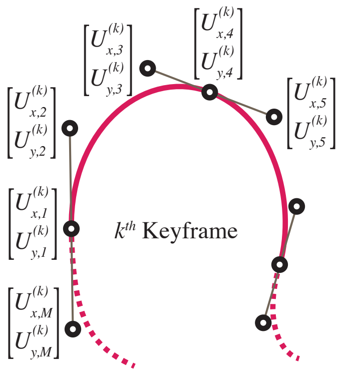
Roto++: Accelerating Professional Rotoscoping using Shape Manifolds (Supplemental Material)

A Notation for the Technical Approach

Table A-1 contains a summary of the notation used in Section 4 with Figure A-1 illustrating the layout of the control points for an input keyframe.

N	Number of frames in the shot
$n \in 1..N$	Index of a frame
K	Number of available keyframes
$k \in 1..K$	Index of a keyframe
M	Number of control points in the closed spline
$m \in 1..M$	Index of a control point
L	Number of trackers
$l \in 1..L$	Index of a tracker
$\{U_k\}$	Set of *input* keyframe splines
$\{Y_n\}$	Set of *output* splines
$\theta, \mathbf{t}, s$	The rotation, translation and scale of a spline
Q	A rotation matrix
R	A mean reference spline for alignment
$\mathbf{x}$	A location on the manifold
$V = \mathbf{F}(\mathbf{x})$	A new spline generated from the manifold
$\mathbf{p}^{(l)}_{m,n}$	l^{th} Tracker output point for the m^{th} control point in the n^{th} frame

Table A-1: *Summary of notation used in Section 4.*



$\begin{bmatrix} U^{(k)}_{x,3} \\ U^{(k)}_{y,3} \end{bmatrix}$
$\begin{bmatrix} U^{(k)}_{x,4} \\ U^{(k)}_{y,4} \end{bmatrix}$
$\begin{bmatrix} U^{(k)}_{x,2} \\ U^{(k)}_{y,2} \end{bmatrix}$
$\begin{bmatrix} U^{(k)}_{x,5} \\ U^{(k)}_{y,5} \end{bmatrix}$
$\begin{bmatrix} U^{(k)}_{x,1} \\ U^{(k)}_{y,1} \end{bmatrix}$
k^{th} Keyframe
$\begin{bmatrix} U^{(k)}_{x,M} \\ U^{(k)}_{y,M} \end{bmatrix}$

Figure A-1: *Notation for the control points of the* k^{th} *input keyframe.*

B The GP-LVM

We begin by introducing Gaussian Processes (GPs) and the GP Latent Variable Model (GP-LVM) which may be used to perform unsupervised manifold learning. We will then provide details of how we use this to generate our shape model.

Gaussian Processes Gaussian Processes [Rasmussen and Williams 2006] represent distributions over functions and may be used as a powerful tool to learn a smooth, non-linear mapping from one space $\mathbf{x} \in \mathbb{R}^Q$ to another $\mathbf{y} \in \mathbb{R}^D$. They are particularly effective when mapping between spaces with different dimensions, in our case we have $Q \ll D$, when the vectors of interest in the higher dimensional space actually lie on a manifold of far lower dimension. We instinctively believe this to be the case for roto-curves since the there are a very large number of possible splines that can be drawn with M control points but only very few of them will map to the shape of the object we are segmenting.

The GP achieves this mapping by modelling the covariance of the higher dimensional vectors as a kernel function in the lower dimensional space. If we assume that all our vectors are normalized to have zero mean then the GP models the probability of a high dimensional vector $\mathbf{y}$ as the multivariate Gaussian

$$P(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \kappa(\mathbf{x},\mathbf{x})\right) \ . \tag{B-1}$$

Here, $\kappa(\mathbf{x},\mathbf{x})$ is a kernel function that encodes how distances in the low dimensional correlate to similar high dimensional vectors.

GP-LVM The GP-Latent Variable Model, [Lawrence 2005], takes an unsupervised approach to learn a generative model. Unlike, the standard GP, we only provide a set of high dimensional vectors $\{\mathbf{y}_k\}$ (the training data) and a desired kernel mapping function $\kappa(\cdot,\cdot)$. The GP-LVM then estimates the best corresponding set of low dimensional vectors $\{\mathbf{x}_k\}$ that generate the high dimensional vectors when passed through the GP. Whilst the full derivation is quite involved, the effect is that we end up with a set of low dimensional vectors and a mapping function to generate high dimensional vectors that are representative of the training data. For our purposes we set the high dimensional vectors to be the spline keyframes (in dimension $D = 2M$) and we learn a set of points $\{\mathbf{x}_k\}$ in $\mathbb{R}^Q$ (we use $Q = 2$) and a mapping such that for every point in the 2D space, we generate a different roto spline.

