Seam-hiding for Looping Videos

James Durrant
james.durrant.13@ucl.ac.uk
University College London

Gabriel Brostow
g.brostow@cs.ucl.ac.uk
University College London

ABSTRACT

The proposed algorithm creates a seamless looping video clip from a real world video of an *almost*-cyclic motion. For a video that has repeating motion, such as a person on a trampoline, the first and last video frames may not precisely line up, even though the content is very similar. Playing back the video in a looping fashion can cause the re-start transition to jump out and appear discontinuous, both spatially and in terms of object velocity. Most work on video looping has sought to find the best re-set point in a longer video's timeline, but we start there, and modify the frames to hide the jump point.

Our approach essentially fits a curve to the (x, y) and RGB coordinates of points in the scene, and then smooths those curves using gradient domain optimisation. We address important qualitative factors, balancing smoothness against preservation of the original trajectories/curves. Our modular system also incorporates video stabilisation and inpainting, to cope with more dynamic videos.

For most videos within our scope, we found that automatic seam-hiding is succesful. For the cases in which the proposed system cannot satisfactorily produce a seamless loop, we hope our framework can be modified with improved components to achieve better results in the future.

CCS CONCEPTS

• **Computing methodologies → Motion processing**;

KEYWORDS

Video Looping, Animated gifs, Video Manipulation, Motion Compensation

ACM Reference Format:
James Durrant and Gabriel Brostow. 2017. Seam-hiding for Looping Videos. In *CVMP 2017: 14th European Conference on Visual Media Production (CVMP 2017), December 11–13, 2017, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3150165.3152766

1 INTRODUCTION

Our aim is to automatically convert a video clip that *almost* loops into one that appears to loop seamlessly. The point in time where the video repeats should be indistinguishable from any other point. After modification, such an image sequence can play endlessly, as sometimes seen in animated gif's.

Previous work on creating looping videos has focused on finding a transition point in time that allows the video to jump backwards and then continue playing without a noticeable cut. This requires that the frames at that transition could plausibly be adjacent frames, limiting these methods to videos in which near-identical frames reoccur. Our work extends this to a much wider class of videos by also adjusting the motion either side of this transition, allowing these seam frames to differ. The only assumption we make is that the content is similar enough that correspondences between these frames can be automatically detected.

Videos within this scope face two inherent difficulties. First, we are trying to remove or reduce jarring changes at the seam as much as possible, so we must also be careful not to *introduce* noticeable changes. Second, correcting for motion means manipulating the image to rearrange parts of the content. Whenever part of an image is moved, it leaves behind a region of undefined pixels. Therefore, an important aspect of the project is to handle these disoccluded regions effectively.

2 RELATED WORK

2.1 Loop detection

Previous work on looping videos has generally focused on the problem of analysing a long sequence of footage and finding subsequences that lend themselves to looping, with a minimally visible seam. There have been a number of different approaches in this area, each with their own set of assumptions and limitations.

Video textures [16] assumed that there exists a re-ordering of input frames, such that motion would be generally preserved but without large differences between adjacent frames. Video textures were proposed to probabilistically switch between frames during playback, with fixed length loops also being possible. Due to their choice of distance metric and subsequent synthesis, the class of videos that could be handled by this method is relatively small.

Liao *et al.*[11, 12] make the assumption that every pixel in the video has a defined looping period. This means that at some point, the same pixel value will occur again at the same spatial location, and so looping the video is a matter of having these looping periods synchronise. They extend this to have the level of dynamism in the clip be adjustable by the user.

Rather than looking at the problem of looping in isolation, methods such as those by Bai *et al.*[2] and Joshi *et al.*[8] focus on creating a wider class of videos, including non-linear videos. They make use of semantic information as well as user interaction [8] to control the output. Probably closest to our own, Sevilla-Lara *et al.*[17] also focuses on unconstrained video, but with the caveat that they expect the video to have a single, dominant foreground object. They argue that provided this foreground object loops smoothly, it is possible to have a less smooth blending or warping in the background without it being too jarring to the user. A subset of their dataset is compared against in Section 10.3.

2.2 Loop synthesis

Although there isn't much work on loop synthesis itself, it is still a component in all of the work described so far. Video textures

CVMP 2017, December 11–13, 2017, London, United Kingdom
2017. ACM ISBN 978-1-4503-5329-8/17/12…$15.00
https://doi.org/10.1145/3150165.3152766

make use of blending and morphing when making transitions to cover the seam. This is similar to the goal of this project, however, their original frames are generally more closely aligned, and the results they get still have a noticeable seam. Tompkin *et al.*[18] use user input to only allow movement in particular parts of the frame. For the parts that are still moving they linearly interpolate new frames at the end of the loops using optical flow. Liao *et al.*[11] use Poisson blending after finding looping periods to again try to hide the transition. In their case, they also anticipate this during the loop detection phase. By knowing that particular types of transitions can be adequately masked by the Poisson blending, they can afford to make loops that might not seem advisable otherwise.

Kwatra *et al.*[9] show how loop synthesis can be seen as an application of their Graphcut Textures. They assume that a video can be overlapped with itself in time to create a loop. By finding a point in time where the transition between these two clips is minimised, they therefore have a continuous smooth loop. Since the requirement is that the two ends of the clip must be aligned, this is generally most suited for videos that include stochastic effects such as fire and waves. In these cases, content that is perceptually similar is likely to repeat even in fairly short sequences.

2.3 Video stabilisation

Video stabilisation is often framed as a generic problem of removing as much camera motion as possible, thereby rendering the video stable. Since this is not a restriction for our project, we look more at instances of guided stabilisation approaches. Grundmann *et al.*[4] show how giving prior knowledge of the camera path as having either constant location, constant velocity, or constant acceleration, can make handheld videos appear to have been filmed using practical stabilisation techniques.

Bai *et al.*[1] use a form of guided stabilisation that stabilises different parts of the frame differently. They use a mixture of completely static sections as well as regions that have had their large scale motion removed (but still have local movement) to create 'cinemagraph'-like videos. There is no requirement in their case that these videos should loop, however.

