Pattern Recognition Letters 50 (2014) 43–54

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Robust shape from depth images with GR^{2}T ☆

Jonathan Ruttle, Claudia Arellano, Rozenn Dahyot *

School of Computer Science and Statistics, Trinity College Dublin, Ireland

ARTICLE INFO

Article history:
Available online 31 January 2014

Keywords:
Shape from depth
Generalised Relaxed Radon Transform (GR^{2}T)
Noise modelling

ABSTRACT

This paper proposes to infer accurately a 3D shape of an object captured by a depth camera from multiple view points. The Generalised Relaxed Radon Transform (GR^{2}T) [1] is used here to merge all depth images in a robust kernel density estimate that models the surface of an object in the 3D space. The kernel is tailored to capture the uncertainty associated with each pixel in the depth images. The resulting cost function is suitable for stochastic exploration with gradient ascent algorithms when the noise of the observations is modelled with a differentiable distribution. When merging several depth images captured from several view points, extrinsic camera parameters need to be known accurately, and we extend GR^{2}T to also estimate these nuisance parameters. We illustrate qualitatively the performance of our modelling and we assess quantitatively the accuracy of our 3D shape reconstructions computed from depth images captured with a Kinect camera.

1. Introduction

With the increasing popularity of 3D displays and 3D printers, capturing automatically and easily a 3D shape of an object of interest using low cost hardware like the Kinect sensor opens interesting perspectives for non-specialist users for archiving, reproducing or displaying objects. Several methods have been recently proposed to recover a 3D shape from depth images (these are reviewed in paragraph 2.1). However all approaches convert the depth images into point clouds in the 3D space with little consideration to how the noise on a pixel of a depth image translates into uncertainty on the corresponding 3D point. Moreover smoothing the depth images is often used as a preprocess to remove noise before conversion to a 3D point cloud but this implies that valuable information can also be lost. In this paper, we defend an alternative strategy for estimating an accurate 3D shape from depth images: the original recorded data is not transformed into a point cloud, and thanks to prior knowledge about the sensor, the noise associated with each recorded pixel is modelled explicitly. As a consequence, our cost function used for 3D shape inference represents well the global uncertainty related to the noise of the depth images.

We use the Generalised Relaxed Radon Transform (GR^{2}T [1] is reviewed in paragraph 2.2) to model the cost function merging the information from all the recorded depth pixels. The resulting objective function ressembles a kernel density estimate with tailored kernels capturing the uncertainty about the depth value and the location of each pixel (Section 3). The distribution of the sensor noise on the pixel is chosen Normal and this allows the optimisation of the cost function with standard gradient ascent algorithms suitable for parallel computation [2].

Fig. 1 summarises our experimental setting: several views are recorded using a Kinect camera from multiple views distributed around the object using a turning table. Accurate extrinsic camera parameters are often difficult to get at calibration stage. We propose to extend GR^{2}T so that these nuisance parameters are re-estimated using the depth images (Section 3.4) before inferring the 3D shape of the object. We assess experimentally our framework in Section 4 and subsequently proposed further possible extensions (Section 5).

2. Background

2.1. 3D reconstruction with RGB-D cameras

Cui et al. [3,4] have proposed to reconstruct an object using sequences of depth images captured with a time of flight camera. Merging several scans together (frame to frame) allows to improve the quality of the mesh despites the strong noise in the depth data. The depth data is converted into a 3D point cloud and the uncertainty (or bias) is modelled by a systematic offset in the direction of the camera ray. This model accounts only for the noise on the depth data but not for the uncertainty associated with the pixel resolution. The camera extrinsic parameters of rotation and

☆ This paper has been recommended for acceptance by Xiaoyi Jiang.

* Corresponding author. Tel.: +353 1 896 1760.
E-mail address: Rozenn.Dahyot@tcd.ie (R. Dahyot).

http://dx.doi.org/10.1016/j.patrec.2014.01.016
0167-8655/© 2014 Elsevier B.V. All rights reserved.

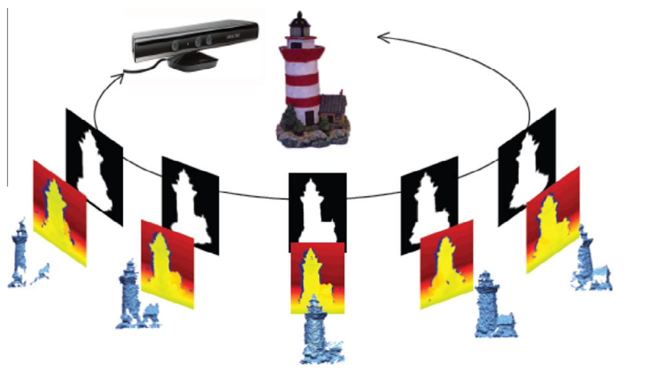

Fig. 1. 3D reconstruction using the Kinect. In practice a turning table is used to capture multiple views of depth data.

translation are estimated by mapping two depth scans converted as 3D point clouds. Their cost function to estimate the rigid transformation can be understood as maximising the likelihood function (the Gaussian mixture) modelled by one point set, while the second corresponds to the observations for computing the likelihood (see Section 2.1.2).

Izadi et al. have proposed a real time 3D reconstruction algorithm (KinectFusion) using depth information [5]. This technique allows to recover a dense 3D map of a scene. As a pre-process, a bilateral filter is used to remove the noise in the depth images before conversion to 3D point clouds. This has two limitations: first any filtering will reduce noise and also discard information of importance, second the uncertainty about the depth information and the pixel positions are not tackled. The depth map is converted into 3D points and normals in the coordinate space of the camera. As the camera is moving, the extrinsic camera parameters are estimated using a fast version of the Iterative Closest Point (ICP, see Section 2.1.1) algorithm in a *frame* to *all frame* strategy. Given the global pose of the camera, oriented points are converted into global coordinates, and a single 3D voxel grid is updated (to update the volumetric surface representation). KinectFusion algorithm uses a fixed voxel representation of the real world captured by the depth camera that can be memory demanding and several extensions have been proposed to tackle this issue [6–8]. Efficiency of KinectFusion algorithm has also been tackled by implementing faster GPU versions [9,10]. KinectFusion is mainly focused on real time scene mapping to the detriment of both robustness and accuracy of the reconstructed object [11]. Dedicated algorithms with prior information about the shape to reconstruct have been proposed (e.g. for human body [12,13] or face [11]). Using prior information about the shape to infer (faces), Hernandez et al. report an average error of 1 mm with respect to the ground truth captured with a Laser scanner [11].

Recovering 3D shapes from depth images has therefore been tackled so far in three distinctive steps: denoising of depth images (by filtering or averaging several recorded frames), converting the depth images in 3D point clouds, and registering the point clouds to recover a coherent surface of the object that has been captured from multiple views. For clarity we now review explicitly the different objective functions that are optimised for finding the transformation t that aligns one point cloud (noted $\mathcal{V} = \{v^{(j)}\}_{j=1,\dots,N_v}$) onto another one (say $\mathcal{U} = \{u^{(i)}\}_{i=1,\dots,N_u}$).

2.1.1. Iterative Closest Point (ICP)

One popular method to perform registration between two point sets is the Iterative Closest Point (ICP) algorithm [14,15]. The ICP algorithm was introduced by Besl and McKay [14] and Zhang [15]. It is based on finding point-to-point correspondences $\mathcal{C} = \{(u^{(i)}, v^{(j)})\}$ between the two data sets using the nearest neighbour criterion. Once the correspondences have been found the transformation t is estimated by maximizing the following likelihood:

$$\hat{t} = \arg\max_t \left\{ \prod_{(u^{(i)},v^{(j)})\in\mathcal{C}} \mathcal{N}(t(v^{(j)}) - u^{(i)}; 0, \Sigma) \right\} \tag{1}$$

where the notation $\mathcal{N}(\Theta; \mu, \Sigma)$ indicates a Normal distribution for random variable Θ with mean μ and isotropic covariance matrix $\Sigma = h^2\mathrm{I}$ controlled by a bandwidth h (with I identity matrix). These two steps (correspondence-transformation) are iterated until the convergence criterion is reached. It is one of the most used algorithms for registration due to its simplicity and low computational complexity. Many improvements have been made to the basic ICP algorithm [16–21] but the approach is sensitive to both outliers and its initialization. ICP requires the initial position of the two point sets to be adequately close. This is usually achieved by matching manually labelled points in both sets [22,23].

2.1.2. Likelihood with Gaussian mixture

Cui et al. model one point cloud as a Gaussian mixture [3,4]:

$$p_\Theta(\Theta) = \frac{1}{N_u}\sum_{i=1}^{N_u} \mathcal{N}(\Theta; u^{(i)}, \Sigma) \tag{2}$$

Assuming the transformed point cloud $\{t(v^{(j)})\}_{j=1,\dots,N_v}$ is sampled from p_Θ, the transformation t is then estimated by maximising the likelihood:

$$\hat{t} = \arg\max_t \left\{ \prod_{j=1}^{N_v} p_\Theta(t(v^{(j)})) = \prod_{j=1}^{N_v} \frac{1}{N_u} \sum_{i=1}^{N_u} \mathcal{N}(t(v^{(j)}); u^{(i)}, \Sigma) \right\} \tag{3}$$

Using this cost function, Cui et al. indicate that while correspondences between points are not required, overlaps in the two point clouds needs to be considered [3,4]: if $v^{(j)}$ is not a point from the same area as described by set $\mathcal{U}$, then $p_\Theta(t(v^{(j)}))$ is going to be close to zero, affecting greatly the estimation of t like an outlier would do.

