ISSN 2007-9737

Depth Map Denoising and Inpainting Using Object Shape Priors

Andrés Díaz[1], Eduardo Caicedo[1], Lina Paz[2], Pedro Piniés[2]

[1] Universidad del Valle,
Colombia

[2] Intel Corporation,
USA

{andres.a.diaz, eduardo.caicedo}@correounivalle.edu.co,
{paz.linapaz, peternac}@gmail.com

Abstract. We present a system that improves the quality of noisy and incomplete depth maps captured with inexpensive range sensors. We use a model-based approach that measures the discrepancy between a model hypothesis and observed depth data. We represent the model hypothesis as a 3D level-set embedding function and the observed data as a point cloud coming from a segmented region associated to the object of interest. The discrepancy between the model and the observed data defines an objective function, that is minimized to obtain pose, scale and shape. The variation in shape of the object of interest is mapped with Gaussian Process Latent Variable Models GPLVM and the object pose is estimated using Lie algebra. The integration of a synthetic depth map, obtained from the optimal model, and the observed depth map is carried out with variational techniques. As a consequence we work in the observed space (depth space) rather than in a high dimensional volumetric space.

Keywords. Shape prior, 3D level-set embedding function, Levenberg-Marquardt, lie algebra, depth integration, variational techniques, Gaussian process latent variable models, denoising and inpainting.

1 Introduction

High-quality depth information is important for many applications, such as 3D scene reconstruction [35], visual odometry [14], action recognition [4, 26], scene understanding [21, 37], 3D segmentation [9, 11], among others. However, specular or light absorbing materials, out of range regions and occlusions produce corrupted depth maps with missing information. These problems are more common in depth sensors that are manufactured with smaller size and lower power requirements but compromising the quality of the data. The most challenging situation arises when unknown regions become large and irregular, affecting considerably the results of the applications.

To overcome these drawbacks we consider the outstanding performance of variational techniques applied to fusion of color images [17] and propose to use them for merging two sources of depth data; one that is noisy and incomplete, coming from the depth camera and the other one that is high-quality data, coming from an optimal 3D model. This model (shape prior) is the one that best fits to its associated depth data in the scene. In this way, we improve the quality of a noisy and incomplete depth map, especially in the region of the object of interest.

The main steps of our system are the following. First, we estimate the optimal model. Second, we place the model using the optimized pose, shape and scale, and read the depth buffer from the current camera pose. Third, we integrate the two depth maps using variational techniques. These steps are intended to serve as a pre-processing stage for the aforementioned applications, specially for 3D scene reconstruction.

We can outline three main contributions: 1. We propose an approach to enhance depth maps by integrating depth measurements and shape prior

data using a novel formulation of an objective function with a variational approach; 2. The object pose optimization is solved using Lie algebra $se(3)$ instead of the group of rigid transformations in 3D space, SE(3); 3. We quantify the accuracy of pose, shape and scale optimizations. Moreover, we compute the completeness and accuracy of inpainted and denoised depth maps.

This work is structured as follows. In section II similar works are presented. In section III the main processes carried out by the system are described: model alignment, truncated signed distance function TSDF estimation, model compression, dimension reduction, estimation of the optimal pose and scale, search of the optimal latent variable, and integration of the optimal model with a novel variational method. In section IV experiments for optimizing pose, scale and shape with synthetic and real data are carried out. Moreover, experiments for quantifying the accuracy of the enhanced depth maps are described. Finally, the conclusions are presented in section V.

2 Related Work

Techniques for depth recovery include morphological filters [32], Laplace filters [23, 33], Markov Random Fields [27], multilateral filters [13, 22], non-linear diffusion and variational frameworks [7, 10, 28] and learning-based methods [12, 19, 34]. Our system presents a novel approach motivated by outstanding works in object shape priors, scene shape priors and variational techniques. Next, we describe some details about these systems and we compare them with ours.

The system [30] is the first one that back projects depth images and evaluates the resulting point cloud in a 3D level-set embedding function that represents an object model implicitly, instead of projecting the model to the image plane and measuring the discrepancy between expected image cues and the observed ones. Moreover, no point correspondence is required, unlike Iterative Closest Point ICP, since the alignment consist in evaluating the closeness of the points to the zero-level of the embedding function. The authors in [18] remove point cloud artifacts like noisy points, missing data and outliers using a learned shape prior of an object of interest.

They use the discrete cosine transform DCT for compressing the SDF values and GPLVM for dimension reduction.

The system [1] learns a semantic prior comprised of a mean shape for a category (common aspects in shape of a category) and a set of weighted anchor points for instances of the category (specific details). The augmented reconstruction consists in matching anchor points (HOG features), warping by anchor points (an extension of thin-plate spline transformations [31]) and refinement. The system [6] combines shape priors and live dense reconstruction using a monocular camera. Photo-consistency and a variational approach are used for building depth maps [25] that are fused for creating a dense reconstruction, PTAM [16] for tracking the camera, a part-based object detector [8], and GPLVM to represent shape priors. The energy function, besides depending on the evaluation of the 3D points on the embedding function like [30] and [18], depends on the matching between the 2D object segmentation, defined by the foreground and background, and the projected 3D SDF.

The system [29] estimates depth maps using a monocular camera in workspaces with large plain structures like floors, walls or ceilings. Good depth data is propagated to an interior pixel (inpainting) from the closest valid pixels along the main 8 star directions by using a non-local high-order regularisation term, in a variational approach, that favours solutions with affine surfaces (prior). The energy is minimized in straight way with the primal-dual algorithm. The system [5] includes a term, besides the data term and regularization term, that depends on three scene priors: planarity of homogeneous color regions (using superpixels), the repeating geometry primitives of the scene (data-driven 3D primitives learned from RGBD data), and the Manhattan structure of indoor rooms (layout estimation and classification of box-like structures).

Our system is not based on scene shape priors like [5, 29] but on object shape priors like [1, 6, 18, 30]. However, it does not uses neither anchor points like [1] nor a depth and intensity-based

energy function like [6], but GPLVM like [6, 18] and a depth-based energy function like [18, 30].

We use a variational technique, like is done in [5, 29] for merging scene shape priors, but we exploit the idea of integrating two depth maps, one coming from the sensor and the other one coming from the object shape prior, in a similar way as is done for aerial color images in [17]. The latter system transforms a large point cloud into a common orthographic aerial view. This noisy color image with undefined areas due to occlusions and non-stationary objects is subjected to denoising and inpainting by integrating redundant observations of the same scene using variational techniques. It uses the primal-dual algorithm for minimizing the proposed energy. Finally, our approach implies that, in a future work, the enhanced depth maps will be fused instead of including directly the shape prior as a volumetric structure into the general reconstruction of the scene like is done in [6].

3 Methodology

The main pipeline of the system is shown in fig. 1. The set of reference models (predefined database of cars) are aligned using ICP for getting models with the same scale, translation and rotation. A volumetric grid with TSDF values is computed and a compression is done using DCT, for maintaining just the low-frequency components of the reference models. A continuous, nonlinear, probabilistic and lower dimensional latent shape space that captures the prior knowledge on the 3D shapes that an object can take is found with Gaussian Process Latent Variable Models GPLVM.

Scaled conjugate gradient SCG is employed for leaning the mapping between the latent variables and the low-frequency components (coefficients). Next, the optimal pose, scale and shape are computed and the coefficients associated to the latent variable that best fits to depth data are estimated. The 3D level-set embedding function encoded in the coefficients is computed with the inverse discrete cosine transform IDCT. The optimal model is used for creating a synthetic depth map by reading the depth buffer of the explicitly represented model seen from the estimated camera pose.

The synthetic depth map is merged with the depth data coming from the sensor using a novel variational formulation. In the next sections, the main modules of the system are explained.

3.1 Alignment and TSDF Estimation of the 3D Models

We downloaded 49 models of cars from Turbosquid, a vast online catalog of 3D models. We align each model w.r.t a base model (see fig. 2(h), model chosen arbitrarily) using just the vertices of the triangular mesh, read from an .obj file. Initially, the point cloud of the base car is translated from their center of gravity to the origin. Then, it is scaled for fitting it to the volumetric grid, and the average distance of each point to the new origin is computed (see [3] for more details):

$$s_{\text{base}} = \frac{\sum_{i=1}^{N_{\text{base}}} \sqrt{X_i^2 + Y_i^2 + Z_i^2}}{N_{\text{base}}}, \tag{1}$$

where N_{base} is the number of points of the base model. The same process is carried out (translation and average distance computation) for the point cloud of the model i for getting s_i. The scale is computed as:

$$s = \frac{s_{\text{base}}}{s_i}. \tag{2}$$

This scale is applied to model i in order to get models with the same size. Then, the models are manually oriented in such a way that their frontal sides point to the same direction, standing over a parallel supporting plane. This initial alignment is refined using ICP, implemented in CloudCompare.

Once the models have been aligned, the model i is loaded in OpenGL using the vertices and facets (just geometry data), creating a continuous surface. The virtual camera is moved in such a way that circular and vertical scanning at different longitudes (changes in azimuth and elevation of 10°) are performed.

