Simultaneous Monocular 2D Segmentation, 3D Pose Recovery and 3D Reconstruction

Victor Adrian Prisacariu, Aleksandr V. Segal, Ian Reid

University of Oxford

Abstract. We propose a novel framework for joint 2D segmentation and 3D pose and 3D shape recovery, for images coming from a single monocular source. In the past, integration of all three has proven difficult, largely because of the high degree of ambiguity in the 2D - 3D mapping. Our solution is to learn nonlinear and probabilistic low dimensional latent spaces, using the Gaussian Process Latent Variable Models dimensionality reduction technique. These act as class or activity constraints to a simultaneous and variational segmentation – recovery – reconstruction process. We define an image and level set based energy function, which we minimise with respect to 3D pose and shape, 2D segmentation resulting automatically as the projection of the recovered shape under the recovered pose. We represent 3D shapes as zero levels of 3D level set embedding functions, which we project down directly to probabilistic 2D occupancy maps, without the requirement of an intermediary explicit contour stage. Finally, we detail a fast, open-source, GPU-based implementation of our algorithm, which we use to produce results on both real and artificial video sequences.

1 Introduction

The three tasks of 2D segmentation, 3D pose tracking and 3D shape recovery are fundamental to computer vision so there exists a large amount of research for each of them. Their interdependence is however often ignored and they are treated either separately or in pairs. For example, some systems recover 2D segmentation and 3D pose jointly, but require a fixed, known 3D model. Others jointly optimise 3D shape and 3D pose but require high quality segmentations.

In this paper we develop a method for simultaneous 2D segmentation, 3D pose recovery and 3D shape recovery, from one or multiple images coming from a monocular camera. The main issue with this aim has always been the very high ambiguity in the mapping from 2D silhouette to 3D pose and 3D shape. To deal with this we learn nonlinear and probabilistic low dimensional latent shape spaces. We represent shapes implicitly, as 3D level set functions, which we project (using known camera calibration) directly to 2D occupancy maps. Such an occupancy map is represented probabilistically with the $p = 0.5$ level giving the implicit representation of the contour. Both pose and shape recovery problems are cast as a single, joint minimisation of an image based energy function, searching inside an n-dimensional space jointly comprising 6 pose DoFs

and n-6 shape DoFs. Segmentation results automatically from the projection of the recovered 3D shape with the recovered 3D pose.

Our work is based on ideas from monocular 3D pose recovery. Assuming a *fixed* 3D shape, there are works such as [1–4] which propose methods for level-set based, monocular, simultaneous 3D pose recovery and 2D segmentation. A first such attempt was made in [1], where a level set-based Chan-Vese like energy function [5], with an added 2D shape term, is minimised alternately, first in an unconstrained manner, then as a function of the 6 DoF 3D pose of the known 3D model. The unconstrained part of the minimisation is removed in [2], where the energy function is evolved approximately, only as a function of 3D pose. A first variational approach to this problem was introduced in [3]. Here a region based (but not level set) energy function, summing an integral over the foreground with one over the background, is differentiated wrt. the pose parameters, using the Leibniz-Reynolds transport theorem. This results in an integral along the 2D contour and two surface integrals on the inside and outside regions of the contour. As in [5], the authors measure image statistics using the region's mean color and variance. This simple formulation allows their two surface integrals to collapse, which would not happen with a more complex energy function. A similar joint optimisation is used in [4], but the contour of the projection is represented implicitly (instead of explicitly as in [3]) as the zero-level of a level set embedding function. This allows for simpler math and more complex region statistics, which results in a larger convergence basin with fewer local minima.

Recovering 3D shape (along with 3D pose and 2D segmentation) is an extremely underconstrained problem, especially in the monocular case we are looking at in this paper. That is to say that the mapping from a 3D shape – pose pair is hugely multimodal. It is therefore unlikely that a full unconstrained 3D shape recovery could be performed successfully with no prior knowledge of pose or segmentation. There exist various methods for adding shape information to the segmentation and tracking process. For example, [6] represent shape knowledge probabilistically, using level set embedding functions and probabilistic confidence maps. This is added to the segmentation process in a maximum a posteriori fashion. Alternatively, instead of learning a model from multiple shapes, [7] use a single shape, but model 2D deformations using a homography. Perhaps the most common solution to our problem comes in the form of low dimensional latent shape spaces. In [8] for example, principal component analysis (PCA) is used to capture the variance in the space of shapes. Segmentation is then cast as a minimisation of an image-based error function in space. The similar method if [9] introduces nonlinearity by using Kernel PCA instead of PCA, followed by [10], where nonlinear and probabilistic spaces are used, by replacing Kernel PCA with GP-LVM. Nonlinearity is essential because most shape spaces tend to be nonlinear and modelling nonlinearity will decrease the number of dimensions needed to capture the shape variance. For example, in [10], a 2 dimensional GP-LVM space captures as much variance as a 10 dimensional PCA space.