2.1.3. Kernel correlation & distance between distributions

The transformation t can be estimated by maximising the kernel correlation [24]:

$$\hat{t} = \arg\max_t \left\{ \frac{1}{N_u N_v} \sum_{i=1}^{N_u}\sum_{j=1}^{N_v} \mathcal{N}(t(v^{(j)}) - u^{(i)}; 0, \Sigma) \right\} \tag{4}$$

The cost function for this estimation consists in the cross correlation between two probability density functions (pdf) each of which represents a point cloud [24]. The pdf corresponding to the reference point cloud is modelled as the empirical density function, while the second point set is modeled with a Gaussian mixture (Eq. (2)). Defining the empirical estimate of the distribution of Θ using observations in $\mathcal{V}$ (using the Dirac kernel δ) by:

$$p^e_\Theta(\Theta) = \frac{1}{N_v}\sum_{j=1}^{N_v} \delta(\Theta - t(v^{(j)})) \tag{5}$$

the cross correlation can be understood as:

$$\hat{t} = \arg\max_t \left\{ \int p^e_\Theta(\Theta)\, p_\Theta(\Theta)\, d\Theta = \langle p^e_\Theta | p_\Theta \rangle \right\} \tag{6}$$

with p_Θ defined in Eq. (2). This technique does not require correspondences to be computed. It is robust to regions that do not overlap in the two point clouds and outperforms ICP based techniques [24].

For a rigid transformation, maximising the correlation between two pdfs can be shown to be equivalent to minimising the Euclidian distance between the pdfs [25]. Minimising the Euclidian distance between two probability density functions is a robust approach for parameter estimation [26].

2.1.4. Remarks

The transformation t of interest in this work corresponds to the displacement of the camera between two frames so that all extrinsic camera parameters associated with each frame can be recovered. These parameters are essential to get a good estimate of the 3D reconstruction. So far, this has been addressed by converting depth images into point clouds [5,3,4]. Converting depth information into 3D points makes the new observations to be directly related to the latent surface of the object that is to infer. Hence transforming depth maps into 3D point clouds is a convenient approach to infer 3D shape. However the uncertainty associated with each 3D point is more difficult to model. Indeed isotropic covariance matrices $\Sigma = h^2 \mathrm{I}$ controlled by a unique bandwidth have been used so far [24,25,4]. We review next the Generalised Relaxed Radon Transform and underline its similarity with the robust kernel correlation cost function.

2.2. Generalised Relaxed Radon Transform (GR^{2}T)

The Generalised Relaxed Radon Transform (GR^{2}T) has recently been proposed for robust inference [1] and its modelling relies on the following stochastic system of equations:

$$\lambda + F(\mathbf{x}, \Theta) = \epsilon \sim p_\epsilon(\epsilon) \tag{7}$$

with $\mathbf{x} \in \mathbb{R}^{d_\mathbf{x}},\ \Theta \in \mathbb{R}^{d_\Theta},\ \lambda \in \mathbb{R}^d,\ \epsilon \in \mathbb{R}^d$ and F a given link function. Using Bayes theorem, the probability density of (λ, Θ) can be computed as an expectation:

$$\begin{aligned} p_{\Theta\lambda}(\Theta,\lambda) &= \int p_{\lambda\Theta|\mathbf{x}}(\lambda,\Theta|\mathbf{x})\ p_\mathbf{x}(\mathbf{x})\ d\mathbf{x} = \langle p_{\lambda\Theta|\mathbf{x}}|\ p_\mathbf{x}\rangle \\ &= \int p_{\lambda|\mathbf{x}\Theta}(\lambda|\mathbf{x},\Theta)\ p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x})\ p_\mathbf{x}(\mathbf{x})\ d\mathbf{x} \\ &= \mathbb{E}_\mathbf{x}[p_{\lambda|\mathbf{x}\Theta}(\lambda|\mathbf{x},\Theta)\ p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x})] \\ &= \mathbb{E}_\mathbf{x}[p_\epsilon(\lambda + F(\mathbf{x},\Theta))\ p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x})] \end{aligned} \tag{8}$$

Note that when the error distribution p_ϵ is the Dirac density function $\delta(\epsilon)$, the probability density function $p_{\Theta\lambda}$ corresponds to the Generalised Relaxed Radon Transform [27]. In a similar fashion as the Kernel correlation (Eq. (6)), GR^{2}T can be understood as the correlation between $p_{\lambda\Theta|\mathbf{x}}$and $p_\mathbf{x}$.

GR^{2}T can also be understood as the expectation of $p_{\lambda\Theta|\mathbf{x}}$ with respect to the observed random variable $\mathbf{x}$. Having collected independent observations $\{\mathbf{x}^{(i)}\}_{i=1,\dots,N}$ of the random variable $\mathbf{x}$, the joint density $p_{\Theta\lambda}(\Theta,\lambda)$ can then be estimated by the empirical average [28,1]:

$$\begin{aligned} \hat{p}_{\Theta\lambda}(\Theta,\lambda) &= \sum_{i=1}^N p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x}^{(i)})\ p_{\lambda|\mathbf{x}\Theta}(\lambda|\mathbf{x}^{(i)},\Theta)\ \pi_i \\ &= \sum_{i=1}^N p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x}^{(i)})\ p_\epsilon(\lambda + F(\mathbf{x}^{(i)},\Theta))\ \pi_i \end{aligned} \tag{9}$$

The priors $\{\pi_i\}_{i=1,\dots,N}$ associated with the observations are often assumed equiprobable $\pi_i = \frac{1}{N}$. When Θ and $\mathbf{x}$ are assumed independent, Eq. (9) becomes:

$$\hat{p}_{\Theta\lambda}(\Theta,\lambda) = p_\Theta(\Theta) \times \underbrace{\sum_{i=1}^N p_\epsilon(\lambda + F(\mathbf{x}^{(i)},\Theta))\ \pi_i}_{\overline{\text{lik}}(\lambda,\Theta)} \tag{10}$$

The term $\overline{\text{lik}}(\lambda,\Theta)$ can be understood as an average of likelihoods that consider only one observation at a time. It can be used as a cost function that provides a one-to-many mapping between the space where observations occur and the latent space to explore [29]. In a similar fashion to the Hough transform, it is a robust modelling because an outlier creates a kernel with value close to zero that do not affect the overall cost function. Ref. [1] explores with more details the relation of GR^{2}T with the following robust framework: the Hough transform [30,29,31,32], M-estimators [33] and Generalized Projection Based M-Estimators [34,35]. When the densities function p_ϵ is chosen differentiable, then the average likelihood can be optimised with standard stochastic exploration techniques and in particular with gradient ascent algorithms [28].

2.3. GR^{2}T for 3D shape reconstruction from multiple views

Ruttle et al. [36] proposed a kernel density estimate as a cost function to infer shape from silhouettes as an extension to the classic 3D histogram for estimating the visual hull by cone intersections [37,38]. Using a differentiable Gaussian kernel p_ϵ allows them to optimise the cost function with gradient ascent algorithms that are suitable for parallel architecture [2]. The link function used by Ruttle et al. [36] corresponds to the relation between the 3D position $\Theta = (\theta_1, \theta_2, \theta_3)$ in the real world and the pixel position $\mathbf{x} = (x_1, x_2)$ where Θ is projected using a projection matrix P associated with a pinhole camera model:

$$F(\mathbf{x},\Theta) = \begin{pmatrix} F_1(\mathbf{x},\Theta) \\ F_2(\mathbf{x},\Theta) \end{pmatrix} = \begin{pmatrix} x_1 - \frac{\theta_1 P_{11} + \theta_2 P_{12} + \theta_3 P_{13} + P_{14}}{\theta_1 P_{31} + \theta_2 P_{32} + \theta_3 P_{33} + P_{34}} \\ \\ x_2 - \frac{\theta_1 P_{21} + \theta_2 P_{22} + \theta_3 P_{23} + P_{24}}{\theta_1 P_{31} + \theta_2 P_{32} + \theta_3 P_{33} + P_{34}} \end{pmatrix} \tag{11}$$

Kim et al. [39] proposed to extend the link function (11) to infer a coloured 3D surface from colour images captured from multiple views. Both Ruttle and Kim [36,39] considered that the camera matrices associated with each recorded image are known exactly. In practice however, despite careful calibration, errors occur on the extrinsic camera parameters.

2.4. Remarks & contributions

In this paper, we propose first to extend the link function (11) to deal with data captured with depth cameras, such that shape-from-silhouettes framework [36] becomes a shape-from-depth (SfD) approach (see Section 3). Second, we extend GR^{2}T framework to estimate the extrinsic parameters of the cameras which are nuisance parameters in the context of 3D shape inference: the extrinsic camera parameters are not of interest but they need to be accurately estimated to allow for an accurate shape reconstruction. A calibration step allows us to get intrinsic camera parameters that remain fixed, and it also provides initial guesses for the extrinsic camera parameters for each captured depth image. The extrinsic parameters are then refined to improve 3d shape reconstruction.

When considering sequences of depth images, it is often assumed that only an incremental transformation occurs between successive frames facilitating the recovery of the extrinsic parameters by standard techniques such as ICP [5]. When this is not verified however, ICP performs poorly [24,4]. We propose here to use GR^{2}T to merge the information from several depth images collected from multiple views that can be sparse and with poor overlaps. Depth images are not converted into point clouds and they are all used directly and simultaneously with GR^{2}T. This allows for the uncertainties about the observations, pixel positions and depth values, to be automatically accounted for in the modelling.

3. Shape from depth images with GR^{2}T

Section 3.1 presents the link function F used to estimate the 3D shape from depth images. It is based on the pin-hole camera model. Paragraph 3.2 gives an explicit expression for the average likelihood used as a cost function modelling the surface of the object in the 3D space. In practice, using a turning table, the extrinsic camera parameters can not be known precisely via calibration and we propose to refine them using the depth images (Section 3.4). A mesh is then inferred using an iterative algorithm that explores the surface of the object characterised by a ridge in the cost function (Section 3.5). Section 3.6 provides more details on the computations involved in our framework.