3 PIPELINE

Figure 1 shows our pipeline, highlighting the most important aspects: the optimisation to obtain smoothly looped pixel displacements, and the rendering stage to produce new frames using these displacements. These also represent our main contributions.

3.1 Optimisation

This step takes as input a sequence of frames along with sets of optical flow-based correspondences between them. Its aim is to produce a set of displacements for every frame. When pixels are projected along these displacements, the resulting image should produce smooth motion between the two frames at the seam, and preserve the motion as much as possible in the rest of the sequence. Details of this stage are described in Sections 5 and 6. These corrective motion models, as well as the frames and their original correspondences, are then passed on to the rendering stage.

For frames close to the seam we used dense correspondences, and for frames further from the seam we use sparse correspondences. We use sparse correspondences since the corrective motion required for those frames is generally low frequency, and can be captured by a coarse motion model.

3.2 Rendering

The goal of this stage is to exploit the information from the previous step, and enhance the footage to produce a final output sequence of the same length and dimensions as the input. Details are given in Section 7, but the three main steps are:

(1) Warping: using the original frames, the pixel values are projected according to the displacements to produce a new image.
(2) Inpainting: due to the nature of the warping step, there are likely to be pixels in the new image that have not been assigned a colour. This step fills in these regions with plausible values that are consistent with the rest of the sequence.
(3) Colour consistency: finally, this step ensures that colours change smoothly across the seam. This accounts for two cases: first, the case where two corresponding pixels represent the same real world location that has changed appearance either due to differences in lighting or actually changing colour. Second, the case when the system has not managed to align the content in the seam frames completely, and so regions that are adjacent in time actually represent different objects.

4 LOOPING IN ONE DIMENSION

4.1 Problem definition

We first illustrate the situation in one dimension, before proceeding with our solution for video. We assume that the input is a time series with a visible seam, where the last point should connect to the first point. If this sequence was manipulated such that the two points were very close or even coincided, we would have a sequence that is *continuous* but does not necessarily appear consistent in terms of motion smoothness. Therefore, we also target C_1 continuity, as shown in Figure 2.

We can think about the problem as being made up of an original, perfectly looping signal that has been affected in some way that causes the seam to appear; an example can be seen in Figure 3. We do not intend to model this perturbation explicitly, instead we only aim to reduce the disparity between the two ends of the data. We make the underlying assumption, however, that the shape of the signal remains perceptually similar despite the perturbation. In Figure 3, even though the original signal has been changed such that there is a disparity in the middle, it still appears to have roughly the same shape.

Doing this operation in reverse - manipulating the signal such that the boundaries are at new locations but the shape remains perceptually similar - is the idea behind Poisson Image Editing [15]; this approach is also a good fit for our space-time problem [3].

By itself, this problem is under-determined and so we must provide boundary constraints. This can be done by specifying fixed points on the curve that effectively 'anchor' the solution. Given that we seek alignment between the two ends of a curve, it makes sense to specify these boundary points as having the same location. We could choose this location to be either of the original points, or

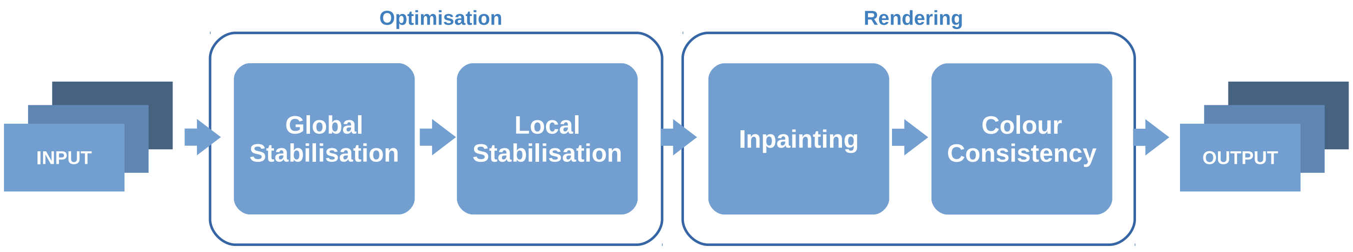


Figure 1: The full pipeline of our system. Unlike many video-looping algorithms, ours focuses just on concealing the re-set point in the timeline of a looped video. The optimisation stage modifies the actual content of many frames, to make the start and end of the input sequence compatible with each other. The overall method works on many videos precisley because the components were selected for their simplicity and reliability.

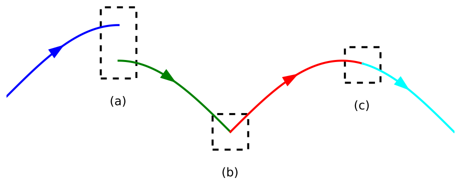


Figure 2: Intersections with different levels of continuity: (a) Discontinuous, (b) C_0 continuous, and (c) C_1 continuous.

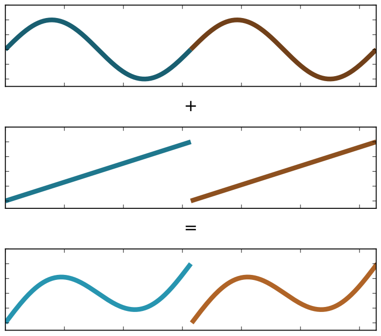


Figure 3: A one dimensional function with a seam, shown twice to show the disparity. It is formed from a periodic function (top) and an unwanted perturbation function (middle).

even their midpoint. For our system, we choose to instead imagine our curve as being shifted in time so the seam is now directly in the middle frame, away from both ends. This is equivalent in terms of what we can achieve, but we now do not have to choose a location for the boundaries, since they already have the same, known location due to coming from the same part of the sequence.

