Nonlinear Shape Manifolds as Shape Priors in Level Set Segmentation and Tracking

Victor Adrian Prisacariu
University of Oxford
victor@robots.ox.ac.uk

Ian Reid
University of Oxford
ian@robots.ox.ac.uk

Abstract

We propose a novel nonlinear, probabilistic and variational method for adding shape information to level set-based segmentation and tracking. Unlike previous work, we represent shapes with elliptic Fourier descriptors and learn their lower dimensional latent space using Gaussian Process Latent Variable Models. Segmentation is done by a nonlinear minimisation of an image-driven energy function in the learned latent space. We combine it with a 2D pose recovery stage, yielding a single, one shot, optimisation of both shape and pose. We demonstrate the performance of our method, both qualitatively and quantitatively, with multiple images, video sequences and latent spaces, capturing both shape kinematics and object class variance.

1. Introduction

Level sets [11] have become very popular in segmentation over recent years, as they provide an effective framework for modelling the shape and evolution of curves. Rather than formulating an explicit representation of a curve, a level set-based method represents the curve as the zero level of a higher dimensional, Lipshitz continuous, embedding function. This representation allows level set-based methods to handle curve topological changes (breaking up, merging, disappearing, etc.) very easily.

Most level set-based methods consist of an unconstrained minimisation of an image-driven, region-based, energy function. Some examples are [1] or [19]. Unfortunately, unconstrained minimisation often fails, because of occlusions, noise, motion blur or simply too similar foreground and background colour statistics. One solution is to supplement the low level information with prior shape knowledge i.e. constrain the minimisation of the energy function to a known shape space. Two schools of thought exist over how the shape information can be generated and used: one proposes to generate the 2D shapes as projections of known 3D models, while the other attempts to learn a (lower dimensional) representation of the shape space.

The first approach was initially explored, in level set-based segmentation, in [14]. Here the authors alternately iterate between a minimisation of a Chan-Vese like energy function [19], with an added shape term, and a 6 DoF 3D pose optimisation. The method was refined in [17] where the shape minimisation is replaced by an approximate evolution from the 6 DoF 3D pose parameters. Finally, a fully variational approach was proposed in [12], where the level set energy function [1] is differentiated with respect to the 6 DoF 3D pose parameters, simultaneously yielding both segmentation and pose. There are several problems with these methods. Often 3D models are not available and, e.g. in the case of articulated or deformable objects, the shape of the object is not defined only by its 6 DoF 3D pose. Accommodating such objects means increasing the search space, which quickly becomes intractable.

The second approach was first explored, for level set-based segmentation, in [10], where principal component analysis (PCA) is used on the set of embedding functions. The evolution of the contour is controlled by adding a shape term to the energy function. The technique is refined in [18] where, again, PCA is used on the set of embedding functions, but the level set energy function is minimised directly in the shape space i.e. with respect to the position in the lower dimensional latent space. In [2] nonlinearity is introduced to model shape dynamics, but PCA is again used. PCA is a linear technique, meaning that all PCA based methods assume the shape space to be linear, which is a rather limiting assumption, as shape spaces are often nonlinear. Nonlinear shape spaces were used in papers like [3] or [16]. The probability distributions on the space of embedding functions is modelled with Gaussians in [16] and with a kernel density estimator in [3]. Similar to the linearity

assumption of PCA, the Gaussianity assumption of [16] is also rather limiting. While no such limitation affects [3] , it does not use dimensionality reduction, making inference difficult. Nonlinear (but not probabilistic) dimensionality reduction is used in [4], in the form of Kernel PCA. However, since KPCA learns a mapping from the higher dimensional space to the lower dimensional one, [4] optimises an energy function in the space of embedding functions, aiming to minimise the distance between the projection of those embedding functions and the known lower dimensional points. This means that the evolution of the contour is not constrained to high probability shapes. The generative process itself is a nonlinear optimisation, making it slow and susceptible to local minima. Finally, [4] needs the parameter of the embedding kernel to be set manually, which can lead to overfitting: a small value for this parameter produces high accuracy but at the cost of a reduced basin of convergence, while a large value leads to the opposite.

Our method is similar in spirit to [18] or [4], in that we find a lower dimensional representation of the shape space. But unlike previous work we use a nonlinear dimensionality reduction technique called *Gaussian Process Latent Variable Models* (GP-LVM) [8]. Also, rather than learning the space of embedding functions, we learn the space of embedded contours, by representing the contours with *elliptic Fourier descriptors* [7]. This allows us to cast the segmentation problem as a direct minimisation of an image-driven, level set-based, energy function, with respect to the position in the latent space. Consequently, our method has the following advantages over previous work: (i) we use GP-LVM, which means we place no assumptions on the distribution of the latent space, while allowing us to use fewer dimensions and capture more of the variance and still explicitly constrain the evolution of the contour to high probability shapes; (ii) each point in the latent space is modelled by a Gaussian, whose mean and variance are found in a principled manner, unlike PCA where the variance is accessed in a simplistic way (squared distance from observed data) (iii) it is possible in principle to learn mappings from the shape space to other spaces i.e. a pose space [5].

The remainder of the paper is structured as follows: we briefly describe elliptic Fourier descriptors, Gaussian Processes and GP-LVM in Sections 2 and 3. We detail the latent space learning phase and latent space-based segmentation in Sections 4 and 5. Section 6 shows how we combine segmentation with 2D pose recovery. We show results obtained by applying our method to several images and video sequences in Section 7 and conclude in Section 8.