The optical axis of the virtual camera is always pointing to the origin that coincides with the center of gravity of the loaded model. Depth images, obtained by reading the depth buffer of OpenGL,

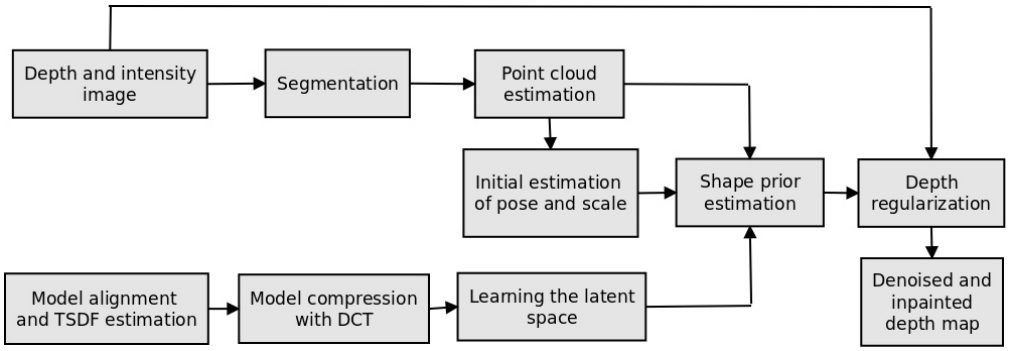


Fig. 1. Main pipeline of the proposed system. The final result is a denoised and inpainted depth map

are fused into a volumetric structure, storing in each voxel a TSDF value. We have $N_m = 49$ 3D models of cars with different shapes (see 9 of the 49 models in fig. 2). Each model is a 3D level-set embedding function defined by a volumetric grid of $N_g \times N_g \times N_g$ voxels, with $N_g = 128$. The matrix $M \in R^{N_m \times N_g^3}$ stores these models as row vectors. However, searching for a model that best fits to data in this high dimensional space is too complex, so we compress the data using DCT. Then, we find a lower dimensional space and the mapping parameters for passing from this latent space to the space of DCT coefficients:

3.2 Compression of the Embedding Function

The DCT transforms a time domain signal to a frequency domine signal. It is used for compression of audio (e.g. MP3) and images (e.g. JPEG) where small high-frequency components can be discarded. We use here this technique for compressing the TSDF values of the volumetric grid. Let $\phi(u, v, w) \in \mathbb{R}^{N_g \times N_g \times N_g}$ be the volumetric grid that stores the TSDF values of a 3D model. The two-dimensional DCT for slices ϕ_i in direction u of ϕ, it means, $c_i(v, w) = DCT(\phi_i(v, w))$ is given by:

$$c_i(v,w) = \alpha(v,w) \sum_{y=0}^{N_g-1} cos\left(\frac{\pi(2y+1)v}{2N_g}\right) \sum_{z=0}^{N_g-1} \phi_i(y,z) cos\left(\frac{\pi(2z+1)w}{2N_g}\right), \quad (3)$$

where $\alpha(v, w) = \alpha_v(v) * \alpha_w(w)$, with:

$$\alpha_v(v) = \alpha_w(w) = \begin{cases} \frac{1}{\sqrt{N_g}} & \text{if } v, w = 0, \\ \frac{2}{\sqrt{N_g}} & \text{if } v, w \neq 0. \end{cases} \quad (4)$$

Since we have $N_g = 128$ slices, the size of ϕ_i is $(N_g \times N_g)$. Equation (3) can be expressed as matrix and array multiplications:

$$c_i = \alpha. * (A\phi_i A^T), \quad (5)$$

where $A \in \mathbb{R}^{N_c \times N_g}$ with N_c the number of DCT coefficients (desired low-frequency components). Each row of A stores the cosine $cos\left(\frac{\pi(2x+1)u}{2N_g}\right)$ for a common u, with $u = 1, 2, ..., N_c$.

We extend it to the three-dimensional DCT and apply the property of separability (see [15] for more details), computing the 3D-DCT in two steps with 2D transformations over the slices for each i along dimension u and a final 1D operation. Figure 3 shows the 3D models obtained using 10, 25, 35, 50, 75 and 100 coefficients of the model visualized in 2(b). Here we can see that the information about the general shape is stored in the low-frequency components and the fine details are store in the high-frequency components.

Note that for 50, 75 and 100 coefficients there are not significant changes in appearance. We consider that $25 \times 25 \times 25$ coefficients are enough for our purposes (like in [18]).

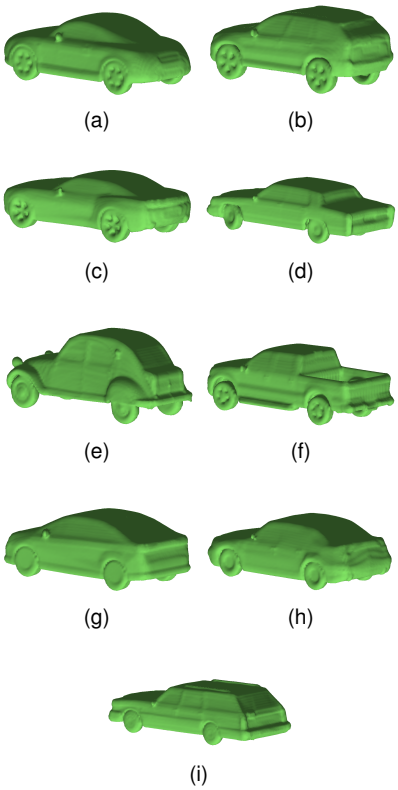


Fig. 2. 9 of the 49 3D models of instances of the class car (object of interest), used for estimating the latent space

3.3 Learning the Latent Space

The Latent variable Model LVM is used for dimensionality reduction, to capture the shape variance as low dimensional latent shape spaces, such that the resulting latent variables have less dimensions than the original observed data. Let $Y \in R^{N_m \times N_c^3}$ be the matrix that stores in row $\boldsymbol{Y_j} \in R^{1 \times N_c^3}$ the $N_c = 25$ low-frequency coefficients in each dimension of the volumetric grid, and for each model j, with $j = 1:1:N_m$.

Let $\chi \in R^{N_m \times N_x}$, with $N_c^3 \gg N_x$, be the matrix that stores in each row the latent variable $\boldsymbol{\chi_j} \in R^{1 \times N_x}$ that represents the coefficients of model j. Now, we learn from the coefficients the parameters $(\theta_1, \theta_2, \theta_3, \theta_4)$ that map any latent variable $\boldsymbol{\chi_j}$ to the corresponding row vector of coefficients $\boldsymbol{Y_j}$, where its relationship is non-linear.

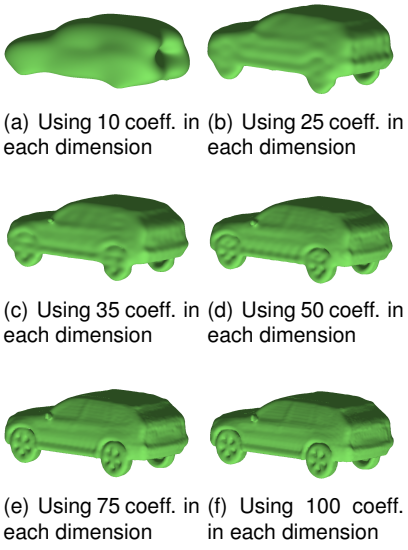


Fig. 3. 3D model of fig. 2(b) using different number of coefficients in each dimension (degree of compression)

Since the mapping $\boldsymbol{\chi_j} \rightarrow \boldsymbol{Y_j}$ is modelled using a Gaussian process, we can define areas of high probability, it means, areas where there are high certainty of getting a valid shape. Moreover, this technique allows to map coefficients from a continuous latent space (no just the ones used for learning), creating a continuous search of the latent variable that best fits to data. We define the vector $\boldsymbol{W} \in R^{1 \times (N_m * N_x + 4)}$ as:

$$\boldsymbol{W} = \{\chi_{1,1} \quad \cdots \quad \chi_{1,N_m} \quad \cdots \quad \chi_{N_x,1} \quad \cdots \quad \chi_{N_x,N_m} \quad \theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4\}. \tag{6}$$

It is the vector of parameters that solves the following optimization problem:

$$\min_{\boldsymbol{W}} E(\boldsymbol{W}) = \frac{N_m}{2} ln(2\pi) + \frac{1}{2} ln|K| + \frac{1}{2} tr(K^{-1}\mathcal{S}), \tag{7}$$

where N_m is the number of models, K is the covariance matrix which is estimated with a Kernel function, and $\mathcal{S} = \mathcal{D}^{-1} Y Y^T$, with $\mathcal{D}$ the number of coefficients for a model (N_c^3). The components of K, denoted as $k_{ij} = K(\chi_i, \chi_j)$, for a latent variable with two dimensions $(N_x = 2)$ and a Radial Basis

Function kernel, are:

$$k_{ij} = \theta_1 exp\left(\frac{-\theta_2}{2}((\chi_{1,i} - \chi_{1,j})^2 + (\chi_{2,i} - \chi_{2,j})^2)\right) + \theta_3 + \delta_{ij}\theta_4, \quad (8)$$

where δ_{ij} is the Kronecker delta function. We initialize the latent variables of the vector of parameters $\boldsymbol{W}$ with the estimation got with Dual Probabilistic PCA. Then, we use the scaled conjugate gradient SCG algorithm for refining the initial estimation. It combines the model trust region approach, known from the Levenberg-Marquardt algorithm, with the conjugate gradient approach.

The pseudocode and more details about SCG can be found in [24]. The parameters for the kernel are set according to [20]; $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = exp(-1)$, and $\theta_4 = exp(-1)$. With these values we can set the initial kernel and continue the iterative process until convergence.

The resulting latent space is employed in section 4.1 for shape optimization. Besides the mean value, each point $\boldsymbol{\chi_p}$ in the latent space has a variance which is computed as:

$$\sigma^2 = K(\boldsymbol{\chi_p}, \boldsymbol{\chi_p}) - K(\chi, \boldsymbol{\chi_p})K^{-1}K(\chi, \boldsymbol{\chi_p})^T. \quad (9)$$

The parameters for mapping converged to $\theta_1 = 10.4195$, $\theta_2 = 1.6784$, $\theta_3 = 84.5378$, and $\theta_4 = 1.1679$.

