0.0402
Hellidropter	1454	10	0.0229	0.0195	-0.0254	0.0182

The Party Onion distortion distributions (bottom part of the image) are close in shape, probably due to most of the mesh having small deformations, while only the tentacle rotates.

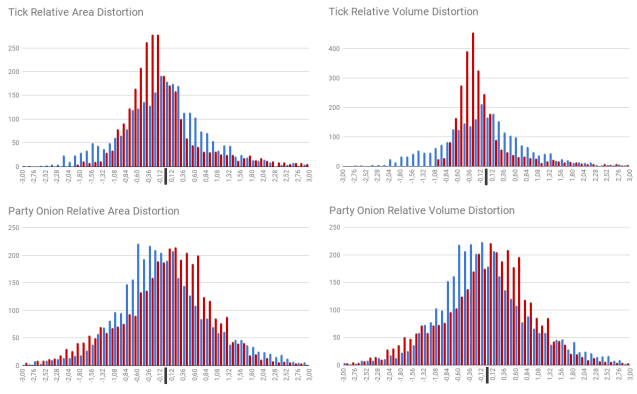

Figure 6: Normalized relative area and volume distortions for Tick (Figure 13) in the top part and Party Onion (Figure 11) in the bottom part of the image. The ARAP interpolation distortions are shown in blue, while the linear distortions are red. This analysis was made on all the faces of each mesh, for one iteration of the interpolation.

Performance. All the experimental results are generated on a laptop with Intel Core i7-4710HQ CPU (2.50 GHz) and 16 GB memory. The code was written in MATLAB. For a typical digitized model (as the example shown in Figure 10) with 3490 triangles it takes 267.634 seconds to register between two poses and 120.065 seconds to interpolate poses between them. The Hausdorff distance between the registered model and the target model (Figure 4) in this case is 0.296. In the future, we could reimplement our pipeline in C++ for faster times.

Nevertheless, even with the current performance, together with the needed time for scanning and cleaning a few character poses, the artist's workload is reduced considerably. The time required for scanning and cleaning a character can take 0.5 - 2 hours, depending on the skill of the artist and the complexity of the character. In the case of digitizing stop motion for 3D games, at least, the artist needs only a small set of key poses to be cleaned up. The intermediate poses are then generated automatically with our pipeline. In the case of an interpolation step $t = 0.1$, we argue that the needed time for an artist to obtain a 3D sequence of poses between two key poses is approximately 10 times faster than doing everything manually.

6 APPLICATIONS

6.1 Enhanced Stop Motion Animation

Based on the reconstructed key poses and interpolated intermediate poses, the existing 2D stop motion animation frames can be easily enhanced using traditional CG animation and composition

techniques. For example, as shown in Figure 7 (see also supplementary video), the input stop motion animation only contains two 2D frames, corresponding to two key poses created by the artist.

We directly import these frames and the in-between 3D meshes into Blender, a popular animation software available online. To combine our 3D shapes with the 2D frames, the virtual camera parameters for projecting 3D shapes to 2D are estimated by manually aligning 3D reconstructed meshes with the existing 2D stop motion animation frames. These camera parameters can be used to render new 2D frames from the 3D shapes to enhance the input animation frames (only two in this case). Moreover, since the 3D geometry of the key pose is available, existing deformation tools (e.g. Free Form Deformation in Blender) can be used to easily create new poses which are challenging to achieve otherwise, as all the resources needs to be physically replicable afterwards, including human resources (artist), and natural resources (plasticine, camera, etc.).