The work most similar to ours is [11]. Here the authors look at improving image segmentation by use of spaces of 3D shapes. They use Kernel PCA to

learn these spaces and represent shapes implicitly using 3D level set embedding functions. The pose optimisation is the same as [3] i.e. using the Leibniz-Reynolds transport theorem and the simple image statistics of [5]. While their system does indeed produce both pose and shape, the authors only look at its segmentation ability. 3D poses and 3D shapes are never shown, quantified or examined.

Similar to this work, we use nonlinear dimensionality reduction to capture shape variance. Unlike [11] however, we use a method that is also probabilistic, in the form of Gaussian Process Latent Variable Models (GP-LVM). Our solution is better suited to capturing shape variance, compared to either PCA or Kernel PCA, as shown in [10, 12]. The GP-LVM generative process is closed form without making any assumptions on the type of energy function, as is done in [11]. This makes it faster and less prone to local minima. Finally, [11] needs manual tuning for the parameters of the kernel embedding functions, which makes it prone to overfitting. GP-LVM learn these parameters automatically. Our method is also similar to [12]. Here GP-LVM is used to learn joint spaces between 2D shapes and other various types of sets of parameters, ranging from 3D pose, to eye gaze and to 3D shape. One of the applications the authors explore is 3D reconstruction. While their system is able to generate 3D shapes, these are not used directly in the optimisation. They learn a shared 2D silhouette – 3D shape space and optimise for the 2D side view by finding the low dimensional latent space that generates the 2D contour that best segments the given image. This latent point is then back-projected to the set of parameters space. 3D shape is recovered when the sets of parameters describes a 3D shape. The obvious flaw of this system is that it will only recover 3D shape when the object is in a predefined (and pretrained) pose. Our method does not have this limitation.

From a fixed model 3D tracker point of view, our system is similar to [4], in that we minimise a level set based energy function wrt. the pose of known 3D model. Unlike [4] though, we represent shapes with 3D distance transforms which we project directly into 2D contour embedding functions. We do not render a vertex-based 3D model and then compute a 2D distance transform, as it is done in [4]. Our method therefore is, to our knowledge, the only *true* level set based 3D tracker, as other works always represent either the 3D shape or its 2D projection contour explicitly at some stage of the algorithm. One advantage of this formulation is that it naturally allows all points on the 3D shape to be considered in the optimisation, not just the ones that are visible from a given pose. As shown in [4], these are important to consider because, often, it is the invisible 3D points (under the current pose) that lead to changes in the shape of the projection. Another advantage is that it makes the energy function minimisation suitable for high level, GPU based, parallelisation.

When working with rigid objects, as they are being tracked throughout a sequence of frames, their shape does not change. Previous works ignored this fact. Here we impose shape consistency by alternating between optimising pose individually for each frame and shape jointly over multiple frames.

Consequently, our method has the following advantages over previous work: (i) we can generate more accurate models from our latent spaces (compared to

PCA or Kernel PCA based methods); (ii) the generated shapes are stable across multiple frames; (iii) the formulation for the rigid object tracking part of our system is more principled and parallelisable.

The remainder of this paper is structured as follows: we begin by describing our energy function in Section 2. We continue in Sections 3 and 4 with details about the minimisation wrt. pose and wrt. shape, respectively. The way we maintain shape consistency is presented in Section 5. Implementation details are explained in Section 6. We show results obtained by applying our method to several images and videos in Section 7 and conclude in Section 8.

2 Energy Function

Standard level-set based segmentation aims to minimise an integral over the entire image with the following form:

$$E(\phi) = \int_{\Omega} H_e(\phi) r_f(x) + (1 - H_e(\phi)) r_b(x) d\Omega \tag{1}$$

where Ω is the image domain, x is a pixel in this domain, ϕ is the 2D level set embedding function, H_e is the smoothed Heaviside function and r_f and r_b are two monotonically decreasing functions, measuring per pixel foreground and, respectively, background model matching scores.

Our energy function is similar:

$$E(\Phi) = \sum_{x \in \Omega} \Big(\pi(\Phi) r_f(x) + (1 - \pi(\Phi)) r_b(x) \Big) \tag{2}$$

where Φ is a 3D level set embedding function (instead of the usual 2D one denoted by ϕ) (discretised as an $256 \times 256 \times 256$ voxel cube). $\pi(\Phi)$ projects Φ to the equivalent of a smoothed Heaviside i.e. a function of value 1 inside the projection and 0 outside, with a smooth transition between the two regions.

