Props Alive: A Framework for Augmented Reality Stop Motion Animation

Llogari Casas* Edinburgh Napier University
Maggie Kosek† Edinburgh Napier University
Kenny Mitchell‡ Edinburgh Napier University

Abstract

Stop motion animation evolved in the early days of cinema with the aim to create an illusion of movement with static puppets posed manually each frame. Current stop motion movies introduced 3D printing processes in order to acquire animations more accurately and rapidly. However, due to the nature of this technique, every frame needs to be computer-generated, 3D printed and post-processed before it can be recorded. Therefore, a typical stop motion film could require many thousands of props to be created, resulting in a laborious and expensive production. We address this with a real-time interactive Augmented Reality system which generates virtual in-between poses according to a reduced number of key frame physical props. We perform deformation of the surface camera samples to accomplish smooth animations with retained visual appearance and incorporate a diminished reality method to allow virtual deformations that would, otherwise, reveal undesired background behind the animated mesh.

Underpinning this solution is a principled interaction and system design which forms our *Props Alive* framework. We apply established models of interactive system design, drawing from an information visualisation framework which, appropriately for Augmented Reality, includes consideration of the user, interaction, data and presentation elements necessary for real-time. The rapid development framework and high performance architecture is detailed with an analysis of resulting performance.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.3.8 [Computer Graphics]: Applications

1 Introduction

Stop motion animation techniques have an extensive history in the film industry. The first representative sample is, ”The Humpty Dumpty Circus” by Albert E. Smith and J. Stuart Blackton, circa 1897. This short animated film featured a circus of steady acrobats and animals in motion. The first Oscar nomination for this genre came in 1966, when Eliot Noyes Jr. redefined the technique of free-form clay animation and settled the foundations for modern stop motion pictures.

Undoubtedly, one of the major exponents of this genre is ”The Nightmare Before Christmas” directed by Henry Selick and produced by Tim Burton in 1993. In this 76 minute long movie, 227 puppets were constructed, using more than 400 different expression heads for the main character. This movie was recorded at 24 frames per second, capturing a remarkable amount of 109,400 frames for its entire length, taking 3 years to produce.

Recently, ”Kubo and The Two Strings” has introduced 3D printing advancements to mass produce puppets and faces, and consequently, accelerate the film production. However, over 66,000 faces were 3D printed, hand-detailed, post-produced, set and captured in order to be ready for capture. Therefore, regardless the advantages that 3D printing brings to stop motion, a tremendous amount of manual post-processing time remains.

This paper introduces a framework that reduces the number of 3D printed puppet components required in a traditional stop motion animation. To do so, the in-between frames of the key poses are computer-generated, having the only requirement to have the key poses physically present. To achieve this transition, we use Augmented Reality to perform deformation of the surface samples projected from the video feed. Further, a diminished reality technique is implemented to deal with virtual deformations which digitally reveal visually erroneous surfaces obscured by the physical prop.

2 Related Work

Animating real world objects in Augmented Reality requires methods of deformation, lighting consistency, diminished reality and information visualization. *Clayvision* introduced a related scheme for building scale objects in the city. The method pre-scanned a set of known locations into reconstructed meshes. The live camera feed was patched with a pre-modified version of the background image, in which the appropriate building was removed. Upon the movement of the animated mesh, the desired deformed pixels were overlaid on the camera image [1].

Object removal for diminished reality is typically attempted with inpainting techniques, and these tend to be computationally expensive and not capable of real-time performance [2][3]. However, inpainting in real-time has recently been approached through a light-weighted algorithm that apply object segmentation and frame redundancy to solve the problem of finding the occluded pixels. Here, the cost function is split into structural and appearance costs in a computationally effective algorithm [4].

Real-time texture remapping from a camera feed has been achieved through the conversion of a 2D hand-painted drawings to a 3D colored mesh [5]. In this particular approach, a static lookup map was built in order to match every single pixel of the drawing to corresponding sample locations on the mesh via fast dependent texture reads.

Cloth simulation is a particular case of mesh deformation with the challenge of resolving self collisions in real-time. Wei and Kuichao [6] employ a spatial hash function for rapid indexing of surface co-locations. In a similar concept, we introduce a vertex hash map structure, but instead for the purpose of computation of the mesh warp without repeated redundant vertex re-projections.