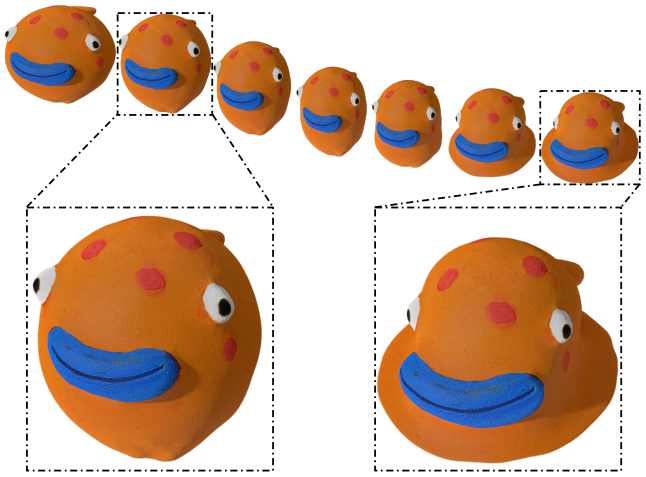

Figure 7: Enhancing existing stop motion animation with additional frames using our approach. In the example figure, the two highlighted frames are the source and target scans, with the same connectivity after registration. The in-between poses are modified versions of the interpolated shapes obtained through our ARAP interpolation. The added modifications create a pleasing squash and stretch effect for the resulting animation.

6.2 Stop Motion-like Games

The resulting meshes are imported as FBX files in a modified Unity demo game. Since the MATLAB code exports results as OBJs, an extra step of importing models into Maya and exporting them as FBX files was necessary. This could be easily automated in the future with a python script.

An empty game object was created, with two C# scripts attached to it. One script controls the stop motion animation effect and the other is in charge of the gameplay. The first script has a public array of game objects named `frames`, where the meshes are loaded. Another public field is `fps` (frames per second), which coincides with how fast the frames will be appearing and disappearing. A sampling variable was also included, so that the user can choose which meshes should be displayed. The resulting game scene is shown in Figure 8. The supplemental video shows a demo of how original (using only key poses) and enhanced (using interpolated poses) stop motion character animations can be used in Unity games.

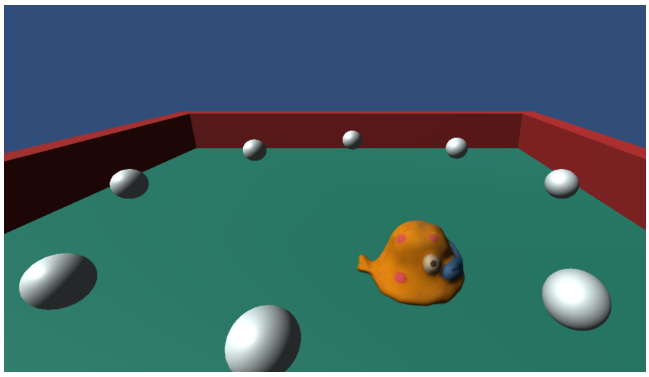

Figure 8: Application of E-StopMotion pipeline for video games. In this example, the Blob Fish animation is imported into an example Unity [29] game (see supplementary material).

7 DISCUSSION

We have presented a novel pipeline for reconstructing and refining stop motion animation for use in 3D games. By combining the handmade craft of stop motion with 3D scanning, non-rigid registration and interpolation, artists are now able to bring their creations onto the screen in three dimensions.

Artists can scan and clean a small set of key poses for their character and then have plausible in-between poses generated automatically. This process makes the animation reconstruction technique accessible to any games company. Users can then interact with a digital version of a handmade character in a game.

Moreover, both 3D and 2D stop motion animation can benefit from our pipeline. The in-between poses can be easily aligned to match the original orientation of a character. Instead of photographing every frame, automatically generated in-between poses can be used to reduce the workload or refine an initial animation.

Limitation. Figure 9 shows an example of a character pose that did not match the target shape as expected. This is due to rotation induced nonlinearities that cannot be approximated well through affine transformations. A guiding skeleton could have been beneficial in such a scenario. Figure 12 shows an example of how a skeleton might guide the deformation before registration is employed.

Future Work. The interpolation process was satisfactory for our needs, but only produced results as good as the registration process allowed it to. This limitation needs to be addressed in the future by adopting a state-of-the-art non-rigid registration method that allows nonisometry between the source and target models. The correspondence problem can be better addressed by identifying

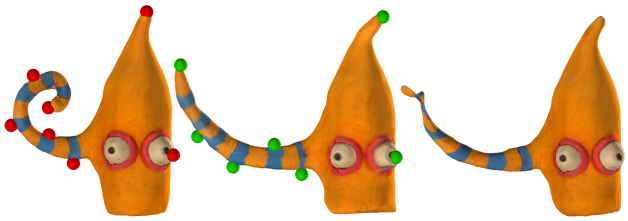

Figure 9: This is an example where the NRICP fails, due to the nonlinearity of deformation. Party Onion source (left) and target (middle) meshes with landmarks and the registered mesh (right). Notice the collapse in the tentacle of the registered mesh.