To obtain r_f and r_b we first manually segment a few frames from the video to be analysed (between 5 and 7 frames from videos with lengths of 100 to 300 frames). Next, we extract 3×3 patches for each pixel in a band around the edge of each manual segmentation, combining RGB colour value and gradient orientation at that pixel. We then use these patches to train a two class random forest classifier, in a manner similar to [13]. We used 32 trees of depth 6. This method leads to considerably better image statistics when compared to either [4] or [11]. This step is not to be confused with a full image segmentation: here we are simply replacing the *per pixel* probability of foreground and background that is more usually obtained from a colour model with the probability obtained from a random forest classifier; this is *not* the segmentation step but analogous to the unary term in an MRF framework.

To define $\pi(\Phi)$ we interpret Φ as the log odds transform of a probability field:

$$\Phi(l) = \log \frac{p_{inside}}{1 - p_{inside}} \tag{3}$$

where $l = (x, y, z)$ represents a 3D voxel location inside the level set function and p_{inside} quantifies the probability of l being inside the closed 3D shape embedded by the level set function. p_{inside} is then extracted using the sigmoid function as:

$$p_{inside}(l) = sigmoid(\Phi(l)) = \frac{\exp(\Phi(l))}{1 + \exp(\Phi(l))} \tag{4}$$

It then follows that, for any image pixel (u, v), we can define a p_{fg} as the probability of (u, v) being the projection of a voxel from inside the 3D level set:

$$p_{fg}(u, v) = 1 - \prod_{l \text{ on ray}} \Big(1 - p_{inside}(l)\Big) \tag{5}$$

with the product being computed for all 3D points that project to (u, v).

For numerical stability we use the log space to compute this probability. We also introduce a parameter ζ which controls the smoothness of the transition between the inside and outside regions. Therefore, our final energy function is:

$$\pi(\Phi) = 1 - \exp\Big(\sum_{l \text{ on ray}} \log\big(1 - \frac{e^{\Phi(l)\zeta}}{e^{\Phi(l)\zeta} + 1}\big)\Big) \tag{6}$$

The smoothness parameter ζ is constant throughout our tests, with a value of 0.75. An example 3D model and corresponding projection is shown in Figure 1.

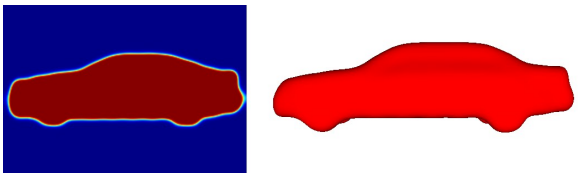

Fig. 1. Example 3D model and projection: left – projection, blue represents $p_{fg} = 0$, red represents $p_{fg} = 1$; right – the 3D model that generated the projection.

3 Pose Optimisation

We optimise pose in a manner similar to [4], by differentiating the energy function wrt. the 6 pose parameters λ_i, $i \in \{1, \ldots, 6\}$, three for transform and three for rotation. We use the Rodrigues notation to parametrise rotation.

The derivative follows as:

$$\frac{\partial E}{\partial \lambda_i} = -\sum_{x \in \Omega} \Big(\big(r_f(x) - r_b(x)\big) \exp(...) \sum_{l \text{ on ray}} \frac{e^{\Phi(l)\zeta}}{e^{\Phi(l)\zeta} + 1} \frac{\partial l}{\partial \lambda_i}\Big) \tag{7}$$

where exp(...) is as defined in Equation 6 and:

$$\frac{\partial l}{\partial \lambda_i} = \Big(\frac{\partial l}{\partial x}\frac{\partial x}{\partial \lambda_i} + \frac{\partial l}{\partial y}\frac{\partial y}{\partial \lambda_i} + \frac{\partial l}{\partial z}\frac{\partial z}{\partial \lambda_i} \Big) \tag{8}$$

with $[x, y, z]$ being OpenGL-style 3D normalised device coordinates. This representation allows $\pi(\Phi)$ to be responsible only for an orthogonal projection, making the selection of the points on the projection ray much easier.

Therefore, we can write:

$$\frac{\partial x}{\partial \lambda_i} = -f_u \frac{1}{Z^2}\big(Z\frac{\partial X}{\partial \lambda_i} - X\frac{\partial Z}{\partial \lambda_i}\big) \quad \frac{\partial y}{\partial \lambda_i} = -f_v \frac{1}{Z^2}\big(Z\frac{\partial Y}{\partial \lambda_i} - Y\frac{\partial Z}{\partial \lambda_i}\big)$$

$$\frac{\partial z}{\partial \lambda_i} = -\frac{1}{Z^2}\frac{\partial Z}{\partial \lambda_i} \tag{9}$$

where (f_u, f_v) represent the focal distance expressed in horizontal and vertical pixels and $[X, Y, Z]$ are 3D points in camera coordinates and functions of the pose (i.e. R and T) and their respective coordinates in the object frame (i.e. inside the level set voxel cube), $[X_0, Y_0, Z_0]$.