A framework for information visualization to display graphical information in a diversity of domains was introduced by Mitchell et al. [7]. In this particular research, an object oriented system design was mapped to different development platforms and use cases. We consider how this applies specifically to our Props Alive framework, and perhaps, in general, to interaction in Augmented Reality.

However, none of these techniques have been applied to the particular criteria of stop motion animation in augmented reality.

*e-mail: L.CasasCambra@napier.ac.uk
†e-mail: M.Kosek@napier.ac.uk
‡e-mail: K.Mitchell2@napier.ac.uk

3 PROPS ALIVE FRAMEWORK

3.1 Scene Setup

Our framework requires to have a physical puppet, either 3D printed or manually formed with its key physical poses. For instance, in the case of our Pirate head, a hand-painted Bossons plaster model. Our digital equivalent meshes are created through photogrammetric capture through a number of digital images from a variety of viewpoints. This virtual object database contains the key frame poses and the in-between artist animated frames.

A prior known location relative to the target marker it is required to track the object's location. The virtual mesh counterpart uses the tracked location to match the same position in the virtual scene and use the resulting dependent mesh vertex positions for real-time texturing. In addition, at present, background composition is limited to solid colors or non-variant textures.

3.2 Real-time Deformed Texturing and Lighting

In order to perform texturing in real-time according to the video feed, we base our approach on Magnenat [5]. However, in our scenario, the texture coordinates need to be updated on a per frame basis, in accordance to the position of the camera and the animation of the virtual mesh. Furthermore, in order to obtain coherency on the deformation of the texture, we extend this method using a hash table map between the initial rest pose vertices and the animated ones. We adopt an approach similar to that introduced by Wei and Kuichao [6].

In more detail, we place a proxy object locator in the pivot position of the physical object and we assume that no movement will be performed on that object. This approach enables an optimized lookup of the animated vertex position as gives us a $\mathcal{O}(1)$ constant access time.

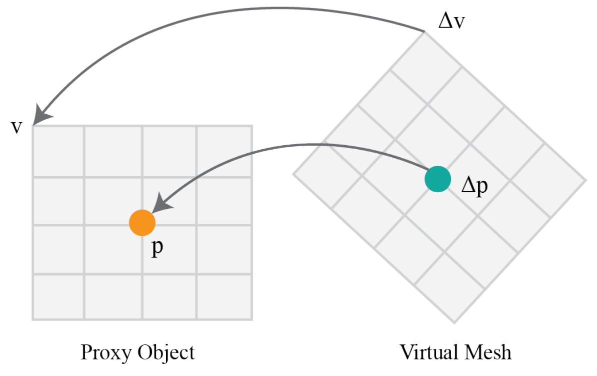

Figure 1: The texture coordinates of an animated in-between frame represented as Δv are back projected via $\mathcal{O}(1)$ hash map reference to the initial rest pose v to achieve real-time deformed texturing sampled from the real world object's vertex locations dependent on the originating physical proxy object's pivot location p.

The required model matrix is computed in a vertex shader. The matrix contains the transformation to locate the proxy virtual object for the physical prop. As Figure 1 indicates, the real-time texture coordinates of the computer generated object will match the proxy location of the physical prop even if an animation is performed on the mesh. An example of the resulting warp can be observed in Figure 2, in which image *c)*, renders the physical object texture in an animated mesh position.

Once the shifted texture coordinates are computed, a screen projection is executed, and as a result, a sample mapping between the image processed from the camera feed and the vertex locations is generated. This approach enables a seamless augmented reality appearance, as the material, lighting and shadowing of the object are updated to the virtual mesh every frame. Any changes applied to the surroundings of the virtual object will perform its expected lighting effects to the virtual mesh. However, large deformations away from the proxy object's reference mesh, degrades realism.

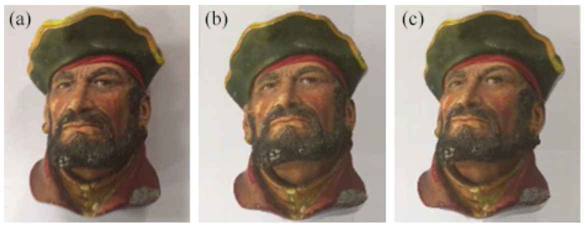

Figure 2: *a)* Real-world Bossons plaster pirate's head. *b)* real-time mesh texturing in its initial key pose showing the proxy object at rest. *c)* real-time mesh texturing of an in-between frame.