nearly-isometric regions like the eyes and mouth and extending correspondences from there based on the smoothness of deformation ([33]). The trajectory problem can be addressed by using coarser structures that guide the large deformation. Deformation graphs ([14]) or skeletons extracted from the mesh ([27]) could significantly improve the deformation. Regarding the look and feel of stop motion animation, it would be interesting to conduct qualitative experiments on the aesthetics differences between linear and ARAP interpolation.

8 ACKNOWLEDGEMENTS

The authors would like to thank Fat Pebble for providing us with their Clay Jam characters for our work. More specifically, we thank Iain Gilfeather, the Technical Director, Chris Roe, the Art Director and Michael Movel, the Design Director for their support and patience. The work is funded by the UK's EPSRC Centre for Doctoral Training in Digital Entertainment (CDE), EP/L016540/1, and CAMERA, the RCUK Centre for the Analysis of Motion, Entertainment Research and Applications, EP/M023281/1.

REFERENCES

[1] Agisoft. 2018. Photoscan. http://www.agisoft.com/.
[2] Marc Alexa, Daniel Cohen-Or, and David Levin. 2000. As-rigid-as-possible Shape Interpolation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 157–164.
[3] Brett Allen, Brian Curless, and Zoran Popović. 2003. The Space of Human Body Shapes: Reconstruction and Parameterization from Range Scans. *ACM Trans. Graph.* 22, 3 (July 2003), 587–594.
[4] Brian Amberg, Sami Romdhani, and Thomas Vetter. 2007. Optimal Step Nonrigid ICP Algorithms for Surface Registration. *2007 IEEE Conference on Computer Vision and Pattern Recognition* (2007), 1–8.
[5] William Baxter, Pascal Barla, and Ken-ichi Anjyo. 2008. Rigid Shape Interpolation Using Normal Equations. In *Proceedings of the 6th International Symposium on Non-photorealistic Animation and Rendering (NPAR '08)*. ACM, New York, NY, USA, 59–64.
[6] P. J. Besl and N. D. McKay. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 2 (Feb 1992), 239–256.
[7] S. Biasotti, A. Cerri, A. Bronstein, and M. Bronstein. 2016. Recent Trends, Applications, and Perspectives in 3D Shape Similarity Assessment. *Computer Graphics Forum* (2016).
[8] Guillaume Dewaele and Marie-Paule Cani. 2004. Interactive Global and Local Deformations for Virtual Clay. *Graph. Models* 66, 6 (Nov. 2004), 352–369.
[9] Stefan Fröhlich and Mario Botsch. 2011. Example-Driven Deformations Based on Discrete Shells. *Computer Graphics Forum* 30, 8 (2011), 2246–2257.
[10] V. Ganapathi-Subramanian, B. Thibert, M. Ovsjanikov, and L. Guibas. 2016. Stable Region Correspondences Between Non-Isometric Shapes. *Computer Graphics Forum* 35, 5 (2016), 121–133.
[11] Angjoo Kanazawa, Shahar Kovalsky, Ronen Basri, and David Jacobs. 2016. Learning 3D Deformation of Animals from 2D Images. *Comput. Graph. Forum* 35, 2 (May 2016), 365–374.
[12] Hao Li. 2010. *Animation Reconstruction of Deformable Surfaces.* Ph.D. Dissertation. ETH Zurich.
[13] Hao Li, Bart Adams, Leonidas J. Guibas, and Mark Pauly. 2009. Robust Single-view Geometry and Motion Reconstruction. *ACM Trans. Graph.* 28, 5, Article 175 (Dec. 2009), 10 pages.
[14] Hao Li, Robert W. Sumner, and Mark Pauly. 2008. Global Correspondence Optimization for Non-rigid Registration of Depth Scans. In *Proceedings of the Symposium on Geometry Processing (SGP '08)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 1421–1430. http://dl.acm.org/citation.cfm?id=1731309.1731326
[15] K. Li, J. Yang, Y. K. Lai, and D. Guo. 2018. Robust Non-Rigid Registration with Reweighted Position and Transformation Sparsity. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1.
[16] Xin Li and S. S. Iyengar. 2014. On Computing Mapping of 3D Objects: A Survey. *ACM Comput. Surv.* 47, 2, Article 34 (Dec. 2014), 45 pages.
