Dense Reconstruction Using 3D Object Shape Priors

Amaury Dame, Victor A. Prisacariu, Carl Y. Ren
University of Oxford
{adame,victor,carl}@robots.ox.ac.uk

Ian Reid
University of Adelaide
ian.reid@adelaide.edu.au

Abstract

We propose a formulation of monocular SLAM which combines live dense reconstruction with shape priors-based 3D tracking and reconstruction. Current live dense SLAM approaches are limited to the reconstruction of visible surfaces. Moreover, most of them are based on the minimisation of a photo-consistency error, which usually makes them sensitive to specularities. In the 3D pose recovery literature, problems caused by imperfect and ambiguous image information have been dealt with by using prior shape knowledge. At the same time, the success of depth sensors has shown that combining joint image and depth information drastically increases the robustness of the classical monocular 3D tracking and 3D reconstruction approaches.

In this work we link dense SLAM to 3D object pose and shape recovery. More specifically, we automatically augment our SLAM system with object specific identity, together with 6D pose and additional shape degrees of freedom for the object(s) of known class in the scene, combining image data and *depth information for the pose and shape recovery. This leads to a system that allows for full scaled 3D reconstruction with the known object(s) segmented from the scene. The segmentation enhances the clarity, accuracy and completeness of the maps built by the dense SLAM system, while the dense 3D data aids the segmentation process, yielding faster and more reliable convergence than when using 2D image data alone.*

1. Introduction

The reconstruction of scene geometry from a single monocular image sequence is a key problem in computer vision. When the camera trajectory is unknown, the joint on-line estimation problem for scene structure and camera pose has become known as visual Simultaneous Localisation and Mapping. Early methods for visual SLAM [7, 11] concentrated on accurate camera pose estimation using only sparse reconstructions. These have various disadvantages, such as their inability to provide occlusion information.

More recently, with the introduction of new GPU-based computation devices, real-time, dense SLAM has become a technical possibility [10, 15]. These dense approaches use the raw image information (such as colour or gradients) to estimate the scene geometry, leveraging weak priors such as scene smoothness in the absence of image texture, and relying on colour constancy as the camera moves; this works for lambertian scenes, but fails in the presence of specularities, when artefacts tend to appear in the reconstruction.

Humans deal with these issues via the use of high level semantic information. In an incremental move towards such high-level representations, works such as [4] or [1] use known objects as features in the SLAM system. While this has the effect of increasing SLAM robustness and accuracy, both approaches are limited by their use of a sparse map and by the fact that they consider the objects to be of fixed and perfectly known shape. A more generic semantic reconstruction is proposed in [8], where shape and layout priors of buildings are learned offline. Once several plans of a building are found in several images, the model that best matches the priors and images is estimated. While this method allows for a semantic understanding of the scene and an estimation of what is not visible, its formulation restricts it to (mainly) planar scenes.

While shape priors have seen limited use in the SLAM literature, they have been extensively used in segmentation and tracking, as a solution to the problem of imperfect raw image information. One of the most effective and popular approaches to represent shape knowledge is to use dimensionality reduction to capture the shape variance as low dimensional latent shape spaces. Initial works, such as [24], focused on (implicitly or explicitly defined) 2D shapes, and used linear dimensionality reduction in the form of principal component analysis (PCA). More recent works use non-linear dimensionality reduction such as Kernel PCA in [6] and Gaussian Process Latent Variable Models (GP-LVM) in [17]. This led to 3D shape priors being first introduced in [23]. Most recently, [19] learn GP-LVM latent spaces of 3D shapes and use them in monocular simultaneous 2D segmentation, 3D reconstruction and 3D pose recovery.

Our objective in this paper is to address these limitations of existing systems by proposing an efficient dense SLAM

1063-6919/13 $26.00 © 2013 Crown Copyright
DOI 10.1109/CVPR.2013.170

approach that integrates a shape-prior-based estimator as-and-when possible. Initially, photo-consistency is used to built a dense representation of the scene as in [15], but once an object of known type is detected (automatically, using a standard 2D sliding-window object-class detector) the input images and resulting depth map from the SLAM are used in a novel energy function minimisation to find the 6D pose and shape of the object. Here, in a manner similar to [19], we represent the shape-prior using GP-LVM and optimise an energy over the pose and a low-dimensional latent shape space. An implicit volumetric representation of the dense reconstruction (similar to that used in [16]) allows for a very efficient fusion of the dense reconstruction with the reconstructed object shape. The resulting system has a number of advantages over previous work: the dense reconstruction provides depth information to the tracker, leading to an improvement in pose and shape convergence and accuracy, while the tracker provides higher level object-based information to the SLAM, which means that (i) (some of the) unseen parts of the scene can be reconstructed, (ii) the scale of the map can be accurately estimated and; (iii) specularities on the object, which often cause image based dense SLAM to be erroneous, are less problematic.

The remainder of the paper is organized as follows. In Section 2 we describe our dense SLAM system. Next, in Section 3 the semantic part of the system is described, including the recognition of an object together and the estimation of its refined pose and shape. In Section 4 we present the way the information provided by the shape prior based estimator can be integrated into the dense SLAM. Finally, results are shown in Section 5.