3.3 Diminished Reality

Following the approach of Herling and Broll [4], our method takes advantage of the fact that the position of the object that needs to be removed can be obtained through the pivot position of the proxy object locator. Therefore, we use that information to segment the area where the occluded pixels need to be inpainted. To do so, we compute the size of the proxy object every frame, creating a rectangle area of pixels that contains the region of the image where the physical object is placed.

Once the area that needs to be inpainted is located, boundary pixel RGB values are sampled as color probes. Each pixel that needs to be replaced is computed through the average RGB values of its boundary counterparts in the X and Y axes.

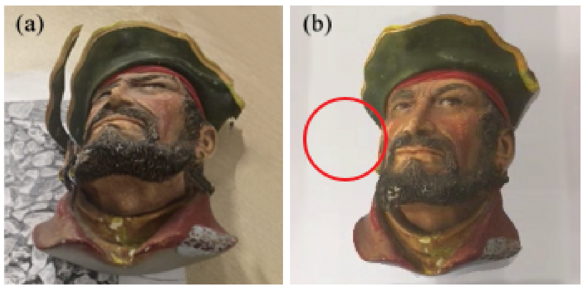

Figure 3: *a)* Animated mesh without a diminished technique. *b)* Animated mesh with a diminished plane implemented.

The inpainted texture is applied to a procedural plane that contains the size of the physical prop. As Figure 3 indicates, the real-world prop would have been shown in the circled area of the image *b)* if a diminished background technique would have not been applied.

4 ARCHITECTURE AND IMPLEMENTATION

We base our system design on the Framework for Information Visualisation [7], which generically considers user interaction with data. This framework unifies models of Model-View-Controller (MVC) with others, such as Presentation-Abstraction-Control (PAC) or User Interface Management Systems (UIMS).

Due to the fact that the user interacts with data through an Augmented Reality scenario, we need to further develop some concepts from the Information Visualisation Framework [7]. For instance, the *observation* process of the system's model of the user observing the visualisation, necessarily includes the camera sensor providing the video feed to augment. In addition, the system's *performance* applies to the processing of the scene *database* that represents the real object. In our specific example, the Pirate head, and modifies this information to produce the animated Augmented Reality visualisation. The major elements of the Props Alive framework deals with the *performance* processing the scene *database*.

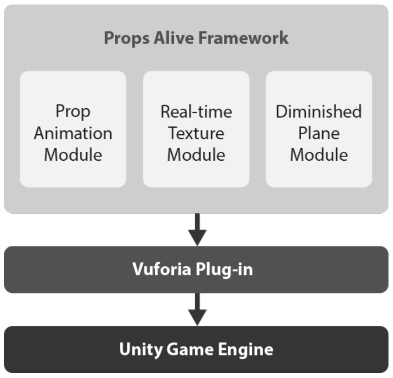

Figure 4: Props Alive framework's structure and layered implementation with the Vuforia Augmented Reality plugin within the Unity game engine.

To achieve this real-time interactive system, we implemented our framework with Unity game engine. This technology provides a cross-platform solution that enables development for an extended variety of devices with rapid prototyping featuring scripting and a component based architectural design. In this particular case, our framework runs on contemporary mobile devices as deployable apps. In addition, Vuforia is used to support tracking of the real-world object in the scene and delivers the video image and pose registration parameters to our Props Alive framework components.

As Figure 4 indicates, the Props Alive framework is built on top of Vuforia and Unity core libraries and split into three core modules: Prop Animation, Real-time Texture and Diminished Plane modules. A standalone C# script along with its associated Shader are used.

In order to deploy with Vuforia, a marker target needs to be created, activated and attached to the main scene camera. In our specific framework, we opted for an Image based marker. This marker provides a by default enabled component that handles event management in real-time for specific Augmented Reality events. As Figure 5 indicates, our framework uses those events in combination with Unity provided ones to deliver Props Alive.

4.1 Prop Animation Module

In order to animate our real-world puppet, we captured the physical object through photogrammetric digital images. The outcome mesh was cleaned and retopologised manually using Autodesk Maya.