The nature of the mapping between the spaces is determined by the kernel function. We use the Radial Basis Function (RBF) kernel

$$\kappa(\mathbf{x}_i, \mathbf{x}_j \mid \alpha, \beta) = \alpha \exp\left(-\frac{1}{2}\beta\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \ , \tag{B-2}$$

where α and β are the variance and length-scale hyper parameters. The RBF kernel is inherently smooth, therefore, as we move around in the low dimensional manifold space the high dimensional space will also vary smoothly. This provides the desirable property for using the manifold for rotoscoping: neighboring frames should have smoothly varying shape and therefore should have nearby embedded spaces. This is demonstrated by Figure 3 where we embed 19 consecutive roto-curves in a 2D manifold.

Learning the Manifold We learn the manifold model by optimising the hyper parameters (α, β) a noise parameter (σ) and the embedded locations ($X = \{\mathbf{x}_k\}$) of the corresponding training keyframe splines ($\{\mathbf{y}_k\}$). We maximise the likelihood of the training examples factored across each of the dimensions as

$$P(Y|X,\Omega) = \prod_{k}^{K} \mathcal{N}\left(Y_k(:)|\mathbf{0}, \kappa(X, X \mid \alpha, \beta) + \sigma^2 I\right) \ , \tag{B-3}$$

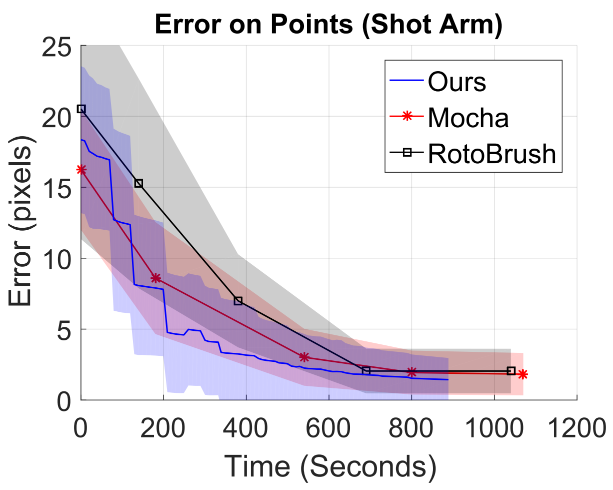


Figure C-2: *Quantitative measure of error against time for shot Arm. We quantitatively compare our method to the commercial Mocha workflow and RotoBrush on our Arm shot (Containing 30 frames, the same shot used in Figure C-3). The shaded regions around each curve represents one standard deviation.*

where Ω denotes the hyper and noise parameters and Y and X are stacked collections of training row vectors such that

$$Y = \begin{bmatrix} Y_1(:) \\ Y_2(:) \\ \vdots \\ Y_K(:) \end{bmatrix} \tag{B-4}$$

and similarly for X. Here we used the notation

$$Y_k(:) = \left[Y_{x,1}^{(k)}\ Y_{y,1}^{(k)}\ Y_{x,2}^{(k)}\ Y_{y,2}^{(k)}\ \ldots\ \right] \in \mathbb{R}^{2M} \tag{B-5}$$

to unwrap Y_k (a $2 \times M$ matrix of spline keypoints) as $1 \times 2M$ row vector. Also, we use the notation $\kappa(X, X \mid \alpha, \beta)$ to denote the covariance matrix

$$\left[\kappa(X, X \mid \alpha, \beta)\right]_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j \mid \alpha, \beta)\ . \tag{B-6}$$

Since the kernel matrix is a non-linear function we use a gradient based non-linear optimiser to maximise Equation B-3 and initialize the low dimensional vectors using linear PCA to reduce the high dimensional vectors to the appropriate number of dimensions (Q). We also place a unit Gaussian prior on the manifold locations to maintain a natural scale.

Generating New Splines Once we have trained the GP-LVM, we can generate new splines from a low dimensional location $\mathbf{x}'$ by conditioning the distribution of Equation B-3 on the training data. In practical terms, this produces a simple matrix multiplication

$$\mathbf{y}' = \mathbf{F}(\mathbf{x}') = \kappa(\mathbf{x}', X|\alpha, \beta)\left[\kappa(X, X|\alpha, \beta)\right]^{-1} Y \tag{B-7}$$

to generate the new normalized spline $\mathbf{y}'$.

C Additional Results

Figure C-3 shows our additional visual comparison to the JumpCut, RotoBrush and Mocha Tracker. The selected clip contains eight adjacent frames (1080p), and represents the difficulties of non-rigid deformation and illumination changes. Within this comparison, we keyframe the first and the last frames across all the trials. For the tests of JumpCut and RotoBrush, we show the curve propagation results in forward and backward separately because the related software does not support the propagation using two keyframes. Our method outperforms the JumpCut and RotoBrush, and yields competitive accuracy when compared to the proprietary Mocha Tracker (4.36 v.s. 6.42 average pixel RMS).

Figure C-2 shows our additional quantitative comparison to the Mocha Tracker and RotoBrush. An experienced artist performs the rotoscoping on a difficult shot (the Arm, 30 frames) by using Mocha Pro 4.1.0 and AfterEffect CC'15 (RotoBrush) respectively. We shows the points error against the time elapsed. The the commercial packages, the artist is required to output the alpha masks at 5 minute intervals after the first two keyframes. Note that this comparison does not include JumpCut because the GUI is not publicly available.

