PWP3D: Real-time segmentation and tracking of 3D objects

Victor A. Prisacariu
victor@robots.ox.ac.uk
Department of Engineering Science, University of Oxford

Ian D. Reid
ian@robots.ox.ac.uk
Department of Engineering Science, University of Oxford

We develop a fast and accurate image segmentation and pose tracking method based on the assumption that, given an accurate 3D model of an object, its segmentation from any given image is fully defined by its pose. Our method allows for fast 2D–3D pose tracking and 2D segmentation using a single, unified, energy function. Unlike most previous work our method consists of a single optimisation step, rather than the usual two step approach.

Inspired by [1] we aim to maximise the posterior per-pixel probability of foreground and background membership as a function of pose. This was shown to yield a better behaved energy function (in the 2D case), when compared to a standard level set formulation such as [2]. In our case we assume a known 3D model and calibrated camera, and seek the six degree of freedom rigid transformation that maximises the pixel-wise posterior energy function. Like [1] we represent the region statistics by colour histograms and adapt these online. The minimization is done using gradient descent (though other more sophisticated minimisation techniques are not precluded).

Let $\mathbf{X_0} = [X_0, Y_0, Z_0]^T \in \mathbb{R}^3$ be a 3D point in the object coordinate frame, while $\mathbf{X} = [X, Y, Z]^T = \mathbf{R}\mathbf{X_0} + \mathbf{T} \in \mathbb{R}^3$ is a point in camera coordinate frame. The image itself is denoted $\mathbf{I}$, and the image domain by, $\Omega \subset \mathbb{R}^2$, with an area element denoted $d\Omega$. An image pixel $\mathbf{x} = [x, y]$ in this domain has the value $\mathbf{I}(\mathbf{x}) = \mathbf{y}$ (in our experiments this is an RGB value).

We assume a calibrated camera, so without loss of generality, $x = X/Z, y = Y/Z$. The pose parameters – i.e. rotation and translation relating points $\mathbf{X_0}$ in the object frame to the camera frame – are denoted by λ_i.

The projection of the occluding contour of the object is denoted by the curve $\mathbf{C}$. This closed curve is the zero level-set of an embedding function $\Phi(\mathbf{x})$ [2]. The curve $\mathbf{C}$ segments the image into disjoint regions Ω_f and Ω_b, for foreground and background respectively. Each region has its own statistical appearance model, $P(\mathbf{y}|M)$ where M is one of M_f or M_b.

Finally, by $H_e(x)$ we denote the smoothed Heaviside step function and by $\delta_e(x)$ the smoothed Dirac delta function [2].

Assuming the level set formulation of [1], the energy given by the negative log (posterior) probability of the shape of the contour (encoded by the embedding function Φ), given the image data, is:

$$P(\Phi|\Omega) = \prod_{i=1}^{N} \Big(H_e(\Phi) P_f + \big(1 - H_e(\Phi) P_b\big) \Big) \Rightarrow$$
$$E(\Phi) = -\log(P(\Phi|\Omega)) = -\sum_{i=1}^{N} \log \Big(H_e(\Phi) P_f + \big(1 - H_e(\Phi) P_b\big) \Big) \tag{1}$$

where P_f and P_b are the posterior probabilities $P(M_f|\mathbf{y})$ and $P(M_b|\mathbf{y})$ respectively (further details are given in [1]).

We differentiate with respect to the pose parameters λ_i, aiming to evolve the contour in a space parametrized by the pose parameters:

$$\frac{\partial E}{\partial \lambda_i} = P(\Phi|\Omega) \sum_{i=1}^{N} \frac{P_f - P_b}{H_e(\Phi) P_f + \big(1 - H_e(\Phi)\big) P_b} \frac{\partial H_e(\Phi)}{\partial \lambda_i} \tag{2}$$

$$\frac{\partial H_e\big(\Phi(x,y)\big)}{\partial \lambda_i} = \frac{\partial H_e}{\partial \Phi} \left(\frac{\partial \Phi}{\partial x} \frac{\partial x}{\partial \lambda_i} + \frac{\partial \Phi}{\partial y} \frac{\partial y}{\partial \lambda_i} \right) \tag{3}$$

At each iteration of the algorithm we re-compute Φ as the signed-distance transform from the projected contour (on the GPU for efficiency), and the differentials $\frac{\partial \Phi}{\partial x}$ and $\frac{\partial \Phi}{\partial y}$ follow trivially.