Several technical solutions are currently available to animate a 3D mesh. Nevertheless, not all of them are computationally effective and capable of a high performance in a real-time domain. For instance, Point Cache Animation provides a high flexible solution, due to the fact that all the vertices of the mesh are baked to the exported file on every frame. However, this is extremely costly in a real-time scenario, as the size of the generated file is dependent to the amount of vertices and length of the animation. To overcome this constraint, we implemented a weighted skeleton bone system to the puppet and performed the resulted animation on the GPU. This approach only requires to store the transformation values of its bones on each frame to animate the entire mesh, and consequently, avoids storing large amount of data on the exported file.

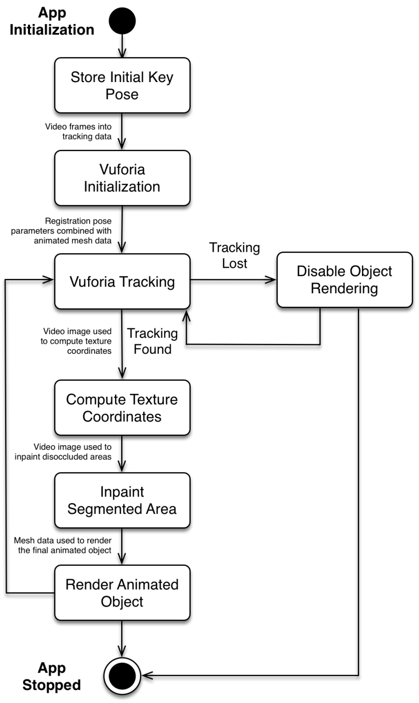
App Initialization
Store Initial Key Pose
Video frames into tracking data
Vuforia Initialization
Registration pose parameters combined with animated mesh data
Vuforia Tracking
Tracking Lost
Disable Object Rendering
Video image used to compute texture coordinates
Tracking Found
Compute Texture Coordinates
Video image used to inpaint disoccluded areas
Inpaint Segmented Area
Mesh data used to render the final animated object
Render Animated Object
App Stopped

Figure 5: Props Alive framework's statechart diagram annotated with data structures used in the Augmented Reality mesh deformation.

The Shader renderers the result of the multiplication between the initial vertex pose position (v), the weighted value for each bone (w_i) and the transformation matrix for each bone (M_i). Thus defining v' as, $v' = \sum_i^n w_i M_i v$.

4.2 Real-time Texture Module

Achieving real-time texturing according to the video feed requires us to compute the texture coordinates on each frame for every vertex. Therefore, the performance of our framework is correlated with the complexity of the given mesh. In our implemented pirate's head, we use a low poly version of the mesh, which contains an approximately 8,000 vertices. In order to avoid unnecessary operations in real-time, we store the initial key pose before initializing the Framework, as Figure 5 indicates. This pipeline reduces the operational cost per vertex, as the only calculation that is performed in the Vertex Shader on the GPU is a multiplication between the current location of the vertex and the original pose position. This operation resolves the real-world location of the physical puppet to retrieve the texture sample from the video feed.

4.3 Diminished Plane Module

Inpainting traditionally requires expensive algorithms that were not capable of real-time performance [2][3]. However, in our implementation, we follow a novel real-time capable approach from Herling and Broll [4]. To do so, we only inpaint the segmented area where the occluded pixels are located with complexity of $\mathcal{O}(N \times M)$. Consequently, we intend to keep N and M, which define the segmented area, as low as possible through object segmentation. N and M values are also dependent to the position of the physical object relative to the camera. If the camera gets closer to the segmented object, those values would increase due to the bigger presence of the object in the screen. Conversely, those would decrease if the object gets further away from it. Hence, the performance of our diminished plane prototype is linked with the size of the segmented area that needs to be inpainted every frame.

In addition, in our implementation of the inpainted texture, we create a 24-bit buffer of $(N \times M)$ size. We allocate 8 bits per each RGB component. The buffer is released from the memory once it is assigned to the diminished plane.

5 Prototype Results using the Framework

The preliminary results enabled us to analyze the performance of our Framework. As Figure 6 indicates, the area that contains a peak over the time in the rendering section defines the framework's state where the marker is detected. Render time is steady when the framework's reaches this tracked state. CPU's usage also increments and varies depending of the amount pixels that need to be inpainted on the diminished plane. We highlight the observation, that the ease of implementation through scripting of mesh and image processes, did not impede overall system performance and effectiveness of results.