2. Elliptic Fourier Descriptors

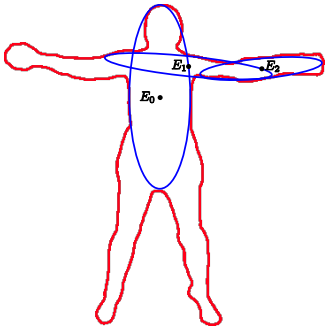

Figure 1. Example 3 harmonics of an elliptic Fourier descriptor.

Elliptic Fourier descriptors [7] model shapes as sums of elliptic harmonics. Each harmonic is described by four coefficients, interpreted geometrically as major axis length, minor axis length, orientation of the major axis and angle of phase of the ellipse.

As shown in [7], any contour can be expressed as two periodic functions $x(t)$ and $y(t)$. For example:

$$x(t) = a_0 + \sum_{k=1}^{\infty} \left(a_k \cos \frac{2k\pi t}{T} + b_k \sin \frac{2k\pi t}{T} \right) \tag{1}$$

where T is the perimeter of the contour, $t = 2\pi l/T$, l is the arc length from a preset starting point, and:

$$a_0 = \frac{1}{T}\sum_{p=1}^{K}\frac{\Delta x_p}{2\Delta t_p}(t_p^2 - t_{p-1}^2) + \zeta_p(t_p - t_{p-1}) \tag{2}$$

$$a_k = \frac{T}{2k^2\pi^2}\sum_{p=1}^{K}\frac{\Delta x_p}{\Delta t_p}\left(\cos\frac{2k\pi t_p}{T} - \cos\frac{2k\pi t_{p-1}}{T}\right) \tag{3}$$

$$b_k = \frac{T}{2k^2\pi^2}\sum_{p=1}^{K}\frac{\Delta x_p}{\Delta t_p}\left(\sin\frac{2k\pi t_p}{T} - \sin\frac{2k\pi t_{p-1}}{T}\right) \tag{4}$$

with K being the number of points in the contour, $\Delta x_p = x_k - x_{k-1}$, $\Delta t_p = \sqrt{(\Delta x_p)^2 + (\Delta y_p)^2}$, and:

$$\zeta_p = \sum_{j=1}^{p-1}\Delta x_j - \frac{\Delta x_p}{\Delta t_p}\sum_{j=1}^{p-1}\Delta t_j \tag{5}$$

$y(t)$ is defined completely analogously in terms of the coefficients c_0, c_k and d_k.

3. Gaussian Processes and GP-LVM

Gaussian Processes (GPs) [13] are distributions over functions. Given some training data, a GP can produce a function approximating the data, together with point-level covariance estimates. A GP is defined by a *mean function* $\mu(\cdot)$ and a *covariance function* $\kappa(\cdot,\cdot)$. If a function is drawn from this GP, then for any number of values X the corresponding function values Y are normally distributed, with:

$$P(Y|X) = \mathcal{N}(Y|\mu(X), \kappa(X,X)) \tag{6}$$

The mean function is usually defined to be zero while the covariance function must only be positive semi-definite. A standard covariance function is the *radial basis function*. Covariance functions are parametrised by *hyperparameters*. Training a GP consists of maximising a log likelihood, wrt. the hyperparameters [13].

Gaussian Processes Latent Variable Models (GP-LVM) [8] is a dimensionality reduction technique, i.e. it allows us to represent a data set $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$, where each $\mathbf{y}_i$ is a data point of dimensionality d, with a lower dimensional set of latent variables $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, where each $\mathbf{x}$ is a latent point of dimensionality q, with $q < d$.

As shown in [8], we can write:

$$P(\mathbf{Y}|\mathbf{X}, \theta) = \prod_{i=1}^{n}\mathcal{N}\big(\mathbf{y}_i|0, K(\theta)\big) \tag{7}$$

where θ are the GP hyperparameters and $K_{ij} = \kappa(x_i, x_j)$. Learning the GP-LVM consists of maximizing Equation 7 with respect to $\mathbf{X}$ and θ.

In this formulation, GP-LVM does not guarantee that points close in the data space will also be close in the latent space. To preserve local distances we impose *back-constraints* [9] i.e. constrain each latent variable to be a smooth mapping from its high dimensional counterpart, and optimise with respect to the parameters of the back-constraining function, rather than the latent variables.

4. Latent Space Learning

Learning the latent space consists of building the elliptic Fourier descriptors for the set of training contours and finding their low dimensional manifold. We first phase align the contours and subsample them to a fixed number of points, a tunable parameter which we denote by n_p. Let n_a be the number of harmonics, in this work between 15 and 50, depending on the complexity of the shape. We then compute the coefficients using Equations 2-4, yielding the data set $\mathbf{Y}$. Finally, we minimise Equation 7 with respect to the parameters of the back-constraining function and the hyperparameters, yielding the space $\mathbf{X}$. The covariance and back-constrain functions are both radial basis functions.

5. Latent Space-based Segmentation

Let I be an image and C be a closed contour, segmenting the image into two disjoint regions, foreground and background, with known appearance models. C is embedded as the zero level of the level set function Φ. We define a monotonic and continuously differentiable energy function $f(\Phi, I)$ as measuring how well the level set function Φ segments the image I. Examples are the Chan-Vese energy function [19] or the Bibby-Reid energy function [1]. In this work $f(\Phi, I)$ is the Bibby-Reid energy function.