3.1. Link function F

To solve SfD, we define the following random variables:

- $\Theta \in \mathbb{R}^3$ is the 3D spatial latent variable of interest. The cost function proposed here is optimised w.r.t. Θ to extract the shape of the object in view in the depth images.
- $\Psi \in \mathbb{R}^6$ correspond to the extrinsic camera parameters modelled as a nuisance random variable. ψ_1 is the roll component of rotation matrix, ψ_2 is the pitch component of rotation matrix, ψ_3 is the yaw component of rotation matrix and (ψ_4, ψ_5, ψ_6) is the 3D translation vector [40]. The roll, pitch and yaw define a 3D rotation matrix noted $R(\Psi)$. The intrinsic camera parameters, noted f_x, f_y, u_0, v_0 (focal length in horizontal and vertical axis f_x, f_y and the coordinates of the centre pixel (u_0, v_0)), are assumed to be accurately estimated by calibration and they are combined with the extrinsic camera parameters to create the projection matrix $P(\Psi)$:

$$P(\Psi) = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} & \psi_4 \\ R(\Psi) & \psi_5 \\ & \psi_6 \end{array} \right] \tag{12}$$

The coordinate of the centre of the camera $C(\Psi)$ can then be computed with:

$$C(\Psi) = -\begin{bmatrix} \\ R(\Psi) \\ \\ \end{bmatrix}' \begin{bmatrix} \psi_4 \\ \psi_5 \\ \psi_6 \end{bmatrix} \tag{13}$$

- $\mathbf{x} \in \mathbb{R}^3$ is an observed random variable corresponding to the pixel spatial positions (x_1, x_2) and the depth value x_3. For each camera, a set of observations for $\mathbf{x}$ has been collected. These sets are noted $\mathcal{S}_1 = \left\{\mathbf{x}_1^{(i)}\right\}_{i=1,\dots,N_1}$ for the depth image recorded by camera 1, $\mathcal{S}_2 = \left\{\mathbf{x}_2^{(i)}\right\}_{i=1,\dots,N_2}$ for the depth image recorded by camera 2, and so on up to $\mathcal{S}_C = \left\{\mathbf{x}_C^{(i)}\right\}_{i=1,\dots,N_C}$ recorded by camera C.
- $\boldsymbol{\epsilon} \in \mathbb{R}^3$ is the random variable modelling the noise on depth images. Its distribution p_ϵ is assumed normal with mean zero and diagonal covariance matrix of bandwidth $h_1 = h_2 = 1$ for the uncertainty on the pixel positions, and $h_3 = 0.0002$ the uncertainty about the depth values (obtained by calibration see paragraph 3.3.2).

The function $F(\mathbf{x}, \Psi, \Theta)$ that links the random variable $\mathbf{x}$, Θ, Ψ is defined as:

$$F(\mathbf{x},\Psi,\Theta) = \begin{pmatrix} F_1(\mathbf{x},\Psi,\Theta) \\ F_2(\mathbf{x},\Psi,\Theta) \\ F_3(\mathbf{x},\Psi,\Theta) \end{pmatrix} = \begin{pmatrix} x_1 - \frac{\theta_1 P(\Psi)_{11} + \theta_2 P(\Psi)_{12} + \theta_3 P(\Psi)_{13} + P(\Psi)_{14}}{\theta_1 P(\Psi)_{31} + \theta_2 P(\Psi)_{32} + \theta_3 P(\Psi)_{33} + P(\Psi)_{34}} \\ x_2 - \frac{\theta_1 P(\Psi)_{21} + \theta_2 P(\Psi)_{22} + \theta_3 P(\Psi)_{23} + P(\Psi)_{24}}{\theta_1 P(\Psi)_{31} + \theta_2 P(\Psi)_{32} + \theta_3 P(\Psi)_{33} + P(\Psi)_{34}} \\ x_3 - \sqrt{(C(\Psi)_1 - \theta_1)^2 + (C(\Psi)_2 - \theta_2)^2 + (C(\Psi)_3 - \theta_3)^2} \end{pmatrix} \tag{14}$$

and the stochastic equation used in our modelling is:

$$\lambda + F(\mathbf{x}, \Theta, \Psi) = \boldsymbol{\epsilon} \sim p_\epsilon(\boldsymbol{\epsilon}) \tag{15}$$

The variable $\lambda \in \mathbb{R}^3$ is here an auxiliary random variable that is added to help the modelling of the cost function [1] and we are only interested in inferring information about Θ in the case $\lambda = 0$. Note that this modelling links explicitly the observed quantity $\mathbf{x}$ from the cameras with the additive perturbation $\boldsymbol{\epsilon}$. The first two functions (F_1, F_2) link the pixel positions to the latent 3D locations and was used in Ruttle et al.'s modelling to infer shape-from-silhouettes [36,39] (cf. Eq. 11). As an extension to [36,39], the last function F_3 relates the depth values to the latent 3D locations.

3.2. Cost function with GR^{2}T

The link function $F(\mathbf{x}, \Theta, \Psi)$ is depending on additional nuisance parameters Ψ and these camera parameters need to be estimated accurately to recover the information Θ of interest. Assuming independence between Θ and $(\mathbf{x}, \Psi)$, the joint density function $p_{\lambda\Theta\Psi}$ can be computed by:

$$p_{\lambda\Theta}(\lambda, \Theta) = p_\Theta(\Theta) \iint p_\epsilon(\lambda + F(\mathbf{x}, \Theta, \Psi))\ p_{\mathbf{x},\Psi}(\mathbf{x}, \Psi)\ d\mathbf{x}\, d\Psi \tag{16}$$

The joint density function $p_{\mathbf{x},\Psi}(\mathbf{x}, \Psi)$ can be approximated by its empirical estimate using the observations in $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_C$:

$$\hat{p}_{\mathbf{x},\Psi}(\mathbf{x}, \Psi) = \frac{1}{C} \sum_{c=1}^{C} \frac{1}{N_c} \sum_{i=1}^{N_c} \delta(\mathbf{x} - \mathbf{x}_c^{(i)})\ \delta(\Psi - \Psi_c) \tag{17}$$

Using this estimate to compute the integral (16), we get the following estimate of $p_{\lambda\Theta}$:

$$\hat{p}_{\lambda\Theta}(\lambda, \Theta) = p_\Theta(\Theta)\ \underbrace{\frac{1}{C} \sum\nolimits_{c=1}^{C} \frac{1}{N_c} \sum\nolimits_{i=1}^{N_c} p_\epsilon(\lambda + F(\mathbf{x}_c^{(i)}, \Theta, \Psi_c))}_{\hat{p}_{\lambda|\Theta}(\lambda|\Theta)} \tag{18}$$

At $\lambda = 0$, the average likelihood $\hat{p}_{\lambda|\Theta}$ is a function of Θ:

$$\overline{\text{lik}}(\Theta) = \frac{1}{C} \sum_{c=1}^{C} \frac{1}{N_c} \sum_{i=1}^{N_c} p_\epsilon(F(\mathbf{x}_c^{(i)}, \Theta, \Psi_c)) = \frac{1}{C} \sum_{c=1}^{C} \overline{\text{lik}}(\Theta, \Psi_c) \tag{19}$$

Fig. 2 shows horizontal slices (for visualisation purposes) of the average likelihood function computed with noiseless depth images created for a computer generated object (stanford bunny) with perfect camera parameters $\{\Psi_c\}_{c=1,\dots,C}$. Note how the surface of the object translates into a ridge in the cost function. The ears are occluding each other in some depth images, and therefore less pixels have contributed to some areas of their surface (e.g. in between the ears). As a consequence the ridge created by the surface is uneven.

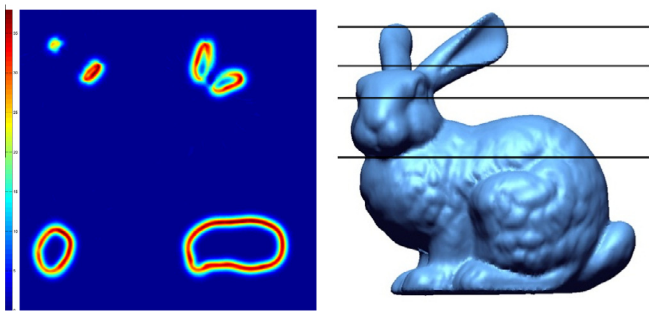

Fig. 2. Slices of the cost function $\overline{\text{lik}}(\Theta)$ (Top: top left corresponding to the ears, top right the middle of the ears, bottom left the head and bottom right the middle of the body) computed for depth images of the Stanford Bunny object (bottom) using 36 depth images generated around the object (the camera parameters are exact).