4.2 1D Solution

As demonstrated by the dotted line in Figure 4, naively solving the Poisson equation such that the signal becomes C_1 continuous at the seam produces undesirable results. In this case the solution has the endpoints that we wanted, and appears to have a similar shape. However, in making the seam boundary smooth it has also

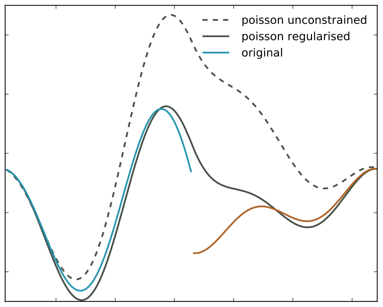


Figure 4: A naive, unconstrained solution using only Poisson's equation to correct the sequence shown produces a new sequence that is smooth and respects the shape of the original, but deviates massively. In contrast, a regularised solution as produced by optimising Equation 1 balances continuity with displacement from the original data.

deviated dramatically from the original curve. We therefore impose the restriction that we want the looped curve to stay close to the observed curve, which is shown by the solid line in Figure 4; this serves to regularise the solution.

Combining this together we state our objective function as:

$$E = ||A\mathbf{h} - (\mathbf{f} - A\mathbf{y})||_2^2 + \lambda||\mathbf{h}||_2^2, \quad (1)$$

where A is a matrix that computes a finite difference approximation to the Laplacian and $\mathbf{f}$, $\mathbf{y}$, and $\mathbf{h}$ are vectors representing the desired Laplacian, the observed data, and the displacements, respectively. The parameter λ is the regularisation parameter that constrains the displacements to be small.

Constraining that $\mathbf{f}$ should stay the same as much as possible, it is set to take the value of $A\mathbf{y}$ for all points in time except at the seam. The resultant vector $\mathbf{f'} = \mathbf{f} - A\mathbf{y}$ is therefore zero almost everywhere and can divided up into two vectors,

$$\mathbf{f'} = f_1'\mathbf{e}_1 + f_2'\mathbf{e}_2, \quad (2)$$

where 1 and 2 refer to the frames either side of the seam, and $\mathbf{e}_i$ is a vector whose i^{th} value is one, and is zero otherwise. From this, the standard ridge regression solution for Equation 1 can be divided into two equations,

$$\mathbf{h}_1 = (A^T A + \lambda^2 I)^{-1} A^T \mathbf{e}_1, \quad (3)$$

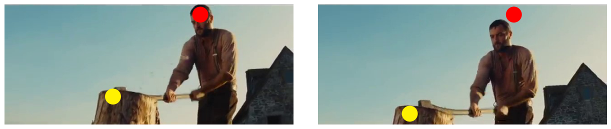


Figure 5: Two frames from a video. The red dots show the same location in both images, but represent different objects. The yellow dots both represent the same object, but are at different locations in their respective frames.

$$\mathbf{h}_2 = (A^T A + \lambda^2 I)^{-1} A^T \mathbf{e}_2, \tag{4}$$

where $\mathbf{h} = f_1'\mathbf{h}_1 + f_2'\mathbf{h}_2$. This means that every solution for this form of the problem can be solved as a weighted combination of two basis solutions $\mathbf{h}_1$ and $\mathbf{h}_1$, given only the current Laplacian at the seams Ay and an estimate f of what the Laplacian should be. Therefore, rather than solving this problem from scratch for every distinct curve, we can instead precompute the solution for a fixed sequence length and cheaply get the solution for any number of similar 1D problems. Additionally, since the solution depends only on the values of f at the seam - and not for any other frames - then in theory the displacement vectors can be computed for all frames in parallel. In practice we propagate correspondences from the seam to the other frames, making it a sequential process, but these correspondences could also be implemented in the parallel case for improved efficiency.

4.3 Extending looping to videos

We have seen how one dimensional time series can be seamlessly looped, but this approach cannot be applied directly to videos as it is. If a video is treated as a three dimensional volume then neighbouring pixels in time will not correspond to the same real world location, and it will not make sense to propagate information between them. This is illustrated in Figure 5, which shows equivalent points with different coordinates, and unrelated points with the same coordinates.

We can account for the disparity between frames by making connections not between pixels with the same (x, y) locations but instead by using their optical flow correspondences. In this way the video volume is no longer a regular grid but instead a graph where each *frame* is a grid, and edges that join these grids correspond to the optical flow between those frames. This formulation has previously been used by Bhat *et al.*[3] as a way to extend gradient domain fusion to videos.

If we consider each point in world space individually, then we have a collection of problems that can all be solved by the 1D solution. Since it was shown in Section 4.2 that we can precompute a general solution, we can solve all of these sub problems in $O(n)$ time. If we solved all of the problems separately we would lose correlation between motion in the video. We propose to use a hierarchical approach that consists of a global stabilisation step, and a subsequent step solving the many sub-problems at the pixel level whilst preserving motion correlation.

5 GLOBAL STABILISATION

Looking only at local neighbourhoods it is tricky to model global movement such as camera motion, that affects all pixels in the frame. Using a parametric transformation model not only allows for capturing these global correlations but is also more robust than trying to achieve the equivalent motion using dense displacements. Unlike in [17] we are not making the assumption that our videos are made up of a foreground and a background. Our global stabilisation instead accounts for the case where a dominant motion affects the frame as a whole. It does not matter whether this is due to movement of the camera or the foreground, as we treat these situations equally. The only assumption is that if there is a single dominant motion, then it can be captured by the estimated transformation.

5.1 Using the 1D solution

Unlike traditional video stabilisation we are only stabilising with respect to errors between two frames. We have seen from the 1D looping solution that this can be expressed as a weighted sum of the Laplacians at the seams, and hence does not require any other information from throughout the sequence. This result means that we can perform the stabilisation for long sequences with very little overhead compared to shorter ones. The overall process is therefore to find some notion of the Laplacians at the two seams and then apply this to the 1D solution. These output transformations can then be used to warp each frame accordingly.

5.2 Interpolating transformations

To compute the Laplacian, we devise a weighted combination of transformations, *i.e.*an interpolation between them. Given two transformations, one can imagine what interpolating them may look like, but it is not obvious what this means quantitatively.