2. Dense SLAM

Our dense SLAM system is structured as follows: firstly, assuming known camera pose from the PTAM system [11], dense depth maps are built using a brightness constancy assumption. Each depth map is subsequently fused into a global volumetric representation of the scene.

2.1. Local Depth Map Estimation

Closely mirroring the approach of [15], we formulate the initial depth map estimation problem as one of finding the depth of each point that is seen in one reference image. Following various prior art (e.g. [21]) [15] assumes known camera intrinsic and extrinsics and uses photo-consistency as a proxy for depth; viz, a point on a lambertian surface in the scene should project to the same colour in all the frames of the sequence, so the search for depth can be treated as a search for best photo-consistency.

More formally, let $\mathbf{u}$ denote the coordinates of a pixel in the reference image $\mathcal{I}_r$. The photo-consistency error of a 3D point along the ray corresponding to the pixel $\mathbf{u}$ and with a local depth d is measured as the normalized accumulated error between the colour on its projection in the reference image $\mathcal{I}_r$ and the colour on its projection onto the following frames $\mathcal{I}_m$:

$$\mathcal{C}(\mathbf{u}, d) = \frac{1}{N_p} \sum_m \|\mathcal{I}_r(\mathbf{u}) - \mathcal{I}_m \left(\pi({}^m M_r \pi^{-1}(\mathbf{u}, d))\right)\|_1. \quad (1)$$

where $\pi^{-1}(\mathbf{u}, d)$ refers to the transformation from the pixel coordinates and depth that brings to its homogeneous coordinates in the camera frame (computed using the intrinsic parameters of the camera). ${}^m M_r$ is the $\mathbb{SE}(3)$ matrix that maps the coordinates of a point in the reference camera frame into its coordinates in the camera frame m, this matrix is available from PTAM providing the world to camera(s) transformation ${}^m M_w$ using ${}^m M_r = {}^m M_w {}^r M_w{}^{-1}$. π is the function that projects the homogeneous coordinates of a point in the camera frame into its pixel coordinates in the image plane. N_p is the number of valid projection of the considered point along the sequence.

Searching for the actual surface along one ray is then equivalent to searching for the depth d that leads to the minimum photo-consistency error. The points along each ray are evenly sampled along the inverse depth so that the corresponding epipolar lines are evenly sampled. The depth estimate resulting from this process is however noisy, since (i) for many pixels, the brightness constancy is not respected; (ii) the pixels themselves are noisy, (iii) the evaluation space is discretised and (iv) uniform regions lack colour information.

To improve the depth map the standard approach is to regularise with a weak prior that favours continuous depth in uniform regions. This yields the following energy minimisation over the depth map $d(\mathbf{u})$:

$$E(d(\mathbf{u})) = \int_{\mathcal{I}} g(\mathbf{u}) \|\bigtriangledown d(\mathbf{u})\|_\epsilon + \gamma \mathcal{C}(\mathbf{u}, d(\mathbf{u})) \mathrm{d}\mathbf{u} \quad (2)$$

where $g(\mathbf{u})$ is computed with respect to the reference image gradients, so that g is equal to 1 where $\mathcal{I}_r$ is uniform (see [15] for further details). $\bigtriangledown d(\mathbf{u})$ is the depth map gradient, γ is a scalar weighting the effect of the regularization over the photo-consistency and $\|\cdot\|_\epsilon$ is the Huber norm of the intensity difference [25]. To solve this optimisation a primal-dual total variational approach is used.

2.2. Robust Map Representation

The process above can be repeated for several reference frames, and the resulting depth maps merged into a single global map. While [15] and others (e.g. [14]) simply accumulate the corresponding meshes, this has several disadvantages: first, it does not give any occupancy information even though this information is fundamental for many applications; second, measurement errors in individual meshes are never corrected so the benefit of the extra data from multiple mesh depth estimates at a single point is lost.

To address this limitation we fuse the local depth maps into a dense volumetric parametrization of the 3D world

akin to that used in [26, 10, 16]. The volume is discretised as a 3D grid of voxels, each containing two components: (i) F, the value of a signed distance function representing an approximation of the distance from the voxel to the closest volume surface, truncated to a maximum value μ to yield a Truncated Signed Distance Function (TSDF); and (ii) W, the weight corresponding to the confidence or amount of information accumulated by the voxel. The surface is recovered from this representation as the TSDF zero level set.

Each time a new depth map is generated, the values in the TSDF are updated to take the new information into account following a similar process to the one in [16]. Each voxel of the volume is updated as a weighted average of F^k, the current estimate of the minimum distance to the surface, and F', the distance as estimated in the new depth map:

$$W^{k+1} = W^k + W' \qquad F^{k+1} = \frac{F^k.W^k + F'.W'}{W^{k+1}} \quad (3)$$

Here $W' \in [0, \max_W]$ denotes the confidence in the new information and is a function of the angle between the surface normal and the optic ray, with greater confidence associated with frontal surfaces (and near zero confidence for surfaces tangential to an optic ray). Thus the new (approximate) distance to the surface is a weighted average of all previous measurements, helping to smooth out errors. Since in almost all respects our implementation closely follows the prior art, we refer the reader to [5] for further details.

