Simultaneous 3D Tracking and Reconstruction on a Mobile Phone

Victor Adrian Prisacariu* University of Oxford
Olaf Kähler† University of Oxford
David W. Murray‡ University of Oxford
Ian D. Reid§ University of Adelaide

ABSTRACT

A novel framework for joint monocular 3D tracking and reconstruction is described that can handle untextured objects, occlusions, motion blur, changing background and imperfect lighting, and that can run at frame rate on a mobile phone. The method runs in parallel (i) level set based pose estimation and (ii) continuous max flow based shape optimisation. By avoiding a global computation of distance transforms typically used in level set methods, tracking rates here exceed 100Hz and 20Hz on a desktop and mobile phone, respectively, without needing a GPU. Tracking ambiguities are reduced by augmenting orientation information from the phone's inertial sensor. Reconstruction involves probabilistic integration of the 2D image statistics from keyframes into a 3D volume. Per-voxel posteriors are used instead of the standard likelihoods, giving increased accuracy and robustness. Shape coherency and compactness is then imposed using a total variational approach solved using globally optimal continuous max flow.

1 INTRODUCTION

Object-wise 3D reconstruction is a cardinal problem in computer vision, with much work being dedicated to it throughout recent years. Most current techniques however require, on the one hand powerful hardware for any sort of timely inference and, on the other, a complicated extrinsic camera calibration procedure, controlled lighting and accurate pre-existing segmentations.

The aim of our work is to eliminate such constraints and provide a system that **(i)** can work in a real world environment, under real world conditions and **(ii)** has a computational cost that is low enough to allow it to run in real time on a mobile phone, without any additional hardware. We believe that such a solution has the potential of "democratising" 3D object reconstruction (and tracking) much in the same way current research is doing, for example, for 3D articulated pose recovery or panorama stitching.

In this work we recover both 3D shape and 3D pose simultaneously and in parallel. Tracking is region and level set based and uses nonlinear optimisation to find the 3D rigid pose that leads to a maximal separation between foreground and background image areas with known statistics. This makes it robust to image artefacts, as caused, for example, by occlusions or motion blur. The implementation of such a tracker boils down to repeated renderings of a 3D model and computations of a level set embedding function (i.e. distance transform) of the rendering. Because of this, related tracking approaches have inevitably had very high computation costs, making real time performance only possible using GPU processing. We instead use a hierarchical rendering pipeline, avoid the global computation of the distance transform and its derivatives and augment the image data with orientation information from the mobile phone's inertial sensor (IMU). This allows for real-time (over 20 fps) performance even on a mobile phone, without the use of GPU hardware. On a desktop PC speeds of over 100 fps are possible.

Reconstruction is keyframe based and is split into two phases. At each keyframe, we back-project 2D image statistics of foreground and background (extracted from the camera image) and fuse them into 3D probability volumes capturing inside/outside probability. This means that, unlike most current research, we do not require prior object segmentation at every frame. Furthermore, we use per-voxel posteriors, which leads to an increase in accuracy and robustness to imperfect image statistics over the standard approach of using likelihoods. After a number of registered keyframes, we impose shape coherency and compactness using a globally optimal total variational / continuous max flow approach.

The paper is structured as follows. We begin in Section 2 with an overview of recent advances in tracking and reconstruction and summarise how they relate to our method. We continue in Section 3 with a description of the graphical model and notation underlying our method. The tracking and reconstruction parts of our algorithm are detailed in Sections 4 and Section 5, respectively. Implementation details are given in Section 6 and results are shown in Section 7. We conclude in Section 8.

2 RELATED WORKS

From the point of view of tracking fixed 3D shapes, our work is related to region-based tracking. This idea was originally proposed in [16], where the Chan-Vese level set energy function [20] is minimised in a two step process, first in an unconstrained manner and second wrt. the 6 DoF pose of the known 3D shape. This two stage approach was later removed in favour of an approximate pose-only evolution in [18]. Our work is most similar to the related, but more recent work of [13] and [15]. These are variational formulations of [16], minimising the pixel-wise posteriors level set energy function of [2] directly wrt. 3D pose, using gradient descent. The main difference between these approaches lies in the 3D shape representation, [13] using a 3D mesh and [15] using a volumetric 3D signed distance transform. Here we use a volumetric representation, but otherwise follow a mathematical formalism very similar to that of [13]. Our novelty however is in the computation of the gradient. Whereas [13] require a GPU and achieve a performance maximum of 25 fps and [15] a maximum of 10 fps, in this work we are able to achieve almost the same speed on a mobile phone, and considerable higher speeds on a desktop PC, both without using a GPU.

Our method is also related to [22, 3, 14] in its use of joint visual-IMU pose recovery. Unlike these works however, here we use a very lightweight fusion mechanism, with the IMU serving as the main source of rotation, with occasional visual-based corrections.

The reconstruction part of our method is primarily related to methods that recover the visual hull from silhouettes using energy minimisation techniques, examples being [21, 19, 5, 4, 8]. Of these, the approach proposed in [21] locally minimises the reprojection error between surface and observed image intensities, making it slow and subject to local minima. The other approaches follow the reverse strategy, by backprojecting the binary segmented views (in [19]) or image statistics extracted from the views (in [5, 4] and [8]) into 3D volumetric representations. All these methods use globally convergent nonlinear minimisation, graph-cuts in [19, 5, 4] and total variational primal-dual optimisation in [8]. In this work, similar to [5, 4, 8] we backproject image statistics into a 3D volumetric

*e-mail: victor@robots.ox.ac.uk
†e-mail: olaf@robots.ox.ac.uk
‡e-mail: dwm@robots.ox.ac.uk
§e-mail: ian.reid@adelaide.edu.au

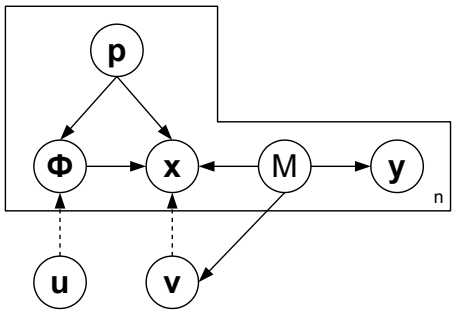

Figure 1: Graphical model for our method

representation. Unlike those works however, we use voxel posteriors instead of likelihoods, which leads to better performance and robustness. Similar to these methods we use a globally convergent optimisation in the form of continuous max flow [23]. This is much faster than the related but discrete graph cuts used in [5, 4, 19] and has faster convergence compared to the primal-dual total variational optimisation used in [8]. Finally, unlike all these works, we do not assume given poses, controlled lighting or static environments, but rather use tracker estimated poses.

Such simultaneous reconstruction and tracking has also been attempted before, with two recent representative methods being [11] and [1]. In [11] the authors assume a static background and track and reconstruct a feature point cloud. The convex 3D shape is then built using Delaunay tetrahedralisation. A static background is also assumed in [1], where the camera pose is tracked using PTAM [7]. The object is segmented in each frame using graph cuts and the segmentations are merged into a 3D volume using a voting based fusion method. In this work we require no explicit per frame segmentation and no point features, either in the environment or on the object. Therefore the environment can be fully dynamic and the system is robust to occlusions and motion blur while still being fully principled (i.e. not requiring ad-hoc silhouette fusion approaches).

A final category of related works included systems for simultaneous localisation and mapping (SLAM) such as PTAM [7] and DTAM [9]. These attempt to track and reconstruct the full environment presented to the camera, producing sparse [7] or dense [9] reconstructions. We limit our model of the world to the actual object that we intend to reconstruct, whereas PTAM and DTAM build a model of everything observed by the camera. Of course the object can be segmented out of the full scene model afterwards and the reconstruction step might even benefit from having a broader field of reference as the observed background will most certainly provide additional constraints for estimating the camera motion. However performance of such systems may degrade in dynamic environments, i.e. with people walking in the background. By modelling only the object and ignoring the rest of the scene, dynamic movements, light changes and anything else not captured by our model are completely ignored as long as they only affect the background.

3 Graphical Model

Figure 1 shows the graphical model describing our method. An overview of the practical implementation is shown and discussed in Section 6. The 3D shape we track and reconstruct is denoted with the random variable $\mathbf{u}$. We use a volumetric shape representation which makes $\mathbf{u}$ a distribution over 3D volumes, with probability 0 identifying voxels outside the shape and 1 inside. The maximum likelihood estimate of the outline of the shape is the 0.5 level set of $\mathbf{u}$. We denote by $\mathbf{v}$ a distribution over voxels in this volume.