We look at global transformations given by the form:

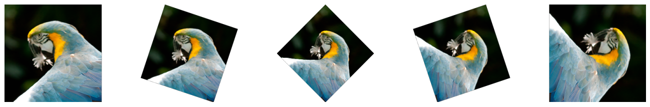
$$\lambda \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{5}$$

where $\mathbf{w} = [u, v]^T$ and $\mathbf{x} = [x, y]^T$ are the points before and after the transformation respectively. The transformation is described by the matrix Φ. This form covers a number of transformations, including the rigid transformation, affine transformation, and projective transformation.

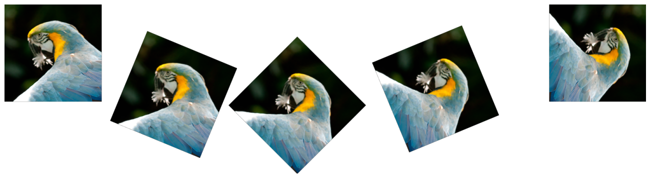
We tried multiple approaches for interpolation, to see which produced the best results, illustrated in Figure 6. We found that interpolating the transformation matrices or transformation parameters could work successfully, but only for specific cases.

Instead, we can take advantage of the fact that the set of transformations we are looking at forms a Lie group. We can map transformations to their corresponding Lie algebra vector space, linearly interpolate them, and then map them back to the original space. The result of this can be seen in Figure 6c and produces the most natural looking transformations we have seen. This procedure only guarantees to hold for relatively small transformations, but as can be seen in the example it is sufficient for the types of transformations we are dealing with.

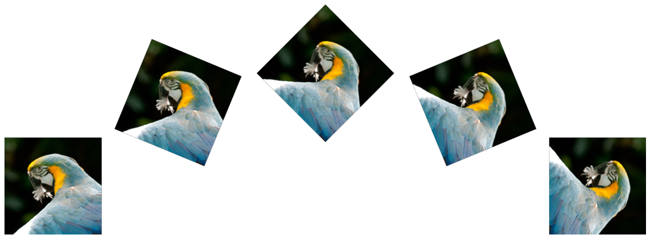
Putting this together with the 1D solution, the global stabilisation is computed by mapping the transformations into their Lie algebra



(a) Linearly interpolating between transformation matrices.



(b) Linearly interpolating the transformation parameters.



(c) Linearly interpolating in the Lie algebra.

Figure 6: Three methods for interpolating a transformation from the far left image to the rotated image on the far right. This shows how even a simple combination of rotation and translation must be handled correctly to produce natural looking motion.

vector space and then using the 1D optimisation on each parameter individually. After applying the 1D optimisation the parameters are recombined to produce a set of models that get passed on to the next stage.

Different transformation types were experimented with and it was found that in most videos an affine transformation was the best. It is not as expressive as a full homography but tended to be more robust. In cases where a less expressive transformation was needed, such as a rigid transformation or even no transformation at all, the affine transformation handled it well, whereas the flexibility of the homography seemed to allow it to overfit the data. This introduced warping unnecessarily. In cases that did require a homography, the combination of an affine transformation and the pixel looping seemed able to approximate this sufficiently.

6 PIXEL LEVEL ADJUSTMENT

This section covers per-pixel stabilisation of the algorithm. The input to this stage is a video that has been globally stabilised, meaning that any disparity at the seam should be resolvable using only local changes.

6.1 Efficient computation

By tracking points through time, the video volume can be split into a number of one dimensional sub problems. Solving all of these sub-problems makes each individual track loop, but they will not necessarily remain correlated. Using a smoothing operator in 2D space should make the displacement at each pixel more similar to that of its neighbours. Therefore if the sub-problems have already been solved and we have a discontinuous displacement field, we can use a smoothing operator to average out these proposed displacements. Nagel et al. [14] explore the idea of using 'oriented smoothness' for optical flow field computation. This can be regarded as anisotropic diffusion: smoothing in the direction of edges but not across them. Given that the optical flow field should be piecewise smooth, we can expect our displacement vector fields to have this property also. Hence, it makes sense to use an edge preserving filter during this stage of the pipeline. This approach was also used by Lang et al. [10] when producing temporally consistent results for various applications.

Rather than solving all of the 1D problems and then doing the smoothing, these two steps are instead combined into a single update *per frame*. Doing this, it is no longer necessary to find all the tracks in advance. The looping solution for a point is already known, given its computed Laplacian, estimated Laplacian, and *point in time*. By assuming that each pixel is part of a track that includes its neighbours forward and backward in time, we effectively smooth along this virtual track. To do this, the values of f_1' and f_2' are propagated through the volume. The actual displacement is then recovered by $\mathbf{h} = f_1'\mathbf{h}_1 + f_2'\mathbf{h}_2$, as described in Section 4.2. This is summarised in Algorithm 1. This algorithm is run in both directions away from the seam, and the two cases can be computed independently from each other. This approach makes the assumption that for every pixel, the optical flow connects it to an equivalent pixel before and after it in time. The rest of this section describes how this is not always possible - even with a perfect flow field - and how it can be addressed.

Algorithm 1: Per-pixel stabilisation

```
Input  : Optical flow field $F$, 1D basis solutions $h_1$, $h_2$
Output : Displacement vector field $D$
Estimate Laplacians for first and last frames $f_1$, $f_2$;
Apply edge preserving filter on $f_1$ and $f_2$;
$D(x, y, 1) := f_1(x, y, 1)h_1 + f_2(x, y, 1)h_2$;
for frame $t = 2...T$ do
    $f_1(x, y, t) := f_1(x - F_x(x, y, t), y - F_y(x, y, t), t - 1)$;
    $f_2(x, y, t) := f_2(x - F_x(x, y, t), y - F_y(x, y, t), t - 1)$;
    Apply edge preserving filter for all $f_1$ and $f_2$ at time $t$;
    $D(x, y, t) := f_1(x, y, t)h_1 + f_2(x, y, t)h_2$;
end
```

6.2 Incorrect correspondences

There are a few ways in which a displacement in the optical flow field may not be valid. A correspondence could be technically correct if it points to a pixel that is out of the frame of one of the images, however the value that it points to is undefined. Likewise if the location it points to is inside the frame - but has become occluded - then it doesn't matter if the flow is correct or not. If it is correct, meaning that it represents the true motion, the location it points to will have the value of the occluding object, which is not what we expect. If it is not correct then it could have the value of some other object. This is also the case for a flow vector that is completely