Typically, in level set-based segmentation, the contour evolves in an unconstrained space, i.e. $f(\Phi, I)$ is optimised by differentiating it with respect to time and using a standard nonlinear minimisation technique. Here we want to limit the evolution of the contour to the space of possible contours, given the learned latent space. We do this by differentiating $f(\Phi, I)$ with respect to the latent variables $\mathbf{x}_i, i = [1, \ldots, n_p]$. We first apply the chain rule:

$$\frac{\partial f}{\partial \mathbf{x}_i} = \frac{\partial f}{\partial \Phi} \begin{bmatrix} \frac{\partial \Phi}{\partial x} & \frac{\partial \Phi}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial \mu} \frac{\partial \mu}{\partial \mathbf{x}_i} \\ \frac{\partial y}{\partial \mu} \frac{\partial \mu}{\partial \mathbf{x}_i} \end{bmatrix} \tag{8}$$

where x and y are defined in Equation 1 and μ is the mean of the Gaussian trained through the GP-LVM mapping, for each $\mathbf{x}_i$. This is an approximation of the true evolution of the level set with respect to its embedded contour, which is not continuous and so not computable analytically. When evaluated only in a small band around the contour, this chain rule expansion is a correct approximation for a slow evolution of the embedding function, with the mention that $\frac{\partial x}{\partial \mu}$ and $\frac{\partial y}{\partial \mu}$ are defined only for the points on the actual contour. We approximate the derivatives for all the other points with those from their respective closest contour points. $\frac{\partial \Phi}{\partial x}$ and $\frac{\partial \Phi}{\partial y}$ are computed using finite differences.

μ is an elliptic Fourier descriptor, so:

$$\mu = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ \ldots & \ldots & \ldots & \ldots \\ a_{n_a} & b_{n_a} & c_{n_a} & d_{n_a} \end{bmatrix} \tag{9}$$

with a_0 and c_0 being considered 0, in order to normalise the descriptor with respect to translation.

We expand $\frac{\partial x}{\partial \mu}$ as an array of $n_p \times n_a \times 4$ elements, $\frac{\partial x_t}{\partial a_k}$, $\frac{\partial x_t}{\partial b_k}$, $\frac{\partial x_t}{\partial c_k}$, $\frac{\partial x_t}{\partial d_k}$ with $t = [1, \ldots, n_p]$, $k = [1, \ldots, n_a]$, and:

$$\begin{aligned} \frac{\partial x_t}{\partial a_k} &= \frac{\cos 2k\pi t}{T} & \frac{\partial x_t}{\partial b_k} &= \frac{\sin 2k\pi t}{T} \\ \frac{\partial x_t}{\partial c_k} &= 0 & \frac{\partial x_t}{\partial d_k} &= 0 \end{aligned} \tag{10}$$

Similarly for $\frac{\partial y}{\partial \mu}$.

In GP-LVM each latent space point $\mathbf{x}_i$ projects to a data point via a Gaussian distribution, with a mean μ and a variance υ = σ^2:

$$\mu_i = K_* K^{-1} \mathbf{Y} \tag{11}$$

$$\upsilon_i = \kappa(\mathbf{x}_i, \mathbf{x}_i) - K_* K^{-1} K_*^T \tag{12}$$

where K_* is the covariance between $\mathbf{x}_i$ and all the learned latent variable points. Therefore $\frac{\partial \mu}{\partial \mathbf{x}_i}$ can be written as:

$$\frac{\partial \mu}{\partial \mathbf{x}_i} = \frac{\partial \kappa(\mathbf{x}_i, \mathbf{X})}{\partial \mathbf{x}_i} K^{-1} \mathbf{Y} \tag{13}$$

In our case $\kappa(\cdot, \cdot)$ is the radial basis function, so $\frac{\partial \kappa(\mathbf{x}_i, \mathbf{X})}{\partial \mathbf{x}_i}$ will be an array of differentials of the radial basis function, with respect to $\mathbf{x}_i$, for each prelearned latent space point.

The approach presented above optimises $f(\Phi, I)$ without considering the per latent space point variance, an indicator of how likely a point is to generate a valid shape. Points with low variance are more likely to generate valid shapes than

points with high variance. This variance was considered in [8], where maximum likelihood is used to find the point $\mathbf{x}$ which generates a known high dimensional point $\mathbf{y}$. We adopt a similar approach:

$$\mathbf{x} = argmax_{\mathbf{x}_*} P(\mathbf{y}|\mathbf{x}_*, Y, X, \theta, I) \tag{14}$$

$$\log P(\ldots) = -\frac{1}{2}\log \upsilon - \frac{1}{2}\frac{f(\Phi, I)}{\upsilon} - \frac{1}{2}\log(2\pi) \tag{15}$$

$$\frac{\partial \log P(\ldots)}{\partial \mathbf{x}} = -\frac{1}{2}\frac{1}{\upsilon}\frac{\partial \upsilon}{\partial \mathbf{x}} - \frac{1}{2}\frac{1}{\upsilon^2}\left(\frac{\partial f}{\partial \mathbf{x}}\upsilon - f\frac{\partial \upsilon}{\partial \mathbf{x}}\right) \tag{16}$$

where $\frac{\partial \upsilon}{\partial \mathbf{x}}$ follows from Equation 12 as:

$$\frac{\partial \upsilon}{\partial \mathbf{x}} = \frac{\partial \kappa(\mathbf{x}_i, \mathbf{x}_i)}{\partial \mathbf{x}_i} - 2\frac{\partial \kappa(\mathbf{x}_i, \mathbf{X})}{\partial \mathbf{x}_i}K_*^T K^{-1} \tag{17}$$