We assume a known set of n views. For each of these views, we denote the distribution over 3D poses of the 3D object with $\mathbf{p}$. We use the standard six degree of freedom representation for pose (three for translation and three for Rodrigues parametrised rotation). The contour of the projection of $\mathbf{u}$ under the pose $\mathbf{p}$ is embedded inside a 2D signed distance transform (SDF), which we denote by Φ. Similarly, a voxel $\mathbf{v}$ under the pose $\mathbf{p}$ projects to a pixel location $\mathbf{x}$. In Figure 1 we denote these deterministic relationships with dotted lines. Note that in this work we consider the 3D poses to be independently distributed. This could be changed, allowing for a motion model to be added.

Each pixel location $\mathbf{x}$ has a corresponding colour $\mathbf{y}$. As with other region-based methods, we assume as known a pair of per-view foreground / background colour models, which we denote by $M \in \{M_f, M_b\}$. Here these are $32 \times 32 \times 32$ bin RGB histograms.

Joint inference on the full graphical model is not tractable, especially on a mobile device. As other works have done before us, we therefore chose to split the inference into a tracking stage, i.e. an estimation of the pose $\mathbf{p}$ and a reconstruction stage, i.e. an estimation of the shape $\mathbf{u}$. In the interest of brevity we use $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{p}$ to denote both estimate and respective probability distribution for the remainder of the paper.

4 Pose Optimisation

The projection of a known 3D shape $\mathbf{u}$, given a pose $\mathbf{p}$, separates any image into a foreground and a background region. Assuming known colour statistics for these regions ($M \in \{M_f, M_b\}$ in the graphical model), the pose optimisation aims to maximise the discrimination between foreground and background wrt. the pose $\mathbf{p}$. The theoretical foundations of this approach have been introduced in [13], and we summarise them in the following.

Treating $\mathbf{u}$ and $\mathbf{v}$ as known in the graphical model, the joint probability becomes similar to the one presented by Bibby and Reid in [2] for the case of 2D tracking and segmentation. This is written as:

$$P(\mathbf{x}, \mathbf{y}, \Phi, M) = P(\mathbf{x}|\Phi, M)P(\mathbf{y}|M)P(M) \tag{1}$$

where we omitted $P(\Phi)$ and $P(\mathbf{p})$ as we consider all SDFs and poses equally likely.

Marginalising wrt. the colour models we obtain

$$P(\Phi|\Omega_p) = \prod_{\mathbf{x}\in\Omega_p} \Big\{ \sum_M P(\mathbf{x}|\Phi, M)P(M|\mathbf{y}) \Big\} \tag{2}$$

with Ω_p being the 2D image domain and

$$P(\mathbf{x}_i|\Phi, M_f) = \frac{H_e(\Phi(\mathbf{x}_i))}{\eta_f} \quad P(\mathbf{x}_i|\Phi, M_b) = \frac{1 - H_e(\Phi(\mathbf{x}_i))}{\eta_b} \tag{3}$$

where H_e denotes the smoothed Heaviside function (commonly used in level set based tracking and segmentation).

The colour posteriors are written as follows:

$$P(M_j|y) = \frac{P(\mathbf{y}|M_j)P(M_j)}{\sum_{i\in f,b} P(\mathbf{y}|M_i)P(M_i)} \qquad P(M_j) = \frac{\eta_j}{\eta} \tag{4}$$

where $j \in \{f, b\}$, η_j is the number of foreground and background pixels respectively and η is the total number of pixels in Ω_p. This choice of posteriors has been shown by [2, 13] to produce better separation between foreground and background over the standard approach of using likelihoods, which in turn leads to more accurate 3D tracking.

Switching to log probabilities, we write:

$$E = \log(P(\Phi|\Omega_p)) = \sum_{\mathbf{x}\in\Omega_p} \log(H_e(\Phi)P_f + (1 - H_e(\Phi))P_b \tag{5}$$

with

$$P_f = \frac{P(\mathbf{y}|M_f)}{\eta_f P(\mathbf{y}|M_f) + \eta_b P(\mathbf{y}|M_b)} \quad P_b = \frac{P(\mathbf{y}|M_b)}{\eta_f P(\mathbf{y}|M_f) + \eta_b P(\mathbf{y}|M_b)} \tag{6}$$

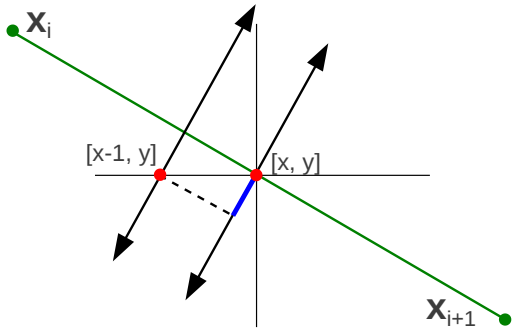

Figure 2: Geometric explanation for the computation of the derivative of the distance transform.

This energy function captures separation between foreground and background wrt. the 2D shape embedded in Φ. In our case this shape is generated as the projection of the 3D shape $\mathbf{u}$ using the pose $\mathbf{p}$. This casts the problem of maximising fg/bg separation as one of optimising E wrt. $\mathbf{p}$ using standard gradient-based methods. This requires evaluating the following derivative:

$$\frac{\partial E}{\partial \mathbf{p}} = \sum_{\mathbf{x}\in\Omega_p} \frac{\delta_e(\Phi)(P_b - P_f)}{H_e(\Phi)P_f + (1-H_e(\Phi))P_b} \left(\frac{\partial \Phi}{\partial x}\frac{\partial x}{\partial \mathbf{p}} + \frac{\partial \Phi}{\partial y}\frac{\partial y}{\partial \mathbf{p}} \right) \quad (7)$$

with δ_e the derivative of the smoothed Heaviside function and x and y the 2D coordinates of points situated on the contour of the projection of the 3D shape. The remaining derivatives $\partial\Phi/\partial\mathbf{x}$ and $\partial\Phi/\partial\mathbf{y}$ are computed numerically and $\partial x/\partial\mathbf{p}$ and $\partial y/\partial\mathbf{p}$ follow trivially as detailed in [13].

The framework presented above has been shown to produce state of the art results in region based 3D tracking [13]. This however comes at the expense of high computational cost, as the projection (i.e. rendering) of the 3D shape and the distance transform Φ of this projection have to be computed once per iteration. This means that a real time implementation is only possible using GPU processing. Even so, speeds higher than 20-25 fps are not easily achieved.

In the remaining part of this section we address the three main speed bottlenecks of this approach: **(i)** the rendering of the 3D shape, **(ii)** the computation of the SDF and its derivatives and **(iii)** the optimisation method. We also discuss the issue of silhouette ambiguity, which concerns tracking reliability instead of speed, but is especially important when doing 3D reconstruction.

Hierarchical Binary Rendering. We use a volumetric representation for the shape $\mathbf{u}$. The established method for rendering a 3D shape represented in such a way is to use a raycasting algorithm [9]. Unfortunately this operation is prohibitively slow without GPU hardware, especially on a mobile phone. Our tracker however only needs a binary rendering with depth only for the pixels located on the edge of that rendering. With this in mind, we chose to do the raycasting operation in a hierarchical manner. We initially raycast a very low resolution image (e.g. 40×30 pixels). We then resize this image by a factor of two, raycast the pixels around the edge and interpolate the others. The process is repeated multiple times until the desired resolution is reached. On a 640×480 image, this process results in a speedup of over $10\times$ over a standard CPU-based raycast and has the added benefit of producing a resolution hierarchy.

Distance Transform and Derivatives. Our pose optimisation method requires several computations of a 2D SDF for each frame. On a mobile phone, standard SDF computation algorithms take many tens of milliseconds to process a single image, so they are too slow for our purposes.

The Euclidean SDF of a contour is designed to increase linearly in the direction normal to the contour. This observation leads to our approximate SDF Φ, where, for a contour point at location $\mathbf{x}$, we increase the value of Φ linearly from a value of $-d$ at location $\mathbf{x}-d\mathbf{n}$ to a value of $+d$ at location $\mathbf{x}+d\mathbf{n}$. Here $\mathbf{n}$ is the normal to the contour at location $\mathbf{x}$, and is computed by applying a Scharr operator [17] to the raycasted binary image.