3. Incorporating object knowledge

Our objective in this paper is to show how the ability to detect objects and incorporate them into a SLAM map is beneficial, as a step towards a more object-based, more semantically meaningful map. We propose a three stage process to this end: (i) in parallel with the depth-map acquisition, we search the raw 2D images acquired by the system for instances of a known object class using a sliding-window detector; from a set of detections, a first estimation of the object's pose and scale is performed; (ii) subsequently (and still in parallel with the depth-map estimation) we segment the object, and refine its pose, shape and scale so as to match the segmentation and depth-based cues; (iii) finally, this is fused with the volumetric representation. This process has been implemented in a multi-threaded architecture that achieves real-time performance.

3.1. Image-based object detection and localisation

While the dense SLAM system continually acquires new depth meshes at key-frames and fuses these into a global volumetric representation, in parallel we run a part-based object-class detector based on the effective procedure described in [9]. Although this cannot proceed at frame-rate, it does not need to. Rather, we process the most recent key-frame with the detector. If an object is found, further processing ensues (see below) but if not, the process retrieves the most recent key-frame and tries again. Using our implementation (based on [22]) we can process a single 320×240 frame in 2.1s on average. The use of this object class detector has several advantages in our context: it can detect object-classes that exhibit considerable in-class variation; it provides an object orientation estimation; we can use the location and types of the parts to estimate a rough 2D segmentation which in turn allows us to bootstrap a foreground/background appearance model from the image data.

To perform the optimization using shape priors, it is first necessary to have a coarse estimation of the pose of the detected object. To estimate this pose, we use a combination of the data available from the detector and from the dense SLAM map as follows. First, we triangulate a rough object centre in 3D by back-projecting the centre of the detection in two different views. We thus require a detection in at least two key-frames before proceeding to estimate the pose [12]. This of course incurs a delay, but does also help eliminate some false positives, since these are less likely to persist across frames than true positives.

Next, we estimate the vertical axis of the object by finding its supporting plane using the dense SLAM map. To do so, we make the assumptions that (i) there is indeed a supporting planar surface; and (ii) the supporting plane is unoccluded in the immediate area around the object. In particular, we sample depth values from pixels located immediately below the object in the key-frames and apply RANSAC to the resulting point cloud (see Figure 1(b)).

To estimate the second (and hence third and final) principal direction we consider the projection of the part configuration from the Felzenszwalb detector [9]. To cope with different generic views of a 3D object (side-on, three-quarter, etc) their algorithm defines a set of detectors, one per view, referred to by the authors as components. For instance, in the car model trained on the PASCAL 2007 dataset, the detector has 6 components from which the principal direction can be estimated. Using only 6 components to represent the full set of positions of the object around the vertical axis provides a coarse direction estimation, which we improve by interpolating the results from at least two views.

Finally, the size of the object is estimated using the size of the projection of the detected object in the first image and the depth of the object available from an initial triangulation. This, together with the intrinsic camera calibration, is sufficient to yield an estimate of the size of the object.

3.2. Low-level image statistics

An important feature of the detector is that it provides the position in the image of the object's parts. We leverage this information to build foreground and background colour models for the detected object as a whole. For each part we assume the availability of a trimap that denotes the

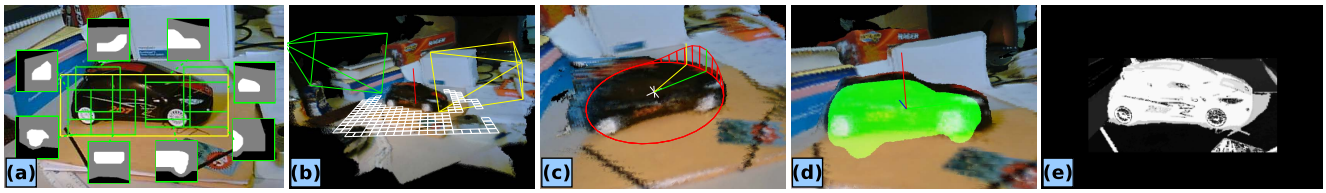

Figure 1. Coarse object pose initialization: (a) Result of the part based detector and related part priors, (b) supporting plane estimation, (c) set of principal angles resulting from the detections (yellow: from detection in first reference, green: second reference) and principal angle interpolation in red, (d) pose from detection, (e) pixel foreground posteriors with which the pose is refined.

set of pixels that, in instances of this part, are always foreground, or always background, or vary[1]. Colour histograms of both foreground $P(\mathbf{y}|C_f)$ and background $P(\mathbf{y}|C_b)$ are then easily created (here $\mathbf{y}$ represents the image pixel colour at location $\mathbf{u}$). Since the parts are not always well located, the histograms can be affected by outliers, so to mitigate this effect, we use only coarse bins ($6 \times 6 \times 6$). For each pixel $\mathbf{u}$, from these histograms can be defined the foreground and background posterior probabilities:

$$P_f(\mathbf{u}) = \frac{P(C_f|\mathbf{y})}{\eta_f} = \frac{P(\mathbf{y}|C_f)}{\eta_f P(\mathbf{y}|C_f) + \eta_b P(\mathbf{y}|C_b)}$$
$$P_b(\mathbf{u}) = \frac{P(C_b|\mathbf{y})}{\eta_b} = \frac{P(\mathbf{y}|C_b)}{\eta_f P(\mathbf{y}|C_f) + \eta_b P(\mathbf{y}|C_b)} \quad (4)$$

with η_f and η_b being the number of pixels in the foreground and background regions respectively.