$f(\Phi, I)$ is the Bibby-Reid energy function [1], so:

$$f(\Phi, I) = -\sum_{x \in \Omega} \log\Big(H_e(\Phi)P_f + \big(1 - H_e(\Phi)\big)P_b\Big) \tag{18}$$

Here Ω is the image domain, H_e is the smooth Heaviside function, x is the pixel in the image, and:

$$P_f = \frac{P(y_i|M_f)}{\eta_f P(y_i|M_f) + \eta_b P(y_i|M_b)} \tag{19}$$

$$P_b = \frac{P(y_i|M_b)}{\eta_f P(y_i|M_f) + \eta_b P(y_i|M_b)} \tag{20}$$

where η_f and η_b are the number of foreground and background pixels, respectively, y_i the colour of the i-th pixel and $P(y_i|M_f)$ and $P(y_i|M_b)$ the foreground and background models over pixel values y_i. We use RGB images and our models are histograms with 32 bins for each channel.

6. Tracking

Our latent spaces only capture the variance of the space of shapes. Therefore, in order to track moving objects, we must also recover the 2D pose. One principled way of doing this is by computing the derivatives of the Equation 18 with respect to the pose parameters, similar to [1] or [12]. For example, following [12], given 4 pose parameters λ_p (translation on the x and y axis, scale and rotation), we can write:

$$\frac{\partial f}{\partial \lambda_p} = \frac{\partial f}{\partial \Phi}\begin{bmatrix}\frac{\partial \Phi}{\partial x} & \frac{\partial \Phi}{\partial y}\end{bmatrix}\begin{bmatrix}\frac{\partial x}{\partial \lambda_p} \\ \\ \frac{\partial y}{\partial \lambda_p}\end{bmatrix} \tag{21}$$

where $\frac{\partial x}{\partial \lambda_p}$ and $\frac{\partial y}{\partial \lambda_p}$ follow trivially.

The two sets of derivatives (with respect to latent space point and to pose) can be used either as part of a joint optimisation towards the latent space position and the pose, or as part of a coordinate descent i.e. alternately iterating between the two sets of parameters. The most principled method is to do a single, one shot, optimisation, of both pose and contour. In our experience, this leads to slower convergence. Optimising pose to convergence and then contour to convergence is faster, as each separate optimisation has a smaller search space, compared to the combined one. However, finding the correct pose for the initialisation contour does not necessarily make it the best pose for the actual contour, and optimising for the latent space point with the wrong pose will often result in an incorrect contour. In practice we noticed that both methods yield similar results, as shown in Section 7.

7. Results

We tested our algorithm with several latent spaces, images and videos, capturing both movement kinematics and object class variance. Throughout our testing we used two dimensional latent spaces, which captured enough of the variance from the high dimensional space, for our test data. Processing time averaged at 280ms per iteration, using an unoptimised Matlab implementation. The algorithm converged within 50 to 100 iterations.

We begin with examples which focus on finding and using a lower dimensional representation of silhouette kinematics. The first example is a star jump action, for which we show the 2D latent space, together with samples from it and from the training data, in Figure 2. Notice that, even though we only use a 2 dimensional latent space, we can still keep most of the variance in the original data, and generate accurate contours.

A typical minimisation of Equation 16, for the star jump latent space, is depicted in Figure 3. Here we evaluated Equation 16 for each meaningful point in the latent space and plotted the convergence trajectory through it, starting from the point marked with *s* and red and ending with the point marked with *t* and green. Notice that the energy function has a well defined minimum, which we are able to reach even from a distant initialisation.