Finally, $\frac{\partial X}{\partial \lambda_i}$, $\frac{\partial Y}{\partial \lambda_i}$ and $\frac{\partial Z}{\partial \lambda_i}$ are computed analytically in a straightforward manner and $\frac{\partial l}{\partial x}$, $\frac{\partial l}{\partial y}$ and $\frac{\partial l}{\partial z}$ are computed numerically.

4 Shape Optimisation

As mentioned in the introduction, the mapping from a single silhouette to a 3D pose – 3D shape pair is hugely multimodal and ambiguous. There is less ambiguity when multiple frames are available, but, especially when those frames are consecutive, 3D shape recovery is still underconstrained. If however an assumption can be made on the class of object or on the activity the object is performing, a space for that class/activity can be learned and used to constrain the shape recovery. Such spaces have a very high dimensionality ($256 \times 256 \times 256$ dimensions in our case), but often actually lie on much lower dimensional manifolds. We find these manifolds using a nonlinear and probabilistic dimensionality reduction technique, called Gaussian Process Latent Variable Models [14].

Given a set of n variables $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$ of dimensionality d, GP-LVM learns a set of variables $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, of dimensionality q, with $q \ll d$, and the hyperparameters of a Gaussian Process (GP) mapping $\mathbf{X}$ to $\mathbf{Y}$. This is done by applying standard nonlinear optimisation techniques to maximise the probability of the data $\mathbf{Y}$, jointly wrt. the latent variables $\mathbf{X}$ and the hyperparameters of the GP. This probability is written as:

$$P(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^{n} \mathcal{N}(\mathbf{y}_i|0, \mathbf{K}) \tag{10}$$

where $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ is GP covariance matrix and $\kappa(\cdot, \cdot)$ is the GP kernel:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \theta_1 \exp\big(-\frac{\theta_2}{2}||\mathbf{x}_i - \mathbf{x}_j||^2\big) + \theta_3 + \theta_4 \delta_{ij} \tag{11}$$

with δ_{ij} being Kronecker's delta function and θ_{1-4} the GP hyperparameters.

The remainder of the GP-LVM learning process is beyond the scope of this paper and the interested reader is referred to [14].

To improve the likelihood of a good convergence, and to precondition the descriptors, in a manner similar to [12], we do not learn the space of level sets directly, but rather compute discrete cosine transforms (DCTs) for each level set and learn the space of DCT harmonic coefficients. We use 25 DCT harmonics for each 3D dimensions, for a total of $25 \times 25 \times 25$ harmonics. This is essential and a very important difference from [11]. Holding in memory a small dataset of just 100 exemplars of $256 \times 256 \times 256$ voxels requires 6.4GB of RAM memory available. This makes it very difficult to extend to larger datasets. The DCT compression allows is to work with the same dataset using just 5.96MB of RAM.

Given a $25 \times 25 \times 25$ descriptor $\mathbf{y}_p$, our level set function Φ is therefore the inverse DCT of $\mathbf{y}_p$, so $\Phi = IDCT(\mathbf{y}_p)$. $\mathbf{y}_p$ is also the high dimensional counterpart of a low dimensional latent point $\mathbf{x}_p$, so $\mathbf{y}|\mathbf{X} \sim N(\mu_p, \sigma_p^2)$, with:

$$\mu_p = \kappa(\mathbf{x}_p, \mathbf{X})\mathbf{K}^{-1}\mathbf{Y} \qquad \sigma_p^2 = \kappa(\mathbf{x}_p, \mathbf{x}_p) - \kappa(\mathbf{x}_p, \mathbf{X})\mathbf{K}^{-1}\kappa(\mathbf{x}_p, \mathbf{X})^T \tag{12}$$

As with the pose optimisation, to optimise shape, we differentiate our energy function, now wrt. each dimension of $\mathbf{x}$, which we denote by $\mathbf{x}_q$:

$$\frac{\partial E}{\partial \mathbf{x}_q} = -\sum_{x \in \Omega} \Big(\big(r_f(x) - r_b(x)\big) \exp(...) \sum_{l \text{ on ray}} \frac{e^{\Phi(l)}}{e^{\Phi(l)} + 1} \frac{\partial l}{\partial \mathbf{x}_q} \Big) \tag{13}$$

It can be shown that the derivative of the inverse DCT is the inverse DCT of the derivative. It follows that $\frac{\partial l}{\partial \mathbf{x}_q} = IDCT\big(\frac{\partial \mu}{\partial \mathbf{x}_q}\big)$, with:

$$\frac{\partial \mu}{\partial \mathbf{x}_q} = \frac{\partial \kappa(\mathbf{x}_q, \mathbf{X})}{\partial \mathbf{x}_i}\mathbf{K}^{-1}\mathbf{Y} \tag{14}$$

with μ defined in Equation 12.