The process is illustrated in Fig.1(a) where each part of the detected object is represented with its relative segmentation (black region represents the background, white is foreground and gray is unknown) and the foreground per-pixel posterior probability shown in Fig.1(e) where white represents high probabilities and black low ones.

3.3. Prior based shape and pose estimation

To segment the object in 3D (and subsequently fuse this information back into the volumetric model), we use a method similar to [19]. 3D shapes are represented volumetrically as Signed Distance Functions (SDFs), with the object surface implicitly defined by the zero-level set – making this a natural candidate for use with the volumetric models produced using the methods in Section 2. Within-class shape variation is represented via a low-dimensional embedding of the otherwise very high-dimensional 3D shape-space. The SDFs are first compressed using the Discrete Cosine Transform, retaining the n lowest frequencies in the DCT. The resulting space of DCT coefficients is then used to train a GP-LVM with a low-dimensional latent space. [18, 19] showed that this representation could more effectively capture useful within-class shape variation, and exclude out-of-class variation, than other competing methods such as PCA [24] or kPCA [6]. Unlike [19] however, in our current context, we have camera pose and depth information available from the SLAM system, which we aim to use to improve the object pose and shape recovery results. This implies, on the one hand, the need for a modified energy function (i.e. one that also takes depth into consideration), and, on the other hand, the need to match the scale between the SLAM system and the learned object coordinate system.

Our aim therefore becomes, for objects detected within the scene, the *simultaneous recovery of 3D shape* (parametrised by the latent space), *6D pose* and *scale*. We do this by defining an image and depth based energy function, finding its derivatives w.r.t. pose, scale and shape and using standard nonlinear minimisation techniques. Next we detail our energy function and its optimisation.

3.3.1 Energy Function

Our dense SLAM system provides pose and depth information over multiple frames coming from a single monocular source. We use the n_v key-frames from this data stream as multiple views in our joint 3D shape / 3D pose optimisation. Considering an image-depth domain Ω^v consisting of the image-depth points $\mathbf{x}$ (i.e. image coordinates and depth) from a key frame v, and the corresponding points $\mathbf{X}_0$ in the object coordinate frame, we write our energy function as:

$$E(\Phi) = \frac{1}{n_v} \sum_v \left(E_i^v(\Phi) + \alpha E_d^v(\Phi) \right) \quad (5)$$

$$E_i^v(\Phi) = \sum_{\mathbf{x} \in \Omega^v} \log(\pi^v(\Phi) P_f^v(\mathbf{x}) + (1 - \pi^v(\Phi)) P_b^v(\mathbf{x})) \quad (6)$$

$$E_d^v(\Phi) = \sum_{\mathbf{x} \in \Omega^v} \left(\frac{\Phi(\mathbf{X}_0)^2}{\Phi(\mathbf{X}_0)^2 + \sigma} \right) \quad (7)$$

where Φ is a 3D SDF. This energy function combines an image based error $E_i^v(\Phi)$ and a depth based one $E_d^v(\Phi)$, with α representing the balance between the two. Note that there is a principled, probabilistic explanation behind this coupling, as each part of the energy function can be written as the log of a per pixel joint probability. Furthermore, since the two parts of the energy function are sums of per-pixel

[1] Though the trimaps could be learned from good segmentations of training data, in practice we crafted these trimaps manually

values, we can perform the multi view information fusion by simply averaging the per-view energy function values.

$E_i^v(\Phi)$ measures the discrimination between statistically defined foreground and background regions, as a function of the projected 3D SDF Φ, using the functions P_f and P_b from eq (4) in each reference view v. This measure, first proposed for 2D tracking by [2], is in contrast to the full random forest classfier used in [19] to obtain image statistics. Here we can use these much weaker (and therefore much faster to compute) image statistics because of the availability of multiple views. $\pi^v(\Phi)$ projects Φ to a 2D occupancy map, with value 1 inside the projection outline and 0 outside. It does this by evaluating, for each pixel in the image-depth domain Ω^v, the probability of it being a projection of a voxel "inside" the 3D SDF Φ. This is written as:

$$\pi^v(\Phi) = 1 - \exp\Big(\sum_{\text{all } \mathbf{X}_0 \text{ on ray}} \log\big(1 - \frac{e^{\Phi(\mathbf{X}_0)\zeta}}{e^{\Phi(\mathbf{X}_0)\zeta}+1}\big)\Big) \quad (8)$$

where ζ controls the smoothness of the projection (we use $\zeta = 0.75$). This expression is matematically equal to the product of the innermost terms of the logarithm, but we chose to evaluate it as an exponential of a sum of logarithms for improved numerical stability.