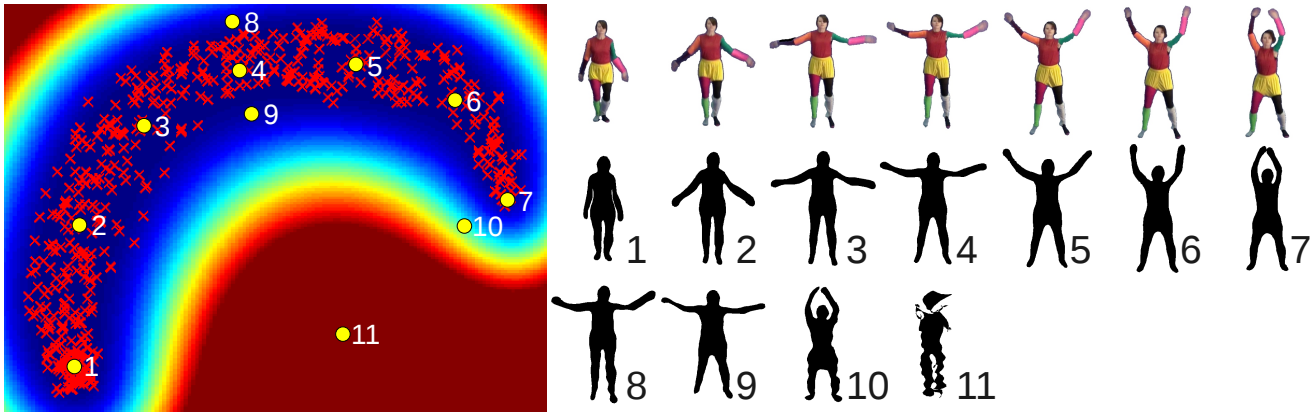

Figure 2. 2D latent space capturing the kinematics of a person performing a star jump (left) together with samples from the training data (right, top row) and the latent space (right, bottom two rows). Warmer colours (e.g. red) indicate higher variance, while colder colours (e.g. blue) indicate lower variance. Samples 1–9 are taken from a low variance region (making them more likely to generate valid shapes) while 10 and 11 are from taken from higher variance regions (making them less likely to generate valid shapes).

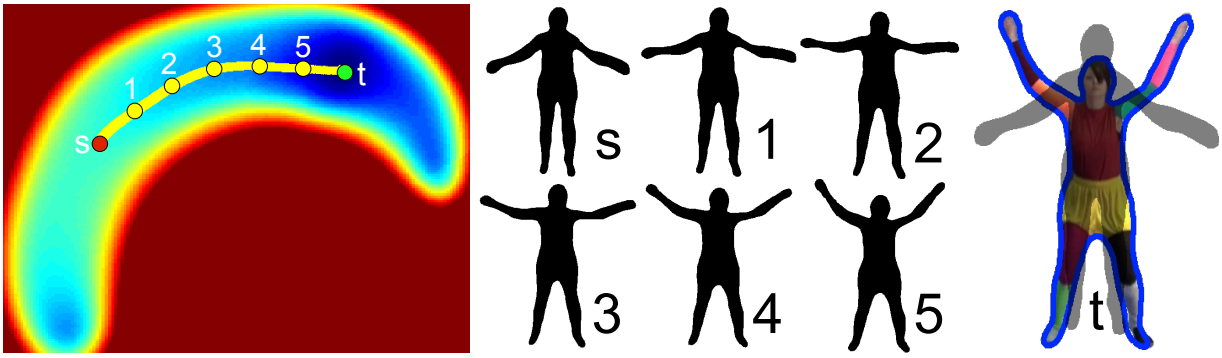

Figure 3. Example minimisation of Equation 16: left – values of the function for meaningful points in the latent space; middle – samples from the convergence trajectory; right – result (surrounded with blue) overlayed with initialisation (transparent black). Minimisation starts at the point marked with *s* and ends at the point marked with *t*.

As shown in Section 6, we can link our latent space-based segmentation with a 2D pose recovery step. Figure 4 shows our algorithm tracking a deforming hand. The pose recovery step keeps the hand centered, at a constant scale and in the upright position. Here the pose optimisation is run to convergence and the object snapshot is processed through the latent space-based segmentation.

The quality of the segmentation is dictated by our ability to generate accurate contours from the latent space. Obviously,

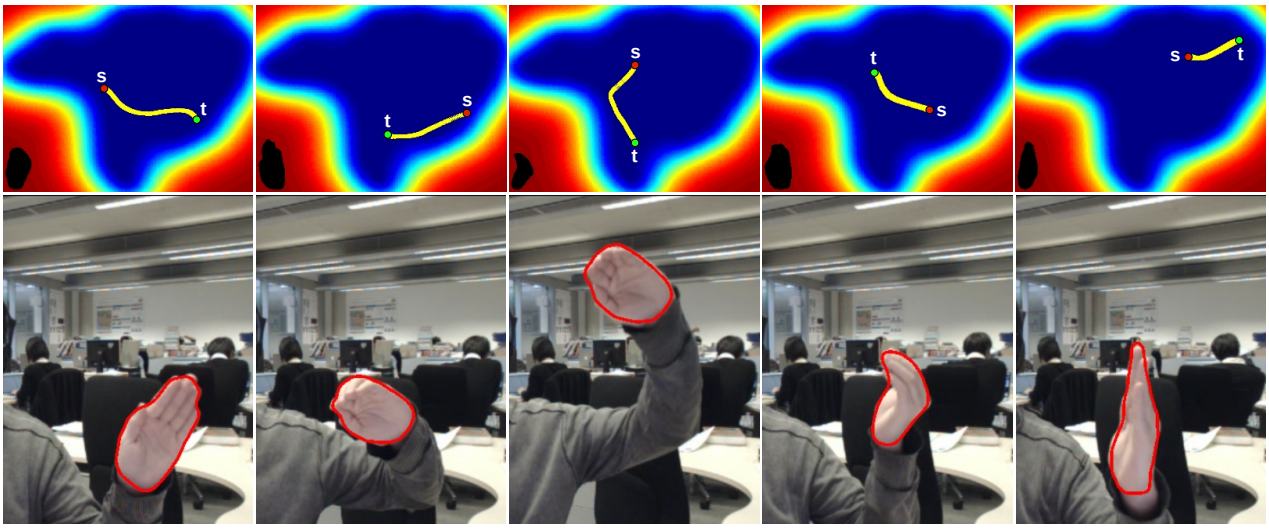

Figure 4. Tracking example. The latent space segmentation is combined with a 2D pose recovery step, with each minimisation being alternately iterated to convergence. The top row shows the latent space and convergence trajectory for each example frame. The starting shape is shown in black. The convergence trajectory starts from the *s* point and ends with the *t* point. The bottom row depicts the result.

accuracy increases with the number of dimensions of the latent space, as shown in Figure 5. A bigger number of dimensions will however result in sparser latent spaces, making optimisations more difficult.