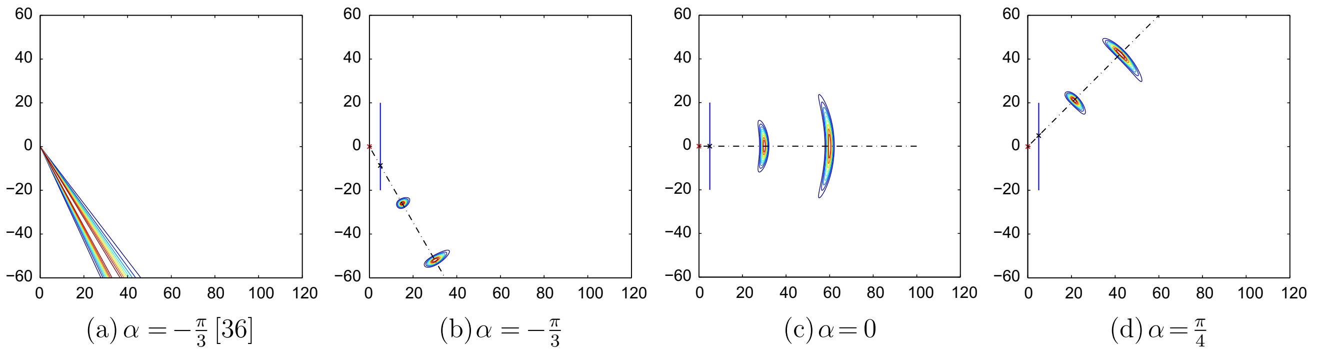

Fig. 3. Kernels of $\overline{\text{lik}}(\theta_1, 0, \theta_3)$: The contour plot (a) shows the kernel cone created with the link functions F_1 and F_2 [36,39] and plot (b) shows the two kernels when considering two points at different depth distance projecting on the same pixel (i.e. two depth values are considered with F_3 in conjunction to the kernel created with F_1 and F_2 shown in (a)). Contour plots (c) and (d) show these two kernels when varying the angle α of projection on the image plane (vertical blue line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 3 shows two kernels $p_\epsilon(\lambda + F(\mathbf{x}_c^{(1)}, \Theta, \Psi_c))$ and $p_\epsilon(\lambda + F(\mathbf{x}_c^{(2)}, \Theta, \Psi_c))$ such that $\mathbf{x}_c^{(1)}$ and $\mathbf{x}_c^{(2)}$ have the same pixel position but two different depth values. When taking into account the noise associated with the pixel position and the depth value, the kernels model their corresponding uncertainty in the 3D space. Note that the shape of the kernels in the 3D space does not correspond to a Normal distribution with an isotropic covariance. While modelling noise on the observations (pixel positions and depth values) with a Normal distribution is a reasonable hypothesis (given prior knowledge about the sensor), this does not translate into a mixture of 3D isotropic Gaussians centered on the 3D point clouds as previously proposed [4].

3.3. Calibration, noise modelling & data capture

Intrinsic parameters and initial estimates for the extrinsic camera parameters for the Kinect camera need first to be computed and this procedure is presented in Section 3.3.1 [41]. In Section 3.3.2, an experiment is carried out to measure the uncertainty of the distances recorded by the depth sensor and this is used for setting the bandwidth h_3 for the error ϵ. Section 3.3.3 indicates how a set of objects have been scanned using the kinect and a turning table to validate our approach. Alternative methodologies that have been recently proposed for depth cameras [42,43] can also be used for modelling accurately the sensor. The same Microsoft Kinect Camera for Xbox 360 (released in 2010) has been used in this study [41].

3.3.1. Calibration

Many methods are available for determining the camera parameters [40]. A common approach is to capture several images of a known geometric pattern such as a planar checkerboard. The knowledge of the relative positions of the corners of the checkerboard in the real world and the corresponding coordinates of the projection of those corners in the image plane can be used to estimate the camera parameters. Using as many as 3D to 2D correspondences as can be available, camera parameters are estimated as well as the radial distortion coefficients. We used the MATLAB toolbox developed by Bouguet [44] for all the camera calibration in this work.

The intrinsic parameters of the RGB camera can be found using standard calibration techniques using a chessboard pattern. For the depth camera this is a bit more difficult. Ideally the same chessboard pattern would be used but the chessboard corners of a standard chessboard would not be visible in the depth image. To overcome this problem a pane of glass is used with a chessboard pattern masked out as seen in Fig. 4. This means the depth camera sees through the unmasked squares and is stopped by the masked squares, resulting in a chessboard pattern seen in the depth image. This is used to calibrate the intrinsic parameters of the depth camera. Viewing this same glass chessboard from both the RGB camera and the depth camera allows a standard stereo calibration to be done to determine the rotation translation between the two cameras. The relationship allows a real world origin to be calculated in the RGB image and from that the extrinsic parameters of the depth camera can be determined, or the depth camera data can be re-projected into the RGB camera space to provide a pixel to pixel match.

3.3.2. Raw depth data conversion & Depth bandwidth selection

The variable x_3 (cf. Eq. (14)) corresponds to a distance and the raw depth values recorded by the depth sensor are actually converted into distances suitable as observations for x_3. A standard generic conversion formula is provided for all Kinect cameras but it does not produce accurate distances. In order to get a more accurate conversion formula, our Kinect camera was pointed at a flat surface (a wall) and repeated depth images were captured from different distances. A calibration chessboard pattern was also placed on the wall so that the distance of the camera from the wall could be determined using standard extrinsic camera calibration. Fig. 5 shows the mean of our recorded raw depth data captured by the Kinect camera w.r.t. the measured real world distance for each capture, as well as the standard error associated with each measurement (in blue). The standard conversion formula provided by Burrus [45] is:

$$x_3 = \frac{1}{d \times (-0.0030711016) + 3.3309495161} \tag{20}$$

This conversion (curve in black in Fig. 5) shows a systematic bias when compared with our measurements. Another conversion formula was proposed by Magnenat [46]:

$$x_3 = 0.1236 \times \tan\left(\frac{d}{2842.5} + 1.1863\right) \tag{21}$$

This is a more accurate conversion formula that fits better our measurements (see curve in red Fig. 5). We propose to use polynomial fitting of power four to give the best least squares fit to link the observed Kinect depth value to the real distance [41]:

$$x_3 = \alpha_0 + \alpha_1\, d + \alpha_2\, d^2 + \alpha_3\, d^3 + \alpha_4\, d^4 \tag{22}$$

Our estimated values for the αs parameters are reported in Appendix A. Note that the abscissa in Fig. 5 is log-transformed to enhance the difference between Magnenat and our proposed polyfit conversion. Our proposed polyfit conversion shown in grey in Fig. 5 is the best fit compared to both Burrus (black) and Magnenat (red).

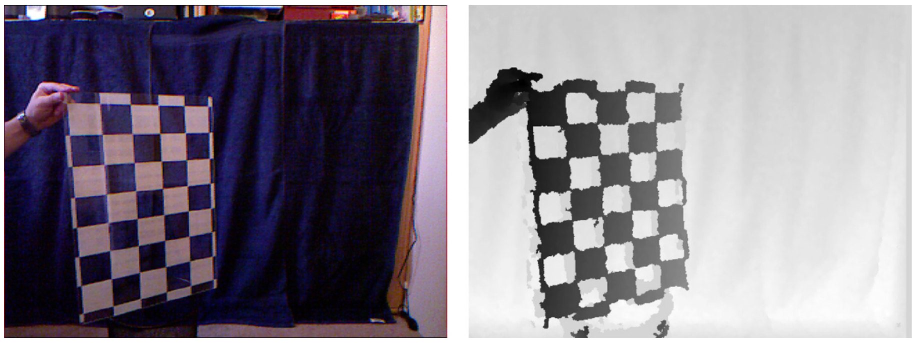

Fig. 4. The glass chessboard used to calibrate the Kinect Camera, both the colour and the depth sensor can see the pattern. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

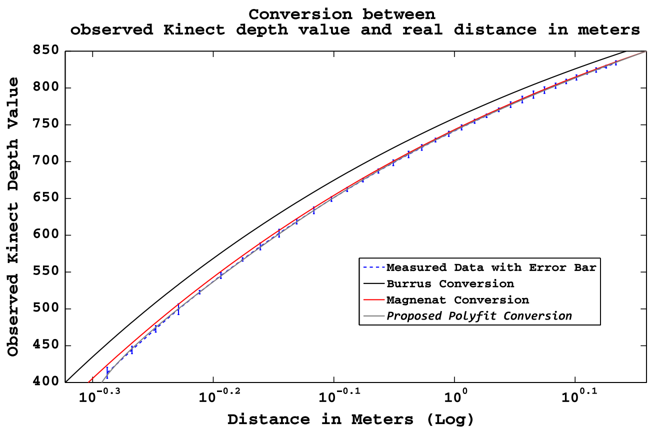

Fig. 5. Conversions of the raw depth data d (y-axis) to the real world (log) distance x_3 (x-axis) [41].

For all the experiments carried out in this work, the object was placed between 0.6 and 0.8 meters away from the camera. In that range, the polynomial curve is nearly linear and the standard deviation of the noise on the depth values is nearly constant. This results in a standard error on the computed distances of $h_3 = 0.0002$ m.

3.3.3. Data capture

A board is placed on top of a large sheet of paper, which is marked every 10 degrees around in a full circle (cf. Fig. 6). The board and the marked sheet of paper allow the board to be placed in the 36 different orientations around a full circle. Initially a set of images is taken with the board in all 36 positions with only a checkerboard pattern on top of the board. This allows all 36 positions to be calibrated, and an initial estimate of the camera parameters to be obtained. Then an object is placed on the board and the board is again placed in all 36 positions, every 10° on a full horizontal circle, with a capture taken at each position. Various objects (cf. Fig. 9) was used from highly complex to relatively plain to test the algorithm. Silhouette images have also been generated using standard background subtraction techniques to segment the foreground and background pixels. The silhouette images were improved using the depth data and when necessary any obvious errors in the segmentation were corrected by hand. No extraordinary measures were taken to get perfect RGB and depth images e.g. no special lights, background or rig was used and the calibration step for the extrinsic parameters is only designed to give good initial guesses in our algorithm.

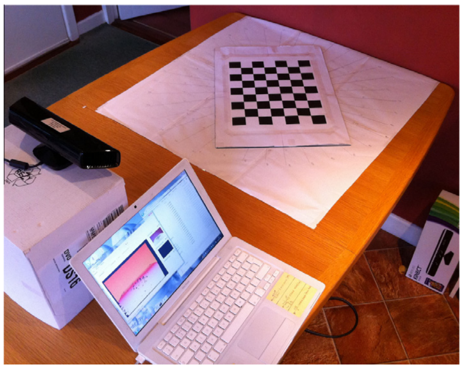

Fig. 6. Setup using the Kinect Camera and rotating platform to generate the real object dataset [41].