[17] Ya-Shu Liu, Han-Bing Yan, and Ralph R. Martin. 2011. As-Rigid-As-Possible Surface Morphing. *Journal of Computer Science and Technology* 26, 3 (01 May 2011), 548–557.
[18] OrbitVU. 2018. Alphashot360. https://orbitvu.com/alphashot-360.
[19] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional Maps: A Flexible Representation of Maps Between Shapes. *ACM Trans. Graph.* 31, 4, Article 30 (July 2012), 11 pages.
[20] Chavdar Papazov and Darius Burschka. 2011. Deformable 3D Shape Registration Based on Local Similarity Transforms. *Comput. Graph. Forum* 30, 5 (2011), 1493–1502. http://dblp.uni-trier.de/db/journals/cgf/cgf30.html#PapazovB11
[21] Fat Pebble. 2013. Fat Pebble Games. http://www.fatpebble.com/
[22] Fat Pebble. 2014. Clay Jam. https://play.google.com/store/apps/details?id=com.zynga.fatpebble.clayjam
[23] Barry JC Purves. 2016. *Stop-Motion Animation: Frame by Frame Film-making with Puppets and Models, 2nd edition.* Fairchild Books.
[24] Andrei Sharf, Dan A. Alcantara, Thomas Lewiner, Chen Greif, Alla Sheffer, Nina Amenta, and Daniel Cohen-Or. 2008. Space-time Surface Reconstruction Using Incompressible Flow. *ACM Trans. Graph.* 27, 5, Article 110 (Dec. 2008), 10 pages.
[25] Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing (SGP '07)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 109–116.
[26] Jochen Süßmuth, Marco Winter, and Günther Greiner. 2008. Reconstructing Animated Meshes from Time Varying Point Clouds. *Computer Graphics Forum* 27, 5 (2008), 1469–1476.
[27] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 2016. 3D Skeletons: A State-of-the-Art Report. *Computer Graphics Forum* (2016).
[28] Gary K. Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank Langbein, Yonghuai Liu, A. David Marshall, Ralph Martin, Xianfang Sun, and Paul Rosin. 2013. Registration of 3D Point Clouds and Meshes: A Survey from Rigid to Nonrigid. *IEEE Transactions on Visualization and Computer Graphics* 19, 7 (July 2013), 1199–1217.
[29] Unity Technologies. 2018. Unity3D. https://unity3d.com/.
[30] Art Tevs, Alexander Berner, Michael Wand, Ivo Ihrke, Martin Bokeloh, Jens Kerber, and Hans-Peter Seidel. 2012. Animation Cartography - Intrinsic Reconstruction of Shape and Motion. *ACM Trans. Graph.* 31, 2, Article 12 (April 2012), 15 pages.
[31] Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. 2011. A Survey on Shape Correspondence. *Computer Graphics Forum* 30, 6 (2011), 1681–1707.
[32] M. Vestner, Z. Lähner, A. Boyarski, O. Litany, R. Slossberg, T. Remez, E. Rodola, A. Bronstein, M. Bronstein, R. Kimmel, and D. Cremers. 2017. Efficient Deformable Shape Correspondence via Kernel Matching. In *2017 International Conference on 3D Vision (3DV)*. 517–526.
[33] Matthias Vestner, Roee Litman, Emanuele Rodolà, Alexander M. Bronstein, and Daniel Cremers. 2017. Product Manifold Filter: Non-rigid Shape Correspondence via Kernel Density Estimation in the Product Space. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 6681–6690.
[34] Kevin Wampler. 2016. Fast and Reliable Example-based Mesh IK for Stylized Deformations. *ACM Trans. Graph.* 35, 6, Article 235 (Nov. 2016), 12 pages.
[35] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel, and Andreas Schilling. 2009. Efficient Reconstruction of Nonrigid Shape and Motion from Real-time 3D Scanner Data. *ACM Trans. Graph.* 28, 2, Article 15 (May 2009), 15 pages.
[36] Yusuke Yoshiyasu, Wan-Chun Ma, Eiichi Yoshida, and Fumio Kanehiro. 2014. As-conformal-as-possible Surface Registration. In *Proceedings of the Symposium on Geometry Processing (SGP '14)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 257–267.
[37] Eduard Zell and Mario Botsch. 2013. ELASTI FACE: Matching and Blending Textured Faces. *Npar* (2013), 15–24.