Figure 5. Impact of latent space dimensionality. A 4D latent space (b) leads to a better segmentation than a 2D latent space (a).

The previous examples focused on finding and using a lower dimensional representation of silhouette kinematics. Another typical scenario in introducing shape information to the segmentation process is using dimensionality reduction to capture the variance of an object class, for instance the variance in the space of all possible cars, etc.

Figure 6 shows an example 2D latent space of side view car silhouettes. Our method is again able to generate very accurate contours from the latent space. Two examples of using this latent space to segment real images are shown in Figure 7. Here we compare our method to Grab Cuts [15] and the PWP level set segmentation of [1], which do not use any prior shape information. We also depict the per pixel value of $P_f - P_b$, where $P_f - P_b > 0$ and P_f and P_b are defined according to Equation 20. In both examples our method produces more accurate results, in spite of working with heavily corrupted per pixel posterior probabilities.

Figure 8 shows an example convergence of our joint pose and shape optimisation. Notice that even though the shapes were quite different and the initial pose was far from the correct one, the result is still correct.

Perhaps the primary reason for introducing shape information in segmentation is trying to achieve robustness to occlusions.

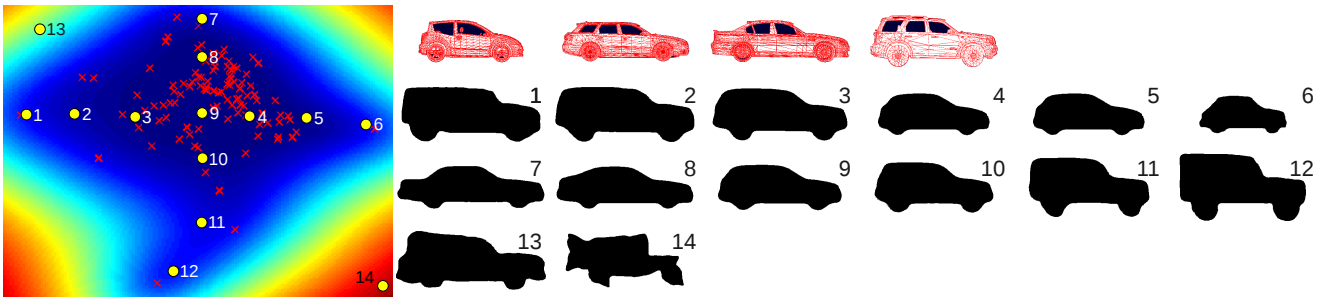

Figure 6. Example 2D latent space of side view car silhouettes (left), together with training data examples (right, top row) and latent space samples (right, bottom three rows). Samples 1-12 are taken from a low variance region of the space (so more likely valid shapes) while samples 13 and 14 are taken from a high variance region (so less likely valid shapes).

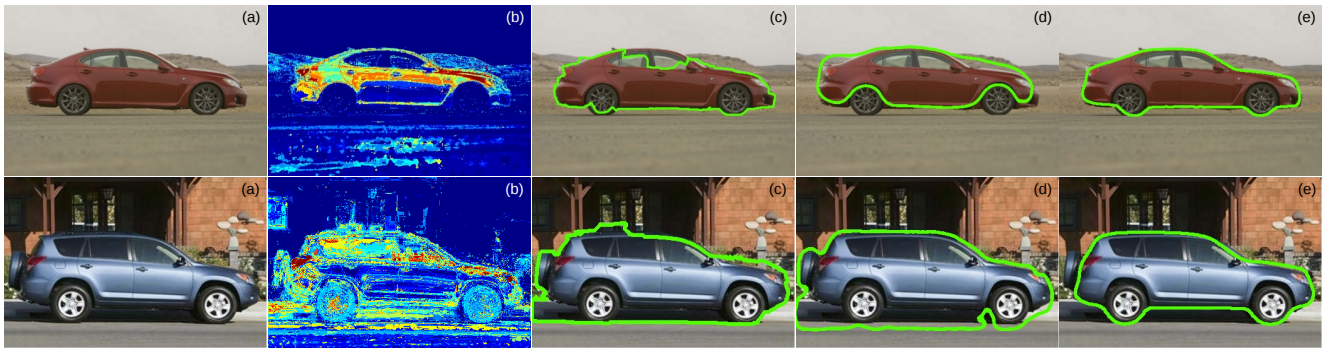

Figure 7. Example segmentations of cars: (a) – original images; (b) – map of $P_f - P_b$, where $P_f > P_b$; (c) – result using Grab Cuts [15]; (d) – result using PWP [1]; (e) – our result.

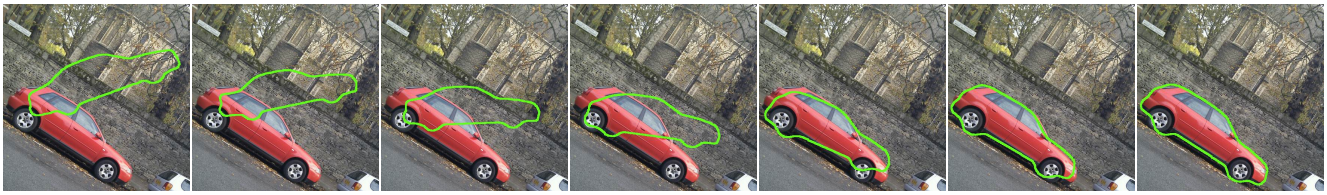

Figure 8. Example convergence steps for the pose/latent space one shot minimisation.