3.4 Shape Prior Estimation. Shape Optimization

We define an energy that depends on the sum of squared residuals:

$$E(\boldsymbol{\Phi}) = \frac{1}{2}\sum(\boldsymbol{\Phi})^2, \quad (10)$$

where the residual vector $\boldsymbol{\Phi}$ is defined with the German-McClure function, which is robust to outliers:

$$\boldsymbol{\Phi} = \frac{\phi(\boldsymbol{X_q})^2}{\phi(\boldsymbol{X_q})^2 + \sigma}, \quad (11)$$

where σ is a constant parameter, $\boldsymbol{X_q} = [X_q \;\; Y_q \;\; Z_q]$ is one of the N_p 3D points that are evaluated (3D interpolation) in the embedding function ϕ. Points that are back-projected outside the volumetric grid are assigned a large value. The derivative of the energy w.r.t. the latent variable $\boldsymbol{\chi_p}$ is computed through the chain rule:

$$\frac{\delta E}{\delta \boldsymbol{\chi_p}} = \frac{\delta E}{\delta \boldsymbol{\Phi}} * \frac{\delta \boldsymbol{\Phi}}{\delta \boldsymbol{\chi_p}}. \quad (12)$$

The first term on the right is the residual vector $\boldsymbol{\Phi}$. The variations of the residual vector $\boldsymbol{\Phi}$ due to changes in the latent variable $\boldsymbol{\chi_p}$ corresponds to the Jacobian:

$$J = \frac{\delta \boldsymbol{\Phi}}{\delta \boldsymbol{\chi_p}} = \frac{\delta \boldsymbol{\Phi}}{\delta \phi} * \frac{\delta \phi}{\delta \boldsymbol{\chi_p}}, \quad (13)$$

where:

$$\frac{\delta \boldsymbol{\Phi}}{\delta \phi} = \frac{2\sigma\phi(\boldsymbol{X_q})}{(\phi(\boldsymbol{X_q})^2 + \sigma)^2}. \quad (14)$$

For defining the second term on the right of eq. 13 we express the embedding function ϕ in terms of the latent variable:

$$\phi = \text{IDCT}(\boldsymbol{\nu}(\boldsymbol{\chi_p})). \quad (15)$$

The function IDCT$(\cdot)$ is the inverse discrete cosine transform applied to the mean $\boldsymbol{\nu}$ of the coefficients associated with the latent variable $\boldsymbol{\chi_p}$ through a Gaussian process:

$$\boldsymbol{\nu}(\boldsymbol{\chi_p}) = K(\chi, \boldsymbol{\chi_p})K^{-1}Y. \quad (16)$$

Considering that the derivative of the IDCT of a variable is the IDCT of the derivative of the variable, we have:

$$\frac{\delta \phi}{\delta \boldsymbol{\chi_p}} = \frac{\delta(\text{IDCT}(\boldsymbol{\nu}))}{\delta \boldsymbol{\chi_p}} = \text{IDCT}\left(\frac{\delta \boldsymbol{\nu}}{\delta \boldsymbol{\chi_p}}\right), \quad (17)$$

where:

$$\frac{\delta \boldsymbol{\nu}}{\delta \boldsymbol{\chi_p}} = \frac{\delta K(\chi, \boldsymbol{\chi_p})}{\delta \boldsymbol{\chi_p}}K^{-1}Y, \quad (18)$$

$K(\chi, \boldsymbol{\chi_p}) \in R^{1 \times N_m}$ is the row kernel for the latent variable $\boldsymbol{\chi_p}$. It depends on the squared euclidean distance between the latent variable $\boldsymbol{\chi_p}$ and each one of the N_m latent variables used for learning the parameters for mapping. Each element of the row kernel for $\boldsymbol{\chi_p}$, with $N_x = 2$, is found using eq. (8), replacing $\boldsymbol{\chi_j}$ by $\boldsymbol{\chi_p}$. The derivative of $K(\chi, \boldsymbol{\chi_p})$ w.r.t each component of the latent variable is computed in a straight way. The latent variable is updated by

addition, $\boldsymbol{\chi_p^{k+1}} = \delta\boldsymbol{\chi_p} + \boldsymbol{\chi_p^k}$, where $\delta\boldsymbol{\chi_p}$ is found with Levenberg-Marquardt:

$$\delta\boldsymbol{\chi_p} = -(J^T J + \alpha \text{diag}(J^T J) I)(J^T \boldsymbol{\Phi}). \quad (19)$$

We start with $\alpha = 1e-4$. After each iteration, the new energy is compared with the previous one. If the energy has decreased, α is reduced by a factor of 10 and the parameters are updated. If the energy has increased, the parameters are not updated and α is increased by a factor of 10.

3.5 Shape Prior Estimation: Pose and Scale Optimization

We now differentiate the energy of eq. (10) w.r.t. the pose and scale, optimizing them in separated and alternating way. The pose is minimally parametrized with a vector $\boldsymbol{\zeta} = [w_x, w_y, w_z, v_x, v_y, v_z]$, where the first three elements define the axis of rotation and its norm defines the magnitude of the rotation. The remaining elements define the translation after carrying out the rotation. The derivative of the energy w.r.t. $\boldsymbol{\zeta}$ is:

$$\frac{\delta E}{\delta \boldsymbol{\zeta}} = \frac{\delta E}{\delta \boldsymbol{\Phi}} * \frac{\delta \boldsymbol{\Phi}}{\delta \boldsymbol{\zeta}}. \quad (20)$$

Again, the first term on the right is the residual $\boldsymbol{\Phi}$. The variations of the residual vector $\boldsymbol{\Phi}$ due to changes in pose $\boldsymbol{\zeta}$ correspond to the Jacobian for this problem:

$$J = \frac{\delta \boldsymbol{\Phi}}{\delta \boldsymbol{\zeta}} = \frac{\delta \boldsymbol{\Phi}}{\delta \phi} * \frac{\delta \phi}{\delta \boldsymbol{\zeta}}. \quad (21)$$

The first term on the right was defined in equation (14). The second term on the right is:

$$\frac{\delta \phi}{\delta \boldsymbol{\zeta}} = \frac{\delta \phi}{\delta \boldsymbol{X_q}} * \frac{\delta \boldsymbol{X_q}}{\delta \boldsymbol{\zeta}}, \quad (22)$$

$\frac{\delta \phi}{\delta \boldsymbol{X_q}}$ is computed numerically, using the central difference formula:

$$\frac{\delta f(x)}{\delta x} = \frac{f(x+h) - f(x-h)}{2h}. \quad (23)$$

Since a point $\boldsymbol{X_q} = [X_q \quad Y_q \quad Z_q]$ (coordinates of voxels referenced to the initial coordinate system q), we get:

$$\frac{\delta \phi}{\delta \boldsymbol{X_q}} = \left[\frac{\delta \phi}{\delta X_q} \quad \frac{\delta \phi}{\delta Y_q} \quad \frac{\delta \phi}{\delta Z_q} \quad 1 \right], \quad (24)$$

with $h = 10^{-8}$ for all the components. For defining $\frac{\delta \boldsymbol{X_q}}{\delta \boldsymbol{\zeta}}$ we first express a point $\boldsymbol{X_q}$ in terms of $\boldsymbol{X_w}$. We have that:

$$\boldsymbol{X_w} = M_{wo} S M_{oq} \boldsymbol{X_q}, \quad (25)$$

M_{wo} is the transformation matrix between the object and the world system. M_{oq} is the transformation matrix between the initial and the object system. It corresponds to a translation to the center of the volumetric grid. Figure 4 shows the initial coordinate system q, the object system o, and the world system w. Since the point cloud is referenced to the world coordinate system, the 3D model is also referenced to this system and then, aligned with the point cloud.

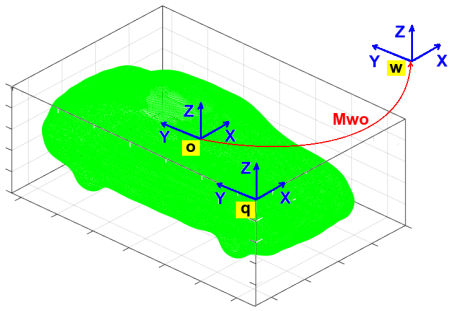


Fig. 4. Initial coordinate systme q, object coordinate system o and world coordinate system w

S scales the points $\boldsymbol{X_o}$ and is defined as:

$$S = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (26)$$

Solving for the point $\boldsymbol{X_q}$, we get:

$$\boldsymbol{X_q} = (M_{oq})^{-1} S^{-1} M_{ow} \boldsymbol{X_w}\ , \quad (27)$$

The transformation matrix M_{ow} is updated as:

$$M_{ow}^{k+1} = \Delta M_{ow} M_{ow}^{k}, \tag{28}$$

where the incremental change in the transformation is expressed using the exponential mapping and the lie algebra:

$$\Delta M_{ow} = exp([\Delta\boldsymbol{\zeta}]_x). \tag{29}$$

The operator $[\cdot]_x$ transform a six-elements vector to the skew matrix:

$$[\Delta\boldsymbol{\zeta}]_x = \begin{bmatrix} 0 & -w_z & w_y & v_x \\ w_z & 0 & -w_x & v_y \\ -w_y & w_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{30}$$

$\Delta\boldsymbol{\zeta}$ is computed in similar way as $\delta\boldsymbol{\chi_p}$ in equation (19). Replacing M_{ow}^{k+1} of eq. (28) and ΔM_{ow} of eq. (29) into eq. (27), we get:

$$\boldsymbol{X_q} = (M_{oq})^{-1} S^{-1} exp([\Delta\boldsymbol{\zeta}]_x) M_{ow}^{k} \boldsymbol{X_w}\ , \tag{31}$$