3.4. Refining the nuisance parameters Ψ

In practice, even with a careful calibration, the extrinsic camera parameters are not accurate enough and these are necessary to get a good cost function $\overline{\text{lik}}(\Theta)$ for inference of the shape. Choosing camera 1 as a reference camera ($\hat{\Psi}_1$ is available), we want to estimate the parameters $\{\Psi_2, \dots, \Psi_C\}$ such that all average likelihoods overlap well in the 3D space. We formulate the problem as follow [47]:

$$\forall c,\ \hat{\Psi}_c = \arg\max_{\Psi_c} \int \overline{\text{lik}}(\Theta, \hat{\Psi}_1)\ \overline{\text{lik}}(\Theta, \Psi_c)\ d\Theta \tag{23}$$

In practice, to compute this integral, we extract independent samples $\left\{\Theta^{(i)}\right\}_{j=1,\dots,N_1}$ of the reference function $\overline{\text{lik}}(\Theta, \hat{\Psi}_1)$ (we simply choose the 3D point cloud $\left\{\Theta^{(i)}\right\}_{j=1,\dots,N_1}$ computed from the depth image of camera 1). The integral (23) becomes:

$$\hat{\Psi}_c = \arg\max_{\Psi_c} \int \overline{\text{lik}}(\Theta, \hat{\Psi}_1)\ \overline{\text{lik}}(\Theta, \Psi_c)\ d\Theta \simeq \sum_{i=1}^{N_1} \overline{\text{lik}}(\Theta^{(i)}, \Psi_c) \tag{24}$$

The estimate $\hat{\Psi}_c$ is then computed using an iterative gradient algorithm with an initial guess given by the initial calibration. Note that we are only interested in recovering an accurate 3D surface of the object, not its exact position in the 3D world. Hence the parameters $\hat{\Psi}_1$ of the selected reference camera are not important and do not need to be accurate w.r.t. a known origin in the 3D world. What is important is that all other cameras are aligned perfectly with the reference camera.

To maximise overlap between average likelihood functions in this optimisation, while $\overline{\text{lik}}(\Theta, \Psi_2)$ is mapped on $\overline{\text{lik}}(\Theta, \hat{\Psi}_1)$ to estimate $\hat{\Psi}_2$, then $\overline{\text{lik}}(\Theta, \Psi_3)$ is then mapped on $\overline{\text{lik}}(\Theta, \hat{\Psi}_2)$ and so on. This process can lead to a propagation of errors on the estimated camera parameters. but these can be reduced by repeating the process with a different reference camera and by calibrating the cameras in reverse order.

3.5. Surface Explore algorithm

Fig. 7 illustrates what is happening when trying to infer the surface of the object: because each portion of the object is not seen equally from all the views recorded by the kinect, the surface of the object is characterised by a ridge of varying height in the density $\overline{\text{lik}}(\Theta)$. As a consequence, all initial guesses converge to only four points corresponding to the four highest points on the ridge. Instead we want to find a set of vertices well distributed to define a mesh of the surface of the object. For this, we propose to hop on the surface by adding a prior p_Θ, hence making full use of $\hat{p}_{\lambda\Theta}(\lambda = 0, \Theta)$ (Eq. (18)). The approach is explained by Algorithm 1.

Algorithm 1. Surface Explore

```
Require: Initial guess $\Theta^{(0)}$
  Compute $\hat{\Theta}^{(0)} = \arg\max_{\Theta^{(0)}} p_{\lambda|\Theta}(0|\Theta^{(0)})$
  repeat
    $\hat{\Theta}^{(n+1)} = \arg\max_{\Theta^{(n+1)}} p_{\lambda|\Theta}(0|\Theta^{(n+1)}) \times p_{\Theta^{(n+1)}}(\Theta^{(n+1)})$
    $n \leftarrow n+1$
until $p_{\lambda|\Theta}(0|\hat{\Theta}^{(n)}) \leqslant T$ or $\hat{\Theta}^{(n)}$ has looped back near
  $\hat{\Theta}^{(i)},\ 0 \leqslant i < n$.
```

The prior is defined such that it depends on the previous point found on the surface $\hat{\Theta}^{(n)}$:

$$p_{\Theta^{(n+1)}}(\Theta^{(n+1)}) \propto \exp\left(\frac{-(\|\Theta^{(n+1)} - \hat{\Theta}^{(n)}\| - r)^2}{2\gamma^2}\right) \tag{25}$$

For simplicity, the resolution r is fixed to sample points at regular interval on the ridge and the scale is fixed by $\gamma = \frac{r}{3}$. The Surface Explore algorithm is constrained to horizontal 2D slices and it defines a sequence of positions $\left\{\hat{\Theta}^{(n)}\right\}$ on rings in these 2D slices. The distance between successive horizontal slices controls the resolution along the vertical direction. The Surface Explore algorithm is computed with several initial guesses $\Theta^{(0)}$ (using the point clouds) in each slice as several rings of interest needs to be found (cf. rabbit ears in Fig. 2). If the likelihood drops below a threshold T or the estimate is within the minimum distance r of any of a past estimate $\hat{\Theta}^{(i)}$ excluding the immediately previous point, then the chain is stopped. The process is repeated for multiple slices of the object to achieve a full reconstruction. It is possible to repeat the process again, now using slices in a different orientation to improve surface completeness.

3.6. Optimisation and computations

The cost function $\overline{\text{lik}}(\Theta)$ is a kernel density estimate computed using $\left(\sum_{c=1}^{C} N_c\right)$ observations. As more images are collected, the computation at a spatial position Θ becomes more intensive. In practice, we consider only the kernels with the observations (pixels) in the vicinity of the projection of Θ in each camera view. This reduces the number of computations needed to evaluate the average likelihood at Θ. Unlike other global optimal formulations for multiple view reconstruction [48], our approach is therefore suitable for merging many depth maps together without being limited by the memory consumption.

Both optimisation algorithms presented in Sections 3.4 and 3.5 can be computed efficiently using parallel architecture [2]. This approach allows to recover a set of vertices uniformly distributed on the surface of the object that are less noisy than the original point clouds. Some edges are recovered using the sequence of points extracted with Surface Explore. Standard meshing software are then used to compute the missing edges and get the full mesh.

4. Experimental results

In this section we illustrate the different stages of our 3D shape inference system. First depth images are captured using a turning table at regular angular positions in the horizontal plane around the object. The extrinsic camera parameters associated with each camera view is known approximately and Section 4.1 shows how these estimates are improved using the approach presented in paragraph 3.4. Then the quality of the reconstruction is assessed against Shape-from Silhouettes (Section 4.2), and its improvement when more depth images are merged together (Section 4.3). Finally, the kinect reconstructions are compared with alternative reconstructions in Section 4.4.

4.1. Extrinsic camera parameters

Fig. 8 shows the camera positions and orientations in the horizontal plane in our setting using the kinect before and after the camera parameters have been refined. Indeed our setting uses a homemade turning table and the deviation of the true camera

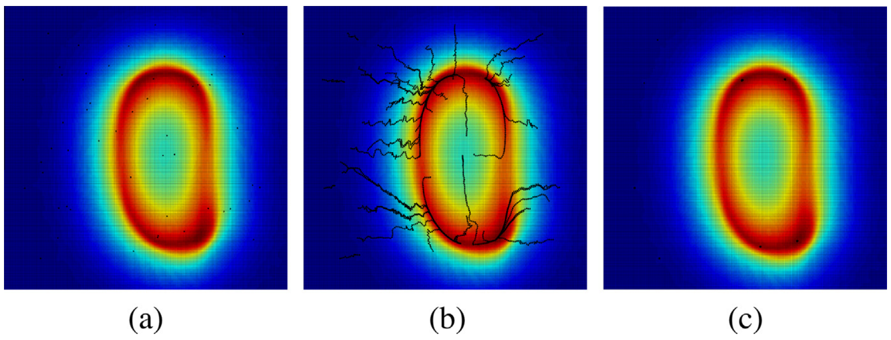

Fig. 7. (a) 50 random points are selected for initial guesses. (b) these points are iteratively updated using Newton's algorithm maximising $\overline{\text{lik}}(\Theta)$ and converge to the nearest maxima (c). Note how the 50 points have converged to only 4 points on the ridge (c).

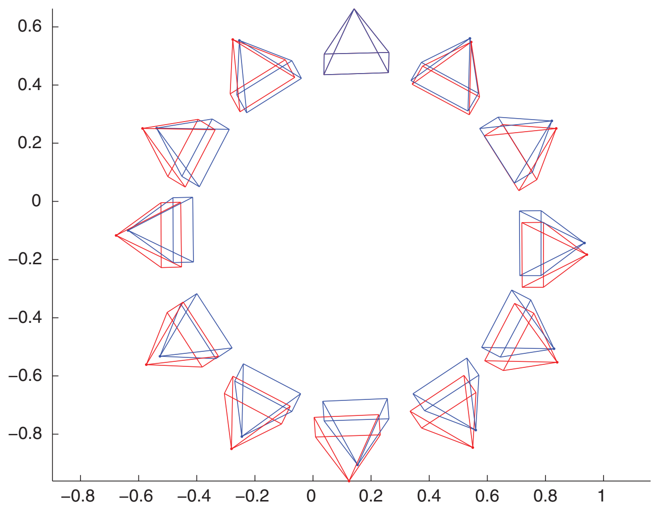

Fig. 8. Camera parameter refinement using a Kinect and a turning table. The red cameras show the original position and orientation of the cameras. The blue cameras show the position and orientation after refinement. Note the reference camera at the top. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

parameters from the original values obtained by calibration is quite important. The axis reports dimension in meters.

