Datamoshing with Optical Flow

Chris Careaga
Simon Fraser University
Burnaby, BC, Canada

Mahesh Kumar Krishna Reddy
Simon Fraser University
Burnaby, BC, Canada

Yağız Aksoy
Simon Fraser University
Burnaby, BC, Canada

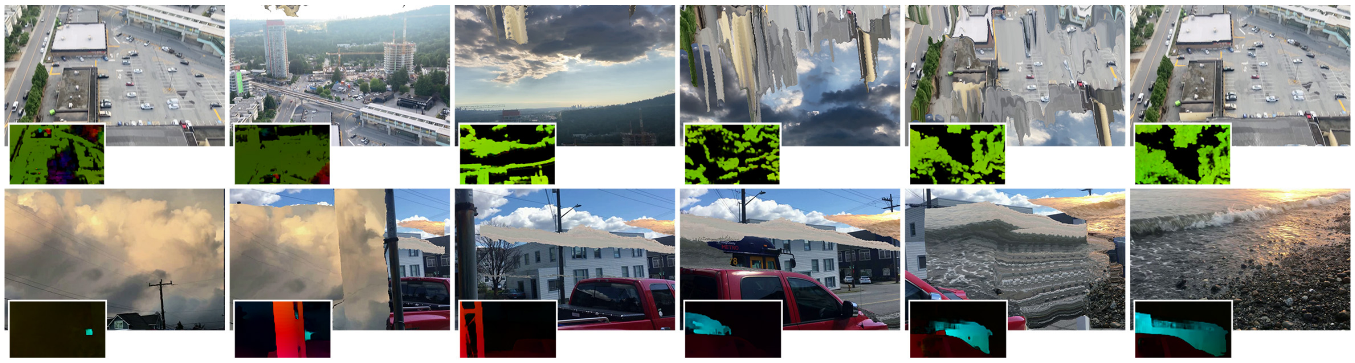

Figure 1: We propose an algorithm to perform data moshing using optical flow. Our algorithm is general and has various applications. Using multiple video sequences, we can create perplexing video transitions where the visual information of one video is distorted or constructed using the motion of another (bottom row). Using a single video clip, we can create seamless looping GIFs with interesting glitch art effects. (top row)

ACM Reference Format:
Chris Careaga, Mahesh Kumar Krishna Reddy, and Yağız Aksoy. 2023. Datamoshing with Optical Flow. In *SIGGRAPH Asia 2023 Posters (SA Posters '23), December 12-15, 2023.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3610542.3626133

An important aspect of video editing is transitions between two consecutive scenes. Most video transitions are simplistic such as jump cuts or fade-ins. With the current rise of short-form video content on social platforms, more creative forms of transitions are an important aspect of creating more engaging content. In this work, we explore the use of optical flow to study a unique form of transition between two distinct scenes inspired by the decades-old technique of using MPEG compression data called *data moshing*.

Glitch art is the practice of corrupting digital media in order to create aesthetically pleasing effects. One specific form of glitch art called data moshing involves carefully altering the encoded data of MPEG videos such that motion information is applied to the incorrect visual data. This creates perplexing transitions between distinct shots, where the visual data of one scene appears to follow the motion of the next. This effect has existed for well over a decade and has gained interest for its use in popular media. Despite its popularity, data moshing has been predominantly carried out by altering compressed video data, which is inflexible and requires specialized software and expertise.

In this work, we propose a simple method for emulating the effect of data moshing, without relying on the corruption of encoded video, and explore its use in different application scenarios. Like traditional data moshing, we apply motion information to mismatched visual data. Our approach uses off-the-shelf optical flow estimation to generate motion vectors for each pixel. Our core algorithm can be implemented in a handful of lines but unlocks multiple video editing effects. The use of accurate optical flow rather than compression data also creates a more natural transition without block artifacts. We hope our method provides artists and content creators with more creative freedom over the process of data moshing.

1 APPROACH

We formulate our approach as an operation performed on two distinct video clips. For simplicity, we first describe a single step of our data moshing process. The input to our algorithm is an image c, and a video clip V consisting of multiple frames:

$$V = \{v_1, v_2, \dots v_t\} \tag{1}$$

The output is a data-moshed version of c using the motion of V. We first compute optical flow for V in order to define the perceived motion between each frame. We denote this as an operation between two frames called $flow$. We use the optical flow from V to displace the pixel values in c using a remapping operator we denote as $remap$:

$$r_1 = remap(c, flow(v_1, v_2)) \tag{2}$$

The remapped image r_1 may have holes where new image content entered the frame between v_1 and v_2. To determine these regions

SA Posters '23, December 12-15, 2023, Sydney, NSW, Australia
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0313-3/23/12.
https://doi.org/10.1145/3610542.3626133