$M_{ow}^{k}\boldsymbol{X_w}$ is the 3D point $\boldsymbol{X_o^k}$ previous to update the transformation matrix M_{ow}. Now, $\frac{\delta \boldsymbol{X_q}}{\delta\boldsymbol{\zeta}}$, considering the scale constant, is:

$$\frac{\delta \boldsymbol{X_q}}{\delta\boldsymbol{\zeta}} = M_{oq}^{-1} S^{-1} \frac{\delta(exp([\delta\boldsymbol{\zeta}]_x))}{\delta\boldsymbol{\zeta}} \boldsymbol{X_o^k}, \tag{32}$$

where the derivative of the exponential w.r.t the first element of the vector $\boldsymbol{\zeta}$ is:

$$\frac{\delta(exp([\delta\boldsymbol{\zeta}]_x))}{\delta w_x} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{33}$$

This derivative is a generator for rotations in x-axis in the lie algebra. The same process is carried out for the remaining five parameters. Multiplying by the homogeneous form of $\boldsymbol{X_o}$ and organizing, we get a 4×6 matrix.

$$\frac{\delta(exp([\delta\boldsymbol{\zeta}]_x))}{\delta\boldsymbol{\zeta}} \boldsymbol{X_o^k} = \begin{bmatrix} 0 & X_{o_z} & -X_{o_y} & 1 & 0 & 0 \\ -X_{o_z} & 0 & X_{o_x} & 0 & 1 & 0 \\ X_{o_y} & -X_{o_x} & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{34}$$

Following a similar process for scale optimization, we consider the parameters of pose constant and derive $\boldsymbol{X_q}$ from eq. (27) w.r.t the scale parameter:

$$\frac{\delta \boldsymbol{X_q}}{\delta s} = -M_{oq}^{-1} S^{-1} \frac{\delta S}{\delta s} S^{-1} \boldsymbol{X_o^k}, \tag{35}$$

where $\frac{\delta S}{\delta s}$ is easily computed.

3.6 Inpainting and Denoising

TV-based methods are well suited for tasks like depth data integration, as was probed in [36]. In our context, an energy defined by a TV-based regularization together with a data term that measures the discrepancy between two sources of data (the depths coming from the model $\boldsymbol{D_m}(\boldsymbol{\mu})$ and from the sensor $\boldsymbol{D_s}(\boldsymbol{\mu})$) and the sought solution $\boldsymbol{D_f}(\boldsymbol{\mu})$, is implemented:

$$\min_{\boldsymbol{D_f}} E(\boldsymbol{D_f}(\boldsymbol{\mu})) = \int (\underbrace{\|\nabla \boldsymbol{D_f}(\boldsymbol{\mu})\|_1}_{\text{regularizer term}} + \underbrace{\lambda \sum_{k=1}^{2} w_k(\boldsymbol{\mu}) \|\boldsymbol{D_f}(\boldsymbol{\mu}) - \boldsymbol{D_k}(\boldsymbol{\mu})\|_\epsilon) d\boldsymbol{\mu}}_{\text{data term}}, \tag{36}$$

where λ defines the balance between the regularizer term and the data term, $\boldsymbol{D_1}(\boldsymbol{\mu}) = \boldsymbol{D_s}(\boldsymbol{\mu})$, and $\boldsymbol{D_2}(\boldsymbol{\mu}) = \boldsymbol{D_m}(\boldsymbol{\mu})$. We apply the robust Huber norm over the data term where:

$$\|\boldsymbol{x}\|_\epsilon = \begin{cases} \frac{\|\boldsymbol{x}\|_2^2}{2\epsilon} & \text{if } \|\boldsymbol{x}\|_2 \leq \epsilon, \\ |\boldsymbol{x}|_1 - \frac{\epsilon}{2} & \text{otherwise} \end{cases} \tag{37}$$

The L_2^2 norm works for small values, which is appropriate for handling Gaussian noise, getting smooth regions, while the L_1 norm allows discontinuities at depth edges (larger errors). Missing data in depth maps (undefined pixels) defines the inpainting domain:

$$w_k(\boldsymbol{\mu}) \in \{0, 1\}, \tag{38}$$

where $w(\boldsymbol{\mu}) = 0$ corresponds to pure inpainting at location $\boldsymbol{\mu}$. In consequence, the regularizer term allows smooth solutions and the data term allows solutions similar to the depth sources. The energy defined in eq. (36) is minimized using a first-order primal-dual algorithm.

3.6.1 Primal-Dual Algorithm

The regulariser and the data term of eq. (36) can be written in a more general form:

$$\min_{\boldsymbol{y}} F(A\boldsymbol{y}) + \sum_{k=1}^{2} G_k(\boldsymbol{y}), \tag{39}$$

In our case, $A = \nabla$ the gradient operator, $F(A\boldsymbol{y}) = \|A\boldsymbol{y}\|_1$, $G_k(\boldsymbol{y}) = \lambda\boldsymbol{\varpi_k}\|\boldsymbol{y} - \boldsymbol{\varphi_k}\|_\epsilon$, $\boldsymbol{y}$, $\boldsymbol{\varphi_k}$ and $\boldsymbol{\varpi_k}$ are row-wise vector versions of the sought solution $\boldsymbol{D_f}(\boldsymbol{\mu})$, the depth sources $\boldsymbol{D_k}(\boldsymbol{\mu})$, and the matrix w_k that defines the inpainting domain, respectively. We do Legendre-Fenchel transformations in this way:

$$F(A\boldsymbol{y}) = \max_{\|\boldsymbol{\varrho}\|_2 \leq 1} \langle A\boldsymbol{y}, \boldsymbol{\varrho}\rangle - F^*(\boldsymbol{\varrho}), \tag{40}$$

$$G_k(\boldsymbol{y}) = \max_{|\boldsymbol{r_k}|_1 \leq \lambda\boldsymbol{\varpi_k}} \langle \boldsymbol{y} - \boldsymbol{\varphi_k}, \boldsymbol{r_k}\rangle - {G_k}^*(\boldsymbol{r_k}), \tag{41}$$

where $\boldsymbol{\varrho}$ and $\boldsymbol{r_k}$ are dual variables associated to the primal variables $\boldsymbol{y}$ and $\boldsymbol{\varphi_k}$ respectively, and $\langle\cdot,\cdot\rangle$ corresponds to the dot product. Replacing (40) and (41) in (39) we get the primal-dual formulation of this nonlinear problem. It is a generic saddle-point problem:

$$\begin{aligned}\min_{\boldsymbol{y}} \max_{|\boldsymbol{r_k}|_1 \leq \lambda\boldsymbol{\varpi_k}, \|\boldsymbol{\varrho}\|_2 \leq 1} &\langle A\boldsymbol{y}, \boldsymbol{\varrho}\rangle - F^*(\boldsymbol{\varrho}) + \\ &\sum_{k=1}^{2}[\langle \boldsymbol{y} - \boldsymbol{\varphi_k}, \boldsymbol{r_k}\rangle - {G_k}^*(\boldsymbol{r_k})],\end{aligned} \tag{42}$$

where $F^*(\boldsymbol{\varrho})$ and ${G_k}^*(\boldsymbol{r_k})$ are the convex conjugates of $F(A\boldsymbol{y})$ and $G_k(\boldsymbol{y})$, respectively, and are defined as:

$$F^*(\boldsymbol{\varrho}) = \delta_{\boldsymbol{\varrho}}(\boldsymbol{\varrho}), \tag{43}$$

$${G_k}^*(\boldsymbol{r_k}) = \delta_{\boldsymbol{r_k}}(\boldsymbol{r_k}) + \frac{\epsilon}{2}\|\boldsymbol{r_k}\|_2^2, \tag{44}$$

where $\delta_{\boldsymbol{\varrho}}$ and $\delta_{\boldsymbol{r_k}}$ are indicator functions of the convex sets, defined as:

$$\delta_{\boldsymbol{\varrho}}(\boldsymbol{\varrho}) = \begin{cases} 0 & \text{if } \|\boldsymbol{\varrho}\|_1 \leq 1, \\ \infty & \text{otherwise,} \end{cases} \tag{45}$$

$$\delta_{\boldsymbol{r_k}}(\boldsymbol{r_k}) = \begin{cases} 0 & \text{if } |\boldsymbol{r_k}|_1 \leq \lambda\boldsymbol{\varpi_k}, \\ \infty & \text{otherwise.} \end{cases} \tag{46}$$

Following the algorithm 1 of [2] and the parameter setting of [17] we set the primal and dual time steps with $\tau = 0.05$ and $\sigma = 1/(8\tau)$, respectively. The Huber norm parameter $\epsilon = 0.1$ and the balance $\lambda = 1.2$. We set the initial primal variable as $\bar{\boldsymbol{y}}_0 = \boldsymbol{\varphi_s}$ since it is the most informative depth source. The dual variables $\boldsymbol{r_k}$ and $\boldsymbol{\varrho}$ are initialized with zeros.