4.2. Comparison with shape from silhouettes

Fig. 9 illustrates the 3D meshes obtained for several objects that have been captured with the turning table and a Kinect camera. Reconstructions using silhouettes [36] and depth images are shown for comparison. While silhouette images do not provide information about the concavities of the object, depth images allows these concavities to be well recovered. All six captured objects have dimensions of the same order in the real world e.g. the Gnome is $18 \times 15 \times 32$ and the Lighthouse is $18 \times 16 \times 26$ (in centimeters). Some very small details are not recovered (e.g. roof tiles of the lighthouse), but the inferred meshes are far less noisy and more detailed than the original depth scans. Some artefacts occur on the surfaces when the meshing algorithm fails to recover some of the edges between the inferred vertices.

4.3. Impact of the number of camera views

Fig. 10 shows the objects reconstructed with merging a different number of depth images. Note the selected depth images

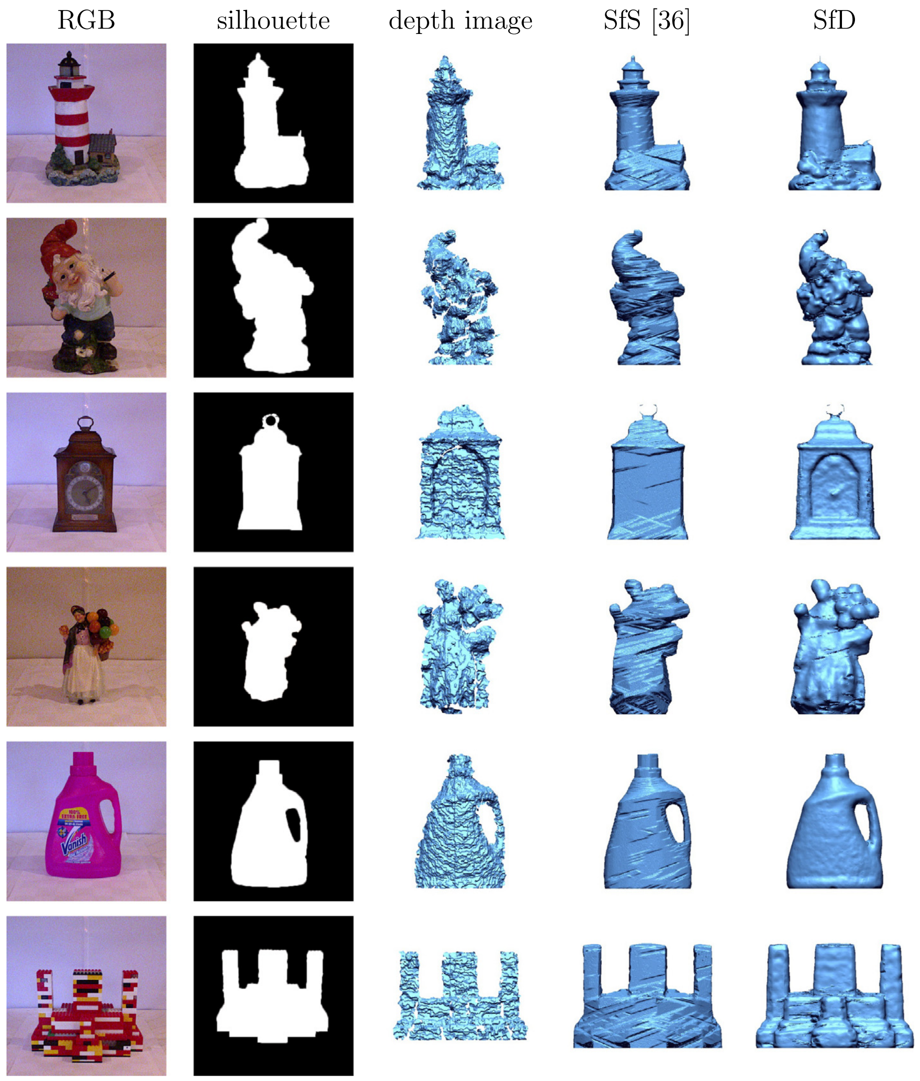

Fig. 9. 3D surface reconstruction using 36 camera views evenly distributed in the horizontal plane. From left to right for one view: RGB image, silhouette, depth image, reconstruction using the 36 silhouettes using Ruttle et al. modelling [36] and reconstruction with depth images using $\overline{\text{lik}}(\Theta)$.

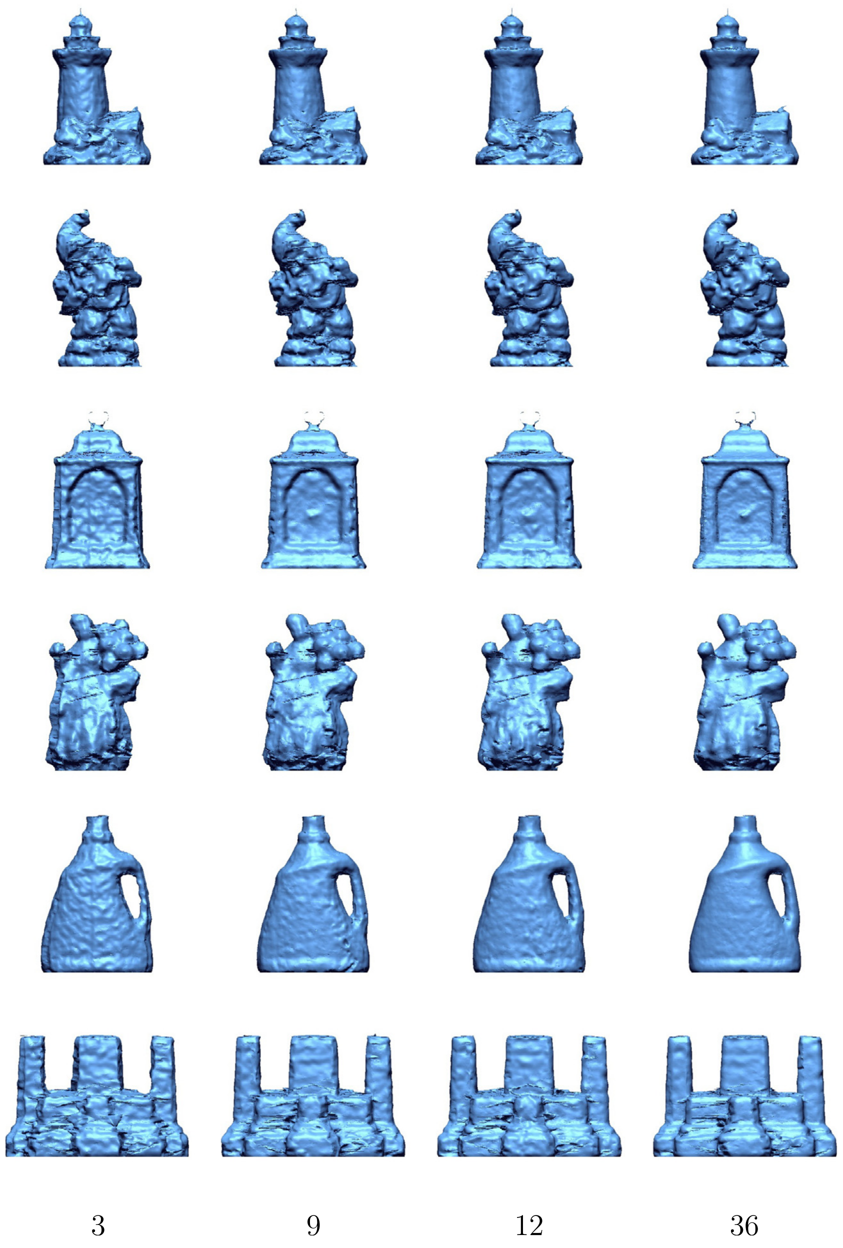

Fig. 10. Objects (cf. Fig. 9) reconstructed with 3, 9, 12, and 36 depth images.

for computing the reconstructions are at regular interval around the objects in each case. As more depth maps are used, the 3D reconstruction improves steadily. This improvement is even more visible for the Vanish Bottle that has a smooth surface.

4.4. Comparison with ground truths

To assess our approach, we compare our reconstructions from 36 depth images (evenly distributed around the object in the horizontal plane) with ground truths. Using the Stanford Bunny mesh (size $10 \times 13 \times 13$ in cm, shown in Fig. 2), we created the 36 depth images with Autodesk 3DS Max (resolution 512×512) from which a reconstruction is computed using our algorithm (cf. Fig. 12, top-left). Our algorithm is performing well apart in two concave areas, between the ears and on the neck, that have not been well captured by the depth images (only few pixels relate with these areas oriented parallel to the horizontal plane). We computed the surface error using the original Stanford Bunny mesh as ground truth (noted G_{Mesh}) and the average error is very small (0.5 mm). Four additional synthetic objects were also tested leading to similar results [41].