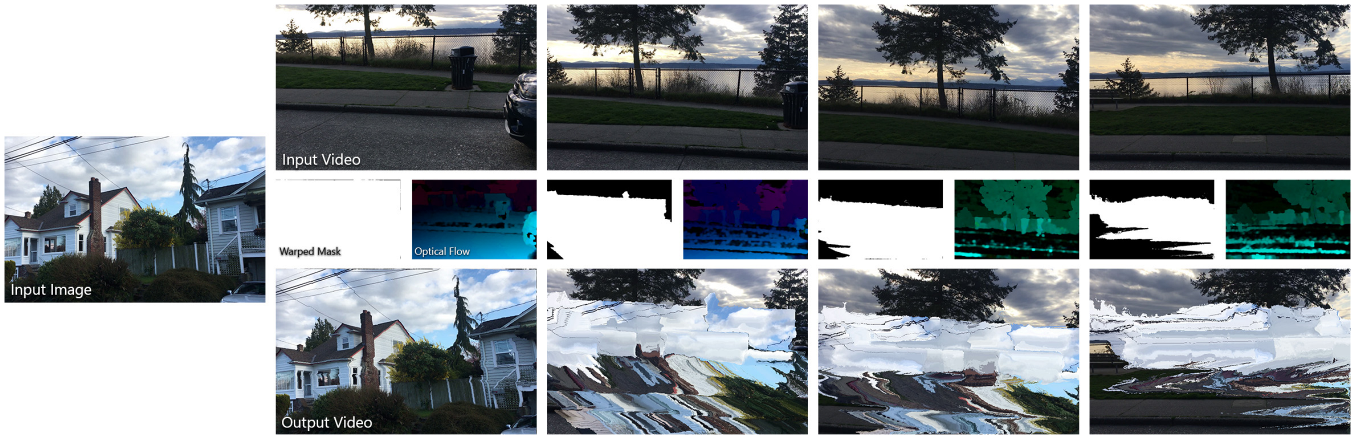

Figure 2: Our algorithm in its simplest form operates on an image and a video sequence. We compute the optical flow of the video sequence and use the motion to warp the pixels of the input image. We keep track of any holes created from warping using a binary mask, the warped image is composited with each from of the input video to create a data moshed output video.

we additionally warp an image of all ones, leaving the holes as zeros:

$$m_1 = remap(\mathbb{1}, flow(v_1, v_2)) \tag{3}$$

Finally, to create our final data moshed frame, we fill any holes in r_1 using the image content from V with m_1 as a mask:

$$d_1 = (m_1 * r_1) + (1 - m_1) * v_2 \tag{4}$$

We then define the rest of the frames iteratively from d_1:

$$\begin{aligned} r_i &= remap(d_{i-1}, flow(v_i, v_{i+1})) \\ m_i &= remap(m_{i-1}, flow(v_i, v_{i+1})) \\ d_i &= (m_i * r_i) + (1 - m_i) * v_{i+1} \end{aligned} \tag{5}$$

This results in a new set of frames representing a data-moshed transition between the visual content of c and the frames of V. The optical flow can be computed using any off-the-shelf algorithm. In this work, we experiment with both the traditional algorithm by Farnebäck [2003], and a network-based approach by Teed and Deng [2020]. We perform the remapping of pixel values using the Kornia library [Riba et al. 2020].

2 APPLICATIONS

The general algorithm described in Section 1 can be utilized to perform multiple video transition effects.

Forward Mode. The default use case for our approach is to perform a transition between two distinct shots. Given two videos, we use the last frame of video 1 as c and use video 2 as V. This results in a combined video where video 1 appears to visually freeze and then be warped and mixed with the visual content of video 2. As our supplementary video demonstrates, we find the most interesting use of the forward mode to be transitioning between two scenes with a panning motion.

Reverse Mode. By playing the result of the forward mode in reverse, we can give the illusion that the visual content of video 1 coalesces from randomness before playing regularly. In this case, the resulting video order and motion are backward, therefore we can reverse the video and frame ordering before running our algorithm in forward mode to achieve this effect while maintaining temporal coherency. The reverse mode gives the most pleasing results when transitioning into a scene with large and localized motion, where the new scene appears to unexpectedly come together from seemingly random motion.

Looping. Instead of utilizing two distinct scenes, our algorithm can be run on a single video clip. For a given video, we can set c to the first frame of the video, and run our algorithm in reverse mode. This results in a data-moshed video where the first frame is reconstructed as the video progresses resulting in a seamless looping video. We find the looping mode to be most interesting for short-form videos that has gained immense popularity in recent years with the emergence of TikTok followed by YouTube Shorts and Instagram Reels. In such platforms, the videos are played in a loop by default, for which our data moshing transition represents an effortless yet interesting way to create a natural transition.

3 CONCLUSION

In this work, we proposed a simple algorithm based on optical flow that can emulate the effect of data moshing. We show that our approach can be applied in multiple different ways to perform interesting video edits. We believe our algorithm allows for more creative freedom than traditional data moshing techniques meaning it can be more easily used by a wide variety of artists and content creators. We invite the reader to view our supplementary video where we demonstrate our applications. Our poster presentation will feature a video player to fully communicate our work.

REFERENCES

Gunnar Farnebäck. 2003. Two-Frame Motion Estimation Based on Polynomial Expansion. In *Image Analysis*.
E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. 2020. Kornia: an Open Source Differentiable Computer Vision Library for PyTorch. In *Proc. WACV*.
Zachary Teed and Jia Deng. 2020. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *Proc. ECCV*.