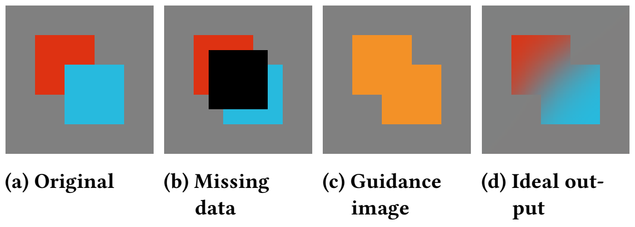


Figure 7: A synthetic example showing guided image filtering on an image with undefined data. This shows how information of boundaries in the colour channels could be transferred to the flow channels by using it as a guidance image

incorrect, whether occluded or not. In these situations we choose not to propagate any information and instead treat those regions as being undefined. We expect to be able to make a reasonable approximation to what the values should be in these regions based on their local neighbourhood.

6.3 Handling missing data

The guided filter as described by He et al.[6] use a guidance image to determine which parts of an image should be smoothed. This is useful for our case since for regions of undefined flow, there is still information in the colour image and we should expect that those edges can still be preserved during filtering. Figure 7 shows a synthetic example. We would expect anything inside the green area of the guidance image to be smoothed, but with the outline remaining sharp. Given that the overlap between the two boxes is hidden in 7b we cannot expect to be able to recover the seam exactly, but the outline should remain the same. The guided filter has no built in method for handling missing data but since it is computed using a series of mean filters then as long as we provide a mechanism for mean filtering with missing data we should be able to leverage the guided filter.

In practice we follow the standard implementation but compute the local linear transforms using a mean filter that only has contributions from valid pixels; the weights of these pixels are also appropriately normalised. The coefficients of these linear transforms are then determined for all pixels using the same mean filter setup. Since the coefficients are now given for all pixels, the final image can be produced as for a normal guided image filter.

7 RENDERING

Once we have a transformation for each pixel, we still need to warp them to get an output frame. The difficulty here is that although the translation is defined for every pixel in the original frames, there will not necessarily be a value defined for every pixel in the new frames. This is because pixels may overlap or be mapped to locations outside of the image window.

Martin *et al.*[13] give an overview of ways to interpolate between images given a flow field. The assumption in that case is that there is a bidirectional flow field and there are two images to interpolate between; a typical case of this is when using stereoscopic images. For our case we only have a single image and set of displacements. In the stereo case, the basis of interpolation is that any point in the

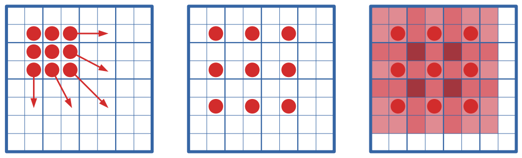


Figure 8: The effect of forward warping pixels with a change in scale. Although the pixels are initially dense, gaps are produced when they spread out (centre). These can be covered by projecting not just a single pixel but also a patch from around each pixel (right)

interpolated image is also defined in at least one of the two stereo images. This implies that missing information from one one can be accounted for by using the other image. With only one frame to use as reference, there will be missing data in the output image that needs to be handled explicitly.

The rendering stage of the pipeline consists of two steps: a forward warping step that uses the original colour data and the new displacements to produce a warped output image, and an inpainting step that fills in the missing data that results from the warping.

7.1 Forward warping

By projecting each pixel according to its displacement, a new image is constructed. This is done such that for a pixel in the original image (i, j) then $(i + \delta_i, j + \delta_j)$ should have the same colour.

For any set of pixels that moves significantly, especially with changes in scale, it is possible for pixels to spread out too much in the output. This ends up with gaps between the pixels as shown in Figure 8. This can be straightforwardly fixed by also considering a small patch around each pixel. As shown in the figure, it only takes a small patch size for all of the missing regions to become filled.

For any regions that have overlapping patches, the colour contribution of each patch is averaged out. This can have the effect of blurring the frame slightly. One way to circumvent this is to use a detail preserving method such as taking the median of the overlapping data or using only the data from the best patch. This is effective, but it slows down the warping significantly. Since the output of this step is only an intermediate step before the inpainting, using the averaged values was found to be sufficient.

7.2 Inpainting missing data

The global stabilisation will necessarily leave areas of missing data at the edges of the frame. Three methods were compared to see which worked best to fix this:

- Cropping: This naturally removes any areas of missing data but also reduces the resolution of the video and can crop out parts of the image that may have been interesting.
- Full frame warping. This assumes that by warping another frame such that is is aligned with the frame that has data missing, we can then use the pixels from the warped frame in those locations where values are not defined. This allows the full image frame to be kept, but only works under certain conditions.

- Inpainting. Inpainting for images is a long studied problem with a large number of different approaches. For videos it is a computationally complex problem although in our case we have extra information that can be taken utilised.

The actual approach used is somewhere between the full frame warping and generic inpainting. As a byproduct of the forward warping, it is possible to construct an inverse nearest neighbour field that matches pixels in the output frame *back* to pixels in the original frames. For areas of missing data these regions will be undefined. By inpainting the *nearest neighbour field* instead of the actual colour values, these regions can be rendered using inverse warped frames such that they are consistent and preserve detail.

After this step on one frame, there will be no more missing data. This inpainted frame then becomes the basis for the subsequent frame. This allows information to propagate from across the seam.

Using these new nearest neighbours, an inverse warping is applied to produce the output frames. The output volume is then run through the process from Section 6, but with colour instead of motion. A simple linear blending is used during this step instead of the 1D solution described previously. This is because it spreads out the error more evenly, and the lack of C_1 continuity is not as obvious in the domain of colour compared to motion.

8 IMPLEMENTATION

There are a number of places in the design in the system where parameters must be specified. We have found that it is possible to produce good results using a range of parameters on a single example, and likewise to get good results out of using the same parameters on a range of examples. Many of these parameters could be adapted such that they are based on the data. There are a couple of cases, however, where the parameters are left to the discretion of the user since they affect the subjective appearance as well as trading off time for improved results.