Figure 9 shows our algorithm dealing with high amounts of occlusion and still producing accurate results.

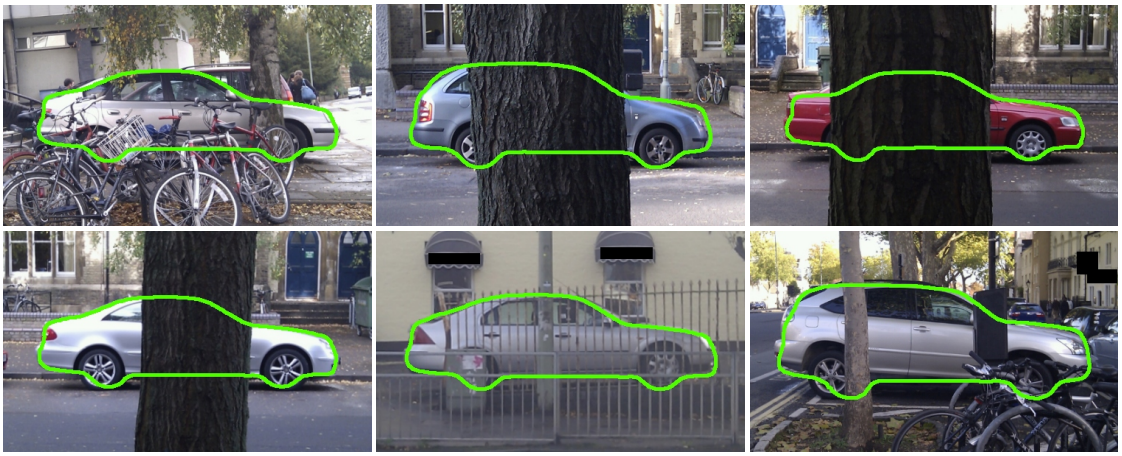

Figure 9. Examples of our algorithm dealing with occlusions.

We show two failure cases of our method, in Figure 10. In this case the specific type of car featured here (or anything close) did not appear among the training data points, so it cannot be generated accurately from a latent space. The results can be improved by introducing more training data. The second failure case is caused by the high degree of occlusion and especially the occlusion of the two extremities of the car, which would have "stretched" the segmentation.

Figure 10. Failure cases: left – failure due to lack of training data; right – failure due to high degree of occlusion.

Usually, when introducing shape priors to level set-based segmentation, PCA is used on the embedding functions [18]. In Figure 11 we show a qualitative comparison between the two methods, by depicting the 2D latent space obtained for a walking sequence with both methods.

A quantitative comparison between [18], [4] and our method, is shown in Figure 12. For each of our training data set (cars, hands, the star jump and the walking actions, a total of over 1700 images) we compute the latent space using the three methods and backproject (or search for) each latent point (for which we know the ground truth). We then measure the accuracy of each backprojected contour using the Pascal VOC Challenge [6] method i.e. divide the area of the intersection (of the test contour and the ground truth), with the area of the union. For a sensible choice of the KPCA kernel parameter (i.e. one that tries to avoid overfitting by compromising between accuracy and convergence), the results show an improvement with GPLVM in all cases. Compared to the PCA method, ours can always recover more variance from fewer dimensions. Using a 2D or 3D space, we produce similar (or better) results to a 10D PCA space. The difference of performance is especially big when the shape deformations are highly nonlinear (the walking and star jump cases). We also ran the accuracy test when using GP-LVM directly on the embedding functions, on the walk dataset. The results show with a minimal ($\sim$1%) improvement over EFDs. The main advantage of the EFDs here is the computational time required. Direct use of the embedding functions is **very** time and memory hungry and is also more prone to local minima. For example, learning a 2D embedding function space takes $\sim$8 hours and generating an embedding function from it takes $\sim$3 seconds. In contrast learning a 2D EFD space takes $\sim$20 minutes and generating EFDs from it takes $\sim$40 milliseconds.

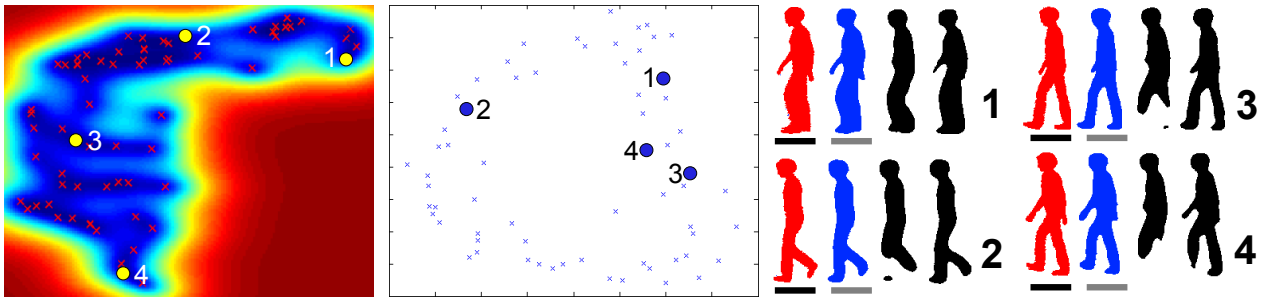

Figure 11. 2D latent space of a walking sequence, using our method (left) and the PCA-based one [18] (middle), with samples from both spaces (right). Red (black underline) shows the ground truth silhouette, blue (grey underline) the one obtained from a 2D space using our method, black-left the one obtained from a 2D PCA space and black-right the one obtained from a 10D PCA space.