Based on [2], the iterative optimization corresponds to perform in alternating way gradient ascent over the dual variables and gradient descent over the primal variable, projecting the results onto the constraints and updating the primal variable, as is summarized next:

$$\begin{cases} \boldsymbol{\varrho}^{n+1} = \text{proj}_{\boldsymbol{\varrho}}(\boldsymbol{\varrho}^n + \sigma A\bar{\boldsymbol{y}}^n), \\ \boldsymbol{r_k}^{n+1} = \text{proj}_{\boldsymbol{r_k}}\left(\frac{\boldsymbol{r_k}^n + \sigma(\bar{\boldsymbol{y}}^n - \boldsymbol{\varphi_k})}{1+\sigma\epsilon}\right) \quad k = 1, 2, \\ \boldsymbol{y}^{n+1} = \boldsymbol{y}^n - \tau(A^*\boldsymbol{\varrho}^{n+1} + \sum_{k=1}^{2}\boldsymbol{r_k}^{n+1}), \\ \bar{\boldsymbol{y}}^{n+1} = \boldsymbol{y}^{n+1} + \Phi(\boldsymbol{y}^{n+1} - \boldsymbol{y}^n), \end{cases} \tag{47}$$

where A^* is the adjoint operator of the gradient operator and corresponds to the negative divergence operator, $\Phi = 1$, $\text{proj}_{\boldsymbol{\varrho}}$ and $\text{proj}_{\boldsymbol{r_k}}$ are projections of the dual variables $\boldsymbol{\varrho}$ and $\boldsymbol{r_k}$, respectively, onto convex sets. They are defined for each element of the vectors as:

$$\text{proj}_{\boldsymbol{\varrho}}(\tilde{\boldsymbol{\varrho}}) = \frac{\tilde{\boldsymbol{\varrho}}}{\max(1, |\tilde{\boldsymbol{\varrho}}|)}, \tag{48}$$

$$\text{proj}_{\boldsymbol{r_k}}(\tilde{\boldsymbol{r}}_{\boldsymbol{k}}) = \begin{cases} \tilde{\boldsymbol{r}}_{\boldsymbol{k}} & \text{if } |\tilde{\boldsymbol{r}}_{\boldsymbol{k}}| < \lambda\boldsymbol{\varpi_k}, \\ \lambda\boldsymbol{\varpi_k} & \text{if } \tilde{\boldsymbol{r}}_{\boldsymbol{k}} > \lambda\boldsymbol{\varpi_k}, \\ -\lambda\boldsymbol{\varpi_k} & \text{if } \tilde{\boldsymbol{r}}_{\boldsymbol{k}} < -\lambda\boldsymbol{\varpi_k}. \end{cases} \tag{49}$$

4 Results

For testing our optimization algorithms when one of the three variables is unknown (shape, pose and scale) we use a down-sampled point cloud coming from the vertices of the triangles computed with the marching cubes algorithm for the compressed version of the base model (fig. 2(h)).

When shape, pose and scale are unknown, we test our algorithms with data coming from the kinect 1.0 and with synthetic data got from a 3D scene. For these experiments we also carry out

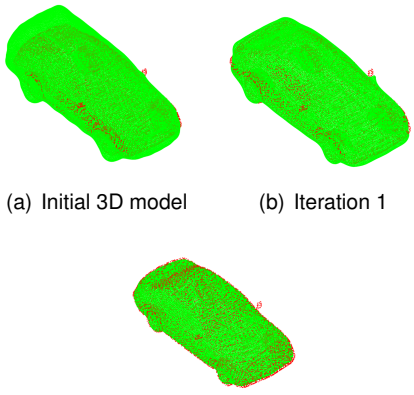
depth integration using our proposed variational formulation and quantify the completness and accuracy of the enhanced depth maps.

4.1 Results in Shape Optimization

The goal is to refine the shape associated to an initial latent variable in order to reduce the residual computed by evaluating the current embedding function in the point cloud (eq. (10)). In this experiment, the pose and scale are considered known. Figure 5(a) shows an explicit representation of the model for the initial latent variable.

Figures 5(b) and 5(c) show the model for iterations 1 and 8, respectively. Figure 6 shows the path of the iterative search in the latent space. It starts with a latent variable $\boldsymbol{\chi_p} = [-0.2723 \quad 2.3048]$ and in iteration 15 $\boldsymbol{\chi_p}$ has evolved to $[0.500114 \quad 1.63515]$, with a ground truth of $\boldsymbol{\chi_p} = [0.3895 \quad 1.6162]$. Note that after iteration 8 ($\boldsymbol{\chi_p} = [0.582774 \quad 1.62765]$) the energy diminishes slowly.

The Euclidean distance between the estimated latent variable and the ground truth decreased from 0.9551 in the initialization to 0.1122 in iteration 15, getting a reduction in error of 88.25%.



(a) Initial 3D model (b) Iteration 1

(c) Iteration 8

Fig. 5. Evolution of the shape in the optimization process. The search is done in the latent space and the model is obtained with the inverse DCT. It takes 15 iterations for converging. The model is drawn in green and the point cloud in red

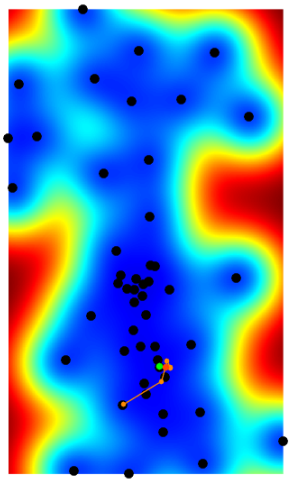


Fig. 6. Latent Space with the 49 latent variables (in black) used for learning the mapping parameters. Low variance regions are drawn in blue while high variance regions are drawn in red. More likely valid shapes are obtained in blue areas. The green point is the ground truth in latent variable. The search path of the latent variable that best fits to data is drawn in orange. The resulting latent variable is drawn in dark orange

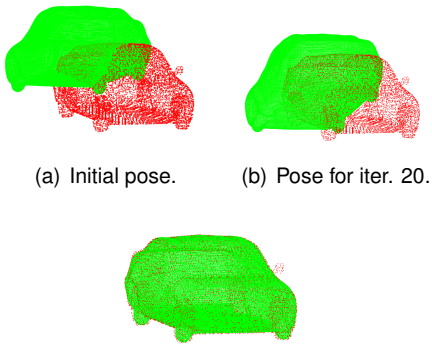
4.2 Results in Pose Optimization

In this experiment the shape and scale are considered known and an initial pose is refined in order to align the embedding function with the point cloud. The ground truth in position is the origin and in orientation is the identity matrix. The initial position is $\boldsymbol{t_{wo}} = [-0.2 \quad 0.2 \quad 0.2]$ with an Euclidean distance to the origin of 0.3464. The initial orientation corresponds to a rotation of $\alpha_z = -30$ around z-axis w.r.t the ground truth.

Figure 7(a) shows the initial pose, while fig. 7(b) and 7(c) show the pose for iterations 20 and 50, respectively. The evolution of the Euclidean distance from the model to the origin and the evolution of the model orientation w.r.t. the ground truth are presented in fig. 8(a) and 8(b), respectively. The orientation R_{wo} is represented with rotations over the fixed axis x, y, and z (in this order).

The final position is $\boldsymbol{t_{wo}} = [0.0032 \quad -0.0104 \quad 0.0035]$, with an Euclidean distance to the origin of 0.0114. The final orientation is $\alpha_x =$

$-0.0017°$, $\alpha_y = -0.0663°$, and $\alpha_z = -0.0209°$. It converges at iteration 59, achieving a reduction in position error of 96.70% and in rotation around z-axis of 99.93%.



(a) Initial pose. (b) Pose for iter. 20.

(c) Pose for iter. 50.

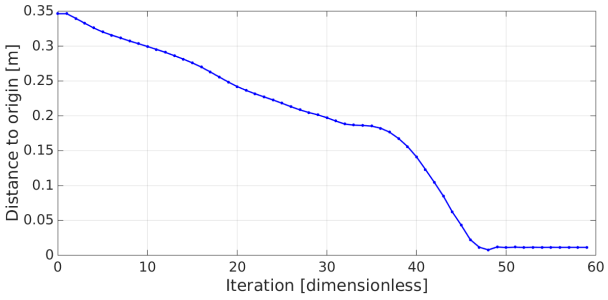
Fig. 7. Evolution of the pose of the embedding function in the optimization process. The model is drawn in green and the point cloud in red

4.3 Results in Scale Optimization

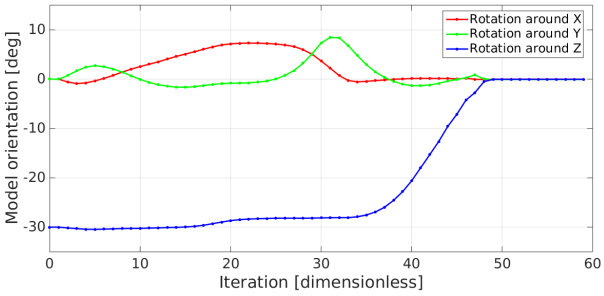
For scale optimization, the shape and pose are considered known. Recall that the same scale is used in the three dimensions. The initial value in scale is $s = 0.4$. The ground truth in scale is $s = 1.0$. Figure 9(a) shows the model with the initial scale ($s = 0.4$), meanwhile figs. 9(b) and 9(c) show the model with the scale of iterations 1 ($s = 1.5855$) and 10 ($s = 1.0053$), respectively. The error for iteration 10 is 0.53% of the ground truth. The reduction in error is 99.12%. Figure 10 shows how the scale gets close to the ground truth ($s = 1.0$).

4.4 Results in Pose, Scale and Shape, with Real Data

Now, we use a point cloud coming from a depth map taken with the Kinect V1 of Microsoft. The Kinect was moved by hand in front of a toy car while VGA images were captured at 30fps. We selected one depth and intensity image (see figs. 14(a) and 14(b)) and manually segmented the toy car, getting the mask of fig. 11(a). The resulting 3D points



(a) Distance from model to origin



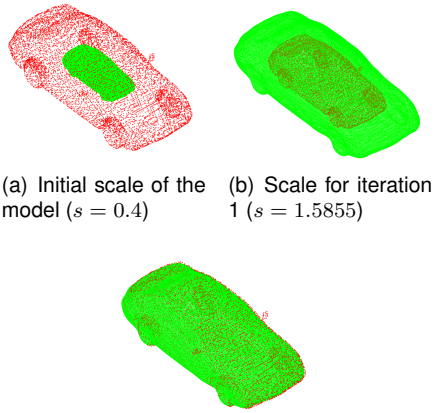
(b) Model Orientation w.r.t world coordinate system.