$E_d^v(\Phi)$ represents the negative log of the joint probability that selected image-depth points $\mathbf{x}$ back-project onto the zero level of Φ (i.e the surface of the 3D object model), using the pose corresponding to view v and assuming pixel-wise independence. The probability that an image-depth point lies on the object surface is equal to the probability that the back projected 3D point lies on the zero-level of the SDF. We compute this probability as the negative exponential of the robust German-McClure function [3]:

$$P(X|\Phi, \Omega^v) = e^{-\frac{\Phi(\mathbf{X}_0)^2}{\sigma + \Phi(\mathbf{X}_0)^2}} \quad (9)$$

This probability equals one when a depth pixel is back-projected onto the zero-level of the SDF and decreases monotonically at a rate controlled by σ (we used $\sigma = 100$). This approach was also used in [20], in which depth data coming from a Microsoft Kinect unit was used for simultaneous model based 3D tracking and calibration. Unlike [20] however, here we (i) also make use of the RGB image data, (ii) adapt the shape of the object and (iii) use the dense SLAM system to provide depth data.

To minimise this energy, we compute its derivatives with respect to pose, shape and scale and use them in a Levenberg-Marquardt style nonlinear minimisation.

3.3.2 Pose/Scale Derivatives

Each image-depth point $\mathbf{x}$ in a view v is the projection of a point $\mathbf{X}$ in the camera coordinate frame, which itself has a corresponding point $\mathbf{X}_0$ in the object coordinate frame. The transformation from $\mathbf{X}$ to $\mathbf{x}$ is parametrised by the camera intrinsic parameters corresponding to view v. The transformation from $\mathbf{X}_0$ to $\mathbf{X}$ is $\mathbf{X} = {}^vM_o\mathbf{X}_0$, where ${}^vM_o = {}^vM_w{}^wM_o$ with ${}^vM_w$ being the $\mathbb{SE}(3)$ transformation from the world to the reference camera v coordinates defined in Section 2. ${}^wM_o$ is the transformation from object to world coordinates, i.e. the unknown object pose and scale. This is in contrast to [19], which required no partition for ${}^vM_o$, as only a single view was used.

Let λ_p, $p \in 1, \ldots, 7$ represent the unknown 6 DoF pose parameters (three for translation and three for Rodrigues represented rotation) and the unknown scale. The derivative of our energy function wrt. λ_p is:

$$\frac{\partial E}{\partial \lambda_p} = \frac{1}{n_v}\sum_v \Big(\frac{\partial E_i^v}{\partial \lambda_p} + \alpha\frac{\partial E_d^v}{\partial \lambda_p}\Big) \quad (10)$$

$$\frac{\partial E_i^v}{\partial \lambda_p} = \sum_{\mathbf{x}\in\Omega} \frac{P_f^v(\mathbf{x}) - P_b^v(\mathbf{x})}{\pi^v(\Phi)P_f^v(\mathbf{x}) + (1-\pi^v(\Phi))P_b^v(\mathbf{x})}\frac{\partial \pi^v}{\partial \lambda_p} \quad (11)$$

$$\frac{\partial E_d^v}{\partial \lambda_p} = \sum_{\mathbf{x}\in\Omega} \frac{2\sigma\Phi(\mathbf{X}_0)}{(\Phi(\mathbf{X}_0)^2+\sigma)^2}\frac{\partial \Phi}{\partial \lambda_p} \quad (12)$$

where

$$\frac{\partial \pi^v}{\partial \lambda_p} = (1-\pi^v(\Phi))\sum_z \frac{e^{\Phi(\mathbf{X}_0)\zeta}}{e^{\Phi(\mathbf{X}_0)\zeta}+1}\frac{\partial \Phi^v}{\partial \mathbf{X}_n^v}\frac{\partial \mathbf{X}_n^v}{\partial \lambda_p} \quad (13)$$

$$\frac{\partial \Phi}{\partial \lambda_p} = -\frac{\partial \Phi}{\partial \mathbf{X}_0}({}^vM_o)^{-1}{}^vM_w\frac{\partial {}^wM_o}{\partial \lambda_p}({}^vM_o)^{-1}\mathbf{X} \quad (14)$$

As in [19], in order to make the computation of $\frac{\partial \pi^v}{\partial \lambda_p}$ easier, we use OpenGL-style normalised device coordinates for Φ and $\mathbf{X}$. In this coordinate system the 3D SDF Φ is transformed into Φ^v, using the pose, scale and intrinsics corresponding to view v. Also, the 3D point that projects to $\mathbf{x}$ under the known camera calibration for view v is now denoted by $\mathbf{X}_n^v$. Therefore, using the chain rule, we can write:

$$\frac{\partial \mathbf{X}_n^v}{\partial \lambda_p} = \frac{\partial \mathbf{X}_n^v}{\partial \mathbf{X}}\frac{\partial \mathbf{X}}{\partial \lambda_p} \quad (15)$$

where $\frac{\partial \mathbf{X}_n^v}{\partial \mathbf{X}}$ are the derivatives of the standard normalised device coordinate conversion (i.e. projection and normalisation of the Z coordinate) and $\frac{\partial \mathbf{X}}{\partial \lambda_p}$ follow in a straightforward manner as derivatives of $\mathbf{X} = {}^vM_w{}^wM_o\mathbf{X}_0$, wrt. pose and scale. Finally, the derivatives $\frac{\partial {}^wM_o}{\partial \lambda_p}$ are computed analogously and $\frac{\partial \Phi}{\partial \mathbf{X}_0}$ and $\frac{\partial \Phi^v}{\partial \mathbf{X}_n}$ are computed numerically.