Firstly, the user can adjust the length of the stabilisation segment. The global stabilisation can be performed on the sequence as a whole, and the regularisation term means that even on long sequences it doesn't have much effect once you get far from the seam. However the user may still choose to use shorter segments either to completely restrict the displacement to 0 outside these segments, or to save on processing; it should be reiterated that the stabilisation step itself takes very little overhead to process further frames, but any stabilised frame is likely to need inpainting as well.

Secondly, they can adjust the length of the local stabilisation segment. This is the same idea, but with the added motivation that this is where the majority of the processing occurs. As with the global stabilisation this can also be adjusted based on how the user wants the output to look.

For all results shown in this paper, the optical flow was computed using DeepFlow [19]. We found that we could get similar results with improved performance by using a faster, but less accurate flow method for the frames not directly at the seam. This was actually done by still using DeepFlow but with a reduced search space on the size of the displacements, and fewer iterations during optimisation. This resulted in reducing the optical flow time for these frames by around 70% on average, and produced comparable results.

We also implemented the Fast Guided Filter [5] as an option to make the filtering step quicker. Results proved comparable to the original filter with downsampling scales of up to 4 times. Beyond 4, they begin to noticeably degrade, but still give a good option to trade off accuracy for speed.

9 RESULTS

The nature of the problem we are addressing means there will be some subjectivity to the results, and previous work on video looping has generally been evaluated on a qualitative basis [2, 8, 17]. We present the outputs of our system but also apply some quantitative measures. The results on a range of videos can be found in the **supplementary materials**. These results include both successes and failures, and illustrate the class of videos for which our method can improve the jump at the seam, as well as where there is still room for improvement. Synthetic data was created to test the system under controlled conditions. Since modules such as optical flow are not guaranteed, it was important to see how the system performed when these were no longer a factor. A set of synthetic cases was produced with the aim of targeting different components of the system, as well as seeing how well those components perform when integrated.

As with many looping videos found online, our real world data was sourced from videos that had not been shot with the intention of being looped afterward. We also compare against a dataset collated by Sevilla-Lara *et al.*[17].

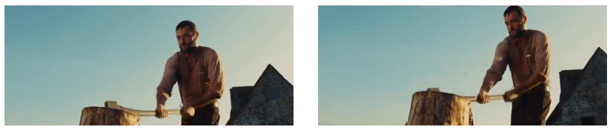
Figure 9 shows the effect of each of the primary features of the system. Some of the effects are difficult to see in a still frame. The per-pixel stabilisation in this case is difficult to see, but it is subtly changing the shape of the man in the sequence. Since the main objects in the sequence are at quite different depths then the global stabilisation can only succeed in making a coarse approximation. For this reason it introduces some skewing in Figure 9b that the per-pixel then corrects since it is trying to stay close to the original shapes; it is not sensitive to global translation however so this correction has no effect on that aspect of the stabilisation. The effect of the colour consistency is quite subtle in this sequence, but it can be seen that the house in the background is changed to match both the frames either side of the seam. Likewise there are time dependent effects such as the man's buckle reflecting light and pieces of wood flying through the air. These were in only one of the original frames but now have continuity across the seam.

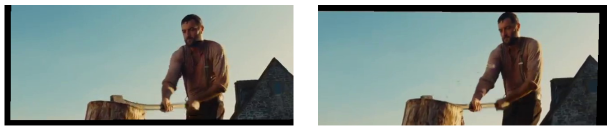
10 EVALUATION

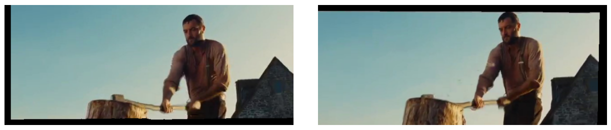
We show in this section quantitative methods that evaluate our system against two criteria of the problem we are addressing: the objective function - that aims to balance smoothness at the seam with adherence to the original video - and the principle that trying to loop a video that does *not* contain a visible seam should have no effect.

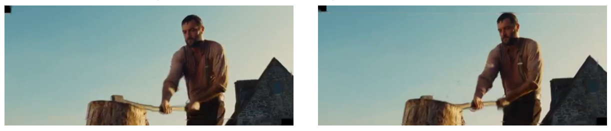
10.1 Objective function

In one case of quantitative evaluation for video looping, Liao et al.[11] compare using an objective function that is similar but separate from the one they optimised on; this is due to difficulties with implementing such an optimisation problem. In the same vein, we


(a) The original images before (left) and after (right) the seam.


(b) After running the system with global stabilisation only.


(c) After running the system with global and local stabilisation.


(d) After running the system with stabilisation and inpainting.

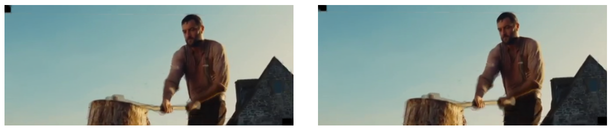

(e) The result after running the images through the complete pipeline. At this point they should now appear to be neighbouring frames.

Figure 9: The effect of the different stages of the system.

Sequence	$E_{original}$	E_{global}	E_{local}	E_{both}
chopping wood	109.0069	10.8360	8.1377	9.0704
cats	3.8650	0.8965	0.7673	0.2795
parrot	732.7081	33.7682	87.0311	39.6415

Table 1: Error according to the objective function of the original sequence, compared to using: only the global stabilisation, only the local stabilisation, and both together. The qualitatively better result from using both types can sometimes have a higher error, and this highlights the difficulty of evaluating solutions to this problem.

compare against the global objective function proposed in equation 1 to verify that using our approximated method does indeed reduce the stated objective.

Although we cannot be sure whether the objective has actually been *minimised*, the results shown in Table 1 indicate that the optimisation is working. In most cases, this also verifies that using either the global stabilisation or per-pixel stabilisation makes an improvement, and the combination is better still.