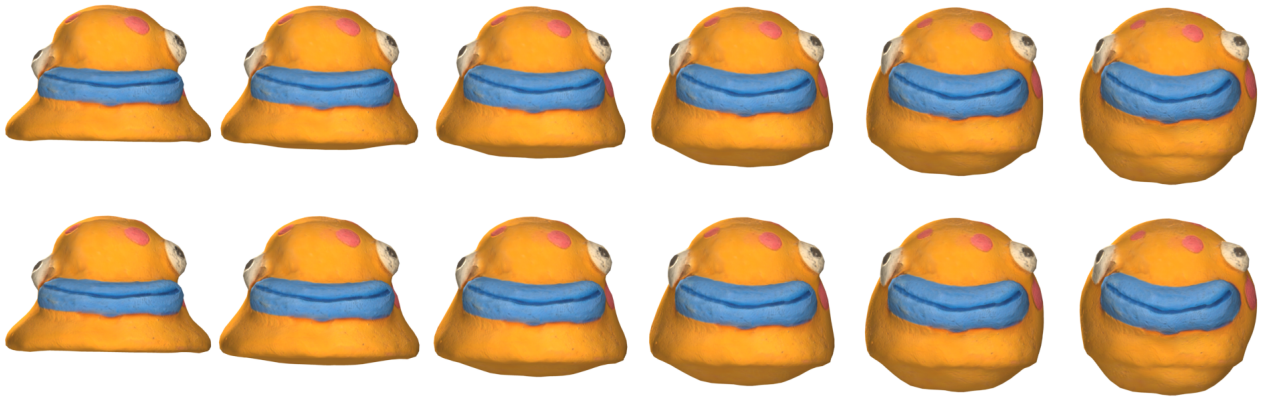

Figure 10: Example linear interpolation (top) versus ARAP interpolation (bottom) for Blob Fish character animation, between two clean scans. Although the results seem similar visually, table 1 reveals how the area distortion is smaller for the latter.

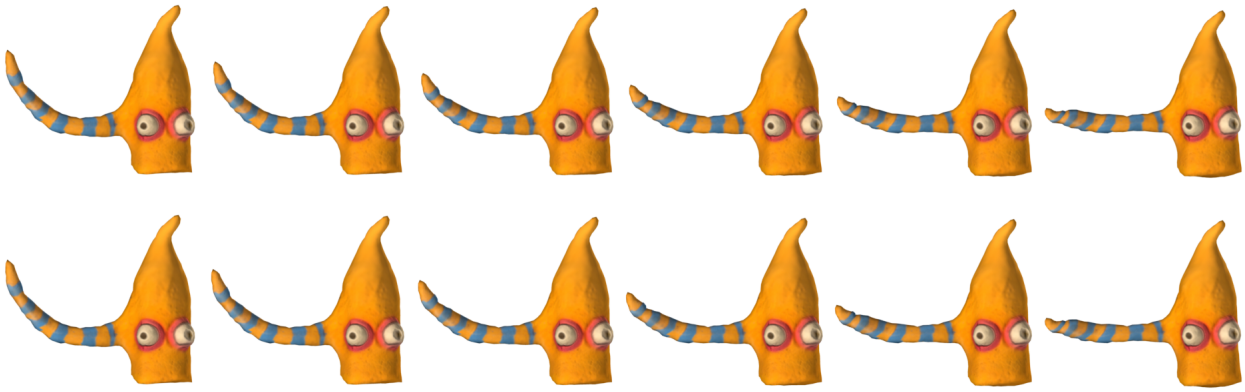

Figure 11: Example linear interpolation (top) versus ARAP interpolation (bottom) for Party Onion character animation, between two clean scans. Although the results seem similar visually, table 1 reveals how the area distortion is smaller for the latter.