3.3.3 Shape Derivative

Our aim is to capture and use prior knowledge on the possible 3D shapes an object can take. We do this, in a manner similar to [19], by using a dimensionality reduction technique called Gaussian Process Latent Variable Models, to learn nonlinear and probabilistic latent shape spaces.

Using a nonlinear minimisation, GP-LVM finds, for a set of n high dimensional variables $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_n]$, a set of low dimensional variables $\mathbf{L} = [\mathbf{l}_1, \ldots, \mathbf{l}_n]$ and the hyperparameters of a Gaussian Process (GP) mapping $\mathbf{L}$ into $\mathbf{H}$. In our case the high dimensional variables $\mathbf{H}$ are $128 \times 128 \times 128$ SDFs, compressed, for tractability, using the first $25 \times 25 \times 25$ harmonics of the discrete cosine transform (DCT). The derivatives of the energy function wrt. each dimension of $\mathbf{l}$, denoted by $\mathbf{l}_q$ follow analogously to those wrt. pose and scale, by replacing $\frac{\partial \Phi^v}{\partial \mathbf{X}_n^v} \frac{\partial \mathbf{X}_n^v}{\partial \lambda_p}$ with $\frac{\partial \Phi^v}{\partial \mathbf{l}_q}$ and $\frac{\partial \Phi}{\partial \lambda_p}$ with $\frac{\partial \Phi}{\partial \mathbf{l}_q}$. These final two derivatives are the ones of the standard GP-LVM generative process [13], on which the inverse DCT transform has been applied.

4. Map update

Once the shape and pose estimation of the object has converged (as measured using the standard Levenberg-Marquardt test), we fuse the shape SDF Φ with the global map. To do so, we use the same formulation as in equation 3. The new distance F' is defined by the object SDF while the confidence W' (or weight) in this distance is defined so that only the voxels close to the object surface are modified:

$$W'(\mathbf{x}) = \begin{cases} \max_W -W^k(\mathbf{x}), & \text{if } \Phi(\mathbf{x}) < \eta \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

W' is set to be maximum close to the object's surface so that the information provided by the prior-based optimization overwhelms the less accurate depth regularization accumulated by the dense SLAM system.

Following this merger, it is likely that some voxels will have a well-defined distance value but no colour information, since they have never been observed directly by the camera, and instead arise from the object shape. In the experiments reported in this paper, we make an expedient assumption that the colour information can be propagated via symmetry: we use the vertical plane that cuts the object in two along its major horizontal axis, and update the colour of each uncoloured voxel of the car with the colour from its symmetrical correspondence.

5. Implementation and results

Our system has been implemented using C++ and CUDA on a quad-core 3.4 GHz PC equipped with two GTX 480 GPU and a webcam acquiring images with a 640×480 pixels resolution.

One thread and one GPU are devoted to performing the dense reconstruction and fusion of the volumetric representation. Since the photo-consistency error computation and TSDF update are highly parallelisable, depth-map generation and volumetric fusion runs in real-time: the accumulation of the photometric error takes 12 ms for each input frames; the total variational energy minimisation typically takes 20 ms; new depth-maps are merged into the global TSDF in 6 ms.

A second CPU thread and second GPU perform the object detections and the subsequent segmentation and pose estimation using shape priors. The detection process uses the open-source code from [22] and takes 2.1s on 320×240 sized images. Though this is time-consuming, only the reference frames that are used to compute a depth map are processed, this happens parallel with depth-map computation and so has no immediate impact on the tracking and mapping part of the system. The shape-prior segmentation and pose optimization runs at 40ms per iteration using the depths and colours in 2 key-frames, and typically requires about a hundred iterations to converge. The addition of further key-frames tends to improve the accuracy and convergence properties, as illustrated in Figure 2, but at the expense of extra computation per iteration (approx. 20ms per extra key-frame, per iteration). After convergence, the merger of the object's TSDF with the global one is accomplished as above, in 6ms.

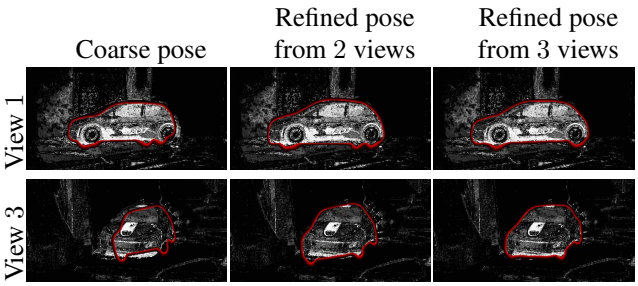

Figure 2. Effect of the number of views used for the object shape/pose estimation. Resulting object projected onto views of the per-pixel fg/bg membership probabilities. (left) initial coarse pose (Section 3.1); (middle) shape/pose estimate after convergence using two key-frames; (right) the addition of a further key-frame yields a better final estimate of the pose and shape.

An example of the system in action is shown in Figure 3 and in the supplementary material. Note that in the reconstruction without considering the known object detection and segmentation (first row) the shape of the car is merged into the background because there is no visual evidence to separate them, while the additional shape knowledge following the detection and segmentation correctly separates the rear, unobserved part of the car from the rest of the scene. A second example in Figure 4 shows how transparent and specular objects result in corrupted surfaces from "standard" dense SLAM, are corrected via the object shape calculation.