There are two exceptions that come up. For the synthetic *parrot* sequence it was generated from a single transformation model and so can be captured by the global stabilisation step alone. In this case, since the Laplacian estimate is only an approximation, the per-pixel stabilisation is actually working against the overall objective and increases the error slightly. This is still a huge improvement on the original sequence, however, and the difference is relatively small.

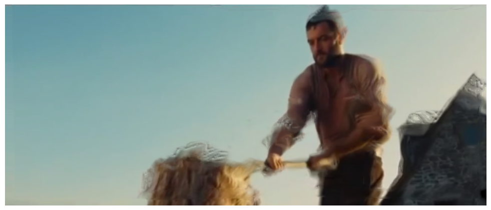


Figure 10: Distortion produced by only using local, per-pixel stabilisation with no global stabilisation beforehand.

For the *chopping wood* sequence, from the objective the per-pixel stabilisation appears to be sufficient and in fact better than using the global stabilisation as well. In actual fact not using the global stabilisation here results in a distorted image that has reduced the error but no longer matches the structure you would expect - this can be seen in Figure 10. This could potentially be avoided by using a far stronger smoothing, however the more generalisable global stabilisation works well despite the sequence displaying a range of depth with few good features to track. This indicates that the global objective function should be improved to be able to reflect these kind of cases.

10.2 Repeated application

Since the objective of the system is to remove the disparity between a pair of given frames, we should expect that if there does not exist any disparity between those frames then the result should be that there is no difference; the output should be identical to the input. The only inaccuracy in this case should come from errors in the optical flow or the estimated Laplacian at the seam. We can therefore use this to evaluate whether our system could be introducing errors by running it on contiguous videos that do not have a visible seam. We quantify this by looking at the magnitude of the displacement vectors that the looping procedure generates.

Given that our objective is to remove the disparity at the seam, we should also expect that running the full system *again* on a video that has already been processed will produce no change. Table 2 shows the mean displacement after running our system for three cases: after a single run through the system, running the output through the system again, running the system on the original sequence where the 'seam' is a point in the middle of the sequence.

The results show that the repeated application produces displacements an order of magnitude smaller than the first time, and for two out of three sequences these were less than a pixel on average. We can consider the sequences with no seam to be a lower bound, and although repeating the process doesn't lower the displacements quite to that level, they are still relatively close compared to the original disparity.

10.3 Comparison against an existing method

The closest existing method to ours is that described in [17]. Their method has the key difference that they are also optimising for

Sequence	Mean displacement (pixels)		
	Looped	Repeated loop	No seam
chopping wood	2.2545	0.2214	0.0287
cats	1.5478	0.2052	0.1404
parrot	22.4548	4.2282	0.7301

Table 2: Applying our system produces displacements in the end frames in order to correct the discontinuity. The average magnitude of this is shown in the left column. Applying our system again on this looped footage (centre column) shows that the displacements are now very small since there is no longer a discontinuity to correct. As a baseline we compare this to running the system on a contiguous segment from the same sequence which should naturally have no discontinuity (right column).

the part of the video that they use to create the loop from. For this reason we compare against three videos from their dataset using clips from the same segment found by their optimisation. The videos we compared against were *basketball*, *ski*, and *tango*.

In general it was found that their method produced loops with fewer artefacts that appear more consistent when looking frame by frame. However, the overall motion when played back had a much more obvious transition point between the ends and didn't appear as smooth as our method. Figure 11 shows an example of typical artefacts produced by our system, which are not present in the equivalent frame. The motion differences can be most clearly seen in the videos in the supplementary material.

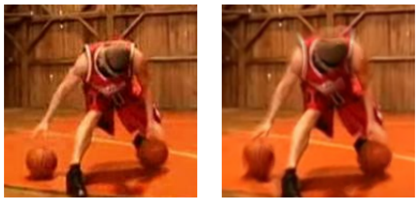


Figure 11: Equivalent frames from the *basketball* sequence produced by [17] (left) and ours (right). Despite introducing an artefact on this frame, when viewed as a video our method can be seen to produce smoother motion across the seam, whereas the result from [17] appears to be only C_0 continuous.

10.4 Failure cases

A few common failure modes were observed when processing clips using the proposed system. For any clips where the optical flow does not capture the scene well, the looping procedure is also likely to fail, such as in Figure 12.

Even when the system has generally performed well there can still be artefacts introduced. Figures 13a and 13b illustrate the case where an incorrect displacement is used during the colour correction. This disparity has not been registered as not having an incorrect alignment and so the system tries to blend it out and propagate this through time. The result is that colours can seem to

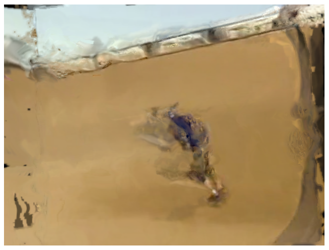
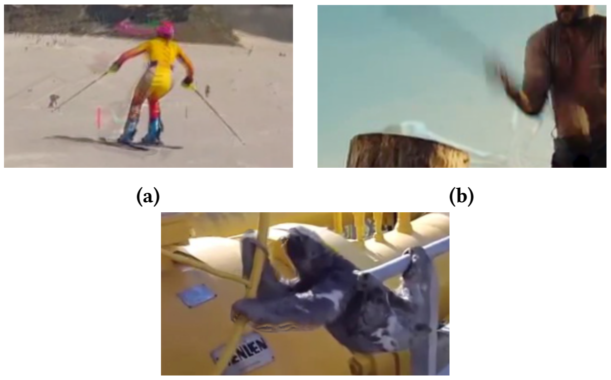


Figure 12: An output frame from the 'skateboarder' sequence where the system creates artefacts due to unreliable optical flow data between the seam frames.



(a) (b) (c)

Figure 13: Some examples of where the system produces rendering artefacts, even when smoothing the motion correctly.

smear across the video. This is still less jarring at the point where the seam was, but the effect lasts longer since it covers more frames.

In Figure 13c, even though the new displacements are good enough to produce a seamless output, there are artefacts due to disocclusions. An occlusion detection process such as that made by Humayun *et al.*[7] could help to eliminate this problem. Once these regions have been identified, then inpainting should help to fill in the missing data, although this process may require some tuning.