The derivative of $\kappa(\cdot, \cdot)$ follows in a straightforward manner.

In [10, 12], where a 2D version of this GP-LVM based shape optimisation is proposed, the authors also use the variance σ^2 to drive the optimisation only along areas of the latent space with high likelihood. Throughout our testing, we did not find this to be necessary in the 3D case.

5 Shape Consistency

When multiple frames are available, the shape of a rigid object should be consistent among all the frames. The naive, adhoc solution to this problem is to choose a single informative frame, find the shape in this frame and use this shape to recover the pose (and implicitly the segmentation) in all the other frames. This is not often a good strategy, as we are unlikely to find a single frame that completely disambiguates the 3D shape, even when using a latent space shape prior.

A common solution to this problem, used throughout the 3D reconstruction literature, is to alternately iterate between shape and pose optimisations. We take a similar approach, by alternating between shape iterations (over all the frames) and pose iterations (for each frame separately). We can perform optimisation in joint space (and indeed have done so), but our experiments suggest that separating these into alternation between pose and shape has no penalty in terms of accuracy, and confers the convenience that we can impose fixed shape over over many frames rather more conveniently.

Since our energy function is a sum of per pixel values, it extends naturally to multiple frames:

$$E(\Phi) = -\sum_{f=1}^{F} \sum_{x \in \Omega_v} \Big(\pi_{\lambda_v}(\Phi) r_f(x) + (1 - \pi_{\lambda_v}(\Phi)) r_b(x) \Big) \tag{15}$$

where f is the frame, F is the total number of frames and $\pi_{\lambda_v}(\Phi)$ is the projection of Φ according to the pose parameters λ_v. The derivative of this energy function wrt. shape is just the sum of the derivatives wrt. shape for each individual frame.

When a new frame is available, we proceed as follows:

- Iterate pose for the new frame using the approximation from the previous frame.
- Repeat until convergence:
  - Iterate the q shape parameters, over all frames (using Equation 15).
  - Iterate the $6 \times f$ pose parameters over all frames, using the new shape.

Note that the same formulation also extends to multiple views coming from different cameras, but this is beyond the scope of this paper.

6 Implementation

To start the tracking process, the user must provide at least one manually segmented image and initial values for the latent point and pose. Potentially, these values could be obtained automatically using a classifier, while the manually segmented image could be obtained automatically using an unconstrained segmenter. Given these initial assumptions, the remaining operations are automatic.

Our algorithm is well suited for large scale parallelism, most operations being either per pixel or per voxel. To take advantage of this we have used the NVIDIA CUDA framework [15] to implement the complete inference algorithm (except for the random forest classification) on the GPU. The complete source code for our implementation is available online at http://www.robots.ox.ac.uk/∼lav.

A standard joint shape pose iteration of our algorithm proceeds as follows:

- Compute the GP-LVM posterior mean using Equation 12.
- Create the level set voxel cube by decompressing the GP-LVM posterior mean with the inverse DCT transform.
- Compute the GP-LVM posterior mean gradient using Equation 14.

– Compute $\frac{\partial l}{\partial \mathbf{x}_q}$ using the inverse DCT transform.
– Project the voxel cube using our projection function (Equation 6).
– Compute and sum per voxel derivatives w.r.t. pose and shape.

Processing Stage	Time
GP-LVM posterior mean computation	0.51 ms
Compute level set voxel cube with the inverse DCT on the GP-LVM mean	31.17 ms
GP-LVM posterior mean gradient	0.21 ms
$\frac{\partial l}{\partial \mathbf{x}_q}$ for a two dimensional space (using the inverse DCT)	58.15 ms
Voxel cube projection	2.69 ms
Per voxel shape/pose derivative	7.12 ms

Table 1. Per iteration processing times

A detailed summary of the processing times required for each of these steps is shown in Table 1. We used 640×480 images and an NVIDIA GTX 480 video card. The average processing time per pose iteration is $\sim$*10ms* and per shape iteration is $\sim$*100ms*. Our algorithm usually converges within 25-50 iteration, so our average per image processing time is between 2.5 and 5 seconds. Note however that (i) in the pose optimisation case, around half the processing time is spent doing the final summation and (ii) in the shape optimisation case, $\sim$90% processing time is spent doing the inverse DCT. Furthermore, since, potentially, the shape does not need to be iterated for every frame, with some further optimisations, our algorithm would be suitable for real time applications.