Every 2D point $\mathbf{x}$ on the contour of the (true) projection of the 3D model has at least one corresponding 3D point $\mathbf{X}$.

$$\frac{\partial x}{\partial \lambda_i} = \frac{\partial}{\partial \lambda_i} \left(\frac{X}{Z} \right) = -\frac{1}{Z^2} \left(Z \frac{\partial X}{\partial \lambda_i} - X \frac{\partial Z}{\partial \lambda_i} \right) \tag{4}$$

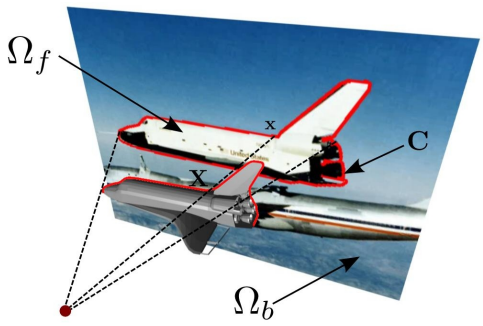

Figure 1: Representation of the object showing: the contour of the projection $\mathbf{C}$ and the corresponding contour around the visible part of the 3D model, the foreground region Ω_f and the background region Ω_b, a 2D point $\mathbf{x}$ on the contour and its corresponding 3D point in the object coordinate frame $\mathbf{X}$

and similarly for y. Noting that X, Y and Z are simple functions of the pose parameters, it is then straightforward, via a further application of the chain rule, to obtain $\frac{\partial X}{\partial \lambda_i}$, $\frac{\partial Y}{\partial \lambda_i}$ and $\frac{\partial Z}{\partial \lambda_i}$.

For image points on the inside of the contour, we backproject to the closest point on the object. For image points not exactly on the contour but outside (and which therefore have no true backprojection), we approximate $\mathbf{X}$ as the backprojection of the point on the contour which is closest in the image. Since the actual calculation takes place in a narrow band around the projected contour, this does not introduce gross errors.

When rotating and translating a 3D model some surfaces appear and others disappear, as the pose transitions between so-called "generic" views. The singular poses between generic views are characterised by ambiguous back-projections: a back-projected image contour point may have multiple tangencies with the object. In these cases the rate of change of the image contour with respect to the pose parameters is not only dictated by the closest points on the 3D model but also by the (infinitesimally invisible) furthest points. In these transition cases we include terms from *both* the closest and distant points to give greater robustness when moving between different generic views of the object.

We allow online adaptation of the foreground and background statistical models. Like [1], this is achieved using linear opinion pools with variable learning rates α_i, $i = \{f, b\}$.

$$P_i(\mathbf{y}|M_i) \leftarrow (1 - (\alpha_i)) P_i(\mathbf{y}|M_i) + (\alpha_i) P_i(\mathbf{y}|M_i) \tag{5}$$

The factors α_i are fixed for all frames. In all our experiments $\alpha_f = 0.01$ and $\alpha_b = 0.02$. To avoid polluting the histograms with incorrect data, we only update when the energy minimization has converged.

Most of the tasks involved in one iteration of our algorithm operate per-pixel and so we can achieve significant gains by exploiting this high degree of parallelism. We make use of the NVIDIA CUDA framework to implement various steps on a GPU, so we are able to achieve real time performance.

In the paper we include our results which demonstrate robustness to motion blur, occlusions and imperfect model information. We also showcase the benefits of using the pixel-wise posterior formulation (rather than the standard log-likelihood one) with a comprehensive experiment.

[1] Charles Bibby and Ian Reid. Robust real-time visual tracking using pixel-wise posteriors. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 831–844, 2008.

[2] Daniel Cremers, Mikael Rousson, and Rachid Deriche. A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape. *International Journal of Computer Vision*, 72(2):195–215, 2007.