11 CONCLUSION

It has been shown that looping real world videos is a difficult problem and that even with a restricted class of videos there are still difficulties. We have proposed an approach that in many cases produces results that are either seamless or significantly improved. It was found that this problem is computationally very heavy and so a number of methods were proposed to improve the performance such that results good be produced in a reasonable time frame.

12 FUTURE WORK

For sequence data, Recurrent Neural Networks (RNNs) have become the de facto standard in recent years. An RNN processes its inputs sequentially, meaning that for data x_i it is making use of $x_{i-1}, x_{i-2}, ...x_0$. The setup of our system is conceptually similar, where we propagate away from the seam ($\mathbf{x}_0$) to find the appropriate displacements for each subsequent frame $\mathbf{x}_i$. Replacing this stage of our system with an RNN could give additional power to make use of more complex long term dependencies that do not depend solely on the behaviour at the seam.

As seen in Section 2.1 there have been a number of approaches that focus more on the problem of detecting looping sequences. In many cases the underlying assumption is that if an appropriate segment is found then little or no extra processing is needed to make this appear consistent. For sequences that do not fit this assumption, we can see our system as being a component that can be added at that stage to produce a better synthesis. This should improve current results as well as expanding the class of possible loops that can be captured by those methods.

ACKNOWLEDGEMENTS

We are supported, in part, by NERC NE/P019013/1.

REFERENCES

[1] Jiamin Bai, Aseem Agarwala, Maneesh Agrawala, and Ravi Ramamoorthi. 2012. Selectively de-animating video. *ACM Transactions on Graphics* 31, 4 (2012), 1–10. https://doi.org/10.1145/2185520.2185562

[2] Jiamin Bai, Aseem Agarwala, Maneesh Agrawala, and Ravi Ramamoorthi. 2013. Automatic cinemagraph portraits. *Computer Graphics Forum* 32, 4 (2013), 17–25. https://doi.org/10.1111/cgf.12147

[3] Pravin Bhat, C. Lawrence Zitnick, Noah Snavely, Aseem Agarwala, Maneesh Agrawala, Michael Cohen, Brian Curless, and Sing Bing Kang. 2007. Using photographs to enhance videos of a static scene. *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (2007), 327—-338. https://doi.org/10.2312/EGWR/EGSR07/327-338

[4] Matthias Grundmann and Irfan Essa. [n. d.]. Auto-Directed Video Stabilization with Robust L1 Optimal Camera Paths. 1 ([n. d.]).

[5] Kaiming He and Jian Sun. 2015. Fast Guided Filter. *CoRR* abs/1505.0 (2015), 2. arXiv:1505.00996 http://arxiv.org/abs/1505.00996

[6] Kaiming He, Jian Sun, and Xiaoou Tang. 2010. Guided Image Filtering (ECCV). *European Confenrence on Computer Vision* 35, 7 (2010), 1–14. https://doi.org/10.1007/978-3-642-15549-9_1

[7] Ahmad Humayun and Gabriel J Brostow. 2011. Learning to Find Occlusion Regions. *CVPR 2011* (2011), 2161–2168.

[8] Neel Joshi, Sisil Mehta, Steven Drucker, Eric Stollnitz, Hugues Hoppe, Matt Uyttendaele, and Michael Cohen. 2012. Cliplets: juxtaposing still and dynamic imagery. *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12* (2012), 251–260. https://doi.org/10.1145/2380116.2380149

[9] Vivek Kwatra, Arno Schodl, Irfan Essa, Greg Turk, and Aaron Bobick. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003* 22, 3 (2003), 277–286.

[10] Manuel Lang, Oliver Wang, Tunc Aydin, Aljoscha Smolic, and Markus Gross. 2012. Practical Temporal Consistency for Image-based Graphics Applications. *ACM Trans. Graph.* 31, 4, Article 34 (July 2012), 8 pages. https://doi.org/10.1145/2185520.2185530

[11] Jing Liao, Mark Finch, and Hugues Hoppe. 2015. Fast computation of seamless video loops. *ACM Transactions on Graphics* 34, 6 (2015), 1–10. https://doi.org/10.1145/2816795.2818061

[12] Zicheng Liao, Neel Joshi, and Hugues Hoppe. 2013. Automated video looping with progressive dynamism. *ACM Transactions on Graphics* 32, 4 (2013), 1. https://doi.org/10.1145/2461912.2461950

[13] N Martin and S Roy. 2008. Fast view interpolation from stereo: Simpler can be better. *Proceedings of 3DPVT'2008-the Fourth ...* (2008). http://www.cc.gatech.edu/conferences/3DPVT08/Program/Papers/paper213.pdf

[14] H H Nagel and W Enkelmann. 1986. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE transactions on pattern analysis and machine intelligence* 8, 5 (1986), 565–593. https://doi.org/10.1109/TPAMI.1986.4767833

[15] Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. *ACM Transactions on Graphics* 22, 3 (2003), 313. https://doi.org/10.1145/882262.882269

[16] Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa. 2000. Video textures. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques SIGGRAPH 00* 7, 5 (2000), 489–498. https://doi.org/10.1145/344779.345012

[17] L. Sevilla-Lara, J. Wulff, K. Sunkavalli, and E. Shechtman. 2015. Smooth Loops from Unconstrained Video. *Computer Graphics Forum* 34, 4 (2015), 99–107. https://doi.org/10.1111/cgf.12682

[18] James Tompkin, Fabrizio Pece, Kartic Subr, and Jan Kautz. 2011. Towards Moment Images: Automatic Cinemagraphs. In *Visual Media Production (CVMP), 2011 Conference for.* 87–93. https://doi.org/10.1109/CVMP.2011.16

[19] Philippe Weinzaepfel, Zaid Harchaoui, Cordelia Schmid, Philippe Weinzaepfel, Zaid Harchaoui, Cordelia Schmid, Zaid Harchaoui, and Cordelia Schmid. 2013. DeepFlow : Large displacement optical flow with deep matching. (2013).