Fig. 8. Evolution of the position and orientation with respect to the ground truth. It takes 59 iterations for converging

X_r associated to the segmented car (red points in fig. 11(b)) are referenced to the camera coordinate system, where the z-axis is perpendicular to the image plane.

Then, we compute 3D points referenced to the world coordinate system, $X_w = M_{wr}X_r$, where M_{wr} is a rotation of $90°$ around the y axis followed by a rotation of $-90°$ around the new z axis. As was doing for model alignment in section 3.1, the point cloud of the toy car referenced to the world system X_w is translated from its center of gravity to the origin $X_w^c = X_w - \bar{X}_w$, and the average distance of each point to the new origin s_{pc} is computed using eq. (1). The same process is carried out for the vertices of the 3D model for getting s_{m}. The initial scale is computed with eq. (2).

The initial position of the car is defined as the center of gravity of the point cloud referenced to the world coordinate system, adding 4% of the x-component (z-axis in the camera coordinate



(a) Initial scale of the model ($s = 0.4$)

(b) Scale for iteration 1 ($s = 1.5855$)

(c) Scale for iteration 10 ($s = 1.0053$)

Fig. 9. Evolution of the scale of the embedding function in the optimization process. The model is drawn in green and the point cloud in red

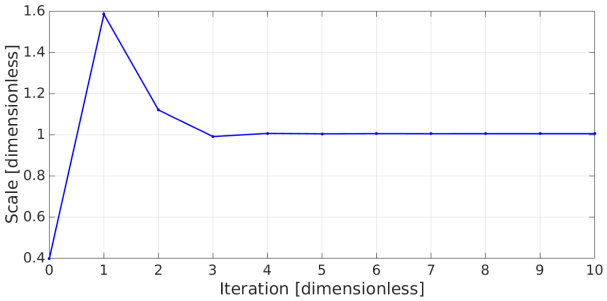


Fig. 10. Evolution of the scale. It takes 10 iterations for converging

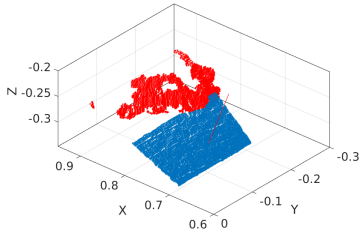
system) to itself since the data belongs just to a side of the whole car:

$$\boldsymbol{t_{wo}} = [1.04\bar{X}_{w_x} \quad \bar{X}_{w_y} \quad \bar{X}_{w_z}]'. \tag{50}$$

As in [6], we assume that the toy car is over a flat surface, so computing the normal of this surface allows us to estimate an initial orientation of the car. This process begins by calculating 3D points for pixels in a region bellow the segmented area which belong to the supporting plane. Then, RANSAC is applied: three points are randomly selected and the parameters that define a plane are computed.

The points that fulfill with the computed plane (under a threshold) are counted. The random

(a) Mask of segmented car.



(b) Point cloud of segmented region (in red), of the supporting plane (in blue) and its normal (red line).

Fig. 11. a) Mask of the segmented car. b) Point cloud associated to the segmented region (lateral side of the toy car)

selection process was repeated 100 times and we chose the parameters that generated the largest number of inliers (points under a threshold when the plane is evaluated).

The resulting normal $\boldsymbol{n}$, shown in figure 11(b), is used for computing two angles α and β that moves the embedding function to the supporting plane:

$$\alpha = acos\left(\frac{n_z}{\sqrt{n_x^2 + n_z^2}}\right), \quad \beta = acos(\sqrt{n_x^2 + n_z^2}). \tag{51}$$

The transformation consists of a rotation of $-\alpha$ over the y-axis followed by a rotation of β over the new x axis. The third angle γ needed for totally defining the orientation of the embedding function is the rotation around the unit normal (new z axis). This angle is found through exhaustive search with α between $10°$ and $360°$ and with an increment of $10°$; the selected α is the one that produces the minimum energy when eq. (10) is evaluated. Finally, the initial transformation matrix M_{wo} is:

$$M_{wo} = \begin{bmatrix} R_{wo} & \boldsymbol{t_{wo}} \\ \mathbf{0_{1\times 3}} & 1 \end{bmatrix}, \tag{52}$$

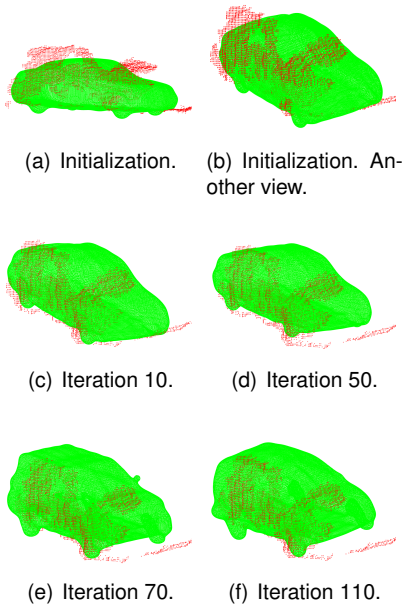
with $R_{wo} = \text{Rot}(Y, -\alpha)\text{Rot}(X', \beta)\text{Rot}(Z'', \gamma)$. Summarizing, the initial position is $\boldsymbol{t_{wo}} = [0.8459 \quad -0.1524 \quad -0.2598]$, the initial orientation corresponds to $\alpha = 20.0017°$, $\beta = 2.18°$ and $\gamma = -20°$.

For comparing this orientation with the final one, we use rotations over fixed axis, x, y and z (in this order), with $\alpha_x = 9.1136°$, $\alpha_y = -18.0095°$, and $\alpha_z = -21.0621°$. The initial scale is $s = 0.1571$, and the initial shape is the one associated to the reference model employed in the model alignment process $\boldsymbol{\chi} = [0.3895 \quad 1.6162]$. Figures 12(a) and 12(b) show the initial conditions for the model.

For this test, we carry out two cycles with the sequence: 20 iterations for pose and 5 iterations for scale. At the end of this sequence, 50 iterations have been done and very close pose and scale estimations are obtained (see 12(d)). With these estimations we can perform exhaustive search over the $N_m = 49$ models of cars used for learning the latent space. The latent variable with the 3D level set that produces the minimum energy $\boldsymbol{\chi} = [0.4560 \quad 2.4675]$ is used as initial value in the following refinement process. Finally, we carry out three cycles with the sequence: 5 iterations for shape, 10 iterations for pose and 5 iterations for scale, getting a refinement in pose and scale for a more approximated shape (see fig. 12(f)).

The final scale is $s = 0.1602$, and the final latent variable is $\boldsymbol{\chi} = [0.5858 \quad 2.6147]$. The final position is $\boldsymbol{t_{wo}} = [0.8499 \quad -0.1377 \quad -0.2530]$ and the final orientation is $\alpha_x = 5.7633°$, $\alpha_y = -9.0734°$ and $\alpha_z = -26.0871°$.

Figure 13 shows the evolution of the energy, drawing with red, green, and blue the pose, scale and shape optimization, respectively. The synthetic depth map got by reading the depth-buffer of OpenGL from the current camera pose is presented in fig. 14(c).



(a) Initialization. (b) Initialization. Another view.

(c) Iteration 10. (d) Iteration 50.

(e) Iteration 70. (f) Iteration 110.

Fig. 12. Evolution of the pose, scale, and shape of the embedding function in the alternating optimization process, for real data using the Kinect. The point cloud is drawn in red

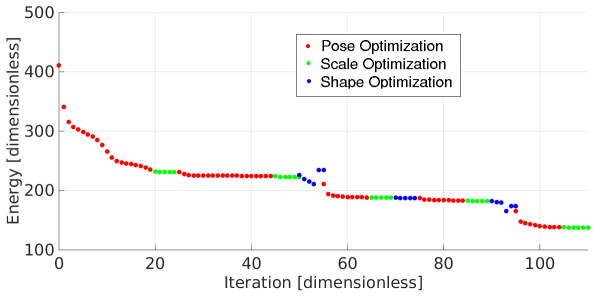
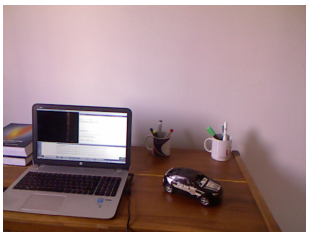


Fig. 13. Evolution of the energy for pose, scale, and shape optimization, in alternating way, using Levenberg Marquardt and the Kinect V1

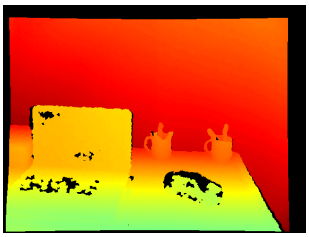
4.4.1 Results in Inpainting and Denoising

Since the main goal is to reduce noise and complete missing data in the depth map, especially in the region of the car, we get depth data of the optimal 3D model seen from the real camera pose. Figure 14(b) shows the depth map coming from the sensor while fig. 14(c) shows the depth map coming from the optimal 3D model. Note that the former image has missing data due to possible dark, very reflective and translucent objects, out of range objects or occlusions. The last image has defined data just for the 3D model. $w^s(\mu)$ and $w^m(\mu)$ defines the inpainting domain, where 1 represents defined data and 0 missing data (see eq. (38)).