When dealing with real objects, we created several ground truths using less noisy sensors. Fig. 11 presents three 3D reconstructions of the Lighthouse captured using different sensors. The first one has been computed using 40 high resolution colour images using an online tool for inferring the 3D shape (`http://www.123dapp.com/catch`) and this is used as a ground truth noted G^{40}_{Colour}. The colour images were recorded both in the horizontal plane around the object and using views from above (following instructions on the website). The second reconstruction has been computed from 4 scans captured with a 3D scanner (Minolta Vivid 700 using its utility software in the '4 scan mode' capturing data

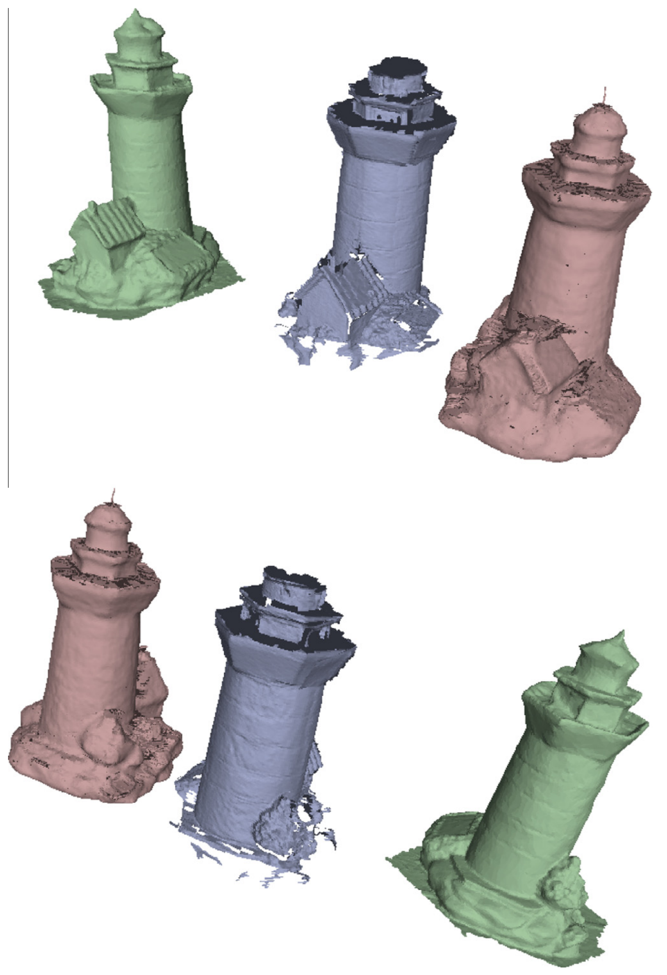

Fig. 11. Visual comparisons of the reconstructions of the Lighthouse: reconstruction from 40 high resolution overlapping colour images (in green) using `http://www.123dapp.com/catch` (G^{40}_{Colour}), reconstruction from 4 scans (in blue) using a 3D scanner Minolta Vivid 700 (G^{4}_{Scan}), and our reconstruction (in pink) from 36 Kinect depth images. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

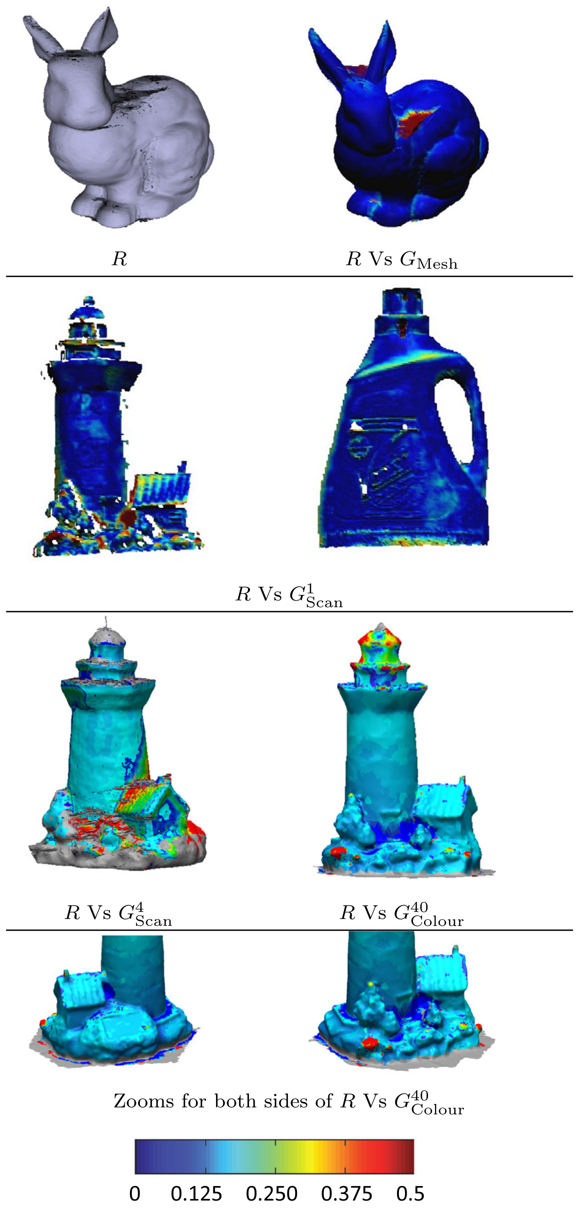
R
R Vs G_{Mesh}
R Vs G^{1}_{Scan}
R Vs G^{4}_{Scan}
R Vs G^{40}_{Colour}
Zooms for both sides of R Vs G^{40}_{Colour}
0
0.125
0.250
0.375
0.5

Fig. 12. Our reconstructions (R) versus ground truth. Top: Stanford Bunny reconstructed with our algorithm (left) and the surface error with ground truth G_{mesh} (right). Bottom: Surface errors for our reconstructions for the Lighthouse and the Vanish bottle computed against several ground truths (the colour scale is in cm). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

every 90° rotation in the horizontal plane using a turning table). This ground truth is noted G^{4}_{Scan} – we also consider one single scan as a ground truth (noted G^{1}_{Scan}). The last reconstruction in Fig. 11 has been computed with our algorithm from the 36 Kinect depth images and it appears smoother with a few artefacts. For instance, tiles on the roof can not be seen in our reconstruction compared with G^{4}_{Scan} and G^{40}_{Colour}. Some errors appear on our reconstruction in areas where little or no data have been captured.

Fig. 12 shows the surface errors computed with the ground truths. Using only one scan as ground truth (G^{1}_{Scan}), the average error on the surface is 1.4 mm for the Lighthouse and 0.8 mm for the Vanish bottle. This compares well with an average error of 1 mm reported by Hernandez et al. reconstructing faces with an algorithm outperforming KinectFusion [11]. The 3D scanner records depth images that are not as noisy as the ones recorded with the Kinect sensor, but it is not a perfect recording system either: notice how the blue colour is not well captured on the label of the Vanish Bottle. The Lighthouse has a more complex shape than the Vanish Bottle, and we compare our reconstruction also against G^{4}_{Scan} and G^{40}_{Colour} (Fig. 12). The average errors are respectively 1.7 mm (R Vs G^{4}_{Scan}) and 2.6 mm (R Vs G^{40}_{Colour}). Prior to computing these errors, all ground truths are registered (rotation and translation) to our Kinect reconstruction R, and scaling using measurements on the real world object Lighhouse has also been applied to the ground truth G^{40}_{Colour} before rigid registration to our Kinect reconstruction R. Note that an average error of 2.5 mm has also been measured (for comparison purposes) between the two ground truths G^{4}_{Scan} and G^{40}_{Colour}.

In general, errors are larger in the areas with very high curvatures compared to flat areas. This limitation could be overcome with adding prior information about the class of object to reconstruct. Our algorithm provides vertices located at regular distances on the surface of the object, and more complex areas would benefit from having locally a higher resolution mesh than flat areas well described with a low resolution mesh.

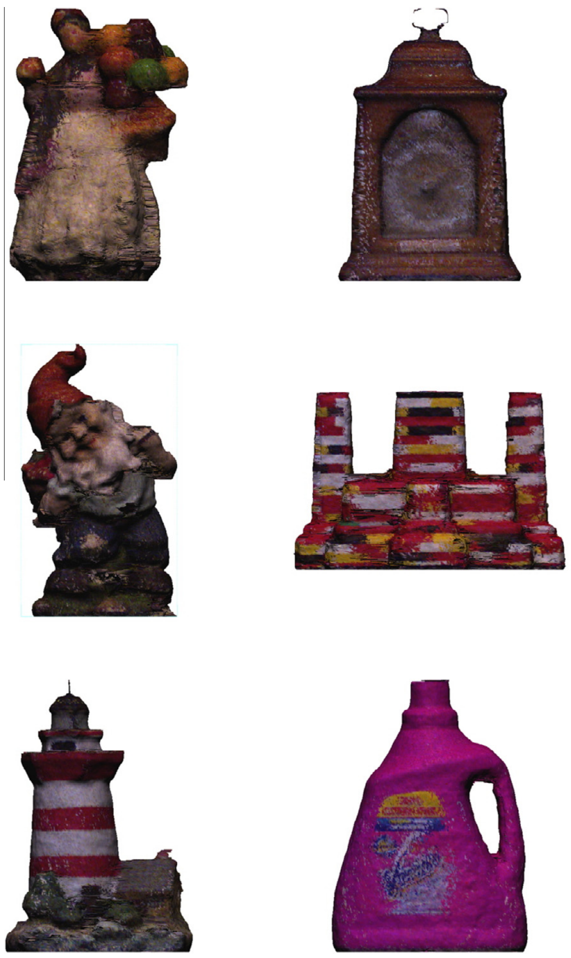

Fig. 13. Projecting colour information. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Conclusion and future works

We have shown that GR^{2}T can be used to merge depth images captured from multiple views, and to infer 3D shape automatically. Without pre-processing stages (e.g. filtering noise) or prior information about the object in view, we have proposed a robust objective function modelling the uncertainty associated with each observation (pixel location and depth). Inference of the surface can be done using standard gradient ascent algorithms that are suitable for parallel processing, and conversion of depth information into 3D point clouds allows to provide excellent initial guesses for these algorithms. Despite the noise on the depth images, the accuracy of the resulting reconstruction using the Kinect sensor compares reasonably with reconstructions using alternative technologies using less noisy sensors.