This is an approximation of the full SDF from two points of view. First, we only compute a local, per contour point SDF, in a $2d$ band around the contour. Since the informative part of the SDF is only situated close or on the actual contour points, this approximation has virtually no effect on the final pose optimisation result, as we show in Section 7. Second, the approximation might produce incorrect distance values around concavities of the contour, but again this did not adversely affect the final outcome of the pose optimisation.

We also need to compute the values of the derivatives $\partial\Phi/\partial x$ and $\partial\Phi/\partial y$. Numerically, these can be obtained with the centred finite differences approximation, using, for example:

$$\frac{\partial \Phi}{\partial x} = \frac{\Phi([x+1,y]) - \Phi([x-1,y])}{2} \quad (8)$$

In this work we obtain the values of $\Phi([x+1,y]), \Phi([x-1,y]), \Phi([x,y-1])$ and $\Phi([x,y+1])$ without explicitly evaluating Φ. The way we do this is represented geometrically in Figure 2. Given two contour points $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, the contour segment linking them is represented in green. Two example normals to this line segment are drawn in black, with arrows. These pass through the centre of the segment $[x,y]$ and the point $[x-1,y]$. The value of $\Phi([x-1,y])$ here then becomes equal to the signed distance between $[x,y]$ and the projection of $[x+1,y]$ on the normal passing through $[x,y]$. In Figure 2 this is the signed distance of the line segmented drawn in blue. The process is identical for $[x+1,y]$, $[x,y-1]$ and $[x,y+1]$.

Optimisation method. Our raycaster produces a hierarchy of object renderings. We use this to speed up our tracker, replacing costly high resolution iterations with cheaper low resolution ones, resulting in a 2-3$\times$ speedup. We use the Levenberg-Marquardt (LM) algorithm to minimise our energy function at each hierarchy level.

Silhouette Ambiguity. The mapping from silhouette to pose is ambiguous, as 3D rigid objects often project to the virtually identical silhouettes, in spite of being under different poses. We experimentally investigate the effect of this ambiguity on tracking in Figure 5, showing that silhouette-only tracking and reconstruction is effectively impossible. Inspired by [14], we use the IMU readily available on the mobile phone to disambiguate rotation.

The relation between the two pose estimations is depicted in Figure 3. $R_p^{(t-1)}$ and $R_p^{(t)}$ are the rotation matrices of the object in the camera coordinate system, at the previous frame and current frame, respectively. Similarly, $R_a^{(t-1)}$ and $R_a^{(t)}$ are consecutive rotation matrices of the camera in the phone coordinate system. Finally, C is the calibration rotation matrix, converting the visual to the IMU coordinate systems and is preset for each type of device. Therefore:

$$R_p^{(t)} = C R_a^{(t)} \left(R_a^{(t-1)} \right)^{-1} C^{-1} R_p^{(t-1)} \quad (9)$$

Between consecutive frames we only optimise for translation, using as rotation estimate the change given by the IMU. To compensate for IMU drift, we use one gradient descent rotation-wise iteration every ten frames. We do not use LM for rotation as due to ambiguity we only trust the visual rotation estimate to correct for slight drift, not to fully dictate the pose.

5 Shape Optimisation

The shape optimisation assumes known pose and per pixel foreground or background likelihoods for each of the n views. These are back-projected into a pair of 3D likelihood volumes, capturing the probability that a voxel $\mathbf{v}$ belongs to the inside and outside of the

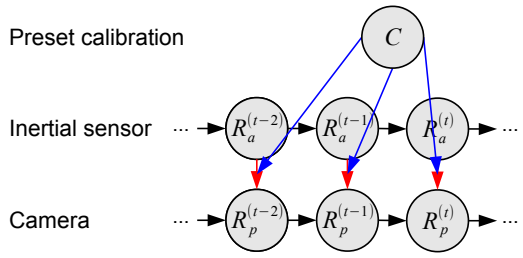

Figure 3: IMU integration.

shape, respectively. The likelihoods are next turned into posteriors, in a manner similar to the one presented in the previous section. Finally, the 3D shape $\mathbf{u}$ is extracted from the two posterior volumes, looking to maximise inside/outside separation. This framework is similar to the established one of [8, 5], but here we use voxel posteriors instead of likelihoods and account for the online accumulation of views. Note that an alternative approach would have been to fuse individual per-view segmentations (obtained using e.g. per-view graph-cuts) instead of probabilities. This approach has been shown in [8] to produce inferior results, because **(i)** individual segmentations often tend to be poor (because of e.g. shadows and reflections) and **(ii)** the camera pose is not perfectly known (so silhouette uncertainly has to be accounted for).

Considering $\mathbf{x}$ and Φ as known in the graphical model, the joint probability becomes:

$$P(\mathbf{u},\mathbf{v},M_{1\ldots n},\mathbf{y}_{1\ldots n}) = P(\mathbf{v}|\mathbf{u},M_{1\ldots n})P(\mathbf{y}_{1\ldots n}|M_{1\ldots n})P(M_{1\ldots n}) \quad (10)$$

Expanding, we write:

$$P(\mathbf{u}|\Omega_s) = \prod_{\mathbf{v}\in\Omega_s}\Big\{\sum_{j\in f,b} P(\mathbf{v}|\mathbf{u},M_{j,1\ldots n})P(M_{j,1\ldots n}|\mathbf{y}_{1\ldots n})\Big\} \quad (11)$$

where Ω_s is the 3D domain of voxels $\mathbf{v}$ and:

$$P(M_{j,1\ldots n}|\mathbf{y}_{1\ldots n}) = \frac{P(\mathbf{y}_{1\ldots n}|M_{j,1\ldots n})P(M_{j,1\ldots n})}{\sum_{i\in f,b}P(\mathbf{y}_{1\ldots n}|M_{i,1\ldots n})P(M_{i,1\ldots n})} \quad (12)$$

$$P(\mathbf{v}|\mathbf{u},M_{f,1\ldots n}) = \frac{\mathbf{u}}{\zeta_f} \qquad P(\mathbf{v}|\mathbf{u},M_{b,1\ldots n}) = \frac{1-\mathbf{u}}{\zeta_b} \quad (13)$$

with ζ_f and ζ_b being the average number of voxels (over all views n) that project to a foreground pixel (with $P(\mathbf{y}|M_f) > P(\mathbf{y}|M_b)$) and a background pixel, respectively.

The final two probabilities $P(\mathbf{y}_{1\ldots n}|M_{1\ldots n})$ and $P(M_{1\ldots n})$ should be computed as the joint probabilities of $P(\mathbf{y_k}|M_k)$ and $P(M_k)$, with $k \in 1\ldots n$. As observed by [8], this joint probability cannot easily be computed numerically, as it tends to be a product of very small numbers. The authors in [8] used the geometric mean to obtain average probabilities. We proceed in a similar manner, and write:

$$P(\mathbf{y}_{1\ldots n}|M_{f,1\ldots n}) = \exp\left(\frac{\sum_k^n \log(P(\mathbf{y}_k|M_{f,k}))}{n}\right) \quad (14)$$

$$P(\mathbf{y}_{1\ldots n}|M_{b,1\ldots n}) = 1-\exp\left(\frac{\sum_k^n \log(1-P(\mathbf{y}_k|M_{b,k}))}{n}\right) \quad (15)$$

The probability of the colour model over n views becomes $P(M_{f,1\ldots n}) = \eta_{fm}/\eta$ with η_{fm} being the average value of η_f (as defined in Equation (4)) over the n views. $P(M_{b,1\ldots n})$ follows analogously.

The final expansion of $P(\mathbf{u}|\Omega_s)$ becomes:

$$E = P(\mathbf{u}|\Omega_s) = \prod_{\mathbf{v}\in\Omega_s}\{\mathbf{u}P_i + (1-\mathbf{u})P_o\} \quad (16)$$

$$P_i = \frac{\eta_{fm}}{\zeta_f}\frac{P(\mathbf{y}_{1\ldots n}|M_{f,1\ldots n})}{P(\mathbf{y}_{1\ldots n}|M_{f,1\ldots n})\eta_{fm} + P(\mathbf{y}_{1\ldots n}|M_{b,1\ldots n})\eta_{bm}} \quad (17)$$

$$P_o = \frac{\eta_{bm}}{\zeta_b}\frac{P(\mathbf{y}_{1\ldots n}|M_{b,1\ldots n})}{P(\mathbf{y}_{1\ldots n}|M_{f,1\ldots n})\eta_{fm} + P(\mathbf{y}_{1\ldots n}|M_{b,1\ldots n})\eta_{bm}} \quad (18)$$

One way to optimise Equation (16) is to consider $\mathbf{u}$ as the smoothed Heaviside of a 3D level set embedding function. This method is subjected to local minima and requires the SDF structure to be maintained. When $\mathbf{u}$ is a probabilistic 3D volume however, [10] shows that $\mathbf{u}$ can be solved for globally using convex optimisation.