7 Results

We tested our algorithm using a two dimensional latent space learned from a 100 car dataset built from Google SketchUp models. For this we show several qualitative examples, a quantitative comparison between the 2D segmentation and 3D reconstruction accuracy obtained by our algorithm and the one of [12] and a quantitative comparison between the 3D poses generated by our method and those of the PTAM system of [16]. We also compare our random forest classifier (RF) with the pixel wise posteriors formulation of [17, 4, 12, 10] (PWP) and provide evidence maintaining shape consistency constraints improves results.

Figure 2 shows our car shape latent space. Blue indicates low variance (i.e. a trusted region of the space) while red indicates high variance (a region with low probability of generating valid shapes). A sample run of our system using this space is shown in Figure 3. Here we intentionally started far from the correct value to show that our algorithm is able to converge despite gross shape and pose errors. We adapted both shape and pose simultaneously and the algorithm converged in ~ 300 gradient descent iterations. More powerful optimisation methods could potentially be used, as there are no mathematical impediments

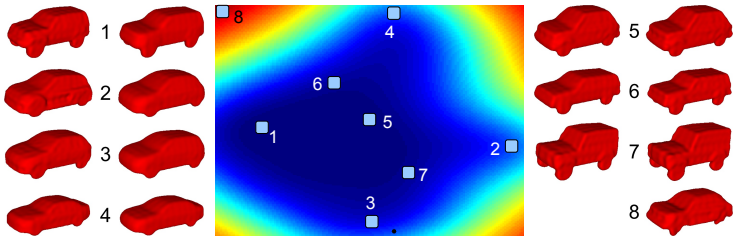

Fig. 2. Example 2D latent space, capturing car inter-class variance. From each shape pair, left represents the ground truth and right the generated one. Shapes 1-7 are points inside the low variance region of the space (in blue), while shape 8 is from the high variance area (in red). The sample from the high variance area does not have a corresponding ground truth because it was not part of the original training data.

to computing a second derivative of our energy function (unlike standard level set formulations that often require the derivative of the Dirac delta).

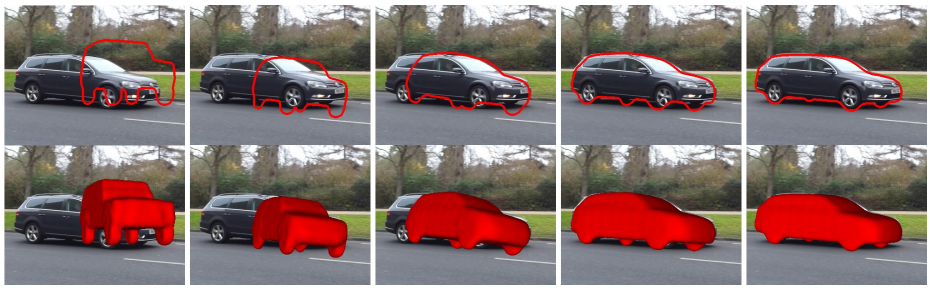

Fig. 3. Example shape and pose convergence for our algorithm.

Two tracking results using the car latent space, for two different types of car (sedan and hatchback), are shown in Figure 6. Both cars are being tracked successfully throughout their respective sequences.

In Figure 4 we compare the 2D segmentation and 3D reconstruction accuracies of our system and the one from [12], where shared shape spaces are learned between 2D car side views and 3D car models. When the system from [12] is shown a car side view the results are good. In all other poses however, the system from [12] fails, whereas ours does not.

Figure 5 shows a comparison between our method and the simultaneous location and mapping PTAM method of [16]. Here, on the same video sequence, we tracked the pose of the camera using PTAM and the pose of the car using our system. The two system produce very similar poses, in spite of the fact that PTAM uses features from the whole image, while our method uses just features from the contour of the car.

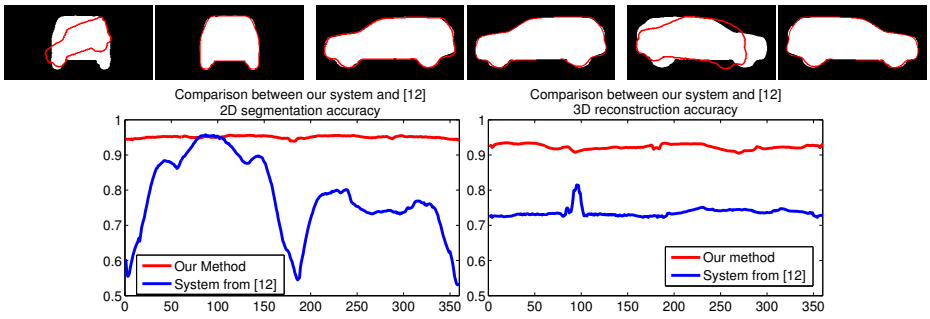

Fig. 4. 2D Segmentation and 3D reconstruction comparison with the system from [12]. We used a known 3D model which we rotated 360 degrees around the Z axis. Example frames are shown above, with results from [12] on the left and our results on the right. On the charts, the X axis shows rotation angle and the Y axis shows accuracy.