After the addition of an object to the map, the process continues as before, merging new depth-maps with the volumetric model, and further detections can be made to merge additional objects. In this case some care must be exercised

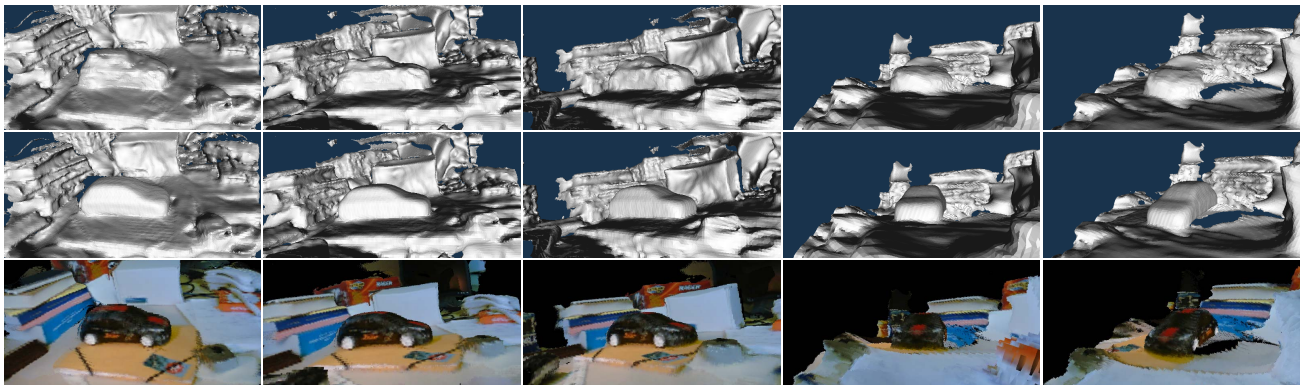

Figure 3. First row : Fong reprojection of the reconstruction without priors, second row: with priors, third row: textured with priors.

to ensure that detections are matched correctly between key-frames. In our present implementation objects are treated sequentially (i.e. only one object can be segmented and merged at a time), with basic colour models used to ensure the fidelity of the matches between key-frames. An example of the resulting reconstruction is illustrated in figure 5.

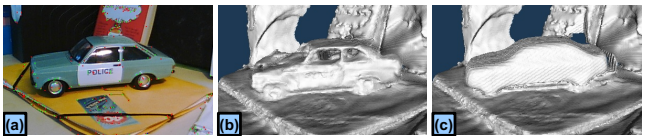

Figure 4. Reconstruction with specularities and transparent surfaces: (a) scene, (b) reconstruction without priors, (c) reconstruction with priors.

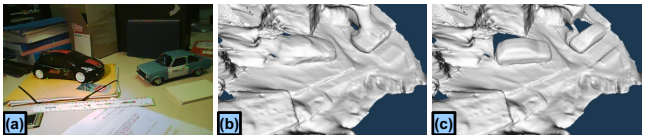

Figure 5. Reconstruction of multiple objects: (a) scene, (b) reconstruction without priors, (c) reconstruction with priors.

Finally, an experiment is proposed to quantify the improvement over the quality of the dense reconstruction. Although obtaining proper ground truth is problematic, we use the dense reconstruction from KinectFusion [16] as a reasonable proxy (note that KinectFusion uses an RGB-D camera and fuses the sensor's depth maps, in contrast to dense monocular SLAM that uses only a single standard camera). The reconstruction error measure is computed to be the average value of the estimated TSDF over the ground truth surface (thus wherever the TSDF zero level coincides with the ground truth surface the error contribution is zero). Figure 6 shows the evolution of the reconstruction error over the whole resulting volume and over the the voxels close to the object together with the reconstruction steps (new depth map integrated in frames 247, 598 and 884 and object merged in frame 1107). As the figure shows, the reconstruction error start at a value of 20, equal to the SDF threshold (no surface defined) and gets lower at every merged depth map as well as when the object is merged particularly in the vicinity of the object, confirming the improvements resulting from the use of the shape priors.

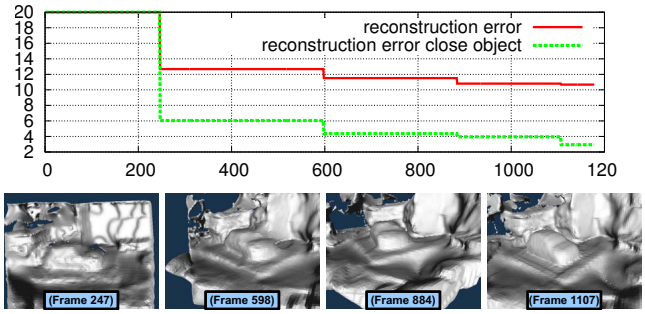

Figure 6. Evolution of the reconstruction surface and its error with the integration of new depth map and object shape and pose.