The globally solvable formulation for our reconstruction is obtained by replacing the logarithmic option pool with a linear one in Equation (11) and adding a weighted length regularization term:

$$E = \sum_{\mathbf{v}\in\Omega_s}\{\mathbf{u}P_i + (1-\mathbf{u})P_o + \alpha|\nabla\mathbf{u}|\} \quad (19)$$

where α is a tunable parameter.

To minimise such an energy function the authors in [8] use the primal-dual algorithm of [12]. We use the continuous min-cut / max-flow formulation of [23], which leads to considerably faster convergence compared to [12]. In a max flow context, minimising Equation (16) wrt. $\mathbf{u}$ such that $\mathbf{u} \in [0,1]$ is equivalent to minimising:

$$E = \max_{p_t,p_s,p}\min_{\mathbf{u}}\sum_{\mathbf{v}}\{\mathbf{u}p_t + (1-\mathbf{u})p_s + \mathbf{u}\,\mathrm{div}p\} \quad (20)$$

such that $p_s(\mathbf{v}) < P_i(\mathbf{v})$, $p_t(\mathbf{v}) < P_o(\mathbf{v})$ and $|p(\mathbf{v})| < \alpha$. In the context of max-flow, p, p_s and p_t are flow capacities, for undirected edges for p, between nodes and source for p_s and between nodes and sink for p_t [23].

The continuous max-flow algorithm of [23] is iterative and uses the augmented Lagrangian function of Equation (20), defined as:

$$L_c(p_s,p_t,p,\mathbf{u}) = \sum_{\mathbf{v}}\{\mathbf{u}p_t + (1-\mathbf{u})p_s + \mathbf{u}\,\mathrm{div}p\} - \tfrac{c}{2}||\,\mathrm{div}p - p_s + p_t||^2 \quad (21)$$

where c is a constant step size. This Lagrangian is used to minimise Equation (20) wrt. p, p_s and p_t in turn. For the k-th iteration the algorithm the authors in [23] write:

$$p^{(k+1)} = \underset{||p||_\infty\le\alpha}{\arg\max}\, L_c(p_s^k, p_t^k, p, \mathbf{u}^k) \quad (22)$$

$$p_s^{(k+1)} = \underset{p_s(\mathbf{v})<P_i(\mathbf{v})}{\arg\max}\, L_c(p_s, p_t^k, p^k, \mathbf{u}^k) \quad (23)$$

$$p_t^{(k+1)} = \underset{p_t(\mathbf{v})<P_o(\mathbf{v})}{\arg\max}\, L_c(p_s^k, p_t, p^k, \mathbf{u}^k) \quad (24)$$

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - c(\mathrm{div}\; p^{(k+1)} - p_s^{(k+1)} + p_t^{(k+1)}) \quad (25)$$

The optimisation is started by setting $\mathbf{u} = P_i - P_o$ where $P_i > P_o$ and zero otherwise, and $p_s = p_t = \min(P_i, P_o)$.

So far we have assumed that all n frames are available simultaneously. This is of course not true, as we run our algorithm online, and it implies three changes on the reconstruction methodology.

First, the likelihood volumes $P(\mathbf{y}_{1\ldots n}|M_{1\ldots n})$ and the values of $\zeta_f, \zeta_b, \eta_{fm}$ and η_{bm} change from being accumulated only once for all n views, to being updated online after every new frame.

Second, after running the max flow algorithm, we rescale the non-zero region of interest in the 3D volume $\mathbf{u}$ to fill the volume,

with a constant padding. This mitigates the effects of scale drift and helps us use the full representational power of the discretisation.

Third, instead of running a single full convergence of the continuous max flow optimisation after all n views have been registered, we run a single iteration for every ten views that have been registered. We run only one iteration for real-time considerations and as the final estimate will change with update input data anyway. This leads to another problem, namely how to transfer the intermediary shape results from one reconstruction to the next. This is not straightforward, as, especially in the early stages of the reconstruction process, large portions of the estimated 3D shape might change and the intermediary values for $\mathbf{u}$ are far from the globally optimal shape embedded by the updated posterior volumes.

To alleviate this problem we propagate the intermediary values of the 3D shape by conditioning $P(\mathbf{v}|\mathbf{u}, M_{f,1\ldots n})$ and $P(\mathbf{v}|\mathbf{u}, M_{b,1\ldots n})$ on the previous 3D reconstruction $\mathbf{u}^{(t-1)}$. Intuitively, this means that the probabilities of a voxel being inside or outside the 3D shape should increase if the voxel was inside or outside, respectively, at the previously available reconstruction. Formally this changes:

$$P(\mathbf{v}|\mathbf{u}, M_{f,1\ldots n}) \text{ to } P(\mathbf{v}|\mathbf{u}, \mathbf{u}_i^{(t-1)}, M_{f,1\ldots n}) = \frac{\mathbf{u}\mathbf{u}_f^{(t-1)}}{\zeta_f} \quad (26)$$

$$P(\mathbf{v}|\mathbf{u}, M_{b,1\ldots n}) \text{ to } P(\mathbf{v}|\mathbf{u}, \mathbf{u}_o^{(t-1)}, M_{b,1\ldots n}) = \frac{(1-\mathbf{u})\mathbf{u}_b^{(t-1)}}{\zeta_b} \quad (27)$$

with:

$$\mathbf{u}_i^{(t-1)} = \beta - 1 + \beta u^{(t-1)} \quad \mathbf{u}_o^{(t-1)} = \beta - 1 + \beta(1 - u^{(t-1)}) \quad (28)$$

where β is a tunable parameter (0.5 in our implementation).

The change in voxel probabilities also leads to slightly different formulas P_i and P_o, which are trivial to compute.

6 Implementation and Timings

The previous sections presented our theoretical framework. In this section we cover important aspects of a practical implementation capable of running in real time on a mobile phone.

Algorithm 1 Image acquisition + 3D tracking thread

1: read and fuse the IMU data. This is done using CoreMotion provided by the Apple iOS SDK.
2: build the interpolated colour models M_f and M_b.
3: compute tracking image statistics: P_f and P_b
4: **if** thread 2 reconstruction is complete **then**
5: copy reconstructed shape to tracking thread memory.
6: **end if**
7: track object in new frame using current shape approximation.
8: **if** new frame is keyframe and reconstruction processing has finished **then**
9: transfer image statistics to reconstruction thread
10: start reconstruction thread
11: **end if**

Inspired by [7], we decouple tracking from reconstruction by using two threads: one for image acquisition and 3D tracking and the other one for reconstruction.

The camera and tracking thread proceeds as described in Algorithm 1. As discussed in Section 4, we use the Levenberg-Marquardt algorithm for tracking. The method used to compute the derivative wrt. pose is outlined in Algorithm 2.

The reconstruction thread proceeds as described in Algorithm 3. Similar to the map update stage from [7], we only register keyframes into the likelihood/posterior volumes. A frame is considered to be a keyframe if **(i)** at least ten frames have passed from the previous keyframe and **(ii)** the rotation read from the IMU has not been observed by more than five times. We allow each rotation multiple times to accommodate updates due to improved pose estimates. Alternatively, we could have used the translation estimate in the keyframe selection and allowed each keyframe to be registered only once, but, experimentally, we found our solution to lead to better results. Also, once a critical mass of frames has been obtained (in our experiments 1500 frames), novel frames do not improve shape accuracy, and we can re-track and re-register previous ones.

Algorithm 2 Pose derivative evaluation

1: project current approximation of the 3D shape with current approximation of the pose, using our hierarchical raycasting method.
2: find contour points using Scharr filtering.
3: **for** each contour point, with location $[x, y]$ **do**
4: compute SDF derivatives $\frac{\partial \Phi}{\partial x}$ and $\frac{\partial \Phi}{\partial y}$
5: compute derivatives wrt. pose $\frac{\partial x}{\partial \mathbf{p}}$ and $\frac{\partial y}{\partial \mathbf{p}}$
6: **end for**
7: compute and sum per-point derivatives of the energy function from Equation (7).