The proposed approach infers vertices well distributed on the surface of the object and future work will focus on also inferring all edges defining the full mesh. This will be done by extending our modelling to include additional information available in the depth images (e.g. pixel neighborhood information in the depth map). Colour information can also be included in the objective function to estimate a coloured surface [39]. As a simple final stage in Fig. 13, we projected each vertex onto the RGB images that have been captured with the depth images and the nearest RGB value is selected to colour the surface. More advanced methods can also be used to map colour information from images onto a 3D mesh [49].

Appendix A. Polyfit conversion parameters

Optimal values (least squares sense) for Eq. (22):

$$\begin{cases} \alpha_0 = 1.503339336445056 \\ \alpha_1 = -8.588638899009191 \times 10^{-3} \\ \alpha_2 = 2.541111798387088 \times 10^{-5} \\ \alpha_3 = -3.180037690249980 \times 10^{-8} \\ \alpha_4 = 1.606516998323764 \times 10^{-11} \end{cases} \tag{A.1}$$

References

[1] R. Dahyot, J. Ruttle, Generalised relaxed radon transform (GR2T) for robust inference, Pattern Recognit. 46 (3) (2013) 788–794, http://dx.doi.org/10.1016/j.patcog.2012.09.026.

[2] B.V. Srinivasan, Q. Hu, R. Duraiswami, GPUML: graphical processors for speeding up kernel machines, in: Workshop on High Performance Analytics – Algorithms, Implementations, and Applications, SIAM Conference on Data Mining, 2010.

[3] Y. Cui, S. Schuon, C. Derek, S. Thrun, C. Theobalt, 3D shape scanning with a time-of-flight camera, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1173–1180.

[4] Y. Cui, S. Schuon, S. Thrun, D. Stricker, C. Theobalt, Algorithms for 3d shape scanning with a depth camera, IEEE Trans. Pattern Anal. Mach. Intell. 35 (5) (2013) 1039–1050, http://dx.doi.org/10.1109/TPAMI.2012.190.

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, A. Fitzgibbon, Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera, in: ACM Symposium on User Interface Software and Technology, 2011.

[6] T. Whelan, J.B. McDonald, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, Kintinuous: spatially extended KinectFusion, in: RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras, Sydney, Australia, 2012.

[7] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, J. McDonald, Robust real-time visual odometry for dense RGB-D mapping, in: IEEE Intl. Conf. on Robotics and Automation (ICRA), 2013, pp. 5724–5731, http://dx.doi.org/10.1109/ICRA.2013.6631400.

[8] H. Roth, M. Vona, Moving volume kinectfusion, in: British Machine Vision Conference, BMVC, 2012, http://dx.doi.org/10.5244/C.26.112.

[9] M. Zeng, F. Zhao, J. Zheng, X. Liu, A memory-efficient kinectfusion using octree, in: S.-M. Hu, R.R. Martin (Eds.), Computational Visual Media, Lecture Notes in Computer Science, vol. 7633, Springer, Berlin, Heidelberg, 2012.

[10] M. Zeng, F. Zhao, J. Zheng, X. Liu, Octree-based fusion for realtime 3d reconstruction, Graph. Mod. 75 (3) (2013) 126–136, http://dx.doi.org/10.1016/j.gmod.2012.09.002.

[11] M. Hernandez, J. Choi, G. Medioni, Laser scan quality 3-d face modelling using a low cost depth map, in: 20th European Signal Processing Conference (Eusipco), Bucharest, Romania, 2012, pp. 1995–1999.

[12] J. Tong, J. Zhou, L. Liu, Z. Pan, H. Yan, Scanning 3d full human bodies using kinects, IEEE Trans. Visual. Comput. Graphics 18 (4) (2012) 643–650.

[13] Y. Chen, Z.-Q. Cheng, Personalized avatar capture using two kinects in a moment, in: SIGGRAPH Asia 2012 Posters, SA'12, ACM, New York, NY, USA, 2012, p. 3:1, http://dx.doi.org/10.1145/2407156.2407160.

[14] J. Besl, N. McKay, A method for registration of 3-d shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 239–256.

[15] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, Int. J. Comput. Vision 13 (1994) 119–152, http://dx.doi.org/10.1007/BF01427149.

[16] L. Zhang, S.-I. Choi, S.-Y. Park, Robust ICP registration using biunique correspondence, in: International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT) 2011, 2011, pp. 80–85. http://dx.doi.org/10.1109/3DIMPVT.2011.18.

[17] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, in: International Conference on 3-D Digital Imaging and Modeling, 2001.

[18] A. Fitzgibbon, Robust registration of 2d and 3d point sets, Image Vision Comput. 21 (13–14) (2003) 1145–1153.

[19] A. Censi, An ICP variant using a point-to-line metric, in: IEEE International Conference on Robotics and Automation, 2008, pp. 19–25.

[20] D. Chetverikov, D. Stepanov, P. Krsek, Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm, Image Vision Comput. 23 (3) (2005) 299–309.

[21] S.L.G. Sharp, D. Wehe, Maximum-likelihood registration of range images with missing data, IEEE Trans. Pattern Anal. Mach. Intell. 30 (1) (2008) 120–130.

[22] T. Weise, S. Bouaziz, H. Li, M. Pauly, Realtime performance-based facial animation, ACM Trans. Graphics 30 (2011) 1–10, http://dx.doi.org/10.1145/2010324.1964972.

[23] B. Amberg, S. Romdhani, T. Vetter, Optimal step nonrigid ICP algorithms for surface registration, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.

[24] Y. Tsin, T. Kanade, A correlation-based approach to robust point set registration, in: European Conference in Computer Vision ECCV, 2004, pp. 558–569.

[25] B. Jian, B. Vemuri, Robust point set registration using gaussian mixture models, IEEE Trans. Pattern Anal. Mach. Intell. 33 (8) (2011) 1633–1645.
[26] D.W. Scott, Parametric statistical modeling by minimum integrated square error, Technometrics 43 (3) (2001) 274–285. <http://www.jstor.org/stable/1271214>.
[27] K. Denecker, J.V. Overloop, F. Sommen, The general quadratic radon transform, Inverse Prob. 14 (3) (1998) 615.
[28] C.P. Robert, G. Casella, Monte Carlo Statistical Methods, Springer Verlag, 1999.
[29] N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic hough transform, Pattern Recognit. 24 (4) (1991) 303–316.
[30] P. Hough, Methods of means for recognising complex patterns, US Patent 3069654, 1962.
[31] J. Princen, J. Illingworth, J. Kittler, A formal definition of the hough transform: properties and relationships, J. Math. Imaging Vision 1 (2) (1992) 153–168.
[32] C.F. Olson, Constrained hough transform for curve detection, Comput. Vision Image Understanding 73 (3) (1999) 329–345, http://dx.doi.org/10.1006/cviu.1998.0728.
[33] P. Huber, Robust Statistics, John Wiley and Sons, 1981.
[34] S. Mittal, S. Anand, P. Meer, Generalized projection based m-estimator: theory and applications, in: Computer Vision and Pattern Recognition Conference, Colorado Springs, CO, 2011, pp. 2689–2696.
[35] S. Mittal, S. Anand, P. Meer, Generalized projection based *m*-estimator, IEEE Trans. Pattern Anal. Mach. Intell. 34 (12) (2012) 2351–2364, http://dx.doi.org/10.1109/TPAMI.2012.52.
[36] J. Ruttle, M. Manzke, R. Dahyot, Smooth kernel density estimate for multiple view reconstruction, in: proceedings of The Seventh European Conference for Visual Media Production, CVMP 2010, 2010, pp. 74–81. doi:10.1109/CVMP.2010.17.
[37] W.N. Martin, J.K. Aggarwal, Volumetric descriptions of objects from multiple views, IEEE Trans. Pattern Anal. Mach. Intell. 5 (2) (1983) 150–158, http://dx.doi.org/10.1109/TPAMI.1983.4767367.
[38] R. Szeliski, Rapid octree construction from image sequences, CVGIP: Image Understanding 58 (1) (1993) 23–32.
[39] D. Kim, J. Ruttle, R. Dahyot, Bayesian 3d shape from silhouettes, Digital Signal Process. 23 (6) (2013) 1844–1855, http://dx.doi.org/10.1016/j.dsp.2013.06.007.
[40] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004.
[41] J. Ruttle, Statistical framework for multi-sensor fusion and 3d reconstruction (Ph.D. thesis), School of Computer Science and Statistics, Trinity College Dublin, Dublin, Ireland, 2012.
[42] D. Herrera, J. Kannala, J. Heikkilä, Joint depth and color camera calibration with distortion correction, IEEE Trans. Pattern Anal. Mach. Intell. 34 (10) (2012) 2058–2064.
[43] A. Teichman, S. Miller, S. Thrun, Unsupervised intrinsic calibration of depth sensors via slam, in: Robotics Science and Systems, Berlin, Germany, 2013.
[44] J. Bouguet, Camera calibration toolbox for matlab, 2010, <http://www.vision.caltech.edu/bouguetj/calib_doc/index.html>
[45] N. Burrus, Kinect calibration, <http://burrus.name/index.php/Research/KinectCalibration> (retrieved August 2013).
[46] S. Magnenat, <https://groups.google.com/forum/#!topic/openkinect/AxNRhG_TPHg> (retrieved August 2013).
[47] J. Ruttle, C. Arellano, R. Dahyot, Extrinsic camera parameters estimation for shape-from-depths, in: 20th European Signal Processing Conference (Eusipco), Bucharest, Romania, 2012, pp. 1985–1989.
[48] C. Zach, M. Pollefeys, Practical methods for convex multi-view reconstruction, in: European Conference on Computer Vision, 2010.
[49] M. Corsini, M. Dellepiane, F. Ganovelli, R. Gherardi, A. Fusiello, R. Scopigno, Fully automatic registration of image sets on approximate geometry, Int. J. Comput. Vision 102 (1–3) (2013) 91–111, http://dx.doi.org/10.1007/s11263-012-0552-5.