6. Conclusion

We have shown that the use of object-specific knowledge can be incorporated into a SLAM system to improve the estimation of the maps produced. In particular we have demonstrated a *real-time, end-to-end fully automatic system* that combines several state-of-art techniques – in dense SLAM, object detection and 3D segmentation and pose estimation – for finding dense scene structure that can take into account high-level prior shape information. This is shown to have the benefit of improving the map in non-lambertian parts of the scene, and even allowing depth estimates in unobserved parts of the scene. More generally we see the incorporation of object-specific knowledge into SLAM, as demonstrated here and by others, as a step towards more semantic levels of representation, a bridge between the geometric reconstruction of traditional SLAM and structure-from-motion, and the more semantically meaningful maps that humans are comfortable with.

One point of weakness of the system is in the limited ap-

pearance model we use for the object segmentation. Though colour histograms can provide accurate segmentations in many cases, if there is colour confusion between the foreground and background the segmentation is prone to fail. Having made a detection using [9], we currently discard the underlying image evidence used by that detector other than to aggregate the colour histogram. A tighter coupling between the detection process and the shape/pose optimisation could improve matters.

At present we have adopted a volumetric representation closely mirroring [16] in which the confidence in the TSDF value at a voxel is given by sensible but ultimately fairly ad-hoc weight. In fusing the dense SLAM depth data this works well, but it is not immediately clear if there is a rigorous way to set the weights for the TSDF coming from the shape-prior source. Presently we take the expedient of using high (maximum) weights for the object TSDF, but this has the disadvantage that further shape refinement cannot be seamlessly intergrated. More ambitiously, a characteristic of the objects we hope to identify and segment as part of the system is that they can potentially move. [19] shows that segmentation and tracking using volumetric models is indeed possible, but our global volumetric representation does not currently admit moving objects.

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

References

[1] S. Y. Bao and S. Savarese. Semantic structure from motion. In *CVPR 2011*, pages 2025–2032. 1

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

[3] M. J. Black and A. D. Jepson. EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. In *IJCV 1998*, pages 329–342. 5

[4] R. O. Castle, D. J. Gawley, G. Klein, and D. W. Murray. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *ICRA 2007*, pages 4102–4107. 1

[5] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, pages 303–312, 1996. 3

[6] S. Dambreville, R. Sandhu, A. Yezzi, and A. Tannenbaum. Robust 3D Pose Estimation and Efficient 2D Region-Based Segmentation from a 3D Shape Prior. In *ECCV 2008*, pages 169–182. 1, 4

[7] A. J. Davison, I. D. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *T-PAMI 2007*, 29(6):1052–1067. 1

[8] A. R. Dick, P. H. S. Torr, and R. Cipolla. A Bayesian Estimation of Building Shape Using MCMC. In *ECCV 2002*, pages 852–866. 1

[9] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *T-PAMI 2010*, 32(9):1627–1645. 3, 8

[10] G. Graber, T. Pock, and H. Bischof. Online 3d reconstruction using convex optimization. In *ICCV 2011 Workshops*, pages 708–711. IEEE, 2011. 1, 3

[11] G. Klein and D. W. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR 2007*, pages 225–234. 1, 2

[12] K. Lai, L. Bo, X. Ren, , and D. Fox. Detection-based object labeling in 3d scenes. In *ICRA*, pages 1330–1337, May 2012. 3

[13] N. Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *JMLR 2005*, 6:1783–1816. 6

[14] S. Lieberknecht, A. Huber, S. Ilic, and S. Benhimane. RGB-D camera-based parallel tracking and meshing. In *ISMAR 2011*, pages 147–155, 2011. 2

[15] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV 2011*, pages 2320–2327. 1, 2

[16] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR 2011*, pages 127–136. 2, 3, 7, 8

[17] V. A. Prisacariu and I. Reid. Nonlinear Shape Manifolds as Shape Priors in Level Set Segmentation and Tracking. In *CVPR 2011*, pages 2185–2192. 1

[18] V. A. Prisacariu and I. Reid. Shared shape spaces. In *ICCV 2011*. 4

[19] V. A. Prisacariu, A. Segal, and I. Reid. Simultaneous Monocular 2D Segmentation, 3D Pose Recovery and 3D Reconstruction. In *ACCV 2012*. 1, 2, 4, 5, 8

[20] C. Y. Ren and I. Reid. A unified energy minimization framework for model fitting in depth. In *ECCV 2012 Workshops*, pages 72–82. 5

[21] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast Cost-Volume Filtering for Visual Correspondence and Beyond. In *CVPR 2011*. 2

[22] D. Rodriguez-Molina and M. J. Marin-Jimenez. LibPaBOD: A library for part-based object detection in C++, 2011. Software available at `http://www.uco.es/~in1majim/`. 3, 6

[23] R. Sandhu, S. Dambreville, A. Yezzi, and A. Tannenbaum. A Nonrigid Kernel-Based Framework for 2D-3D Pose Estimation and 2D Image Segmentation. *T-PAMI 2011*, 33(6):1098–1115. 1

[24] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, E. Grimson, and A. Willsky. A Shape-based Approach to the Segmentation of Medical Imagery Using Level Sets. *T-MI 2003*, 22(2):137–154. 1, 4

[25] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 Optical Flow. In *BMVC 2008*. 2

[26] C. Zach, T. Pock, and H. Bischof. A Globally Optimal Algorithm for Robust TV-L^1 Range Image Integration. In *ICCV 2007*. 3