Algorithm 3 Reconstruction thread

1: update $P(\mathbf{y}_{1\ldots n}|M_{f,1\ldots n})$ and $P(\mathbf{y}_{1\ldots n}|M_{f,1\ldots n})$ using the image statistics from the new frame
2: **if** has enough new registered keyframes **then**
3: compute reconstruction statistics P_i and P_b
4: find a new approximation for the 3D shape $\mathbf{u}$ using continuous max flow.
5: rescale shape to fill 3D volume.
6: **end if**

One other important implementation detail is the initialisation of the colour models M_f and M_b. Here we require the user to move around the object at a few positions (5 to 10) and initialise the object and background statistics. Automatic method for model initialisation, such as used in [4] could also be employed, but this is beyond the scope of the current publication. On the mobile phone the user can do this very easily by keeping the object centred and adjusting a slider which changes the threshold value of a binarisation algorithm. On the desktop we manually generate binary foreground / background masks. For each labelled sample image we also know the IMU rotation, which allows us to produce at run time a per-view pose-dependant pair of colour models, using linear interpolation weighted by distance in rotation space. The final colour models are built by composing the pose-dependant colour model with an online adapted component, using linear interpolation.

Example timings for our method are shown in Table 1, obtained for the sequence shown in Figure 10. We run the tracker at every frame which works at 107.4 fps on a desktop PC (Intel Core i7-3960X 3.3GHz CPU) and 23.04 fps on an iPhone 5. Performance degrades roughly 25% on a lower end iPhone 4S. This makes reconstruction not feasible on lower end devices, but tracking still possible. For tracking we use a $128 \times 128 \times 128$ volume on the desktop PC and $64 \times 64 \times 64$ volume on the mobile phone. The reconstruction stages run simultaneously with tracking and are executed at most every 10th and every 100th frame, respectively. Both desktop PC and iPhone use $128 \times 128 \times 128$ volumes for reconstruction.

7 Results

We have tested our method quantitatively and qualitatively, both on artificial and real data, in static and dynamic environments.

Quantitative Testing. We begin in Figure 4 with a quantitative

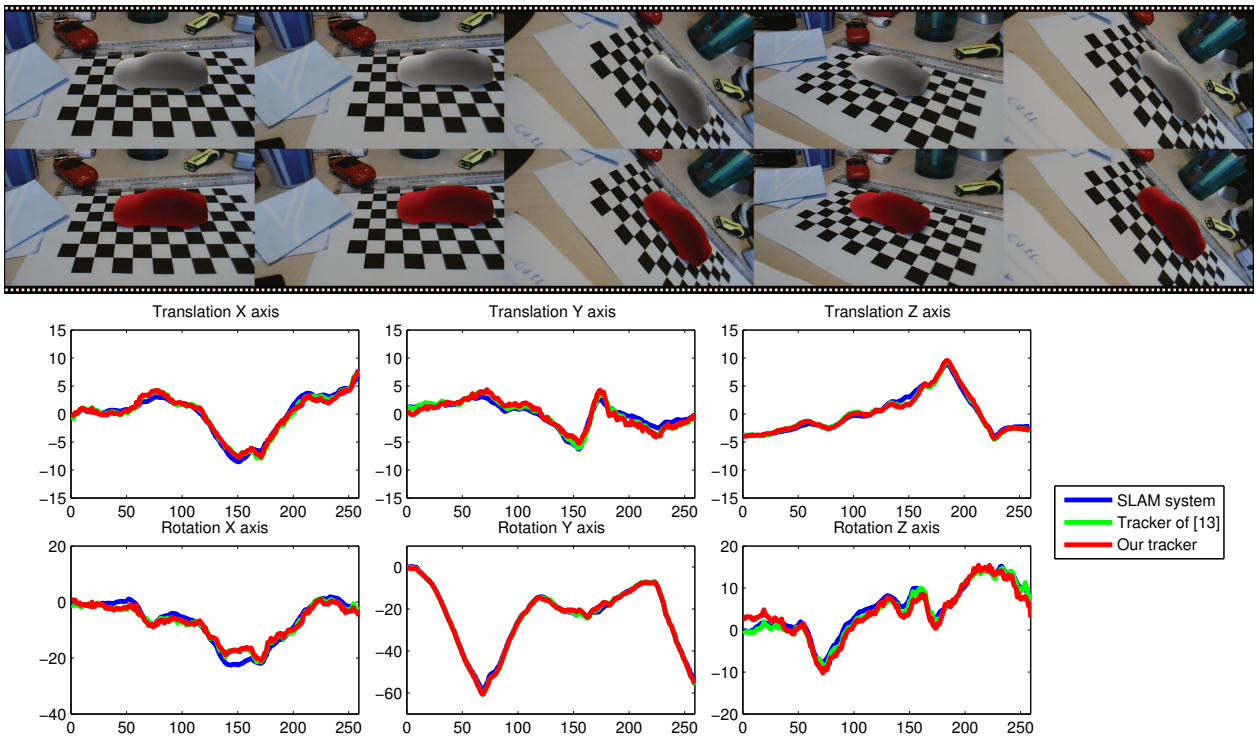

Figure 4: Comparison between the camera pose recovered by the SLAM system of [7], our tracker and the tracker of [15]. The filmstrip shows frames from the experiment, with the top row showing our results and the bottom row the results from [15].

Table 1: Per frame example timings for our method.

	Tracking	Frame registration	Max flow
Desktop PC	9.3ms	18ms	102ms
iPhone 5	43.3ms	198ms	667ms

comparison between the PTAM system [7], our tracker and the one presented in [15], which minimises the same energy function in a fully analytical manner, without any approximations. The results produced by the object trackers are nearly identical. Compared to the SLAM system, [15] produced a difference of 0.96 cm and 1.98^o. Our tracker has an average difference of 1.13 cm and 2.52^o. The improvement of 0.17 cm and 0.54^o in [15] comes however as the expense of twice the processing time and requiring a powerful GPU.

Figure 5 shows a quantitative comparison between our tracker with and without using the IMU and PTAM [7]. Without the IMU, the average difference between tracker and SLAM is 12.86 cm in translation and 11.97^o in rotation. With the IMU the difference decreases to an average of 1.14cm in translation and 1.27^o in rotation. The very large difference incurred when the IMU is not available is due to silhouette ambiguity, as shown in columns 2, 4 and 6 of Figure 5(top). The silhouette provides too little information for an accurate rotation estimation and the optimisation converges to a incorrect local minima. When aided by the IMU, rotation is no longer ambiguous and the overall tracking errors become much smaller.

Note that in both of the above tests, the difference between trackers and SLAM is caused by one using information from only around the projected contour, whereas the other using the whole image. Furthermore, while the PTAM system is more likely to be closer to the ground truth, it is not guaranteed to be the *actual* ground truth.

Figures 6 and 7 show quantitative comparisons between the reconstruction obtained using the posterior voxel probabilities presented in this paper and the likelihoods used in [8], which we consider as current state-of-the-art. Here we generated artificial ground truth data using two 3D models, the upper body shown in Figure 6 and the car shown in Figure 7. The two generated sequences showed the respective object rotated on each axis between -180^o and 180^o and translated by a random amount. We ran the reconstruction algorithm using the two types of voxel statistics with four configurations: **(i)** ground truth histograms and known pose (Figs. 6 and 7, left chart); **(ii)** ground truth histograms and rotation and optimisation for translation (Figs. 6 and 7, left chart); **(iii)** noisy histograms and known pose (Figs. 6 and 7, right chart) and **(iv)** noisy histograms, known rotation + optimisation for translation (Figs. 6 and 7, right chart).

For each figure, the red contour in the lower left figures shows the area of foreground that was added to the estimation of the background histogram in order to corrupt it. The matching score plotted in the four charts is the standard intersection vs reunion measure of overlap [6], evaluated between ground truth and reconstructed volume. Note that we do not run any shape alignment since all versions of the algorithm should produced aligned shapes.

When the histograms are not noisy all four methods produce similar results, with the best results being obtained when using posteriors and ground truth poses. Likelihoods favour thicker reconstructions while posteriors favour thinner ones. This is particularly visible when using estimated translation in Figure 6. Here the posteriors lead to a decrease in accuracy around the hands, whereas likelihoods have same effect around the upper arms. Overall however all four methods lead to the roughly similar errors. The differences between posteriors and likelihoods become much more pronounced when the histograms are imperfect, such as would often be the case in real world usage. Likelihoods lead to much worse results, up to the point where using posteriors and estimating translation leads to better results than using likelihoods with ground truth pose.