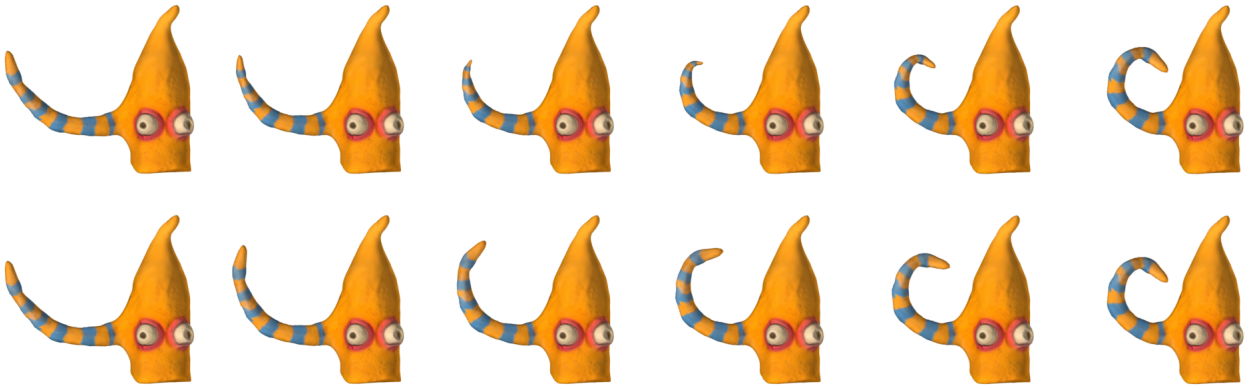

Figure 12: Example linear interpolation (top) versus ARAP interpolation (bottom) for Party Onion character animation, between two clean scans. The target shape in this scenario was created by rigging the tentacle of the source shape and rotating the joints. This example was created for clarifying the difference between linear and ARAP interpolation. Notice how from the third pose, the linear interpolation (top) shrinks the volume of the tentacle, while the ARAP interpolation (bottom) preserves the shape relatively better.

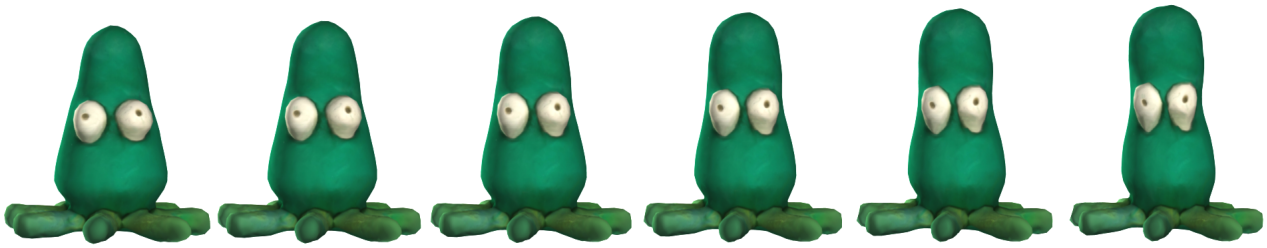

Figure 13: Example of ARAP interpolation for Tick character animation between two clean scans.

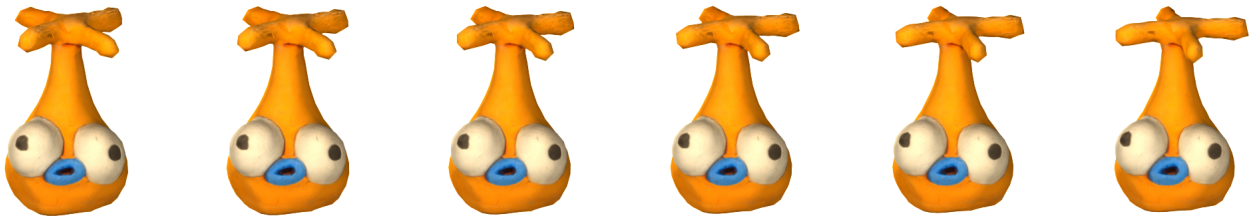

Figure 14: Example of ARAP interpolation for Hellidropter character animation between two clean scans. The animation here is subtle, notice how the propeller rotates slightly.