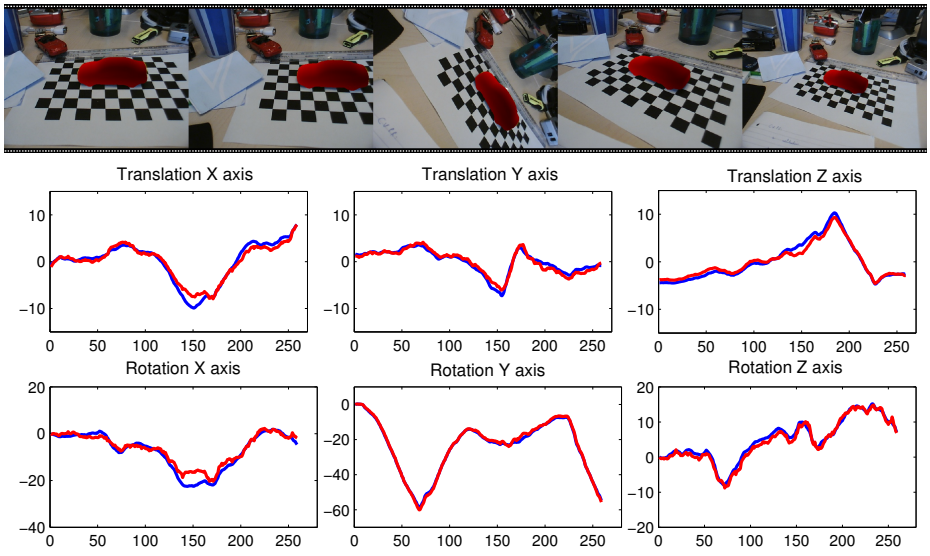

Fig. 5. 3D pose tracking comparison between our method (red) and the system from [16] (blue). The video sequence used is shown above. The X axis shows the frame number while the Y axis shows centimetres for transition and degrees for rotation.

In Figure 7 we compare the pixel wise fg/bg separation obtained by using RFs and the PWP formulation of [17, 10, 12, 4]. For each pixel, a fg probability P_f and a bg probability P_b are computed using both methods. Figure 7 shows the difference between the two probabilities, where $P_f - P_b > 0$. The RF classifier achieves better separation, which in turn leads to more reliable tracking.

Enforcing shape consistency is essential. Figure 8 shows two charts, test 1 corresponding to the video sequence from Figure 9 and test 2 for the one in Figure 9. We ran each sequence twice, once with and once without shape consistency constraints. Both times the result was smoother when shape consistency was used. Furthermore, in the second example the lack of shape consistency leads to a complete failure of tracking. As shown in Figure 9, when the cyclist occludes the car, not using shape consistency causes the algorithm to incorrectly adapt shape, ultimately leading to failure. When shape consistency is kept the algorithm can use information from unoccluded frames, which insures stable tracking.

Fig. 6. Film strips showing our algorithm successfully tracking different types of cars in different environments.

8 Conclusions

In this article we proposed a method for simultaneous 2D segmentation, 3D pose recovery and 3D shape reconstruction. We have shown that constraining the shape search space with Gaussian Process Latent Variable Models latent spaces is effective and leads to high quality reconstructions. The rigid body part of our formulation is, to our knowledge, the first true level set based tracker, since we don't switch from implicit shape representations to explicit ones at any point during the algorithm. Shapes are represented via 3D level set embedding

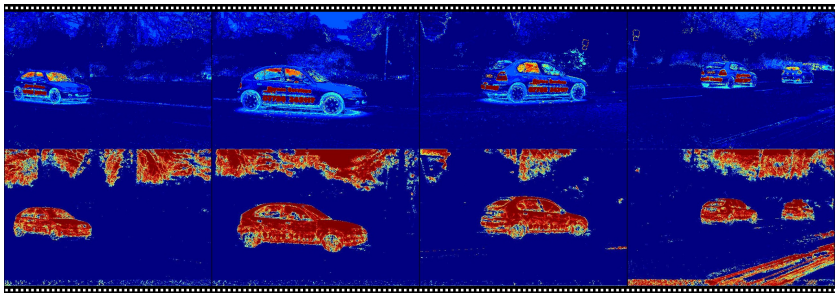

Fig. 7. Comparison between the foreground / background separation provided by the PWP method used of [17, 10, 12, 4] and our RF classifier. Warm colours indicate high probability and cold ones low probability.