Qualitative Testing. The remaining five figures (8, 9, 10, 11 and 12) show qualitative reconstruction results obtained on real

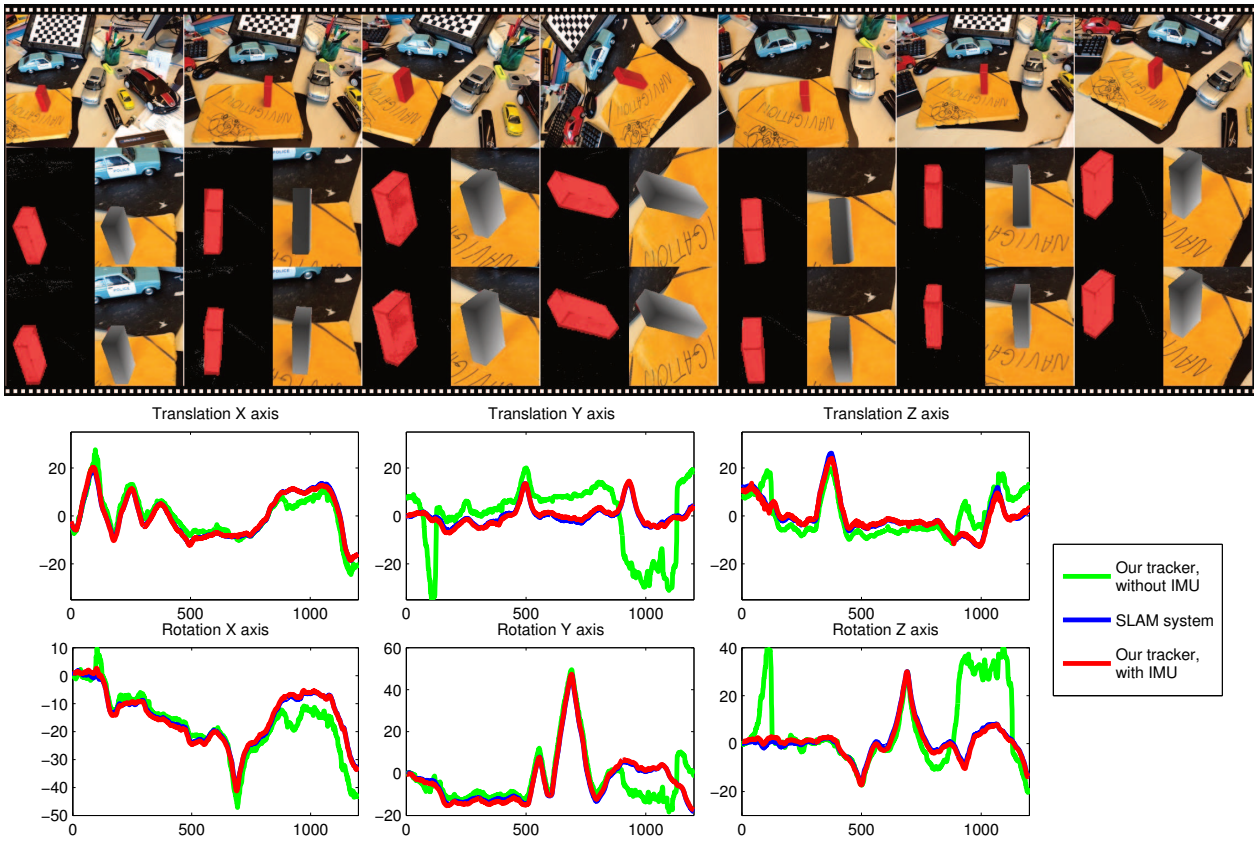

Figure 5: Comparison between the SLAM system of [7] and our tracker with and without the IMU. The filmstrip shows frames from the experiment with row 1 showing the original image, row 2 the result obtained without the IMU and row 3 the results obtained with the IMU.

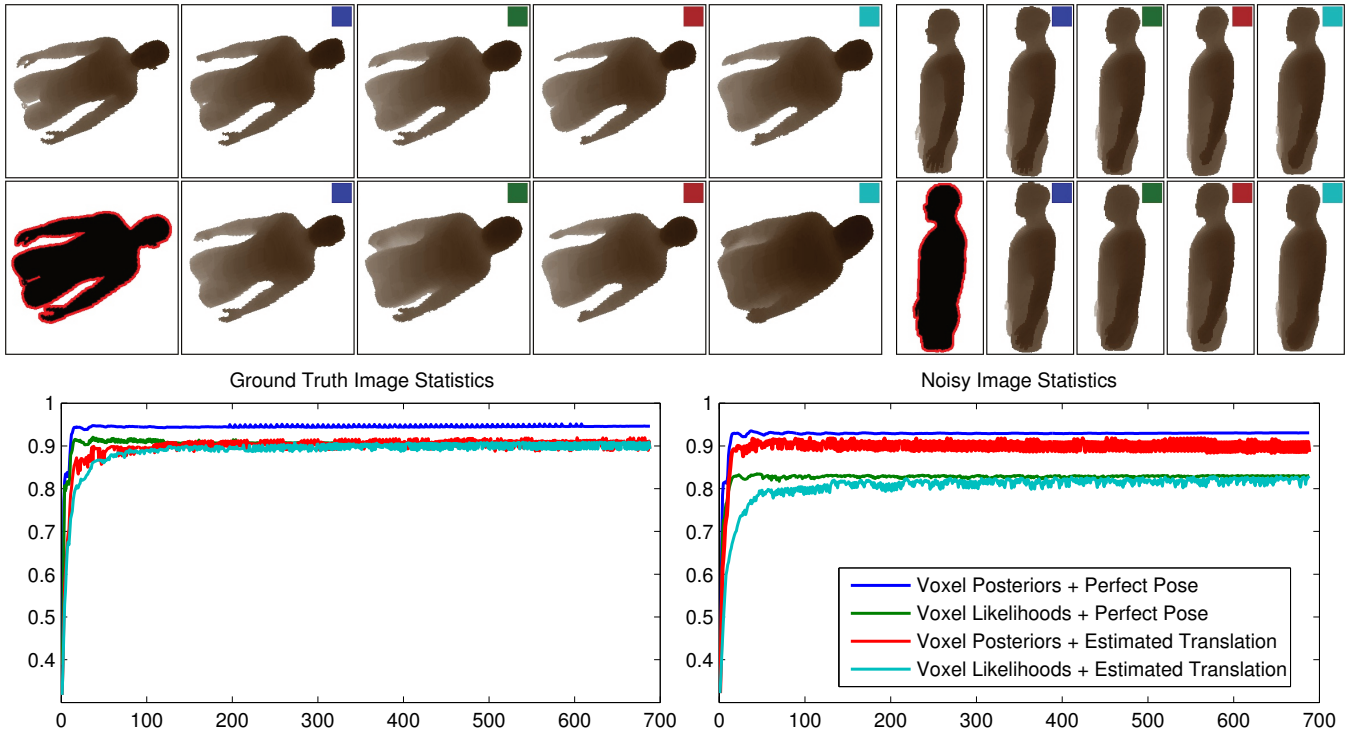

Figure 6: Reconstruction comparison between our system and that of [8]. The top 2 rows show views of the 3D reconstructed shape, with the lower left figures showing in red the foreground area that was added to the background histogram to corrupt it. Results obtained with ground truth histograms appear in the left chart and top row and results obtained with imperfect histograms in the right chart and bottom row.