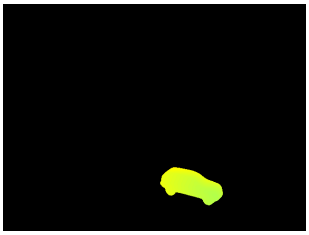
Figure 14(d) shows the sought solution got with the optimization process for iteration 200. Note that the filled areas are smooth and undetectable, achieving a real appearance. The inpainting in the top and right side of the depth map is not so good since there is not data neither in D_s nor D_m and the missing data covers large areas.



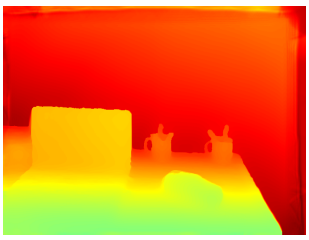
(a) RGB image.



(b) Depth image $D_s(\mu)$.



(c) Depth map $D_m(\mu)$ coming from the model.



(d) $D_f(\mu)$ for iteration 200.

Fig. 14. Sought depth map $D_f(\mu)$ for iteration (a) 10, (b) 40, (c) 100 and (e) 200 of the primal-dual algorithm

We analyzed the performance of the inpainting and denoising algorithm using a slice around the x-axis (see fig. 15). Note that missing data stays mainly in the image borders, in the laptop region and in the car region (inside the vertical lines in fig. 15). It is 22.81% of the total amount of data in the slice.

The solution remains similar to the sensor data but it fills missing data, integrates depth data coming from the model, is smooth but preserves discontinuities. Evaluating the completeness of the car, we can state that the algorithm fills 33.28% of missing data. This estimation was done for pixels belonging to the segmented area of the whole car.

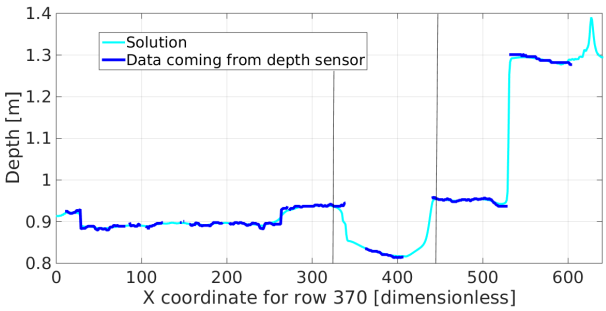
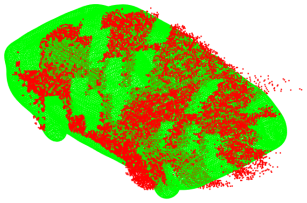


Fig. 15. Depth map denoising and inpainting using shape priors and variational techniques along a slice. The vertical lines encompass the car

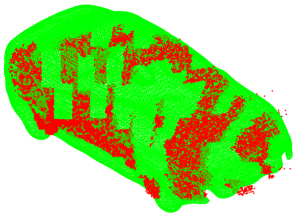
4.5 Results in Pose, Scale and Shape with Synthetic Data

For this experiment we have created in Blender a 3D scene composed of a floor, a desk, a car, a mug and a lamp. A depth map (see fig. 17(f)) and a rgb image have been rendered from the virtual camera. The depth data is in the range $[1.2045 \quad 8.9610]m$. We added Gaussian white noise with SNR of 5dB to the rgb image (see fig. 17(a)) and 30dB to the depth map. Moreover, we simulated missing data in the depth maps with rectangles of 40×20 pixels containing nans, randomly placed in the depth maps. The percentages of missing data that we used were 23.65% and 40.02% (see figs. 17(b) and 17(g), respectively).

The process for estimating the optimal shape prior (initial estimation and the iterative refinement)



(a) Aligned 3D model with the point cloud for missing data of 23.65%.



(b) Aligned 3D model with the point cloud for missing data of 40.02%.

Fig. 16. Results of the aligment process between a 3D-level set and the point cloud. a) For depth map with missing data of 23.65% and b) For depth map with missing data of 40.02%

and for getting a depth map of this model is similar to the one carried out previously for real data. Figures 16(a) and 16(b) show the aligned models with the point clouds for 23.65% and 40.02% of missing data, respectively.

Our algorithm performs 500 iterations for the depth map with 23.65% of missing data and 700 iterations for the one with 40.02%. We compare the results with the ones obtained with the outstanding system of Pertuz and Kamarainen [28], employing their code publicly available.

Table 1. Comparison of RMSE[m] of our algorithm and [28]

missing data - algorithm	Ours	[28]
23.65%	*0.1498*	*0.1600*
40.02%	0.1869	0.2188

It uses anisotropic diffusion and a region-based approach for inpainting sparse depth maps. Table 1 shows that our algorithm outperforms [28] for both percentages of missing data. In fig 17 we can see that the inpainted and denoised depth map got with our algorithm is smoother and preserves discontinuities in a better way than [28]. We did not use more percentage of missing data since it affects the alignment of the model, producing incorrect estimations of the optimal model.

4.5.1 Time

Our system for inpainting and denoising runs on a laptop Hewlett Packard with a processor Intel Core i7-2.2GHz, 11.7GB of RAM memory, a graphic processor NVIDIA GEFORCE 755M and Ubuntu 14.04 as operating system. OpenCV is used for processing images, Eigen3 for matrix operations, OpenGL for drawing the 3D model and getting depth information from it, and Cuda 7.5 for speeding up the algorithm for merging the depth data coming from the sensor and from the optimal model.

Table 2. Processing time for inpainting, denoising and merging

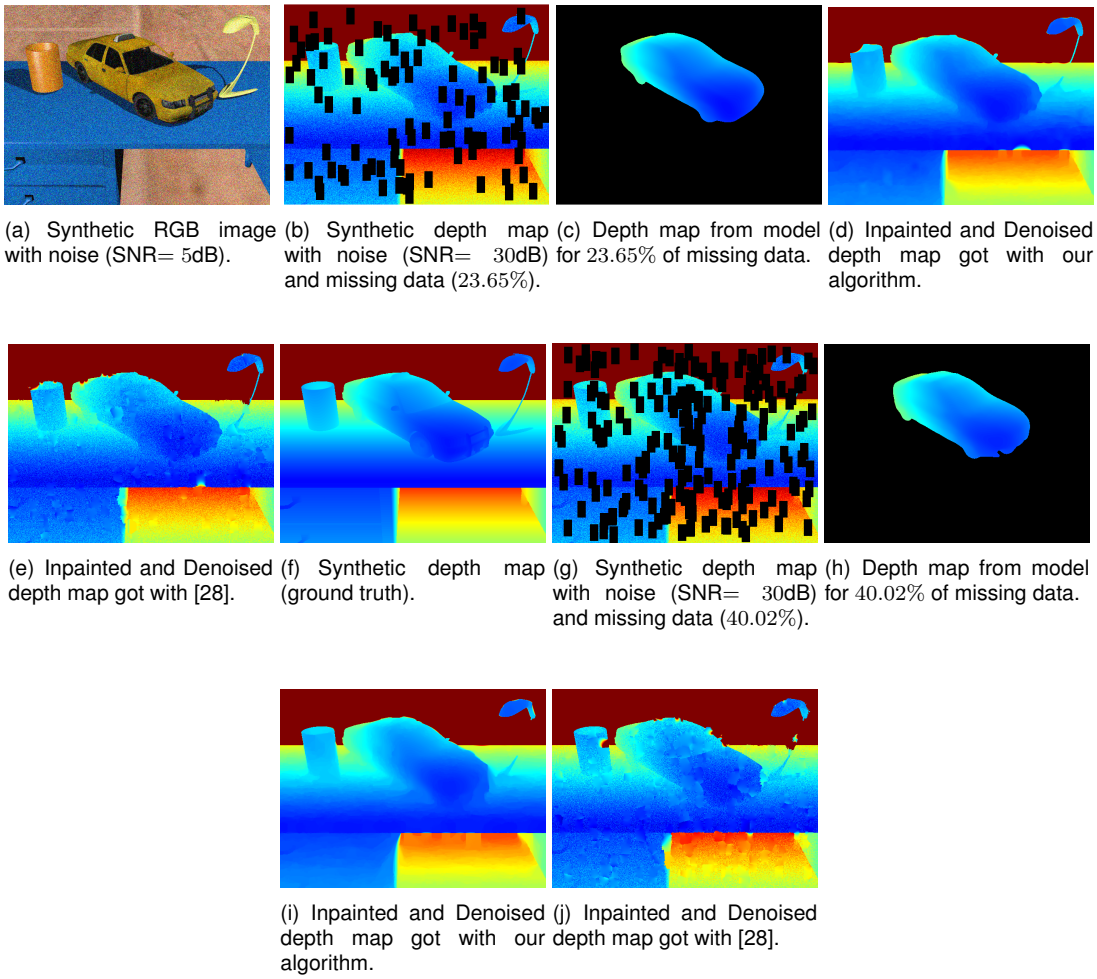
Process	Time[ms]
First line of eq. (47)	6.501
Second line of eq. (47)	5.373
Third line of eq. (47)	3.247
Remaining processes	0.573
TOTAL ITERATION	15.694

The algorithm for estimating the optimal model was implemented in Matlab. Table 2 shows the average on processing time for the main processes. The implementation on GPU of the algorithm for estimating the optimal model is left for future work.

The algorithm takes 7.847s (500 iterations) for the depth map with 23.65% of missing data and 10.96s (700 iterations) for the depth map with 40.02% of missing data.

5 Conclusion

We have developed a system that optimizes a 3D model, represented as a 3D level-set embedding function, w.r.t. pose, scale and shape and uses



(a) Synthetic RGB image with noise (SNR= 5dB).
(b) Synthetic depth map with noise (SNR= 30dB) and missing data (23.65%).
(c) Depth map from model for 23.65% of missing data.
(d) Inpainted and Denoised depth map got with our algorithm.
(e) Inpainted and Denoised depth map got with [28].
(f) Synthetic depth map (ground truth).
(g) Synthetic depth map with noise (SNR= 30dB) and missing data (40.02%).
(h) Depth map from model for 40.02% of missing data.
(i) Inpainted and Denoised depth map got with our algorithm.
(j) Inpainted and Denoised depth map got with [28].