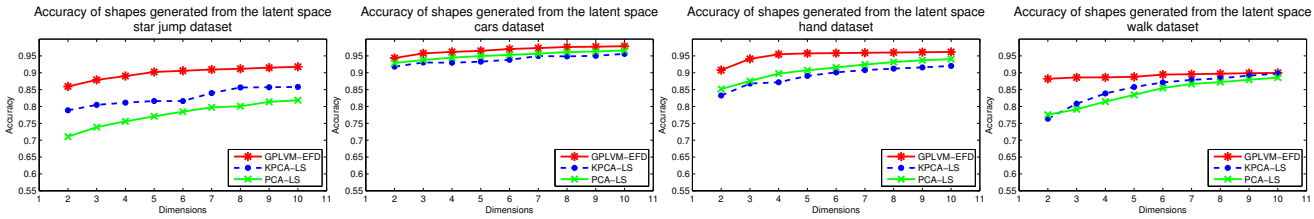

Figure 12. Accuracy comparison between our method (red, star marking) and the standard ones of using KPCA (blue, dot marking) and PCA (green, x marking) on the embedding functions.

8. Conclusion

In this article we have presented a method for finding nonlinear shape manifolds and using them as shape priors in level set-based segmentation and tracking. We represent shapes with elliptic Fourier descriptors and use GP-LVM to learn mappings from a lower dimensional space.

Compared to the standard method of using PCA on the level set embedding function, our method allows us to use fewer dimensions and keep more of the variance. Our experiments show that we can use just two or three dimensions to generate meaningful shapes, whereas PCA would require 10 or more.

Our method also provides a principled way of measuring the probability of a latent space point generating a valid contour. Segmentation is thus both variational and probabilistic, and is achieved by differentiating an image-driven, level set-based energy function, with respect to the lower dimensional latent space point. We also show that our segmentation can be easily linked with 2D pose recovery.

We are currently investigating several extensions to this work: using hierarchical latent spaces to improve segmentation quality, learning shared latent spaces and using spherical Fourier descriptors to generate 3D shapes from the lower dimensional latent space.

References

[1] C. Bibby and I. Reid. Robust real-time visual tracking using pixel-wise posteriors. In *ECCV 2008*, pages 831–844, 2008. 1, 4, 5, 7, 8

[2] D. Cremers and G. Funka-lea. Dynamical statistical shape priors for level set based tracking. *T-PAMI*, 28:1262–1273, 2006. 1

[3] D. Cremers, S. J. Osher, and S. Soatto. Kernel density estimation and intrinsic alignment for knowledge-driven segmentation: Teaching level sets to walk. In *IJCV*, pages 36–44, 2004. 1, 2

[4] S. Dambreville, Y. Rathi, and A. Tannenbaum. A framework for image segmentation using shape models and kernel space shape priors. *T-PAMI*, 30(8):1385–1399, 2008. 2, 9

[5] C. H. Ek, J. Rihan, P. H. Torr, G. Rogez, and N. D. Lawrence. Ambiguity modeling in latent spaces. In *MLMI 2008*, pages 62–73, 2008. 2

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 9

[7] F. P. Kuhl and C. R. Giardina. Elliptic fourier features of a closed contour. *CGIP*, 18(3):236–258, 1982. 2

[8] N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *JMLR*, 6:1783–1816, 2005. 2, 3, 5

[9] N. D. Lawrence and J. Quinonero-Candela. Local distance preservation in the gp-lvm through back constraints. In *ICML 2006*, pages 513–520, 2006. 3

[10] M. Leventon, E. Grimson, and O. Faugeras. Statistical shape influence in geodesic active contours. volume 1, pages 316–323 vol.1, 2000. 1

[11] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *JCP*, 79(1):12–49, 1988. 1

[12] V. Prisacariu and I. Reid. Pwp3d: Real-time segmentation and tracking of 3d objects. In *BMVC 2009*, 2009. 1, 5

[13] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. 2005. 3

[14] B. Rosenhahn, T. Brox, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *IJCV*, 73(3):243–262, 2007. 1

[15] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. *ACM TOG*, 23(3):309–314, 2004. 7, 8

[16] M. Rousson and N. Paragios. Shape priors for level set representations. In *ECCV 2002*, pages 78–92, 2002. 1, 2

[17] C. Schmaltz, B. Rosenhahn, T. Brox, D. Cremers, J. Weickert, L. Wietzke, and G. Sommer. Region-based pose tracking. In *IbPRIA 2007*, 2007. 1

[18] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, E. Grimson, and A. Willsky. A shape-based approach to the segmentation of medical imagery using level sets. *T-MI*, 22(2):137–154, 2003. 1, 2, 9

[19] L. A. Vese and T. F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *IJCV*, 50(3):271–293, 2002. 1, 4