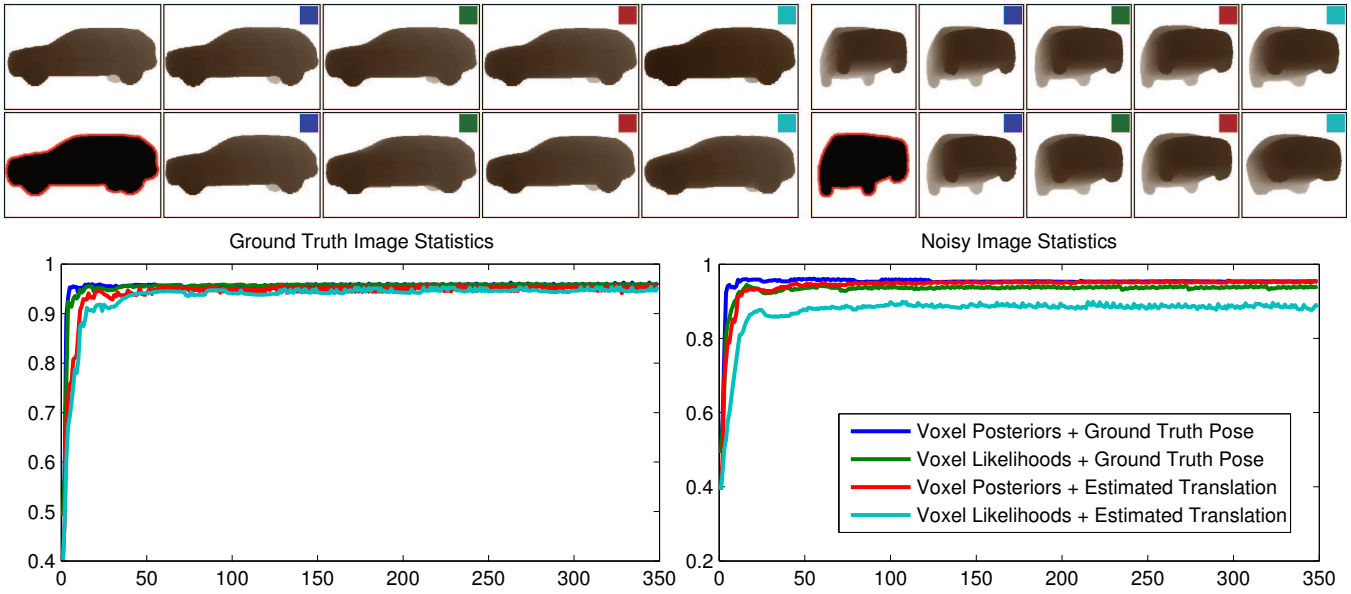

Figure 7: Reconstruction comparison between our system and that of [8]. The top 2 rows show views of the 3D reconstructed shape, with the lower left figures showing in red the foreground area that was added to the background histogram to corrupt it. Results obtained with ground truth histograms appear in the left chart and top row and results obtained with imperfect histograms in the right chart and bottom row.

data. In each figure, the first three columns of the film strip show the beginning of the reconstruction process and are taken within the first 50 processed keyframes. The next two columns show results obtained around the 300-400 keyframe mark. The last column shows the final result obtained after a maximum of 1000 keyframes. Figures 8 and 9 show our reconstruction working in a museum, in a fully unstructured and dynamic environment, corrupted by occlusions, reflections and imperfect illumination. Our method obtains good results, with artefacts appearing only around the head of each figure. These are caused by specular reflections, which changed the colour of the object from red and black respectively to white. To our knowledge, no other object reconstruction method could process these videos because of the large number of image imperfections. Figures 10, 11 and 12 show reconstruction results obtained in a slightly better lit office environment. In all three cases we can obtain very good reconstructions, with the shape usually converging with 300 to 400 keyframes (i.e. 3-4 minutes of data at an average of 2-3 keyframes per second). The one exception is the horse shown in Figure 12 which requires up to 1000 keyframe to recover the legs.

Failure Cases. The long processing time required to recover the legs of the horse in Figure 12 and the loss of the hand in Figure 6 highlight the main failure case of our algorithm, namely its brittleness when dealing with thin structures. These occupy a small area in the image which might not be enough to "pull" the 3D tracker into the right direction. This leads to an inaccurate translation recovery, which in turn diminishes the amount of backprojected view overlap around the thin parts of the shape, causing low voxel probability around the same areas and thus slow convergence or an incorrect reconstruction. Another possible source of failure in our system is rotation drift. Limited amounts of drift are compensated for using the visual rotation optimisation, but large amounts of drift result in decreased tracking and subsequently rotation accuracy.

8 Conclusions

In this paper we proposed a novel framework for simultaneous 3D reconstruction and tracking. Tracking is cast as a minimisation in 3D pose space of a region based level set energy function, which measures separation between image foreground and background. Unlike other approaches, we use a local instead of a global computation of the signed distance transform and its derivatives and obtain fast 3D shape renderings using hierarchical raycasting. This enables our tracker to achieve real time performance on a mobile phone and speeds of over 100 fps on a desktop PC, without using GPU acceleration. We currently require a high-end mobile phone (an iPhone 5), but this is a temporary limitation that will disappear over time. Rotation is disambiguated using the IMU available on mobile phones. Reconstruction is done in parallel with tracking, using continuous max flow, and maximising separation between probabilistically defined inside and outside regions. We use posterior voxel probabilities, which lead to increased robustness to imperfect image data over the standard approach of using likelihoods.

The main direction for future research is addressing the decrease in reconstruction quality caused by thin structures. This could be mitigated for example, by either improving the tracking results or by marginalising the pose error. Another interesting avenue of research is the additional use of texture information along with histograms. This could further improve tracking accuracy.

Acknowledgments We gratefully acknowledge the support of EPSRC EP/H050795/1 and EU FP7 287713 REWIRE.

References

[1] J. Bastian, B. Ward, R. Hill, A. van den Hengel, and A. Dick. Interactive modelling for AR applications. In *ISMAR 2010*, pages 199–205.

[2] C. Bibby and I. Reid. Robust Real-Time Visual Tracking Using Pixel-Wise Posteriors. In *ECCV 2008*, pages 831–844.

[3] G. Bleser, C. Wohlleber, M. Becker, and D. Stricker. Fast and stable tracking for ar fusing video and inertial sensor data. In *WSCG 2006*, pages 109–115.

[4] N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Automatic object segmentation from calibrated images. In *CVMP 2011*, pages 126–137.

[5] N. D. F. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Automatic 3D object segmentation in multiple views using volumetric graph-cuts. *ImaVis 2010*, 28(1):14–25.

[6] M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV 2010*, 88(2):303–338.

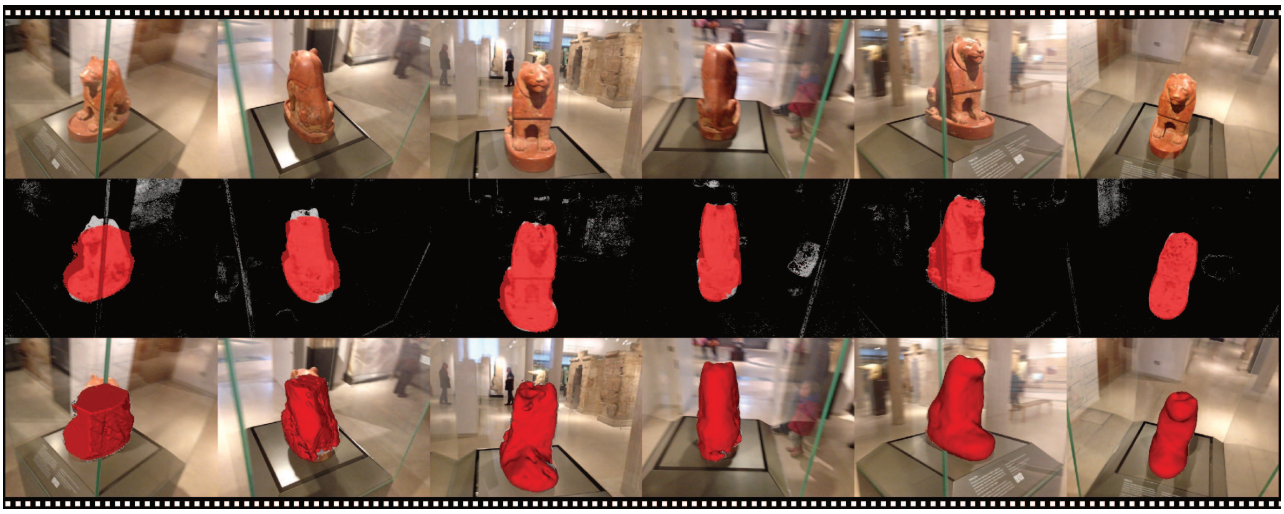

Figure 8: Lion tracking and reconstruction example.