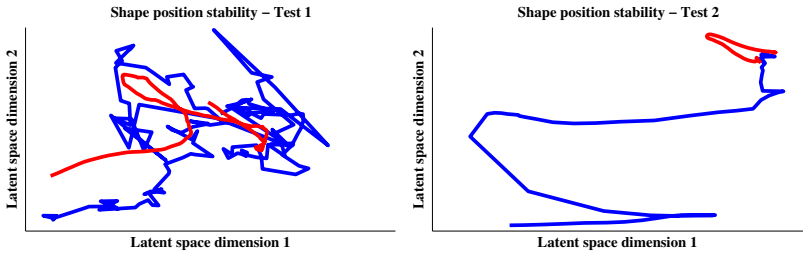

Fig. 8. Charts showing recovered latent space positions, with (in red) and without (in blue) shape consistency. When shape consistency is used the trajectories in the recovered latent space is smoother and tighter.

Fig. 9. Failure due to occlusion, combined with lack of shape consistency. When the cyclist passes in front of the camera and shape consistency is not kept the system fails (top two rows). When information from multiple frames is integrated the system successfully tracks through the occlusions (bottom two rows).

functions (discretised as voxel cubes) and are projected down directly to 2D occupancy maps, avoiding the need for an explicit contour representation. We also propose a fast, potentially real time, GPU based implementation, which we make available online as an open source package.

One possible extension to this work is the processing of MRI or Ultrasound data. Shared spaces, as used [12], between 3D shapes and sets of parameters could also be learned. This could, for example, enable articulated poses to be recovered as part of our current framework.

Acknowledgement This work was supported by the REWIRE FP7 project and by EPSRC through a doctoral prize award.

References

1. Rosenhahn, B., Brox, T., Weickert, J.: Three-dimensional shape knowledge for joint image segmentation and pose tracking. IJCV **73** (2007) 243–262
2. Schmaltz, C., Rosenhahn, B., Brox, T., Cremers, D., Weickert, J., Wietzke, L., Sommer, G.: Region-based pose tracking. In: IbPRIA 2007. (2007) 56–63
3. Dambreville, S., Sandhu, R., Yezzi, A., Tannenbaum, A.: Robust 3D pose estimation and efficient 2D region-based segmentation from a 3D shape prior. In: ECCV 2008. (2008) 169–182
4. Prisacariu, V.A., Reid, I.: PWP3D: Real-Time Segmentation and Tracking of 3D Objects. (IJCV) 1–20
5. Vese, L.A., Chan, T.F.: A multiphase level set framework for image segmentation using the mumford and shah model. IJCV **50** (2002) 271–293
6. Rousson, M., Paragios, N.: Prior Knowledge, Level Set Representations & Visual Grouping. IJCV **76** (2008) 231–243
7. Riklin-raviv, T., Kiryati, N., Sochen, N.: Prior-based segmentation and shape registration in the presence of projective distortion. IJCV **72** (2007) 309 – 328
8. Tsai, A., Yezzi, A., Wells, W., Tempany, C., Tucker, D., Fan, A., Grimson, E., Willsky, A.: A shape-based approach to the segmentation of medical imagery using level sets. T-MI **22** (2003) 137–154
9. Dambreville, S., Rathi, Y., Tannenbaum, A.: A framework for image segmentation using shape models and kernel space shape priors. T-PAMI **30** (2008) 1385–1399
10. Prisacariu, V., Reid, I.: Nonlinear shape manifolds as shape priors in level set segmentation and tracking. In: CVPR 2011. (2011) 2185–2192
11. Sandhu, R., Dambreville, S., Yezzi, A., Tannenbaum, A.: A Nonrigid Kernel-Based Framework for 2D-3D Pose Estimation and 2D Image Segmentation. T-PAMI **33** (2011) 1098–1115
12. Prisacariu, V., Reid, I.: Shared shape spaces. In: ICCV 2011. (2011)
13. Santner, J., Unger, M., Pock, T., Leistner, C., Saffari, A., Bischof, H.: Interactive Texture Segmentation using Random Forests and Total Variation. In: BMVC 2009. (2009)
14. Lawrence, N.: Probabilistic non-linear principal component analysis with gaussian process latent variable models. JMLR **6** (2005) 1783–1816
15. NVIDIA: NVIDIA CUDA Programming Guide 4.1. (2012)
16. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: ISMAR 2007. (2007) 1–10
17. Bibby, C., Reid, I.: Robust real-time visual tracking using pixel-wise posteriors. In: ECCV 2008. (2008) 831–844