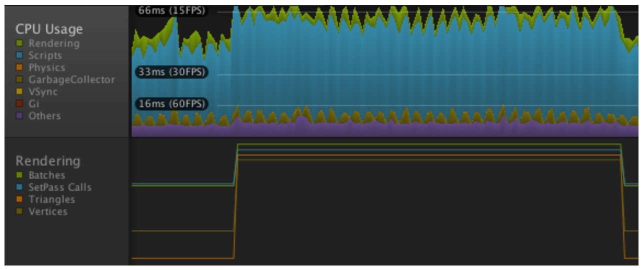

Figure 6: Performance of the CPU and rendering calls made when the scene database is detected and tracked through our Framework.

As Figure 7 indicates, preliminary results visually suggest that proposed method could achieve good quality and performance on the in-between frames between physical key poses. However, developed method needs to be compared against traditional stop motion techniques to be able to analyze and compare the benefits and the efficiency of our approach. For instance, to compare the deformed surface appearance accuracy against ground truth 3D printed frames.

6 Conclusion and Next Steps

Our next steps will be focused on spotting the constraints and limitations in which our framework could deliver efficient results. For instance, real-time computing constraints, such as evaluating the maximum number of props that could be using this method at the same time in the same scenario, or animation constraints, such as understating the amount of translation units and rotation degrees that can be performed on top of the physical prop without incurring degraded quality under texture deformations.

In addition, comparison between traditional stop motion techniques and our approach will be performed in order to obtain qualitative and quantitative results, in terms of performance, costs and visual outcomes. Next steps anticipate work closely with traditional stop motion artists in order to provide an accurate and user-friendly solution to their needs.

Even though our prototype results look visually acceptable, accurate conclusions can not be performed until more data is collected and compared to traditional stop motion techniques. However, the framework presented provides a strong basis for this further exploration and experimentation.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 642841.

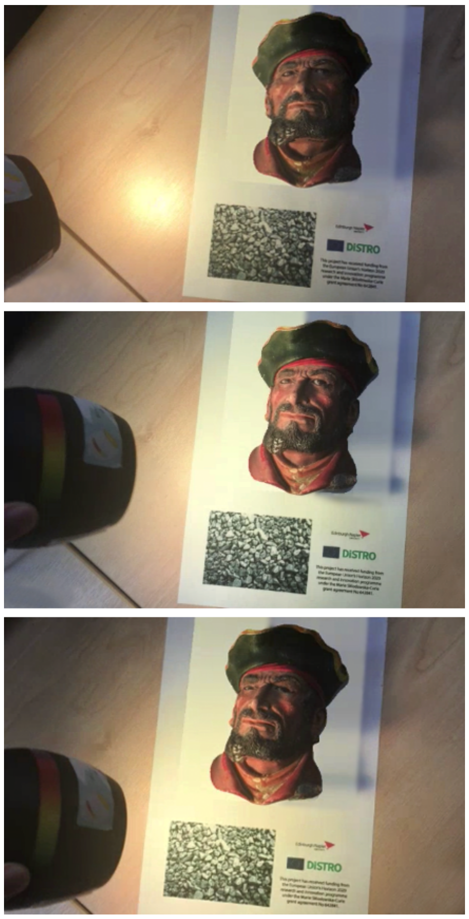

Figure 7: Animated sequence of a real-world Bossons pirate's head.

References

[1] Yuichiro Takeuchi and Ken Perlin. Clayvision: The (elastic) image of the city. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2411–2420, New York, NY, USA, 2012. ACM.

[2] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *Trans. Img. Proc.*, 13(9):1200–1212, September 2004.

[3] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[4] Jan Herling and Wolfgang Broll. Pixmix: A real-time approach to high-quality diminished reality. In *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, ISMAR '12, pages 141–150, Washington, DC, USA, 2012. IEEE Computer Society.

[5] S. Magnenat, D. T. Ngo, F. Znd, M. Ryffel, G. Noris, G. Rothlin, A. Marra, M. Nitti, P. Fua, M. Gross, and R. W. Sumner. Live texturing of augmented reality characters from colored drawings. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1201–1210, Nov 2015.

[6] Zhao Wei and Yu Kuichao. Improved collision detection for cloth simulation system. In *World Automation Congress 2012*, pages 1–4, June 2012.

[7] Jessie B. Kennedy, Kenneth J. Mitchell, and Peter J. Barclay. A framework for information visualisation. *SIGMOD Rec.*, 25(4):30–34, December 1996.