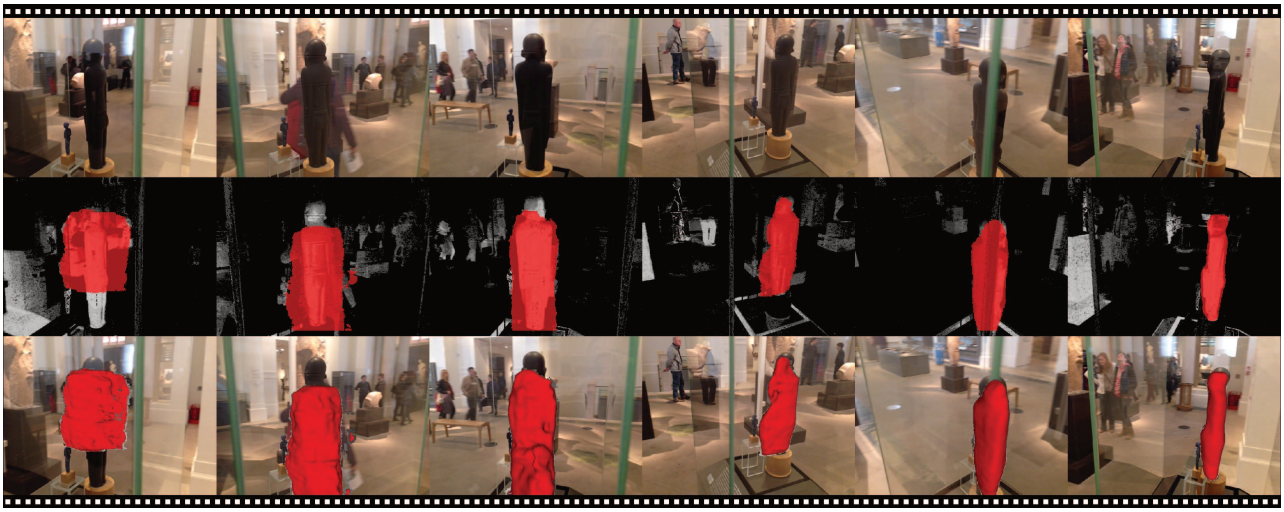

Figure 9: Idol tracking and reconstruction example.

Figure 10: Shoe tracking and reconstruction example.

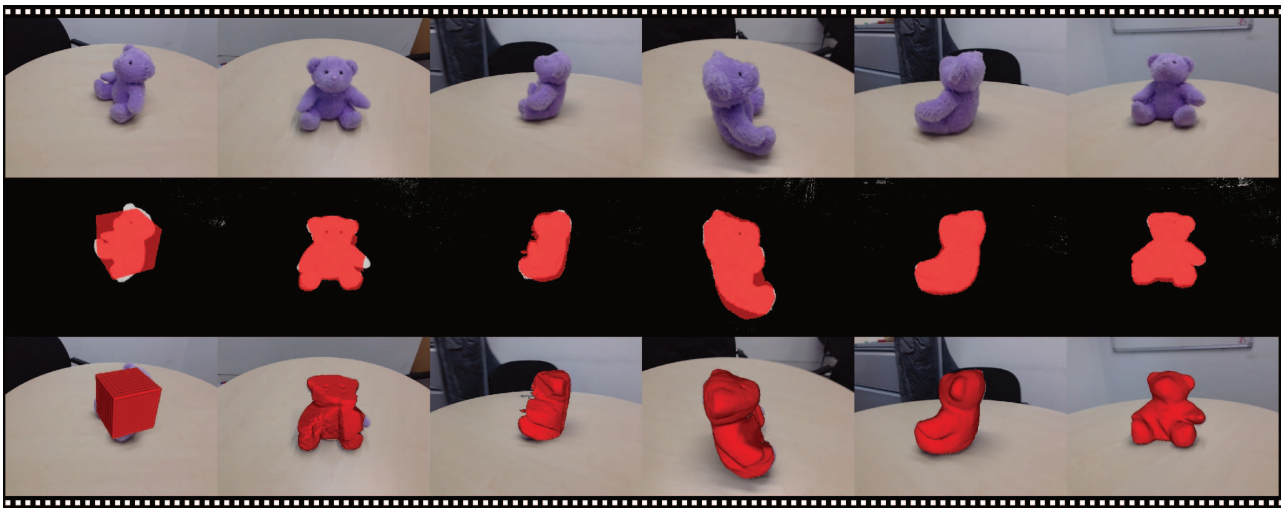

Figure 11: Teddy bear tracking and reconstruction example.

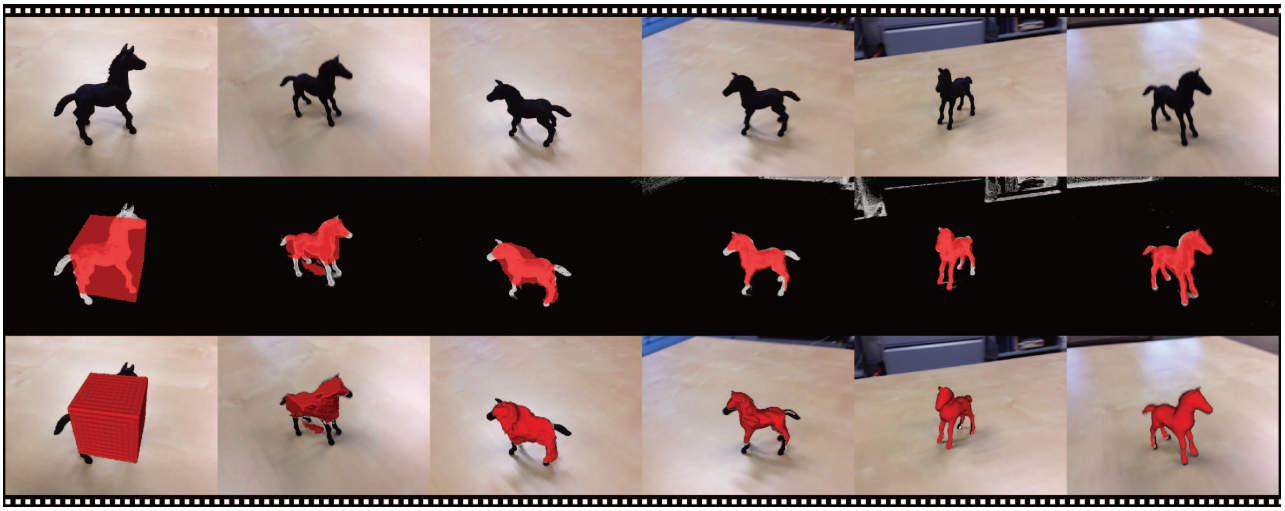

Figure 12: Toy horse tracking and reconstruction example.

[7] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR 2007*, pages 1–10.

[8] K. Kolev, T. Brox, and D. Cremers. Fast Joint Estimation of Silhouettes and Dense 3D Geometry from Multiple Images. *IEEE Trans. on PAMI 2012*, 34(3):493–505.

[9] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV 2011*, pages 2320–2327.

[10] M. Nikolova, S. Esedoglu, and T. Chan. Algorithms for Finding Global Minimizers of Image Segmentation and Denoising Models. *SIAM-JAM 2006*, 66(5):1632–1648.

[11] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *BMVC 2009*, pages 112.1–112.11.

[12] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. An Algorithm for Minimizing the Piecewise Smooth Mumford-Shah Functional. In *ICCV 2009*, pages 1133–1140.

[13] V. A. Prisacariu and I. Reid. PWP3D: Real-Time Segmentation and Tracking of 3D Objects. *IJCV 2012*, pages 1–20.

[14] V. A. Prisacariu and I. Reid. Robust 3D hand tracking for human computer interaction. In *FG 2011*, pages 368–375.

[15] V. A. Prisacariu, A. Segal, and I. Reid. Simultaneous Monocular 2D Segmentation, 3D Pose Recovery and 3D Reconstruction. In *ACCV 2012*, pages 593–606.

[16] B. Rosenhahn, T. Brox, and J. Weickert. Three-Dimensional Shape Knowledge for Joint Image Segmentation and Pose Tracking. *IJCV 2007*, 73(3):243–262.

[17] H. Scharr. *Optimal Operators in Digital Image Processing*. PhD thesis, Heidelberg University, 2000.

[18] C. Schmaltz, B. Rosenhahn, T. Brox, J. Weickert, D. Cremers, L. Wietzke, and G. Sommer. Occlusion Modelling by Tracking Multiple Objects. In *DAGM 2007*, pages 173–183.

[19] D. Snow, P. Viola, and R. Zabih. Exact Voxel Occupancy with Graph Cuts. In *CVPR 2000*, pages 345–352.

[20] L. Vese and T. Chan. A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model. *IJCV 2002*, 50(3):271–293.

[21] A. Yezzi and S. Soatto. Stereoscopic Segmentation. *IJCV 2003*, 53(1):31–43.

[22] S. You and U. Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *VR 2001*, pages 71–78.

[23] J. Yuan, E. Bae, and X.-C. Tai. A study on continuous max-flow and min-cut approaches. In *CVPR 2010*, pages 2217–2224.