Fig. 17. The first figure in the first and second rows are the rgb image with noise and the ground truth in depth. For the remaining figures the first row contains images for the depth map with missing data of 23.65% while the second row for the depth map with missind data of 40.02%. From left to right: depth map with noise and missing data, depth map coming from the optimal model, our results, results using [28]

depth data of the optimal model for depth denoising and inpainting of a raw depth map coming from a depth sensor, achieving significant results in completeness, specially in the area associated to the object of interest that has a high percentage of missing data. The results have been satisfactory.

First, the embedding function was successfully compressed from 128^3 to 25^3 DCT coefficients (compression ratio of $134,22$) and reduced from 25^3 to 2 dimensions (bi-dimensional latent space) using GPLVM, for a more efficient search of the optimal model.

Second, the optimization was outstanding: the shape optimization converge in 15 iterations with a reduction in error of 88.25% w.r.t. the initial estimation; the pose optimization converge in 59 iterations achieving a final error in position of 1.14cm and less that one tenth of degree in rotation around each axis (reduction in position error of 96.70% and in rotation error around z-axis of

99.93%) validating our proposed technique based on Lie algebra; the scale optimization converge in 10 iterations, getting a final error of 0.53% of the ground truth (reduction in scale error of 99.12%).

Third, the inpainting and denoising process using variational techniques achieved a significant increase in completeness of the depth map coming from the Kinect 1.0, filling 32.28% of missing data of the segmented region of the car, smoothing the data but preserving discontinuities. Moreover, the accuracy of the enhanced depth maps got from synthetic depth maps with 23.65% and 40.02% of missing data is higher than the one obtained with the outstanding algorithm for depth recovery [28].

For future work we will use shape priors and depth integration with variational techniques for enhancing the quality of depth maps built with a monocular camera and we will fuse them into a volumetric structure for getting an improved dense reconstruction.

Acknowledgment

The authors would like to thank Fundación CEIBA for the financial support that has made the development of this work possible.

References

1. **Bao, S. Y., Chandraker, M., Lin, Y., & Savarese, S. (2013).** Dense object reconstruction with semantic priors. *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, IEEE Computer Society, Washington, DC, USA, pp. 1264–1271.
2. **Chambolle, A. & Pock, T. (2011).** A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, Vol. 1, No. 40, pp. 120–145.
3. **Chaouch, M. & Verroust-Blondet, A. (2008).** A novel method for alignment of 3d models. *2008 IEEE International Conference on Shape Modeling and Applications*, pp. 187–195.
4. **Chen, C., Jafari, R., & Kehtarnavaz, N. (2015).** Improving human action recognition using fusion of depth camera and inertial sensors. *IEEE Transactions on Human-Machine Systems*, Vol. 45, No. 1, pp. 51–61.
5. **Concha, A., Hussain, W., Montano, L., & Civera, J. (2015).** Incorporating scene priors to dense monocular mapping. *Autonomous Robots*, Vol. 39, No. 3, pp. 279–292.
6. **Dame, A., Prisacariu, V. A., Ren, C. Y., & Reid, I. (2013).** Dense reconstruction using 3d object shape priors. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1288–1295.
7. **Drozdov, G., Shapiro, Y., & Gilboa, G. (2016).** Robust recovery of heavily degraded depth measurements. *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 56–65.
8. **Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010).** Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 9, pp. 1627–1645.
9. **Feng, J., Price, B., Cohen, S., & Chang, S. F. (2016).** Interactive segmentation on rgbd images via cue selection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 156–164.
10. **Ferstl, D., Reinbacher, C., Ranftl, R., Ruether, M., & Bischof, H. (2013).** Image guided depth upsampling using anisotropic total generalized variation. *2013 IEEE International Conference on Computer Vision*, pp. 993–1000.
11. **Fu, H., Xu, D., & Lin, S. (2017).** Object-based multiple foreground segmentation in rgbd video. *IEEE Transactions on Image Processing*, Vol. 26, No. 3, pp. 1418–1427.
12. **Hornácek, M., Rhemann, C., Gelautz, M., & Rother, C. (2013).** Depth super resolution by rigid body self-similarity in 3d. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1123–1130.
13. **Jung, S. W. (2013).** Enhancement of image and depth map using adaptive joint trilateral filter. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 23, No. 2, pp. 258–269.
14. **Kerl, C., Sturm, J., & Cremers, D. (2013).** Robust odometry estimation for rgb-d cameras. *2013 IEEE International Conference on Robotics and Automation*, pp. 3748–3754.
15. **Khayam, S. A. (2003).** The discrete cosine transform (dct): Theory and application". department of electrical computing engineering.
16. **Klein, G. & Murray, D. W. (2007).** Parallel tracking and mapping for small AR workspaces. *Proc. Sixth*

IEEE and ACM International Symposium on Mixed and Augmented Reality.

17. **Kluckner, S., Pock, T., & Bischof, H. (2010).** Exploiting redundancy for aerial image fusion using convex optimization. In **Goesele, M., Roth, S., Kuijper, A., Schiele, B., & Schindler, K.**, editors, *Pattern Recognition: 32nd DAGM Symposium*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 303–312.
18. **Krenzin, J. & Hellwich, O. (2016).** Reduction of point cloud artifacts using shape priors estimated with the gaussian process latent variable model. In **Rosenhahn, B. & Andres, B.**, editors, *Pattern Recognition: 38th German Conference, GCPR 2016, Hannover, Germany, September 12-15, 2016, Proceedings*. Springer International Publishing, Cham, pp. 273–284.
19. **Kwon, H., Tai, Y.-W., & Lin, S. (2015).** Data-driven depth map refinement via multi-scale sparse representation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 159–167.
20. **Lawrence, N. (2005).** Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Mach. Learn. Res.*, Vol. 6, pp. 1783–1816.
21. **Lin, D., Fidler, S., & Urtasun, R. (2013).** Holistic scene understanding for 3d object detection with rgbd cameras. *2013 IEEE International Conference on Computer Vision*, pp. 1417–1424.
22. **Liu, M. Y., Tuzel, O., & Taguchi, Y. (2013).** Joint geodesic upsampling of depth images. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 169–176.
23. **Liu, S., Wang, Y., Wang, H., & Pan, C. (2013).** Kinect depth inpainting via graph laplacian with tv21 regularization. *2013 2nd IAPR Asian Conference on Pattern Recognition*, pp. 251–255.
24. **Møller, M. F. (1993).** A scaled conjugate gradient algorithm for fast supervised learning. *NEURAL NETWORKS*, Vol. 6, No. 4, pp. 525–533.
25. **Newcombe, R. A., Lovegrove, S. J., & Davison, A. J. (2012).** DTAM: Dense tracking and mapping in real-time.
26. **Ni, B., Moulin, P., Yang, X., & Yan, S. (2015).** Motion part regularization: Improving action recognition via trajectory group selection. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3698–3706.
27. **Park, J., Kim, H., Tai, Y.-W., Brown, M. S., & Kweon, I. (2011).** High quality depth map upsampling for 3d-tof cameras. *2011 International Conference on Computer Vision*, pp. 1623–1630.
28. **Pertuz, S. & Kamarainen, J. (2017).** *Region-based Depth Recovery for Highly Sparse Depth Maps*.
29. **Piniés, P., Paz, L. M., & Newman, P. (2015).** Dense Mono Reconstruction: Living with the Pain of the Plain Plane. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA.
30. **Ren, C. Y. & Reid, I. (2012).** A unified energy minimization framework for model fitting in depth. *Proceedings of the 12th International Conference on Computer Vision - Volume 2*, ECCV'12, Springer-Verlag, Berlin, Heidelberg, pp. 72–82.
31. **Rohr, K., Stiehl, H. S., Sprengel, R., Beil, W., Buzug, T. M., Weese, J., & Kuhn, M. H. (1996).** Point-based elastic registration of medical image data using approximating thin-plate splines. In **Höhne, K. H. & Kikinis, R.**, editors, *Visualization in Biomedical Computing: 4th International Conference, VBC'96 Hamburg, Germamy, September 22–25, 1996 Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 297–306.
32. **Teng, Y., Zhang, Y., Chen, Y., & Ti, C. (2016).** Adaptive morphological filtering method for structural fusion restoration of hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, Vol. 9, No. 2, pp. 655–667.
33. **Ti, C., Xu, G., Guan, Y., & Teng, Y. (2017).** Depth recovery for kinect sensor using contour-guided adaptive morphology filter. *IEEE Sensors Journal*, Vol. 17, No. 14, pp. 4534–4543.
34. **Tosic, I. & Drewes, S. (2014).** Learning joint intensity-depth sparse representations. *IEEE Transactions on Image Processing*, Vol. 23, No. 5, pp. 2122–2132.
35. **Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J. J., & Mcdonald, J. (2015).** Real-time large-scale dense rgb-d slam with volumetric fusion. *Int. J. Rob. Res.*, Vol. 34, No. 4-5, pp. 598–626.
36. **Zach, C., Pock, T., & Bischof, H. (2007).** A globally optimal algorithm for robust tv-l1 range image integration. *IEEE 11th International Conference on Computer Vision*, pp. 1–8.

37. Zhu, H., Weibel, J. B., & Lu, S. (2016). Discriminative multi-modal feature fusion for rgbd indoor scene recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2969–2976.

Article received on 02/09/2018; accepted on 11/11/2019.
Corresponding author is Andrés Díaz.