References

Lawrence, N. 2005. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research 6*, 1783–1816.

Rasmussen, C. E., and Williams, C. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

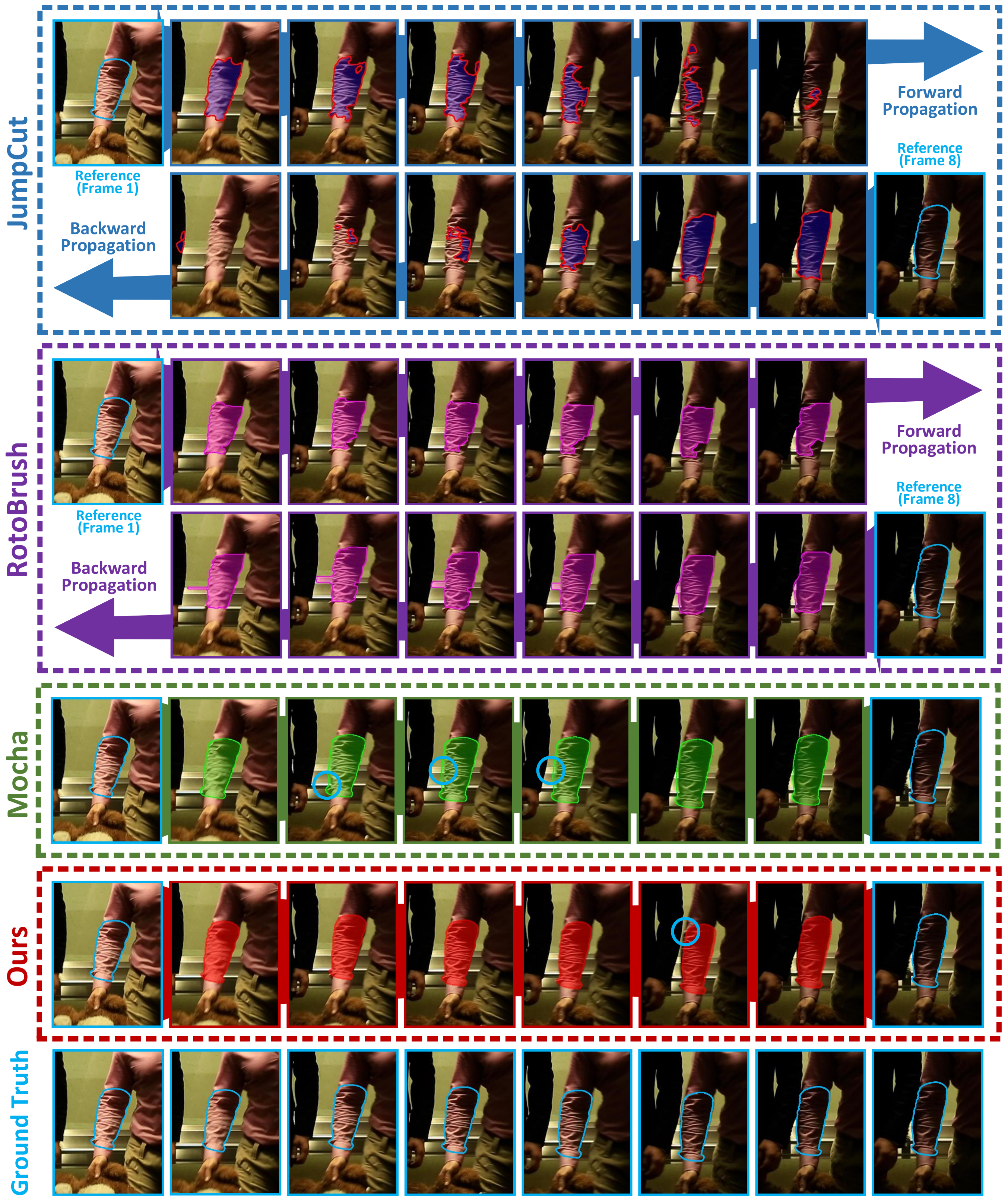


Figure C-3: *Qualitative comparisons of different rotoscoping methods. We propagate a curve from a reference frame to the other frames of the arm shot. From* ***Top to Bottom****, The* ***First*** *group shows the JumpCut results by propagating the reference curve in forward (first row) and backward (second row) directions respectively. The* ***Second*** *group illustrates the results of RotoBrush (Adobe AfterEffect CC 2015). The* ***Third*** *group shows the results by using the Mocha Tracker (ver. 4.1.0) which takes into account two reference frames. The* ***Fourth*** *group gives the results of our method which uses the manifold (trained by two reference keyframes) and the blender planar tracker. The* ***Bottom*** *row shows the ground truth for each frame. Note that the cyan circles highlight details for the results.*